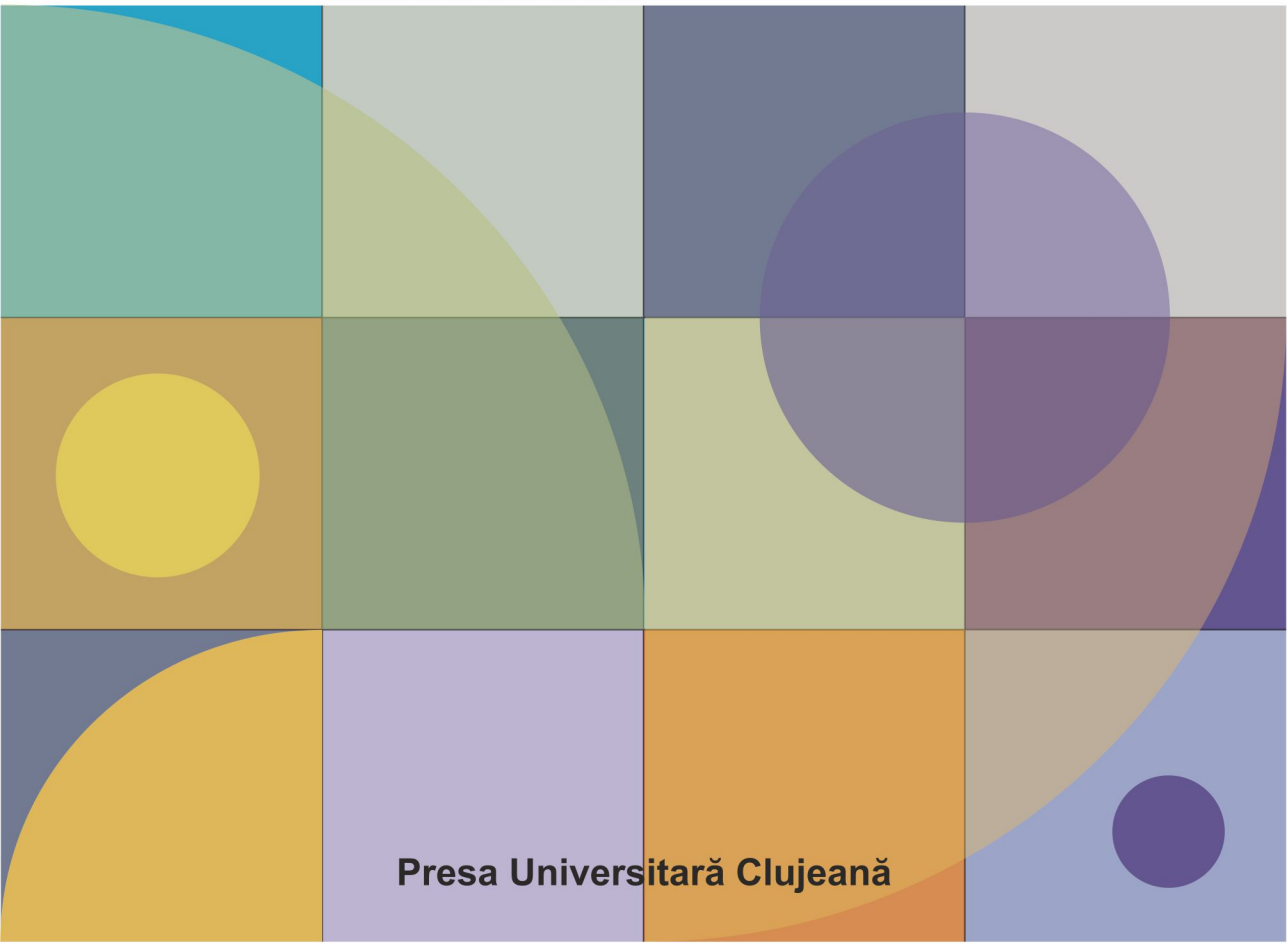
Mircea Comșa

Data mining pentru științele sociale

Volumul 2.
Modele de clasificare în RapidMiner Studio



Presa Universitară Clujeană

MIRCEA COMȘA

•

DATA MINING PENTRU ȘTIINȚELE SOCIALE

VOLUMUL II

Modele de clasificare în RapidMiner Studio

Volum finanțat de Universitatea Babeș-Bolyai
prin Fondul de Dezvoltare UBB 2023,
grant de tip seed (cod GS-UBB-SOCASIS-MIRCEACOMSA).

MIRCEA COMȘA

DATA MINING
PENTRU ȘTIINȚELE SOCIALE

VOLUMUL II

Modele de clasificare în RapidMiner Studio

PRESA UNIVERSITARĂ CLUJEANĂ

2024

Referenţi ştiinţifici:

Prof. univ. dr. Dan Chiribucă,
Universitatea „Babeș-Bolyai" din Cluj-Napoca

Prof. univ. dr. Bogdan Voicu,
Universitatea „Lucian Blaga" din Sibiu

http://www.editura.ubbcluj.ro/bd/ebooks/epub/Data-mining-pentru-stiintele-sociale-vol-2.epub

Notă: Lucrarea conține GIF-uri animate vizibile doar pe varianta epub, disponibilă pe pagina web a editurii.

ISBN general: 978-606-37-1496-2
ISBN specific vol. II: 978-606-37-2303-2

Universitatea Babeş-Bolyai
Presa Universitară Clujeană
Director: Codruţa Săcelean
Str. B.P. Hasdeu nr. 51
400371 Cluj-Napoca, România
Tel.: (+40)-744.687.884
E-mail: editura@ubbcluj.ro
http://www.editura.ubbcluj.ro/
https://libraria.ubbcluj.ro/

CUPRINS

LISTA TABELELOR

LISTA FIGURILOR

LISTA GRAFICELOR

LISTA EXEMPLELOR

1. INTRODUCERE

1.1. Structura și logica volumului

Volumul doi al manualului „Data mining pentru științele sociale” continuă demersul început în primul volum. În primul volum am pus oarecum bazele conceptuale ale unui proces de analiză de tip data mining și am ilustrat procesul de pregătire a datelor pentru analiză folosind softul RapidMiner Studio. Astfel, în primul volum am arătat ce presupune și cum se poate face curățarea datelor, transformarea datelor, reducerea datelor, tratarea valorilor lipsă, respectiv câteva analize simple (pașii 1-3 ai modelului CRISP-DM). În acest volum sunt prezentați și ilustrați următorii doi pași și anume: (4) analiza / modelarea datelor, realizarea modelului / modelelor și (5) evaluarea modelului / modelelor. Am preferat să lăsăm aspectele mai complexe ce țin de optimizarea și combinarea modelelor, respectiv implementarea și re-evaluarea modelelor pentru un volum ulterior. Tot cu scopul de a păstra volumul informațiilor într-o limită rezonabilă, în acest volum ne-am limitat la modelele de clasificare, respectiv la doar câțiva dintre algoritmii de clasificare.

Cum poate fi folosit acest volum?

Cu siguranță, înainte de a citi volumul doi este absolut necesară stăpânirea cunoștințelor teoretice și practice descrise în primul volum. Ulterior, procesul de învățare a modului în care se construiește, validează și evaluează un model de clasificare poate continua cu citirea atentă a acestui text, a exemplelor comentate, în paralel cu rularea analizelor, de preferat în ordinea prezentării lor (mai ales în cazul primelor patru capitole).

O atenție specială ar trebui acordată proceselor oferite ca exemple. Pentru o înțelegere mai bună a rolului fiecărui operator și mai ales a opțiunilor asociate, este necesar ca procesele să fie rulate prima dată în forma lor originală. Ulterior,

procesele pot fi modificate, prima dată la nivelul opțiunilor disponibile, adică la nivelul parametrilor și valorilor acestora, apoi se poate trece la modificări mai consistente (setul de date, includerea și a altor operatori etc.). Foarte important, după fiecare modificare, e util să observăm efectele acesteia.

După parcurgerea acestui ciclu de învățare se poate trece la adaptarea analizelor (modificarea proceselor) în vederea utilizării lor pentru realizarea propriilor proiecte de data mining. E necesar ca toate deciziile și comenzile să fie documentate (în sensul că trebuie să fie însoțite de note / comentarii care să precizeze clar ce face fiecare acțiune, procedură, metodă, model etc.; e de preferat ca adnotarea să fie realizată în interiorul proceselor, dar și în textul asociat analizei) astfel încât să fie asigurată reproductibilitatea analizelor. Pe scurt, dacă peste o perioadă de timp, cineva (cel care a realizat anterior analizele sau altcineva) dorește să verifice sau să reutilizeze procesele, ar trebui să poată face asta ușor, rapid și fără probleme. Se asigură astfel auditul activităților de analiză (dacă nu s-a schimbat nimic în datele și procesele folosite ar trebui să se obțină aceleași rezultate), se pune baza realizării mai eficiente a unor demersuri viitoare de analiză (noile proiecte vor fi realizate mult mai ușor), respectiv se produce un proces de învățare instituțională cumulativă (noile proiecte pot fi mult mai ușor îmbunătățite). Instituțiile vor deveni astfel mai performante, deciziile vor fi luate mai transparent, vor fi mai obiective, iar resursele disponibile vor fi alocate mai eficient.

Cui se adresează acest volum?

În mare măsură, acest manual este rezultatul activităților de predare susținute de autor pe parcursul ultimilor 10 ani în cadrul cursului „Metodologia analizei datelor. Tehnici de data mining”. Inițial, cursul a fost conceput pentru masteranzii Masteratului „Managementul Strategic al resurselor Umane” de la Facultatea de Sociologie și Asistență Socială, Universitatea „Babeș-Bolyai” din Cluj-Napoca. Ulterior, el a fost preluat ca opțional și la masteratele „Analiza Datelor Complexe” și „Cercetare Sociologică Avansată”. În consecință și prin extensie, manualul se adresează în principal studenților și masteranzilor din domeniul Științelor Sociale. Desigur, manualul poate fi util tuturor celor care doresc să învețe despre analiza de data mining într-o manieră mai puțin tehnică, folosind un soft extrem de intuitiv și în același timp profesionist. Funcție de tipul de licență disponibil, utilizatorii pot aplica rapid cele învățate în cadrul unor proiecte personale sau instituționale de anvergură diferită.

Temele discutate

Deși nu am menționat explicit acest lucru, volumul este organizat pe două linii majore. În prima parte a volumului discuția este axată pe conceptul de model de clasificare. Astfel, în capitolele 2-4 sunt prezentate informații și exemple despre construcția unui model de clasificare, respectiv validarea și evaluarea unui model de clasificare. Astfel de cunoștințe sunt absolut necesare indiferent care este algoritmul utilizat în cadrul unui model de clasificare. În capitolul 2 discutăm despre algoritmii de învățare automată (definiție, elemente componente, clasificarea algoritmilor, organizarea algoritmilor în RapidMiner, identificarea algoritmilor de învățare potriviți într-o situație specifică), prezentăm sintetic și comparativ algoritmii de clasificare discutați în volum, descriem logica generală de construire a unui model de clasificare și discutăm conceptul de eroare a unui model de clasificare (distorsiune și varianță, tipuri de modele în funcție de distorsiune și varianță, relația dintre eroarea și complexitatea modelului). În capitolul 3 discutăm conceptul de validare a unui model de clasificare (importanța validării, diferite tipuri de validare). Capitolul 4 descrie cum poate fi evaluat un model de clasificare, prilej cu care sunt prezentate și ilustrate conceptele de matrice de confuzie și măsuri ale performanței clasificării bazate pe aceasta, calcularea măsurilor de performanță în RapidMiner, vizualizarea performanței (curbele ROC și PR, măsurile sintetice AUROC și AUPRC, graficele de tip Gain și Lift).

În a doua parte sunt prezentați o serie de clasificatori (algoritmi de clasificare). Astfel, capitolele 5-9 prezintă următorii clasificatori: regresie logistică (Logistic Regression), bayes naiv (Naive Bayes), cei mai apropiați vecini (k Nearest Neighbor, k-NN), arbore decizional (Decision Tree) și rețea neuronală (Neural Net). Prezentarea și ilustrarea fiecărui clasificator urmează aproximativ aceeași structură: logica și pașii algoritmului, avantaje și dezavantaje, prezentarea și discutarea unui exemplu didactic, respectiv a două exemple de modelare în RapidMiner, unul mai simplu și altul relativ mai complex. În capitolul final prezentăm și discutăm modelarea automată a datelor cu ajutorul RapidMiner (Auto Model).

Resursele disponibile

Similar cu volumul 1, includem textul este însoțit de datele (seturi și baze de date, conexiuni) și procesele asociate acestor exemple. Folderul (Depozitul de date / Repository) care conține toate aceste materiale poate fi descărcat de la adresa:

https://s.go.ro/lkd3ufcb
parola de acces: 166011

Desigur, resursele de învățare menționate în cadrul primului volum sunt utile și pentru volumul secund. O resursă generală de învățare este site-ul companiei care a produs softul RapidMiner Studio (https://rapidminer.com/). Aici pot fi accesate următoarele categorii de resurse:

- ilustrări ale utilizării softului pentru a rezolva diferite probleme practice organizate pe domenii (Communications, Health, Insurance etc.) și studii de caz (Churn Prevention, Customer Segmentation, Fraud Detection etc.);
- resurse educaționale: Academia RapidMiner și Training & Certification;
- resurse generale (blog, studii de caz, dicționar, rapoarte & unelte, webinarii & materiale video) și evenimente;
- ajutor: documentație (manuale), comunitate, suport;
- softuri conexe softului RapidMiner.

Există două manuale oficiale asociate programului RapidMiner, ambele disponibile online (https://docs.rapidminer.com/). Primul dintre acestea, „RapidMiner Studio Manual" (RapidMiner, 2014) conține informații cu privire la terminologia RapidMiner, instalarea softului, realizarea unui proces (analize), vizualizarea datelor și a rezultatelor, depozitul de date (Repository). Al doilea manual, „RapidMiner 9. Operator Reference Manual" (RapidMiner, 2022) conține informațiile de bază relativ la toate comenzile disponibile (operatorii) în RapidMiner, organizate logic, pe categorii majore (Data Access, Blending, Cleansing, Modeling, Scoring, Validation, Utility, Extensions, Deployment) și sub-categorii. Manualele oficiale RapidMiner au constituit o sursă importantă de informații pentru scrierea acestui volum.

Pentru nevoi relativ mai specifice de informare, foarte utile sunt rapoartele, studiile de caz și blogul RapidMiner. Temele discutate aici sunt extrem de diverse, oferă sfaturi și împărtășesc experiențe ale unor practicieni ai domeniului data mining. Secțiunea RapidMiner Community a site-ului oferă un spațiu de informare și discuție, respectiv o cale rapidă prin care se poate obține ajutorul membrilor comunității. Postările pot fi căutate folosind diferite tag-uri sau propriile cuvinte.

Pentru a accesa diferite prezentări video putem apela la site-ul RapidMiner sau la pagina RapidMiner de pe YouTube. O serie impresionantă de resurse este accesibilă în secțiunea RapidMiner Academy a site-ului RapidMiner. Aceasta organizează informațiile în trei sub-secțiuni: Get Started, Content Library și Certification. Toate

resursele oferite sunt gratuite. Informații generale relativ la această resursă pot fi accesate aici.

O altă resursă utilă pentru învățare o reprezintă cărțile care folosesc softul RapidMiner Studio pentru a produce analize punctuale sau proiecte de data mining pe teme specifice. Deși uneori sunt folosite versiuni mai vechi ale RapidMiner Studio analizele și procesele prezentate în acestea pot fi reluate fără probleme în versiunile recente ale softului. Cele mai importante cărți, utilizate și în acest volum, prezentate în ordinea publicării, sunt următoarele:

- Data mining: a tutorial-based primer (Roiger, 2017)
- Data Science: Concepts and Practice (Kotu & Deshpande, 2019)
- Data Analytics for the Social Sciences: Applications in R (Garson, 2021)
- Machine Learning for Social and Behavioral Research (Jacobucci et al., 2023)
- Machine Learning for Business Analytics: Concepts, Techniques and Applications in RapidMiner (Shmueli et al., 2023)

1.2. Analizele de data mining în contextul resurselor umane

Exemple de beneficii și output-uri

Analizele de tip data mining, sau, mai general, inteligența artificială (AI), pot fi utilizate în relație cu o multitudine de sectoare din domeniul resurselor umane. O listă organizată în funcție de sector, beneficii așteptate și măsuri de output este prezentată în Tabelul 1.2-1. Observăm că retenția angajaților reprezintă doar unul dintre multiplele obiective și puncte de interes pentru specialiștii HR și managerii unei instituții. Dată fiind importanța acestui aspect și mai ales pentru a simplifica procesul de învățare am preferat să ilustrăm discuțiile și analizele prezentate în acest volum folosind doar conceptul de retenție a personalului (fluctuația de personal). Firesc, setul de date folosit pentru realizarea exemplelor de analiză include această variabilă, la nivel individual, iar analizele prezentate urmăresc predicția apartenenței la clasele variabilei de interes (angajatul părăsește sau nu compania).

Tabelul 1.2-1. Beneficiile și output-urile așteptate ale utilizării AI în HR

Application of AI in HR	Examples of expected benefits	Examples of outcome measures
Enhanced candidate experience	• More informative pre-hire communication • Better match of job seekers to roles	• Candidate conversion rate • New hire productivity
Efficient and effective recruitment	• Better prioritization of job requisitions • Accelerated time-to-hire • Accurate assessment of diverse candidates • Identification of the most qualified candidates	• Skill shortages or unskilled vacancies • Average time to fill open positions • Selection ratios of minority and majority candidates • New hire productivity
Enhanced motivation	• Better manager support for their employees • Improved employee experience	• Employee retention • Engagement or experience survey scores
Smarter compensation planning	• Increased pay transparency for employees • Optimized compensation budgets aligned with business strategy	• Compensation satisfaction survey scores • Overpaid or underpaid worker count
Personalized learning	• Accelerated employee skill acquisition • Better alignment of employee skills with business strategy • Enhanced learning experience	• Match between current and required skills mix, skill gap closure • Employee productivity • Course enrollments and completion rates
Career development for all	• Employee driven career management • Employee clarity on opportunities	• Career satisfaction survey scores • Number of internal job applications and moves
24/7/365 Employee support	• Better informed and more productive employees via faster, more accurate answers to questions • Reduced number of support center staff	• Number of process violations or exceptions • Labor costs

Sursa: Nigel Guenole & Sheri Feinzig. 2018. The Business Case for AI in HR With Insights and Tips on Getting Started.

Atunci când vine vorba despre avantajele utilizării AI în domeniul resurselor umane, exemplul care apare frecvent este cel al IBM.[1] Conform declarațiilor managerilor de top ai IBM, sistemul AI utilizat de companie poate prezice cu o acuratețe de 95% cine sunt angajații care sunt pe cale de părăsi locul de muncă. Alte beneficii menționate sunt următoarele:

- Prelucrarea rapidă a celor peste 8000 de CV-uri primite zilnic.
- Reducerea cu 300 milioane a costurilor asociate retenției de personal.
- Reducerea cu 30% a numărului de angajați din departamentul de resurse umane, majoritatea pozițiilor eliminate fiind cele rutiniere, slab plătite, cu valoare asociată redusă.
- Orientarea angajaților spre dobândirea unor competențe utile, necesare companiei.
- Oferirea de feedback în carieră angajaților.

Fluctuația de personal

Fluctuația de personal se referă la diferența dintre plecările și angajările realizate la nivelul unei organizații (diviziuni a acesteia) într-o perioadă de timp de referință (de obicei un an). Printre cauzele plecărilor (încetării contractului de muncă) se numără cel puțin următoarele: pensionarea angajatului, mutarea într-o altă localitate / țară, starea de sănătate, decesul, schimbarea carierei, mutarea la altă companie și desființarea postului. Atunci când plecările nu sunt dorite de organizație și aceasta trebuie să angajeze alte persoane în locul celor plecate, apar o serie de probleme și costuri suplimentare. Se estimează că înlocuirea unui angajat costă o companie, în medie, 6-9 salarii asociate poziției respective. Aici intră în principal costurile directe cu angajarea (recrutare, selecție și angajare efectivă), respectiv cu integrarea (instruirea și perioada de acomodare). Nu sunt însă de neglijat nici costurile indirecte generate de:

- scăderea productivității, deci și a încasărilor, mai ales dacă angajații care pleacă sunt valoroși, au calificări greu de găsit;
- scăderea stocului de cunoaștere la nivelul organizației;

[1] Rosenbaum, Eric. 2019. IBM artificial intelligence can predict with 95% accuracy which workers are about to quit their jobs. https://www.cnbc.com/2019/04/03/ibm-ai-can-predict-with-95-percent-accuracy-which-employees-will-quit.html

- scăderea gradului de dedicare la nivelul angajaților rămași;
- afectarea reputației organizației;
- influențele negative asupra culturii organizaționale.[2]

Cu privire la situațiile de încetare a contractului de muncă, literatura de limbă engleză folosește, adesea interșanjabil, unul dintre conceptele „employee attrition", „employee turnover" sau „employee churn". Alteori, se consideră că „employee attrition" și „employee turnover" reprezintă două forme diferite ale „employee churn" (Tabelul 1.2-2). Conform acestei perspective teoretice, prin fiecare dintre cele trei concepte înțelegem următoarele:

- **„employee attrition"** se referă la diminuarea / scăderea numărului de angajați al unei companii; conceptul este adesea tradus în română prin „uzura angajaților / personalului"; având în vedere sensurile și conotațiile conceptului de uzură în limba română, credem că această alegere terminologică nu este cea mai potrivită, prin urmare apreciem că „diminuarea / scăderea numărului de angajați" și, de ce nu, atriție[3], reflectă mai bine sensul original; prin urmare, „rata atriției" se referă la raportul dintre numărul angajaților care au plecat și numărul mediu al angajaților relativ la perioada de referință (de obicei un an) (Raza et al., 2022);
- **„employee turnover"** se referă la situația în care un angajat al unei companii este înlocuit de altul (indiferent de motiv); este tradus în română prin sintagme precum „fluctuația personalului / angajaților", „fluctuația de personal" sau „rotația personalului / angajaților"; apreciem că sintagma de „înlocuirea personalului / angajaților" surprinde cel mai bine sensul inițial din engleză (semnifică simultan cele două fenomene legate: situația de încetare a unui contract de muncă, concomitent cu angajarea altei persoane pe aceeași poziție); astfel, „rata înlocuirilor" se referă la raportul dintre numărul angajaților care au fost înlocuiți, indiferent de motiv, și numărul mediu al angajaților relativ la perioada de referință (de obicei un an);
- **„employee churn"** se referă la ambele fenomene descrise anterior; probabil, o traducere potrivită ar fi „fluctuația de personal" sau „ fluctuația angajaților" (termenul de fluctuație include situațiile de scădere / creștere a numărului de angajați, respectiv de înlocuire a angajaților).

---

[2] Archive User. 2016. Unlock the secrets to employee retention with predictive analytics. Roberts, William. 2019. How IBM and a supply chain company predict employee retention.

[3] Termenul de atriție apare deja în dicționarele online, uneori ca sinonim pentru atrițiune, și pare să fie utilizat frecvent de către specialiștii din domeniul resurselor umane.

Tabelul 1.2-2. Fluctuația de personal și formele acesteia: atriția și înlocuirea angajaților

Fluctuația de personal (employee churn)	**Plecarea este ...**	**Organizația ...**
Atriția angajaților (employee attrition)	voluntară	nu angajează o altă persoană pe acel post sau elimină postul din organigramă
Înlocuirea angajaților (employee turnover)	(in)voluntară	angajează o altă persoană

Pentru a înțelege mai exact la ce anume se referă aceste concepte vom considera un exemplu simplu. Să presupunem că, la finalul anului trecut, o companie avea 250 de angajați; până la finalul anului, în companie, au mai fost angajate 20 de persoane (deci la finalul anului compania are 270 angajați, adică o medie anuală de 260 angajați), 40 de angajați au părăsit voluntar sau involuntar compania, iar pe 9 dintre pozițiile care au fost părăsite voluntar, compania nu a mai angajat pe altcineva (restul de 31 poziții au fost ocupate de alte persoane). Pentru acest exemplu, rata atriției / pierderii angajaților (employee attrition rate) este 3.5% (9 / 260 x 100, unde 9 este numărul de angajați care au plecat și nu au fost înlocuiți), iar rata de înlocuire a angajaților (employee turnover rate) este 11.9% (31 / 260 x 100, unde 31 este numărul de angajați care au plecat și au fost înlocuiți). Uneori, „turnover rate" este calculat, ca raport între numărul total de plecări și numărul mediu al angajaților (în acest caz ar fi 40 / 260 x 100, adică 15.4%). În această situație, traducerea conceptului de „turnover rate" prin „rata înlocuirilor" nu mai reflectă formula de calcul, fiind de preferat sintagma „rata plecărilor".

Factori care influențează plecarea din companie

Pornind tot de la experiența companiei IBM,[4] pot fi identificate două surse majore ale părăsirii companiei: paternurile de promovare și zone de fricțiune. Astfel, cu cât timpul petrecut pe aceeași poziție este mai mare, cu atât acesta este mai nemulțumit, prin urmare cresc șansele de plecare din companie. Efectul este oarecum moderat de context, mai exact de ratele de promovare ale angajaților similari. Zonele de fricțiune se referă în principal la prezența navetei și a muncii peste program: șansele de plecare cresc în funcție de distanța, timpul și costul

[4]McLaren, Samantha. 2019. Here's How IBM Predicts 95% of Its Turnover Using Data. https://www.linkedin.com/business/talent/blog/talent-strategy/ibm-predicts-95-percent-of-turnover-using-ai-and-data

asociate navetei; similar, cei care muncesc peste program mai frecvent, mai multe ore, din cauze externe, sunt și cei cu șanse mai mari să plece din companie.

Un studiu realizat de compania Percepyx[5] pe un eșantion de 100 mii angajați despre care aveau informații din două anchete (engagement survey și exit interview) a arătat că plecarea din companie corelează negativ cu nivelul de implicare (măsurat prin patru itemi: angajatul are intenția de a rămâne în companie, recomandă compania altora, este motivat intrinsec, este mândru că lucrează în companie), respectiv pozitiv cu calitatea relației cu șeful direct.

1.3. Setul de date „ibm-hr-analytics-attrition-dataset”

Descrierea generală a setului de date

Principalul set de date utilizat pentru a ilustra algoritmii de clasificare prezentați în acest volum este unul deja clasic în domeniul resurselor umane: „ibm-hr-analytics-attrition-dataset”.[6] Setul de date este unul fictiv, creat de specialiștii în analiza datelor de la IBM. Deși a fost postat relativ recent (2021) pe site-ul Kaggle, a fost analizat deja, pe același site doar, de 611 ori (numărul de analize ce apar sub forma unor notebooks), respectiv a fost descărcat de aproximativ 116 mii ori.[7]

Setul de date a fost creat cu scopul de a ilustra impactul pe care diferiți factori situați la nivel individual îl au asupra șanselor ca un angajat să plece sau nu dintr-o companie. Setul conține 35 de atribute și 1470 cazuri (angajați ai unei companii). Un extras din acest set de date este prezentat în Figura 1.3-1. Versiunea utilizată pentru analizele din acest volum se află în folderul Data. Versiunea originală este inclusă atât ca format de tip Excel („employee_attrition”), cât și RapidMiner („employee_attrition_original”). Analizele din volum sunt realizate folosind varianta RapidMiner ușor modificată („employee_attrition”), respectiv subseturi ale acesteia (selecția subseturilor a fost realizată folosind o eșantionare de tip stratificat aleator):

- pentru instruirea și validarea modelelor am folosit subsetul de date denumit „employee_attrition_training_validation” (1029 cazuri);

[5] Killham, Emily. 2022. Employee Attrition Analytics: The Who, When & Why Of Employee Turnover. https://blog.perceptyx.com/employee-attrition-analytics

[6] Link-ul (https://www.ibm.com/communities/analytics/watson-analytics-blog/hr-employee-attrition/) nu mai este funcțional în prezent.

[7] https://www.kaggle.com/datasets/pavansubhasht/ibm-hr-analytics-attrition-dataset

- pentru testarea modelelor am folosit subsetul „employee_attrition_testing” (441 cazuri).

Relativ la acest set de date, variabila de interes, „Attrition”, este de tip binominal și înregistrează situația în care se află fiecare angajat, mai exact, dacă a părăsit sau nu compania, indiferent de motiv sau cauză. În contexte reale de cercetare este important să distingem între tipurile de plecări (attrition / turnover) pentru simplul motiv că acestea pot avea distribuții diferite, respectiv relații diferite cu variabilele independente. De exemplu, un studiu relativ recent (An, 2019) a arătat că rata plecărilor involuntare are o relație de tip U inversat cu performanța organizațională (inițial impactul este pozitiv, apoi neutru, apoi negativ), în timp ce relația dintre rata plecărilor voluntare și performanță este neclară. Cu privire la plecările de tip voluntar distingem între plecări voluntare funcționale, respectiv disfuncționale. La rândul lor, plecările voluntare disfuncționale pot fi de două tipuri: plecări ce pot fi evitate, respectiv care nu pot fi evitate (David, 2008, p. 2). În practică este util să analizăm separat fiecare dintre aceste tipuri de plecări pentru simplul motiv că au, cel mai adesea, cauze diferite. În cazul de față, datorită lipsei informației, vom considera că atributul Attrition include toate tipurile de plecări (voluntare sau involuntare, cu sau fără înlocuirea foștilor angajați).

În acest set de date, variabila EmployeeNumber (redenumită Id în cazul de față) este de tip id (cod numeric care identifică fiecare angajat). Restul atributelor măsoară diferite tipuri de caracteristici asociate angajaților, și anume:

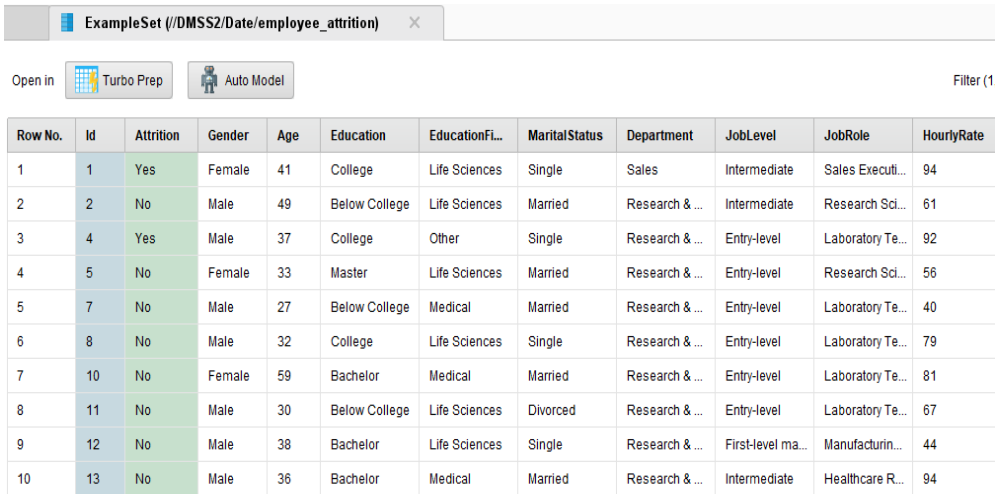
- Socio-demografice:
  - □ Gender: genul angajatului (masculin / feminin),
  - □ Age: vârsta angajatului (ani),
  - □ Education: nivelul educației formale (liceu, mai puțin decât colegiu, colegiu, master, doctorat),
  - □ EducationField: domeniul studiilor (științele vieții, medicină, marketing, tehnic, resurse umane, alt domeniu),
  - □ MaritalStatus: starea civilă (necăsătorit, căsătorit, divorțat).

- Locul de muncă actual:
  - □ Department: departamentul (cercetare & dezvoltare, vânzări, resurse umane),
  - □ JobRole: poziția ocupată (Sales Executive, Research Scientist, Laboratory Technician, Manufacturing Director, Healthcare Representative, Manager, Sales Representative, Research Director, Human Resources),

- JobLevel: nivelul poziției ocupate (Entry-level, Intermediate, First-level management, Middle management, Senior management),[8]
- HourlyRate, DailyRate, MonthlyRate: tariful perceput (USD pe oră / zi / lună),
- MonthlyIncome: salariul lunar (USD),
- PercentSalaryHike: creșterea salarială (procentuală),
- StockOptionLevel: stocul de acțiuni deținut (deloc, mic, mediu, mare),
- BusinessTravel: călătorii în interes de serviciu (deloc, rar, frecvent),
- OverTime: lucrează peste program (da / nu),
- TrainingTimesLastYear: numărul de training-uri la care a participat anul anterior,
- JobInvolvement: implicarea la locul de muncă (redusă, medie, mare, foarte mare),
- WorkLifeBalance: echilibrul muncă – viață (prost, bun, mai bun, cel mai bun),
- JobSatisfaction: satisfacția muncii (redusă, medie, mare, foarte mare),
- RelationshipSatisfaction: satisfacția relativ la relațiile de muncă (redusă, medie, mare, foarte mare),
- EnvironmentSatisfaction: satisfacția relativ la condițiile de muncă (redusă, medie, mare, foarte mare),
- PerformanceRating: nivelul de performanță (slab, bun, excelent, excepțional),
- TotalWorkingYears: numărul de ani de muncă,
- YearsAtCompany: numărul de ani ca angajat al companiei,
- YearsInCurrentRole: numărul de ani ca angajat pe poziția actuală,
- YearsSinceLastPromotion: numărul de ani de la ultima promovare,
- YearsWithCurrentManager: numărul de ani cu șeful din prezent,
- DistanceFromHome: distanța până la locul de muncă (mile),
- NumCompaniesWorked: numărul de companii pentru care a lucrat anterior.

[8] În setul de date original acest atribut lua valori numerice de la 1 la 5. Aici le-am recodat astfel: 1 = Entry-level, 2 = Intermediate, 3 = First-level management, 4 = Middle management, 5 = Senior management.

Figura 1.3-1. Un extras din setul de date „employee attrition”



ExampleSet (//DMSS2/Date/employee_attrition)

Open in Turbo Prep Auto Model Filter (1

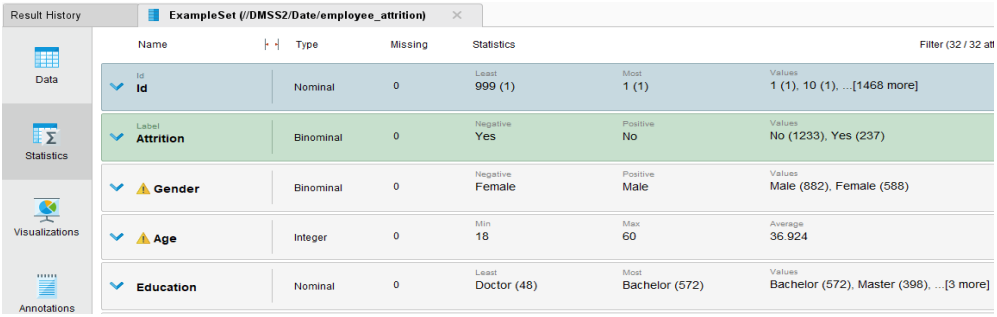
Row No.	Id	Attrition	Gender	Age	Education	EducationFi...	MaritalStatus	Department	JobLevel	JobRole	HourlyRate
1	1	Yes	Female	41	College	Life Sciences	Single	Sales	Intermediate	Sales Executi...	94
2	2	No	Male	49	Below College	Life Sciences	Married	Research & ...	Intermediate	Research Sci...	61
3	4	Yes	Male	37	College	Other	Single	Research & ...	Entry-level	Laboratory Te...	92
4	5	No	Female	33	Master	Life Sciences	Married	Research & ...	Entry-level	Research Sci...	56
5	7	No	Male	27	Below College	Medical	Married	Research & ...	Entry-level	Laboratory Te...	40
6	8	No	Male	32	College	Life Sciences	Single	Research & ...	Entry-level	Laboratory Te...	79
7	10	No	Female	59	Bachelor	Medical	Married	Research & ...	Entry-level	Laboratory Te...	81
8	11	No	Male	30	Below College	Life Sciences	Divorced	Research & ...	Entry-level	Laboratory Te...	67
9	12	No	Male	38	Bachelor	Life Sciences	Single	Research & ...	First-level ma...	Manufacturin...	44
10	13	No	Male	36	Bachelor	Medical	Married	Research & ...	Intermediate	Healthcare R...	94

Desigur, nu toate atributele incluse în setul de date sunt și neapărat utile pentru a prezice părăsirea companiei de către angajați. Atributele reținute relativ mai frecvent în modelele finale de predicție sunt: Age, Education, JobLevel, MonthlyIncome, NumCompaniesWorked, PercentSalaryHike, PerformanceRating, TotalWorkingYears, YearsAtCompany, YearsInCurrentRole, YearsSinceLastPromotion, YearsWithCurrManager (Raza et al., 2022).

Grafice & analize univariate

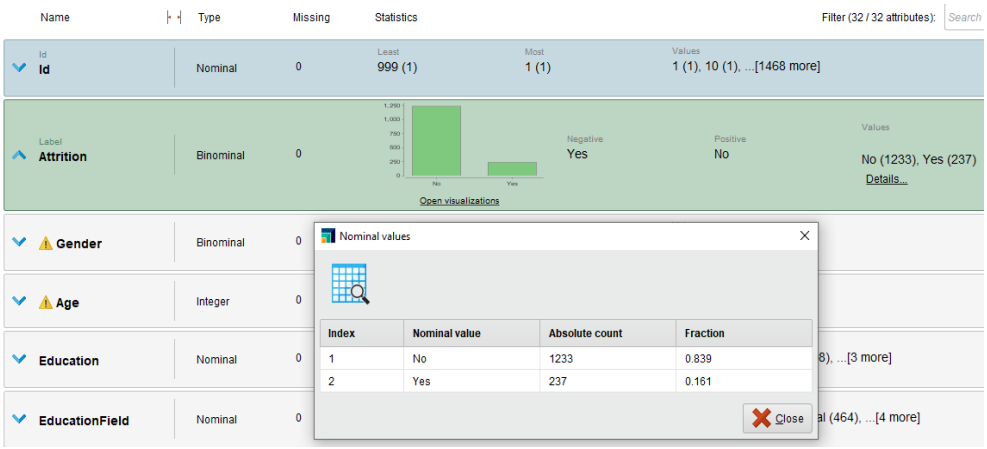
Pentru a analiza fiecare variabilă separat, selectăm tabul Statistics (Figura 1.3-2). Aici, fiecare atribut apare pe o linie separată ce include informații cu privire la numele atributului (Name), tipul acestuia (categorial, numeric etc.) (Type), numărul de cazuri fără valori (Missing) și câteva măsuri relativ la distribuția statistică, diferențiat în funcție de nivelul de măsurare al atributului (Statistics). În cazul acestui set de date observă că avem atribute de tip categorial, unele cu două categorii (binominal), altele cu trei sau mai multe categorii (nominal), respectiv atribute numerice cu valori întregi (integer).

Figura 1.3-2. O descriere sintetică a atributelor incluse în setul de date



Atunci când selectăm un atribut observăm că sunt afișate câteva informații suplimentare, diferențiat în funcție de tipul atributului (nivelul de măsurare asociat acestuia). Astfel, pentru atributele de tip nominal, putem vizualiza un grafic de tip bară care indică numărul de cazuri asociat fiecărei categorii de răspuns, respectiv putem afișa un tabel care conține aceeași informație la care adaugă și ponderile categoriilor (%). În cazul atributelor nominale de tip binominal este menționată suplimentar asocierea dintre fiecare categorie și eticheta pozitiv / negativ (care răspuns este considerat a fi pozitiv și care negativ; acest lucru va stabili ordinea în care sunt afișate informațiile în tabele, grafice, matricea de confuzie etc.). În cazul atributului de interes, Attrition, observăm că setul de date conține 1233 de angajați care au rămas în companie (adică 84%) și 237 care au plecat (16%) (Figura 1.3-3).

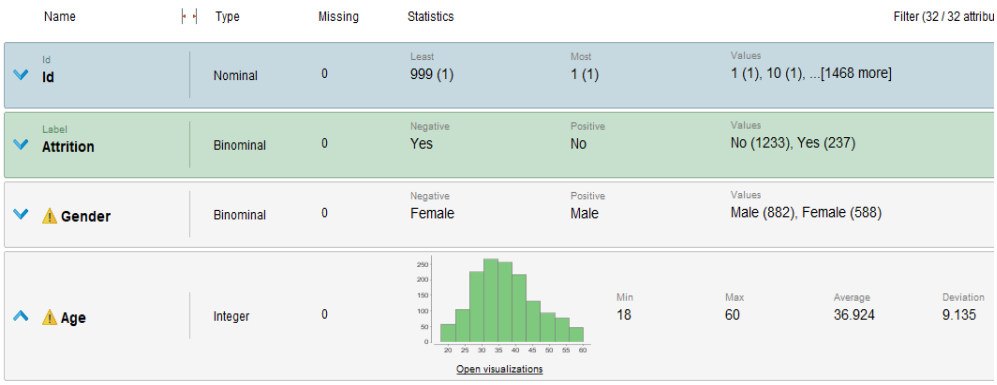
Figura 1.3-3. Descrierea sintetică a unui atribut de tip Binominal (Attrition)



Dacă atributul este de tip numeric, la selecție vor fi afișate următoarele: un grafic de tip histogramă cu numărul de cazuri care i-au valori în diferite intervale (acestea

sunt stabilite automat), valoarea minimă, valoarea maximă, media și abaterea standard. În cazul atributului Age, observăm că distribuția valorilor este apropiată de normalitate, intervalul de variație este [18-60], media 36.9, iar abaterea standard 9.1 (Figura 1.3-4).

Figura 1.3-4. Descrierea sintetică a unui atribut de tip Integer (Age)



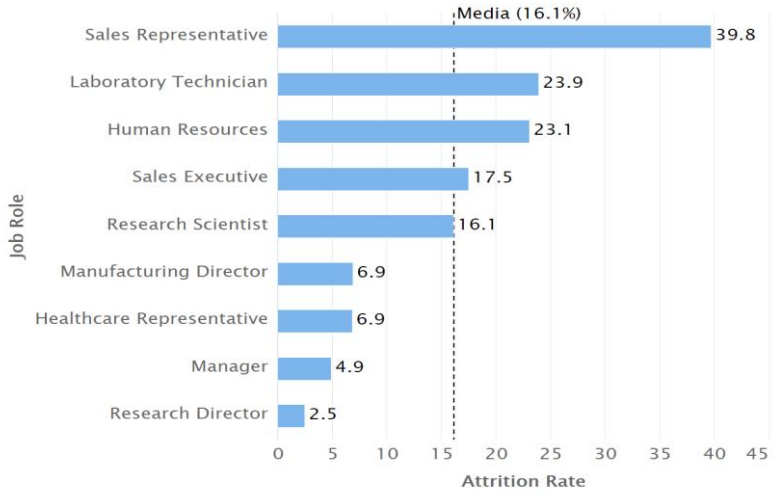
Grafice & analize bivariate

Prin analiză bivariată ne referim la analiza relațiilor dintre două variabile, în acest caz la analiza relației dintre variabila dependentă (atriție) și fiecare variabilă independentă (o parte din analize vor fi relativ la două variabile independente). Rata atriției la nivelul întregului set de date este 16.1%. Ne întrebăm dacă rata atriției variază sau nu de la o categorie de angajați la alta. Graficele prezentate în continuare urmăresc să răspundă tocmai la această întrebare. Pentru a ilustra capacitățile de vizualizare a datelor ale softului RapidMiner, pe alocuri, vom prezenta aceeași relație folosind diferite reprezentări grafice. Toate aceste grafice pot fi realizate cu ajutorul proceselor incluse în folderul „Grafice”.[9]

[9] De exemplu, pentru a realiza Graficul 1.3-1, vom rula procesul „attrition by job_role.rmp”. Procesul pregătește datele necesare pentru realizarea graficului (selectează atributele, agregarea datelor etc.), apoi realizează graficul folosind setările indicate în fișierul de tip json salvat anterior (numit „attrition by job_role.json”; acesta conține toate setările graficului: tip, axe, font etc.), iar în final îl salvează ca fișier png („attrition by job_role.png”). Procesul „_grafice.rmp” realizează și salvează toate graficele care pot fi realizate direct din setul de date, cele care nu necesită în prealabil o agregare a datelor. Și în acest caz setările graficelor au fost stabilite și salvate în prealabil, pentru fiecare grafic în parte, în format json. Operatorul „Plot Data” permite indicarea fișierului json care conține setările dorite ale graficului, dimensiunile și numele sub care să fie salvat graficul.

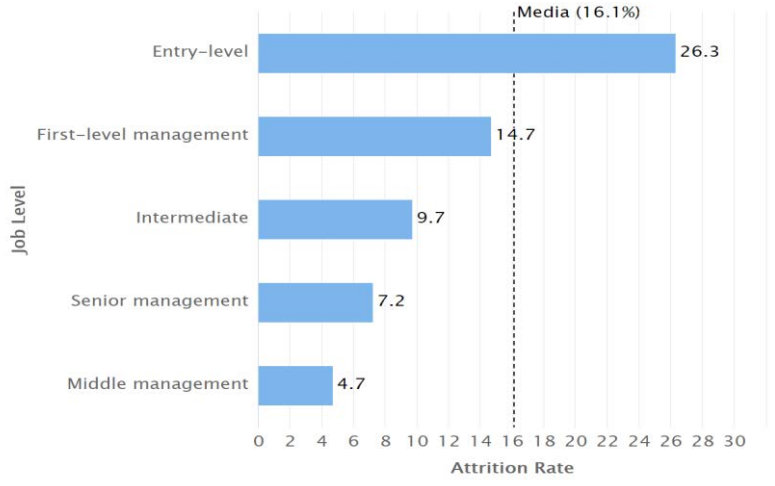
Pentru început, observăm că există o variație foarte mare a atriției în funcție de poziția ocupată (Graficul 1.3-1): cei care ocupă poziția de reprezentant de vânzări au cea mai mare rată de atriție (39.8%), iar managerii și directorii de cercetare cea mai mică (4.9%, respectiv 2.5%).

Graficul 1.3-1. Rata atriției în funcție de poziția ocupată (bar plot)



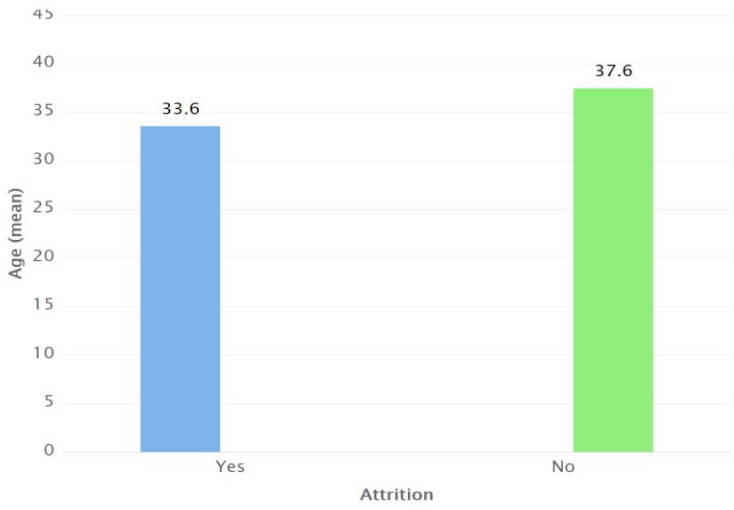
Firesc, o variație mare apare și în funcție de nivelul poziției ocupate (Graficul 1.3-2): pozițiile de tip „entry-level" au cea mai mare rată de atriție (26.3%), iar cele de conducere (middle & senior management) cea mai mică (4.7%, respectiv 7.2%)

Graficul 1.3-2. Rata atriției în funcție de nivelul poziției ocupate (bar plot)

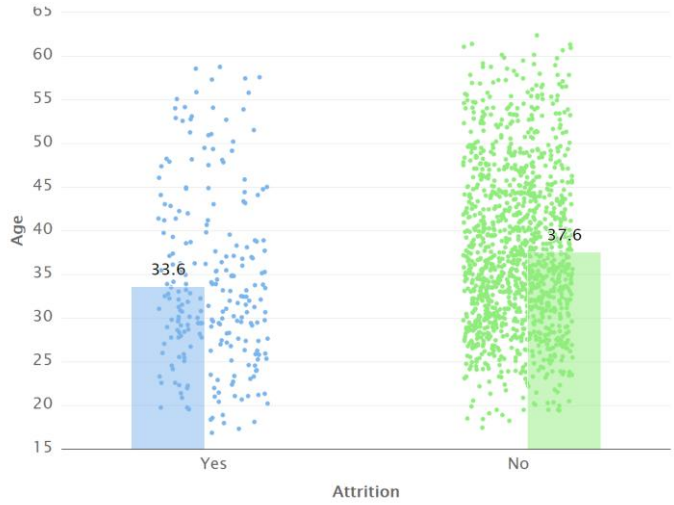


Următoarele trei grafice (Graficul 1.3-3 - Graficul 1.3-5) prezintă, în diferite forme, relația dintre vârsta angajaților și atriție, mai exact, descriu fiecare dintre cele două categorii ale atributului atriție (pleacă / nu pleacă) în funcție de vârstă. Firesc, ne așteptăm ca angajații care pleacă să fie, în medie, ceva mai tineri. Se observă că așa și stau lucrurile, deși diferența de vârstă este mai degrabă redusă: vârsta medie a angajaților care părăsesc compania este 37.6 ani, iar a celor care rămân este 33.6 ani. Această relație poate fi reprezentată sub forma unor bare (verticale în acest caz) (Graficul 1.3-3), a unor bare alături de care apar și cazurile (Graficul 1.3-4), respectiv sub forma unui box plot (Graficul 1.3-5). În acest din urmă caz, graficul prezintă, separat pentru fiecare categorie de atriție, valoarea mediană a vârstei, intervalul de vârstă în care se situează cele 50% dintre cazuri poziționate la mijlocul distribuției atributului vârstă, intervalele de vârstă asociate celor mai tineri angajați, respectiv celor mai în vârstă angajați. Și în acest grafic se observă că vârsta celor care pleacă este mai mică cu aproximativ 4 ani, iar forma celor două distribuții este relativ similară. Fiecare tip de grafic reprezintă un compromis diferit între nevoia de simplitate a reprezentării cu scopul de a facilita înțelegerea și a transmite cât mai repede mesajul, respectiv cât din cantitatea inițială de informație este păstrată. Funcție de situația specifică în care ne aflăm vom prefera una sau alta dintre aceste reprezentări grafice.

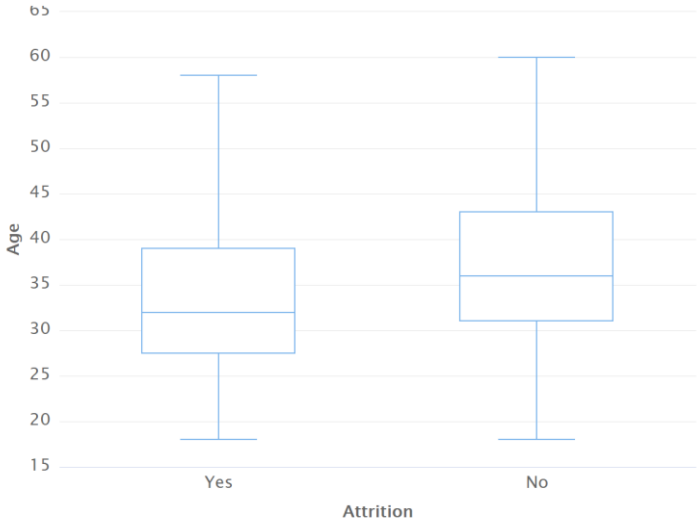
Graficul 1.3-3. Vârsta medie în funcție de starea de atriție (bar plot)



Graficul 1.3-4. Vârsta medie în funcție de starea de atriție (bar + scatter plot)

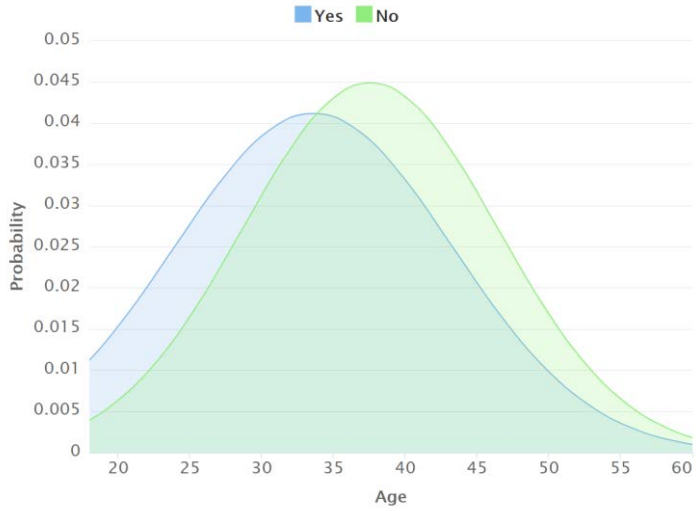


Graficul 1.3-5. Vârsta mediană în funcție de starea de atriție (box plot)

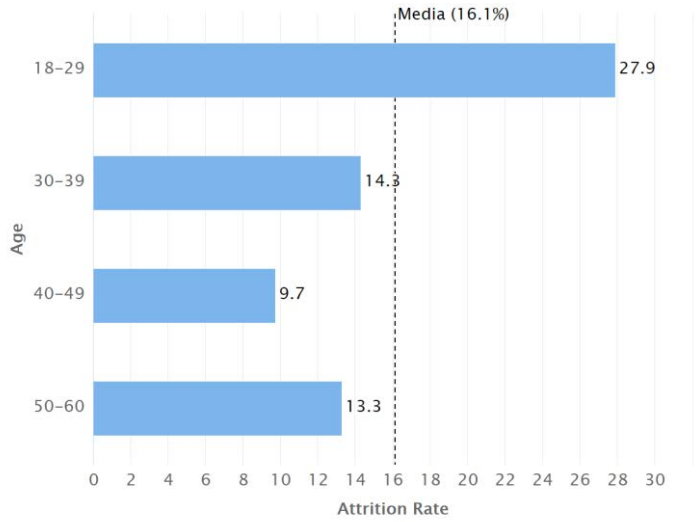


În Graficul 1.3-6 este analizată tot relația dintre atriție și vârstă, de această dată din perspectiva vârstei. Graficul reprezintă probabilitățile de părăsire, respectiv rămânere în companie în funcție de vârsta angajaților. Observăm din nou similaritatea distribuției probabilităților și aceeași diferență (cei care pleacă au cu aproximativ patru ani mai puțin). Dacă ne interesează să calculăm rata atriției pentru diferite categorii de vârstă, putem alege un grafic de tip bar (în prealabil trebuie să recodăm vârsta în categoriile dorite) (Graficul 1.3-7). Observăm din nou că angajații tineri (vârsta 18-29 ani) au cea mai mare rată de atriție (27.8%).

Graficul 1.3-6. Probabilitatea atriției în funcție de vârstă (bell curve plot)



Graficul 1.3-7. Rata atriției în funcție de vârstă (bar plot)

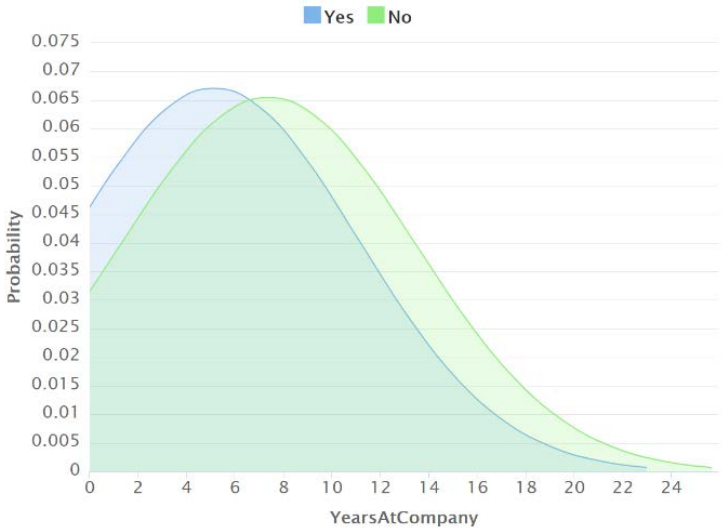


Analizând graficele de mai jos (Graficul 1.3-8 - Graficul 1.3-11), observăm că probabilitatea atriției este mai mare în cazul angajaților ...

- cu o vechime în companie mai mică,
- care au lucrat mai puțin timp cu actualul manager,
- care sunt pe poziția din prezent de mai puțin timp,

– în cazul cărora a trecut mai puțin timp de la ultima promovare (în acest caz diferența este foarte mică).

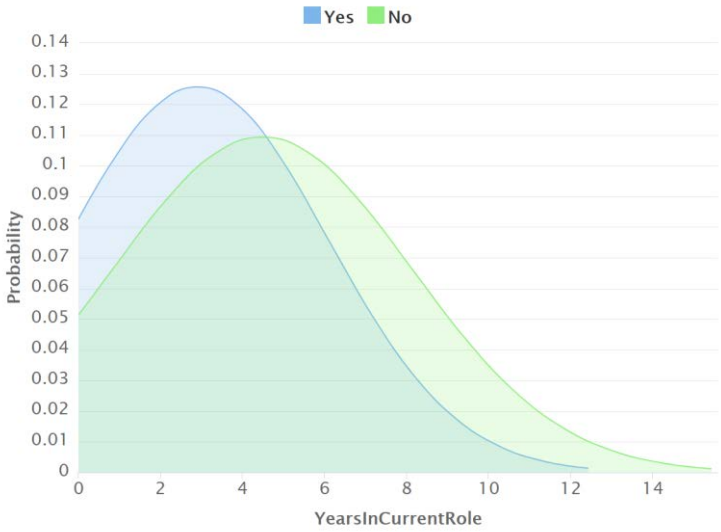
Graficul 1.3-8. Probabilitatea atriției în funcție de vechimea în companie (bell curve plot)



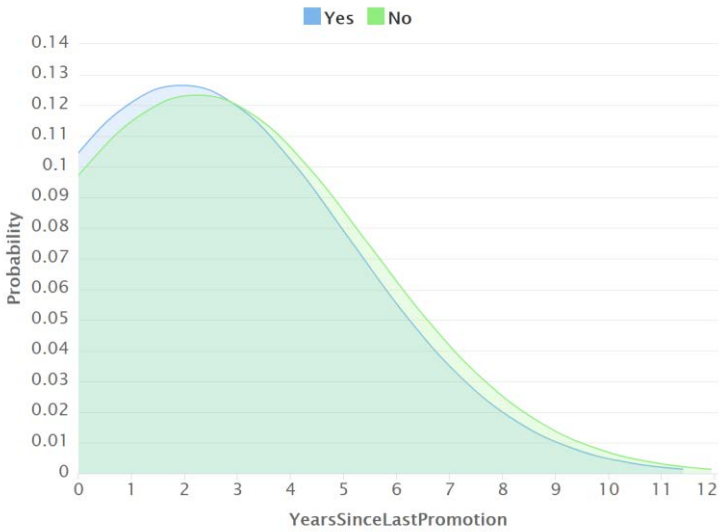
Oarecum surprinzător, creșterea procentuală a salariului este aproape identică în cele două grupuri (cei care pleacă vs. cei care rămân) (Graficul 1.3-12).

Graficul 1.3-9. Probabilitatea atriției în funcție de vechimea alături de managerul actual (bell curve plot)

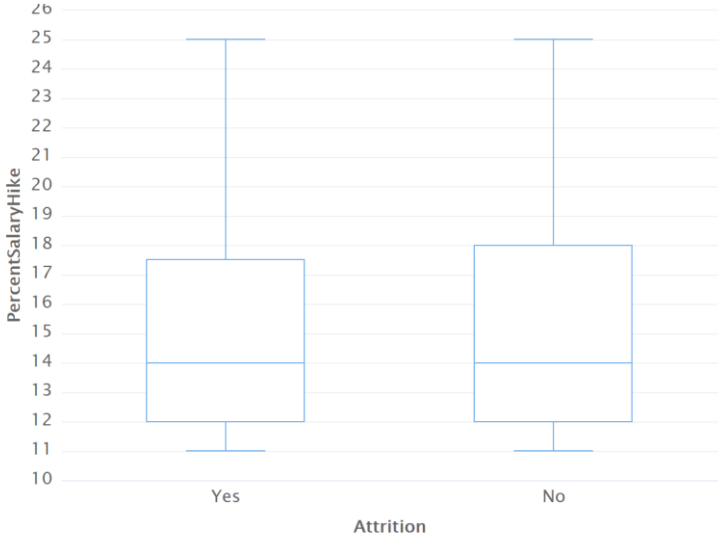
Graficul 1.3-10. Probabilitatea atriției în funcție de vechimea în rolul actual (bell curve plot)



Graficul 1.3-11. Probabilitatea atriției în funcție de numărul de ani de la ultima promovare (bell curve plot)



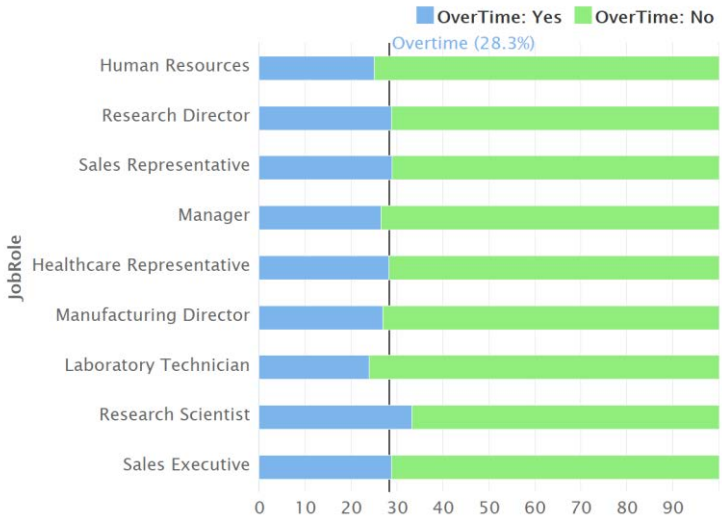
Graficul 1.3-12. Creșterea procentuală a salariului în funcție de starea atriției (box plot)



Adesea este util să analizăm și relațiile dintre predictori. Aici oferim doar două exemple. Primul dintre acestea prezintă relația dintre poziția ocupată și munca peste program folosind două tipuri de grafice (bar & stacked bar). Se observă că ponderea celor care lucrează peste program nu variază prea mult în funcție de poziția ocupată (Graficul 1.3-13, Graficul 1.3-14).

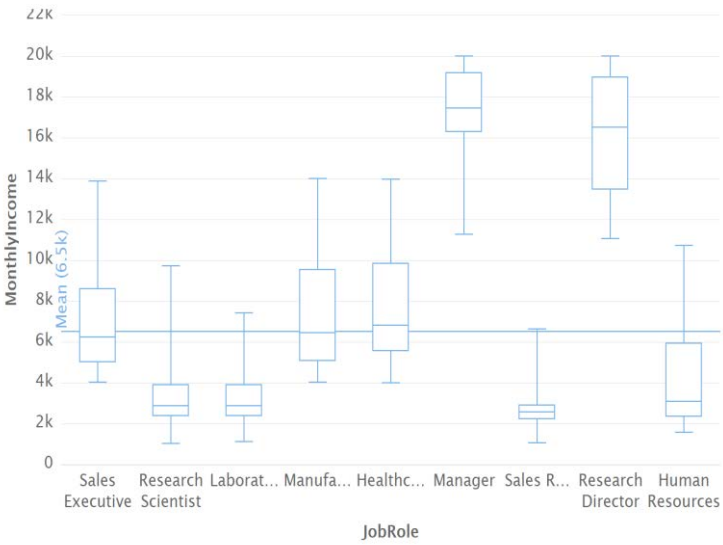
Graficul 1.3-13. Ponderea celor care muncesc peste program în funcție de poziția ocupată (bar plot)

Graficul 1.3-14. Ponderea celor care muncesc peste program în funcție de poziția ocupată (stacked bars plot)



Al doilea exemplu prezintă variația salariului lunar în funcție de poziția ocupată (Graficul 1.3-15). În acest caz observăm o variabilitate foarte mare a salariului (de exemplu, toți managerii și directorii de cercetare câștigă semnificativ mai mult comparativ cu restul pozițiilor).

Graficul 1.3-15. Salariul lunar în funcție de poziția ocupată (box plot)

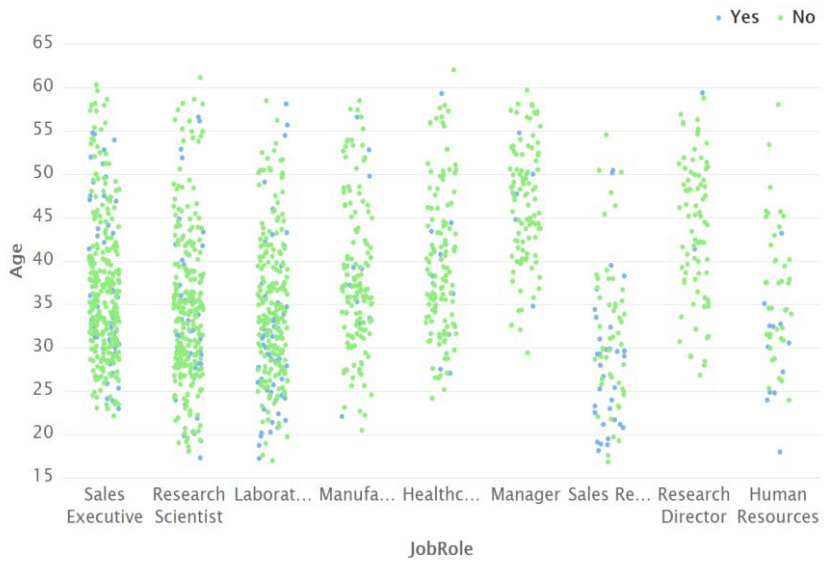


Grafice & analize trivariate

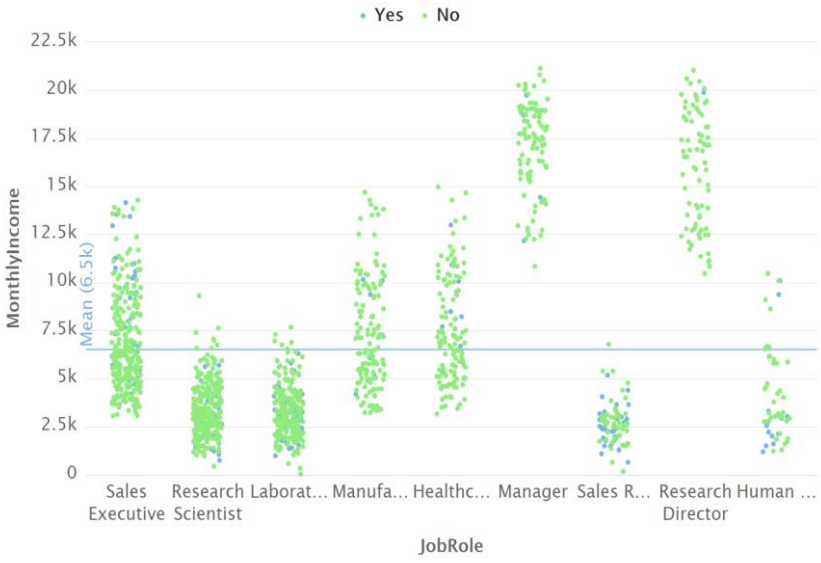
Relațiile dintre atribute sunt adesea complexe, motiv pentru care e necesar să considerăm trei sau chiar mai multe atribute simultan. Aici prezentăm doar relații între trei atribute, unul dintre ele fiind atributul central al analizei, atriția:

- putem observa distribuția cazurilor de atriție în funcție de poziția ocupată și vârstă simultan (Graficul 1.3-16); observăm că anumite combinații de vârstă și poziție cresc semnificativ probabilitatea atriției (de exemplu, tineri angajați pe poziții de reprezentanți de vânzări);
- observăm că salariile variază mult în funcție de poziția ocupată și că atriția este relativ mai prezentă în cazul unor combinații precum salariu mic și reprezentant de vânzări (Graficul 1.3-17);
- angajații care lucrează peste program au salarii ceva mai mari, dar, în același timp, părăsesc compania într-o pondere relativ mai mare (Graficul 1.3-18);
- rata atriției este semnificativ mai mare în cazul anumitor combinații între nivelul poziției și lucrul peste program (Graficul 1.3-19); astfel, mai mult de jumătate dintre angajații pe poziții „entry-level" care lucrează peste program părăsesc compania (rata atriției pentru această categorie de angajați este de trei ori mai mare decât rata atriției la nivelul tuturor angajaților).

Graficul 1.3-16. Atriția în funcție de vârstă și poziția ocupată (scatter plot)

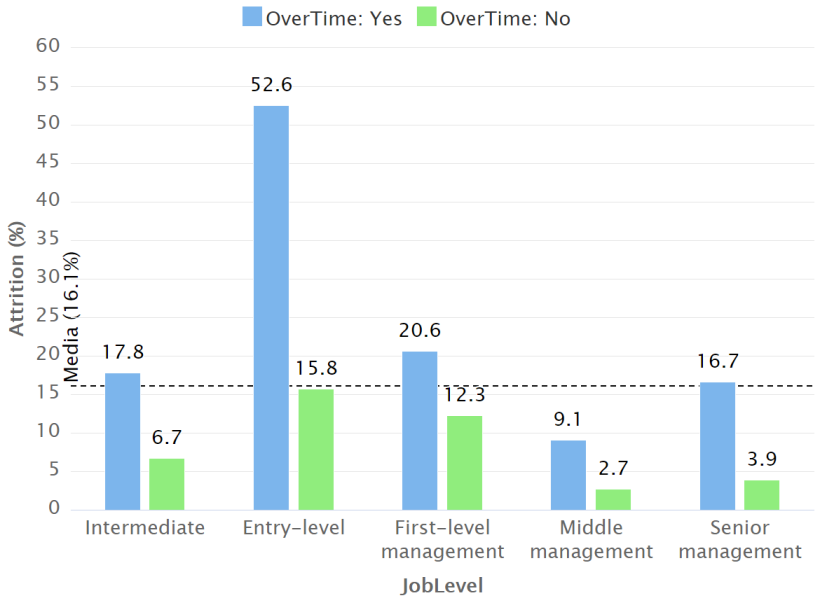


Graficul 1.3-17. Atriția în funcție de salariul lunar și poziția ocupată (scatter plot)



Graficul 1.3-18. Salariul lunar în funcție de munca peste program și atriție (bar + scatter plot)

Graficul 1.3-19. Rata atriției în funcție de munca peste program și nivelul poziției (bar plot)



2. CONSTRUCȚIA UNUI MODEL DE CLASIFICARE

Învățarea automată (machine learning) înseamnă realizarea unei sarcini, rezolvarea unei probleme prin folosirea experienței anterioare și minimizarea erorii. Concret, experiența poate lua forma unor date (metrice și non-metrice / categoriale), texte, sunete și imagini. Dincolo de forma informației, ceea ce contează e volumul mare al acesteia, volum dat (și) de numărul cazurilor. Învățarea se bazează pe aceste cazuri, deci este o învățare de tip inductiv. Rezultatul învățării ia forma unei funcții matematice sau a unui set de reguli, adică a unui model care are anumiți parametri (coeficienți) și care transformă valorile de input în unul dintre posibilele rezultate urmărite. În practică, pentru a ajunge la învățare, e nevoie să includem în model și o serie de asumpții. O primă asumpție e relativ la forma generală pe care o va avea modelul. Putem alege dintre următoarele variante: imitarea creierului uman (rețele neuronale), combinarea logică a variabilelor de input (arbori decizionali, modele liniare), reamintirea cazurilor similare (KNN, SVM), calcularea unor distribuții de probabilitate (modele bayesiene) și simularea evoluției, selecției naturale (programe genetice). O a doua categorie de asumpții se referă la valorile hiper-parametrilor (acolo unde e cazul, adică algoritmul de clasificare conține hiper-parametri), iar a treia la distribuția datelor. Desigur, nu e obligatoriu ca noul model să reînceapă învățarea de la zero. Cel mai adesea putem reutiliza cunoașterea obținută anterior în cazul unor sarcini similare. Astfel, noul model poate refolosi un model anterior (tipul și arhitectura acestuia), poate refolosi valorile anterioare ale hiper-parametrilor și/sau valorile deja învățate ale parametrilor.

2.1. Despre algoritmii de învățare automată

Funcție de domeniul de studiu, de criteriile folosite și de preferințele personale ale autorilor, literatura de specialitate prezintă diferite clasificări ale algoritmilor de învățare automată (Domingos, 2012; James et al., 2021; Wendler & Gröttrup, 2021). În acest sub-capitol vom discuta la modul general despre o clasificare care, dincolo de diferențele de terminologie, pare să fie relativ preferată, vom descrie clasificarea

utilizată de RapidMiner Studio pentru a grupa algoritmii, respectiv vom prezenta o aplicație care ne poate ajuta la identificarea algoritmilor potriviți într-o anumită situație. Însă, înainte de toate acestea, e util să vedem ce înțelegem prin conceptul de algoritm de învățare.

Ce este un algoritm de învățare automată?

Un algoritm este format din una sau mai multe instrucțiuni legate logic între ele. Desigur, nu orice set de instrucțiuni reprezintă un algoritm. Pentru a forma un algoritm, instrucțiunile trebuie să fie suficient de precise, adică să răspundă clar la următoarele întrebări: Care sunt pașii? În ce constă exact fiecare pas? Care este ordinea pașilor? Cum sunt pașii interconectați logic? Algoritmii fac posibilă comunicarea dintre om și computer (acesta este format în principal din miliarde de întrerupătoare numite tranzistori). Tot ceea ce face un algoritm este să închidă și deschidă de foarte multe ori o parte din întrerupătoare, într-o anumită ordine (un întrerupător poate fi doar în una din două stări, zero sau unu, curentul trece sau nu). Un bit dintr-un computer al ANAF specifică dacă am plătit sau nu taxele, alt bit de la CNAS indică faptul că suntem asigurați sau nu la sistemul de sănătate publică etc. Fiecare din aceste două informații este produsă de un algoritm. Combinând cei doi algoritmi obținem un algoritm nou, relativ mai complex (Domingos, 2015, pp. 1–3).

Un algoritm are ca input datele și ca output rezultatele. Ceea ce numim învățare automată (machine learning) funcționează exact invers: inputul ia forma datelor (informației) și a rezultatului dorit, iar outputul constă în algoritmul care transformă datele în rezultatul dorit. În acest caz, un algoritm a produs un alt algoritm. Un algoritm care are capacitatea să învețe se numește „learner” (Domingos, 2015, p. 6). În acest volum vom învăța despre astfel de algoritmi.

Fiecare algoritm are la bază anumite asumpții. Un algoritm poate învăța doar dacă se bazează pe aceste asumpții. Totodată, asumpțiile făcute de un algoritm determină ceea ce acesta poate învăța, mai exact cât de bine poate performa în relație cu o sarcină specifică și un set de date. În concluzie, performanța unui algoritm depinde de tipul de sarcină. Până când nu vom descoperi un „Master Algorithm” (Domingos, 2015) și nu vom include ca input alături de date și asumpțiile majore, e necesar să știm să utilizăm mai mulți algoritmi. Din fericire există doar câteva categorii majore.

Componentele unui algoritm de învățare automată

A învăța înseamnă a realiza trei activități distincte dar legate între ele, și anume: **reprezentare**, **evaluare** și **optimizare** (Domingos, 2012). Fiecare dintre cele trei componente răspunde în ordine la una dintre următoarele trei întrebări: „Cum arată modelul?" (reprezentarea), „Cum putem distinge între modelele bune și cele proaste?" (evaluarea) și „Care este procesul prin care găsim modelele bune printre modelele posibile?" (optimizarea).

În cazul regresiei liniare, reprezentarea modelului nostru este o funcție liniară. Indiferent care vor fi valorile estimate ale coeficienților, modelul va fi întotdeauna o funcție liniară, deci nu va putea reprezenta corect date și relații non-liniare. Generalizând, atunci când alegem o anumită reprezentare pentru un algoritm de învățare automată, alegem de fapt un set de clasificatori pe care acesta îi poate învăța. Dacă un clasificator nu apare în spațiul modelelor permise / posibile (hypotheses space), el nu poate fi învățat. Desigur, pentru ca un computer să-l poată înțelege, orice model / clasificator ... trebuie să fie reprezentat într-un limbaj formal.

Evaluarea sau funcția de evaluare, numită și funcție cost sau funcție obiectiv (adică funcția pe care ne dorim să o reprezentăm), ne ajută să comparăm calitatea predicțiilor făcute de diferiți clasificatori / diferite modele, deci să-l alegem pe cel mai bun. De exemplu, în cazul unui model de clasificare, ne interesează să evaluăm diferența dintre clasa reală și clasa prezisă. În această fază evaluăm funcția cost asociată unor valori diferite ale parametrilor. Putem avea o funcție cost pentru algoritmul intern (de exemplu log loss) și alta pentru modelul extern (de exemplu acuratețea sau aria de sub curba ROC). Ultimul tip de funcție este numit adesea „măsură a performanței modelului". În acest context ne întrebăm dacă toate erorile de predicție au aceeași importanță relativă. De exemplu, ne putem întreba dacă o eroare de tip fals pozitiv ar trebui să conteze la fel de mult ca o eroare de tip fals negativ.

În continuare avem nevoie de un algoritm care să ne ajute să alegem clasificatorul / modelul cu cel mai bun scor, cea mai bună evaluare. Tehnica de optimizare este crucială pentru a obține un algoritm de învățare automată cât mai eficient. Suplimentar, în cazul în care funcția de evaluare are mai mult de un optim, optimizarea ne ajută să alegem soluția potrivită.

Relativ la fiecare componentă a unui algoritm de învățare putem alege una dintre mai multe variante. Important însă, unele combinații au sens, altele nu. De exemplu, o reprezentare de tip categorial poate fi aleasă doar împreună cu o optimizare de tip combinatoric. De obicei, textele de specialitate organizează capitolele în funcție

de reprezentare (ce algoritmi de învățare putem folosi pentru a reprezenta relațiile din date). Însă, alegerile cu privire la evaluare și optimizare sunt adesea la fel de importante, dacă nu chiar mai importante în anumite situații.

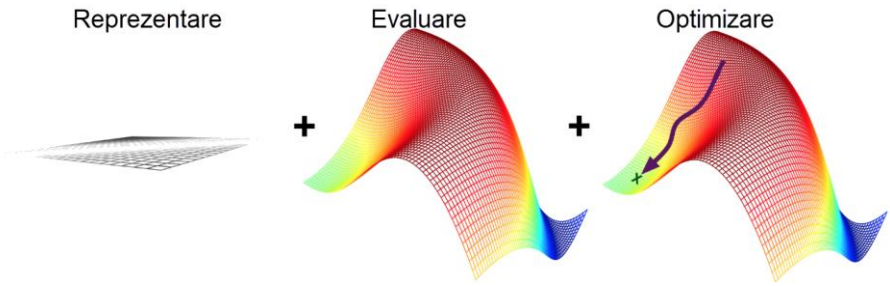
Figura 2.1-1. Cele trei componente ale unui algoritm de învățare automată

Representation	Evaluation	Optimization
Instances	Accuracy/Error rate	Combinatorial optimization
K-nearest neighbor	Precision and recall	Greedy search
Support vector machines	Squared error	Beam search
Hyperplanes	Likelihood	Branch-and-bound
Naive Bayes	Posterior probability	Continuous optimization
Logistic regression	Information gain	Unconstrained
Decision trees	K-L divergence	Gradient descent
Sets of rules	Cost/Utility	Conjugate gradient
Propositional rules	Margin	Quasi-Newton methods
Logic programs		Constrained
Neural networks		Linear programming
Graphical models		Quadratic programming
Bayesian networks		
Conditional random fields		

Sursa: (Domingos, 2012)

Pentru a oferi concretețe discuției, să considerăm un exemplu simplu de algoritm de învățare automată. Fie o rețea neuronală cu doar doi neuroni de input, deci doi coeficienți (weights) θ_1 θ_2 (Figura 2.1-2). Valorile posibile luate de acești coeficienți / parametri definesc un spațiu de căutare bidimensional (spațiul valorilor posibile sau spațiul ipotezelor). Hiperparametrii rețelei, adică arhitectura acesteia, numărul și tipul neuronilor, rata de învățare etc. sunt stabiliți de utilizator și sunt ficși pe perioada instruirii. Aceasta este componenta reprezentare. Pentru a calcula calitatea predicțiilor folosim o funcție cost. Funcția este estimată pe setul de instruire. Desigur, putem evalua doar anumite combinații ale parametrilor (coeficienților), deci nu putem observa întreaga suprafață a funcției. Aceasta este componenta evaluare. În continuare căutăm setul optim de valori ale parametrilor folosind unul dintre algoritmii potriviți, de exemplu „gradient descent”: $\theta_i^{nou} = \theta_i^{anterior} + \frac{L(f_\theta)}{\partial\theta_1}$. Aceasta este componenta optimizare.

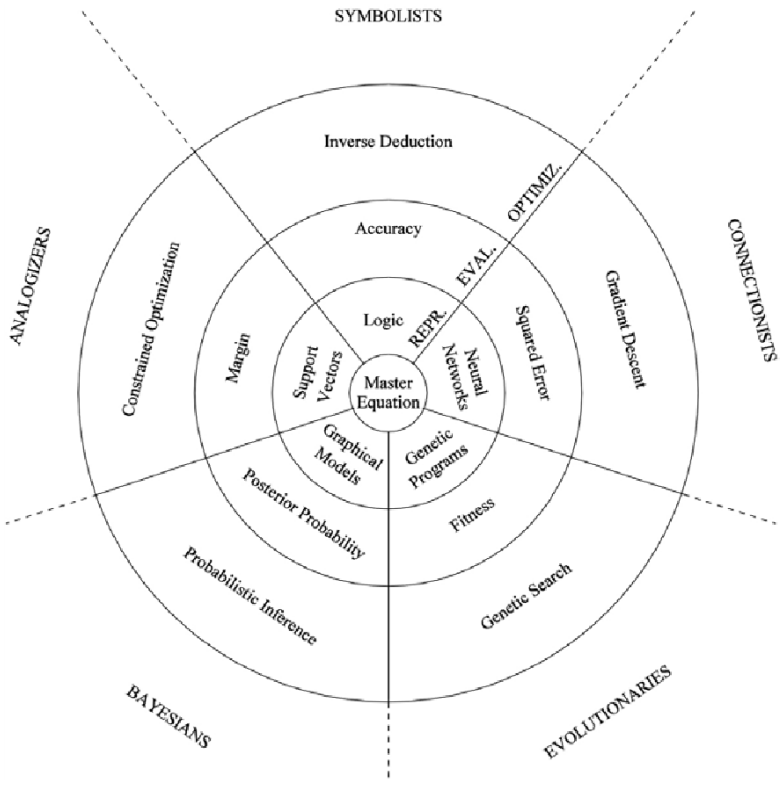
Figura 2.1-2. O reprezentare vizuală a componentelor unui algoritm de învățare automată în cazul unei rețele neuronale foarte simple



Sursa: Vanschoren, Joaquin. 2023. Machine learning for engineers

Dincolo de acest exemplu simplu, logica prezentată se aplică tuturor categoriilor majore de algoritmi și algoritmilor care formează o categorie (Figura 2.1-3).

Figura 2.1-3. Categoriile majore de algoritmi și exemple relativ la fiecare componentă

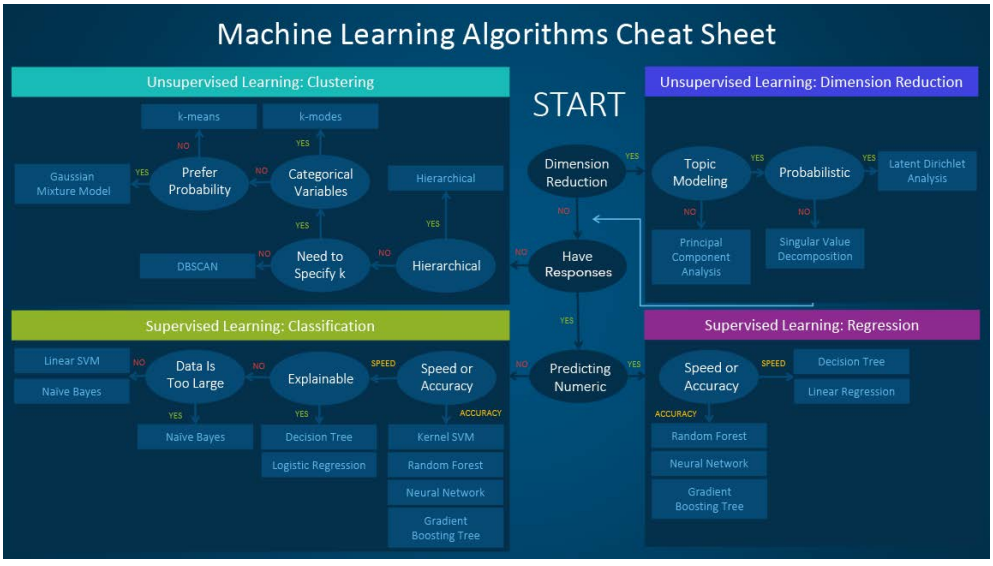


Sursa: (Domingos, 2015, p. 240)

O clasificare a algoritmilor de învățare automată

Cu referire la diferitele tipuri de analiză a datelor, literatura de specialitate din domeniul statisticii distinge adesea între modele / tehnici de dependență și co/inter-dependență (Culic, 2004). În esență, distincția dintre cele două tipuri de modele ține de prezența sau nu a unei variabile dependente, alături de alte variabile (independente). Evident, modelele de dependență sunt cele care includ o variabilă dependentă, respectiv urmăresc să analizeze variația acesteia în funcție de variația variabilelor independente. Similar cu această clasificare, în știința datelor (data mining, machine learning) distincția relevantă este învățare asistată vs. neasistată, respectiv învățare supervizată vs. nesupervizată (Foster et al., 2021, p. 151). În acest context, învățarea asistată sau supervizată se suprapune (doar parțial) cu modelele de dependență, iar învățarea neasistată sau nesupervizată cu modelele de interdependență.

Figura 2.1-4. O clasificare a algoritmilor de învățare automată: nesupervizată vs. supervizată



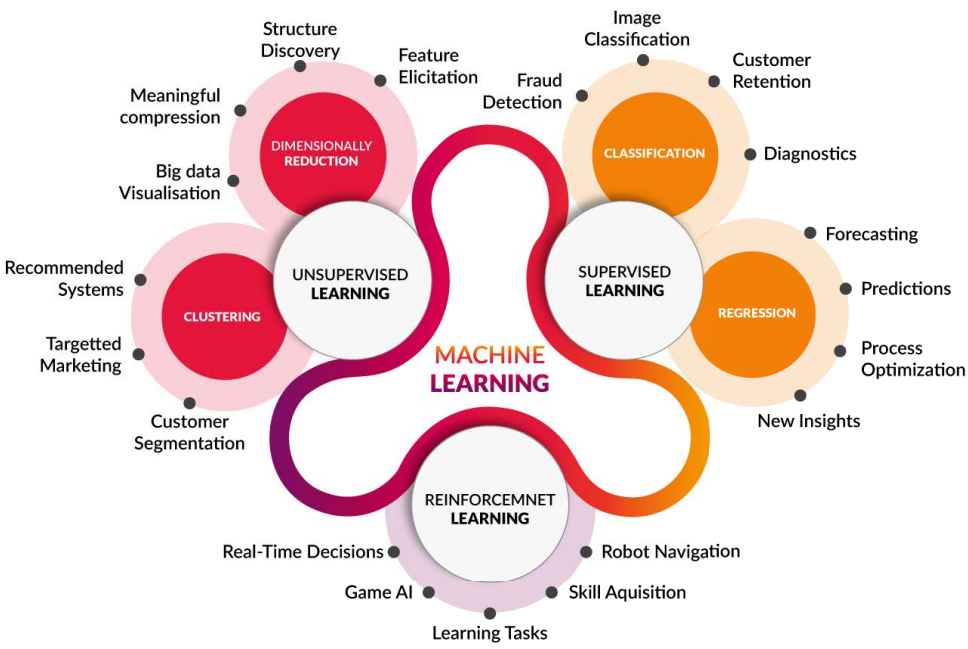
Sursa: Hui Li. 2020. Which machine learning algorithm should I use?

În funcție de care este obiectivul concret al analizei (parțial și de tipul datelor), putem alege unul dintre drumurile indicate în cele patru situații din Figura 2.1-4:

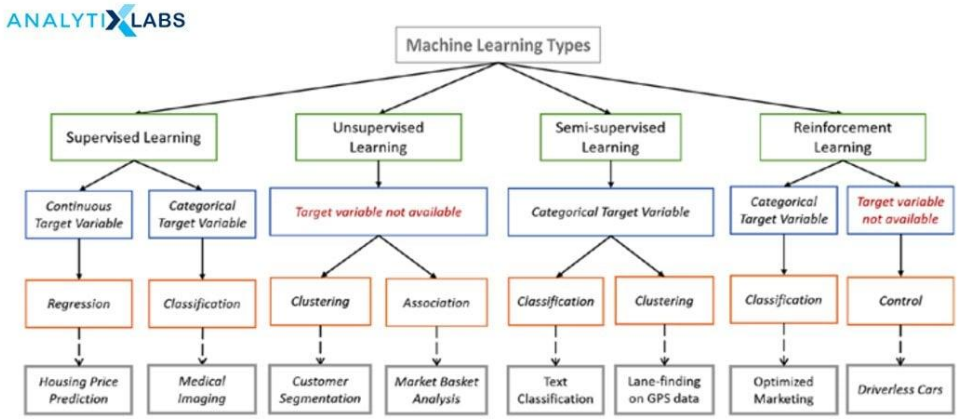
(1) învățare asistată pentru dependente metrice (Regression), (2) învățare asistată pentru dependente categoriale (Classification), (3) învățare neasistată cu scopul reducerii dimensiunilor (Dimension Reduction) și (4) învățare neasistată cu scopul grupării (Clustering). Unii dintre algoritmii de învățare asistată (predictivi) pot gestiona ambele tipuri de probleme (clasificare și regresie): rețelele neuronale, arborii de decizie, mașinile cu suport vectorial. Alții algoritmi pot gestiona fie probleme de clasificare (regresia logistică, cei mai apropiați k vecini, naive bayes), fie probleme de regresie (regresia liniară). În categoria algoritmilor care își propun să grupeze cazurile se află k-Means, DBSCAN și Gaussian Mixture Model, iar în categoria algoritmilor care își propun să reducă numărul variabilelor se află analiza componentelor principale, descompunerea în valori unice, analiza latentă Dirichlet.

Desigur, există și alte clasificări ale algoritmilor (Figura 2.1-5). Una dintre acestea include o nouă categorie numită Reinforcement Learning, adică învățarea bazată pe recompensarea comportamentelor dezirabile și pedepsirea celor indezirabile. O altă clasificare introduce o nouă categorie de algoritmi numită învățare semi-asistată, situată între învățarea asistată și cea neasistată.

Figura 2.1-5. Alte clasificări ale algoritmilor de învățare automată



Sursa: Ankit Gupta. 2017. Introduction to Machine Learning

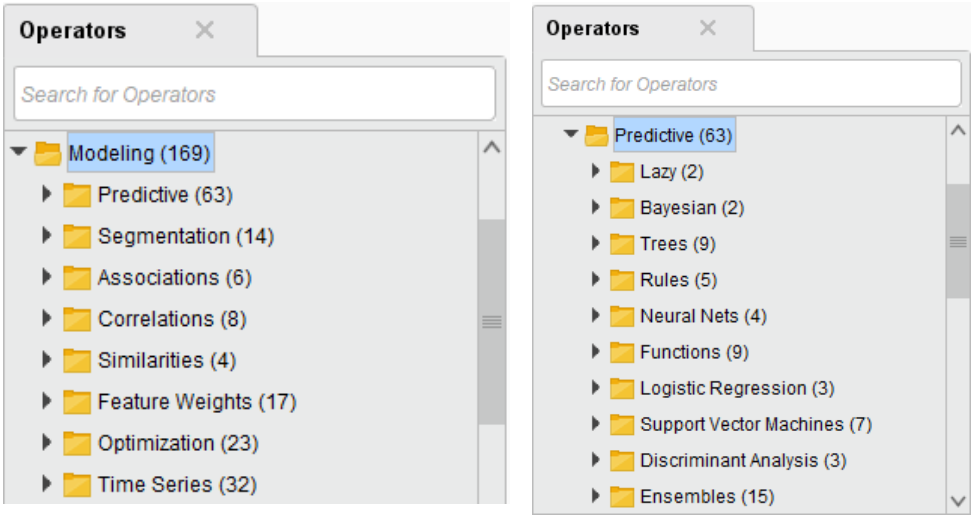


Sursa: Analytix Labs. 2020. Different Types of Machine Learning Algorithms

Organizarea algoritmilor de învățare automată în RapidMiner (Modeling)

Tabul Modeling din fereastra Operators conține operatorii utili pentru modelarea datelor, grupați, în funcție de principalul obiectiv urmărit, în șase categorii majore (Figura 2.1-6): Predicție, Segmentare, Asociere, Corelație, Analiza similarității și Serii de timp.

Figura 2.1-6. Operatorii din categoria Modeling și sub-categoria Predictive



Alături de acestea, apar alte două categorii, Importanța Atributelor și Optimizarea, operatorii incluși aici având o utilitate transversală. Gruparea operatorilor de modelare (algoritmilor) în aceste categorii este una mai degrabă fundamentată practic, cu scopul de a ușura identificarea rapidă a operatorului potrivit într-o situație particulară de analiză. Gruparea este realizată oarecum prin combinarea a trei criterii de clasificare: tipul de sarcină (obiectivul modelării), tipul de date (serii de timp vs. altele) și scop (modelare vs. îmbunătățirea modelării).

Dacă avem în vedere strict obiectivul modelării, putem distinge între algoritmi care rezolvă sarcini de clasificare, regresie, grupare (clusterizare), asociere & corelație și detectare a anomaliilor. Fiecare dintre aceste sarcini vizează întrebări specifice de cercetare:

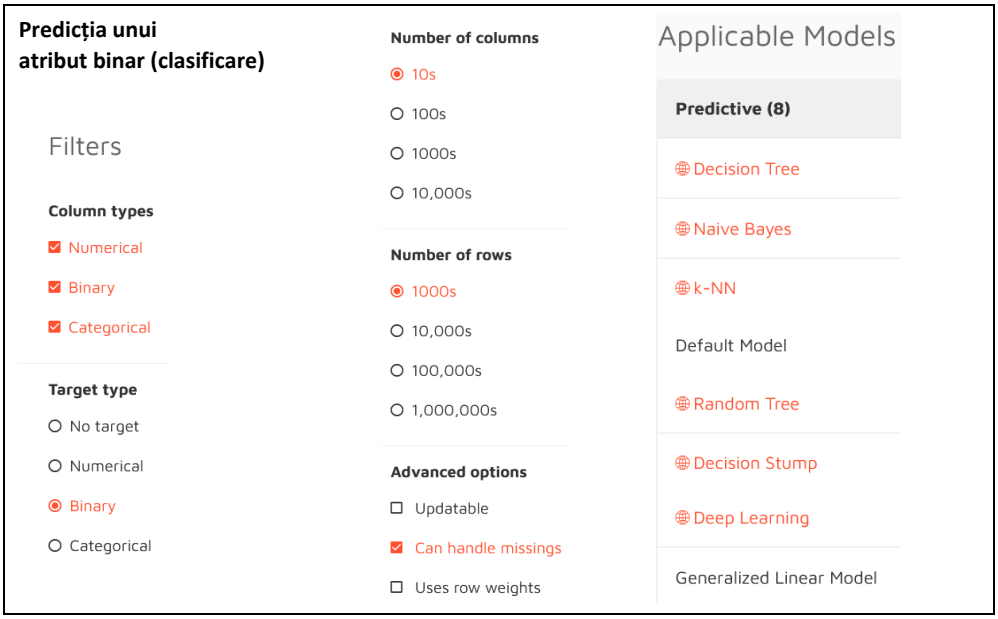
- **Clasificarea:** Este acesta A sau B? Va fi acesta A sau B? Cum putem să creștem / reducem probabilitatea să fie A sau B?
- **Regresia:** Cât de mult sau câți / câte sunt și / sau vor fi?
- **Gruparea:** Cum sunt organizate cazurile? Care sunt diferite? Dar similare?
- **Asocierea & Corelația:** Ce se întâmplă concomitent? Ce se schimbă împreună și cum?
- **Detectarea anomaliilor:** Este acest lucru ciudat, foarte puțin probabil?

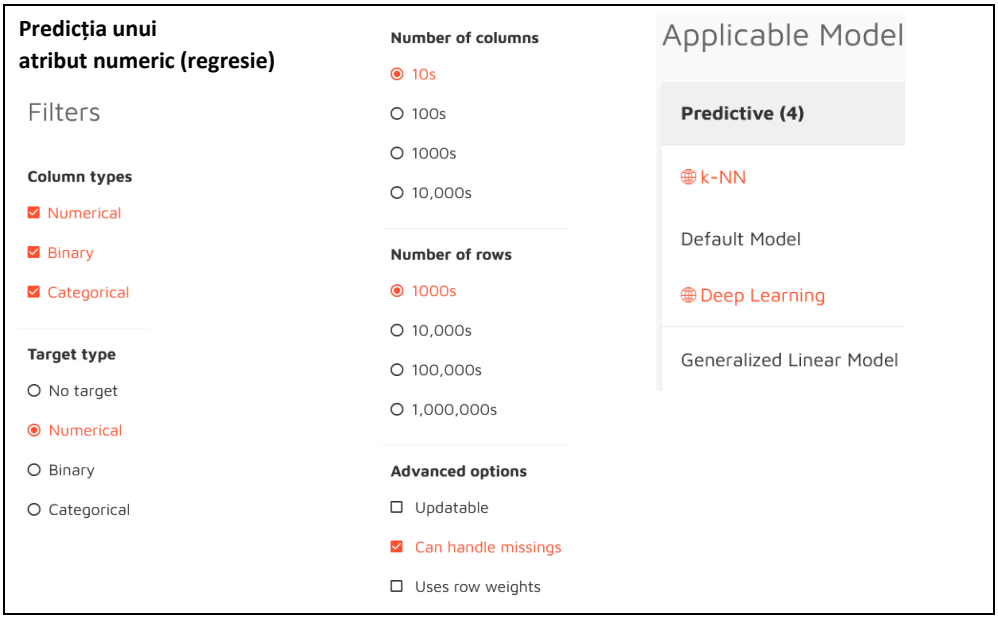
Identificarea algoritmilor de învățare potriviți într-o situație specifică

RapidMiner Studio include un număr foarte mare de algoritmi (o parte dintre aceștia provin din softul Weka și sunt incluși automat în RapidMiner, alții pot fi accesați cu ajutorul operatorilor RapidMiner care comunică cu Python, respectiv R). În funcție de tipul de date disponibile și de scopul analizei pot fi folosiți doar unii dintre aceștia. Pentru a identifica rapid care sunt algoritmii potriviți în cazul unei analize specifice, putem folosi o aplicație online oferită de RapidMiner (poate fi accesată la adresa https://mod.rapidminer.com/#app). În Figura 2.1-7 am ilustrat utilizarea aplicației în două situații aproape identice – singura diferență constă în tipul atributului pe care dorim să-l prezicem (target): categorial (cu două sau mai multe categorii) sau numeric. În cazul celorlalte filtre de selecție am considerat următoarele:

- setul de date conține atribute (column types) de tip numeric, binar (două categorii) și categorial (trei sau mai multe categorii);
- numărul de atribute este mic (≤10);
- numărul de cazuri este mic (≤1000);
- algoritmul poate lucra cu atribute care au valori lipsă.

Figura 2.1-7. Identificarea algoritmilor potriviți într-o situație specifică





Link aplicație: https://mod.rapidminer.com/#app

În cazul celor două situații specifice descrise anterior putem utiliza doar anumiți algoritmi. Astfel, dacă variabila dependentă (target) este binară, putem alege unul

dintre următorii algoritmi predictivi de clasificare (doar pe acestea le-am afișat în Figura 2.1-7): Decision Tree, Naive Bayes, k-NN, Default Model, Random Tree, Decision Stump, Deep Learning și Generalized Linear Model. Dacă variabila dependentă este de tip numeric avem relativ mai puține opțiuni: k-NN, Default Model, Deep Learning și Generalized Linear Model. Desigur, funcție de caracteristicile setului de date și de scopul urmărit, opțiunile pot fi setate mai specific, paleta alegerilor modificându-se.

2.2. Algoritmii de clasificare prezentați în volum

În cadrul acestui volum suntem interesați doar de algoritmii care rezolvă probleme de predicție (Predictive, Figura 2.1-6). Algoritmii din categoria Predictive sunt folosiți pentru a construi modele de predicție a unor atribute metrice și / sau non-metrice (categoriale). Unii algoritmi pot fi folosiți pentru a face predicții indiferent care este nivelul de măsurare al atributului de interes, alții doar în cazul unor atribute de tip categorial, respectiv metric. În acest volum, dintre toți algoritmii de predicție, vom discuta doar despre cei care vizează probleme de clasificare, adică despre algoritmii care prezic apartenența cazurilor la clasele unei variabile categoriale (variabila dependentă este non-metrică / categorială), deci la algoritmii care urmăresc clasificarea cazurilor. Desigur, vom discuta doar câțiva dintre algoritmii care intră în această categorie.

În viața obișnuită fiecare dintre noi rezolvăm probleme de clasificarea destul de des. Rezolvarea unor probleme de clasificare este importantă și pentru o mulțime de activități din domeniul public și privat. Pentru a le rezolva putem folosi datele și algoritmii potriviți. Câteva exemple de astfel de probleme sunt următoarele:

- clasificarea unui email ca spam sau non-spam;
- stabilirea unui diagnostic (pacientul este bolnav sau sănătos) sau a rezultatului unei analize medicale (pozitiv sau negativ);
- clasificarea unei tranzacții financiare ca normală sau fraudulentă;
- identificarea angajaților care vor pleca din companie în următorul an (respectiv a celor care vor rămâne).

În cadrul volumului vom folosi sistematic acest ultim tip de problemă de clasificare. Exemplul preferat se referă tocmai la nevoia de a identifica angajații care au șanse mai mari să plece din companie. Vom relua acest exemplu în contextul prezentării fiecărui algoritm, la diferite nivele de complexitate a modelării.

Clasificarea algoritmilor de clasificare

Algoritmii de clasificare pot fi grupați în funcție de diferite criterii. Una dintre tipologii folosește drept criteriu de clasificare forma asumată a relației dintre atribute și distinge între algoritmi de tip liniar, non-liniar și bazați pe reguli (Wendler & Gröttrup, 2021, p. 759). Algoritmii (non-)liniari exprimă relațiile dintre atribute sub formă matematică (funcții) și, cel mai adesea, transformă datele (le normalizează sau standardizează). În cazul algoritmilor non-liniari funcțiile sunt relativ mai complexe. Algoritmii bazați pe reguli generează un set de afirmații de tip „dacă x atunci y" folosind datele originale, netransformate.

O altă tipologie distinge între algoritmi bazați pe probabilitate, distanță, optimizare și căutare (Moreira et al., 2019). Algoritmii bazați pe distanță stabilesc clasa la care aparține un caz în funcție de distanța dintre acesta și cazurile similare, din vecinătate (k-NN este exemplul tipic). Algoritmii bazați pe probabilitate stabilesc clasa la care aparține un caz în funcție de probabilitatea estimată. Pentru a estima probabilitățile e nevoie de un număr foarte mare de cazuri, fiind posibil doar să aproximăm valorile parametrilor. Aproximarea se poate face generativ (estimează probabilitatea ca un caz cu caracteristicile X să aparțină clasei C_1; Logistic Regression) sau discriminativ (estimează probabilitatea clasei C_1, respectiv probabilitatea ca un caz din clasa C_1 să fie cu caracteristicile X; Naive Bayes). Algoritmii bazați pe optimizare stabilesc clasa la care aparține un caz încercând să optimizeze o funcție, adică să găsească valorile parametrilor care produc cea mai mică / mare valoarea a acestei funcții (exemplele tipice sunt Neural Networks și Support Vector Machines). Algoritmii bazați pe căutare urmăresc să identifice atributele, respectiv valorile asociate acestora, care reduc relativ cel mai mult eroarea de clasificare a cazurilor (exemplul tipic este Decision Tree).

O a treia tipologie, mai simplă, distinge între două tipuri mari de algoritmi: unii algoritmi produc aceleași reprezentări ale datelor, cu aceeași complexitate, indiferent cât de mult crește numărul cazurilor (fixed-size learners), alții produc reprezentări mai complexe atunci când numărul cazurilor crește (variable-size

learners) (Domingos, 2012).[1] În prima categorie intră clasificatorii liniari precum regresia logistică, iar în a doua clasificatorii bazați pe reguli precum Decision Tree. În principiu, clasificatorii cu o mărime variabilă a reprezentării pot învăța orice funcție dacă au suficiente date, dar în practică lucrurile nu stau așa datorită unor factori precum costul computațional, optimul local sau numărul prea mare al dimensiunilor.

Pornind de la aceste clasificări, prezentăm în acest volum cinci algoritmi, câte unul sau doi din fiecare tip. Selecția acestora, alături de categoria la care aparțin, apare în Tabelul 2.2-1.

Tabelul 2.2-1. Algoritmii de clasificare prezentați în volum

Tip decizie	Liniar	Liniar	Non-liniar	Non-liniar	Reguli (if-then)
Bazat pe ...	**Probabilitate**	**Non-liniar Probabilitate**	**Distanță**	**Optimizare**	**Căutare**
Denumire algoritm	Regresie logistică	Bayes naiv	Cei mai apropiați k vecini	Rețea neuronală	Arbore decizional
Operator RapidMiner	Logistic Regression	Naive Bayes	k-NN	Neural Network	Decision Tree

Sursa: (Moreira et al., 2019; Wendler & Gröttrup, 2021, p. 759).

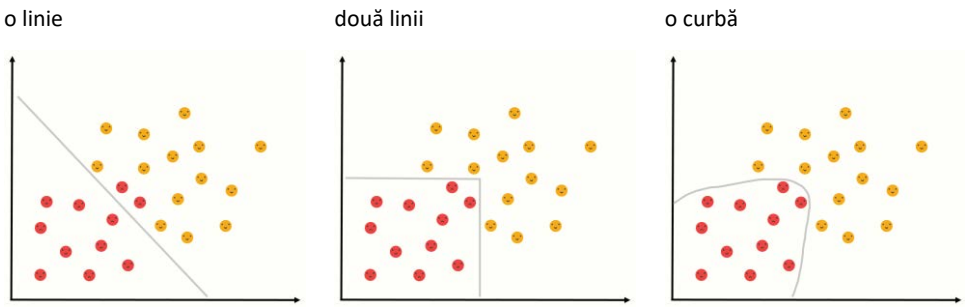
Cum învață algoritmii de clasificare? O reprezentare vizuală

Orice model de clasificare urmărește să identifice o funcție care descrie o linie sau o curbă (o combinație de linii și curbe) ce separă cât mai bine clasele atributului de interes. Linia / curba respectivă reprezintă ceea ce se numește frontiera dintre clase sau frontiera deciziei (decision boundary). Modelul va prezice că observațiile care sunt de o parte a frontierei vor aparține unei clase, restul celeilalte clase (presupunând că atributul de interes are doar două clase). În Figura 2.2-1 am prezentat trei exemple simple de frontiere. Desigur, frontierele pot lua forme dintre cele mai diverse, mai ales dacă numărul predictorilor și claselor este mai mare de

[1] Același autor a propus și o tipologie cu cinci tipuri de algoritmi: Symbolists, Connectionists, Evolutionaries, Bayesians și Analogizers. Fiecare din aceste „triburi" are la bază un set de credințe majore (asumpții), respectiv este interesat în principal de un anumit tip de problemă (Domingos, 2015).

doi, iar relațiile dintre atribute sunt mai complexe. Chiar și în aceste condiții, teoretic putem defini frontiere suficient de complexe încât să separe perfect datele existente, deci să clasifice cazurile fără nicio eroare. Însă acest lucru nu este de dorit. Procedând astfel vom include în model și „zgomotul" din date, deci doar vom avea iluzia că am produs un model perfect. Atunci când modelul va fi confruntat cu date noi, acesta va produce multe erori de clasificare. Prin urmare e de preferat să folosim frontiere mai complexe, dar doar până la un anumit nivel de complexitate (Wendler & Gröttrup, 2021, p. 763).

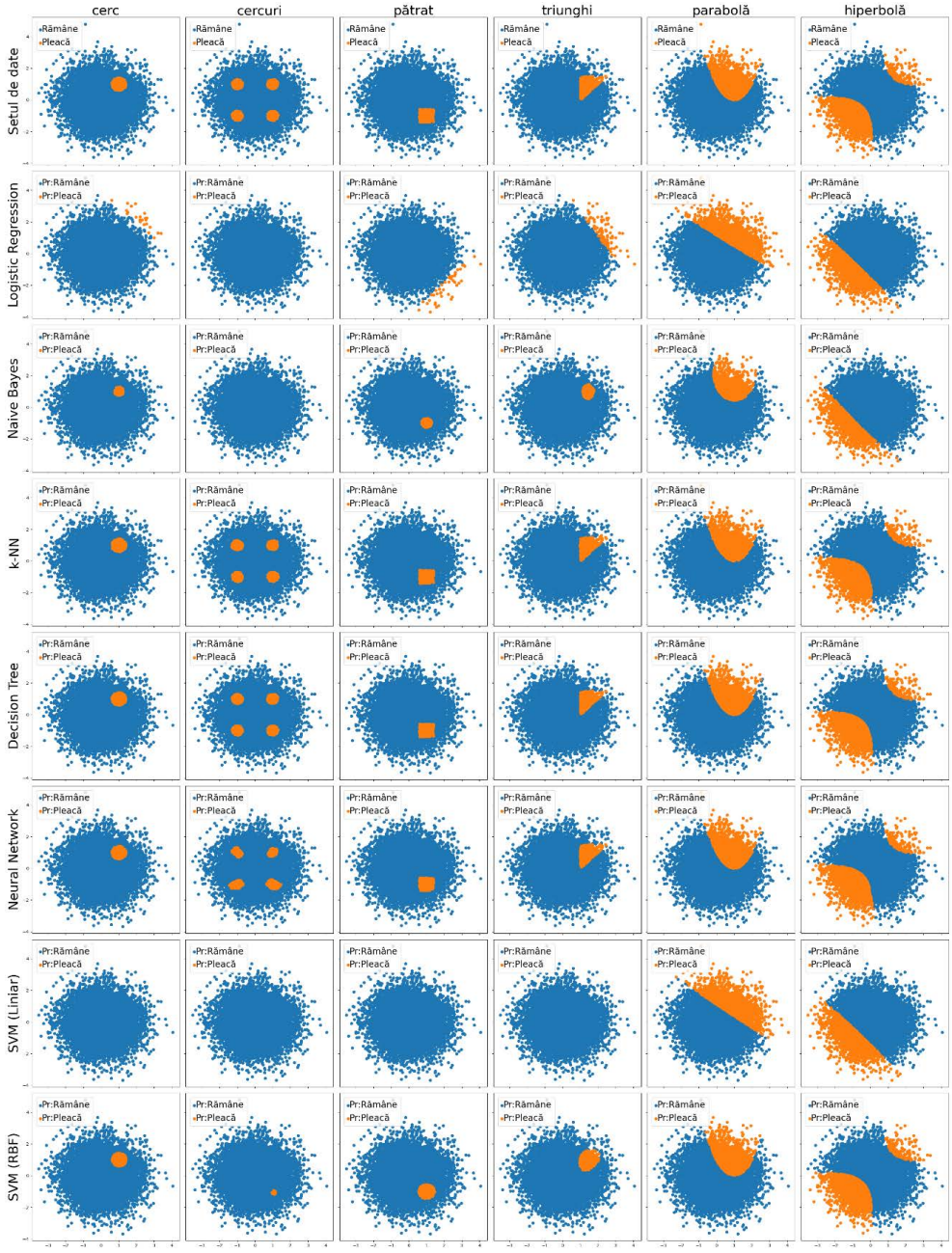
Figura 2.2-1. Trei exemple simple de frontiere între două clase



În toate exemplele vizuale care urmează vom considera că setul de date include un atribut de interes cu două clase și doi predictori metrici. În toate aceste exemple datele sunt simulate, adică știm care sunt funcțiile care au produs aceste date (Data Generation Process). Faptul că știm exact cum arată realitatea, înainte ca procesul de învățare să aibă loc, ne ajută să evaluăm, pentru acest exemplu doar intuitiv, performanța diferiților algoritmi de clasificare.

În Figura 2.2-2 realitatea ia una dintre formele care apar în graficele de pe prima linie. Punctele portocalii îi reprezintă pe angajații care părăsesc compania, iar cele albastre pe cei care rămân. Setul de date conține doi predictori metrici (cele două dimensiuni / axe). Relația dintre cei doi predictori și plecarea din companie ia forme relativ simple: cerc, cercuri, pătrat, triunghi, parabolă sau hiperbolă. Fiecare dintre următoarele linii cu grafice ilustrează capacitatea fiecărui algoritm de clasificare de a învăța (descoperi) relațiile din datele respective. Observăm că performanța este destul de variabilă și că doar unii dintre algoritmi au capacitatea de a învăța (descoperi) toate „realitățile" simulate (k-NN, Decision Tree, Neural Network).

Figura 2.2-2. Capacitatea algoritmilor de clasificare de a învăța diferite „realități”

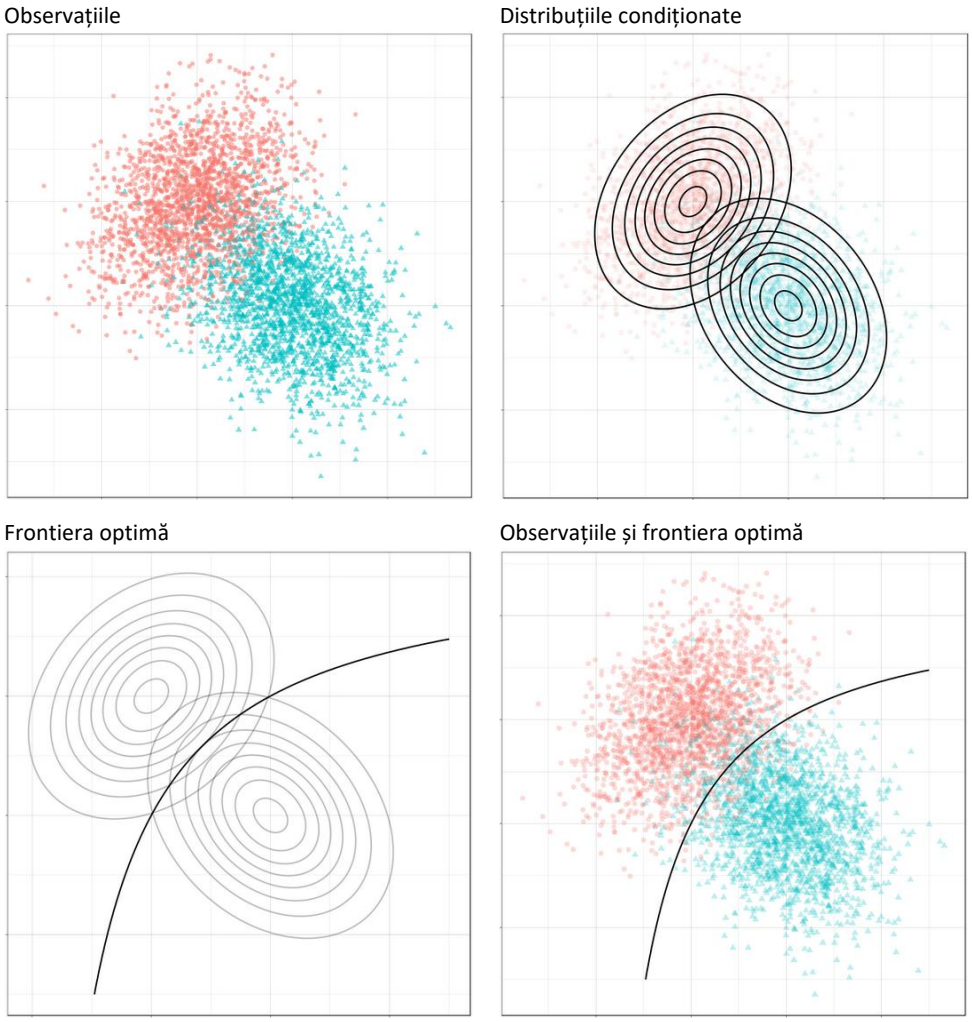


Sursa: Exemplul este o variantă modificată a analizei Stirring The Pile / Stirring the pile effectively

În Figura 2.2-3, în partea stângă sus, apare distribuția cazurilor care compun un alt posibil de set de date. Intuitiv, cele două clase sunt relativ bine separate cu ajutorul

predictorilor, fapt ilustrat și de distribuțiile de densitate (dreapta sus). În acest caz frontiera optimă care separă cele două clase este o curbă simplă (graficele de jos). Din nou, ne întrebăm în ce măsură această frontieră poate fi descoperită, adică învățată sau aproximată, de către diferiți algoritmi. Reamintim faptul că, în realitate, frontiera optimă nu este cunoscută niciodată.

Figura 2.2-3. Frontiera optimă în cazul unor relații simple



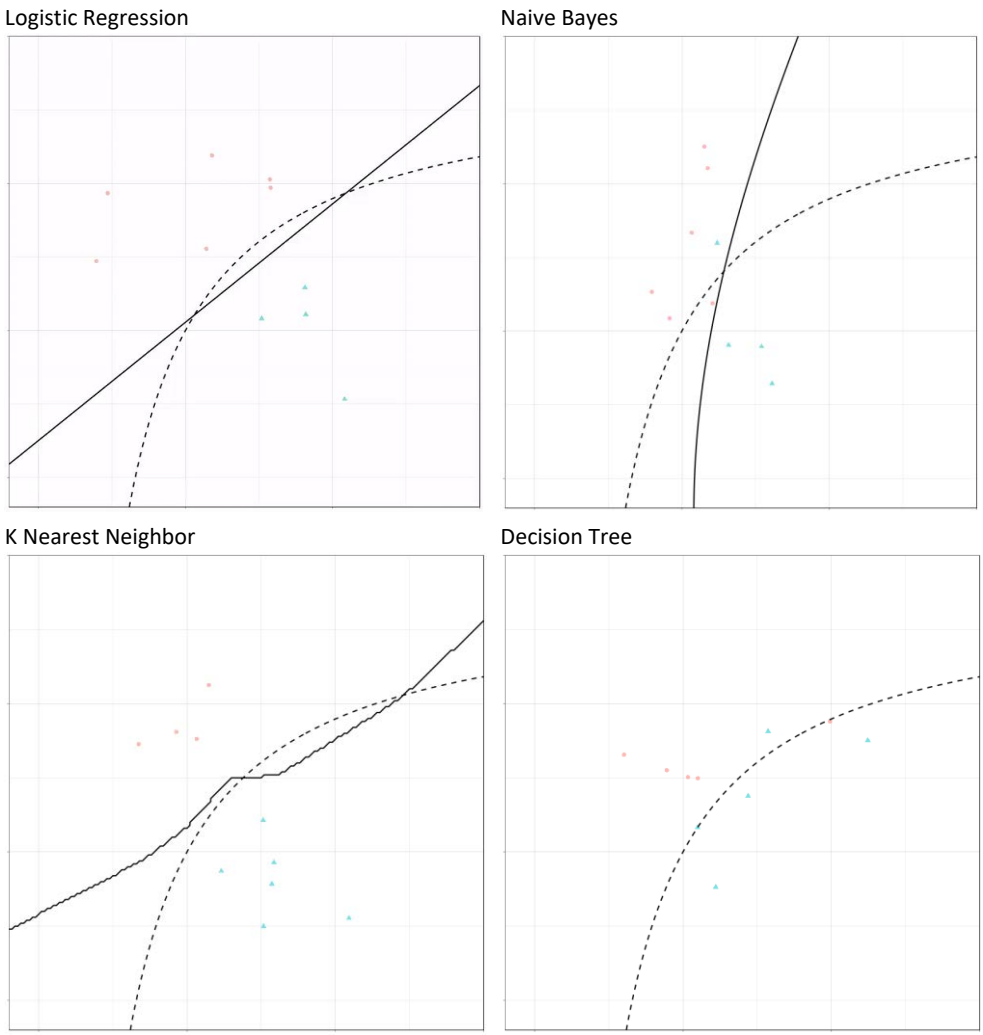
Sursa: Optimal Decision Boundaries; două clase și doi predictori metrici; observațiile asociate unei clase provin dintr-o distribuție normală multivariată.

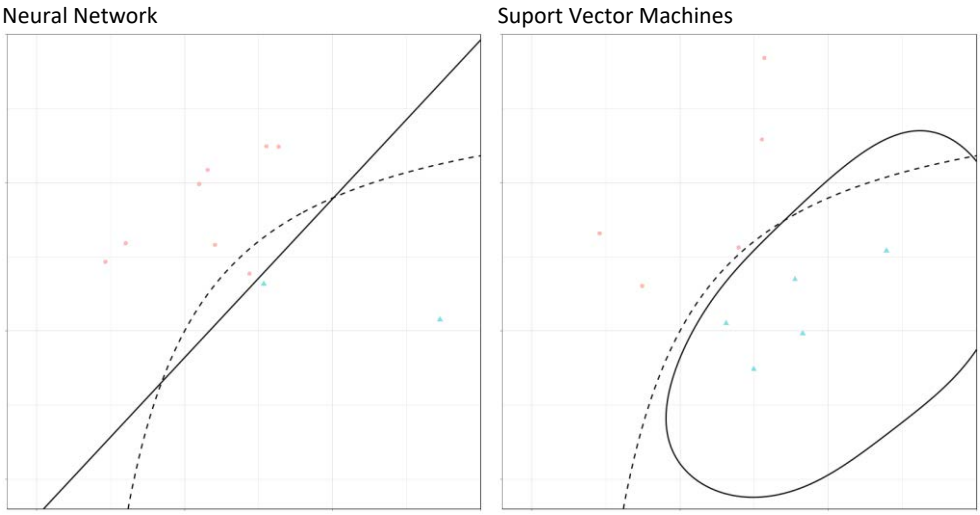
Pentru a aproxima frontiera optimă folosim un model de clasificare bazat pe unul sau mai mulți algoritmi. Dat fiind faptul că un algoritm are la bază asumpții specifice, gradul

de corectitudine a aproximării depinde de gradul de suprapunere dintre asumpțiile algoritmului și realitate (o suprapunere mai mare va crește performanța aproximării).

În Figura 2.2-4 am prezentat comparativ frontiera optimă și frontiera estimată de diferiți algoritmi (imaginile prezintă și procesul de estimare, de la stadiul inițial la cel final). Observăm că suprapunerea dintre frontiera reală și cea estimată variază destul de mult. Dat fiind faptul că frontiera reală este o curbă, algoritmii care asumă că relațiile sunt liniare (regresia logistică, naive bayes) produc frontiere relativ mai puțin corecte. Important, în general, frontierele inițiale și cele finale diferă destul de mult, atât la nivelul unui singur algoritm cât și între algoritmi.

Figura 2.2-4. Aproximarea frontierei optime în cazul unor relații simple



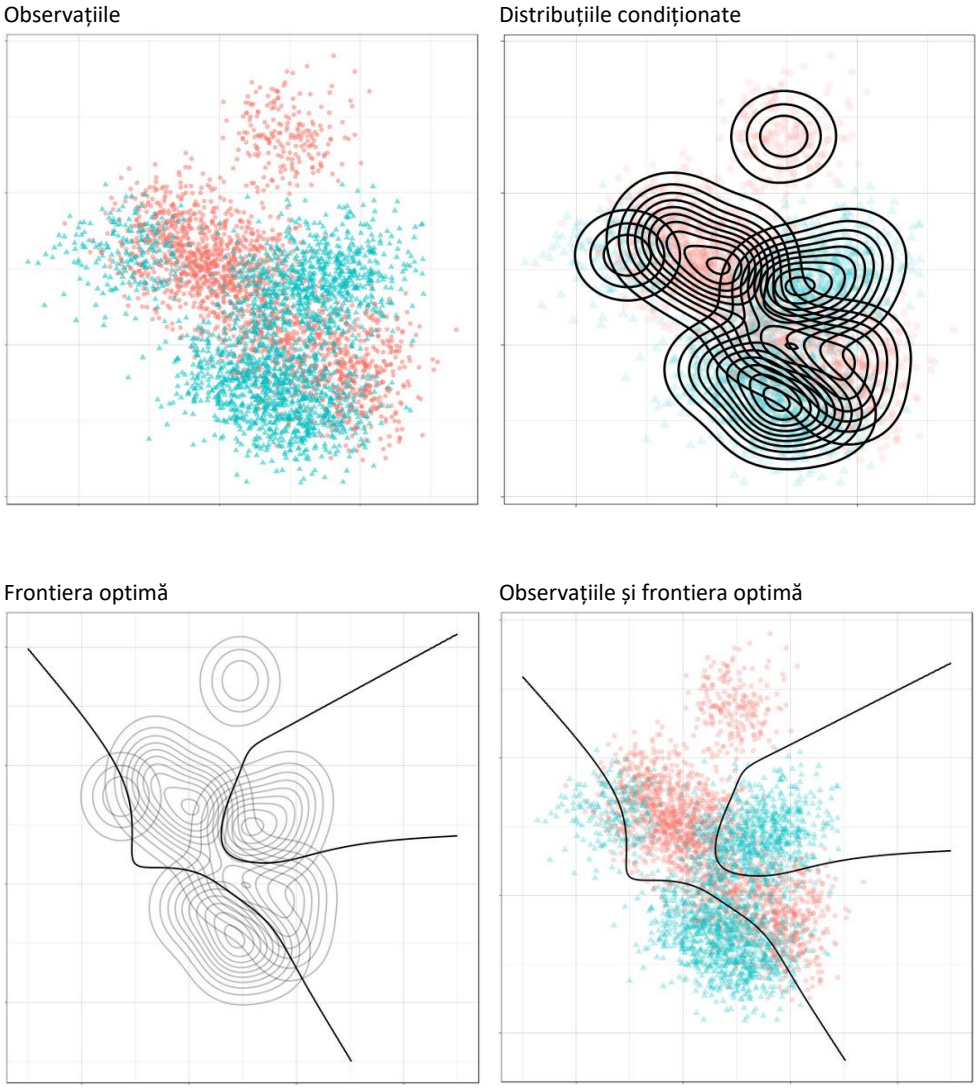


Sursa: Learning Decision Boundaries

În exemplul anterior relațiile dintre variabile au fost relativ simple. În lumea reală o astfel de situație este mai degrabă o excepție. Firesc, ne întrebăm cum arată lucrurile în cazul unor relații relativ mai complexe. Complexitatea poate să apară în relație cu aspecte precum distribuția variabilelor (realitatea / datele sunt produse de o singură distribuție vs. mai multe; distribuția este normală vs. non-normală), forma relației (liniară vs. non-liniară) și variația intensității relației în funcție de valorile luate de variabile și / sau de prezența / absența altor variabile (efecte de moderare, mediere). Pentru a putea utiliza același fel de reprezentări vizuale, vom considera tot o situație cu doi predictori și un atribut de interes cu două clase. Însă, în acest caz, observațiile unei clase nu mai sunt produse de o singură distribuție normală ci de zece (Figura 2.2-5). Desigur, în realitate, complexitatea poate crește și relativ la aspecte precum numărul de predictori și interacțiunile dintre aceștia.

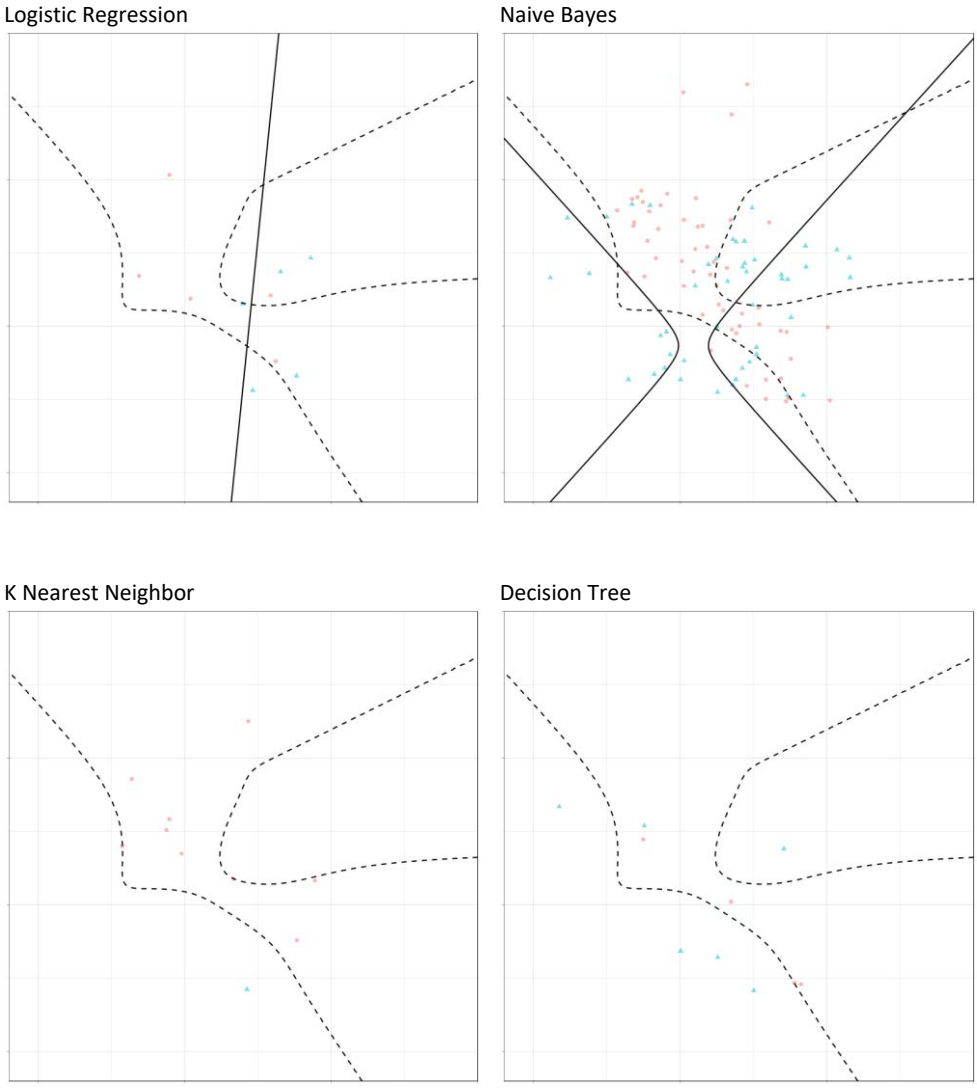
În cazul unor date de acest tip frontiera dintre clase ia o formă relativ mai complexă, deci poate fi descoperită mai greu. Și în acest caz algoritmii care asumă că relațiile sunt liniare (regresia logistică, naive bayes) produc frontiere mai degrabă greșite, în timp ce restul algoritmilor se descurcă mult mai bine (Figura 2.2-6).

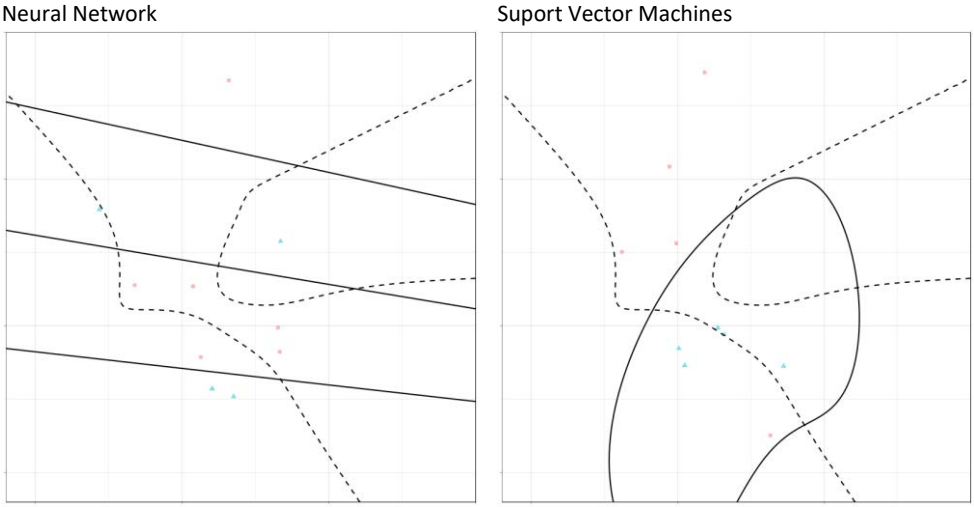
Figura 2.2-5. Frontiera optimă în cazul unor relații relativ mai complexe



Sursa: Optimal Decision Boundaries; două clase și doi predictori metrici; fiecare observație dintr-o clasă provine din una din cele zece distribuții normale posibile.

Figura 2.2-6. Aproximarea frontierei optime în cazul unor relații complexe





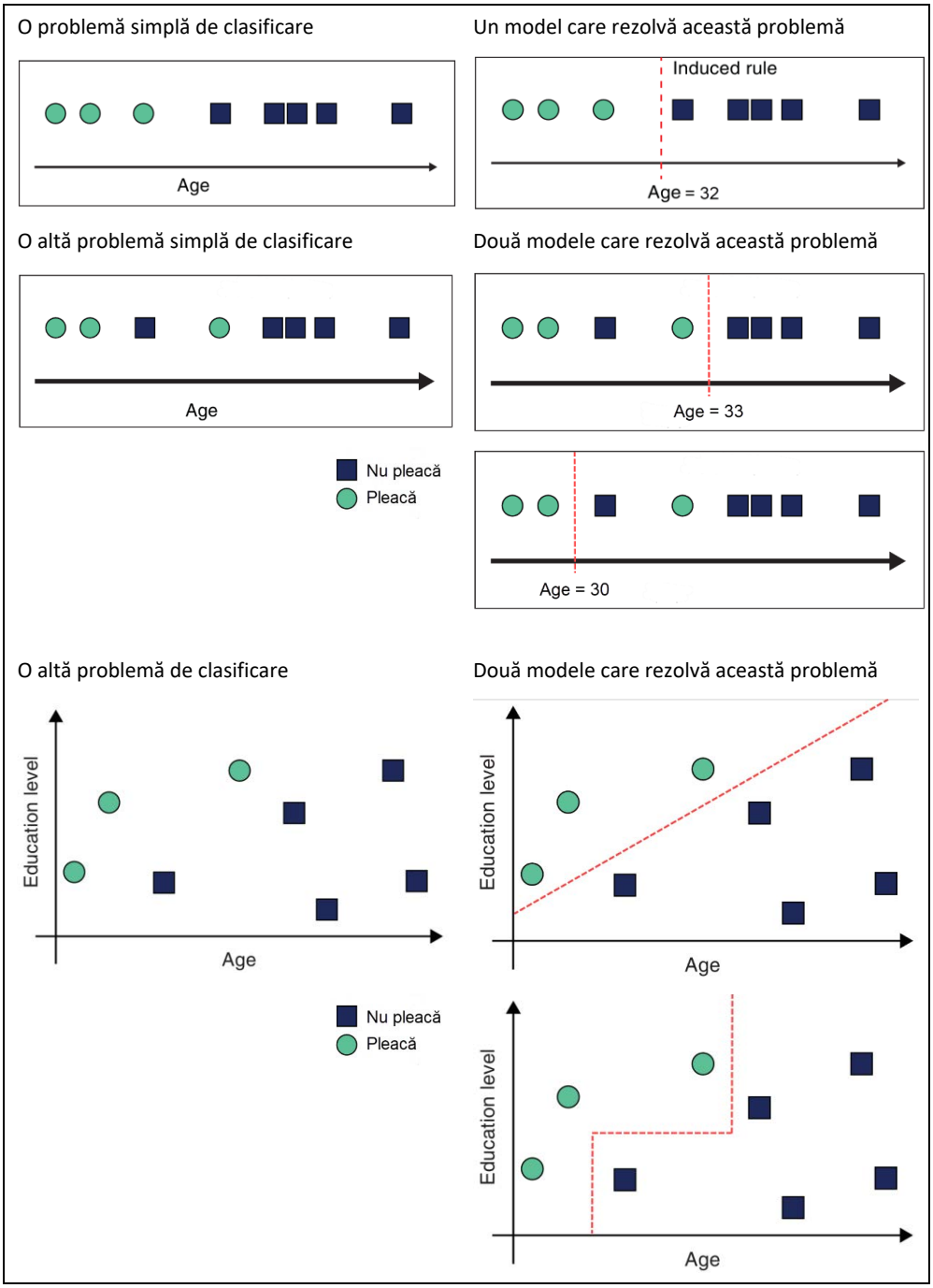
Sursa: Learning Decision Boundaries

2.3. Logica generală de construire a unui model de clasificare

Să presupunem că ne interesează să știm care sunt angajații care vor părăsi compania. Pentru a atinge acest obiectiv ne uităm la caracteristicile foștilor și actualilor angajați. Observăm că din opt angajați, cinci sunt angajați și în prezent, iar trei nu mai sunt. Pentru acest exemplu presupunem că știm suplimentar doar vârsta angajaților. Aceste informații sunt reprezentate grafic în Figura 2.3-1 (stânga sus). Ne interesează să găsim o regulă care să ne permită să distingem între angajații care pleacă și cei care nu pleacă. Observăm că toți angajații care pleacă au vârsta mai mică de 32 ani, iar toți cei care rămân au cel puțin 32 de ani. Folosind această regulă putem construi un model de clasificare („classifier" = clasificator) foarte simplu pe care îl putem folosi pentru a prezice dacă un angajat pleacă sau nu. Astfel, vom prezice că un angajat nou în vârstă de cel mult 31 ani va pleca, respectiv că un angajat nou în vârstă de cel puțin 32 ani nu va pleca din companie.

Cel mai adesea, relațiile nu sunt atât de simple și clare, fiind posibil să existe și angajați tineri care nu pleacă, respectiv angajați mai în vârstă care pleacă (vezi a doua problemă de clasificare din Figura 2.3-1). Regula descoperită anterior nu ne mai ajută să distingem la fel de bine între cele două tipuri de angajați, deci va trebui să căutăm alte reguli precum „angajații sub 30 ani pleacă" sau „angajații de cel puțin 33 ani rămân". Folosind una dintre aceste reguli vom putea clasifica mai bine angajații, însă nu perfect. Indiferent pe care din cele două reguli o folosim, vom clasifica greșit un angajat.

Figura 2.3-1. Exemple de probleme de clasificare și de modele care rezolvă aceste probleme



Sursa: Exemple inspirate din (Moreira et al., 2019)

O opțiune mai bună este să colectăm și alte date relativ la angajați. De exemplu, dacă știm și care este nivelul lor de educație, problema de clasificare se mută dintr-un spațiu unidimensional în unul bidimensional. Observăm că acum putem separa din nou perfect cele două categorii de angajați, doar că de această dată folosim o dreaptă. Dreapta respectivă este descrisă de o ecuație care sintetizează relația dintre vârstă și educație. Angajații care sunt deasupra dreptei sunt angajații care pleacă, iar cei care sunt sub dreaptă sunt angajații care rămân. Tocmai am construit un nou model de clasificare care ne ajută să distingem între angajații care pleacă și cei care rămân.

Desigur, această dreaptă nu este singura dreaptă posibilă (o altă dreaptă, situată puțin mai sus / jos sau cu o pantă puțin diferită, separă la fel de bine cazurile). Mai mult, putem separa spațial angajații care pleacă de cei care rămân și altfel decât printr-o dreaptă. O posibilitate diferită (un alt model de clasificare) este descrisă de linia frântă din Figura 2.3-1 (dreapta jos). Raționamentul prezentat anterior relativ la situațiile cu una, respectiv două dimensiuni poate fi extins și în cazul în care avem trei sau mai multe dimensiuni.

În cazul acestor exemple didactice cu doar două atribute, opt cazuri și separare (aproape) completă între clase, am găsit regula rapid, simplu, fără calcule. Desigur, în lumea reală, cea în care numărul de clase și atribute este mult mai mare, respectiv frontierele dintre clase sunt mult mai complexe, vom utiliza diferiți algoritmi pentru a construi un model de clasificare și a clasifica cazurile.

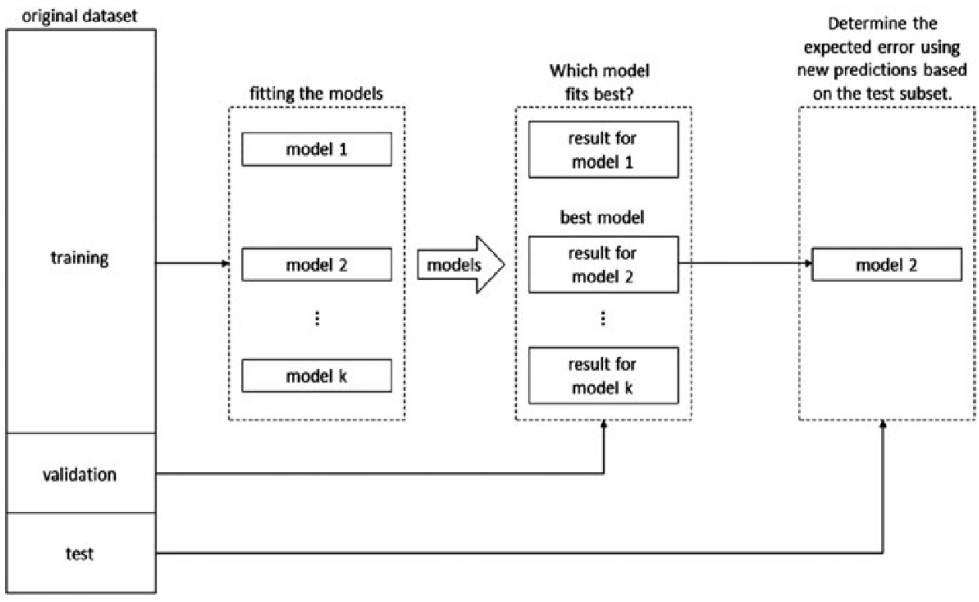
În exemplele anterioare am știut care este categoria din care face parte fiecare angajat (clasa), respectiv care sunt valorile luate în cazul fiecărei dimensiuni (variabile independente / predictor). Din punct de vedere practic, să prezicem ceva ce știm deja nu are nicio utilitate. Ceea ce ne interesează este să prezicem lucruri pe care nu le știm, folosindu-ne de ceea ce știm. Concret, relativ la exemplul anterior, vrem să prezicem care vor fi angajații care vor pleca știind care sunt caracteristicile lor. Pentru a face astfel de predicții, folosim un model de tipul celor identificate mai sus. Însă, înainte de a le folosi, trebuie să ne asigurăm că modelul respectiv descrie suficient de bine nu doar una dintre realitățile posibile, cea definită de algoritmul și datele folosite pentru instruirea acestuia (faza de învățare), ci și alte realități posibile, unele definite de seturi de date alternative (ca număr de cazuri și atribute, combinații de valori ale atributelor, tipuri de relații între atribute), respectiv algoritmi alternativi. Simplu spus, e necesar să ne asigurăm că modelul nostru poate fi generalizat fără a pierde prea mult din performanță (calitatea clasificării). Dat fiind faptul că, în această fază, nu avem informații cu privire la performanța viitoare a modelului, tot ceea ce putem face e să simulăm în faza de instruire diferite realități

și să ne asigurăm că modelul nostru performează suficient de bine relativ la toate acestea. Pașii prin care putem atinge acest obiectiv sunt descriși în continuare.

Realizarea unui model de clasificare (predicție, în general) presupune parcurgerea a trei faze: **instruire** (training), **validare** (validation) și **testare** (testing). Toate seturile de date utilizate în cadrul acestor faze includ atât predictorii cât și variabila de interes / dependentă (label). Recomandarea este ca fiecare fază să fie realizată pe un set diferit de date. Însă, în practică, cel mai adesea validarea este realizată pe un sub-set sau pe o serie de sub-seturi de date selectate din setul de date utilizat în faza de instruire.

În **prima fază**, de **instruire**, construim o serie de modele de predicție și calculăm diferite măsuri ale performanței acestora (performance metrics) (Figura 2.3-2). În **faza secundă**, de **validare**, folosim modelele construite în stadiul unu pentru a face predicții pe un alt set de date, calculăm măsurile de performanță și le comparăm cu valorile de performanță obținute în stadiul unu. Selectăm modelul (sau modelele) care performează cel mai bine pe ambele seturi de date (instruire și validare). Dacă performanța nu este similară, revenim la prima fază și refacem modelele sau construim altele. Adesea, primii doi pași sunt realizați împreună, urmând un proces iterativ.

Figura 2.3-2. Cele trei faze ale construirii unui model de clasificare: instruire, validare, testare

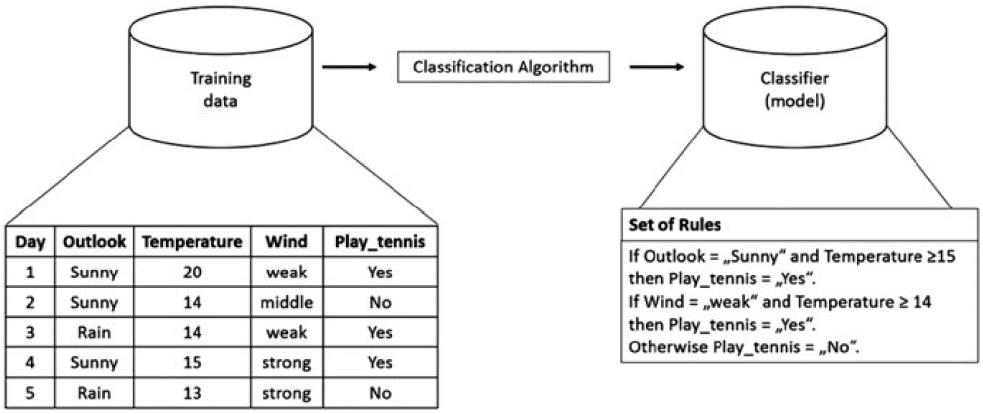


Sursa: (Wendler & Gröttrup, 2021, p. 757)

În **faza trei**, de **testare**, modelul este aplicat pe un nou set de date și i se măsoară din nou performanța. Și în acest caz ne interesează să obținem o performanță similară cu cea obținută în faza de validare. Dacă performanța modelului nu variază între faze, șansele de generalizare a acestuia cresc. Scopul acestor pași este de a ne asigura că modelul final are capacitatea de generalizare. Simplu spus, dorim să fim cât mai siguri că în urma aplicării modelului pe un set de date nou în care variabila de interes nu apare (în acest caz, nu știm dacă angajatul va părăsi sau nu compania), vom obține predicții cât mai apropiate de realitate și cu o calitate relativ similară celei obținute în etapele de instruire și validare. Situația de preferat este aceea în care performanța modelului de clasificare este ridicată și nu scade prea mult atunci când trecem de la etapa de instruire la cea de validare și apoi la cea de testare. Desigur, testul real va fi cel realizat pe un nou set de date, unul care nu conține variabila de interes. În acest caz vom ști care este performanța reală a modelului doar după o perioadă de timp de la realizarea predicțiilor și doar dacă măsurăm variabila de interes la finalul acestei perioade.

În urma procesului descris anterior, adică prin combinarea datelor și a unui algoritm de clasificare (putem folosi și o combinație de algoritmi), obținem un model de clasificare (clasificator) (Figura 2.3-3). Observăm că setul de date include predictorii și atributul de interes, iar rezultatul (clasificatorul) ia, în acest caz, forma unui set de reguli (sau funcții, dacă folosim un algoritm de clasificare care nu este bazat pe reguli).

Figura 2.3-3. Date + Algoritm de clasificare = Model de clasificare (Classifier)

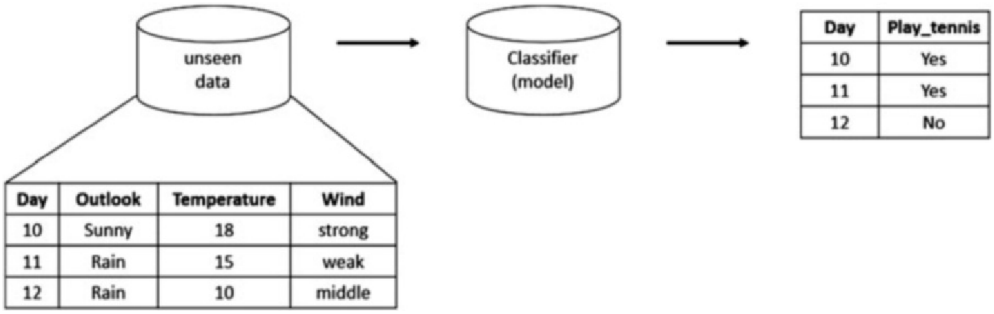


Day	Outlook	Temperature	Wind	Play_tennis
1	Sunny	20	weak	Yes
2	Sunny	14	middle	No
3	Rain	14	weak	Yes
4	Sunny	15	strong	Yes
5	Rain	13	strong	No

Sursa: (Wendler & Gröttrup, 2021, p. 758)

Folosind clasificatorul obținut și un nou set de date, unul care conține doar predictorii (exact aceeași predictori folosiți anterior), putem obține predicțiile (Figura 2.3-4). De exemplu, pentru ziua 10, predicția este Yes (jucăm tenis) deoarece modelul include regula „dacă Outlook = Sunny & Temperature ≥ 15, atunci Yes", iar cazul 10 ia valorile „Outlook = Sunny" și „Temperature = 18". Desigur, urmează să vedem dacă decizia a fost una corectă (vremea ne-a permis să jucăm tenis).

Figura 2.3-4. Date + Clasificator (Model) = Predicții



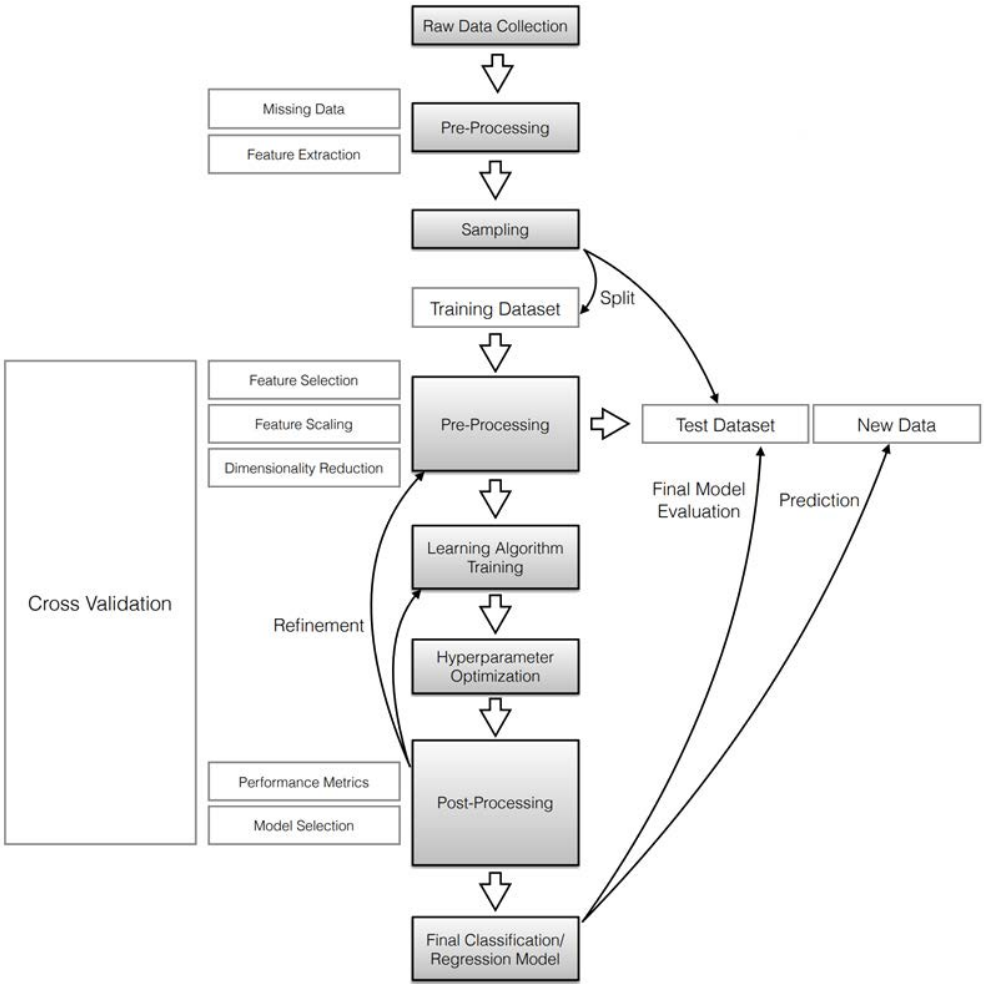
Sursa: (Wendler & Gröttrup, 2021, p. 758)

Fiecare dintre cele trei faze ale construirii unui model de clasificare poate fi divizată într-o serie de pași interconectați. O posibilă reprezentare grafică a întregului proces este cea din Figura 2.3-5. Totul începe cu datele brute, apoi continuă cu pre-procesarea inițială a datelor (extragerea atributelor, tratarea valorilor lipsă), construirea celor trei seturi de date (instruire, validare, testare), din nou pre-procesarea datelor (selectarea atributelor, scalarea atributelor, reducerea numărului de dimensiuni), realizarea fazelor de instruire și validare, post-procesarea datelor (calcularea metricilor de performanță, selectarea modelului), testarea modelului și apoi realizarea predicțiilor pe un set nou de date. Funcție de performanța obținută în urma primelor două faze (instruire și validare), procesul poate fi reluat de la partea de pre-procesare și/sau folosind un alt algoritm de clasificarea și/sau alte valori ale hiper-parametrilor algoritmului.

Un hiper-parametru este un parametru care influențează procesul de învățare, valorile acestuia fiind stabilite de utilizator sau automat de softul de analiză, înainte de începerea fazei de instruire. Adesea, performanța unui model variază mult în funcție de valorile alese ale hiper-parametrilor (atunci când algoritmul are hiper-

parametri), deci procesul de instruire presupune adesea și găsirea valorilor optime ale hiper-parametrilor (optimizarea hiper-parametrilor).

Figura 2.3-5. Pașii construirii unui model de predicție (învățare automată supervizată)



Sursa: Raschka, Sebastian. 2014. Supervised learning flowchart

Câteva exemple de hiper-parametri sunt următorii: gamma și C în cazul SVM, rata de învățare în cazul rețelei neuronale, numărul de vecini în cazul KNN etc. Spre deosebire de un hiper-parametru, un parametru este învățat din datele de instruire (estimat pe baza acestora). Câteva exemple de parametri sunt următorii: ponderările (weights) în cazul rețelei neuronale, probabilitățile în cazul Naive Bayes, coeficienții de regresie logistică etc.

2.4. Eroarea unui model de clasificare

Modelele de predicție, respectiv clasificare, urmăresc să găsească cea mai bună aproximare a relațiilor dintre un set de predictori (input) și o variabilă de interes / prezisă (output). Identificarea „celui mai bun model" nu este o sarcină ușoară, iar realitatea nu se lasă simplificată ușor și fără costuri, adică fără apariția unor erori de predicție / clasificare. Prin urmare, orice model este doar o aproximare a realității și, indiferent care este modelul și/sau algoritmul folosit, ne vom confrunta cu problema erorilor de predicție / clagsificare.

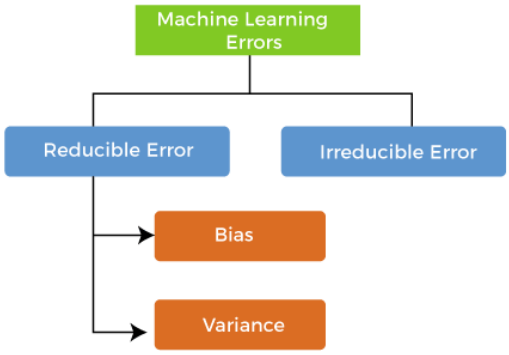
Relativ la un set de date putem extrage o cantitate limitată de informație. Această cantitate depinde în principal de numărul cazurilor, respectiv de numărul și calitatea predictorilor. Dacă setul de date conține mai multă informație și dacă modelul este unul relativ mai complex, în principiu putem extrage relativ mai multă informație. Procesul de extragere a informației este departe de a fi perfect, fiind afectat de o serie de erori.

Eroarea se referă la diferența dintre valorile prezise de un model / algoritm și valorile „reale". În acest context presupunem că valorile reale sunt identice cu valorile observate, respectiv valorile din setul de date, deși, evident, toate acestea pot diferi între ele din cauze care țin de procesul de colectare a datelor. În cazul problemelor de predicție eroarea ia forma diferenței dintre valorile prezise și valorile reale. În cazul problemelor de clasificare eroare se referă la faptul că unele dintre cazurile clasificate ca aparținând uneia dintre clase aparțin în realitate altei clasei. Media acestor erori, respectiv ponderea erorilor în cazul problemelor de clasificare, ne indică cât de bine modelul aproximează realitatea așa cum este descrisă aceasta de datele disponibile.

Eroarea totală poate fi separată în două mari tipuri de erori: eroarea care poate fi redusă (reducible) și eroarea care nu poate fi redusă (irreducible) (Figura 2.4-1). Eroarea ireductibilă se referă la eroarea inerentă setului de date, eroarea care nu pot fi redusă indiferent de modelul / algoritmul folosit. Cel mai adesea un set de date conține cazuri identice cu excepția variabilei de interes. Într-o astfel de situație, niciun model / algoritm nu poate distinge între acele cazuri, deci, pentru unele dintre cazuri, valorile / clasele prezise vor fi greșite. Să ne gândim de exemplu situația în care doi angajați au aceeași vechime și doar unul dintre ei pleacă din companie. Strict pe baza informației relativ la vechime, distincția (relativ la plecare) dintre acești angajați este imposibilă. În acest context, este util ca setul de date să

conțină un număr cât mai mare de predictori suficient de relevanți, iar erorile de măsurare (ceea ce se numește „zgomot” / „noise”) să fie cât mai reduse.[2] Erorile reductibile sunt erorile care, cel puțin teoretic, pot fi reduse. Acestea sunt de două tipuri: distorsiune (bias) și variantă (variance). Ele reprezintă subiectul principal al acestui sub-capitol.[3]

Figura 2.4-1. O clasificare a erorilor de predicție / clasificare



$$Bias(\hat{Y}) = E(\hat{Y}) - Y$$

$$Variance = E\left[(\hat{Y} - E(\hat{Y})^2\right]$$

Sursa: Bias and Variance in Machine Learning, cu modificări;
Y = valoarea adevărată a parametrului Y; Ŷ = estimatorului lui Y; E(Ŷ) = valoarea așteptată a estimatorului.

Distorsiune și variantă (Bias & Variance)

Pentru a putea învăța, orice model (algoritm) are la bază una sau mai multe asumpții specifice. Prin urmare, un algoritm poate modela cu succes doar datele care respectă asumpțiile făcute de acel algoritm. Atunci când datele nu respectă aceste asumpții rezultă o serie de erori sistematice de predicție. Prin erori sistematice ne referim la erorile care au același sens, aceeași direcție. De exemplu, un algoritm care presupune că relațiile dintre atribute sunt liniare va căuta și modela doar astfel de relații, deci nu va fi capabil să „vadă” și relații non-liniare. În consecință va considera că toate relațiile sunt liniare, deci, în cazul relațiilor non-liniare, erorile de predicție vor avea sistematic același semn (pozitive sau negative), aceeași direcție.

[2] Gentle Introduction to the Bias-Variance Trade-Off in Machine Learning.
[3] Două prezentări video la nivel introductiv a conceptelor de distorsiune și variantă: Machine Learning Fundamentals: Bias and Variance și Bias/Variance.

Distorsiunea (biasul) se referă tocmai la aceste erori sistematice care apar ca urmare a incongruenței dintre asumpțiile modelului și date.[4]

Dacă un model / algoritm face mai puține asumpții și / sau face asumpții relativ mai neimportante atunci când estimează funcția (definește regulile) de clasificare, erorile de clasificare vor fi relativ mai rare, deci distorsiunea va fi relativ mai mică. Dacă un model / algoritm face mai multe asumpții și / sau face asumpții relativ mai importante atunci când estimează funcția (definește regulile) de clasificare, erorile de clasificare vor fi relativ mai dese, deci distorsiunea va fi relativ mai mare. Unii algoritmi fac mai puține asumpții cu privire la forma funcției, prin urmare tind să aibă o distorsiune relativ mai redusă (de exemplu Decision Trees, k-Nearest Neighbors și Support Vector Machines). Alți algoritmi fac mai multe asumpții, deci tindă să aibă o distorsiune relativ mai mare (de exemplu Linear Regression, Linear Discriminant Analysis și Logistic Regression).

Distorsiunea poate fi de două tipuri: de căutare, respectiv de reprezentare. Distorsiunea de căutare se referă la criteriile preferate de algoritm atunci când caută o ipoteză în spațiul ipotezelor. De exemplu, unii algoritmi preferă ipotezele simple în defavoarea celor relativ mai complexe. Distorsiunea de reprezentare reduce spațiul ipotezelor posibile la ipotezele care respectă anumite condiții. De exemplu, algoritmul se limitează strict la ipotezele care pot fi reprezentate doar prin funcții de tip liniar (Moreira et al., 2019).

Varianța (variance) se referă la variația performanței clasificării asociate unui model atunci când setul de date de instruire se modifică (folosim un alt set de

[4] Termenul de bias poate fi folosit în diferite contexte, cu sensuri parțial diferite (Doty, Chris. 2021. How to Think about Bias in Machine Learning). În spațiul public (real world bias) este folosit uneori cu sens de discriminare (gen, vârstă, rasă etc.). În statistică apare în relație cu eșantionul (sampling bias) indicând faptul că acesta reprezintă distorsionat unele caracteristici ale populației. Atunci când construim un model de predicție, ambele tipuri de bias interacționează și influențează gradul de distorsiune asociat modelului. Să presupunem că o companie favorizează sistematic bărbații atunci când face angajări, mai ales când e vorba de poziții de conducere. Dacă folosim datele anterioare cu privire la candidați și angajări pentru a instrui un model de clasificare, modelul nostru va include ambele tipuri de distorsiune (real world bias și sampling bias), deci va reflecta aceste distorsiuni în predicțiile făcute. În relația cu știința datelor, putem vorbi de distorsiunea asociată unui model de predicție (model bias). Să considerăm cazul unei companii care folosește un model pentru a evalua candidații care aplică online. După un an compania constată că scorurile primite de candidații de culoare sunt sistematic mai mici comparativ cu scorurile primite de restul candidaților, chiar dacă o serie de caracteristici relevante (vechime, poziție, specializare etc.) iau aceleași valori indiferent de rasă. În acest caz vorbim de un model distorsionat (model bias).

date). Atunci când un algoritm este utilizat pentru estimarea unei funcții folosind diferite seturi de date este de preferat să obținem o performanță relativ similară (doar așa putem avea încredere că modelul nostru va avea o performanță similară și în viitor, deci îl putem generaliza). Dacă atunci când folosim un alt set de date de instruire, fără să schimbăm nimic altceva, performanța modelului crește sau scade (oscilează) relativ mult, varianța este mare, deci performanța viitoare, pe alte seturi de date, este incertă. Când performanța este stabilă, varianța este mică, deci ne așteptăm ca și în viitor să obținem o performanță similară. Simplu spus, varianța ne arată cât de sensibil e un model (performanța lui) atunci când modificăm setul de date de instruire.

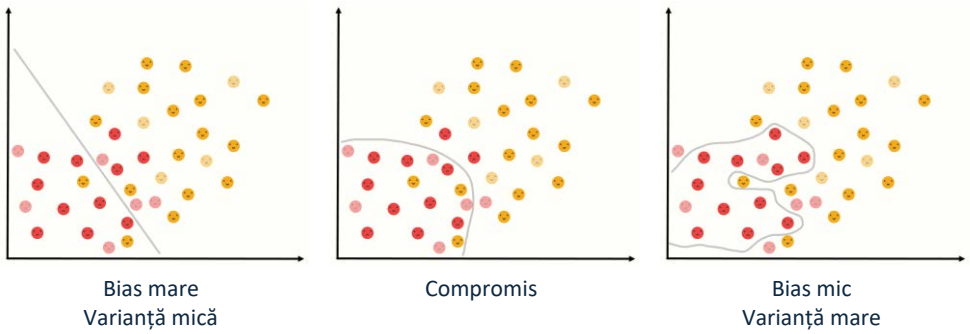
Desigur, preferăm un model cu o performanță stabilă (varianță mică), mai puțin în cazurile în care aceasta indică un model mult prea simplu relativ la complexitatea relațiilor din setul de date de instruire (underfitting). Dacă modelul este mult prea sensibil la schimbările din setul de date de instruire, va avea o varianță mare, deci șansele de generalizare la alte seturi de date vor fi reduse (overfitting). Unii algoritmi produc rezultate relativ similare indiferent de setul de date de training utilizat (mai exact de variațiile normale la nivelul setului de date), prin urmare au o eroare mai redusă (de exemplu Linear Regression, Linear Discriminant Analysis și Logistic Regression). Alți algoritmi produc rezultate care variază mai mult în funcție de particularitățile setului de date de training, prin urmare au o eroare mai mare (de exemplu Decision Trees, k-Nearest Neighbors și Support Vector Machines).[5]

Pentru a avea și o reprezentare vizuală a celor două concepte ne putem folosi de graficele din Figura 2.4-2. În acest caz am considerat un exemplu simplu de clasificare, cu două clase și doi predictori metrici. Concluziile se aplică și în cazul în care avem mai mulți predictori (categoriali și/sau metrici). În graficul din stânga cele două clase au fost separate folosind o funcție simplă ce definește o dreaptă. În graficul din centru modelul este reprezentat de o curbă, funcția asociată fiind doar puțin mai complexă comparativ cu funcția liniară. În graficul din dreapta complexitatea modelului / funcției crește semnificativ. Relativ la setul de date de instruire (culorile mai aprinse), observăm că numărul de erori de clasificare a fețelor roșii scade de la stânga la dreapta (3, 2, 0), deci modelul din dreapta este cel mai performant. În cazul setului de date de test (culorile mai fade), cele mai puține erori apar în cazul modelului cu o complexitate intermediară (1). Celelalte două modele produc 3, respectiv 4 erori. Observăm că, relativ la setul de test, modelul din mijloc are cea mai bună performanță. Simultan, modelul are o variație a performanței mică

[5] Desigur varianța poate fi redusă prin modificarea valorilor parametrilor importanți. De exemplu, în cazul k-NN creștem valoarea lui k (numărul de vecini), iar în cazul SVM creștem valoarea lui C.

(face aproximativ același număr de erori pe ambele seturi de date). În același timp, modelul este relativ simplu. Pe baza acestor argumente, preferăm modelul din mijloc.

Figura 2.4-2. Relația dintre algoritmul de clasificare utilizat și distorsiune, respectiv varianță



Fețele cu culori relativ mai estompate reprezintă cazurile din setul de testare, celelalte cazurile din setul de instruire.

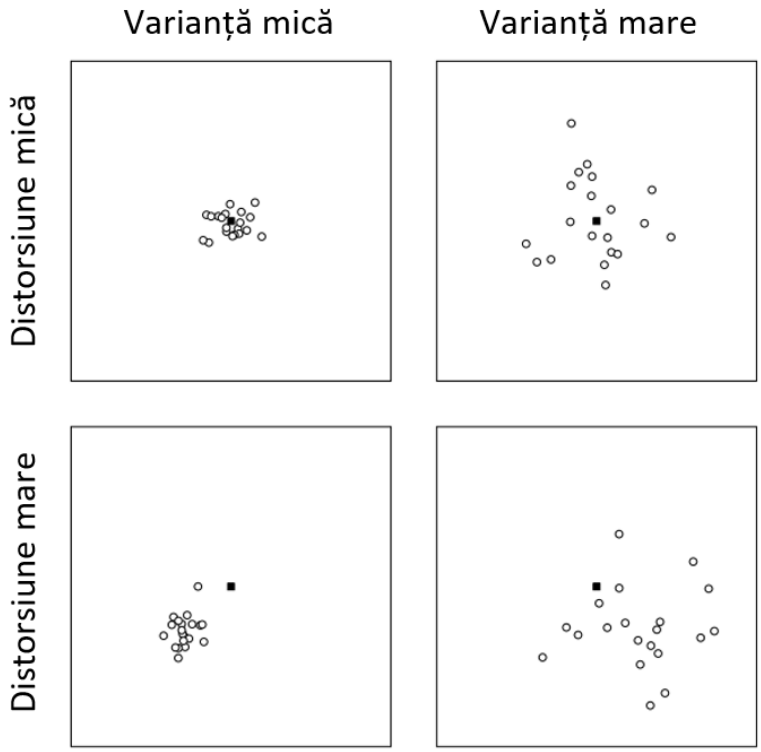
Tipuri de modele în funcție de distorsiune și varianță

Simplificând lucrurile și combinând cele două tipuri de erori, rezultă patru tipuri de modele (Figura 2.4-3). Pătratele pline din această figură reprezintă „valoarea reală", iar cercurile goale reprezintă valorile estimate. Cele mai bune modele apar în cadranul din stânga-sus (low variance + low bias), iar cele mai rele în dreapta-jos (high variance + high bias). Cele patru tipuri pot fi descrise astfel:

- **Distorsiune mică + Varianță mică**: este modelul ideal; un astfel de model reproduce relațiile dintre variabile în cazul majorității seturilor de date de instruire folosite, prin urmare predicțiile vor fi corecte; modelul va produce sistematic predicții corecte și în cazul setului de date de test; în practică întâlnim rar astfel de modele.
- **Distorsiune mică + Varianță mare**: la nivelul setului de date de instruire un model de acest tip aproximează în medie relativ bine relațiile dintre variabile, dar performanța sa este variabilă (unele predicții vor fi corecte, altele incorecte); la nivelul setului de date de test modelul va produce în egală măsură atât predicții corecte cât și greșite; spunem că modelul suferă de supra-ajustare (overfitting).

- **Distorsiune mare + Varianță mică**: la nivelul setului de date de instruire un astfel de model produce predicții sistematic similare, dar acestea diferă față de realitate în același sens; același lucru se va întâmpla și la nivelul setului de date de test; spunem că modelul suferă de sub-ajustare (underfitting).
- **Distorsiune mare + Varianță mare**: modelul reproduce incorect relațiile dintre variabile fiind în același timp și foarte sensibil la schimbările de la nivelul setului de date de instruire (predicțiile vor fi instabile, respectiv incorecte în aceeași direcție); variabilitatea mare și incorectitudinea sistematică a predicțiilor se vor regăsi și în cazul setului de test.

Figura 2.4-3. O tipologie a modelelor în funcție de distorsiune (bias) și varianță (variance)



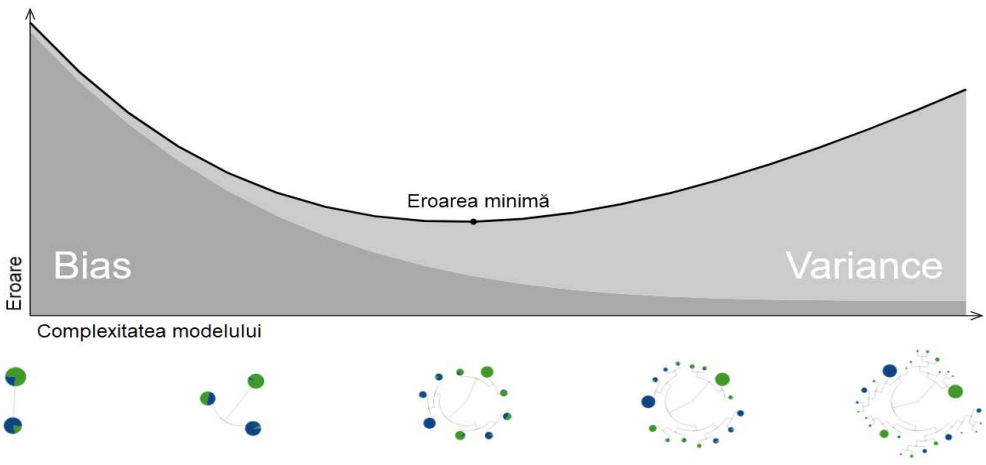
În Tabelul 2.4-1 am prezentat câte un exemplu pentru fiecare dintre cele patru tipuri de combinații. Am ales ca măsură a performanței acuratețea (valoarea maximă e 100% = toate cazurile sunt clasificate corect). Pentru a simplifica situația am considerat că nu există eroare ireductibilă. Pe ultima coloană am indicat diferite soluții pentru a crește performanța clasificării în cazul modelelor care au distorsiune mare, respectiv varianță mare.

Tabelul 2.4-1. Ce putem face pentru a reduce distorsiunea și varianța unui model?

Acuratețe instruire	Acuratețe testare	Distorsiune	Varianță	Ce putem face pentru a reduce ...?
63	62	Mare 37 (100-63)	Mică 1 (63-62)	**... distorsiunea** – creștem complexitatea modelului; – adăugăm predictori (mai buni); – reducem regularizarea; – creștem numărul cazurilor; – folosim ansambluri de modele.
99	85	Mică 1 (100-99)	Mare 14 (99-85)	**... varianța** – reducem complexitatea modelului; – eliminăm predictorii irelevanți, reducem numărul dimensiunilor; – mărim regularizarea; – creștem numărul cazurilor; – folosim ansambluri de modele.
63	42	Mare 37 (100-63)	Mare 21 (63-42)	Modelul este slab. Implementăm soluții din ambele liste.
99	98	Mică 1 (100-99)	Mică 1 (99-98)	Modelul este excelent. Nu îl modificăm.

Am considerat că o predicția optimă are o acuratețe de 100%. În unele cazuri predicția optimă poate avea o acuratețe mai mică.

Figura 2.4-4. Relația dintre distorsiune (bias) și varianță (variance)



Sursa: Model Tuning and the Bias-Variance Tradeoff

Scopul fazei de instruire este de a reduce eroarea totală, mai exact suma celor două tipuri de erori (distorsiune și varianță), adică eroarea reductibilă. Aceasta poate să scadă doar până la un anumit punct, după care crește (Figura 2.4-4). Punctul respectiv indică de altfel și nivelul optim de complexitate a modelului. Distorsiunea și varianța corelează negativ. De aici și imposibilitatea de a le reducem pe ambele simultan (cel puțin nu sub o anumită valoare specifică fiecărui set de date și model). Tot ce putem face e să găsim punctul optim în care suma lor să fie minimă.

Relația dintre eroarea și complexitatea modelului (bias – variance tradeoff)

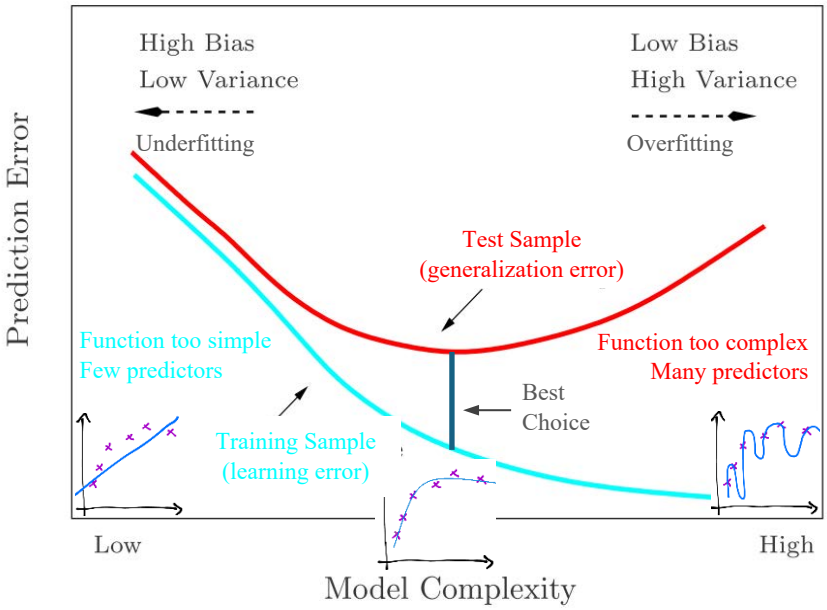
Distorsiunea și varianța sunt dependente de complexitatea modelului. Complexitatea unui model este dată de numărul de variabile (predictori), relațiile dintre variabile și forma acestor relații, algoritmul utilizat și parametrii acestuia, respectiv de numărul de cazuri / observații din setul de date. Complexitatea unui model crește atunci când acesta include mai multe variabile și relații complexe între ele, algoritmi mai complecși și / sau mai mulți, cu mai mulți parametri, respectiv mai multe cazuri. În Figura 2.4-5 am sugerat complexitatea unui model ipotetic prin cele trei imagini mici poziționate în partea de jos: în stânga, complexitatea redusă este ilustrată prin linia dreaptă (modelul considerat este regresia liniară) pe care o folosim pentru a reprezenta o relație non-liniară („realitatea”); în dreapta, aceeași „realitate” este descrisă perfect de un model mult prea complex, cu șanse foarte mici de generalizare pe un alt set de date; în centru apare reprezentarea optimă, modelul de compromis între cele două extreme (raportul optim între distorsiune și varianță; modelul care minimizează suma celor două tipuri de erori).

Același grafic prezintă relația dintre complexitatea modelului și eroarea de predicție asociată, în funcție de tipul de setul de date utilizat. Dacă avem în vedere setul de date de instruire, observăm că eroarea de predicție scade non-liniar (mai mult la început și tot mai puțin ulterior) pe măsură ce modelul devine tot mai complex (folosim algoritmi mai complecși, includem mai mulți predictori sau definim relații mai complicate între aceștia). Raportat la setul de test, se observă o scădere a erorii de predicție până la un anumit punct, după care eroarea începe să crească. Punctul în care eroarea de predicție începe să crească ne indică nivelul optim de complexitate a modelului, compromisul între distorsiune și varianță.

În acest context, putem înțelege mai bine conceptele de supra-ajustare (overfitting) și sub-ajustare (underfitting). Supra-ajustarea (partea dreapta a imaginii) se referă

la situația în care modelul de predicție reproduce foarte bine realitatea așa cum este ea aproximată de setul de date de training (erorile de predicție pe acest set de date sunt foarte mici), dar nu mai atinge aceeași performanță pe setul de date de test (în acest caz erorile de predicție sunt mult mai mari). Altfel spus, modelul surprinde prea mult „zgomotul" din date. Sub-ajustarea se referă la situația în care modelul de predicție nu reușește să surprindă „realitatea" suficient de corect nici măcar în setul de date de training. Aceasta se întâmplă deoarece modelul este mult prea simplu comparativ cu realitatea pe care încearcă să o descrie. Ambele tipuri de situații trebuie evitate deoarece au ca rezultat o performanță relativ redusă a clasificării la nivelul setului de date de test.

Figura 2.4-5. Relația dintre eroarea și complexitatea modelului (bias – variance tradeoff)[6]



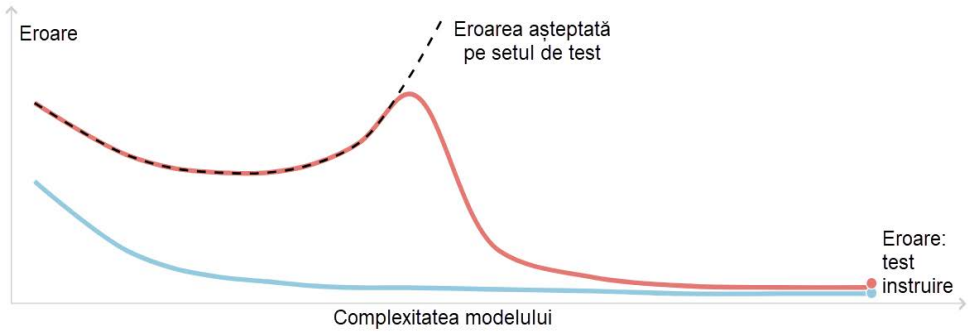
Sursa: (Hastie et al., 2018, p. 57; Robette, 2022), cu modificări

Necesitatea unui compromis între distorsiune și variantă, nevoia identificării nivelului optim de complexitate a modelului, sau, simplu spus, faptul că un model nu poate îndeplini simultan ambele criterii este văzut cel mai adesea ca o axiomă. Însă, unele analize relativ mai recente arată că e posibil să obținem un model care

[6] O prezentare interactivă extrem de sugestivă a relației dintre eroarea și complexitatea modelului în cazul algoritmului arbore decizional poate fi vizualizată aici: Model Tuning and the Bias-Variance Tradeoff.

să aibă simultan și o distorsiune mică și o varianță redusă. Este vorba despre fenomenul „double descent” (Belkin et al., 2019; Neal et al., 2019). O ilustrare a acestui fenomen ar putea arăta precum în Figura 2.4-6. Dezbaterea este departe de a fi finalizată. Contrar acestei idei, alți autori consideră că fenomenul „double descent” aduce mai degrabă argumente suplimentare în favoarea axiomei (Double Descent - Part 1: A Visual Introduction).

Figura 2.4-6. O ilustrare a fenomenului „double descent”



3. VALIDAREA UNUI MODEL DE CLASIFICARE (VALIDATION)

Să presupunem că firma în care lucrăm se confruntă cu o atriție mare a personalului. Firesc, firma își propune să vadă care sunt cauzele acestei situații negative. În calitate de analist, citim despre fenomenul atriției și factorii care sunt asociați cu variația acestuia, respectiv îi determină variația. Ne gândim la situația specifică în care ne aflăm (caracteristicile firmei, ale angajaților, pieței etc.). În baza acestor informații ajungem la un model teoretic care ar putea explica atriția din compania noastră. Colectăm datele necesare sau le preluăm din sursele existente și construim un prim model de clasificare. Ne mai gândim, discutăm și cu alți angajați, transformăm variabilele existente, identificăm și adăugăm și alți predictori etc. Ajungem astfel la un model teoretic final. Testăm modelul teoretic pe datele existente, facem unele modificări astfel încât modelul să aibă o predicție cât mai bună pe acele date și apoi folosim modelul final pentru a prezice probabilitatea de plecare din companie a tuturor angajaților de la acel moment. În următorul an (sau după câțiva ani) o parte din angajați părăsesc compania. Pornind de la predicții și datele reale observate, calculăm calitatea predicției și observăm că e semnificativ mai mică în comparație cu calitatea așteptată, cea calculată pe datele inițiale folosind modelul final. Firesc, suntem nedumeriți, ne întrebăm alături de cei din conducere dacă nu am greșit ceva, poate chiar ajungem să credem că o analiză de data mining este mai degrabă inutilă. Răspunsul la această întrebare e simplu: nu am validat modelul. Simplu spus, nu ne-am asigurat că modelul poate fi generalizat dincolo de setul de date utilizat pentru producerea lui. Sau, altfel spus, nu ne-am asigurat că modelul ajunge aproximativ la aceeași reprezentare a realității indiferent de setul de date utilizat (presupunând că acesta e reprezentativ pentru populația de interes). În cadrul acestui capitol vom discuta despre validare și importanța acesteia, respectiv vom descrie și ilustra formele pe care le poate lua aceasta.

Necesitatea validării unui model s-a impus relativ mai recent, odată cu apariția softurilor specifice analizelor de tip data mining / machine learning. În general, softurile clasice destinate analizelor statistice nu includ(eau) proceduri automate de validare a modelelor de predicție. Dacă dorim să vedem cât de bine performează un

model la modul real (care este calitatea predicțiilor), validarea acestuia este un pas absolut necesar. Validarea se face cel mai adesea pe un set de date independent de setul de date folosit pentru construirea modelului sau pe o serie de sub-seturi de date ale setului de date folosit pentru construirea modelului. Scopul este de a evalua capacitatea modelului de a produce predicții de calitate pe alte seturi de date (out-of-sample).

Validarea reprezintă un pas esențial dacă dorim să obținem un model care să fie caracterizat simultan printr-o performanță foarte bună a clasificării și o capacitate mare de generalizare. Validarea unui model predictiv (de clasificare) se poate face urmând una dintre strategiile hold-out și k-fold. Fiecare dintre aceste strategii are avantaje și dezavantaje. Strategia hold-out realizează validarea modelului pe un singur sub-set de date independent de setul de date de instruire, deci presupune rularea modelului de doar două ori (de aici și costurile computaționale mai reduse). De asemenea, fiind implicat un singur sub-set de date, performanța estimată a modelului va avea o varianță relativ mai mare. În cazul strategiei k-fold, setul de date utilizat pentru training este divizat urmând o procedură de eșantionare de tip stratificat aleator în k sub-seturi (5-10 cel mai adesea), modelul este estimat pe rând pe câte k-1 sub-seturi și validat pe cel de al k-lea. Rezultatele obținute la nivelul fiecărei validări sunt agregate și exprimate sub forma unor măsuri medii a calității predicției (performance metrics) precum acuratețea (accuracy sau ponderea cazurilor corect clasificate din total cazuri), curba ROC etc. Dat fiind faptul că este necesară rularea mai multor modele (k+1), costurile computaționale sunt mai mari; iar varianța evaluării performanței modelului este mai mică (comparativ cu strategia holdout).

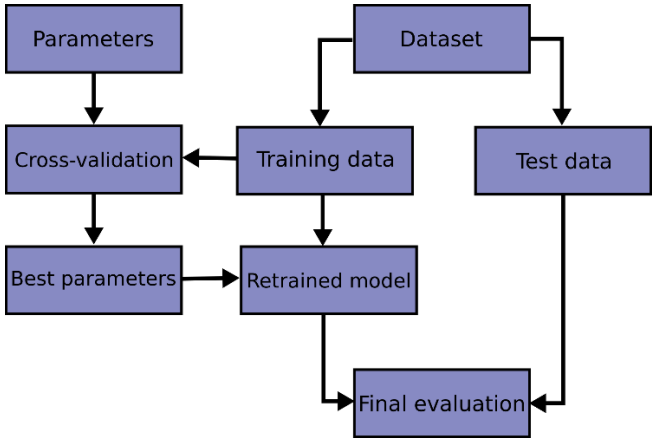
Tabul Validation din fereastra Operators include o serie de operatori utili pentru validarea modelelor de predicție. Acești operatori sunt grupați în trei mari categorii: validarea propriu-zisă a unui model de predicție (Validation), estimarea performanței unui model de predicție (Performance) și evaluarea vizuală a calității unui model de predicție (Visual). În cele ce urmează vom prezenta și ilustra fiecare dintre operatorii mai importanți asociați acestor categorii.

3.1. Importanța validării

Adesea, modelele produc predicții bune relativ la setul de date utilizat pentru construirea lor, însă, atunci când le folosim pentru a face predicții pe date noi, performanța acestor modele scade considerabil. Simplu spus, performanța

predictivă (calitatea predicțiilor) a unui model calculată la nivelul setului de date de instruire este în general semnificativ mai mare comparativ cu performanța reală, cea obținută în producție, deci pe date noi (care nu au fost folosite pentru estimarea modelului). Un model care se comportă astfel, nu doar că este inutil, dar poate duce și la pierderi considerabile (timp, bani, resurse umane) pentru organizația care își fundamentează deciziile pe el. Pentru a reduce distanța dintre așteptări și realitate, fiecare model trebuie validat și testat. Prin validare ne referim la procesul de căutare și identificare a parametrilor modelului care produc cele mai bune predicții la nivelul diferitelor sub-seturi de date extrase din setul de date de instruire. După finalizarea validării, modelul obținut este supus unui test final, de această dată pe un set de date (setul de date de test) care nu a avut nicio legătură cu procesul de estimare a modelului. Schema logică a acestui proces este ilustrată în Figura 3.1-1.

Figura 3.1-1. Schema logică a estimării și evaluării unui model de predicție



Sursa: scikit-learn. Cross-validation: evaluating estimator performance

Pentru a ilustra importanța validării și testării unui model de predicție, prezentăm o analiză comparativă în Tabelul 3.1-1. Datele din tabel prezintă erorile de predicție obținute pentru o combinație de trei modele de predicție și patru seturi de date. Pentru toate situațiile, atributul prezis are două categorii, restul atributelor din setul de date fiind folosite ca predictori. Coloana „Șansă" prezintă eroarea de predicție în cazul în care nu folosim niciun model, adică prezicem că toate cazurile vor lua aceeași valoare. Firesc, ponderea erorilor este egală cu ponderea clasei relativ mai puțin frecventă. Atunci când folosim un algoritm de predicție (oricare e acesta), ponderea erorilor scade. Observăm că scăderea variază mult în funcție de tipul de set de date (datele utilizate pentru instruirea modelului, respectiv pentru testarea

acestuia). Astfel, în cazul seturilor de date de instruire, erorile sunt semnificativ mai mici comparativ cu erorile obținute pe seturile de date de test. În medie, subestimarea erorilor se situează în intervalul 7-11 puncte procentuale (punctual ajunge și la 18 puncte – setul de date Sonar, algoritmul Random Forest). Simplu spus, dacă limităm analiza la faza de instruire a modelului, vom obține erori mult mai mici comparativ cu erorile pe care le vom obține în practică (vom subestima erorile reale, deci vom supraestima calitatea modelelor de predicție). În concluzie, erorile obținute pe datele de instruire nu sunt utile pentru a anticipa erorile pe care le vom obține în practică / producție, prin urmare e absolut necesar să testăm toate modelele pe seturi de date care nu au avut nicio legătură cu faza de instruire.

Tabelul 3.1-1. Eroarea de predicție în cazul setului de date de instruire vs. de test pentru câteva seturi de date și clasificatori

Model predicție	**Șansă**	**Random Forest**		**Regresie Logistică**		**5-NN**	
Setul de date	**Eroare**	**Test**	**Instruire**	**Test**	**Instruire**	**Test**	**Instruire**
Diabetes	34.9	32.6	27.0	32.6	23.2	28.7	19.7
Ionosphere	35.9	17.1	4.6	18.1	12.3	21.9	12.5
Sonar	46.6	32.3	14.4	29.0	18.3	25.8	13.5
Wine	22.6	18.4	14.5	11.8	10.1	21.9	10.2
Media erorilor	**35.0**	**25.1**	**15.1**	**22.9**	**16.0**	**24.6**	**14.0**
Subestimarea medie a erorilor			**10.0**		**6.9**		**10.6**

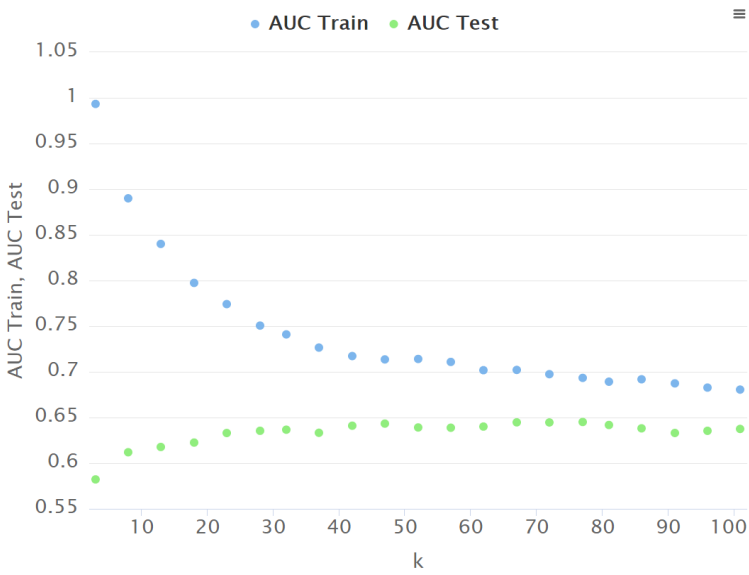
Sursa: Adaptare după (RapidMiner, 2017); valorile reprezintă procente.

Un exemplu simplu, produs cu ajutorul softului RapidMiner (vezi procesul „3.1 Train vs. Test", folderul DMSS2/Procese[1]), ne arată foarte clar că performanța unui model este supra-estimată atunci când ne raportăm la setul de date de instruire comparativ cu setul de date de test. În cadrul acestui proces am calculat comparativ, setul de date de instruire vs. testare, performanța clasificării folosind măsura AUROC (vezi sub-capitolul 4.3; o valoare mai apropiată de unu indică o performanță mai mare) pentru diferite valori ale parametrului k (numărul de vecini, clasificatorul KNN). Graficul rezultat (Figura 3.1-2), arată foarte clar că performanța în cazul setului de date de instruire este întotdeauna mai mare comparativ cu performanța

[1] Prima dată se rulează modelul, apoi se alege la Results tabul „ExampleSet (Log to Data)", apoi Visualizations și se importă (de la 📂 asociat graficului) configurația „train vs test (AUC).json" din folderul Grafice.

în cazul setului de date de testare. Foarte important, performanța la care ne vom raporta, cea pe care ne așteptăm să o regăsim atunci când modelul este aplicat unor date noi, este cea observată pe setul de date de test (punctele verzi), adică performanța mai mică dintre cele două.

Figura 3.1-2. Performanța estimată a unui model k-NN în funcție de valoarea parametrului k: instruire vs. testare



Validarea unui model se poate realiza în diferite moduri. În continuare vom prezenta validarea simplă (split), încrucișată (cross) și bootstrapping.

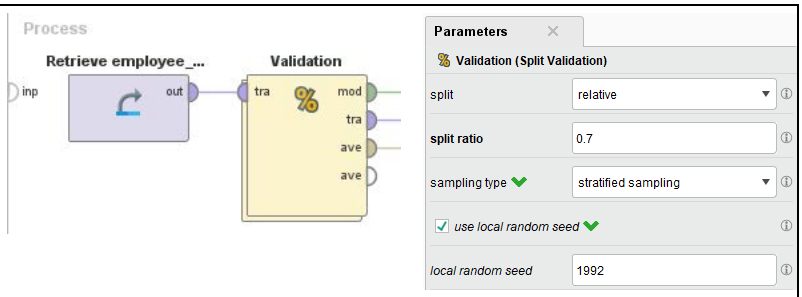
3.2. Validarea simplă (Split Validation)

Cea mai simplă formă de validare este de tip split, adică divizare, numită și hold-out. În acest caz, setul de date (acesta conține și valorile observate în cazul variabilei dependente / prezise) este împărțit în două sub-seturi, de obicei unul mai mare comparativ cu celălalt (70 vs. 30% este o divizare frecventă). Divizarea se poate face considerând valorile absolute (numărul de cazuri), respectiv valorile relative (ponderea cazurilor). Tipul de eșantionare poate fi de tip liniar, stratificat, amestecat sau automat. Sub-setul mai mare este folosit pentru obținerea modelului de predicție, iar cel mai mic pentru testarea modelului. În exemplul din Figura 3.2-1 am

realizat o divizare relativă (70% vs. 30%), folosind o eșantionare de tip stratificat aleatoriu. Pentru a obține aceleași rezultate de fiecare dată când este rulat procesul, am marcat opțiunea „use local random seed". Modelul a prezis corect 82.77% din totalul cazurilor incluse în setul de date de testare, deci ne așteptăm ca, în producție, acuratețea modelului să fie apropiată de 83% (chiar dacă mai mică).

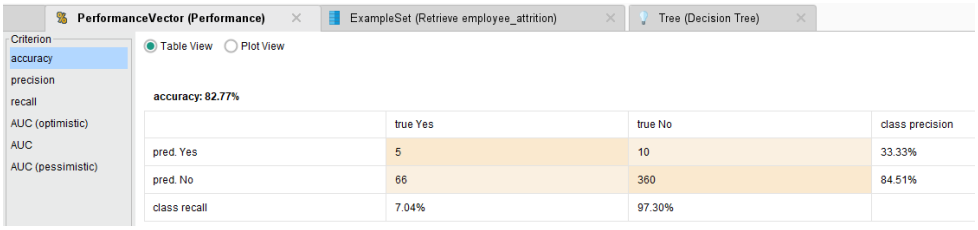
Figura 3.2-1. Validarea simplă a unui model de predicție

Pasul 1:
Conectăm setul de date și operatorul Split Validation. Setăm valorile parametrilor conform imaginii.



Pasul 2:
În interiorul operatorului Split Validation conectăm operatorii din imagine.

Rezultate:
Obținem trei tipuri de rezultate: confusion matrix (imaginea de mai jos); sub-setul de date utilizat pentru testarea modelului, adică setul care include 30% dintre cazuri (1-0.7=0.3, de aici 30%); tot aici vor fi incluse și predicțiile realizate de către model; modelul de predicție utilizat (Decision Tree în acest caz).



	true Yes	true No	class precision
pred. Yes	5	10	33.33%
pred. No	66	360	84.51%
class recall	7.04%	97.30%	

O variantă similară a acestui algoritm de validare este implementată în operatorul „Wrapper Split Validation”. Singura diferență constă în includerea unui pas suplimentar, ponderarea atributelor. Pentru un exemplu de aplicare a acestui algoritm se poate consulta procesul din arhiva asociată acestui volum.

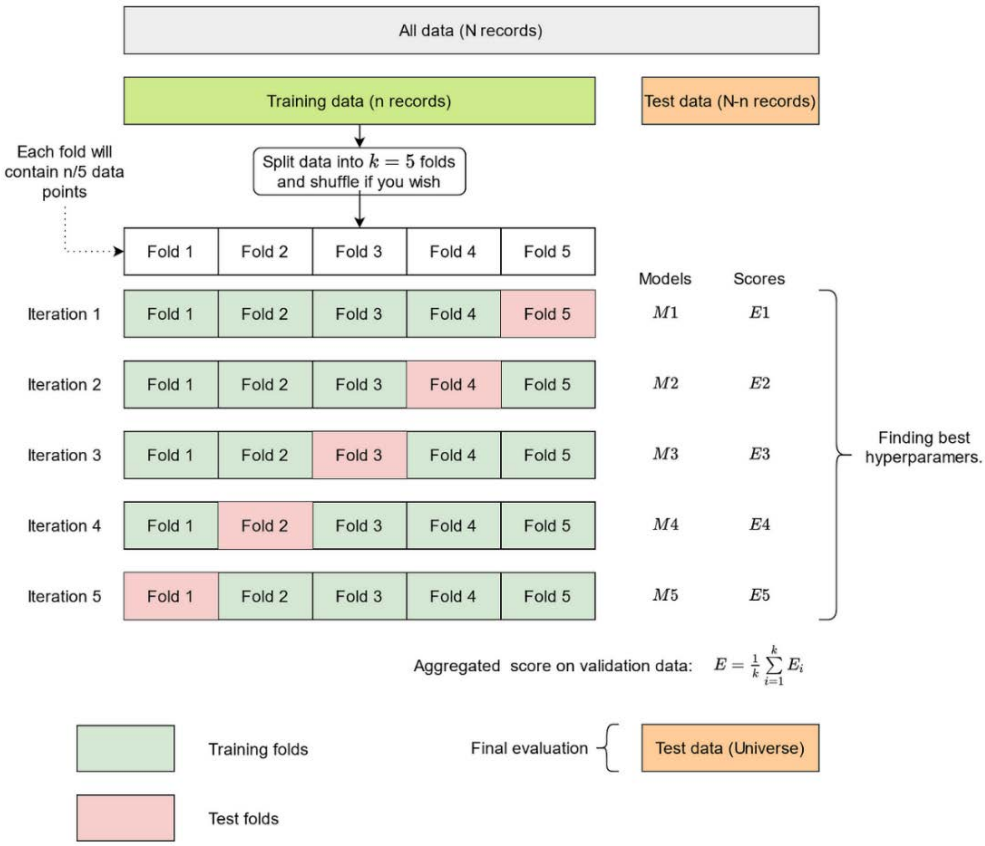
3.3. Validarea încrucișată (Cross Validation)

Validarea încrucișată (cross) este relativ mai complexă. De această dată setul de date este divizat într-un număr k de sub-seturi (numite folds), de obicei 5-10[2] (mai puține dacă avem insuficiente cazuri), fiecare cu aproximativ același număr de cazuri. Tipul de eșantionare poate fi de tip liniar, stratificat, amestecat sau automat. Modelul de predicție ales este estimat de k ori, fiind instruit de fiecare dată pe o combinație de k-1 sub-seturi și testat pe sub-setul rămas. Logica de validare încrucișată este ilustrată în Figura 3.3-1. Pentru simplitate, aici valoare lui k este 5. Setul de date de instruire este divizat în 5 sub-seturi (folds), deci modelul este rulat de 5 ori, de fiecare dată folosind 4 sub-seturi pentru estimarea parametrilor modelului și un sub-set pentru testarea lor.[3] În figură, suplimentar, este ilustrată și necesitatea testării finale a modelul validat pe un alt sub-set de date (niciunul dintre cazurile incluse în acesta nu apare în seturile utilizate pentru validarea modelului). Această cerință trebuie îndeplinită indiferent de tipul de validare ales.

[2] Numărul de sub-seturi (valoarea lui k) poate fi oriunde în intervalul [2;n], unde n este numărul de cazuri din setul de date original. Pentru k=2 vom obține două sub-seturi de date de mărime egală, deci vom realiza o validare simplă (split sau hold-out). Pentru k=n, vom obține n sub-seturi, fiecare cu un singur caz, deci validarea se va realiza de n ori, de fiecare dată setul de date de instruire având n-1 cazuri, iar cel de test un singur caz (acest tip de validare se numește Leave-One-Out Cross-Validation, adică LOOCV). În practică, atunci când n este mare, LOOCV necesită foarte multe resurse de calcul, deci este puțin fezabilă. În plus, modelele vor fi aproape identice (variabilitate redusă), iar valorile indicatorilor de calitate a modelului de predicție nu vor diferi semnificativ de cazul în care k=10.

[3] Pentru o prezentare video introductivă cu privire la procesului de validare încrucișată se poate consulta prezentarea video Cross Validation Introduction.

Figura 3.3-1. Logica validării încrucișate

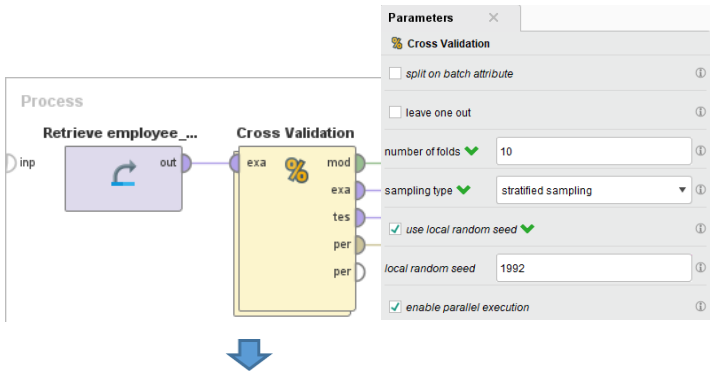


Sursa: Kiprono Elijah Koech. 2021. Cross Validation in Machine Learning

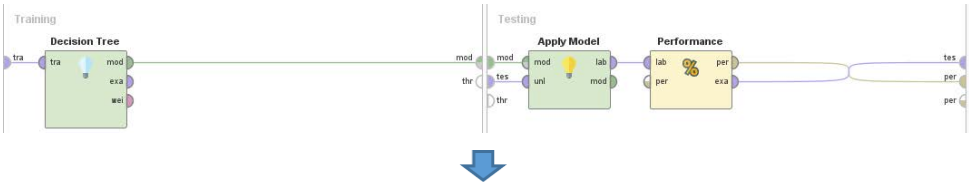
În exemplul din Figura 3.3-2 am realizat o divizare în 10 sub-seturi folosind eșantionarea de tip stratificat aleatoriu. Pentru a obține aceleași rezultate de fiecare dată când este rulat procesul, am marcat opțiunea „use local random seed”. Modelul a prezis corect, în medie, 82.86% din totalul cazurilor incluse în setul de date de testare, deci ne așteptăm ca, în producție, acuratețea modelului să fie apropiată de 83% (chiar dacă mai mică). Stabilitatea acestei predicții este foarte mare (abaterea medie de la media acurateței celor 10 predicții este ± 1.5%). O stabilitate ridicată este importantă deoarece crește probabilitatea de generalizare a modelului de predicție (șansele ca acesta să performeze aproape la fel de bine și pe alte seturi de date).

Figura 3.3-2. Validarea încrucișată a unui model de predicție

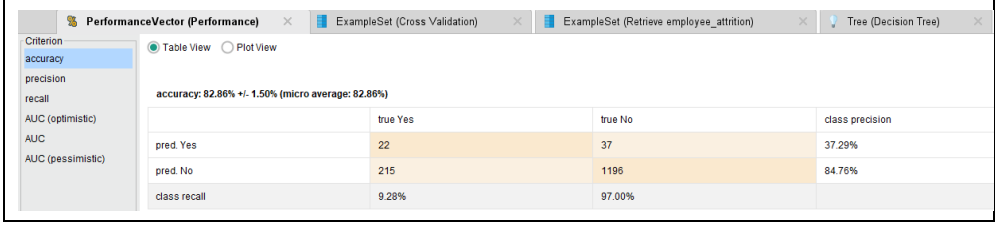
Pasul 1:
Conectăm setul de date și operatorul Cross Validation. Setăm valorile parametrilor conform imaginii.



Pasul 2:
În interiorul operatorului Cross Validation conectăm operatorii din imagine.



Rezultate:
Obținem patru tipuri de rezultate: confusion matrix (imaginea de mai jos); setul de date utilizat pentru instruirea și validarea modelului, cu și fără predicțiile incluse; modelul de predicție utilizat (Decision Tree în acest caz).



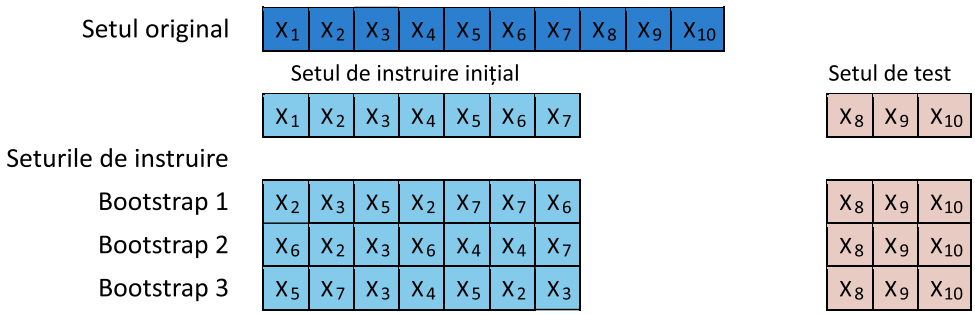
	true Yes	true No	class precision
pred. Yes	22	37	37.29%
pred. No	215	1196	84.76%
class recall	9.28%	97.00%	

O variantă similară a acestui algoritm de validare este implementată în operatorul „Wrapper X-Validation”. Singura diferență constă în includerea unui pas suplimentar, ponderarea atributelor. Pentru un exemplu de aplicare a acestui algoritm se poate consulta procesul din arhiva asociată acestui volum.

3.4. Validarea bootstrapping (Bootstrapping Validation)

Validarea bootstrapping are la bază eșantionarea aleatoare cu înlocuire. Acest tip de eșantionare se realizează astfel: (1) considerând setul de date de interes, se selectează aleator un caz; (2) cazul selectat este reintrodus în setul de date din care a fost extras; (3) pașii 1 și 2 se repetă de un număr de ori egal cu numărul de cazuri pe care dorim să le extragem. Procedând astfel, un caz poate intra de mai multe ori în eșantionul final, iar altul deloc. În cazul acestui tip de validare, eșantionarea aleatoare cu înlocuire este folosită pentru a selecta direct setul de date de instruire și, indirect, setul de date de testare. Prima dată se selectează cazurile care vor intra în setul de instruire, de obicei 70% din cazurile originale. Foarte important, instruirea modelului nu se realizează pe exact cele 70% din cazurile selectate în setul de instruire inițial, ci pe un eșantion de tip bootstrap selectat din acest set inițial. Procesul de selecție a unui set de date de instruire este realizat de mai multe ori (de obicei 10), modelul fiind estimat pentru fiecare dintre aceste situații. Cazurile care nu au fost alocate setului de instruire inițial sunt alocate setului de testare, adică aproximativ restul de 30%[4] dintre cazurile originale. Calitatea predicției este calculată pentru fiecare situație, rezultatele fiind agregate, de unde și posibilitatea de a estima stabilitatea modelului de predicție. O schemă simplă care ilustrează logica acestui tip de validare este prezentată în Figura 3.4-1.

Figura 3.4-1. Logica validării bootstrapping (algoritmul din RapidMiner)



[4] În practică, datorită selecției cu înlocuire, rămân relativ mai multe cazuri pentru setul de testare.

O variantă alternativă a acestui tip de validare este prezentată în Figura 3.4-2. În acest exemplu, setul de date conține 10 cazuri (x_1-x_{10}), iar numărul de eșantioane extrase este 3. Observăm că unele cazuri intră în fiecare eșantion (set de date de instruire), altele în doar unul, iar altele în niciunul. Cazurile care nu sunt selectate în setul de instruire sunt alocate setului de test (de exemplu, la setul 1, cazurile 3, 7 și 10 nu apar în setul de instruire, doar în cel de test). Această variantă este de preferat deoarece folosește de fiecare dată toate cazurile disponibile în setul de date original (dacă un caz nu intră în setul de instruire va apărea automat în cel de testare).

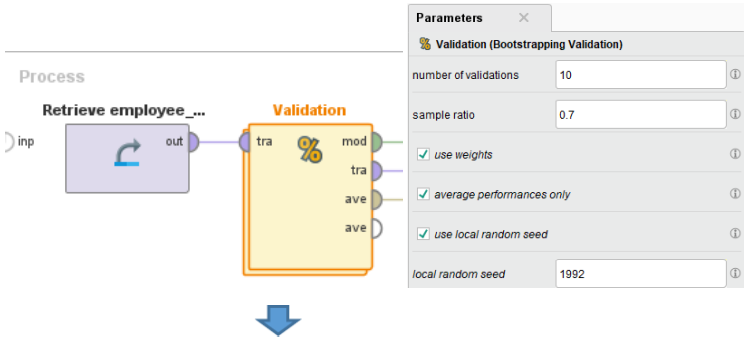
Figura 3.4-2. Logica validării bootstrapping (alternativă)

Sursa: https://rasbt.github.io/mlxtend/user_guide/evaluate/bootstrap_point632_score/

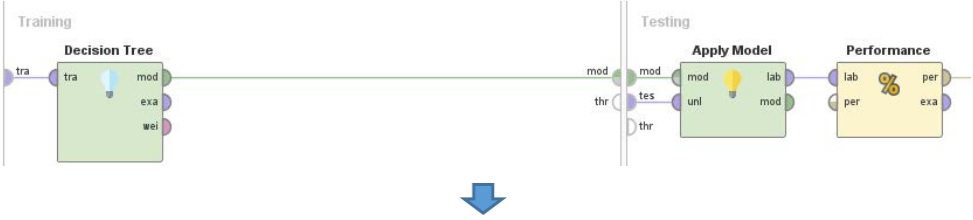
În exemplul din Figura 3.4-3 am realizat 10 validări, fiecare dintre acestea folosind două sub-seturi (70% vs. 30% dintre cazurile din setul de date). Pentru a obține aceleași rezultate de fiecare dată atunci când procesul este rulat din nou, am marcat opțiunea „use local random seed". Modelul a prezis corect, în medie, 82.44% din totalul cazurilor incluse în setul de date de testare, deci ne așteptăm ca, în producție, acuratețea modelului să fie apropiată de 83% (chiar dacă mai mică). Stabilitatea acestei predicții este foarte mare (abaterea medie de la media acurateței celor 10 predicții este ± 0.96%). O stabilitate ridicată este importantă deoarece crește probabilitatea de generalizare a modelului de predicție (șansele ca acesta să performeze aproape la fel de bine și pe alte seturi de date).

Figura 3.4-3. Validarea boostrapping a unui model de predicție

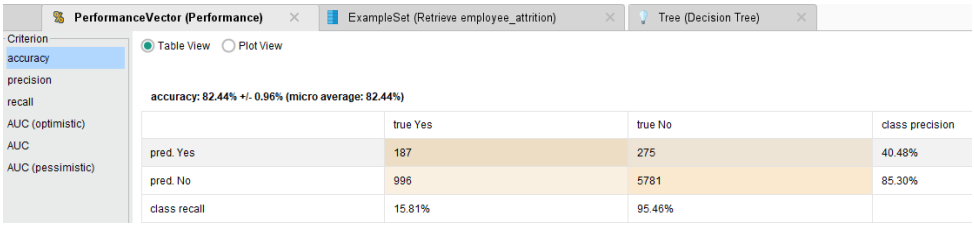
Pasul 1:
Conectăm setul de date și operatorul Bootstrapping Validation. Setăm valorile parametrilor conform imaginii.



Pasul 2:
În interiorul operatorului Bootstrapping Validation conectăm operatorii din imagine.



Rezultate:
Obținem trei tipuri de rezultate: confusion matrix (imaginea de mai jos); setul de date utilizat pentru instruirea și validarea modelului; modelul de predicție utilizat (Decision Tree în acest caz).



	true Yes	true No	class precision
pred. Yes	187	275	40.48%
pred. No	996	5781	85.30%
class recall	15.81%	95.46%	

3.5. Cum realizăm corect validarea încrucișată?

Să presupunem că am realizat validarea și testarea unui model respectând pașii și condițiile descrise anterior. Chiar și în acest caz, atunci când aplicăm modelul pe un alt set de date, se întâmplă frecvent să obținem o calitate a predicțiilor ceva mai redusă decât cea așteptată, adică o eroare de predicție puțin mai mare decât eroarea observată în cazul modelului validat. Sursa acestei diferențe poate fi modalitatea parțial greșită în care a fost validat modelul (RapidMiner, 2017).

Principala condiție ce trebuie îndeplinită pentru a realiza corect validarea încrucișată vizează separarea completă a celor două seturi de date (instruire și test). Există însă și alte forme de contaminare, mai subtile. Chiar dacă acestea nu vor influența modelul de predicție obținut, adesea pot produce iluzia unei performanțe așteptate mai bune a acestuia (acuratețea obținută în urma validării va fi în general mai mare comparativ cu acuratețea reală, cea obținută pe date noi). Simplu spus, orice acțiune asupra datelor utilizate pentru construirea unui model de predicție trebuie să aibă loc în interiorul operatorului de validare, nu în exteriorul acestuia, respectiv acțiunea să fie aplicată doar setului de date de instruire, nu și celui de test. Prin „acțiune asupra datelor" ne referim la procese precum normalizarea atributelor, optimizarea parametrilor modelului sau optimizarea selecției atributelor. Pentru fiecare dintre aceste situații prezentăm în continuare comparativ acuratețea clasificării pe mai multe seturi de date, respectiv pe setul nostru de date.

Impactul normalizării atributelor asupra performanței anticipate

În Tabelul 3.5-1 am prezentat erorile de predicție pentru fiecare dintre cele două modalități de validare încrucișată – normalizare în exteriorul vs. interiorul validării – în cazul a patru seturi de date, folosind trei clasificatori. Cu excepția clasificatorului Random Forest, care nu depinde de normalizarea datelor, în toate celelalte situații eroare de clasificare obținută în cazul normalizării externe validării este mai mică față de cea obținută în cazul normalizării interne validării. Supra-estimarea medie a calității modelelor este de 3-4 puncte procentuale, sau, altfel spus, dacă alegem să normalizăm datele în exteriorul procesului de validare vom obține modele aparent mai bune, cu erori de predicție mai mici decât cele reale, cele care vor fi obținute ulterior în procesul de producție. Important de menționat, validarea corectă nu ne ajută să obținem modele mai bune (acestea sunt identice), doar ne oferă o perspectivă relativ mai corectă cu privire la calitatea anticipată a modelului.

Tabelul 3.5-1. Impactul modalității de realizare a validării încrucișate asupra performanței anticipate –normalizarea atributelor

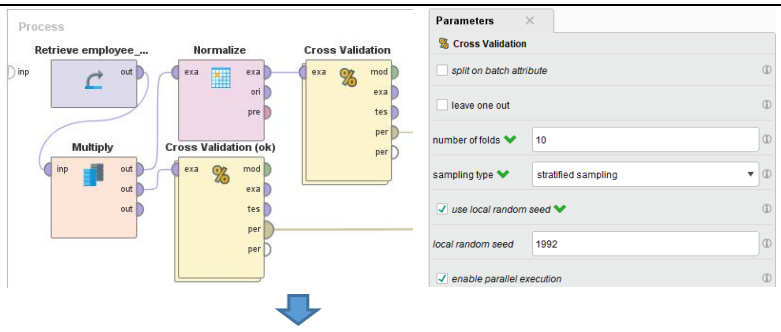
Model predicție	**Random Forest**		**Regresie Logistică**		**5-NN**	
Setul de date	Greșit	Corect	Greșit	Corect	Greșit	Corect
Diabetes	27.5	27.5	24.1	29.6	26.6	28.1
Ionosphere	11.4	11.4	15.7	16.8	15.4	19.4
Sonar	30.7	30.7	24.0	32.2	18.7	26.0
Wine	16.5	16.5	11.1	11.9	11.6	12.2
Media erorilor		**0.0**		**3.9**		**3.4**

Sursa: (RapidMiner, 2017); valorile reprezintă erorile de predicție (%); greșit = normalizare exterioară validării; corect = normalizare interioară validării.

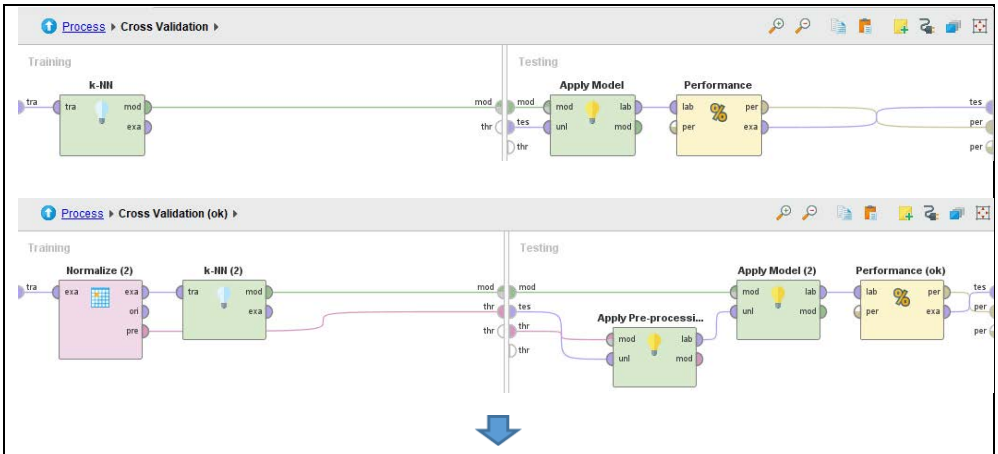
În Figura 3.5-1 am ilustrat comparativ cele două modalități de validare încrucișată (greșită vs. corectă) în cazul setului „employee attrition". Observăm că în cazul validării greșite normalizarea datelor este realizată în afara operatorului de validare, în timp ce în cazul validării corecte normalizarea apare în interior, în fereastra de instruire. Prima dată datele de instruire sunt normalizate, apoi sunt modelate (KNN), iar apoi modelul predictiv obținut și parametrii necesari pentru normalizarea datelor sunt livrați fazei de testare. Aici, prima dată sunt transformate datele, apoi este realizată predicție și este calculată calitatea acesteia.

Figura 3.5-1. Impactul modalității de realizare a validării încrucișate asupra performanței anticipate –normalizarea atributelor

Pasul 1:
Conectăm setul de date și operatorii din imagine.
Setăm valorile parametrilor.



Pasul 2:
În interiorul operatorului Cross Validation / Cross Validation (ok) conectăm operatorii din imagine. Observăm că în prima imagine (validare greșită) normalizarea datelor se realizează înainte de validare, pentru tot setul de date. În a doua imagine (validare corectă), normalizarea se realizează în interiorul operatorului de validare, în faza de instruire, ulterior fiind aplicată și setului de date de test.



Rezultate:

Observăm că acuratețea clasificării celor două modele este aproape aceeași, fiind puțin mai mare în cazul în care normalizarea datelor s-a realizat exterior validării. În alte condiții (alt set de date, alt clasificator), probabil diferența ar fi fost mai mare.

Validare greșită (normalizarea datelor este exterioară validării)

accuracy: 83.88% +/- 1.98% (micro average: 83.88%)

	true Yes	true No	class precision
pred. Yes	28	28	50.00%
pred. No	209	1205	85.22%
class recall	11.81%	97.73%	

Validare corectă (normalizare datelor se realizează în interiorul validării)

accuracy: 83.74% +/- 2.01% (micro average: 83.74%)

	true Yes	true No	class precision
pred. Yes	27	29	48.21%
pred. No	210	1204	85.15%
class recall	11.39%	97.65%	

Impactul optimizării parametrilor asupra performanței anticipate

În Tabelul 3.5-2 am prezentat erorile de predicție pentru fiecare dintre cele două modalități de validare încrucișată – optimizarea parametrilor modelului în exteriorul vs. interiorul validării – pentru aceleași seturi de date și clasificatori. Valorile din tabel reprezintă erorile de predicție. În majoritatea situațiilor validarea încrucișată corectă duce la estimări puțin mai mari ale erorilor de predicție (acestea vor fi însă mai apropiate de cele reale).

Tabelul 3.5-2. Impactul modalității de realizare a validării încrucișate asupra performanței anticipate –optimizarea parametrilor

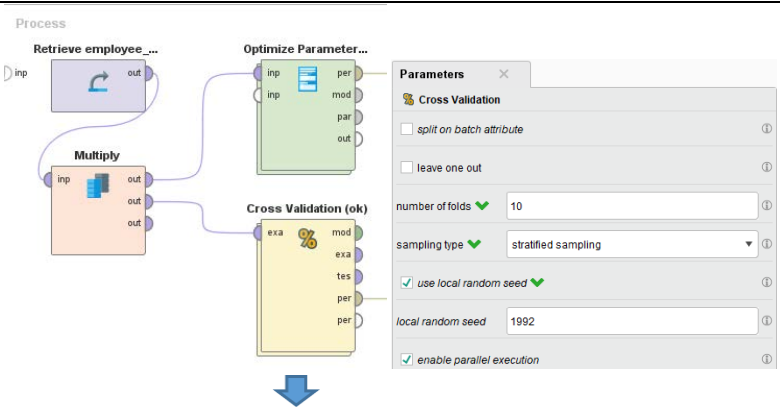
Model predicție	Random Forest		Regresie Logistică		5-NN	
Setul de date	Greșit	Corect	Greșit	Corect	Greșit	Corect
Diabetes	27.1	27.5	23.2	23.6	24.5	25.5
Ionosphere	8.3	9.4	12.0	13.4	16.0	16.5
Sonar	25.5	25.5	22.1	22.1	19.8	19.8
Wine	15.5	15.7	10.2	10.7	16.4	16.5
Media erorilor		**0.4**		**0.6**		**0.4**

Sursa: (RapidMiner, 2017); valorile reprezintă erorile de predicție (%); greșit = optimizare exterioară validării; corect = optimizare interioară validării.

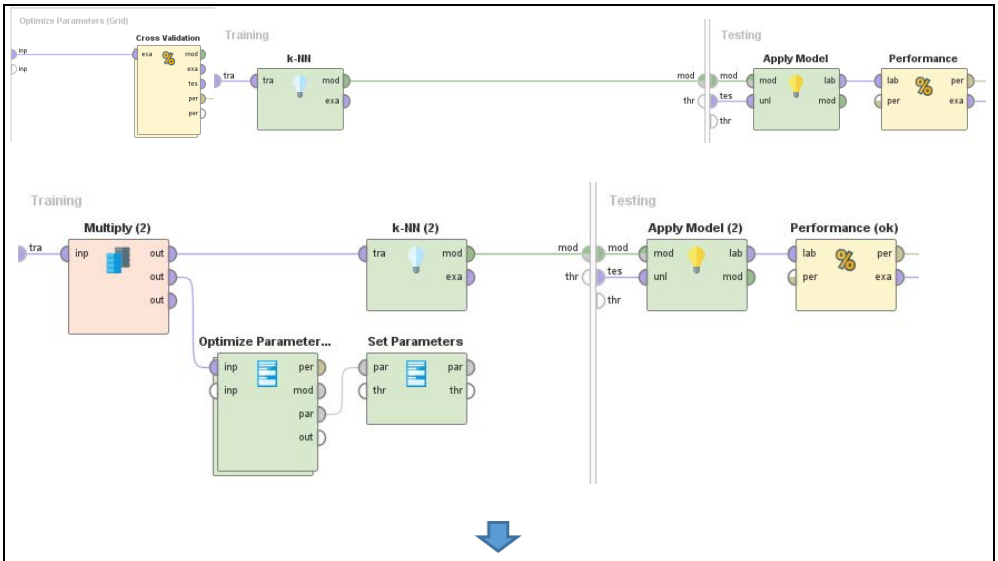
În Figura 3.5-2 am ilustrat comparativ cele două modalități de validare încrucișată (greșită vs. corectă) în cazul setului „employee attrition". Observăm că în cazul validării greșite optimizarea parametrilor este realizată în afara operatorului de validare, în timp ce în cazul validării corecte optimizarea se realizează în interiorul validării, în fereastra de instruire, folosind doar setul de date de instruire. Și în acest caz observăm că validarea greșită este responsabilă de o supra-estimarea a calității predicțiilor.

Figura 3.5-2. Impactul modalității de realizare a validării încrucișate asupra performanței anticipate –optimizarea parametrilor

Pasul 1:
Conectăm setul de date și operatorii din imagine.
Setăm valorile parametrilor.



Pasul 2:
În cazul validării greșite (prima imagine), operatorul Cross Validation apare în interiorul operatorului Optimize Parameters. În cazul validării corecte (a doua imagine), optimizarea se realizează în interiorul operatorului de validare, în faza de instruire, ulterior fiind aplicată și setului de date de test.



Rezultate:

Observăm că în cazul validării greșite acuratețea clasificării este relativ mai mare, iar precizia estimării este mai mică. Ambele duc la o supra-estimare a calității predicțiilor.

Validare greșită (normalizarea datelor este exterioară validării)

accuracy: 84.29% +/- 0.81% (micro average: 84.29%)

	true Yes	true No	class precision
pred. Yes	14	8	63.64%
pred. No	223	1225	84.60%
class recall	5.91%	99.35%	

Validare corectă (normalizare datelor se realizează în interiorul validării)

accuracy: 82.86% +/- 1.89% (micro average: 82.86%)

	true Yes	true No	class precision
pred. Yes	28	43	39.44%
pred. No	209	1190	85.06%
class recall	11.81%	96.51%	

Impactul optimizării selecției atributelor asupra performanței anticipate

În Tabelul 3.5-3 am prezentat erorile de predicție pentru fiecare dintre cele două modalități de validare încrucișată – optimizarea selecției atributelor în exteriorul vs. interiorul validării – pentru aceleași seturi de date și clasificatori. Valorile din tabel reprezintă erorile de predicție. În toate situațiilor validarea încrucișată corectă duce la estimări puțin mai mari a erorilor de predicție (acestea vor fi însă mai apropiate

de cele reale). În cazul a doi clasificatori subestimarea erorilor de predicție este destul de mare (3-4 puncte procentuale).

Tabelul 3.5-3. Impactul modalității de realizare a validării încrucișate asupra performanței anticipate – optimizarea selecției atributelor

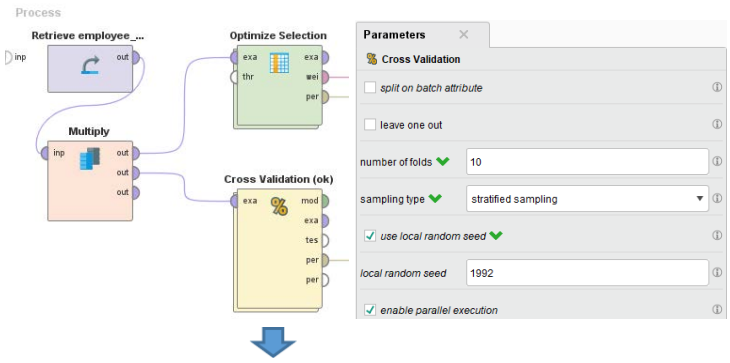
Model predicție	**Random Forest**		**Regresie Logistică**		**5-NN**	
Setul de date	Greșit	Corect	Greșit	Corect	Greșit	Corect
Diabetes	25.5	26.3	22.5	23.6	25.0	26.2
Ionosphere	7.1	13.7	16.5	17.7	8.8	13.1
Sonar	22.2	26.7	22.5	23.7	11.6	22.6
Wine	12.3	12.3	13.0	13.0	10.0	10.1
Media erorilor		**3.0**		**0.9**		**4.2**

Sursa: (RapidMiner, 2017); valorile reprezintă erorile de predicție (%); greșit = optimizare exterioară validării; corect = optimizare interioară validării.

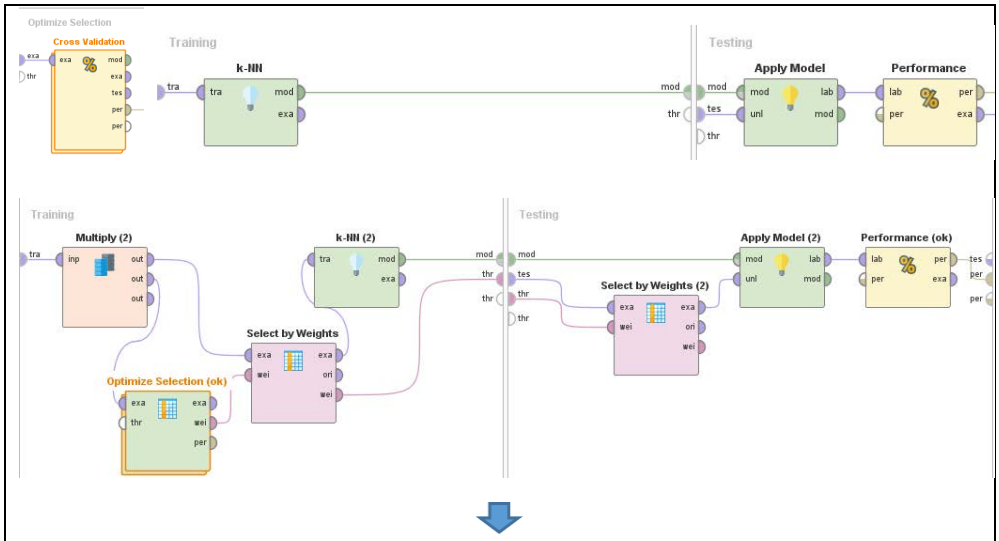
În Figura 3.5-3 am ilustrat comparativ cele două modalități de validare încrucișată (greșită vs. corectă) în cazul setului „employee attrition". Observăm că în cazul validării greșite optimizarea selecției atributelor este realizată în afara operatorului de validare, în timp ce în cazul validării corecte optimizarea se realizează în interiorul validării, în fereastra de instruire, folosind doar setul de date de instruire. Și în acest caz observăm că validarea greșită este responsabilă de o supra-estimarea a calității predicțiilor.

Figura 3.5-3. Impactul modalității de realizare a validării încrucișate asupra performanței anticipate –optimizarea selecției atributelor

Pasul 1:
Conectăm setul de date și operatorii din imagine.
Setăm valorile parametrilor.



Pasul 2:
În cazul validării greșite (prima imagine), operatorul Cross Validation apare în interiorul operatorului Optimize Parameters. În cazul validării corecte (a doua imagine), optimizarea se realizează în interiorul operatorului de validare, în faza de instruire, ulterior fiind aplicată și setului de date de test.



Rezultate:
Observăm că în cazul validării greșite acuratețea clasificării este foarte puțin mai mare (la fel precizia estimării). Ambele duc la o (foarte mică) supra-estimare a calității predicțiilor.

Validare greșită (normalizarea datelor este exterioară validării)

accuracy: 84.49% +/- 1.15% (micro average: 84.49%)

	true Yes	true No	class precision
pred. Yes	17	8	68.00%
pred. No	220	1225	84.78%
class recall	7.17%	99.35%	

Validare corectă (normalizare datelor se realizează în interiorul validării)

accuracy: 84.15% +/- 1.24% (micro average: 84.15%)

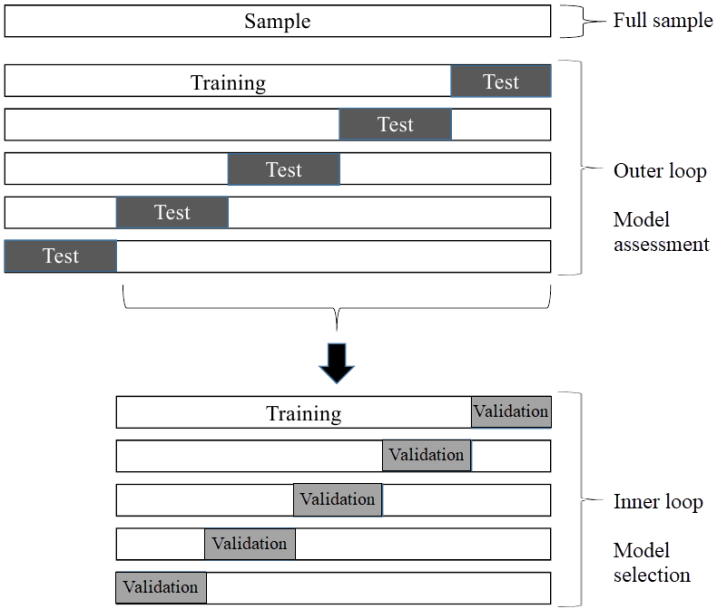
	true Yes	true No	class precision
pred. Yes	18	14	56.25%
pred. No	219	1219	84.77%
class recall	7.59%	98.86%	

Concluzie: validarea încrucișată de tip cuib (nested cross-validation)

În concluzie, nu este suficient să validăm încrucișat doar modelul de predicție / clasificare ci și toți ceilalți pași realizați anterior construirii modelului precum normalizarea datelor, optimizarea parametrilor și selecția predictorilor. Simplu spus, strategia corectă este să folosim validarea încrucișată de tip cuib (nested cross-validation). Pentru a realiza o astfel de validare folosim operatorul Cross Validation și includem în interiorul lui operatorul de optimizare a selecției predictorilor sau de optimizare a selecției valorilor hiper-parametrilor. Prin urmare modelul va conține

două bucle (loops) separate, unul pentru partea de evaluare a modelului și unul pentru partea de selecție / optimizare (Jacobucci et al., 2023, p. 55). O schemă ilustrativă a acestui tip de validare apare în Figura 3.5-4.

Figura 3.5-4. Validarea încrucișată de tip cuib (nested cross-validation)



Sursa: (Jacobucci et al., 2023, p. 55)

4. EVALUAREA UNUI MODEL DE CLASIFICARE

Uneori, evaluarea unui model este divizată în două componente majore: evaluarea calității modelului și evaluarea performanței modelului (performanța clasificării). Prin calitatea modelului avem în vedere măsurarea gradului în care modelul reușește concomitent să aibă o complexitate redusă și să reproducă datele care au fost folosite pentru producerea lui. Pentru a exprima numeric gradul de reproducere în raport cu complexitate, se folosesc măsuri statistice probabiliste precum AIC (Akaike Information Criterion) și BIC (Bayesian Information Criterion). Dat fiind faptul că nu există o modalitate general valabilă pentru a calcula numărul de parametri, BIC și AIC sunt specifice, particulare unui anumit model, deci nu le putem folosi pentru a compara între ele modele produse cu diferiți clasificatori, respectiv date diferite. Din această cauză, AIC și BIC nu sunt implementate ca operator generic în RapidMiner. Sunt implementate însă o serie de măsuri care măsoară performanța unui model de clasificare. Toate aceste măsuri au la bază datele din matricea de confuzie (confusion matrix). În acest capitol introducem conceptul de matrice de confuzie apoi prezentăm și ilustrăm indicii utilizați cel mai adesea pentru a măsura performanța unui model de clasificare.

4.1. Matricea de confuzie și măsuri ale performanței bazate pe aceasta

În acest sub-capitol vom prezenta elementele unei matrice de confuzie (confusion matrix) și o serie de măsuri utile (derivate din valorile acesteia) pentru evaluarea performanței unei model de clasificare. Pentru a ușura înțelegerea, vom discuta doar situația unei variabile de interes (dependente / prezise) de tip binominal (două categorii).

Matricea de confuzie (confusion matrix)

Matricea de confuzie prezintă numărul de cazuri clasificate corect, respectiv incorect, pentru fiecare dintre categoriile atributului de interes (variabila dependentă sau prezisă). În varianta cea mai simplă, considerând că variabila de interes are doar două categorii, matricea de confuzie arată precum în Figura 4.1-1.[1] Acronimele au următoarea semnificație:

- **N** (Negative) = număr cazuri negative (NU / 0 / –) în realitate;
- **P** (Positive) = număr cazuri pozitive (DA / 1 / +) în realitate;
- **PN** (Predicted Negative) = număr cazuri prezise ca negative (NU / 0 / –);
- **PP** (Predicted Positive) = număr cazuri prezise ca pozitive (DA / 1 / +);
- **TP** (True Positive) = număr cazuri prezise corect ca aparținând clasei pozitive (DA / 1 / +);
- **TN** (True Negative) = număr cazuri prezise corect ca aparținând clasei negative (NU / 0 / –);
- **FP** (False Positive) = număr cazuri prezise incorect ca aparținând clasei pozitive (DA / 1 / +);
- **FN** (False Negative) = număr cazuri prezise incorect ca aparținând clasei negative (NU / 0 / –).

Figura 4.1-1. Matricea de confuzie (confusion matrix): forma generală și un exemplu fictiv

Realitate / Predicție	Negativ (**N**) Nu / 0 / –	Pozitiv (**P**) Da / 1 / +
Negativ (**PN**) Nu / 0 / –	realitate – predicție – **TN** (True Negative)	realitate + predicție – **FN** (False Negative)
Pozitiv (**PP**) Da / 1 / +	realitate – predicție + **FP** (False Positive)	realitate + predicție + **TP** (True Positive)

Realitate / Predicție	Negativ (**N**) Nu / 0 / –	Pozitiv (**P**) Da / 1 / +
Negativ (**PN**) Nu / 0 / –	400	200
Pozitiv (**PP**) Da / 1 / +	300	100

În exemplul din Figura 4.1-1 numărul de cazuri prezise corect ca aparținând clasei pozitive (TP) este 100, numărul de cazuri prezise corect ca aparținând clasei negative (TN) este 400, numărul de cazuri prezise incorect ca aparținând clasei pozitive (FP)

[1] Funcție de sursa bibliografică, reprezentarea matricei de confuzie poate diferi: uneori, clasa pozitivă este pe prima poziție și cea negativă pe a doua, alteori invers (varianta din figura noastră); uneori, predicția este pe coloane și situația reală pe linii, alteori invers (varianta din figura noastră). De asemenea, denumirile celor patru combinații diferă uneori: TN este numită și „Correct rejection", TP „Hit", FN „Miss" sau „Type II error", FP „False alarm" sau „Type I error".

este 300, iar numărul de cazuri prezise incorect ca aparținând clasei negative (FN) este 200. Predicțiile incorecte (FP și FN) sunt denumite uneori și erori de clasificare, cazurile FP fiind erori de tip I (type I error), iar cazurile FN erori de tip II (type II error). În aceeași figură, numărul de cazuri prezise ca negative este 600 (400+200), numărul de cazuri prezise ca pozitive este 400 (300+100), numărul real de cazuri negative este 700 (400+300), iar numărul real de cazuri pozitive este 300 (200+100). Evident, numărul total de cazuri este 1000 (400+300+200+100).

Acuratețea (accuracy) și eroarea de clasificare (classification error)

Două dintre cele mai simple și utilizate măsuri ale performanței clasificării sunt acuratețea clasificării, respectiv reversul acesteia, eroarea clasificării. Acuratețea (ACC) indică ponderea cazurilor clasificate corect de model. De exemplu, dacă modelul clasifică corect 500 cazuri din totalul celor 1000, acuratețea este 0.5 sau 50% (Tabelul 4.1-2). Numărul cazurilor clasificate corect este egal cu suma dintre TP (true positive) și TN (true negative) (Tabelul 4.1-1). Cu cât ACC ia valori mai apropiate de 1 (100%), cu atât modelul este mai bun, are o performanță mai mare. Eroarea de clasificare (classification error - ERR) reprezintă ponderea cazurilor clasificate incorect de model. De exemplu, dacă modelul clasifică incorect 500 cazuri din totalul celor 1000, eroarea este 0.5 sau 50% (Tabelul 4.1-2). Numărul cazurilor clasificate incorect este egal cu suma dintre FP (false positive) și FN (false negative). Cu cât ERR ia valori mai apropiate de 0 (0%), cu atât modelul este mai bun, are o performanță mai mare. Desigur, ERR este egal cu diferența dintre unitate și ACC (1 – ACC sau 100% – ACC).

Tabelul 4.1-1. Matricea de confuzie (variabilă binominală) și câteva măsuri ale performanței clasificării

Realitate / Predicție	Negativ (**N**) Nu / 0 / –	Pozitiv (**P**) Da / 1 / +	
Negativ (**PN**) Nu / 0 / –	**TN** (400)	**FN** (200)	NPV PRC – $= \frac{TN}{PN} = \frac{TN}{TN+FN}$
Pozitiv (**PP**) Da / 1 / +	**FP** (300)	**TP** (100)	PPV PRC + $= \frac{TP}{PP} = \frac{TP}{TP+FP}$
	TNR / SPC / REC – $= \frac{TN}{N} = \frac{TN}{TN+FP}$	TPR / SEN / REC + $= \frac{TP}{P} = \frac{TP}{TP+FN}$	ACC $= \frac{TP+TN}{P+N} = \frac{TP+TN}{TP+TN+FP+FN}$
			ERR $= 1 - ACC$

Acuratețea (respectiv eroarea) reprezintă o măsură globală a performanței clasificării. Valorile luate de aceste măsuri depind foarte mult de ponderea claselor.

În cazul în care cele două clase (categorii ale variabilei de interes) au o pondere foarte diferită (unbalanced dataset), acuratețea și eroarea pot lua mult mai ușor valori extreme (apropiate de 1, respectiv 0). De exemplu, dacă doar 1% dintre cazurile din setul de date sunt pozitive (evenimentul de interes are loc), modelul obține automat o acuratețe a clasificării de 99% (respectiv o eroare de 1%). Această acuratețe ar fi de altfel obținută și dacă doar am ghici apartenența cazurilor la clase fără să știm nimic altceva despre ele (am prezice că toate cazurile sunt negative = evenimentul nu are loc). Pentru o situație de acest tip, valoarea extrem de mare a acurateței clasificării este consecința faptului că 99% dintre cazuri sunt negative. Simplu spus, modelul nostru va clasifica corect mult mai ușor cazurile negative, iar cum acestea sunt dominante, acuratețea va fi automat mare. Pentru astfel de situații, dat fiind interesul nostru pentru identificarea cazurilor din clasa pozitivă, o acuratețe de 99% nu ne este de niciun folos. Pentru situații de acest gen e mai util dacă folosim ca măsuri ale performanței clasificării măsuri care țin cont de ponderea inegală a claselor precum acuratețea echilibrată (BACC = balanced accuracy) (Tabelul 4.1-2), F_1 sau, mai bine, MCC. Alte soluții posibile sunt echilibrarea setului de date, respectiv atribuirea unor costuri diferite erorilor de clasificare.[2]

Evaluarea performanței clasificării: măsuri simple

Pornind de la cele patru tipuri de cazuri (TN, TP, FN, FN) putem calcula o serie de măsuri ale performanței clasificării (Tabelul 4.1-2). Cu referire strict la clasa pozitivă (acestea măsuri se calculează similar și în relație cu clasa negativă), probabil cele mai folosite două măsuri sunt Reamintirea și Precizia (denumirile folosite în matricea de confuzie din RapidMiner sunt Recall și Precision).

Recall (REC) se mai numește și Sensitivity, Hit rate, Power, Probability of detection sau TPR (True Positive Rate). Măsura Recall / Reamintire se calculează simplu ca raport între numărul de cazuri clasificate corect ca pozitive și numărul de cazuri pozitive (TP/P). Pentru a ușura asocierea între denumire și formulă (interpretare) ne putem gândi că măsura Recall indică ponderea cazurilor pozitive pe care un model de clasificare reușește să și le reamintească, să le redescopere.

Precision (Precizie) (PRC) sau Positive Predictive Value (PPV) se calculează ca raport între numărul de cazuri pozitive prezise corect și numărul total de cazuri prezise ca pozitive (TP / PP). Pentru a ușura asocierea între denumire și formulă (interpretare) ne putem gândi că măsura Precision / Precizie ne arată cât de bine „țintește" un model de clasificare atunci când prezice că un caz este pozitiv.

O altă măsură populară, respectiv utilă în relație cu alte măsuri (curba ROC de exemplu) este Specificity (Specificitate) (SPC). Se mai numește și Recall în relație cu clasa

[2] Ambele teme vor fi discutate în volumul trei al acestui manual.

negativă sau True Negative Rate (TNR). Specificity se calculează ca raport între numărul de cazuri negative prezise corect și numărul total de cazuri negative (TN / N).

Toate aceste măsuri alături de alte câteva sunt prezentate sintetic în Tabelul 4.1-2. Pentru a evalua performanța unui model de clasificare folosim, funcție de interes, câteva dintre aceste măsuri. Una singură ar oferi o perspectivă limitată asupra performanței. De exemplu, o valoare mare a TPR / REC ne va spune prea puține despre capacitatea modelului de a detecta cazurile negative (TNR) (Wendler & Gröttrup, 2021, p. 766). În practică, măsurile sunt totuși legate între ele: de exemplu, atunci când creștem TPR (hit rate), FPR (false alarm) va crește automat, deci va trebui să facem un compromis (trade-off) între cele două măsuri, funcție de situația concretă pe care o analizăm și interesele practice.

Tabelul 4.1-2. Evaluarea performanței clasificării: măsuri simple [3]

Măsură	Descriere	Formulă	Exemplu	%
ACC = accuracy **CCR = Correct Classification Rate**	pondere cazuri corect clasificate (rată acurateță clasificare)	$= \frac{TP + TN}{P + N}$	$= \frac{100 + 400}{1000}$	50
ERR = error rate **Misclassification Rate**	pondere cazuri incorect clasificate (rată eroare clasificare)	$= \frac{FP + FN}{P + N}$	$= \frac{300 + 200}{1000}$	50
PRC – = precision – **NPV = negative predictive value**	pondere cazuri clasificate corect ca negative din cazuri clasificate ca negative	$= \frac{TN}{PN}$	$= \frac{400}{600}$	67
PRC + = precision + **PPV = positive predictive value**	pondere cazuri clasificate corect ca pozitive din cazuri clasificate ca pozitive	$= \frac{TP}{PP}$	$= \frac{100}{400}$	25
TNR = true negative rate **REC – = recall –** **SPC = specificity**	pondere cazuri clasificate corect ca negative din cazuri negative	$= \frac{TN}{N}$	$= \frac{400}{700}$	57
TPR = true positive rate **REC + = recall +** **SEN = sensitivity** **Probability of detection** **Hit rate** **Power**	pondere cazuri clasificate corect ca pozitive din cazuri pozitive	$= \frac{TP}{P}$	$= \frac{100}{300}$	33
FPR = false-positive rate given true negative **Fallout** **Probability of false alarm**	pondere cazuri clasificate greșit ca pozitive din cazuri negative	$= \frac{FP}{N}$	$= \frac{300}{700}$	43
FNR = false-negative rate given true positive **Miss rate**	pondere cazuri clasificate greșit ca negative din cazuri pozitive	$= \frac{FN}{P}$	$= \frac{200}{300}$	67
FDR = false discovery rate **False-positive rate given classified positive**	pondere cazuri clasificate greșit ca pozitive din cazuri clasificate ca pozitive	$= \frac{FP}{PP}$	$= \frac{300}{400}$	75
FOR = false omission rate **False-negative rate given classified negative**	pondere cazuri clasificate greșit ca negative din cazuri clasificate ca pozitive	$= \frac{FN}{PN}$	$= \frac{200}{600}$	33
BACC = balanced accuracy (acurateță echilibrată)	pondere cazuri corect clasificate (echilibrat)	$= \frac{TPR + TNR}{2}$	$= \frac{33 + 57}{2}$	45

[3] O parte dintre măsurile de performanță a clasificării prezentate aici pot fi calculate și online, la adresele: https://onlineconfusionmatrix.com/ sau https://bci-lab.hochschule-rhein-waal.de/en/acc.html.

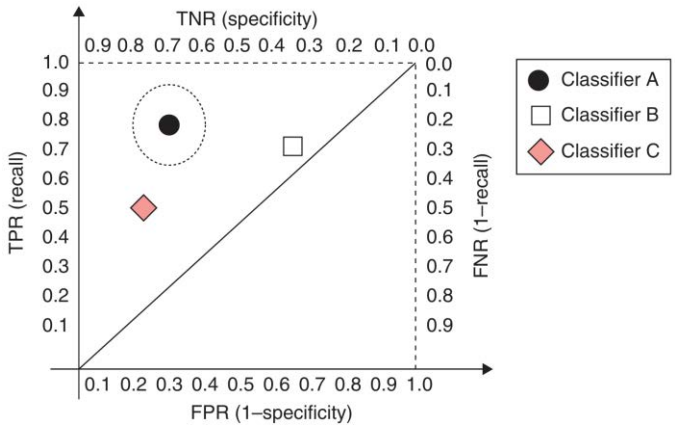
(continuare)

Măsură	Descriere	Formulă	Exemplu	Valoare
Prevalence	Incidența cazurilor +	$= \frac{P}{P+N}$	$= \frac{300}{1000}$	30
Positive likelihood ratio (LR+)	Rata de verosimilitate asociată cazurilor +	$= \frac{TPR}{FPR}$	$= \frac{33}{43}$	0.78
Negative likelihood ratio (LR−)	Rata de verosimilitate asociată cazurilor –	$= \frac{FNR}{TNR}$	$= \frac{67}{57}$	1.17
Diagnostic odds ratio (DOR)	Raportul șanselor	$= \frac{LR+}{LR-}$	$= \frac{0.78}{1.17}$	0.67
Prevalence threshold	Pragul prevalenței	$= \frac{\sqrt{TPR * FPR} - FPR}{TPR - FPR}$	$= \frac{\sqrt{0.33 * 0.43} - 0.43}{0.33 - 0.43}$	0.60

** Valorile de pe ultimele două coloane exemplifică formula și valorile obținute în cazul matricei de confuzie din Tabelul 4.1-1.*

Măsurile prezentate anterior pot fi folosite pentru a evalua comparativ și performanța mai multor clasificatori (modele). Un exemplu simplu, fictiv, este prezentat în Figura 4.1-2. Fiecare dintre cele patru axe reprezintă o măsură simplă a performanței (TPR, FPR, TNR și FNR). Performanța fiecărui clasificator este reprezentată printr-un semn distinct. Conform valorilor observate, clasificatorul A este cel mai performant. Relativ la clasa de interes (pozitivă), clasificatorul A identifică corect aproape 80% dintre cazuri (are cea mai mare valoare TPR, mai mare decât a clasificatorului B) și face simultan un număr relativ mic de erori de clasificare (valoarea FPR este doar puțin mai mare comparativ cu cea observată în cazul clasificatorului C).

Figura 4.1-2. Un exemplu fictiv de comparare a performanței unor clasificatori



Sursa: (Moreira et al., 2019, p. 202)

Evaluarea performanței clasificării: măsuri complexe

Măsurile discutate în secțiunea anterioară oferă fiecare în parte o perspectivă relativ limitată asupra performanței. În această secțiune vom prezenta câteva măsuri care țin cont simultan de mai multe măsuri simple, le combină (Tabelul 4.1-3).

O primă măsură complexă este **coeficientul Kappa** a lui Cohen (Cohen's kappa). Kappa a fost propus în alt context, dar este folosit adesea și pentru evaluarea generală a performanței clasificării. Kappa pune în relație acuratețea observată și cea așteptată și este egal cu raportul dintre diferența celor două și 1 minus Acuratețea așteptată. Prin urmare, atunci când Acuratețea observată este mult mai mare comparativ cu cea așteptată, numărătorul și numitorul tind spre 1, prin urmare Kappa se apropie de unitate, deci modelul are o performanță excelentă.

Coeficientul Youden (numit și informedness sau DeltaP') ține cont simultan de măsurile Sensitivity (Recall + / TPR) și Specificity (Recall − / TNR). Formula este foarte simplă: din suma celor două măsuri Recall se scade 1. Dat fiind faptul că fiecare dintre măsurile Recall poate lua maximum valoarea 1 (indică un model care clasifică perfect datele), valoarea maximă a lui Youden este tot 1 (clasificare perfectă).

Coeficientul psep (numit și markedness sau DeltaP) ia în calcul tot două măsuri, PPV (Precision +) și NPV (Precision −). Și aici formula este simplă: din suma celor două măsuri Precision se scade 1. Dat fiind faptul că fiecare dintre măsurile Precision poate lua maximum valoarea 1 (indică un model care clasifică perfect datele), valoarea maximă a lui psep este tot 1 (clasificare perfectă).

Youden și psep au la bază fiecare doar una dintre tipurile de măsuri Recall și Precision, relativ la ambele clase (pozitivă și negativă). **Scorul F (F_1)** ia în calcul ambele tipuri de măsuri, dar doar relativ la clasa pozitivă (reprezintă un compromis între aceste măsuri). Dat fiind faptul că putem defini oricare dintre cele două clase ca pozitivă, valoarea lui F_1 va depinde de această alegere. Concret, F_1 este egal cu media armonică dintre măsurile Precision + și Recall +. Spre deosebire de media aritmetică, media armonică penalizează situațiile în care valorile pentru care calculăm media sunt foarte diferite. Astfel, pentru exemplul din tabel (precizie = 35.3, reamintire = 7.6), media aritmetică este 21.5, iar media armonică (F_1) este 12.5. Dacă cele două valori ar fi fost mai apropiate (de exemplu, precizie = 25.3, reamintire = 17.6), media aritmetică ar fi fost tot 21.5, iar media armonică (F_1) ar fi

fost 20.8, deci sensibil mai apropiată. Și în cazul acestei măsuri, valoarea 1 indică un model de clasificare perfect.

Coeficientul de corelație al lui Matthew (MCC) este o măsură și mai complexă, în formula de calcul a acestuia intrând 4 sau 8 măsuri simple, funcție de formula folosită (Tabelul 4.1-3). Similar cu coeficientul de corelație, MCC ia valori în intervalul [-1;1], unde 1 semnifică o clasificare perfectă (0 indică un model la fel de bun precum modelul naiv, iar -1 un model total greșit). Pentru ca MCC să ia o valoare mare, adică să indice o clasificare bună, modelul trebuie să clasifice cât mai multe dintre cazurile pozitive, respectiv negative, corect, indiferent de ponderea celor două clase (pozitivă / negativă). Dat fiind faptul că MCC ia în calcul toate cele patru celule ale matricei de confuzie (TN, TP, FN, FP), valoarea acestuia nu este atât de influențată de ponderea diferită a claselor și / sau de felul în care definim clasa pozitivă, prin urmare este o măsură mai bună comparativ cu ACC, Kappa și F_1 (Chicco & Jurman, 2020). De altfel, MCC a fost propus tocmai pentru a evalua performanța în cazul unor clase dezechilibrate ca pondere.

Fowlkes-Mallows Index (FMI) este egal cu media geometrică a măsurilor simple PPV și TPR. Dat fiind faptul că ambele măsuri țin cont doar de clasa pozitivă, valoarea FMI va fi dependentă și de ponderea acestei clase. Threat Score (TS), numit și Critical Success Index (CSI) sau Jaccard Index este o altă măsură care evaluează calitatea predicției relativ la cazurile pozitive. Formula este simplă: raportul dintre TP și suma cazurilor TP, FN și FP. Pentru ambele măsuri, valoarea 1 semnifică o clasificare perfectă (valoarea minimă este 0). Aceste măsuri nu sunt calculate de niciunul dintre operatorii de evaluare a performanței clasificării din RapidMiner.

Alte două măsuri utile, AUC și lift, sunt prezentate și ilustrate în sub-capitolele 0 și 4.4. Pentru a înțelege aceste măsuri e necesară o prezentare relativ mai extinsă care include și utilizarea unor grafice dinamice. Menționăm aici doar valorile relativ la care măsurile pot fi interpretate: în cazul AUC valoarea 1 indică o clasificare perfectă; în cazul lift, cu cât valoarea este mai mare decât 1, cu atât modelul este mai performant.

Tabelul 4.1-3. Evaluarea performanței clasificării: măsuri complexe

Măsură	Formulă	Exemplu	Valoare
Informedness / Youden's J statistic / Youden's index	$= TPR + TNR - 1$ $= SEN + SPC - 1$	= 0.33+0.57-1	-0.10
Markedness (MK) / deltaP (Δp) / PSEP	= PPV+NPV-1	= 0.25+0.67-1	-0.08
F_1 Score	$= \frac{2PPV * TPR}{PPV + TPR} = \frac{2TP}{2TP + FP + FN}$	$= \frac{200}{200 + 300 + 200}$	0.29
Fowlkes-Mallows index (FMI)	$= \sqrt{PPV * TPR}$	$= \sqrt{0.25 * 0.33}$	0.29
Matthews correlation coefficient (MCC) / phi coefficient[4]	$= \sqrt{TPR * TNR * PPV * NPV}$ $- \sqrt{FNR * FPR * FOR * FDR}$	$= \sqrt{0.33 * 0.57 * 0.25 * 0.67}$ $- \sqrt{0.67 * 0.43 * 0.33 * 0.75}$	-0.09
Threat score (TS) / critical success index (CSI) / Jaccard index	$= \frac{TP}{TP + FN + FP}$	$= \frac{100}{100 + 300 + 200}$	0.17
Cohen's kappa	$= \frac{ACC_o - ACC_e}{1 - ACC_e}$	ACC_o = acuratețea observată ACC_e = acuratețea așteptată	-0.09
	$ACC_e = \frac{(TP + FP) * (TP + FN) + (FN + TN) * (FP + TN)}{(P + N)^2}$		
	$= \frac{2 * (TP * TN - FN * FP)}{(TP + FP) * (FP + TN) + (TP + FN) * (FN + TN)}$		

** Valorile de pe ultimele două coloane exemplifică formula și valorile obținute în cazul matricei de confuzie din Tabelul 4.1-1.*

4.2. Calcularea măsurilor de performanță în RapidMiner

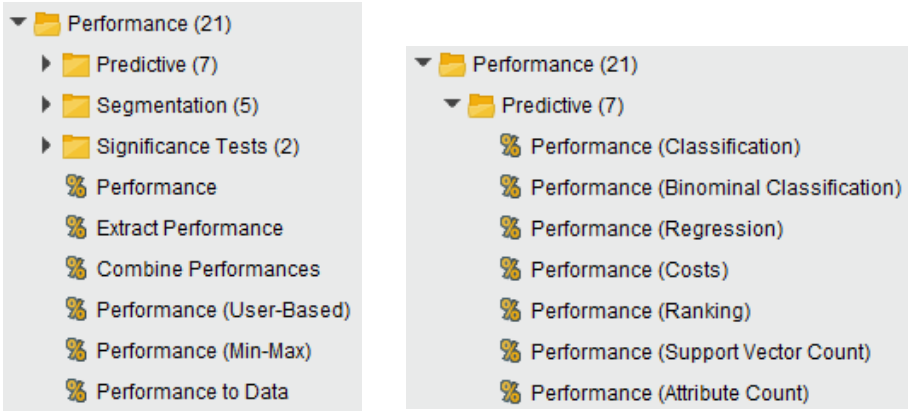
Categoria Performance a operatorilor RapidMiner include o serie de operatori utili pentru a calcula și opera cu diferite măsuri ale performanței modelelor (indiferent de tipul acestora: predicție, segmentare) (Figura 4.2-1). Pentru acest volum ne interesează modelele de predicție ale unor atribute categoriale, prin urmare vom discuta doar despre câțiva dintre operatorii din sub-categoria Predictive. Astfel, vom

[4] O formulă alternativă este: $\frac{TP*TN-FN*FP}{\sqrt{(TP+FP)*(TP+FN)*(TN+FP)*(TN+FN)}}$

discuta despre calcularea măsurilor de performanță pentru fiecare dintre următoarele trei tipuri de situații:

- variabilă dependentă multinominală - „Performance (Classification)”,
- variabilă dependentă binominală - „Performance (Binominal Classification)” și
- variabilă dependentă bi/multinominală + erori de clasificare cu costuri diferite - „Performance (Costs)”.

Figura 4.2-1. Operatorii din categoria Performance, respectiv Predictive Performance



Operatorul Performance (Binominal Classification)

Acest operator poate fi folosit pentru a calcula performanța unui model de clasificare a cărui variabilă de interes (prezisă) are doar două categorii / clase. Una dintre acestea este considerată a fi pozitivă, cealaltă negativă (putem defini care clasă este pozitivă și care negativă). Procesul care exemplifică utilizarea acestui operator este inclus în fișierele atașate volumului și în Figura 4.2-2. Operatorul „Performance (Binominal Classification)” poate calcula mai multe tipuri de măsuri ale performanței și anume (între paranteze apar valorile măsurilor de performanță ale modelului de clasificare inclus în procesul asociat acestei secțiuni):

Tabelul 4.2-1. Măsuri ale performanței – operatorul Performance (Binominal Classification)

Măsură	Interval	Valoare model
accuracy	[0;100]	82.86% +/- 1.50%
classification error	[0;100]	17.14% +/- 1.50%
Cohen's kappa	[0;1][5]	0.069 +/- 0.079
AUC	[0.5;1][6]	0.612 +/- 0.072
precision	[0;100]	35.29%
recall	[0;100]	7.59% +/- 6.26%
lift	[100;∞]	218.91%
fallout	[0;100]	2.68% +/- 2.02%
f measure	[0;100]	12.50%
false positive	[0;∞]	3.300 +/- 2.497
false negative	[0;∞]	21.900 +/- 1.524
true positive	[0;∞]	1.800 +/- 1.476
true negative	[0;∞]	120.000 +/- 2.449
sensitivity	[0;100]	7.59% +/- 6.26%
specificity	[0;100]	97.32% +/- 2.02%
youden	[0;1][7]	0.049 +/- 0.056
positive predictive value	[0;100]	35.29%
negative predictive value	[0;100]	84.58% +/- 0.83%
psep	[0;1][8]	0.199

** Clasa Da (pleacă) este definită ca pozitivă. Valorile cu roșu indică un model care clasifică perfect datele. Valorile de pe ultima coloană sunt cele calculate de RapidMiner pentru procesul asociat acestui sub-capitol. În tabel am trecut doar etichetele măsurilor așa cum apar în RapidMiner. Unitățile de măsură sunt cele utilizate în RapidMiner (de exemplu, valorile pentru măsurile lift și F sunt afișate ca procente).*

[5] Teoretic, valoarea minimă a lui Kappa este -1. În practică, Kappa ia foarte rar o valoare mai mică de 0 (în contextul analizei valorilor din matricea de confuzie, deci atunci când ne așteptăm să existe o asociere pozitivă între situația reală și predicție).

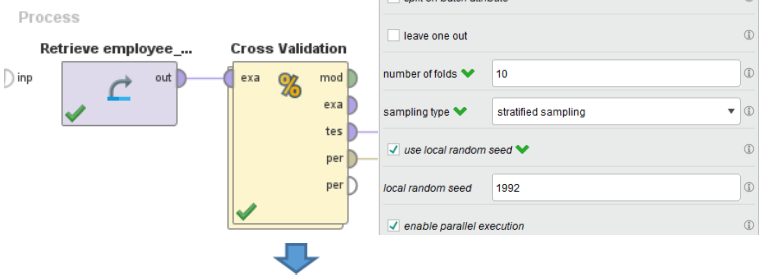
[6] Teoretic, valoarea minimă a lui AUC este 0. În practică, AUC ia foarte rar o valoare mai mică de 0.5.

[7] Teoretic, valoarea minimă este -1, dar, în practică, ia foarte rar o valoare mai mică de 0.

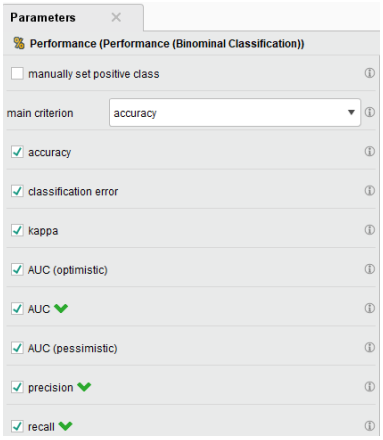
[8] Teoretic, valoarea minimă este -1, dar, în practică, ia foarte rar o valoare mai mică de 0.

Figura 4.2-2. Calcularea performanței predicției unui atribut de tip binominal

Pasul 1:
Conectăm setul de date și operatorii din imagine.
Setăm valorile parametrilor.



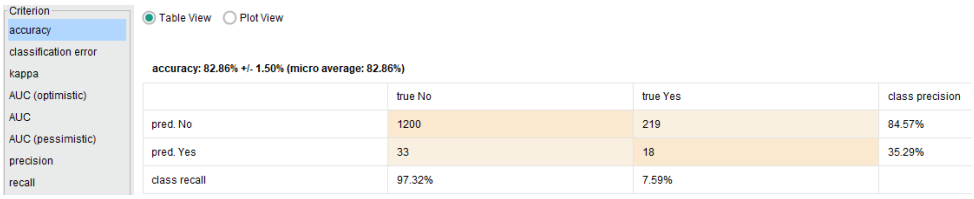
Pasul 2:
În interiorul operatorului de Validare, fereastra de testare a modelului de predicție, conectăm operatorii „Apply Model" și „Performance (Binominal Classification)".
La operatorul Performance alegem toate măsurile performanței disponibile (vezi imaginea alăturată). Dacă dorim să specificăm clasa (una dintre cele două categorii ale atributului Attrition) considerată a fi pozitivă, marcăm opțiunea „manually set positive class" și apoi indicăm clasa (în cazul de față, clasa pozitivă a fost definită anterior în setul de date).



Rezultate:
Câteva măsuri ale performanței modelului de predicție:

- acuratețe ~83%: (1200 + 18) / (1200 + 219 + 33 + 18)
- eroarea (~17%): (219 + 33) / (1200 + 219 + 33 + 18)
- AUC ~61%

Dat fiind faptul că acuratețea e mare și eroarea e mică, putem spune că modelul performează bine pe ansamblu. La o privire mai atentă însă, calitatea modelului de predicție e redusă (AUC e doar puțin peste 50%, kappa e doar 0.07), mai ales în cazul categoriei de interes - cei care părăsesc compania. Astfel, din totalul celor 237 de persoane care părăsesc compania, modelul prezice corect doar 18 cazuri (8% = precision +), iar din totalul de 51 de angajați prezișI de model că vor pleca, doar 18 chiar pleacă (35% = recall +).



	true No	true Yes	class precision
pred. No	1200	219	84.57%
pred. Yes	33	18	35.29%
class recall	97.32%	7.59%	

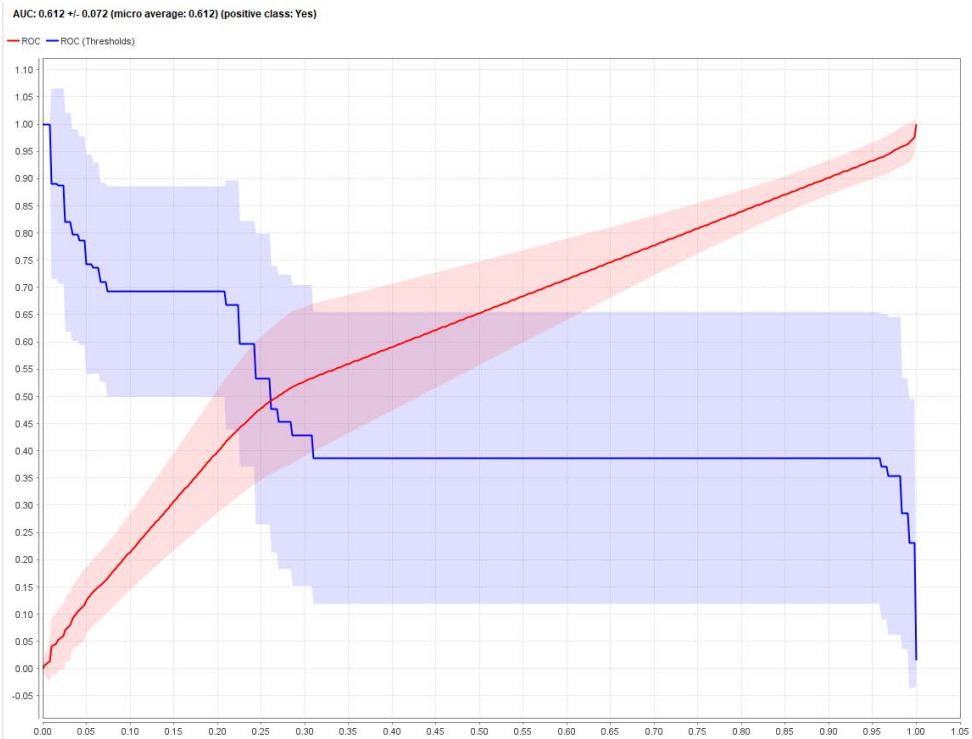
Criterion
accuracy
classification error
kappa
AUC (optimistic)
AUC
AUC (pessimistic)
precision
recall

Table View Plot View

classification_error: 17.14% +/- 1.50% (micro average: 17.14%)

	true No	true Yes	class precision
pred. No	1200	219	84.57%
pred. Yes	33	18	35.29%
class recall	97.32%	7.59%	

Graficul ROC și măsura AUC sunt discutate pe larg în sub-capitolul 0. Aici menționăm faptul că o valoare a AUC de 0.6 este relativ mică (puțin peste valoarea de referință de 0.5), deci performanța clasificării este redusă în acest caz.



Operatorul Performance (Classification)

Acest operator poate fi folosit pentru a calcula performanța unui model de clasificare a cărui variabila de interes (prezisă) este categorială cu două sau mai multe clase. Procesul care exemplifică utilizarea acestui operator este inclus în fișierele atașate volumului, dar nu îl mai prezentăm aici (pașii sunt identici cu cei din Figura 4.2-2). Operatorul „Performance (Classification)" poate calcula mai multe tipuri de măsuri ale performanței, și anume (între paranteze apar valorile măsurilor de performanță ale modelului de clasificare inclus în procesul asociat acestei secțiuni):

Tabelul 4.2-2. Măsuri ale performanței – operatorul Performance (Classification)[9]

Măsură	Descriere	Interval	Valoare model
accuracy	acuratețea clasificării	[0;100]	82.86% +/- 1.50%
classification error	eroarea clasificării	[0;100]	17.14% +/- 1.50%
Cohen's kappa	Coeficientul Kappa a lui Cohen	[0;1][10]	0.069 +/- 0.079
weighted mean recall	media ponderată a valorilor recall	[0;100]	52.46% +/- 2.82%
weighted mean precision	media ponderată a valorilor precision	[0;100]	58.81% +/- 12.75%
spearman rho	corelația rangurilor dintre clasa reală și clasa prezisă	[0;1][11]	0.094 +/- 0.108
kendall tau	măsură a asocierii dintre clasa reală și clasa prezisă	[0;1][12]	0.094 +/- 0.108
absolute error	media abaterii absolute a predicției[13] față de valoarea reală	[0;1][14]	0.252 +/- 0.013
relative error	media abaterii absolute a predicției față de valoarea reală împărțită la valoarea reală	[0;100]	25.18% +/- 1.31%
relative error lenient	media abaterii absolute a predicției față de valoarea reală împărțită la maximul dintre valoarea reală și valoarea prezisă	[0;100]	25.18% +/- 1.31%
relative error strict	media abaterii absolute a predicției față de valoarea reală împărțită la minimul dintre valoarea reală și valoarea prezisă	[0;100][15]	∞%

[9] Rezultatele din tabel sunt cele furnizate de RapidMiner (vezi procesul asociat). Alături de acest proces, am inclus un proces în care am calculat manual câteva măsuri (4.2 Exemple_Calcul_Manual_Masuri).

[10] Teoretic, valoarea minimă a lui Kappa este -1. În practică, utilizat în astfel de contexte, Kappa ia foarte rar o valoare mai mică de 0.

[11] Teoretic, valoarea minimă este -1. În practică, ia foarte rar o valoare mai mică de 0.

[12] Teoretic, valoarea minimă este -1. În practică, ia foarte rar o valoare mai mică de 0.

[13] În cazul acestei măsuri și a celor care urmează, prin predicție ne referim la probabilitatea calculată de model ca un anumit caz să aparțină unei anumite clase, deci la coloanele „confidence" (încredere) din setul de date. Confidence ia valori în intervalul [0;1]. Măsurile de performanță compară în diferite moduri aceste valori ale încrederii cu situația reală.

[14] În cazul atributelor categoriale. Pentru cele metrice, valoarea maximă poate fi oricât.

[15] Dacă variabila dependentă este categorială, intervalul de variație este [0;∞] deoarece numitorul, adică minimul dintre valoarea reală și valoarea prezisă, este întotdeauna 0.

Măsură	Descriere	Interval	Valoare model
normalized absolute error	eroarea absolută împărțită la eroarea obținută în situația în care valoarea prezisă ar fi fost media	[0;1][16]	0.300 +/- 0.016
root mean squared error	media valorilor RMSE = rădăcina pătrată a erorii medii pătratice; numită și abaterea standard a erorii	[0;1][17]	0.374 +/- 0.015
root relative squared error	media valorilor RRSE[18]	[0;1][19]	0.446 +/- 0.018
squared error	media pătratelor erorilor	[0;1][20]	0.140 +/- 0.011
correlation	coeficientul de corelație dintre valoarea reală și cea prezisă	[0;1]	0.109 +/- 0.082
squared correlation	pătratul coeficientului de corelație	[0;1]	0.018 +/- 0.017
cross entropy	suma logaritmilor valorilor încrederii (confidence) asociate clasei corecte împărțită la numărul de cazuri	[0;∞]	∞
margin	valoarea minimă a încrederii asociate clasei corecte	[0;1]	0.016 +/- 0.051
soft margin loss	media diferențelor dintre 1 și încrederea asociată clasei corecte	[0;1]	0.252 +/- 0.013
logistic loss[21]	media expresiei ln(1+exp(-[conf(CC)])), unde „conf(CC)" reprezintă încrederea asociată clasei corecte	[0;∞]	0.396 +/- 0.004

** Clasa Da (pleacă) este definită ca pozitivă. Valorile cu roșu indică un model care clasifică perfect datele. Valorile de pe ultima coloană sunt cele calculate de RapidMiner pentru procesul asociat acestui sub-capitol. În tabel am trecut doar etichetele măsurilor așa cum apar în RapidMiner.*

Am văzut în tabelele anterioare că unele dintre măsurile de performanță nu au o valoarea maximă fixă. În cazul acestora, mai ales atunci când numărul și proporția claselor modelelor pe care le comparăm diferă, trebuie să fim foarte atenți la interpretarea valorilor obținute. De exemplu, în cazul măsurii logloss, valoarea de referință, cea relativ la care ar trebui să ne raportăm atunci când evaluăm un model

[16] În cazul atributelor categoriale. Pentru cele metrice, valoarea maximă poate fi oricât.

[17] În cazul atributelor categoriale. Pentru cele metrice, valoarea maximă poate fi oricât.

[18] $RRSE = \sqrt{\frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{n}(y_i - \bar{y})^2}}$, unde: n = # observații, y_i = valorile observate, $\hat{y}_i$ valorile prezise, $\bar{y}$ = media valorilor observate.

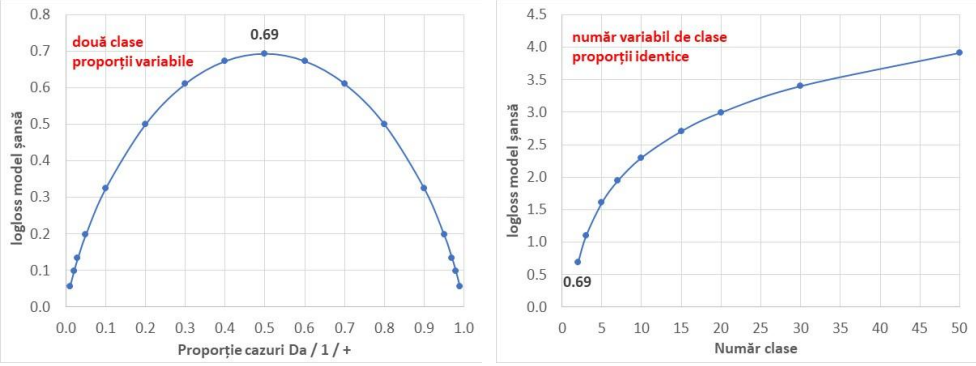
[19] În cazul atributelor categoriale. Pentru cele metrice, valoarea maximă poate fi oricât.

[20] În cazul atributelor categoriale. Pentru cele metrice, valoarea maximă poate fi oricât.

[21] În cazul măsurilor loss, valoarea maximă diferă foarte mult în funcție de numărul de clase și de ponderea claselor.

de clasificare, variază destul de mult în funcție de numărul și proporția claselor (Figura 4.2-3). În situația în care avem două clase egale ca pondere, valoarea de referință (cea observată în cazul modelului naiv) a logloss este 0.69, deci modelul nostru va fi mai bun dacă are o valoarea logloss mai mică de 0.69. Astfel, pentru această situație, o valoare logloss de 0.2 indică un model foarte bun. În situația în care avem două clase cu o pondere foarte diferită, să zicem 2% vs. 98%, valoarea de referință scade foarte mult (0.1). Prin urmare, în acest caz, spre deosebire de situația anterioară, o valoare logloss de 0.2 înseamnă că modelul este inutil (nu ne ajută deloc să creștem calitatea clasificării, dimpotrivă). Dacă avem trei clase cu o pondere egală, valoare de referință crește la 1.1, iar pentru cinci clase la 1.6. Relativ la aceste două situații, o valoare logloss de 1.1 indică un model foarte prost în primul caz, respectiv un model foarte bun în al doilea caz.

Figura 4.2-3.Valoarea de referință (model naiv = „ghicire") a măsurii logloss în funcție de proporția (stânga) și numărul (dreapta) claselor



Care este măsura potrivită pentru evaluarea performanței clasificării?

Răspunsul la această întrebare nu este deloc simplu. Pe scurt, am putea răspunde: „depinde". Mai exact, depinde de situația concretă în care ne aflăm, de datele și de obiectivul urmărit. Fiecare măsură are utilitatea ei, fiind mai mult sau mai puțin potrivită în funcție de datele și scopurile noastre. De exemplu, atunci când clasele au ponderi foarte diferite (unbalanced dataset), acuratețea (respectiv reversul ei, eroare de clasificare) nu este o măsură prea bună a performanței generale a unui model de clasificare. Însă, chiar și într-o situație de acest tip, putem folosi acuratețea pentru a compara performanța generală a clasificării în cazul unei serii de modele parțial diferite, dar construite pe același set de date de instruire.

Precizia, relativ la clasa pozitivă (PPV), este o măsură utilă mai ales atunci când este important să reducem numărul cazurilor fals pozitive (FP). De obicei, ne dorim acest lucru atunci când resursele sunt limitate și/sau costul (bani, timp etc.) asociat „includerii" unui caz fals pozitiv este foarte mare, deci este foarte important să identificăm doar cazuri pozitive.

Reamintirea, relativ la clasa pozitivă (TPR), este o măsură potrivită mai ales atunci când e important să reducem numărul cazurilor fals negative (FN). De obicei, ne dorim acest lucru atunci când avem resurse necesare și/sau costul (bani, timp etc.) asociat „excluderii" unui caz pozitiv este foarte mare, deci este foarte important să identificăm cât mai multe dintre cazurile pozitive.

În concluzie, Precision + și Recall + pot fi măsuri foarte bune dacă ceea ce ne interesează în principal este să facem cât mai puțin erori de predicție relativ la clasa de interes, respectiv dacă dorim să găsim cât mai multe dintre cazurile de interes. Însă, dacă ne interesează simultan ambele obiective, e de preferat să folosim măsuri mai complexe care țin cont simultan și de precizie și de reamintire. Prin urmare, în ceea ce urmează vom compara câteva dintre măsurile complexe (precum Kappa, F1, Youden, MCC), sintetizând recomandările din literatura de specialitate.

Accuracy este foarte probabil măsura utilizată cel mai adesea pentru evaluarea modelelor de clasificare (Chicco & Jurman, 2022). Cu toate acestea, o constatare comună a diferitelor studii de evaluare este că Accuracy (dar și F1) produc rezultate de încredere doar dacă ponderea claselor este echilibrată, respectiv produc rezultate înșelătoare în cazul în care clasele sunt dezechilibrate (Chicco & Jurman, 2020). În niciunul dintre studiile prezentate în continuare, cele două măsuri nu au fost indicate ca potrivite în cazul în care clasele au ponderi foarte diferite (unbalanced dataset). Există însă studii care arată că o versiune modificată a BACC (acuratețea echilibrată) numită BACC1,[22] produce rezultate comparabil mai bune cu alte măsuri (ACC, PRC, SEN, SPC, Kappa, F1), pe date simulate, dar și reale, indiferent de gradul de dezechilibru dintre clase (Vanacore et al., 2022).[23]

În unul dintre studii, numit sugestiv BenchMetrics (Canbek et al., 2021), autorii compară performanța măsurilor TPR, TNR, PPV, NPV, ACC, INFORM (Youden), MARK

[22] Deși măsurile BACC și BACC1 au un comportament similar, superior celorlalte măsuri, ultima este de preferat deoarece ține cont, pe lângă gradul de dezechilibru dintre clase, și de diferențele relativ la predicția naivă / prin șansă (Vanacore et al., 2022).

[23] BACC1 poate fi generalizat pentru matricele cu mai mult de două clase (Vanacore et al., 2023). În același studiu este introdusă și varianta ponderată a BACC1.

(psep), BACC, G (Fowlkes-Mallows Index), NMI (Normalized Mutual Info), F1, CK (Cohen's Kappa) și MCC. Comparația vizează performanța clasificării în cazul unui atribut cu două clase (binominal), ponderile claselor fiind alese astfel încât să varieze de la o pondere echilibrată la una dezechilibrată. Pentru evaluarea performanței măsurilor sunt folosite mai multe criterii. Rezultatele obținute au indicat faptul că robustețea măsurilor utilizate în majoritatea articolelor de specialitate este redusă și că MCC este măsura cea mai robustă.

Într-un alt studiu (Luque et al., 2019), în urma analizei rezultatelor unor simulări (matrice de confuzie 2x2, deci tot două clase; de această dată clasele au ponderi diferite, gradul de dezechilibru fiind variabil), a rezultat că Youden și MCC sunt măsurile de preferat. Dacă ne interesează strict situațiile de clasificare corectă, Youden este măsura cea mai bună. Dacă avem în vedere atât cazurile clasificate corect cât și erorile de clasificare, MCC este cea mai bună alegere.

Într-un studiu care compară valorile măsurilor MCC și Kappa în funcție de gradul de dezechilibru dintre clase (Delgado & Tibau, 2019), autorii constată că acestea se comportă diferit. Dacă ponderea claselor este similară, MCC și Kappa iau valori similare (identice atunci când matricea e perfect simetrică). Însă, atunci când asimetria matricei de confuzie crește (clase dezechilibrate), MCC și Kappa nu mai spun aceeași poveste. Recomandarea autorilor e clară: MCC este măsura mai robustă, deci de preferat pentru astfel de situații (unbalanced dataset), în timp ce Kappa trebuie evitat (Delgado & Tibau, 2019).

Alte două studii, tot relativ la cazul unei matrice de confuzie 2x2 cu privire la care ne interesează ambele clase în aproximativ același grad, constată că MCC este o măsură mai potrivită (informativă) comparativ cu măsurile:

- acuracy și F_1 Score (Chicco & Jurman, 2020);
- balanced accuracy, bookmaker informedness (Youden în RapidMiner) și markedness (psep în RapidMiner) (Chicco, Tötsch, et al., 2021).

Într-un alt articol (Zhu, 2020), autorul realizează o serie de simulări pentru diferite scenarii arătând că anumite măsuri ale performanței clasificării sunt mai potrivite decât altele în situația în care clasele au ponderi foarte diferite (testează tot cazul 2x2). Rezultatele prezentate arată că măsurile nepotrivite sunt ACC, Kappa, F_1, GPR (media geometrică a măsurilor Precision + și Recall +) și MCC, respectiv că măsura de preferat este Youden. O altă măsură relativ mai potrivită este GBA (media geometrică a măsurilor Recall + și Recall –). Spre deosebire de alte analize (Chicco,

Tötsch, et al., 2021; Chicco, Warrens, et al., 2021; Chicco & Jurman, 2020), în acest caz, a rezultat că MCC nu este o măsură potrivită în cazul în care clasele sunt dezechilibrate.[24]

Toate articolele prezentate anterior au încercat să găsească „măsura perfectă". Însă, e foarte posibil ca o astfel de măsură nici să nu existe, în sensul că este puțin probabil ca o măsură să poată îndeplini simultan și maximal toate criteriile relevante pe care este de dorit să le îndeplinească (minim și maxim, simetrie, distanță, monotonicitate, valoare de referință constantă etc.) (Gösgens et al., 2022). Suplimentar, ne confruntăm și cu problema incertitudinii asociate fiecărei măsuri. Funcție de numărul de cazuri din fiecare celulă a matricei de confuzie, valorile măsurilor pot varia destul de mult și în grade diferite, deci comparațiile între măsurile de performanță ar trebui să țină cont de incertitudinea asociată (Lovell et al., 2022). Câteva vizualizări asociate acestui articol pot fi găsite aici: Interactive Confusion Simplexes, Desmos Calculator, Animation of Matthews Correlation Coefficient. Totuși, o soluție alternativă poate lua forma unei agregări a mai multor măsuri complexe ale performanței într-o măsură unică. O astfel de soluție este propusă sub numele General Performance Score (GPS) (De Diego et al., 2022). Pentru a agrega diferitele măsuri, GPS calculează media armonică a valorilor acestora (astfel, valorile mici sunt penalizate). Măsurile incluse în GPS pot fi selectate în funcție de obiectivul specific urmărit (combinația de obiective). Suplimentar, un avantaj important al metodologiei GPS constă în faptul că poate combina măsuri ale performanței atât pentru probleme de clasificare cu două clase, cât și pentru trei sau mai multe clase.

Operatorul Performance (Costs)

Erorile de predicție reduc calitatea modelului și utilitatea acestuia. În cazul unui model de clasificare, simplificat, erorile de predicție sunt de două tipuri: (1) predicție Da + realitate Nu și (2) predicție Nu + realitate Da. Erorile de tip 1 se mai numesc și fals pozitiv, iar cele de tip 2 fals negativ. În unele situații, la nivel practic, unul dintre cele două tipuri de erori are consecințe negative într-o măsură mai mare. De exemplu, în cazul atriției, costul plecării unui angajat poate fi în medie semnificativ mai mare comparativ cu costul unei măsuri implementate cu scopul de a-l păstra în organizație. În acest caz, o eroare de predicție de tip 1 (prezicem că angajatul va

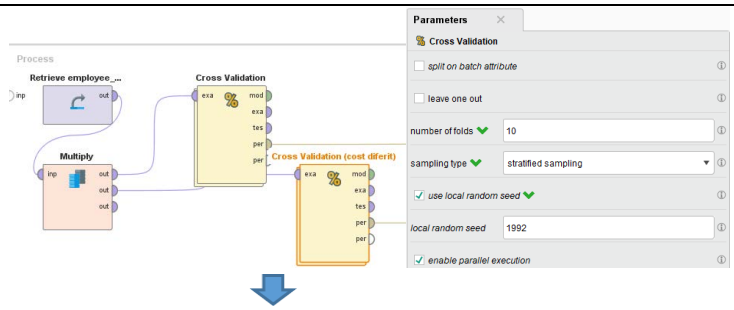
[24] „The higher the imbalance ratio, the MCC measurements tend to be more skewed and behave nonlinearly with respect to the linear increases of TP and TN values" (Zhu, 2020).

pleca, dar acesta rămâne) are un impact negativ mai mic comparativ cu o eroare de predicție de tip 2 (prezicem că angajatul va rămâne, dar acesta pleacă).

Pentru a ține cont de această diferență relativ la impactul tipurilor de erori, putem alege una dintre următoarele soluții: (1) construim modele de clasificare care să facă mai puține erori din tipul celor cu impact negativ mai mare și (2) atribuim o importanță mai mare erorilor de clasificare asociate tipul de eroare cu cost mai mare. Soluția 1 poate fi implementată prin alegerea unor clasificatori care fac mai puține erori „costisitoare" sau prin modificarea pragului de probabilitate a clasificării (de obicei acesta e setat la 50%, deci un caz cu o probabilitate de 50%+ va fi alocat la o clasă, iar unul cu o probabilitate mai mică de 50% la cealaltă clasă). Soluția 2 se referă la atribuirea unui cost diferit celor două tipuri de erori. Eroarea mai costisitoare va primi o pondere mai mare, iar cealaltă o pondere mai mică. De exemplu, în Figura 4.2-4, am prezentat comparativ costul erorii de clasificare în cazul în care erorile au același impact, respectiv un impact diferit (eroarea de tip 2 este de trei ori mai costisitoarea comparativ cu eroarea de tip 1).[25]

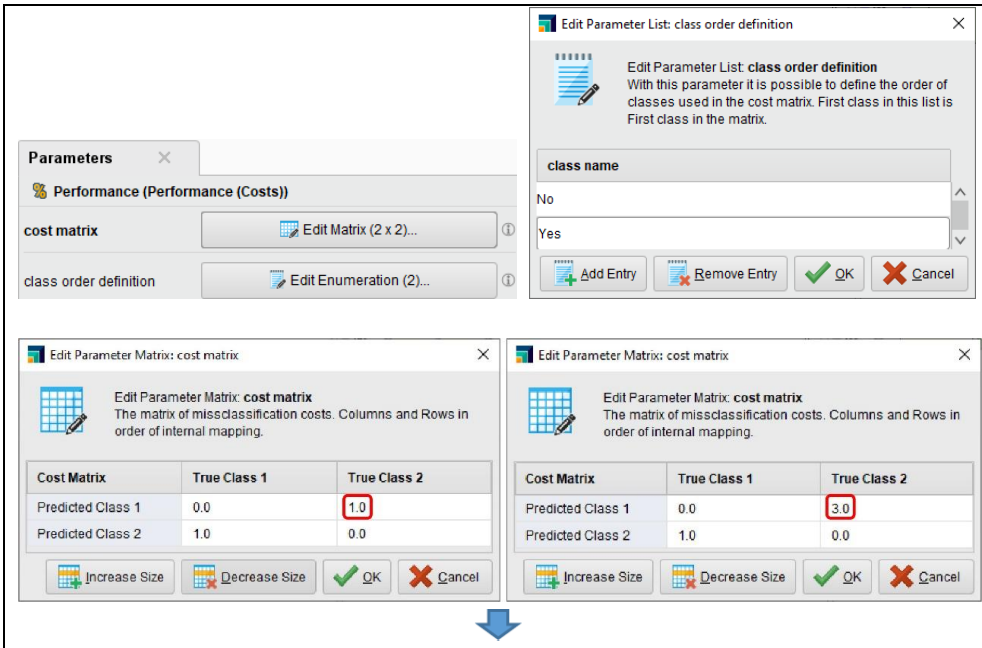
Figura 4.2-4. Calcularea performanței predicției în două situații: erori cu cost identic vs. diferit

Pasul 1:
Conectăm setul de date și operatorii din imagine.
Setăm valorile parametrilor.



Pasul 2:
În interiorul operatorului de Validare, fereastra de testare a modelului de predicție, conectăm operatorii „Apply Model" și „Performance (Costs)".
Pentru fiecare operator „Performance (Costs)", la parametrul „class order definition" definim ordinea dorită a claselor. În cazul de față prima clasă este No (nu pleacă) și a doua este Yes (pleacă). Apoi definim parametrul „cost matrix" (costul fiecărei erori de clasificare). În prima situație costul este identic și egal cu 1, în a doua este diferit (3 și 1), fiind mai mare în cazul erorilor de clasificare relativ mai costisitoare pentru companie. În acest caz am apreciat (valorile pot fi estimate mai exact în baza unor informații concrete) că unul dintre costuri, cel asociat erorii în care prezicem că angajatul rămâne, dar acesta pleacă, e de trei ori mai mare decât costul asociat celeilalte erori (prezicem că angajatul pleacă, dar acesta rămâne).

[25] O prezentare video introductivă cu privire la impactul atribuirii unor costuri diferite erorilor de clasificare (dar și clasificărilor corecte), puteți găsi aici: Cost Sensitive Scoring - Basics.



Rezultate:

Dacă apreciem că toate tipurile de erori de clasificare au același cost pentru companie, eroarea de predicție este 17.1%. Dacă apreciem că unul dintre tipurile de eroare de clasificare (prezicem că angajatul nu pleacă, dar acesta pleacă; avem 215 erori de acest tip) are un cost de trei ori mai mare, eroarea de predicție devine 46.4%.

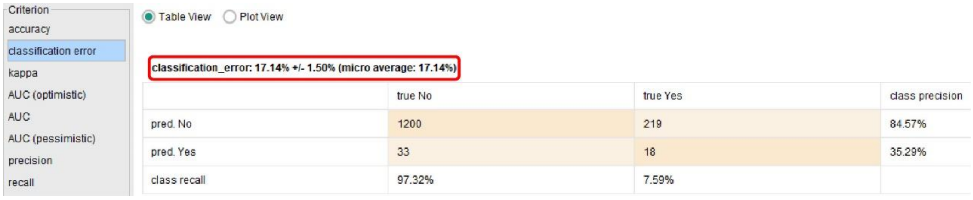
Misclassification costs:
Identice (1 și 1) **0.171** +/- 0.015 (micro average: 0.171)
Diferite (3 și 1) **0.469** +/- 0.027 (micro average: 0.469)

Cost identic	True No	True Yes	Total
Pred. No	1200	219	1419
Pred. Yes	33	18	51
Total	1233	237	1470

Cost diferit	True No	True Yes	Total
Pred. No	1200	3*219=657	1419
Pred. Yes	33	18	51
Total	1233	237	1470

(219 + 33) / 1470 = 17.1% **eroarea de clasificare** (657 + 33) / 1470 = 46.9%

Eroarea de clasificare și confusion matrix calculate anterior cu operatorul „Performance (Binominal Classification)”



4.3. Vizualizarea performanței (Visual): Curbele ROC și PR și măsurile AUROC și AUPRC

ROC reprezintă acronimul pentru „Receiver Operating Characteristic", iar AUROC acronimul pentru „Area Under ROC" (uneori este numită și AUC, fiind necesară totuși menționarea faptului că e vorba de aria subsumată unei curbe ROC). Curba ROC și măsura AUROC sunt utilizate frecvent pentru evaluarea performanței unui model de clasificare. În ambele cazuri atributul de interes trebuie să fie de tip binominal, adică să aibă doar două categorii / clase.[26] ROC și AUROC sunt calculate pe baza TPR (True Positive Rate) și FPR (False Positive Rate), deci ne ajută să identificăm zona de compromis dintre cele două obiective majore ale unei clasificări: (1) predicția corectă a unei proporții cât mai mari a cazurile din clasa de interes / pozitivă (TPR) și (2) realizarea unui număr cât mai mic de predicții incorecte relativ la clasa de interes (FPR).

PRC reprezintă acronimul pentru „Precision-Recall Curve", iar AUPRC acronimul pentru „Area Under PRC". Curba PR și măsura AUPRC sunt utilizate pentru evaluarea performanței unui model de clasificare strict în relație cu clasa pozitivă a unui atribut de interes de tip binominal. O curbă PR sumarizează compromisul dintre măsurile precizie și reamintire pentru diferite praguri de probabilitate. PRC și AUPRC sunt calculate pe baza măsurilor precizie și reamintire asociate clasei pozitive, deci ne oferă o imagine mai corectă a performanței în situația în care setul de date nu este echilibrat (de obicei clasa pozitivă, de interes, are o pondere mult mai mică decât cea negativă).

Logica ROC și AUROC

Pentru a înțelege cât mai ușor logica acestor măsuri, prezentăm câteva exemple simple în Figura 4.3-1.[27] Curba ROC prezintă sub o formă grafică compromisul (trade-off) dintre TPR și FPR pentru diferite praguri de probabilitate (thresholds) (Tan et al., 2019, pp. 525–528). Valorile TPR apar pe axa verticală (cresc de jos în sus), iar valorile FPR pe cea orizontală (cresc de la stânga la dreapta). Ambele măsuri iau

[26] Dacă atributul are mai multe categorii, putem folosi ROC și AUC doar după ce am grupat categoriile în două clase. De exemplu, dacă atributul are patru categorii, A/B/C/D, le putem grupa în categoriile A și non-A.

[27] O serie de prezentări video care pot fi utile pentru înțelegerea acestor concepte sunt următoarele: ROC and AUC, Clearly Explained!, SAS Tutorial | ROC Charts in SAS, ROC Curves and Lift Chart | RapidMiner.

valori în intervalul 0-100 sau 0-1, funcție de notație (interpretarea e aceeași). Câteva exemple limită de modele sunt următoarele: un model definit prin valorile TPR=0 și FPR=0 prezice că toate cazurile aparțin clasei negative; un model definit prin valorile TPR=1 și FPR=1 prezice că toate cazurile aparțin clasei pozitive; un model definit prin valorile TPR=1 și FPR=0 prezice corect toate cazurile (nu face nicio eroare de clasificare), deci este perfect. Un model de clasificare este cu atât mai bun cu cât este localizat cât mai aproape de colțul din stânga sus al graficului. Un model care este situat pe diagonală sau aproape de aceasta este un model inutil, un model care nu este mai bun decât unul care prezice la întâmplare apartenența cazurilor la una dintre cele două clase.

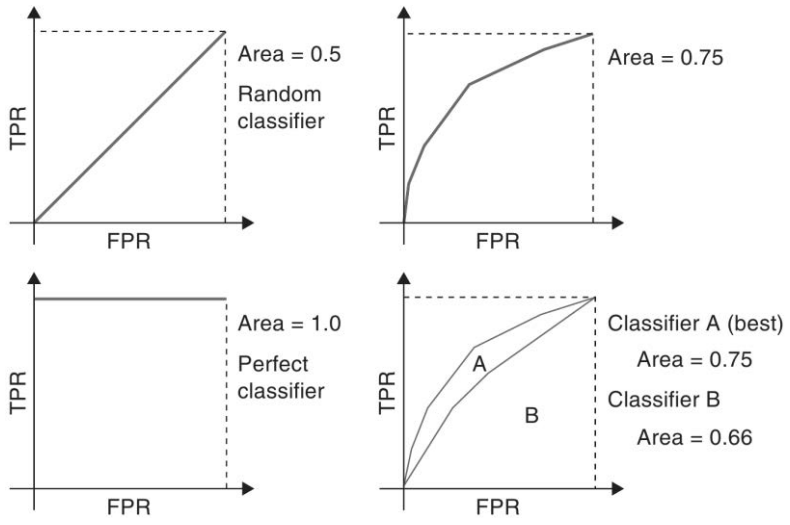
Linia care unește cele două colțuri ale graficului (punctele cu coordonatele [0;0] și [1;1]) este tocmai curba ROC. Putem construi această linie / curbă prin unirea punctelor definite de valorile [TPR;FPR]. Valorile TPR și FPR se calculează pentru diferite praguri / nivele de probabilitate (thresholds). Cu cât TPR ia valori mari, cu atât modelul de clasificare identifică o pondere mai mare a cazurilor pozitive (de interes). Cu cât FPR ia valori mai mici, cu atât modelul de clasificare face mai puține erori atunci când identifică cazurile pozitive. În consecință, ne dorim ca un model de clasificare să aibă simultan valori cât mai mari pentru TPR concomitent cu valori cât mai mici pentru FPR.

În panelul din stânga sus (Figura 4.3-1), curba ROC este chiar diagonala, deci o creștere a TPR este însoțită de o creștere identică a FPR. Simplu spus, performanța modelului de clasificare este egală cu performanța modelului naiv (predicția apartenenței la clase este realizată aleatoriu), deci modelul de clasificare este inutil. În panelul din dreapta sus, curba ROC crește mai rapid la început, apoi tinde să se aplatizeze. Asta înseamnă că, la început, modelul de clasificare identifică cazurile pozitive și face puține erori, iar ulterior, încercând să găsească și restul de cazuri pozitive, face sistematic tot mai multe erori (confundă cazurile negative cu cele pozitive). În panelul din stânga jos, curba ROC crește la valoarea maximă încă de la început, apoi rămâne constantă, prin urmare modelul prezice perfect apartenența cazurilor la cele două clase. Desigur, în practică, situațiile mai des întâlnite sunt cele descrise de graficul din panelul din stânga jos. Observăm că, indiferent de pragul considerat, modelul A are o performanță a clasificării mai bună comparativ cu modelul B. Cu privire la același grafic, este foarte posibil să avem situații în care modelul A performează mai bine pentru anumite praguri, iar modelul B performează mai bine pentru alte praguri. Astfel de situații fac oarecum mai dificilă compararea performanței celor două modele.

Pentru a compara mai ușor două modele, putem considera toate pragurile și calcula o măsură unică a performanței. Această măsură se numește aria de sub curba ROC, pe scurt AUROC. Evident, AUROC ia valori tot în intervalul [0;1] sau [0;100], funcție de notație, dar de obicei este mai mare de 0.5 (50%). O valoare mai mare indică un model mai bun. În Figura 4.3-1 sunt calculate valorile AUROC pentru diferite curbe ROC:

- în panelul din stânga sus, AUROC ia valoarea 0.5, deci modelul de clasificare are o calitate identică cu modelul bazat pe „ghicire", adică pe predicția aleatoare; de exemplu, dacă clasa de interes are o pondere de 50% în populație, un model bazat pe „ghicire" care prezice că fiecare caz aparține clasei de interes va „ghici" corect apartenența la clasa de interes în jumătate din situații;
- în panelul din dreapta sus, AUROC este 0.75, deci modelul de predicție este clar mai bun comparativ cu predicția aleatoare;
- în panelul din stânga jos, AUROC ia valoarea 1, deci modelul de predicție este perfect;
- în panelul din dreapta jos, modelul A (AUROC = 0.75) are o performanță mai bună comparativ cu modelul B (AUROC = 0.66).

Figura 4.3-1. Câteva exemple de curbe ROC și măsura AUROC asociată acestora



Sursa: (Moreira et al., 2019, p. 204)

Construcția curbei ROC și calcularea AUROC

Pentru a ilustra procesul de construcție a unei curbe ROC și de calcul a valorii AUROC, vom folosi un exemplu simplu cu doar 15 cazuri. Dintre acestea, 8 cazuri sunt pozitive (evenimentul are loc) și 7 negative (evenimentul nu are loc). Să presupunem că modelul de clasificare folosit de noi a estimat probabilitățile de apartenență la clasa pozitivă așa cum apar în Tabelul 4.3-1. În tabel cazurile sunt sortate descendent în funcție de probabilitatea estimată. Tot aici apare informația cu privire la situația reală (clasa la care aparține fiecare caz), respectiv valoarea cumulată pentru TP (true positive) și FP (false positive). Observăm că modelul prezice că primul caz este pozitiv (cu o probabilitatea estimată de 99%), cazul este în realitate pozitiv, deci predicția este corectă. Prin urmare, la acest pas, avem un caz de tip TP și niciunul FP, prin urmare TPR (true positive rate) este 0.13 (TP/P, adică 1/8), iar FPR (false positive rate) este 0.00 (FP/N, adică 0/7). Considerând primele două cazuri (cazul doi este tot TP), valorile se schimbă: TPR = 0.25 (2/8), iar FPR = 0.00 (0/7). Coborând pragul de certitudine (probabilitate) la 0.85 (cazul 5), observăm că apare prima predicție incorectă: cazul 5 este prezis de model ca pozitiv, dar este în realitate negativ. La acest pas, TPR = 0.50 (4/8), iar FPR = 0.14 (1/7), adică modelul a identificat 4 din totalul celor 8 cazuri pozitive cu prețul unei erori din cele 7 posibile. În continuare, procedăm la fel pentru a calcula aceste valori pentru toate situațiile.

Pentru a calcula AUROC am folosit formula

$$AUROC = (TPR_{t+1} - TPR_t) / 2 * (FPR_{t+1} - FPR_t),$$

unde coeficienții t și t+1 indică ordinea cazurilor (rank-ul). Pe ultima linie sunt însumate toate valorile AUROC calculate pentru fiecare dintre combinațiile de două cazuri alăturate. Valoarea AUROC este 0.857, deci modelul de predicție este unul foarte bun.

Tabelul 4.3-1. Curba ROC și măsura AUROC - un exemplu simplu

rank	prob. clasa P	situația reală	TP	FP	TPR (TP / P)	FPR (FP / N)	AUROC
			0	0	0.00	0.00	
1	0.99	1	1	0	0.13	0.00	0.000
2	0.95	1	2	0	0.25	0.00	0.000
3	0.94	1	3	0	0.38	0.00	0.000
4	0.91	1	4	0	0.50	0.00	0.000
5	0.85	0	4	1	0.50	0.14	0.071
6	0.82	1	5	1	0.63	0.14	0.000
7	0.76	1	6	1	0.75	0.14	0.000
8	0.66	0	6	2	0.75	0.29	0.107
9	0.55	1	7	2	0.88	0.29	0.000
10	0.45	0	7	3	0.88	0.43	0.125
11	0.38	0	7	4	0.88	0.57	0.125
12	0.24	1	8	4	1.00	0.57	0.000
13	0.18	0	8	5	1.00	0.71	0.143
14	0.12	0	8	6	1.00	0.86	0.143
15	0.06	0	8	7	1.00	1.00	0.143
						Sumă	0.857

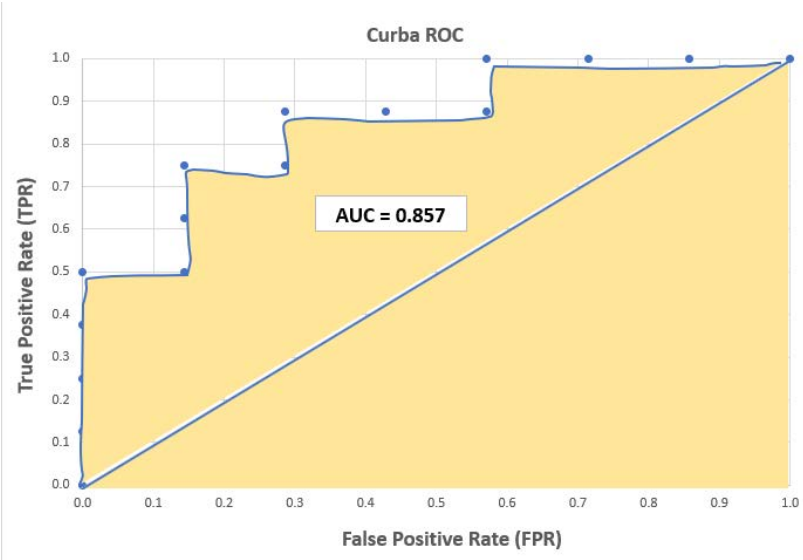
	Realitate	
Pred.	Da(1)	Nu(0)
Da(1)	TP	FP
Nu(0)	FN	TN

0.85	Realitate	
Pred.	Da(1)	Nu(0)
Da(1)	4	1
Nu(0)	4	6

** 1 = clasa pozitivă, 0 = clasa negativă*

Citirea unui tabel de acest tip este destul de dificilă chiar și atunci când setul de date conține puține cazuri. Pentru a ușura citirea și interpretarea, putem reprezenta valorile importante din tabel sub formă grafică. În Figura 4.3-2 am reprezentat, pentru fiecare prag de probabilitate, valorile relevante din Tabelul 4.3-1, mai exact perechea de valori TPR (true positive rate, numită și sensitivity) și FPR (false positive rate; obținem același lucru dacă scădem din 1 valoarea specificity). În pasul următor am unit punctele cu o linie de aceeași culoare (albastru), obținând curba ROC. Aria de sub curba ROC (colorată în portocaliu) reprezintă valoarea AUROC (0.857 sau 85.7%).

Figura 4.3-2. Curba ROC și AUROC (AUC) - un exemplu simplu



Construcția curbei PR și calcularea AUPRC

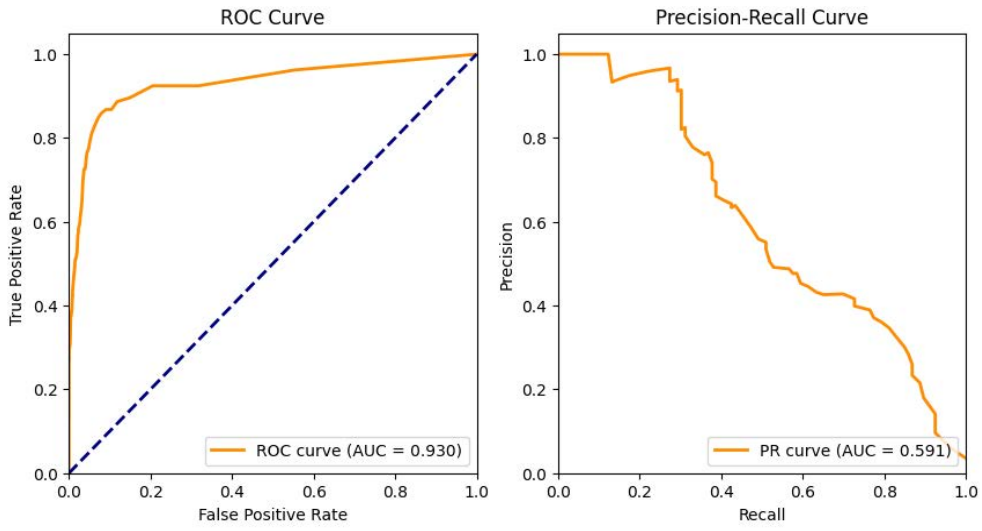
Atunci când clasele variabilei de interes au o pondere foarte diferită (unbalanced dataset), ROC și AUROC supra-estimează performanța modelului de clasificare. Aceasta se întâmplă deoarece (1) clasa negativă are o pondere semnificativ mai mare și (2) modelul clasifică corect o mare parte a cazurilor negative, deci per total, chiar dacă prezice corect o mică parte a cazurilor pozitive, de interes, numărul total de cazuri prezise corect este relativ mai mare. Pentru astfel de situații este mai potrivit să folosim un grafic numit Precision-Recall Curve (curba PR) (Saito & Rehmsmeier, 2015). Măsura numită reamintire este același lucru cu TPR (ponderea cazurilor prezise corect ca pozitive din total cazuri pozitive), această măsură fiind folosită și în cazul curbei ROC. Măsura numită precizie este egală cu raportul dintre numărul de cazuri pozitive prezise corect și numărul total de cazuri prezise ca pozitive (TP / PP). Prin urmare, curba PR ignoră numărul mare de cazuri negative, deci calculează performanța modelului doar în relație cu clasa pozitivă (cazurile de fraudă, de părăsire a companiei, de boală etc.), cea care din punct de vedere practic este cel mai adesea de interes pentru analist.

Curba PR prezintă sub o formă grafică compromisul (trade-off) dintre precizie și reamintire pentru diferite praguri de probabilitate (thresholds) (Tan et al., 2019, pp. 530–532). Valorile măsurii precizie apar pe axa verticală (cresc de jos în sus), iar

valorile măsurii reamintire pe cea orizontală (cresc de la stânga la dreapta). Ambele măsuri iau valori în intervalul 0-100 sau 0-1, funcție de notație (interpretarea e aceeași). Un model definit prin valori minime (0) ale ambelor măsuri este cel mai prost posibil, iar un model caracterizat prin valori maxime (1) ale preciziei și reamintirii este un model perfect. Valorile asociate unui model bazat pe ghicire (random classifier) definesc o linie orizontală (paralelă cu axa ce indică valoarea măsurii reamintire), adică valoarea preciziei este de fiecare dată aceeași, iar valoarea reamintirii variază. Sintetic, un model de clasificare este cu atât mai bun cu cât acesta este localizat cât mai aproape de colțul din dreapta sus al graficului.

Similar cu AUROC, și în cazul unui grafic PRC putem calcula o măsură sintetică, numită AUPRC, care combină într-o singură valoare informația din graficul PRC (Figura 4.3-3).[28] AUPRC variază tot în intervalul [0;1] ([0;100]), o valoare mai mare indicând un model mai performant. Astfel, dacă AUPRC este egal cu unu, adică măsurile precizie și reamintire iau ambele valoarea maximă, adică 1, modelul clasifică perfect cazurile pozitive, iar dacă AUPRC este 0.5, performanța modelului nu este mai bună decât performanta unui model bazat pe ghicire (model aleator).

Figura 4.3-3. Un exemplu simplu de curbe ROC și PRC, respectiv valoarea AUC asociată fiecăreia



Sursa: Juan Esteban de la Calle. 2023. How and Why I Switched from the ROC Curve to the Precision-Recall Curve to Analyze My Imbalanced Models: A Deep Dive

[28] Modul în care este calculată valoarea AUPRC diferă uneori de la un soft la altul, ceea ce produce valori ușor diferite (Chen et al., 2024).

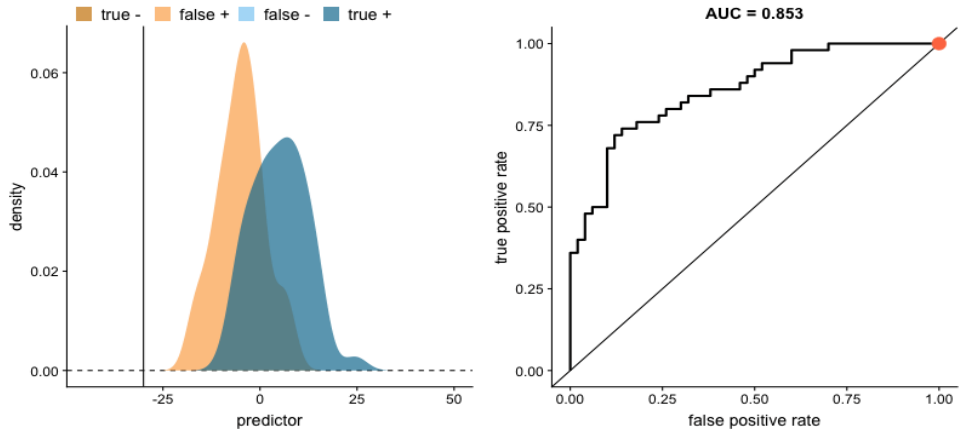
Este important să reținem faptul că în cazul în care setul de date nu este echilibrat (ponderea celor două clase ale variabilei de interes este foarte diferită), un grafic PRC este mai informativ cu privire la performanța modelului (algoritmului de clasificare) comparativ cu un grafic ROC (Davis & Goadrich, 2006), prin urmare preferăm curba PR și măsura AUPRC.

Relația dintre curba ROC, AUC/AUROC și AUPRC

Pentru a înțelege și mai bine relația dintre distribuția claselor variabilei de interes, pragurile de referință, curbe și măsurile de performanță, prezentăm o serie de grafice animate.[29] În Figura 4.3-4, variația pragului de probabilitate (cutoff value = linia verticală, graficul din stânga) este însoțită de modificarea valorilor TPR și FPR (ilustrate de punctul roșu, graficul din dreapta). Gif-ul ilustrează compromisul (trade-off) pe care trebuie să-l facem atunci când alegem să prezicem corect ambele clase: atunci când TPR crește, FPR crește; reciproc, atunci când preferăm o valoare FPR mai mică, TPR scade implicit. La o extremă putem alege să prezicem că toate cazurile sunt pozitive prețul fiind că prezicem incorect toate cazurile negative (TPR și FPR iau valoarea maximă, adică 1; punctul roșu este poziționat în colțul din dreapta sus). La cealaltă extremă putem alege să prezicem că niciun caz nu este pozitiv, deci nu facem nicio predicție incorectă în ceea ce privește cazurile pozitive (TPR și FPR iau valoarea minimă, adică 0; punctul roșu este poziționat în colțul din stânga jos). Desigur, niciuna dintre cele două situații nu ne ajută practic. În funcție de situația concretă de analiză și de obiectivul principal al analizei, vom alege valori intermediare ale pragului. Relativ la acest exemplu, probabil vom prefera situația definită de un TPR egal cu 0.75 și un FPR egal cu 0.15, ceea ce reprezintă un compromis rezonabil între nevoia de a identifica corect cât mai multe dintre cazurile pozitive simultan cu producerea unui număr cât mai mic de erori de predicție (cazuri pe care modelul le prezice că sunt pozitive dar care, în realitate, sunt negative).

[29]Toate aceste animații au fost preluate de pe contul de github al Dariya Sydykova: https://github.com/dariyasydykova/open_projects/tree/master/ROC_animation.

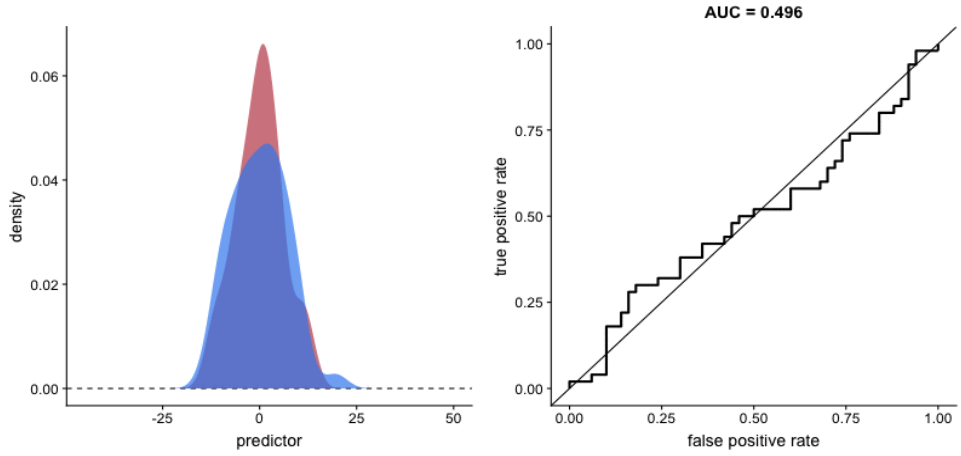
Figura 4.3-4. Ilustrarea relației dintre pragul ales (cutoff value) și valorile TPR și FPR



Sursa:
https://github.com/dariyasydykova/open_projects/blob/master/ROC_animation/animations/cutoff.gif

În Figura 4.3-5 se poate observa relația dintre calitatea modelului și forma curbei ROC, respectiv valoarea AUC. Dacă modelul nu distinge între cele două clase (nu poate prezice apartenența cazurilor la una dintre cele două clase), curba ROC se suprapune cu diagonala principală, iar AUC este aproximativ egal cu 0.5. Dacă modelul prezice (separă) perfect cele două clase, curba ROC crește perpendicular până la valoarea maximă, apoi merge paralel cu axa orizontală, iar AUC ia valoarea maximă, adică 1. Restul situațiilor sunt intermediare acestor două extreme.

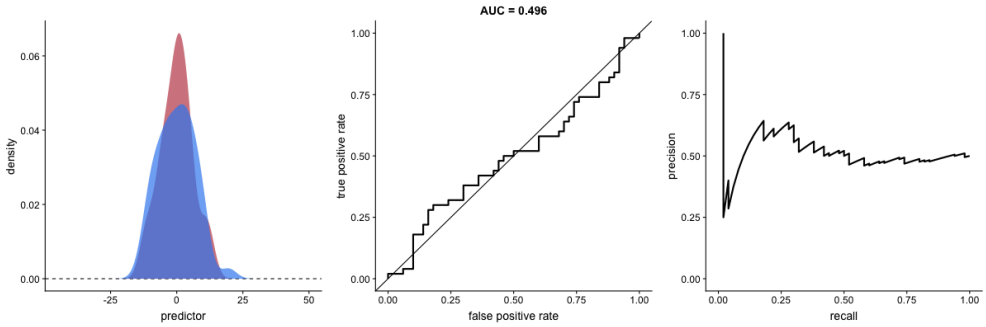
Figura 4.3-5. Variația curbei ROC și valorii AUC (AUROC) în funcție de capacitatea predictivă a modelului



Sursa:
https://github.com/dariyasydykova/open_projects/blob/master/ROC_animation/animations/ROC.gif

Figura 4.3-6 este similară cu figura anterioară dar prezintă suplimentar variația măsurilor precizie și reamintire (curba PR, precision-recall). Cu cât modelul de clasificare separă mai bine între cele două clase, cu atât valorile măsurilor precizie și reamintire cresc (nu neapărat în același ritm la fiecare pas). Simplu spus, un model performant separă mai bine între cele două clase, adică (1) identifică cât mai multe dintre cazurile pozitive și simultan face puțin erori de predicție (cazuri în realitate negative pe care le prezice că sunt pozitive) (curba ROC), respectiv (2) găsește cât mai multe dintre cazurile pozitive și simultan prezice corect cât mai multe dintre cazurile prezise ca pozitive (curba PR).

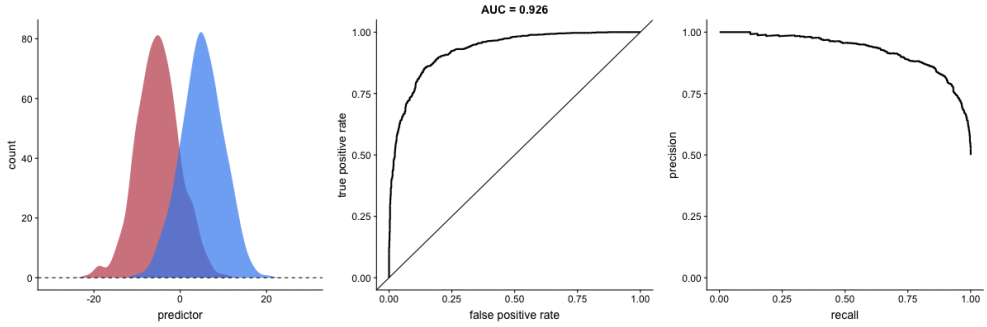
Figura 4.3-6. Variația curbelor ROC și PR în funcție de capacitatea predictivă a modelului



Sursa:
https://github.com/dariyasydykova/open_projects/blob/master/ROC_animation/animations/PR.gif

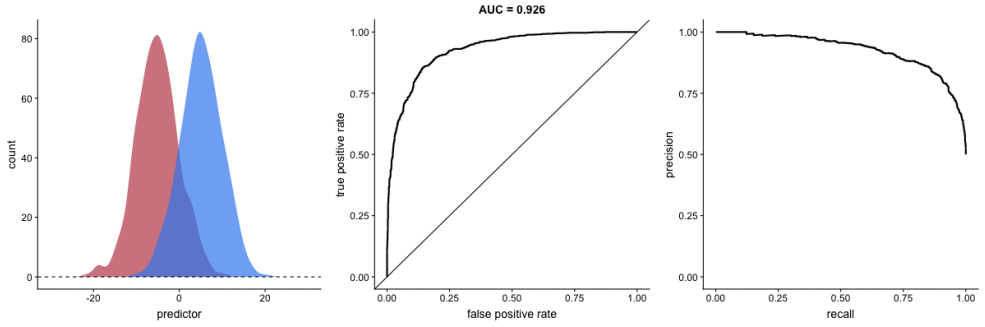
În Figura 4.3-7 distribuția celor două clase variază (în gif, se modifică vizual doar numărul de cazuri albastre, de la foarte puține la aproximativ același număr cu cazurile roșii), dar calitatea modelului de predicție nu se schimbă. Cu toate acestea, estimările performanței modelului, așa cum sunt reflectate de curbele ROC și PR, respectiv valoarea AUC (AUROC), se modifică. Observăm că graficul PRC este afectat semnificativ mai mult de diferența dintre numărul de cazuri asociat celor două clase. Același efect apare și în Figura 4.3-8, de această dată în relație cu clasa roșie. Diferențele dintre cele două grafice ROC sunt mici. La start (în momentul în care cele două clase au același număr de cazuri), cele două grafice PRC sunt identice. La momentul final, definit printr-un dezechilibru maxim între numărul de cazuri asociate claselor (highest imbalance), diferențele dintre graficele PRC sunt evidente în timp ce graficele ROC sunt aproape identice.

Figura 4.3-7. Variația curbelor ROC și PRC în funcție de distribuția celor două clase (clasa pozitivă / albastră este minoritară; clasa negativă / roșie este dominantă)



Sursa:
https://github.com/dariyasydykova/open_projects/blob/master/ROC_animation/animations/imbalance.gif

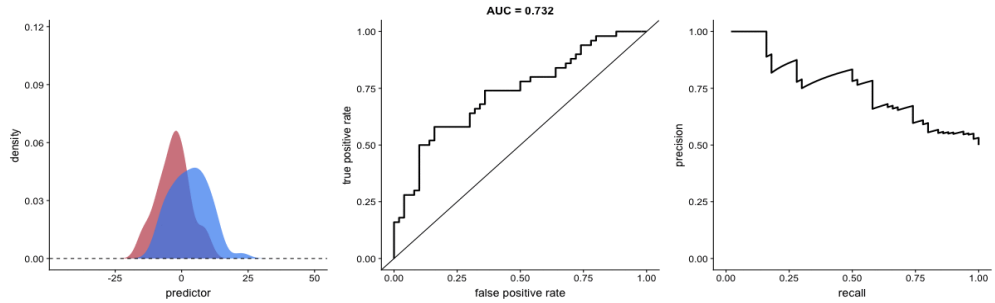
Figura 4.3-8. Variația curbelor ROC și PR în funcție de distribuția celor două clase (clasa pozitivă / albastră este dominantă; clasa negativă / roșie este minoritară)



Sursa:
https://github.com/dariyasydykova/open_projects/blob/master/ROC_animation/animations/imbalance2.gif

Măsurile de performanță nu sunt influențate doar de gradul de dezechilibrul dintre numărul de cazuri asociat celor două clase (imbalance), ci și de diferențele observate la nivelul abaterii standard a variabilei independente (variabila predictor) asociate celor două clase. În Figura 4.3-9, observăm că atunci când abaterea standard asociată clasei albastre variază, variază și măsurile de performanță. De această dată, influența este mai mare în cazul curbei ROC și a valorii AUC, graficul PRC fiind mai puțin afectat. Atunci când abaterea standard este mai mică, curba ROC crește mai repede, iar valoarea AUC crește, ceea ce ar trebui să denote o calitate mai bună a predicției. Însă, de fapt, observăm că pentru valori mici ale FPR, calitatea predicției (măsurată prin indicatorul precizie) este sensibil mai mică.

Figura 4.3-9. Variația curbelor ROC și PR în funcție de abaterea standard la nivelul clasei pozitive (clasa albastră)



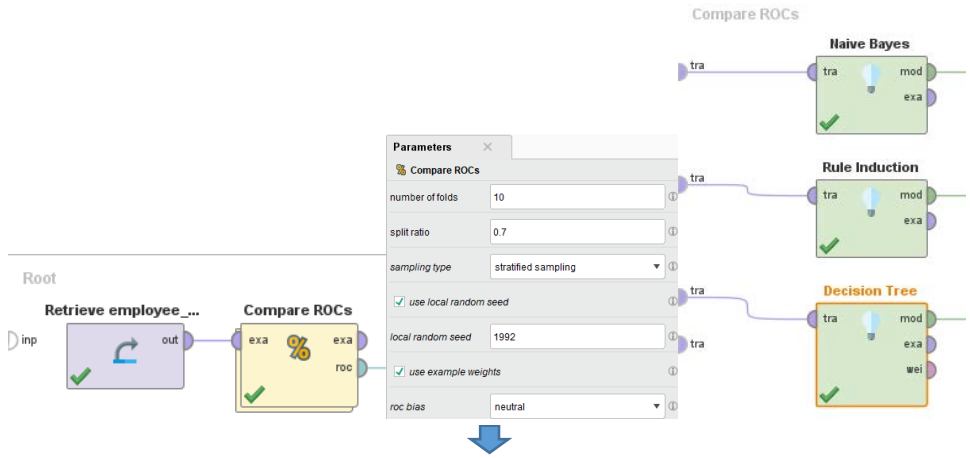
Sursa:
https://github.com/dariyasydykova/open_projects/blob/master/ROC_animation/animations/SD.gif

Realizarea în RapidMiner a unui grafic care compară curbele ROC (Compare ROCs)

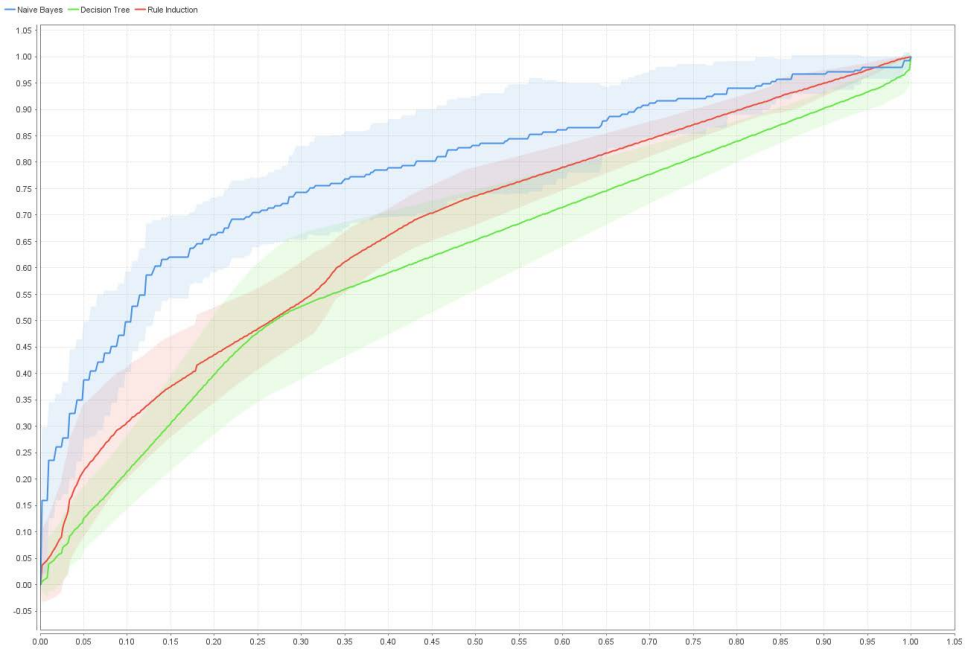
Operatorul Compare ROCs permite compararea curbelor ROC asociate mai multor modele (Figura 4.3-10). Putem astfel evalua vizual și compara performanța modelelor de clasificare dorite, global, respectiv pe secțiuni. Simplu spus, putem vedea dacă un model performează sistematic mai bine comparativ cu altul sau dacă există zone în care acestea performează diferit.

Figura 4.3-10. Operatorul Compare ROCs

Pași:
Conectăm setul de date și operatorul Compare ROCs. Setăm valorile parametrilor conform imaginii. În interiorul operatorului Compare ROCs conectăm operatorii din imagine.



Rezultate:
Obținem un grafic care compară curbele ROC aferente clasificatorilor incluși în analiză. Curba care crește relativ mai repede indică modelul relativ mai performant (Naive Bayes în acest caz). Observăm că performanța modelul Naive Bayes (linia albastră) este sistematic mai bună comparativ cu performanța celorlalte modele, cu excepția colțurilor din stânga-jos și dreapta-sus, adică atunci când ambele măsuri (TPR și FPR) iau valori minime (0), respectiv maxime (1). De asemenea, observăm că în general performanța modelului Rule Induction este superioară performanței modelului Decision Tree (performanțele sunt similare în jurul valorilor TPR = 0.5 și FPR = 0.25-0.3). În funcție de zona de interes, diferențele dintre performanța celor trei modele variază (firesc, în punctele minim și maxim performanțele sunt aceleași).



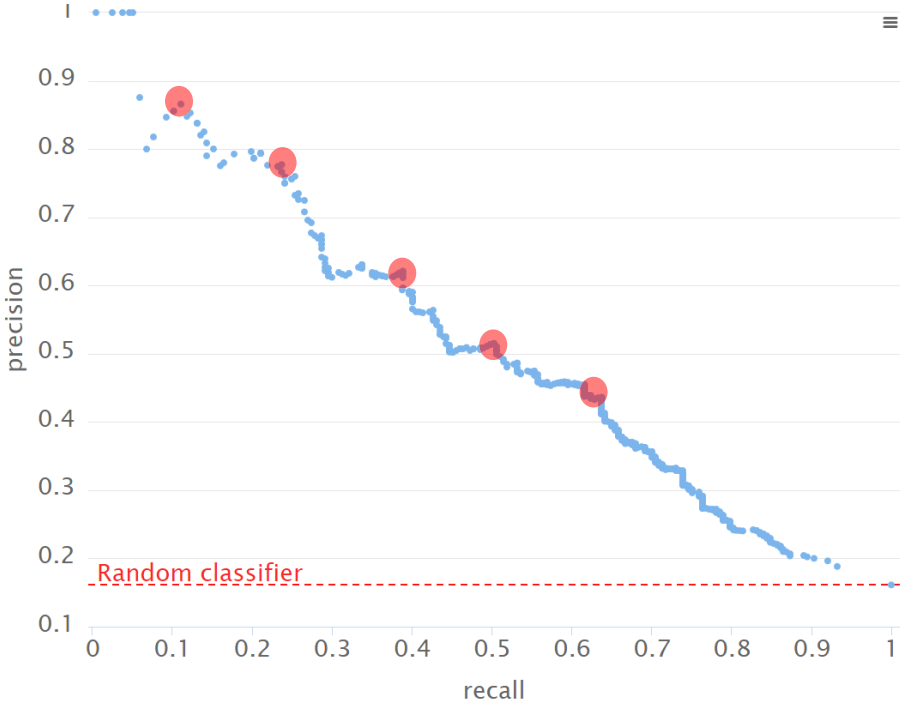
Dacă dorim să vedem care sunt efectiv valorile din spatele unei curbe ROC, folosim operatorul „ROC Curve to Example Set” (vezi procesul 4.3 ROC to Data; în acest caz valorile asociate curbei ROC sunt calculate în cazul unui model în care am folosit validarea încrucișată). Curba ROC asociată clasificatorului Naive Bayes din graficul anterior poate fi construită folosind un grafic de tip scatter populat cu valorile TPR și FPR din setul de date rezultat în urma folosirii operatorului „ROC Curve to Example Set”.

Realizarea în RapidMiner a unui grafic care prezintă relația dintre Precision și Recall (AUPRC)

Pentru a calcula în RapidMiner valoarea AUPRC folosim operatorul Performance (AUPRC) (vezi extensia Operator Toolbox, apoi selectează folderul Performance). Un exemplu de calcul este inclus în procesul „4.3 Compute AUROC & AUPRC". În procesul respectiv am calculat acuratețea modelului (80.5%), graficul și valoarea AUC/AUROC (0.784, adică 78.4%) și valoarea AUPRC în relație cu clasa pozitivă (Yes) (0.516 +/- 0.081, adică 51.6%).

Pentru a construi un grafic cu curba PR (Figura 4.3-11), putem folosi procesul „4.3 PRC Plot". În cadrul acestui proces am folosit același set de date (employee attrition) și am calculat valorile precizie și reamintire pentru 500 praguri de probabilitate (după rularea procesului, pragurile apar la rezultate, tabul Optimize Parameter (Grid), coloana Try.Threshold). Pentru a construi / vedea graficul, folosim valorile precizie și reamintire afișate în setul de date de la tabul ExampleSet (Log to Data). La Visualizations alegem un grafic de tip Scatter / Bubble, apoi punem pe axa OX măsura reamintire, iar pe OY precizia. Pentru a indica modelele de șansă, putem defini o linie orizontală în dreptul punctului 0.161 (rata atriției din setul de date este 16.1%) de pe axa oy (Y-Axis, Plot lines, apoi +, introducem valoarea 0.161 și definim culoarea, tipul și grosimea liniei). Desigur, setările graficului pot fi preluate ușor folosind formatarea definită în fișierul prc.json (folderul Grafice). Dacă dorim o precizie maximă, identificăm doar o mică parte din cazurile pozitive (sub 10% din total), adică reamintirea ia valori mici (modelele din colțul din stânga sus). Dacă dorim o reamintire maximă, precizia va fi foarte mică, adică 0.161 (modelele din colțul din dreapta jos). funcție de situația concretă și de obiectivele analizei, probabil vom prefera modelele situate cât mai aproape de colțul din dreapta sus, adică modelele care pentru aceeași precizie au valori mai mari ale reamintirii (marcate cu fundal roșu).

Figura 4.3-11. Construirea unui grafic cu curba PR



4.4. Vizualizarea performanței (Visual): Gain & Lift charts

În această secțiune vom prezenta alte două măsuri ale performanței unui model: Gain și Lift. Acestea pot fi calculate și reprezentate grafic cu ajutorul operatorului Create Lift Chart.

Logica măsurilor Gain și Lift

Cu privire la un set de date ne interesează să estimăm probabilitatea ca evenimentul de interes să aibă loc. Predicția poate fi realizată cu sau fără un model de clasificare. Predicția fără model, numită și predicție naivă sau aleatoare, se realizează fără ajutorul unui model și, implicit, fără a folosi alte informații despre cazuri cu excepția informației cu privire la proporția cazurilor de interes din total cazuri. În cazul setului nostru de date, dat fiind faptul că 16% dintre angajați au părăsit organizația, putem prezice că probabilitatea ca un angajat să părăsească organizația este 16%. Altfel

spus, dacă alegem aleator 100 angajați din companie și, fără a ne folosi de alte informații, prezicem că fiecare dintre aceștia va părăsi compania, vom ghici corect în aproximativ 16 cazuri. Predicția cu ajutorul unui model se referă la probabilitatea de a prezice corect dacă un caz este pozitiv atunci când folosim un model de clasificare, respectiv și informații despre cazuri. În mod normal, predicția cu ajutorul unui model ar trebui să producă estimări relativ mai corecte. Măsura lift reprezintă gradul în care performanța generală a clasificării se îmbunătățește ca urmare a utilizării unui model de predicție. Cu cât valoarea lift este mai mare decât unitatea (1), cu atât modelul este mai bun. Graficul lift reprezintă vizual eficiența unui model de clasificare. Pentru a realiza acest lucru calculează raportul dintre rezultatele obținute cu ajutorul unul model de predicție, respectiv rezultatele obținute fără un model. În situația în care clasa de interes este cea pozitivă, formula pentru calcularea măsurii lift[30] este:

$$lift = \frac{Predicție\ cu\ model}{Predicție\ fără\ model} = \frac{TP\ /\ P}{P\ /\ N}$$

unde, TP = # cazuri prezise corect ca pozitive, P = # cazuri pozitive, N = # total de cazuri.

În Tabelul 4.4-1 am prezentat sintetic rezultatele unui model de clasificare. Setul de date conține 1000 cazuri, 445 dintre acestea fiind pozitive (de interes). Pentru a calcula măsurile gain și lift procedăm astfel:

- sortăm cazurile descrescător în funcție de probabilitatea estimată de model ca acel caz să fie pozitiv;
- împărțim cazurile în 10 decile (primele 10%, următoarele 10% etc.); funcție de numărul total de cazuri putem alege să le divizăm în 4, 5, 20 sau orice alt număr de grupuri;
- la nivelul fiecărei decile aflăm numărul de cazuri pozitive; de exemplu, în decila 1 toate cele 100 cazuri sunt pozitive;
- calculăm numărul cumulat de cazuri pozitive; de exemplu, considerând primele două decile, avem 195 cazuri pozitive;
- la nivelul fiecărei decile calculăm ponderea cazurilor pozitive identificate în acea decilă din total cazuri pozitive; de exemplu, în cazul primei decile, cele 100 cazuri pozitive reprezintă 22.5% din totalul de 445 cazuri pozitive;

[30] O serie de prezentări video care pot fi utile pentru înțelegerea conceptului de lift sunt următoarele: ROC Curves and Lift Chart | RapidMiner, SAS Tutorial | Lift and Response Charts in SAS, Lift Charts - Classification Evaluation - Business Intelligence with Data Mining.

- calculăm ponderea cumulată a cazurilor pozitive; valoarea obținută reprezintă măsura Gain; de exemplu, la nivelul primelor patru decile (adică primele 40% dintre cazuri), regăsim 80% dintre cazurile pozitive;
  calculăm raportul dintre gain și ponderea cazurilor selectate până la acel moment; de exemplu, la nivelul primei decile, valoarea Lift e 2.25 (22.5 / 10; gain= 22.5, iar ponderea cazurilor e 10%), în cazul decilei 2 Lift ia valoarea 2.19 (43.8 / 20) etc.

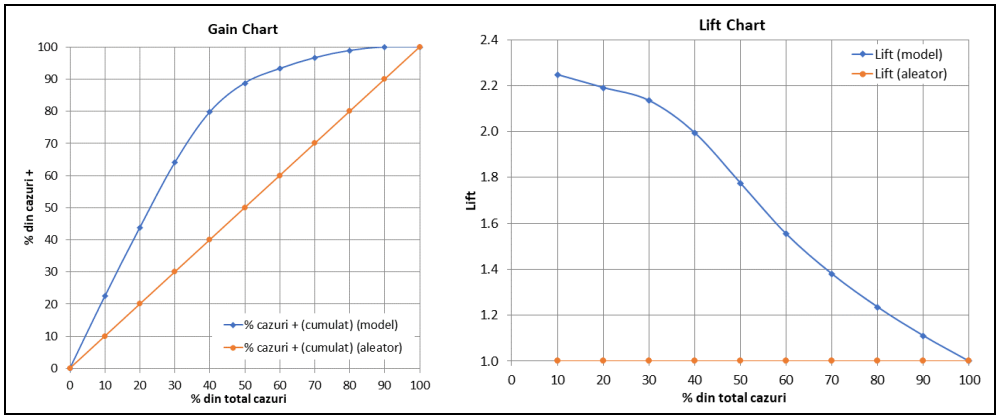
Tabelul 4.4-1. Măsuri ale performanței clasificării: Gain și Lift

decila	# cazuri	% cazuri (cumulat)	# cazuri +	# cazuri + (cumulat)	% cazuri + din total cazuri	Gain (cumulativ)	Lift (cumulativ)
1	100	10	100	100	22.5	22.5	2.25
2	100	20	95	195	21.3	43.8	2.19
3	100	30	90	285	20.2	64.0	2.13
4	100	40	70	355	15.7	79.8	1.99
5	100	50	40	395	9.0	88.8	1.78
6	100	60	20	415	4.5	93.3	1.55
7	100	70	15	430	3.4	96.6	1.38
8	100	80	10	440	2.2	98.9	1.24
9	100	90	5	445	1.1	100.0	1.11
10	100	100	0	445	0.0	100.0	1.00

Pentru a interpreta cât mai ușor valorile calculate în tabelul anterior, le putem reprezenta sub formă grafică (Figura 4.4-1). În graficul din stânga (Gain Chart) observăm că atunci când selectăm cazurile aleator (linia portocalie), ponderea cazurilor pozitive identificare crește liniar în funcție de ponderea cazurilor incluse. De exemplu, selectând aleator 10% din cazuri, găsim 10% din totalul cazurilor pozitive, iar selectând aleator 20% din cazuri, găsim 20% dintre cazurile pozitive etc. Dacă folosim pentru selecția cazurilor un model de clasificare (linia albastră), adică ordonăm cazurile descrescător în funcție de probabilitatea prezisă (relativ la clasa pozitivă), atunci când selectăm primele 10% dintre cazuri (cele cu probabilitatea cea mai mare), găsim 22% din totalul cazurilor pozitive, iar când selectăm 20% dintre cazuri, găsim 44% din totalul cazurilor pozitive etc. Conform graficului Gain, cel mai rentabil e să selectăm 40% dintre cazuri, situație în care vom găsi 80% dintre cazurile pozitive (curba crește abrupt până la acel punct, apoi tinde să se aplatizeze).

Graficul Lift (dreapta) prezintă la nivelul fiecărei decile, comparativ, valorile Lift asociate situației în care clasificăm cazurile aleator (linia portocalie), respectiv folosim un model (linia albastră). Raportat la prima decilă, găsim de două ori mai multe cazuri pozitive atunci când clasificăm cazurile folosind un model comparativ cu situația în care ghicim (clasificare naivă). Pe măsură ce includem și decilele următoare, avantajul relativ al clasificării cu ajutorul modelului se diminuează, ajungând ca în final să fie zero (atunci când lift ia valoarea 1). În baza acestor date, funcție de situația concretă, o orientare a interesului doar spre cazurile din decila 1 sau decilele 1 și 2 sau 1, 2 și 3 sau chiar 1, 2, 3 și 4 reprezintă fiecare o opțiune posibilă, funcție și de obiectivul urmărit.

Figura 4.4-1. Măsuri ale performanței clasificării: Gain și Lift

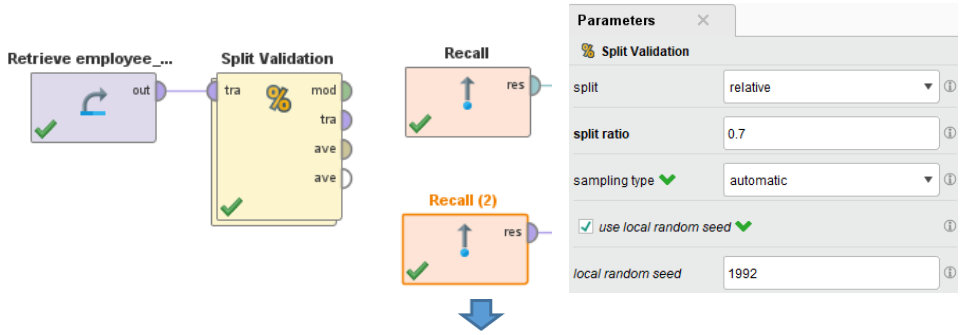


Realizarea unui grafic de tip lift (Create Lift Chart)

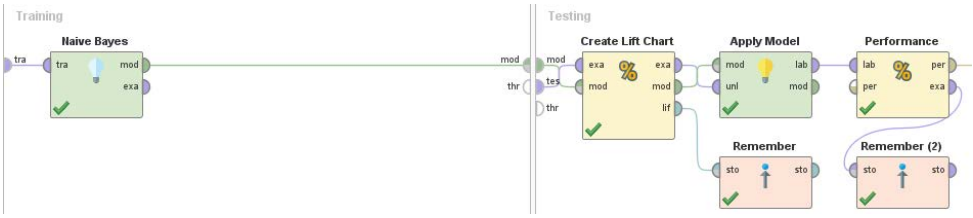
Un exemplu de construcție în RapidMiner a unui grafic de tip Lift este prezentat în Figura 4.4-2. Tot acolo sunt descrise procesul și pașii parcurși pentru a obține graficul. Important, graficul Lift din RapidMiner prezintă valorile Gain și Lift folosind o reprezentare diferită de cea ilustrată în Figura 4.4-1 însă informația este aceeași, iar interpretarea rămâne neschimbată.

Figura 4.4-2. Operatorul Create Lift Chart

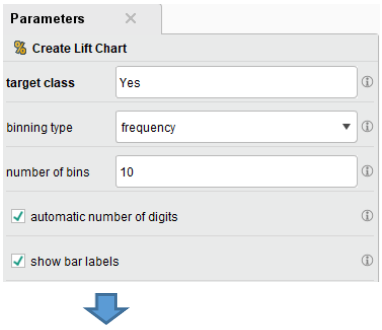
Pasul 1:
Conectăm setul de date și operatorul Split Validation. Setăm valorile parametrilor conform imaginii. Adăugăm operatorul Recall pentru ca softul să afișeze graficul lift (punem numele dorit - Lift Chart în acest caz - la parametrul name și alegem LiftParetoChart la parametrul io object). Similar, folosim Recall pentru a afișa la rezultate și setul de date de test.



Pasul 2:
În interiorul operatorului Split Validation conectăm operatorii din imagine. Operatorii Remember sunt folosiți pentru a păstra în memorie graficul lift și setul de date de test cu predicțiile aferente.



Pasul 3:
La operatorul Create Lift Chart setăm valorile parametrilor precum în imagine.
La operatorul Remember, la parametrul name punem numele graficului (Lift Chart în acest caz), iar la parametrul io object alegem LiftParetoChart.



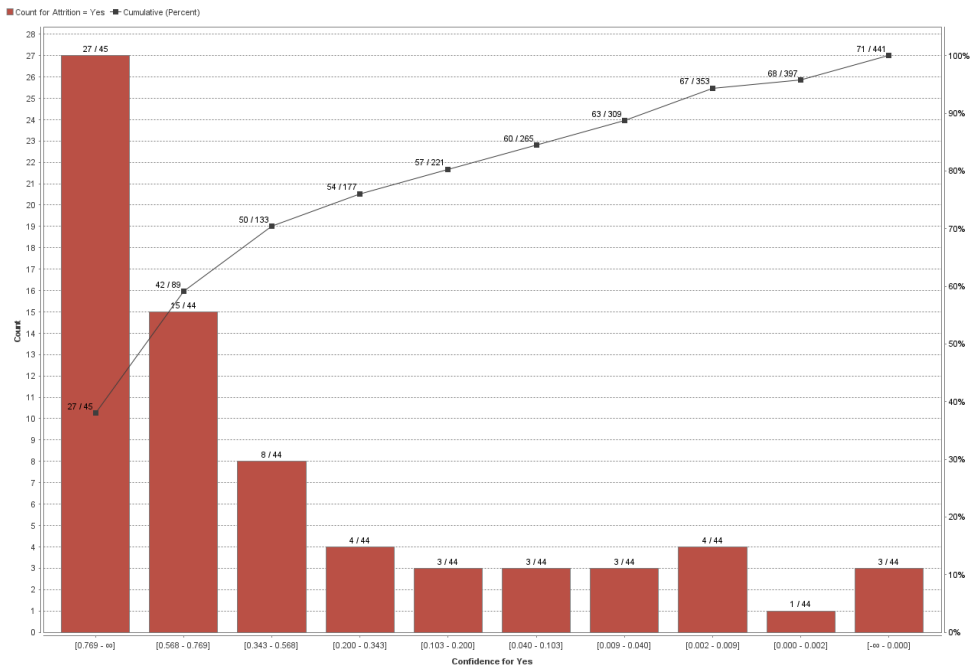
Rezultate:
Obținem un grafic care prezintă o serie de valori utile în procesul de luare a unei decizii de intervenție. Pașii necesari pentru calcularea valorilor incluse în acest grafic sunt următorii:

- se estimează modelul de clasificare (Naive Bayes);
- pe baza acestuia se calculează pentru fiecare caz probabilitatea de interes, adică probabilitatea de a părăsi compania;
- cazurile sunt sortate descendent în funcție de această probabilitate (primele cazuri au probabilitatea cea mai mare, ultimele cea mai mică);

- folosind tipul de grupare (binning) dorit, cazurile sunt divizate într-un număr de grupuri (ales); în acest caz am grupat cazurile în 10 grupuri în funcție de frecvență (10 decile); dat fiind că numărul total de cazuri este 441, avem 44 de cazuri în fiecare grup (45 în primul);
- la nivelul fiecărui grup (decilă) se identifică numărul de cazuri prezise corect ca fiind angajați care pleacă, respectiv numărul de cazuri din acel grup; de exemplu, pentru primul grup (cazurile care au probabilitatea cea mai mare de plecare 77-100%), avem 27 de cazuri de plecare prezise corect (predicția a fost că angajatul pleacă și chiar a plecat) din totalul de 45 de cazuri; dacă ne limităm doar la acest grup, vom găsi aproximativ 38% dintre angajații care efectiv pleacă (27/71=38%); deci, alegând 10% dintre cazuri identificăm 38% dintre plecări; dacă îi includem și pe cei din decila doi, vom găsi aproximativ 59% dintre plecări (27+15=42; 42/71=59%); includerea și a cazurilor din decila trei crește procentul la 70%; simplu spus, controlând 30% dintre cazuri identificăm 70% dintre plecări; acest prag este de altfel și cel optim în cazul acestui exemplu (desigur, funcție și de situația concretă: resurse, costuri etc.).

Curba ascendentă din acest grafic reprezintă curba Gain. De exemplu, pentru decila 1 valoarea Gain este 0.6 (=27/45), iar pentru decila 2 este 0.47 (=42/89). Valorile Lift nu apar, dar pot fi calculate ușor din valorile Gain. De exemplu, pentru decila 1 valoarea Lift este 3.7 (27/45=0.60; 71/441=0.16; 0.60/0.16=3.7); în decila 1 avem 27 cazuri pozitive din 45; în tot setul de test avem 71 de cazuri pozitive din 441; raportul dintre cele două probabilități, 60% și 16%, reprezintă valoarea Lift și este egală cu 3.7.

În concluzie, rezultatele analizei sugerează că este cel mai avantajos să ne concentrăm intervenția (măsura pe care o luăm pentru a reduce situațiile de părăsire a companiei) asupra cazurilor din primele două decile (eventual primele trei), adică cele care au un prag al probabilității estimate de plecare de 0.57 (57%). Procedând astfel, vom identifica 42 (50 dacă luăm primele trei decile) din cele 71 de cazuri de angajați care pleacă.



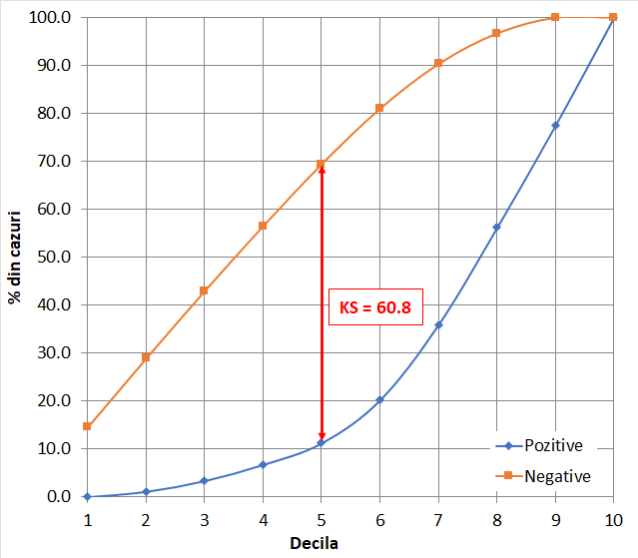
Graficul K-S

Măsurile Lift și Gain iau în calcul doar numărul de cazuri pozitive, prin urmare sunt relativ mai utile atunci când ne interesează calitatea modelului în cazul clasei pozitive. Graficul K-S (Kolmogorov-Smirnov plot) reprezintă performanța unui model de clasificare folosind informația cu privire la numărul de cazuri pozitive, respectiv negative, deci măsura este mai utilă dacă suntem interesați de calitatea generală a modelului. Simplu spus, K-S este o măsură a gradului de separare dintre distribuțiile cazurilor pozitive și negative. K-S ia valori în intervalul 0-100, unde 100 indică o separare perfectă (un model perfect), iar 0 contrariul. Pornind de la același exemplu (Figura 4.4-1), putem construi un tabel care include numărul de cazuri pozitive și negative, numărul cumulat al acestora, respectiv ponderea cumulată (Tabelul 4.4-2). Măsura K-S este egală cu diferența dintre ponderea cumulată a cazurilor pozitive și ponderea cumulată a cazurilor negative. Aceste valori pot fi reprezentate vizual sub forma graficului K-S (Graficul 4.4-1). Diferența maximă se atinge în cazul decilei 6, valoarea K-S fiind 60.8, deci undeva puțin peste mijlocul intervalului de variație. Dat fiind faptul că, în practică, K-S ia cel mai adesea valori peste 50 (modelul de clasificare propus este în general cel puțin egal cu modelul de clasificare naiv), rezultă că modelul nostru are o performanță relativ bună.

Tabelul 4.4-2. O măsură a performanței clasificării: K-S

Decila	# cazuri +	# cazuri -	# cazuri + cum.	# cazuri – cum.	% cazuri + cum.	% cazuri – cum.	K-S
1	0	445	0	445	0.0	14.5	14.5
2	5	440	5	885	1.1	28.9	27.8
3	10	430	15	1315	3.4	43.0	39.6
4	15	415	30	1730	6.7	56.5	49.8
5	20	395	50	2125	11.2	69.4	58.2
6	40	355	90	2480	20.2	81.0	60.8
7	70	285	160	2765	36.0	90.4	54.4
8	90	195	250	2960	56.2	96.7	40.6
9	95	100	345	3060	77.5	100.0	22.5
10	100	0	445	3060	100.0	100.0	0.0

Graficul 4.4-1. O măsură a performanței clasificării: K-S



5. Regresia logistică (Logistic Regression)

Nume operator	Logistic Regression
Categoria majoră	învățare asistată (supervised learning)
Tip model	clasificare (classification)
Grup	algoritm de regresie
Atribut dependent	categorial
Atribute independente	numerice, categoriale
Gestionează valorile lipsă	da (independente)
Distorsiune (bias)	mare
Varianță (variance)	mică

Bibliografie utilizată și recomandată:

- Analiza datelor în cercetarea psihologică (Sava, 2011, Chapter 7)
- Applied Logistic Regression (Hosmer et al., 2013)
- Logistic Regression: Binary and Multinomial (Garson, 2014)
- Practical Guide to Logistic Regression (Hilbe, 2016)
- Data mining: a tutorial-based primer (Roiger, 2017, Chapter 11.5)
- Discovering Statistics using IBM SPSS (Field, 2018, Chapter 20)
- Data Science: Concepts and Practice (Kotu & Deshpande, 2019, Chapter 5.2)
- Logistic Regression: A Primer (Pampel, 2020)
- Data Analytics for the Social Sciences: Applications in R (Garson, 2021, Chapter 4)
- Machine Learning for Social and Behavioral Research (Jacobucci et al., 2023, Chapter 4.3)
- Machine Learning for Business Analytics: Concepts, Techniques and Applications in RapidMiner (Shmueli et al., 2023, Chapter 10)

Termenul de „regresie” din denumirea acestui algoritm este oarecum înșelător. Dacă avem în vedere obiectivul urmărit, regresia logistică urmărește să clasifice

cazurile, să prezică apartenența acestora la clasele atributului de interes (variabilei dependente), nu să prezică valorile unei variabile metrice precum în cazul regresiei liniare. Prin urmare, considerăm că regresia logistică nu este un algoritm de regresie, ci unul de clasificare. Probabil termenul de regresie a trecut testul timpului ca urmare a obișnuinței și / sau pentru că algoritmul de regresie logistică apelează la regresia liniară atunci când estimează coeficienții modelului.

Principala caracteristică a acestui clasificator (și de altfel a tuturor celorlalți prezentați în acest manual) constă în faptul că variabila dependentă este categorială, mai exact, în cazul exemplelor discutate în acest volum, categorială de tip binominal / binar.[1] Altfel spus, variabila dependentă are doar două clase, ia doar două valori (variante de răspuns), 0 și 1 (Shmueli et al., 2023, p. 271).[2] Câteva exemple de astfel de clase sunt următoarele: adevărat / fals, da / nu, pleacă / nu pleacă, cumpără / nu cumpără, fraudă / non-fraudă, bolnav / sănătos.

Un model de regresie logistică asumă faptul că probabilitatea de succes este determinată de funcția standard de densitate cumulativă logistică (cumulative standard logistic distribution). Un astfel de model estimează probabilitatea ca variabila dependentă să ia valoarea 1 (succes / evenimentul de interes are loc) pentru o anumită combinație de valori ale predictorilor. Pentru estimare se folosește metoda „maximum likelihood" (verosimilitate maximă) (spre deosebire de regresia liniară care folosește metoda celor mai mici pătrate) (Shmueli et al., 2023, p. 275).

5.1. Logica și pașii algoritmului

Începem acest sub-capitol cu prezentarea și ilustrarea unor concepte esențiale pentru înțelegerea regresiei logistice: probabilitate, șanse și raport de șanse (probability, odds, odds ratio), respectiv relațiile dintre ele. În continuare descriem tipurile de coeficienți obținuți în urma estimării unui model de regresie logistică, prezentăm asumpțiile algoritmului și posibilele probleme care pot să apară.

[1] Desigur, putem construi modele de predicție și în cazul unor variabile dependente cu mai mult de două categorii / clase. Dacă dorim să realizăm în RapidMiner un model de regresie logistică în care dependenta este categorială, e mai simplu să folosim operatorul Logistic Regression, iar dacă dependenta este multi-categorială trebuie să folosim operatorul GLM (Generalized Linear Model).

[2] În general convenția este că valoarea 0 este considerată a fi un rezultat negativ (eșec), iar valoarea 1 un rezultat pozitiv (succes).

Probabilitate, șanse, raport de șanse (probability, odds, odds ratio)

Pentru a descrie logica algoritmului de regresie logistică prezentăm un exemplu simplu, intuitiv, cu un singur predictor. Pe parcursul acestuia vom introduce și câteva concepte necesare pentru înțelegerea logicii și pașilor algoritmului de regresie logistică. Să presupunem că ne interesează să analizăm relația dintre existența copiilor și părăsirea companiei. Mai concret, vrem să vedem dacă angajații fără copii tind să-și dea demisia într-o măsură mai mare comparativ cu angajații cu copii. Observăm că ambele variabile au două variante de răspuns, da/nu, deci sunt de tip binar. Prin urmare, urmărim să prezicem dacă un angajat pleacă sau nu pleacă în funcție de existența copiilor sau, mai exact, estimăm probabilitatea / șansele[3] ca un angajat să plece (implicit, să nu plece) în funcție de atributul „are copii".

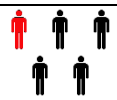
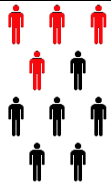
Pe un eșantion de 10 cazuri (Tabelul 5.1-1), observăm că 4 angajați pleacă și 6 nu pleacă (pe ultima linie a tabelului 4 persoane sunt colorate cu roșu, cele care pleacă, 6 cu negru, cele care nu pleacă). Relativ la un atribut categorial putem calcula probabilitatea (probability) ca o persoană să aparțină unei anumite clase ca raport între numărul persoanelor din acea clasă și numărul total de persoane. Astfel, pentru acest exemplu, probabilitatea ca un angajat să părăsească firma este 40% (4/10=0.4 sau 40%). Probabilitatea de a nu pleca este 60% (6/10=0.6 sau 60%). Observăm că suma celor două probabilități este 1 (100%). Nu este deloc o întâmplare. Dacă însumăm probabilitățile asociate tuturor claselor unui atribut, vom obține de fiecare dată 1 (100%). Considerând atributul care indică existența sau nu a copiilor, observăm că avem un număr egal de persoane cu copii, respectiv fără copii (5), deci și în acest caz probabilitatea asociată fiecărei clase este 50% (5/10=0.5 sau 50%), iar suma este 1 (100%).

Pentru a calcula șansele (odds), împărțim numărul de cazuri dintr-o clasă la numărul de cazuri din cealaltă clasă. Alternativ, șansele pot fi calculate ca raport între cele două probabilități asociate claselor. Pentru acest exemplu, la nivelul întregului eșantion (Tabelul 5.1-1, ultima linie), șansele clasei „pleacă" relativ la clasa „nu pleacă" sunt egale cu 4:6 (4 la 6), deci 1:1.5, adică 0.67 (obținem aceeași valoare dacă împărțim între ele cele două probabilități asociate claselor (0.4/0.6=0.67)). În traducere, la fiecare angajat care pleacă există un angajat și jumătate care nu pleacă

[3] În limbajul statistic, spre deosebire de limbajul comun, conceptele șanse (odds) și probabilitate (probability) se referă la lucruri diferite (deși strâns legate între ele), după cum vom vedea în continuare.

(sau, schimbând clasa de referință, la fiecare angajat care nu pleacă, există 0.67 angajați care pleacă).

Tabelul 5.1-1. Probabilitate, șansă, raport de șanse: un exemplu de calcul

	Clasa de interes este Pleacă (Roșu)	**Probabilitate (probabilities-p)**	**Șanse (odds)**	**Raport de șanse (odds ratio - OR)**	**Logit[4]**
Are copii	**Formulă Observații / Cazuri**	$p = \frac{\#\ obs.R}{\#\ obs.}$	$odds = \frac{\#\ obs.R}{\#\ obs.N} = \frac{p}{1-p}$	$OR_{roşu} = \frac{odds_{gi}}{odds_{gj}}$	ln(OR)
0 (Nu)		3/5 (3 din 5) 60% sau 0.6	3:2 (1.5 la 1) 0.6 / 0.4 = 1.5	$OR_{roşu\|g0} = \frac{1.5}{0.25} = 6$	1.792
1 (Da)		1/5 (1 din 5) 20% sau 0.2	1:4 (1 la 4) 0.2 / 0.8 = 0.25	$OR_{roşu\|g1} = \frac{0.25}{1.5} = 0.17$	-1.792
Total		4/10 40% sau 0.4	4:6 = 1:1.5 (1 la 1.5) 0.4 / 0.6 = 0.67	-	-

Desigur, probabilitățile și șansele pot fi calculate și la nivelul unor sub-populații / sub-eșantioane (de exemplu, tineri vs. vârstnici, femei vs. bărbați, conducere vs. execuție), nu doar la nivelul întregii populații / întregului eșantion. Pentru exemplul nostru, dacă împărțim cazurile în funcție de existența sau nu a copiilor, probabilitatea cazurilor R (roșii, cei care pleacă) este 60% la nivelul Grupului 0 (Fără copii), respectiv 20% la nivelul Grupului 1 (Cu copii). Dacă un angajat are copii (Grupul 1), șansele să părăsească firma sunt 1 la 4 (0.25). Dacă un angajat nu are copii (Grupul 0), șansele să plece sunt 3 la 2, adică 1.5 la 1 (1.5).

Ce se întâmplă cu șansele de a părăsi firma în funcție de existența sau nu a copiilor? Sunt mai mari, mai mici, la fel? Putem răspunde la această întrebare calculând raportul de șanse (odds ratio = OR), adică, în acest caz, raportul dintre șansele de a pleca în cazul celor două clase: angajați cu copii (odds = 0.25) și angajați fără copii (odds = 1.5). Funcție de clasa de referință aleasă, OR (raportul de șanse) este egal

[4] La ce putere trebuie să-l ridicăm pe e (numărul lui Euler sau baza logaritmilor naturali; e = 2.718...) pentru a obține valoare OR (odds ratio) respectivă? În acest caz, $e^{1.79} = 6$, iar $e^{-1.79} = 0.17$, respectiv ln(6) = 1.792, iar ln(0.17) = -1.792.

cu 6 sau 0.17, deci șansele de a pleca din firmă sunt de 6 ori mai mari în cazul angajaților fără copii raportat la cei cu copii. Alternativ, putem spune că șansele de plecare sunt de 6 ori mai mici în cazul angajaților cu copii raportat la cei fără copii.

Generalizând, o valoare supraunitară a raportului de șanse indică relativ mai multe șanse ca evenimentul analizat să aibă loc atunci când o observație / un caz aparține unei anumite clase (predictorul ia o anumită valoare, în cazul predictorilor metrici). Invers, o valoare subunitară semnifică șanse relativ mai puține, iar un raport de șanse egal cu 1 arată că nu există o relație de asociere între cele două atribute (în termeni cauzali, predictorul în discuție nu modifică / influențează șansele de apartenență la clasa de interes a atributul prezis). De exemplu, dacă raportul de șanse asociat exemplului din Tabelul 5.1-1 ar fi fost 1 (sau apropiat de 1), am fi concluzionat că nu există o relație de asociere între calitatea de părinte și părăsirea companiei.

De la probabilitate la șanse, apoi la logit și înapoi

Regresia logistică este folosită pentru a modela / prezice apartenența cazurilor la clasele unui atribut binar. De obicei ne interesează apartenența la clasa pozitivă / de interes / da / evenimentul are loc (etichetată cu valoarea 1). Pentru a face această predicție trebuie să calculăm probabilitatea de apartenență la clasa de interes (adică 1) (probabilitatea ia valori în intervalul [0;1]), respectiv să stabilim un prag de referință, adică o valoare a probabilității în funcție de care alocăm cazurile la clasa 1, respectiv 0. De obicei, softurile de analiză a datelor permit stabilirea pragului dorit chiar dacă, implicit, pragul este setat la valoarea 0.5. Un prag setat la 0.5 înseamnă că toate cazurile cu o probabilitate estimată mai mică de 0.5 (50%) vor fi alocate la clasa 0 (prezicem că aparțin clasei 0), iar cele cu o probabilitate estimată de cel puțin 0.5 vor fi alocate la clasa 1 (prezicem că aparțin clasei 1).

Pentru a estima aceste probabilități nu putem folosi regresia liniară, cel puțin pentru următoarele două motive:

- este foarte posibil ca unele dintre predicțiile realizate folosind un model de regresie liniară să ia valori în afara intervalului de variație [0;1], ori, o probabilitate nu poate lua valori mai mici de 0 sau mai mari de 1;
- asumpția varianței constante a erorilor (necesară în cazul regresiei liniare) nu este respectată, deci erorile standard asociate coeficienților nu sunt

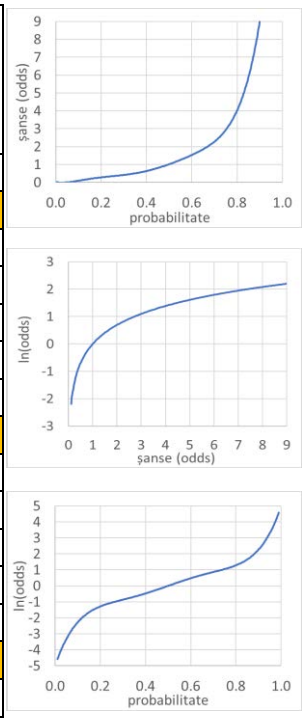
valide, prin urmare inferențele realizate în baza unui astfel de model pot fi greșite.

Ce putem face în acest caz? Răspunsul e destul de simplu: translatăm probabilitățile într-un sistem de măsurare care îndeplinește condițiile menționate anterior; adică, în loc să prezicem probabilitățile, prezicem altceva, acest altceva fiind legat de probabilități printr-o formulă (funcție); în cazul regresiei logistice funcția folosită este logit, deci vom prezice valorile logit, adică logaritmul șanselor sau ln(odds); după estimarea modelului și realizarea predicțiilor, acestea din urmă sunt transformate în probabilități. Dincolo de această descriere generală, concret procedăm astfel (vezi exemplul din Tabelul 5.1-2):

- Observăm că probabilitățile i-au valori în intervalul [0;1], valoarea 0.5 indicând o probabilitate egală de apartenență la cele două clase.
- **Pasul 1 - transformarea probabilităților în șanse (odds).** Pentru a calcula șansele (odds), împărțim probabilitatea clasei pozitive (1) la probabilitatea clasei negative (0). O valoarea a șanselor (odds) egală cu 1 indică faptul că probabilitatea asociată fiecăreia dintre cele două clase este 0.5 sau 50% (deoarece 0.5/0.5=1). Observăm că intervalul de variație asociat măsurii odds este [0;∞), deci nici această măsură nu este potrivită pentru nevoile noastre deoarece ar trebui ca valorile măsurii să acopere, teoretic, tot intervalul (-∞;∞). Mai mult, dat fiind faptul că odds ia valori în intervalul [0;∞), este incorect să folosim regresia liniară pentru a modela relația dintre cele două atribute (modelul ar prezice și valori negative, deci mai mici de 0). Totuși, dacă transformăm valorile odds în valori logit, putem apela la regresia liniară.
- **Pasul 2 - transformarea șanselor (odds) în valori logit.** Această transformare se face prin logaritmarea valorilor odds, adică folosind formula logit = ln(odds). Măsura nou obținută se numește logit. Observăm că, de această dată, valorile rezultate acoperă tot intervalul (-∞;∞) (exact ceea ce ne dorim). Valorile logit sunt și cele pe care le vom modela / prezice. Observăm totodată că valorile logit sunt relativ mai dificil de interpretat (nu au o semnificație imediată).
- **Pasul 3 - transformarea valorilor logit în șanse (odds)** prin exponențierea valorilor logit, respectiv în **probabilități** prin calcularea raportului dintre șanse (odds) și 1 + șanse.

Tabelul 5.1-2. „Drumul" de la probabilitate la șanse, apoi la logit și înapoi

Prob. $= p$	→ Odds $= \frac{p}{1-p}$	→ Logit $= \ln(odds)$ $= a + bX$	→ Odds $= e^{logit}$ $= e^{a+bX}$	→ Prob. $= \frac{e^{logit}}{1+e^{logit}}$
0.000	0.000	-∞	0.000	0.000
0.001	0.001	-6.907	0.001	0.001
0.010	0.010	-4.595	0.010	0.010
0.100	0.111	-2.197	0.111	0.100
0.200	0.250	-1.386	0.250	0.200
0.300	0.429	-0.847	0.429	0.300
0.400	0.667	-0.405	0.667	0.400
0.500	1.000	0.000	1.000	0.500
0.600	1.500	0.405	1.500	0.600
0.700	2.333	0.847	2.333	0.700
0.800	4.000	1.386	4.000	0.800
0.900	9.000	2.197	9.000	0.900
0.990	99.000	4.595	99.000	0.990
0.999	999.000	6.907	999.000	0.999
1.000	∞	∞	∞	1.000



** Probabilitatea prezisă (p) nu poate fi exact 0 sau 1.*

Să presupunem că probabilitatea ca un angajat oarecare să părăsească firma la care lucrează este 40%, adică $p = 0.40$, deci probabilitatea ca un angajat să nu plece este 60%. Prin urmare, șansele (odds) ca un angajat să plece relativ să nu plece sunt 0.4/0.6, adică 1 la 1.5 (la fiecare 1.5 angajați care nu pleacă există un angajat care pleacă) sau 0.667. Valoarea logit asociată acestei șanse este $ln(0.667) = -0.405$. Aceasta este de altfel și valoarea pe care o obținem atunci când estimăm pe aceleași date un model de regresie logistică fără niciun predictor (valoarea pe care o ia constanta ecuației de regresie logistică, $a = -0.405$). Prin urmare, ecuația de regresie logistică pentru situația fără predictori ia forma:

$$logit(p) = ln(odds) = ln\left(\frac{p}{1-p}\right) = a$$

Dacă exponențiem ambele părți ale ecuației, rezultă că $\frac{p}{1-p} = e^{a}$, adică $\frac{p}{1-p} = e^{-0.405} = 0.667$, deci $p = 0.4$ (exact probabilitatea de la care am plecat).

Dacă includem un predictor, ecuația de regresie logistică devine:

$$logit(p) = ln(odds) = ln\left(\frac{p}{1-p}\right) = a + bX$$

unde p este probabilitatea ca evenimentul să aibă loc, a este constanta, X este valoarea luată de variabila independentă (predictorul), iar b este coeficientul de regresie logistică nestandardizat. De exemplu, relația dintre părăsirea companiei și gradul de insatisfacție poate fi exprimată astfel:

$$logit(plece) = ln(odds\ plece\ relativ\ la\ nu\ plece) = -5.158 + 0.804 * insatisfactie$$

Dacă în cazul unui angajat insatisfacția este nulă (X= 0), valoarea logit este -5.158, deci șansele ca acesta să plece relativ la să nu plece sunt egale cu 0.006 ($e^{-5.158}$), iar probabilitatea de a pleca este 0.006$\left(\frac{0.006}{1+0.006}\right)$. Dacă un alt angajat are un scor de insatisfacție egal cu 5, valoarea logit devine -2.746 (-5.158 + 0.804 * 5 = -5.158 + 4.020 = -1.137), deci șansele ca acesta să plece relativ la să nu plece sunt egale cu 0.321 ($e^{-1.137}$), iar probabilitatea de a pleca este 0.243$\left(\frac{0.321}{1+0.321}\right)$. Desigur, putem calcula probabilitatea folosind direct formula:

$$p = \frac{e^{a+b*X}}{1 + e^{a+b*X}}$$

Ce și cum interpretăm?

În urma estimării unui model de regresie logistică obținem patru tipuri de măsuri: coeficienți logit nestandardizați, coeficienți logit standardizați, rapoarte de șanse (odds ratio) și probabilități. Fiecare dintre acestea poate fi folosită pentru a exprima sintetic relațiile dintre atributul de interes și predictori. Însă, modul în care interpretăm măsurile, cât de intuitive și utile sunt acestea diferă. În cele ce urmează prezentăm pentru fiecare măsură interpretarea generală, un exemplu simplu, respectiv o evaluare a utilității și interpretabilității.

Interpretarea în termeni de **coeficienți logit nestandardizați**. Fie b coeficientul de regresie logistică (logit) nestandardizat. Unitatea de măsură a unui coeficient nestandardizat este unitate de măsură a predictorului asociat (unitatea de măsură originală); atunci când variabila independentă se modifică (crește / scade) cu o unitate, ln(odds) asociat clasei de interes a variabilei dependente se modifică (crește / scade) cu b unități, ceilalți predictori fiind constanți. O valoare a lui b mai mare de

zero semnifică o relație pozitivă cu variabila prezisă, iar o valoare mai mică de zero semnifică o relație negativă; în cazul exemplului nostru, atunci când insatisfacția (variabila independentă) crește cu un punct (pe o scală de la 0 la 10), valoarea ln(odds) asociată clasei „pleacă" crește, în medie, cu 0.804; prin urmare, un nivel ridicat al insatisfacției este însoțit de șanse mai mari de părăsire a companiei. Este relativ mai dificil să interpretăm ce semnifică exact o anumită valoare logit deoarece: (1) operăm rar cu logaritmi și șanse, deci e relativ mai dificil să intuim ce reprezintă concret logaritmul unei șanse, (2) variabilele independente au adesea unități de măsură, respectiv intervale de variație, destul de diferite între ele, deci coeficienții logit nestandardizați nu pot fi comparați între ei și (3) relația dintre logit și probabilitate este non-liniară (mai exact, este în formă de S, adică este descrisă de o funcție de tip sigmoid). Mai mult, nu putem compara nici măcar coeficienții acelorași predictori în cazul a două modele aflate într-o relație de tip „cuib" (nested).[5] Pentru aceste motive, interpretarea în termeni de coeficienți logit nestandardizați este utilizată relativ mai rar.

Interpretarea în termeni de **coeficienți logit standardizați**. Fie b_{std} coeficientul logit standardizat. Pentru a obține valorile b_{std}, putem standardiza în prealabil variabilele incluse în model (le transformăm astfel încât media fiecăreia devine 0, iar abaterea standard 1) sau aplicăm o formulă coeficientului nestandardizat (vezi mai jos). RapidMiner calculează automat și coeficienții standardizați, fără ca utilizatorul să trebuiască să standardizeze variabilele. Unitatea de măsură a unui coeficient standardizat devine abaterea standard. În funcție de obiectivele urmărite standardizarea se poate face relativ la variabila independentă, la variabila dependentă sau la ambele (complet). Standardizarea coeficientului logit asociat predictorului X relativ la variabila fiecare tip de variabilă se realizează astfel:

- independentă, X: $coef.std(X) = coef.nestd(X) * sd(X)$
- dependentă, Y: $coef.std(X) = coef.nestd(X) * \frac{1}{sd(Y^*)}$
- ambele, X și Y: $coef.std(X) = coef.nestd(X) * \frac{sd(X)}{sd(Y^*)}$

unde:

coef.std(X) = coeficientul logit standardizat asociat predictorului X;

[5] Adăugarea unui predictor la un model poate schimba coeficienții predictorilor existenți chiar dacă predictorul nou nu corelează cu niciunul dintre aceștia (Norton & Dowd, 2018; Williams & Jorgensen, 2023). Pentru a putea totuși compara efectele, putem folosi una dintre următoarele soluții: standardizarea lui Y, metoda KHB (Karlson/Holm/Breen) și utilizarea efectelor maginale (vezi secțiunea despre probabilități).

coef.unstd(X) = coeficientul logit nestandardizat asociat atributului X;
sd(X) = abaterea standard a atributului X;
sd(Y^*) = abaterea standard a variabilei dependente Y (latente, nu observate).

Formulele utilizate pentru fiecare dintre tipurile de standardizare diferă parțial de la un autor la altul sau de la un soft la altul. Aici am prezentat soluția implementată în Stata (varianta Agresti, pachetul spost). Firesc, valorile obținute trebuie interpretate în funcție de tipul de standardizare utilizat. În cazul standardizării complete, interpretarea este următoarea: o creștere / scădere cu o abatere standard a variabilei independente este însoțită de o creștere / scădere cu b_{std} abateri standard a ln(odds) asociate clasei de interes a variabilei dependente. O valoare a coeficientului b_{std} mai mare de zero semnifică o relație pozitivă cu variabila prezisă, iar o valoare mai mică de zero o relație negativă. Pentru exemplul nostru, valoarea standardizată (complet, adică ținând cont de ambele variabile) a coeficientului asociat atributului insatisfacție este aproximativ 0.84, deci o creștere cu o abatere standard a insatisfacției (3.5 puncte în acest caz) este însoțită de o creștere cu 0.84 abateri standard a ln(odds) asociate clasei „pleacă". Spre deosebire de coeficienții logit nestandardizați, cei standardizați pot fi comparați între ei (desigur, doar dacă s-a folosit același tip de standardizare, respectiv același model), însă, în rest, prezintă aceleași dezavantaje sau chiar unele noi.[6]

Interpretarea în termeni de **odds ratio**. În acest caz transformăm coeficientul logit nestandardizat (b) în raport de șanse (odds ratio = OR) folosind formula $OR = e^b$. OR se interpretează astfel: o creștere / scădere a variabilei independente cu o unitate, este însoțită de o creștere / scădere a șanselor de apartenență la clasa pozitivă (1) relativ la cea negativă (0), cu un factor egal cu valoarea OR (adică e^b). O valoare OR mai mare de unu semnifică o relație pozitivă cu variabila prezisă (cresc șansele relative de apartenența la clasa 1), iar o valoare mai mică de unu indică o relație negativă (scad șansele relative de apartenență la clasa 1). În cazul exemplului nostru, atunci când insatisfacția crește cu un punct (pe o scală de la 0 la 10), șansele de a pleca relativ la a nu pleca se dublează aproximativ ($e^{0.804} = 2.2$).[7] Alternativ, putem spune că raportul de șanse în favoarea plecării se modifică cu un factor egal cu 2.2,

[6] Ce înseamnă o abatere standard în cazul unui atribut binar? Dacă varianța (abaterea standard) unui predictor diferă între grupuri, valorile standardizate vor diferi și ele, deci nu e corect să comparăm coeficienții standardizați asociați acelor grupuri.

[7] Putem calcula raportul de șanse și pentru situații în care variabila independentă se modifică cu mai multe unități. De exemplu, dacă nivelul de insatisfacție crește cu 3 puncte, șansele ca un angajat să plece din companie sunt de 11 ori mai mari comparativ cu șansele ca angajatul să rămână (celelalte condiții fiind egale) ($e^{3*b} = e^{3*0.804} = e^{2.412} = 11.16$).

respectiv șansele relative de plecare cresc cu 120% ((2.2 – 1) * 100).[8] Analiza relațiilor în termeni de OR este o modalitate intuitivă și utilă practic, dar valorile obținute pot fi interpretate doar în termeni relativi. Ce înseamnă practic „o dublare a șanselor"? Pare să fie foarte mult însă acest lucru nu e neapărat adevărat: dacă șansele inițiale erau 10^{-6}, adică una la un milion, dublarea șanselor modifică foarte puțin situația concretă deoarece probabilitatea de apariție a acelui fenomen rămâne în continuare extrem de mică. Însă, pe de altă parte, raportul de șanse exprimă sintetic (o singură valoare numerică) relația dintre două variabile. Simplu spus, OR este același indiferent de porțiunea de referință din intervalul de variație a variabilei independente. Spre deosebire de OR, probabilitatea poate să varieze diferit, pe anumite porțiuni putem observa o creștere / scădere mai accelerată, pe altele una mai redusă.

Interpretarea în termeni de **probabilități**. Folosind modelul de regresie logistică estimat, putem calcula valorile logit prezise pentru diferite combinații de valori ale variabilelor independente. În continuare, putem transforma aceste valori logit în probabilități, apoi putem reprezenta grafic relația dintre probabilități și una sau mai multe variabile independente. Probabilitatea este o măsură mult mai intuitivă, practică și interpretabilă în termeni absoluți. De exemplu, în cazul unui angajat cu un nivel al insatisfacției de 10 puncte, probabilitatea de plecare este 95%,[9] iar în cazul unuia cu un nivel al insatisfacției egal cu 5, probabilitatea este 42%. Cu privire la o astfel de situație putem afirma că primul angajat are o probabilitate foarte mare de a pleca, al doilea este indecis, respectiv că șansele de plecare ale primului sunt de două ori mai mari.

Interpretarea în termeni de probabilități este folosită tot mai frecvent, cel mai adesea fiind preferată o formă grafică de exprimare a relației dintre variabila dependentă și variabila(ele) independente,[10] una de tipul celei ilustrate în Figura 5.1-1. În grafic se observă că probabilitatea de plecare din companie este sistematic

---

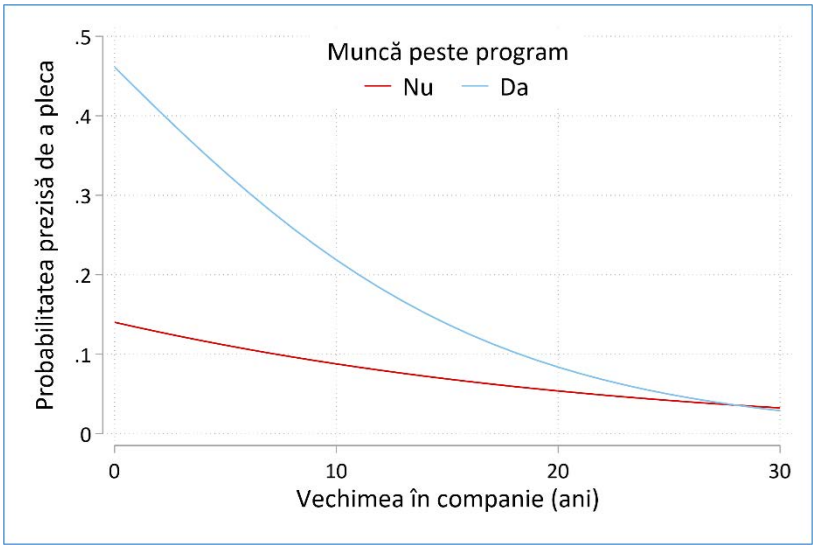
[8] Valoarea OR depinde de ce date și model folosim, deci corect ar fi să spunem că valoarea obținută este doar o estimare condiționată de restul atributelor incluse în model. Prin urmare, strict tehnic nu putem compara estimări ale OR obținute pe seturi de date diferite, sub-populații diferite, respectiv modele diferite (Norton & Dowd, 2018).

[9] $$p = \frac{e^{-5.158+0.804*10}}{1+e^{-5.158+0.804*10}} = \frac{e^{2.883}}{1+e^{2.883}} = \frac{17.871}{18.871} = 0.947$$

[10] Aceasta se întâmplă pentru că, spre deosebire de coeficienții logit și odds ratio (OR), în cazul probabilităților relația dintre independentă și probabilitate este non-liniară și non-aditivă, deci nu o putem exprima folosind o singură valoare. Dacă efectul lui X (independentei) asupra lui Y (dependentei) în termeni de OR este constant pentru orice valori ale lui X, efectul în termeni de probabilități variază în funcție de valorile lui X. Din acest punct de vedere, OR prezintă un avantaj comparativ cu probabilitățile.

mai mare în cazul angajaților care lucrează peste program, respectiv scade pe măsură ce vechimea în companie crește. Mai mult, observăm că probabilitatea de a pleca nu se modifică liniar în funcție de valorile variabilelor independente: pe unele porțiuni ale intervalului de variație asociat atributului „vechime" variația probabilității este mai accentuată, pe altele mai puțin. În plus, diferența dintre probabilitatea de plecare a angajaților care lucrează peste program și probabilitatea celor care nu lucrează peste program scade pe măsură ce crește vechimea, ajungând în final să fie nulă (există un efect de interacțiune între cei doi predictori). Simplu spus, în cazul angajaților cu vechime mică în companie, munca peste program are un impact negativ major asupra rămânerii în companie. Pe măsură ce vechimea crește, impactul muncii peste program asupra probabilității de a pleca scade, ajungând să fie nul în cazul celor cu vechime de cel puțin 25 ani (pe măsură ce vechimea crește, probabilitatea de a pleca depinde tot mai puțin de munca peste program).

Figura 5.1-1. Probabilitatea prezisă de a pleca din companie în funcție de vechime și munca peste program (model cu efecte de interacțiune)



Există **două tipuri de probabilități** ce pot fi calculate și utilizate pentru a exprima relația dintre predictori și dependentă: **predicții ajustate (Adjusted Predictions)** și **efecte marginale (Marginal Effects)**. Pentru fiecare dintre aceste două tipuri, funcție de interes, pot fi calculate trei măsuri: „Adjusted Predictions at the Means" (APM), „Adjusted Predictions at Representative values" (APR), „Average Adjusted Predictions" (AAP), „Marginal Effects at the Means" (MEM), „Marginal Effects at

Representative values" (MER), „Average Marginal Effects" (AME) (Williams, 2012). Dintre toate acestea, de preferat sunt cele relativ la valori reprezentative (vezi Figura 5.1-1 pentru o ilustrare a APR; graficul MER ar arăta similar, dar ar afișa o singură curbă – diferența dintre cele două curbe din graficul respectiv). Dacă dorim să exprimăm relația (efectul) printr-o singură valoare, AAP și AME reprezintă variantele de preferat (Long & Freese, 2014).

Pentru cei care doresc să înțeleagă relativ mai în profunzime modul în care este estimat un model de regresie logistică, prezentăm două exemple simple, cu câte un singur predictor, prima dată binar (Tabelul 5.2-1), apoi metric (Tabelul 5.2-2). În ambele exemple, variabila prezisă (dependentă) este „dacă angajatul pleacă din companie" (1=da / 0=nu). Înainte de aceste exemple, prezentăm o scurtă discuție cu privire la trei teme importante:

(1) Care sunt asumpțiile regresiei logistice?
(2) Ce problemele care pot să apară atunci când estimăm un model de regresie logistică?
(3) Cum învață un model de regresie logistică relații de tip liniar vs. nonliniar?

Asumpțiile regresiei logistice

Ca orice metodă de analiză statistică, regresia logistică are la bază anumite asumpții. Dacă aceste asumpții nu sunt respectate, coeficienții obținuți sunt distorsionați (biased) sau au asociate erori standard foarte mari, prin urmare estimările probabilităților de succes și predicțiile realizate pe baza modelului nu sunt corecte, respectiv pot fi imprecise. În cazul regresiei logistice asumpțiile sunt următoarele:

- relația dintre probabilitățile condiționate reale și predictori / variabilele independente ia forma unei funcții de tip logistic (în formă de S, simetrică);[11]
- modelul nu omite variabile importante;
- modelul nu include variabile irelevante;
- variabilele independente sunt măsurate fără eroare;
- cazurile / observațiile sunt independente;
- niciuna dintre variabilele independente (predictori) nu este o combinație liniară a celorlalte variabile independente (variabilele respective nu sunt multicoliniare).

[11] În cazul regresiei logistice se asumă că erorile de predicție au o distribuție de tip logistic. O distribuție logistică standard are media 0 și varianța $\pi^2/3$ (aproximativ 3.29).

Posibile probleme, sursele acestora și cum le putem rezolva

Funcție de datele utilizate și de complexitatea modelului, uneori algoritmul de regresie logistică nu găsește o soluție, adică nu estimează valorile coeficienților (nu converge) sau, dacă totuși converge, erorile standard asociate coeficienților sunt extrem de mari (soluția obținută este extrem de instabilă, deci are șanse reduse de generalizare). Principalele două motive pentru care se întâmplă acest lucru sunt informația insuficientă (combinații rare, adică fără cazuri sau cu cazuri foarte puține) și separarea completă (predicția perfectă).

Informația insuficientă: se referă la situația în care setul de date nu include niciun caz[12] în una dintre clasele variabilei dependente pentru una sau mai multe dintre combinațiile de predictori; în exemplul de mai jos (Tabelul 5.1-3) nu avem cazuri de tip „Da" (Pleacă) pentru combinația „are copii + în ultimii 2 ani nu i-a crescut salariul".

Tabelul 5.1-3. Problemă: informație insuficientă

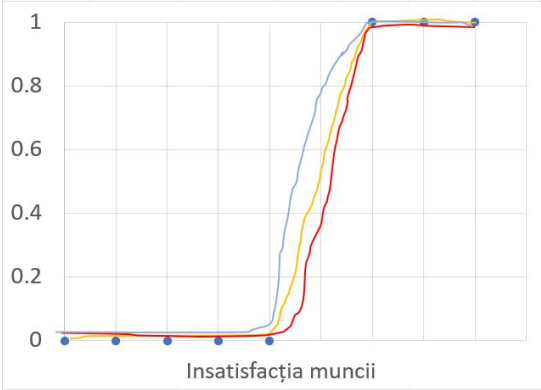
Are copii	Mărire salariu (<2 ani)	Părăsește compania
Nu	Nu	Da
Nu	Da	Da
Da	Nu	?
Da	Da	Da

Separarea completă: se referă la situația în care variabila dependentă este prezisă perfect de una dintre variabilele independente sau de o combinație de variabile independente (Tabelul 5.1-4); altfel spus, (cvasi-)separarea sau predicția perfectă se referă la situația în care variabila dependentă nu variază în cazul anumitor categorii ale predictorilor; în exemplul de mai jos, dacă X (insatisfacția muncii) ≤ 4, Y (pleacă) va fi întotdeauna 0 (nu pleacă), iar dacă X > 4, Y = 1; în aceste condiții, curba de regresie logistică poate avea o diversitate de forme (pentru exemplul nostru, am ilustrat trei curbe posibile), deci algoritmul nu poate estima o soluție unică a ecuației (valori unice ale coeficienților).

[12] Sau dacă aproximativ mai mult de 20% dintre combinații (celule) conțin cel mult 5 cazuri.

Tabelul 5.1-4. Problemă: separare completă

X Insat	Y Pleacă
0	0
1	0
2	0
3	0
4	0
6	1
7	1
8	1



Soluția la problema combinațiilor rare este să includem în model doar predictori care au suficiente cazuri în fiecare categorie. Dacă avem atribute care au foarte puține cazuri în unele clase, fie le scoatem din analiză, fie le includem / combinăm într-o altă categorie (de exemplu, dacă în cazul atributului „educație formală” avem foarte puține cazuri care formează clasele „școala generală” și „10 clase”, putem uni aceste categorii în categoria „maximum 10 clase”.

În cazul separării complete, presupunând că nu sunt erori de introducere sau recodificare a datelor, soluția cea mai simplă constă de asemenea în unirea categoriilor slab reprezentate (simplificarea modelului). Alternativ, putem calcula modele de regresie logistică speciale (exact, firth, bayesian). Chiar și atunci când setul de date are foarte multe cazuri, dacă numărul cazurilor din clasa de interes este mic (unbalanced dataset), modelul de regresie logistică este dificil de estimat (funcție și de numărul și tipul predictorilor, respectiv a relațiilor dintre aceștia).

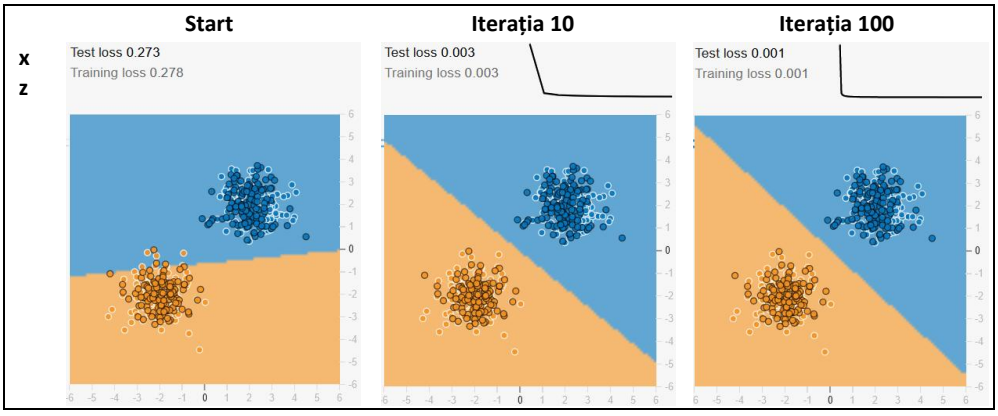
Învățarea unor relații liniare vs. nonliniare

Forma pe care o poate lua frontiera estimată depinde atât de algoritmul utilizat cât și de includerea sau nu în modelul de clasificare a unor valori transformate ale atributelor originale. În cazul regresiei logistice, dacă includem ca predictori doar atributele inițiale, frontiera dintre clase va fi întotdeauna liniară (un punct, o dreaptă,

un plan etc., funcție de numărul de predictori[13]), deci vom putea modela doar relații simple, de tip liniar. Dacă includem ca predictori și transformări ale atributelor (de exemplu, pe lângă atributul X putem include puteri ale acestuia precum x^2, x^3, $x^{0.5}$, $x^{0.33}$, respectiv alte funcții sau combinații de funcții (de exemplu ln(x), cos(x), sin(x) etc.); de asemenea putem include produsul a două atribute, raportul dintre două atribute etc.), vom putea modela și relații mai complexe. Evident, în astfel de situații, granița dintre clase va fi una relativ mai complexă, non-liniară. Desigur, creșterea complexității modelului merită, adică aduce un plus la calitatea clasificării, doar dacă datele (relațiile dintre atribute) sunt complexe.

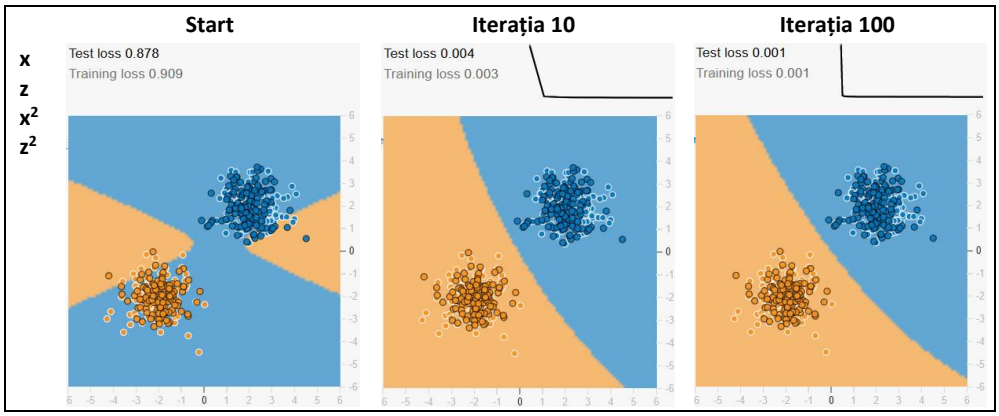
Pentru a ilustra aceste idei, vom considera două exemple simple. Dacă frontiera dintre clase este clară și liniară (Figura 5.1-2), un model de regresie logistică cu doar doi predictori poate prezice aproape perfect apartenența cazurilor la clase după doar 10 iterații.[14] Includerea ca predictori a atributelor originale transformate (pătratul lor) nu crește calitatea predicției. În cazul în care frontiera dintre clase este nonliniară (relații complexe), lucrurile stau cu totul altfel (Figura 5.1-3). Astfel, în acest caz modelul de regresie logistică care include doar predictorii originali nu poate învăța frontiera dintre clase nici măcar după 100 iterații (clasifică incorect 52% din cazuri). Dacă includem ca predictori și pătratul atributelor originale, modelul învață relațiile din setul de date foarte repede: după 10 cicluri sunt clasificate corect 94% din cazuri, iar după 100 cicluri 99.6%.

Figura 5.1-2. Învățarea unor relații liniare: atribute originale vs. transformate



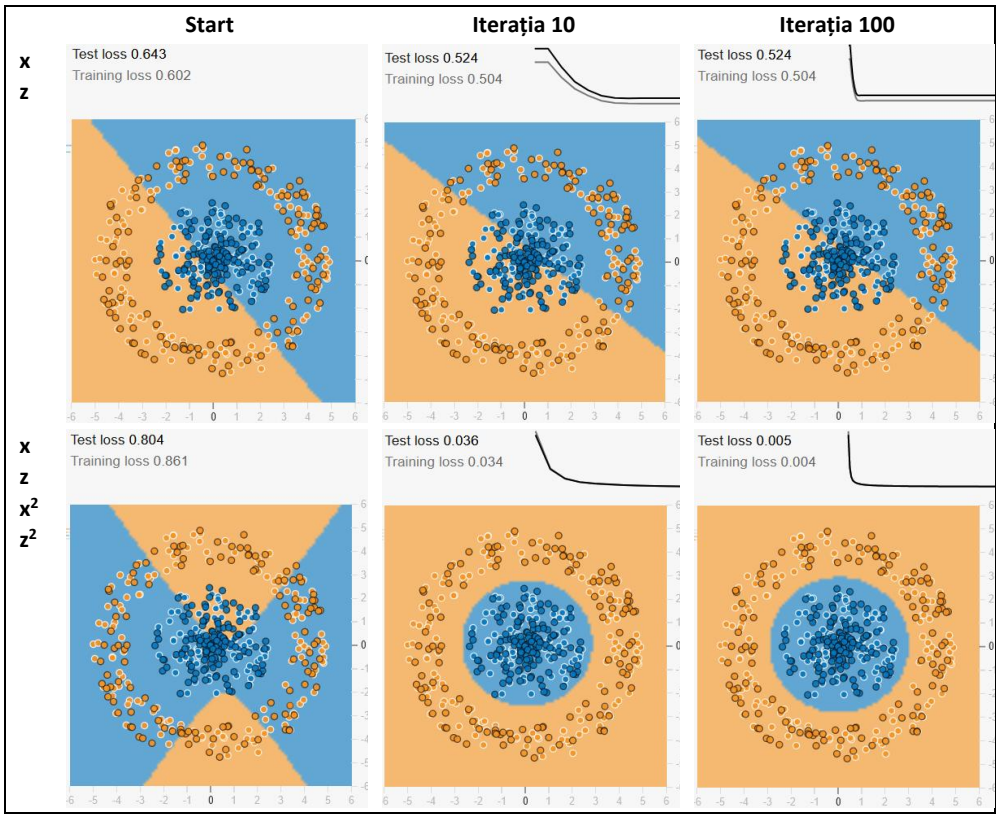
[13] Dacă atributul de interes are trei clase, vom avea nevoie de două puncte / linii / planuri / etc. pentru a defini granița. Dacă are patru clase vor fi necesare trei puncte / linii / planuri / etc.

[14] Pentru aceste exemple am estimat modelele de regresie logistică ca modele de rețele neuronale fără neuroni ascunși, de aici și parametrul număr iterații / cicluri de învățare.



Sursa: https://playground.tensorflow.org/; predictori: x și z (prima linie); x, z, x^2 și z^2 (ultima linie)

Figura 5.1-3. Învățarea unor relații nonliniare: atribute originale vs. transformate



Sursa: https://playground.tensorflow.org/; predictori: x și z (prima linie); x, z, x^2 și z^2 (ultima linie)

5.2. Regresia logistică binară cu un singur predictor: două exemple didactice

În această secțiune prezentăm două exemple didactice de regresie logistică binară simplă, unul cu un predictor binar, altul cu un predictor metric.

Predictor binar

Pentru acest exemplu considerăm predictorul „angajatul are copii" cu două variante de răspuns: 1=da și 0=nu. Observăm că setul de date considerat (Tabelul 5.2-1) este identic cu cel discutat la începutul capitolului (Tabelul 5.1-1). În total avem 10 cazuri: 3 angajați fără copii care pleacă din companie, 2 angajați fără copii care rămân, 1 angajat cu copii care pleacă și 4 angajați cu copii care rămân. Ne interesează să estimăm care este raportul de șanse relativ la categoria pleacă în funcție de existența copiilor (clasa de referință este 1, adică are copii). Anterior (Tabelul 5.1-1), am estimat că șansele de plecare în cazul angajaților cu copii sunt de 6 ori mai mici comparativ cu șansele angajaților fără copii, deci OR este 0.167 (1:6). Valoarea logit corespunzătoare lui OR = 0.167 este -1.792 (ln(0.167) = -1.792), aceasta fiind și valoarea estimată de algoritmul de regresie logistică (Tabelul 5.2-1). Pentru a ajunge la această estimare, algoritmul alege[15] niște valori de start ale coeficienților a (constanta) și b (logit) ai ecuației de regresie, calculează discrepanța dintre date și predicții (măsura LL), modifică valorile coeficienților astfel încât discrepanța să scadă (adică LL să crească), și tot așa, iterativ,[16] până când valorile coeficienților se stabilizează, iar suma valorilor LL este cât mai mare posibil (valoarea maximă posibilă este 0 și indică o predicție perfectă).

[15] De obicei, programele de analiză statistică aleg ca valori de start estimările coeficienților obținute în cazul regresiei liniare (Pampel, 2020, p. 56).

[16] Dacă, după o iterație, suma valorilor LL nu mai crește cu o cantitate superioară pragului indicat/ setat automat, algoritmul se oprește și reține soluția cea mai bună obținută. Pentru o explicație mai detaliată a modului de calcul a LL și câteva exemple se poate consulta capitolul 3 al volumului „Logistic Regression: A Primer" (Pampel, 2020).

Tabelul 5.2-1. Un exemplu de estimare a unui model de regresia logistică simplă: predictor binar

Id	Copii	Pleacă	Logit	e^{logit}	P(1)	LL
1	0	1	0.100	1.105	0.525	-0.644
2	0	1	0.100	1.105	0.525	-0.644
3	0	1	0.100	1.105	0.525	-0.644
4	0	0	0.100	1.105	0.525	-0.744
5	0	0	0.100	1.105	0.525	-0.744
6	1	1	0.200	1.221	0.550	-0.598
7	1	0	0.200	1.221	0.550	-0.798
8	1	0	0.200	1.221	0.550	-0.798
9	1	0	0.200	1.221	0.550	-0.798
10	1	0	0.200	1.221	0.550	-0.798
					Suma	**-7.213**

Logit	e^{logit}	P(1)	LL
0.405	1.500	0.600	-0.511
0.405	1.500	0.600	-0.511
0.405	1.500	0.600	-0.511
0.405	1.500	0.600	-0.916
0.405	1.500	0.600	-0.916
-1.386	0.250	0.200	-1.609
-1.386	0.250	0.200	-0.223
-1.386	0.250	0.200	-0.223
-1.386	0.250	0.200	-0.223
-1.386	0.250	0.200	-0.223
		Suma	**-5.867**

Coeficienții logit la start:
a = 0.1
b = 0.1

Coeficienții logit finali:
a = 0.405
b = -1.792

** 0 = Nu, 1 = Da;*
*** Logit = a + b x Copii*
elogit = exp(Logit)
P(1) = Probabilitate(Da) = elogit / (1 + elogit);
dacă Pleacă = 1, atunci LL = ln(P(1))
dacă Pleacă = 0, atunci LL = ln(1 - P(1))
LL = Log Likelihood; Suma = ∑ LL;
**** Tabelul din stânga conține valorile calculate folosind coeficienții de start, cel din dreapta valorile finale, estimate în baza coeficienților finali.*

Predictor metric

Similar cu exemplul anterior, putem estima un model de regresie logistică binară simplă și în cazul în care predictorul este de tip metric (Tabelul 5.2-2). Pentru acest exemplu am considerat predictorul „Insatisfacția relativ la locul de muncă". Acesta ia valori în intervalul 0-10 (scală ordinală, tratată aici ca metrică), unde 10 reprezintă insatisfacția maximă. Firesc, ne așteptăm ca un nivel ridicat de insatisfacție să crească șansele de părăsire a companiei (coeficientul logit asociat predictorului să fie pozitiv, iar odds ratio > 1). Valorile obținute în urma rulării algoritmului de regresie logistică susțin pe deplin așteptările: coeficientul logit (b) este 0.804, iar OR asociat este 2.235 (exp(0.804) = 2.235), ceea ce se traduce astfel: atunci când insatisfacția crește cu o unitate, șansele (odds = e^{logit}) aproximativ se dublează (mai exact cresc cu 123%) indiferent de nivelul de referință al insatisfacției (relația este

liniară). Nu același lucru este valabil și în cazul probabilității de plecare din companie (P(1)). Aceasta variază non-liniar, adică în funcție de poziția relativ la care are loc creșterea. Mai specific, creșterea nivelului de insatisfacție cu o unitate în zona valorilor extreme mărește probabilitatea de plecare cu un factor mai mic comparativ cu creșterea cu o unitate în zona valorilor de mijloc. De exemplu, o creștere a insatisfacției de la 1 la 2 (sau de la 9 la 10), produce o creștere a probabilității de a pleca mai mică comparativ cu o creștere a insatisfacției de la 5 la 6 (deși în ambele cazuri e vorba de o creștere cu un punct) (Figura 5.2-1).

Tabelul 5.2-2. Un exemplu de estimare a unui model de regresia logistică simplă: predictor metric

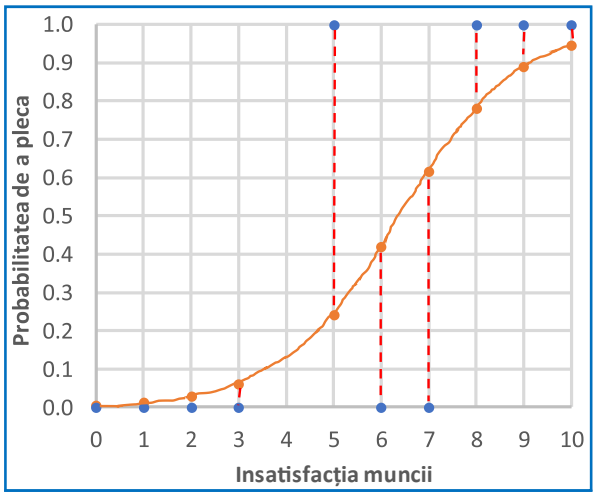
Id	Insat	Pleacă	Logit	e^{logit}	P(1)	LL	Logit	e^{logit}	P(1)	LL
1	10	1	1.100	3.004	0.750	-0.287	2.883	17.871	0.947	-0.054
2	9	1	1.000	2.718	0.731	-0.313	2.079	7.997	0.889	-0.118
3	8	1	0.900	2.460	0.711	-0.341	1.275	3.579	0.782	-0.246
4	7	0	0.800	2.226	0.690	-1.171	0.471	1.601	0.616	-0.956
5	6	0	0.700	2.014	0.668	-1.103	-0.333	0.717	0.417	-0.540
6	5	1	0.600	1.822	0.646	-0.437	-1.137	0.321	0.243	-1.415
7	3	0	0.400	1.492	0.599	-0.913	-2.746	0.064	0.060	-0.062
8	2	0	0.300	1.350	0.574	-0.854	-3.550	0.029	0.028	-0.028
9	1	0	0.200	1.221	0.550	-0.798	-4.354	0.013	0.013	-0.013
10	0	0	0.100	1.105	0.525	-0.744	-5.158	0.006	0.006	-0.006
					Suma	**-6.963**			**Suma**	**-3.440**

Coeficienții logit la start:
a = 0.1
b = 0.1

Coeficienții logit finali:
a = -5.158
b = 0.804

** 0 = Nu, 1 = Da; Insat = Insatisfacția relativ la locul de muncă;*
*** Logit = a + b x Insat* — *dacă Pleacă = 1, atunci LL = ln(P(1))*
elogit = exp(Logit) — *dacă Pleacă = 0, atunci LL = ln(1 - P(1))*
P(1) = Probabilitate(Da) = elogit / (1 + elogit); — *LL = Log Likelihood; Suma = ∑ LL;*
**** Tabelul din stânga conține valorile calculate folosind coeficienții de start, cel din dreapta valorile finale, estimate în baza coeficienților finali.*

Figura 5.2-1. Un exemplu de estimare a unui model de regresia logistică simplă: predictor metric



* ● = *valorile reale asociate cazurilor;*
● = *valorile prezise asociate cazurilor;*
funcția de regresie logistică estimată
erorile de predicție (relativ la probabilitate)

5.3. Exemple de utilizare a regresiei logistice în RapidMiner

În această secțiune vom prezenta două exemple de analiză în care am folosit algoritmul de regresie logistică clasică din RapidMiner (Modeling > Predictive > Logistic Regression).[17] Varianta clasică este o versiunea simplificată[18] a operatorului

[17] Pentru alte exemple se pot consulta diferite texte de specialitate (Kotu & Deshpande, 2019, Chapter 5; Shmueli et al., 2023, Chapter 10).

[18] Parametrul Family este setat implicit ca binominal, iar parametrul Link ca logit. Alți parametri importanți pot fi setați de utilizator. Dacă dorim să construim un model mai flexibil, trebuie să folosim operatorul GLM (varianta din RapidMiner este cea implementată de compania H2O.ai). Operatorul Logistic Regression implementează, similar cu operatorul GLM, o combinație a penalizărilor de tip L1 și L2, indicate prin parametrii λ și α. Când parametrul λ = 0, nu se folosește nicio regularizare, α este ignorat, deci operatorul estimează un model de regresie logistică obișnuită) (Shmueli et al., 2023, p. 283). Regularizarea este importantă deoarece reduce complexitatea modelului (alte metode, backward elimination și forward stepwise, urmăresc același lucru, dar au două limite majore: atunci când numărul predictorilor este mare, resursele computaționale cresc exponențial; modelul preferat poate include predictori irelevanți). Implicit, operatorul din RapidMiner nu folosește regularizarea. Dacă dorim să o folosim, trebuie să marcăm parametrul „use regularization". Regularizarea urmărește reducerea supra-ajustării la date a modelului (overfitting) cu scopul de a crește șansele de generalizare ale acestuia. Pentru aceasta, regularizarea reduce complexitatea modelului prin impunerea unor penalități. Prin urmare, modelul final nu va mai include practic

GLM (Generalized Linear Model). Pe lângă varianta clasică, în RapidMiner sunt implementate alte două variante ale algoritmului de regresie logistică (SVM și Evolutionary; pentru ultimul am inclus un exemplu în folderul asociat volumului). Dacă instalăm extensia Weka, putem folosi și alte variante (W-Logistic, W-SimpleLogistic, W-LogisticBase, W-BayesianLogisticRegression).

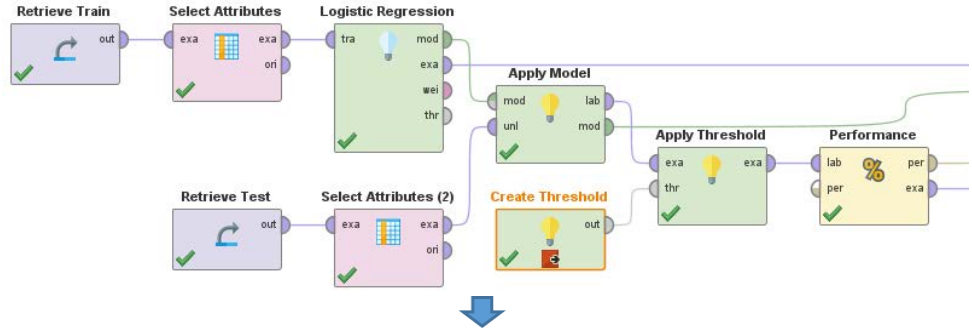
Un model simplu

Pentru acest exemplu vom folosi doar trei atribute din setul de date „employee_attrition". Vom realiza un model de clasificare bazat pe regresia logistică cu scopul de a prezice situațiile de părăsire a companiei în funcție de vechimea în muncă și munca peste program. În acord cu literatura, ne așteptăm ca o vechime mai mare să reducă șansele de plecare, iar munca peste program să le crească. Pașii realizării acestui model în RapidMiner, rezultatele obținute și interpretarea lor apar în Figura 5.3-1.

Figura 5.3-1. Regresia logistică: un model simplu în RapidMiner

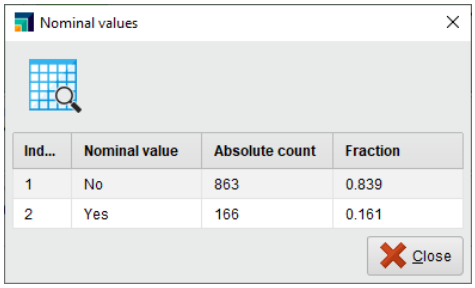
Pasul 1:
Realizăm procesul din imaginea de mai jos. În folderul cu date asociat volumului am inclus două versiuni ale acestui proces: o versiune mai simplă, fără setarea pragului de probabilitate (acesta este stabilit implicit la 0.5) și o versiune care oferă posibilitatea setării pragului de probabilitate. Setul de date utilizat pentru instruirea modelului este „employee_attrition_training_validation". Setul utilizat pentru testarea modelului este „employee_attrition_testing". Relativ la ambele seturi de date păstrăm doar trei atribute: Attrition, OverTime, YearsAtCompany (operatorul „Select Attributes"). Includem operatorul Logistic Regression cu setările originale. Acestea iau în calcul doar parametrii „standardize" (calculează și varianta standardizată a coeficienților logit), „add intercept" (adaugă o constantă) și „remove colliniar columns" (elimină atributele care corelează perfect între ele).
În cazul modelului 2 includem operatorii care ne ajută să setăm pragul de probabilitate dorit (Create Threshold și Apply Threshold). Includem operatorul necesar pentru aplicarea modelului pe setul de date de testare, respectiv operatorul care calculează măsurile de performanță. Realizăm conexiunile și rulăm procesul. La modelul 1, pragul de probabilitate este setat automat la 0.5 (prima clasă este No, a doua Yes). În acest caz modelul va prezice că angajații în cazul cărora valoarea atributului „confidence(Yes)" va fi mai mare de 0.5 vor pleca, iar restul vor rămâne. La modelul 2, pragul este stabilit manual la 0.3.

o parte dintre atributele incluse inițial (coeficienții asociați acestor atribute vor fi reduși spre zero). Important, dacă alegem să folosim regularizarea, e important să standardizăm atributele (prin selectarea parametrului standardize), altfel penalizarea va fi disproporționat mai mare în cazul atributelor cu scale relativ mai mari (Jacobucci et al., 2023, p. 83).



Rezultate (setul de instruire):
Seturile de date conțin doar atributele selectate anterior (în imagine apar primele 5 cazuri din setul de instruire). Distribuția statistică a atributului de interes – Attrition – este 0.161 pentru clasa Yes (pleacă) și 0.839 pentru clasa No (nu pleacă).

Row No.	Id	Attrition	OverTime	YearsAtCompany
1	4	Yes	Yes	0
2	5	No	Yes	8
3	7	No	No	2
4	10	No	Yes	1
5	14	No	No	5



Rezultate (setul de testare):
Relativ la setul de testare, modelul produce predicțiile din imagine (am inclus primele 5 cazuri și alte trei de interes în contextul acestui exemplu). În imagine am prezentat doar predicțiile relativ la pragul 0.5. Coloana Attrition prezintă situația reală, iar coloana „prediction(Attrition)" predicția realizată de model. Observăm că o parte dintre predicții sunt greșite. Relativ la pragul 0.5, cazurile prezise greșit sunt cele cu Id 1, 90, 165 și 167. Relativ la pragul 0.3, cazurile care au la atributul confidence(Yes) o valoare prezisă cel puțin egală cu 0.3 vor fi prezise ca aparținând de clasa Yes (atributul prediction(Attrition)). Probabilitățile asociate predicțiilor apar pe coloanele „confidence". Relativ la fiecare caz, clasa care are asociată cea mai mare probabilitate (confidence) prezisă, determină clasa la care modelul prezice că aparține acel caz. Deoarece toate probabilitățile asociate coloanei „confidence(Yes)" sunt mai mici de 0.5 (50%), niciuna dintre predicții nu este Yes (angajatul pleacă), deci modelul prezice că toate cazurile aparțin clasei No (angajatul nu pleacă). Însă, aceasta nu înseamnă că toți angajații au asociate aceleași probabilități de a pleca. Cea mai mică probabilitate estimată de a pleca, 0.006 adică 0.6%, apare în cazul angajatului 165. Acesta are o vechime foarte mare și nu muncește peste program (OverTime=No și YearsAtCompany=40). Cea mai mare probabilitate estimată de a pleca, 0.424 adică 42.4%, apare în cazul angajatului 167. Acesta are o vechime foarte mică și muncește peste program (OverTime=Yes și YearsAtCompany=0). Dat fiind faptul că pragul de probabilitate ales este 0.5, modelul prezice că acest angajat nu va pleca. Pentru exemplul cu pragul setat la 0.3, modelul prezice că un angajat cu aceste caracteristici pleacă.

Row No.	Id	Attrition	prediction(Attrition)	confidence(No)	confidence(Yes)	OverTime	YearsAtCompany
1	1	Yes	No	0.695	0.305	Yes	6
2	2	No	No	0.926	0.074	No	10
3	8	No	No	0.907	0.093	No	7
4	11	No	No	0.853	0.147	No	1
5	12	No	No	0.920	0.080	No	9
...							
24	90	Yes	No	0.597	0.403	Yes	1
...							
40	165	Yes	No	0.994	0.006	No	40
...							
41	167	Yes	No	0.576	0.424	Yes	0

...

Modelul de predicție estimat pe setul de date de instruire apare în secțiunea de rezultate „Logistic Regression Model". Tabul Data conține coeficienții logit nestandardizați și standardizați, eroarea standard și semnificația statistică asociate acestora. Atenție, clasa de referință pentru atributul OverTime (munca peste program) este No. În cazul atributelor metrice (OverTime) standardizarea e relativ la X (predictor). În cazul atributelor binare (OverTime) standardizarea nu schimbă valoarea coeficientului (coeficientul standardizat este egal cu cel nestandardizat). Observăm că ambii coeficienți sunt negativi și semnificativi statistic diferiți de zero. Pragul de semnificație asociat coeficienților este cel obișnuit, 95%. Respectarea acestui prag este indicată de faptul că valoarea absolută a scorurilor z depășește 2, respectiv de valorile p mai mici sau egale cu 0.05. Prin urmare, putem fi aproape siguri că vom regăsi aceste relații și pe alte seturi de date (eșantioane) produse în aceleași condiții. Aceste rezultate pot fi interpretate astfel: șansele de a părăsi compania sunt mai mari în cazul angajaților care muncesc peste program, respectiv scad pe măsură ce crește vechimea în muncă la acea companie.

Tabelul nu prezintă și coeficienții odds ratio (or = raport de șanse), dar îi putem calcula simplu. De exemplu, în cazul atributului binar OverTime.No, coeficientul logit (nestandardizat) este -1.365, deci coeficientul or asociat este 0.255 ($e^{-1.365}$ = 0.255). Prin urmare, putem afirma că șansele ca un angajat care nu muncește peste program să părăsească firma sunt de aproximativ patru ori mai mici (0.255:1) comparativ cu șansele unui angajat care muncește peste program (celelalte caracteristici fiind aceleași). Alternativ, putem spune că șansele ca un angajat care muncește peste program să părăsească firma sunt de aproximativ patru ori mai mari comparativ cu șansele unui angajat care nu muncește peste program ($e^{1.365}$ = 3.916). În cazul atributului de tip numeric YearsAtCompany, coeficientul logit (nestandardizat) este -0.086, prin urmare un an în plus ca angajat al companiei reduce șansele de a pleca cu un factor egal cu $e^{-0.086}$, deci cu 0.918. Alternativ, o vechime mai mică cu un an crește șansele de plecare cu un factor egal cu 1.090 ($e^{0.086}$ = 1.090), adică 9%.

De asemenea, putem calcula valorile logit, respectiv probabilitatea de a pleca din companie pentru orice combinație de valori asociate predictorilor. De exemplu ...

- un angajat care NU muncește peste program și are o vechime de zero ani are asociată o valoare logit prezisă de a pleca egală cu -1.671 (-0.086*0 - 1.365*1 - 0.306 = -1.671), deci probabilitatea ca acesta să plece este 0.158 ($e^{-1.671}$ / (1+ $e^{-1.671}$)) = 0.158);
- un angajat care muncește peste program și are o vechime de zero ani are asociată o valoare logit prezisă de a pleca egală cu -0.306 (-0.086*0 - 1.365*0 - 0.306 = -0.306), deci probabilitatea ca acesta să plece este 0.424 ($e^{-0.306}$ / (1+ $e^{-0.306}$)) = 0.424);

Dacă ne uităm la setul de date cu probabilitățile estimate de RapidMiner, coloana Confidence(Yes), regăsim valorile calculate mai sus (vezi de exemplu cazurile cu Id 364 și 167). Similar, putem calcula

și coeficienții logit, respectiv probabilitățile de plecare atunci când predictorul de interes este metric:

- un angajat care muncește peste program și are o vechime de **zero** ani are asociată o valoare logit prezisă de a pleca egală cu -0.306 (-0.086*0 - 1.365*0 - 0.306 = -0.306), deci probabilitatea ca acesta să plece este 0.424 ($e^{-0.306}$ / (1+ $e^{-0.306}$)) = 0.424);
- un angajat care muncește peste program și are o vechime de **zece** ani are asociată o valoare logit prezisă de a pleca egală cu -1.166 (-0.086*10 - 1.365*0 - 0.306 = -1.166), deci probabilitatea ca acesta să plece este 0.238 ($e^{-1.166}$ / (1+ $e^{-1.166}$)) = 0.238);
- un angajat care muncește peste program și are o vechime de **douăzeci** ani are asociată o valoare logit prezisă de a pleca egală cu -2.026 (-0.086*20 - 1.365*0 - 0.306 = -2.026), deci probabilitatea ca acesta să plece este 0.117 ($e^{-2.026}$ / (1+ $e^{-2.026}$)) = 0.117);

La tabul Description apar o serie de informații relativ la model, măsurile de calitate și matricea de confuzie fiind cele mai importante. Valorile măsurilor indică faptul că avem mai degrabă un model modest: varianța explicată este relativ redusă (R^2=0.095, deci 10%), media erorilor de clasificare este mare (31%), iar AUC este 0.72 (aproximativ la mijloc între 0.5 și 1).

Attribute ↓	Coefficient	Std. Coefficient	Std. Error	z-Value	p-Value
YearsAtCompany	-0.086	-0.532	0.019	-4.439	0.000
OverTime.No	-1.365	-1.365	0.178	-7.686	0.000
Intercept	-0.306	-0.907	0.166	-1.840	0.066

Data

Description

Annotations

Logistic Regression Model

```
Model Metrics Type: BinomialGLM
 Description: N/A
 model id: rm-h2o-model-logistic_regression-12
 frame id: rm-h2o-frame-logistic_regression-12
 MSE: 0.12241583
 RMSE: 0.34987974
 R^2: 0.09520651
 AUC: 0.72244483
 pr_auc: 0.37135756
 logloss: 0.40125626
 mean_per_class_error: 0.30807704
 default threshold: 0.1471324861049652
 CM: Confusion Matrix (Row labels: Actual class; Column labels: Pre
        No  Yes   Error         Rate
    No  586  277  0.3210   277 / 863
   Yes   49  117  0.2952    49 / 166
Totals  635  394  0.3168  326 / 1,029
Gains/Lift Table (Avg response rate: 16.13 %, avg score: 16.23 %):
```

Secțiunea PerformanceVector conține matricea de confuzie și diferite măsuri ale performanței clasificării (cele selectate) calculate relativ la setul de date de testare. Am observat anterior că pragul de probabilitate asociat clasei Yes (pleacă) stabilit la 50% nu a fost depășit în niciun caz. Ca urmare, modelul a prezis că toți angajații rămân. Prin urmare, modelul prezice corect toate situațiile în care angajații rămân, dar niciuna în care pleacă. Acuratețea predicției este mare (84%) doar pentru că majoritatea angajaților nu pleacă. Relativ la clasa de interes pentru noi, cei care pleacă, calitatea predicției e nulă: măsurile de performanță precizie și reamintire au valoarea 0.

Desigur, putem să reducem pragul probabilității și să refacem calculele. De exemplu, dacă stabilim un prag de 30%, toate cazurile cu o probabilitate asociată clasei Yes de cel puțin 30% vor fi alocate clasei prezise Yes (pleacă). Observăm că, în acest caz, acuratețea predicției scade ușor (de la 84% la 81%), dar măsurile precizie și reamintire asociate clasei Yes (pleacă) cresc foarte mult: precizia crește de la 0% la 41%, iar reamintirea de la 0% la 38%. Altfel spus, dacă anterior niciun caz nu era prezis ca Yes, acum modelul prezice că 66 angajați vor pleca. Dintre aceștia, pleacă doar 27, deci modelul prezice greșit situația celorlalți 39. Raportat la angajații care în realitate chiar pleacă, modelul identifică doar o parte dintre ei (27), pe restul de 44 alocându-i greșit la clasa No (nu

pleacă). Dacă dorim să identificăm o pondere și mai mare a angajaților care pleacă, putem reduce pragul (Threshold) și mai mult. Desigur, în acest caz ponderea erorilor de tip I va crește.

accuracy: 83.90%

	true No	true Yes	class precision
pred. No	370	71	83.90%
pred. Yes	0	0	0.00%
class recall	100.00%	0.00%	

set testare, prag 50%

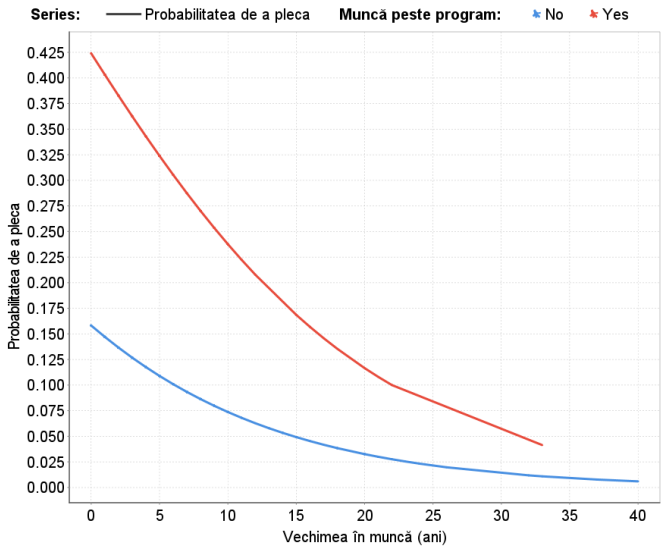
accuracy: 81.18%

	true No	true Yes	class precision
pred. No	331	44	88.27%
pred. Yes	39	27	40.91%
class recall	89.46%	38.03%	

set testare, prag 30%

Dacă dorim, putem folosi predicțiile rezultate pentru a reprezenta grafic probabilitatea de a pleca din companie în funcție de vechimea în acea companie și munca peste program (Figura 5.3-2). Dat fiind faptul că pentru acest exemplu nu am inclus și interacțiunea dintre cei doi predictori, graficul rezultat arată puțin diferit comparativ cu cel prezentat anterior (Figura 5.1-1). O altă sursă a diferenței constă în faptul că în acest grafic, produs în RapidMiner, probabilitățile prezise sunt calculate ca medii ale probabilităților asociate cazurilor din setul de date (deci, dacă setul de date nu conține cazuri pentru anumite combinații, probabilitățile nu apar în grafic).

Figura 5.3-2. Probabilitatea prezisă de a pleca din companie în funcție de vechime și munca peste program (model fără efecte de interacțiune)

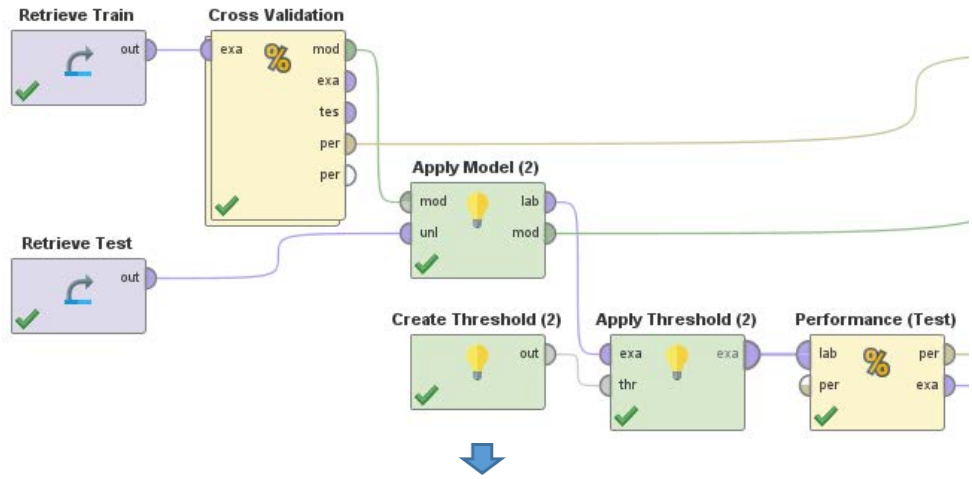


Un model relativ mai complex

Pentru acest exemplu vom folosi toate atributele din setul de date „employee_attrition”. Vom realiza un model de clasificare bazat pe regresia logistică cu scopul de a prezice situațiile de părăsire a companiei în funcție de toate atributele incluse în setul de date. Pașii realizării acestui model în RapidMiner, rezultatele obținute și interpretarea lor apar în Figura 5.3-3.

Figura 5.3-3. Regresia logistică: un model relativ mai complex în RapidMiner

Pasul 1:
Realizăm procesul din imaginea de mai jos. În folderul asociat volumului am inclus două versiuni ale acestui proces: o versiune mai simplă, fără setarea pragului de probabilitate (acesta este stabilit implicit la 0.5) și o versiune care oferă posibilitatea setării pragului de probabilitate. Setul de date utilizat pentru instruirea modelului este „employee_attrition_training_validation”. Setul utilizat pentru testarea modelului este „employee_attrition_testing”. Includem operatorul Logistic Regression cu setările originale. În cazul modelului 2 includem operatorii care ne ajută să setăm pragul de probabilitate dorit (Create Threshold și Apply Threshold). Includem operatorul necesar pentru aplicarea modelului pe setul de date de testare, respectiv operatorul care calculează măsurile de performanță. Realizăm conexiunile și rulăm procesul. La modelul 1, pragul de probabilitate este setat implicit la 0.5 (prima clasă este No, a doua Yes). Prin urmare, modelul va prezice că angajații în cazul cărora valoarea atributului „confidence(Yes)” va fi mai mare de 0.5 vor pleca, iar restul vor rămâne. La modelul 2, pragul este stabilit manual la 0.3.



Rezultate (setul de testare):
Relativ la setul de testare, modelul produce predicțiile din imagine (am inclus primele 5 cazuri și alte trei de interes în contextul acestui exemplu). Coloana Attrition prezintă situația reală, iar coloana „prediction(Attrition)” predicția realizată de model. Observăm că unele predicții sunt greșite (în imagine, doar cea cu Id 165). Probabilitățile asociate predicțiilor apar pe coloanele „confidence”. Pentru fiecare caz este selectată ca predicție clasa care are cea mai mare probabilitate (confidence).

Row No.	Id	Attrition	prediction(Attrition)	confidence(No)	confidence(Yes)
1	1	Yes	Yes	0.236	0.764
2	2	No	No	1.000	0.000
3	8	No	No	0.952	0.048
4	11	No	No	0.974	0.026
5	12	No	No	0.876	0.124
...					
24	90	Yes	Yes	0.306	0.694
...					
40	165	Yes	No	0.994	0.006
...					
41	167	Yes	Yes	0.025	0.975

...

Modelul de predicție estimat pe setul de date de instruire apare în secțiunea de rezultate „Logistic Regression Model" (nu le mai prezentăm aici). Tabul Data conține coeficienții logit nestandardizați și standardizați (standardizarea e relativ la variabila fiecare variabilă independentă; dacă variabila independentă este binară, coeficientul standardizat este egal cu cel nestandardizat), eroarea standard și semnificația statistică asociate acestora. La tabul Description apar o serie de informații relativ la model, măsurile de calitate și matricea de confuzie fiind cele mai importante. Observăm că modelul de predicție are o performanță foarte bună: R^2 = 0.46, media erorilor de clasificare = 0.20, iar AUC = 0.90.

Secțiunea PerformanceVector conține matricea de confuzie și diferite măsuri ale performanței clasificării (cele selectate), toate calculate relativ la ambele seturi de date (instruire și testare). Observăm că acuratețea predicției este de fiecare dată peste 80%, ceea ce indică șanse mari de generalizare a modelului de predicție. Chiar dacă de obicei calitatea clasificării este mai bună în cazul setului de instruire comparativ cu cel de test, de această dată este invers. Comparativ cu modelul care a inclus doar doi predictori, modelul cu toți predictorii are o acuratețe a clasificării similară, dar reușește să clasifice corect o parte relativ mai mare a cazurilor de angajați care pleacă (ceea ce e foarte util din punct de vedere practic). În continuare, modelul prezice mai bine situația în care angajații nu pleacă (precizia = 91/93%, reamintirea = 94/97%) comparativ cu situația în care angajații pleacă (precizia = 68/76%, reamintirea = 49/65%).

accuracy: 87.37% +/- 2.75% (micro average: 87.37%)

	true No	true Yes	class precision
pred. No	820	87	90.41%
pred. Yes	43	79	64.75%
class recall	95.02%	47.59%	

instruire, prag 50%

accuracy: 84.25% +/- 3.64% (micro average: 84.26%)

	true No	true Yes	class precision
pred. No	762	61	92.59%
pred. Yes	101	105	50.97%
class recall	88.30%	63.25%	

instruire, prag 30%

accuracy: 89.34%

	true No	true Yes	class precision
pred. No	359	36	90.89%
pred. Yes	11	35	76.09%
class recall	97.03%	49.30%	

testare, prag 50%

accuracy: 89.34%

	true No	true Yes	class precision
pred. No	348	25	93.30%
pred. Yes	22	46	67.65%
class recall	94.05%	64.79%	

testare, prag 30%

6. BAYES NAIV (NAIVE BAYES)

Nume operator	Naive Bayes
Categoria majoră	învățare asistată (supervised learning)
Tip model	clasificare (classification)
Grup	algoritm bayesian (bayesian learner)
Atribut dependent	categorial
Atribute independente	numerice, categoriale
Gestionează valorile lipsă	da (dependentă și independente)
Distorsiune (bias)	mare
Varianță (variance)	mică

Bibliografie utilizată și recomandată:

- Data mining: a tutorial-based primer (Roiger, 2017, Chapter 11.1)
- Data Science: Concepts and Practice (Kotu & Deshpande, 2019, Chapter 4.1)
- Data Analytics for the Social Sciences: Applications in R (Garson, 2021, Chapter 4)
- Machine Learning for Social and Behavioral Research (Jacobucci et al., 2023, Chapter 4.3)
- Machine Learning for Business Analytics: Concepts, Techniques and Applications in RapidMiner (Shmueli et al., 2023, Chapter 10)

6.1. Introducere

Probabilitate vs. Verosimilitate (Probability vs. Likelihood)

În limbajul comun, conceptele Probability și Likelihood sunt folosite interșanjabil, iar dicționarele le tratează ca sinonime. Însă, în statistică, cei doi termeni desemnează concepte diferite (deși legate între ele). Înainte de a prezenta termenii echivalenți în română folosiți în acest manual, e necesar să înțelegem la ce se referă cele două concepte și care e diferența dintre ele. Putem atinge mai ușor acest obiectiv pornind de la un exemplu simplu care implică două bile, una roșie și una neagră, aflate într-o urnă.

Dacă asumăm că bilele sunt identice (cu excepția culorii), anticipăm că atunci când se va extrage aleator o bilă din urnă, șansele ca aceasta să fie o bilă roșie (respectiv neagră) sunt una din două, adică 0.5 sau 50%. Altfel spus, rezultatul unei extrageri poate fi o bilă roșie sau una neagră, deci cele două rezultate posibile sunt exhaustive și mutual exclusive. În acest caz am considerat că este adevărat că bilele sunt identice și apoi am colectat datele (am extras o bilă, am notat culoarea, am reintrodus-o în urnă, am extras din nou o bilă și tot așa de un număr de ori). Gândind astfel, asumăm că modelul (parametrii acestuia) este corect și apoi colectăm datele (realizăm o serie de extrageri și numărăm în câte cazuri a ieșit o bilă roșie, respectiv una neagră). Pentru astfel de situații ne referim la șanse în sens de (sinonim cu) probabilitate. Probabilitatea se referă la șansa ca un anumit rezultat să aibă loc în baza valorilor parametrilor unui model. Altfel spus, atunci când calculăm probabilitatea unui rezultat (a unei ipoteze), asumăm că parametrii modelului sunt de încredere. Deci, **probabilitatea (P)** ne răspunde la întrebarea: **Dacă asumăm că modelul este adevărat, care sunt șansele ca ipoteza să fie adevărată?** (Tabelul 6.1-1)

$$P(eveniment) = \frac{număr\ evenimente\ favorabile}{număr\ evenimente\ posibile}$$

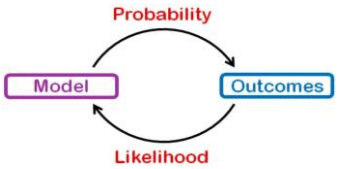
$$P(bilă\ roşie) = \frac{o\ bilă\ roşie}{două\ bile} = \frac{1}{2} = 0.5 = 50\%$$

Desigur, dacă adunăm cele două probabilități (teoretice, dar și practice) asociate exemplului nostru, suma lor este unu. În cazul probabilităților, acest lucru este adevărat întotdeauna, indiferent care este situația concretă analizată (de exemplu dacă sunt posibile trei sau mai multe tipuri de evenimente / rezultate).

Continuând exemplul anterior, să presupunem că am realizat 100 extrageri și că bila roșie a ieșit de doar 20 de ori. În acest caz spunem că șansele ca bilele să fie identice (mecanismul de extragere să fie corect) sunt destul de reduse. Altfel spus, că e puțin plauzibil sau verosimil ca bilele să fie identice / extragerea să fie corectă. Dacă procesul ar fi fost corect, ne-am fi așteptat ca bila roșie să fie extrasă în aproximativ jumătate din situații (50), deci să aibă o probabilitate empirică de apariție apropiată de 50%. Pentru a diferenția între sensul asociat conceptului de șanse așa cum este utilizat în cazul unei situații de acest tip față de situația descrisă anterior, în statistică se folosește conceptul de verosimilitate (likelihood). Prin urmare, atunci când vorbim despre verosimilitate considerăm că avem datele, presupunem că acestea sunt adevărate și ne întrebăm cât de plauzibil este modelul, parametrii acestuia (în cazul exemplului nostru, dacă probabilitatea este într-adevăr 50%, adică dacă procesul de extragere este corect). Altfel spus, verosimilitatea se referă la gradul în care datele existente oferă suport pentru anumite valori ale parametrilor unui model. Deci, **verosimilitatea (L)** ne răspunde la întrebarea: **Dacă asumăm că ipoteza este adevărată, care sunt șansele ca modelul să fie adevărat?** (Tabelul 6.1-1)

Tabelul 6.1-1. Probability vs Likelihood

Probabilitate / șansă (Probability)	**Verosimilitate / plauzabilitate (Likelihood)**
Probabilitate = P(Ipoteză | Datele) P(Ipoteză | Evidența) P(Rezultat | Modelul) P(Valoare | Distribuția)	Verosimilitate = L(Date | Ipoteza) L(Evidență | Ipoteza) L(Model | Rezultatul) L(Distribuție | Valoarea)
Asumăm că ultimul termen (Datele / Evidența / Modelul / Distribuția statistică) este adevărat și ne întrebăm care este probabilitatea ca primul termen (Ipoteza / Rezultatul / Valoarea) să fie adevărat.	Asumăm că ultimul termen (Ipoteza / Rezultatul / Valoarea) este adevărat și ne întrebăm care este verosimilitatea ca primul termen (Datele / Evidența / Modelul / Distribuția statistică) să fie adevărat.



? Nu știm / Vrem să aflăm	**Rezultat**	√ Asumăm că știm / adevărat
√ Asumăm că știm / adevărat / fix	**Model**	? Nu știm / vrem să aflăm / poate varia

Verosimilitatea (likelihood) nu este o probabilitate, dar este proporțională cu o probabilitate. Dacă H reprezintă ipoteza, D datele și K o constantă pozitivă, putem scrie ecuația $L(H) = K \times P(D|H)$. Adesea, H se referă la valorile parametrilor unui model, de exemplu valorile luate de media și abaterea standard asociate unei distribuții normale (Etz, 2018). Dacă suma probabilităților este întotdeauna egală cu unu, suma valorilor verosimilităților nu este aproape niciodată unu. Dacă o valoare de tip probabilitate poate fi interpretată izolat, o valoare de tip verosimilitate poate fi interpretată doar prin comparație cu o altă valoare de tip verosimilitate.

Reluând exemplul anterior, asumând că procesul de extragere este corect, adică P(R) = P(N) = 0.5, ne întrebăm care este probabilitatea să extragem cinci bile roșii din cinci extrageri. Dat fiind faptul că extragerile sunt independente între ele și P(R) = 0.5, $P(RRRRR \mid P(R)=0.5) = 0.5^5 = 0.03125$. Pe de altă parte, dacă deja am realizat cinci extrageri și a rezultat succesiunea RRRRR, ne întrebăm care este verosimilitatea ca procesul de extragere să fie unul corect, adică ne întrebăm dacă într-adevăr P(R) = P(N) = 0.5. Făcând calculele,[1] rezultă că $L(P(R)=0.5 \mid RRRRR) = 0.03125$.[2] Pentru alte valori ale

[1] Folosim funcția dbinom(x, n, p) din R, unde x = numărul evenimentelor favorabile (numărul de bile roșii extrase), n = numărul total de evenimente (numărul de extrageri), iar p = probabilitatea (probabilitatea să extragem o bilă roșie).

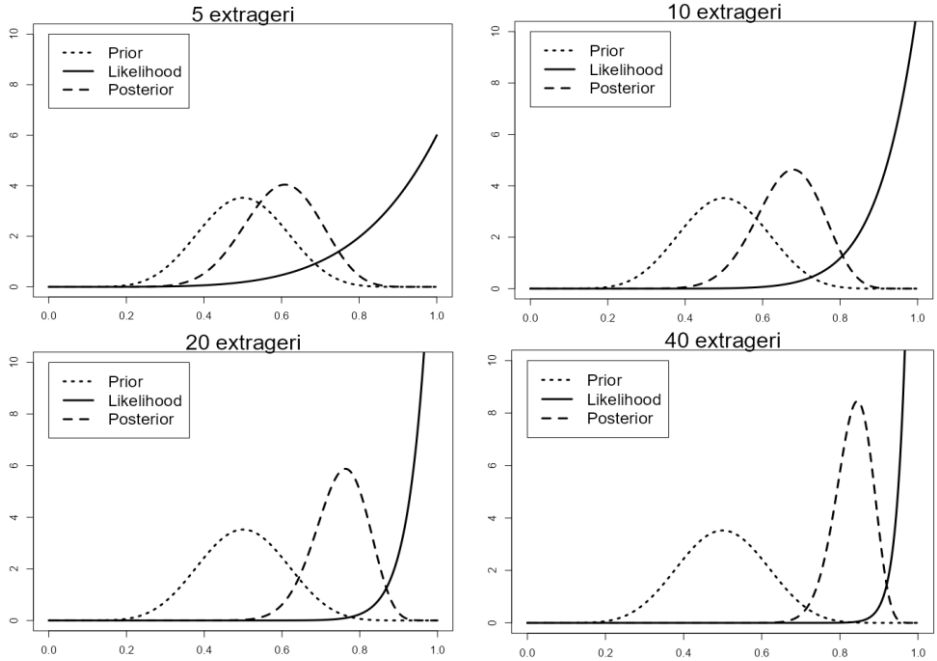
[2] Observăm că această valoare este identică cu cea obținută anterior în cazul probabilității. Acest lucru se întâmplă de fiecare dată atunci când probabilitatea de referință este aceeași (0.5 în acest caz).

lui P, valorile lui L sunt: L(P(R)=0.6 | RRRRR) = 0.07776, L(P(R)=0.7 | RRRRR) = 0.16807, L(P(R)=0.8 | RRRRR) = 0.32768, L(P(R)=0.9 | RRRRR) = 0.59049. Deoarece valoarea cea mai mare a lui L, relativ la valorile P testate, este pentru P = 0.9, rezultă că e relativ mai verosimil ca secvența RRRRR să fie produsă de o distribuție binominală cu P(R) = 0.9 și nu de una cu P(R) = 0.5. Putem spune că e relativ mai verosimil ca procesul de extragere să nu fie corect însă nu putem exclude total posibilitatea ca acesta să fie corect. Prin urmare, în acest caz, cele două ipoteze testate nu sunt nici exhaustive și nici exclusive.[3] Dacă am crește numărul de extrageri și am obține din nou tot doar bile roșii, verosimilitatea ca procesul de extragere să fie corect, adică P(R) = 0.5, ar fi relativ tot mai mică pe măsură ce numărul extragerilor ar crește: L(P(R)=0.5 | RRRRRR) = 0.015625, L(P(R)=0.5 | RRRRRRR) = 0.0078125 etc.[4]

În exemplul anterior am considerat un atribut categorial. Desigur, probabilitatea și verosimilitatea pot fi calculate și dacă atributul este metric. Să considerăm un atribut numit Scor care are o distribuție normală (gaussiană) cu media 100 și abaterea standard 15 (Figura 6.1-1). Cu privire la această situație ne întrebăm care este probabilitatea ca o persoană să obțină un scor de cel puțin 120 puncte, adică

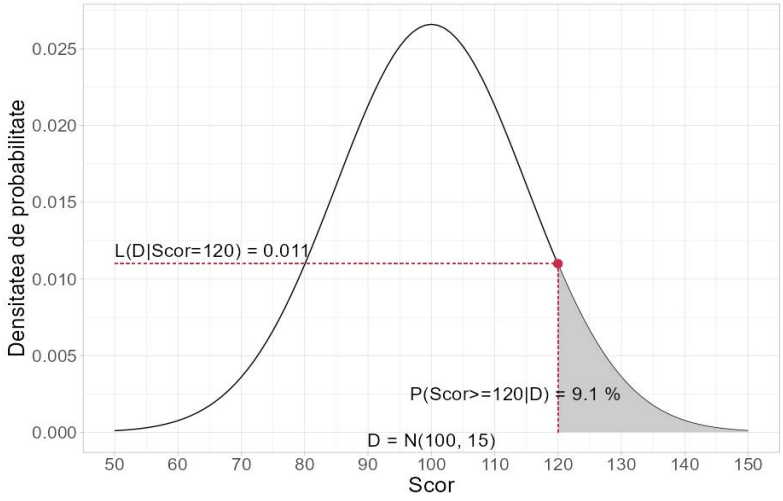
[3] Pentru o altă valoare a lui P (0.95 de exemplu), L este mai mare. De asemenea, e posibil ca funcția să fie una bimodală.

[4] Anticipând puțin discuția din sub-capitolul următor, prezentăm aici distribuțiile prior, likelihood și posterior în cazul unui număr diferit de extrageri (valorile parametrilor care definesc densitatea distribuției binomiale prior sunt a=10 și b=10). Observăm că pe măsură ce crește numărul extragerilor, ipoteza unui proces incorect devine tot mai probabilă relativ la ipoteza unui proces corect.



P(Scor≥120 | N(100, 15)) = 9.1% (aria gri; toată aria de sub curbă este egală cu 1). Dacă am observat că scorul este 120, verosimilitatea ca distribuția să fie N(120, 15), adică L(N(100, 15) | Scor=120) = 0.011. Desigur, dacă ne raportăm la o distribuție cu alte valori ale parametrilor, valoarea L se schimbă dat fiind același scor. De exemplu, dacă media noii distribuții este 120 (abaterea standard e tot 15), vom avea L(N(120, 15) | Scor=120) = 0.027 (adică L maxim pentru o distribuție cu acești parametri).

Figura 6.1-1. Probabilitate vs. Verosimilitate (Probability vs. Likelihood) în cazul unui atribut metric



Linia în formă de clopot reprezintă funcția de densitate a probabilității (probability density function - PDF) asociată distribuției D, adică N(100, 15); D = distribuție statistică; N(100, 15) = o distribuție statistică normală (gaussiană) cu media 100 și abaterea standard 15; L = verosimilitate (likelihood); P = probabilitate (probability); aria gri reprezintă probabilitatea ca scorul să fie cel puțin 120 (0.091 sau 9.1%); punctul roșu indică intersecția dintre perpendiculara pe axa orizontală asociată scorului 120 și curbă (are coordonatele (120; 0.011)); linia orizontală roșie e perpendiculara din punctul roșu pe axa verticală, intersecția indicând valoarea verosimilității (likelihood) (0.011).

Teorema lui Bayes

Pentru a înțelege cum funcționează algoritmul bayesian este necesar să înțelegem teorema lui Bayes. Pentru a ajunge la teoremă, pornim de la un exemplu intuitiv simplu.[5] Să ne imaginăm următoarea situație ipotetică: Ion a făcut un test pentru cancer. Testul a ieșit pozitiv. Ion consideră că sigur are cancer. Doctorul îi spune că nu e sigur că are cancer și că ar trebui ca Ion să refacă testul și/sau să facă un alt test. Doctorul îl încurajează pe Ion spunându-i că în baza informațiilor disponibile la

[5] Exemplul prezentat în această secțiune este inspirat din cartea „How Machines Learn: An Illustrated Guide to Machine Learning", capitolul Bayesian algotithms (Edwards, 2016).

acest moment este mai probabil să nu aibă cancer. Ion se uită la doctor cu neîncredere, spunându-și în minte că acesta îi ascunde adevărul deoarece nu vrea să-l sperie. Cine are dreptate? Doctorul sau Ion?

Pentru a putea răspunde la această întrebare, pe lângă faptul că știm rezultatul testului (pozitiv), avem nevoie de alte două informații:

- Care este incidența (probabilitatea) cancerului la nivelul populației?
- Cât de sigur este testul? Mai exact, e necesar să știm probabilitatea ca testul să descopere cancerul atunci când acesta chiar există (true positive rate), respectiv probabilitatea ca testul să indice că o persoană are cancer deși aceasta nu are cancer (false positive rate).

Din datele statistice știm că o persoană din o sută are cancer, deci probabilitatea de cancer la nivelul populației este 1% (0.01). În ceea ce privește calitatea testului, știm că „true positive rate" este 80% (0.8) și că „false positive rate" este 10% (0.1). Simplu spus, dacă testăm o sută de persoane care chiar au cancer, testul va arăta (corect) că doar 80 din ele au cancer, deci nu detectează 20 cazuri de cancer; dacă testăm o sută de persoane care nu au cancer, testul va indica (greșit) că 10 din ele au cancer. Toate aceste date pot fi puse într-un tabel simplu (Tabelul 6.1-2).

Tabelul 6.1-2. Relația dintre cancer și rezultatul testului (date fictive)

Incidență cancer / **Rezultat test**	Fără cancer (99%)	Cu cancer (1%)
Negativ	90% (true -)	20% (false -)
Pozitiv	10% (false +)	80% (true +)

Prin urmare, dacă ...:

- Ion are cancer, șansele ca testul să fie ...

  □ pozitiv sunt 80%
  □ negativ sunt 20%.

- Ion nu are cancer, șansele ca testul să fie ...

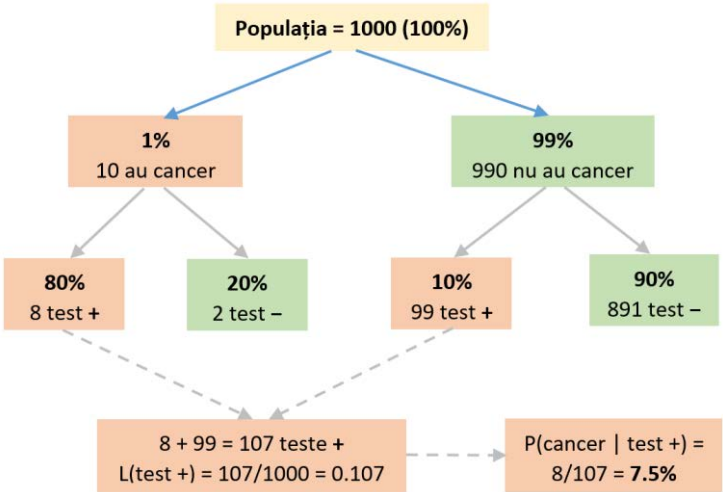
  □ pozitiv sunt 10%
  □ negativ sunt 90%.

Știm că testul lui Ion este pozitiv, prin urmare verosimilitatea (likelihood) ca Ion să fie un caz de tip „true positive" (chiar să aibă cancer) este 0.008 (ponderea

cancerului în populație * TPR, adică 1% * 80% = 0.008). Verosimilitatea ca Ion să fie un caz de tip „false positive" (să nu aibă cancer deși testul a indicat că are) este 0.099 (Tabelul 6.1-3). Prin urmare, verosimilitatea ca Ion să primească un răspuns pozitiv (că are cancer), indiferent dacă acest lucru este adevărat sau nu, este 0.107 (0.008 + 0.099). De aici rezultă că, în urma testului pozitiv, probabilitatea ca Ion să aibă cancer este 7.5% (0.008 / 0.107 = 0.075). Observăm că această valoare este relativ mai apropiată de 0 decât de 1, deci medicul a avut dreptate: deși a fost testat pozitiv, este puțin probabil ca Ion să aibă cancer (chiar dacă probabilitatea a crescut în urma testului pozitiv; înainte de test era 1%; după testul pozitiv e 7.5%). De ce se întâmplă asta? În principal pentru că incidența cancerului este foarte scăzută. Secundar, deoarece testul nu este perfect.

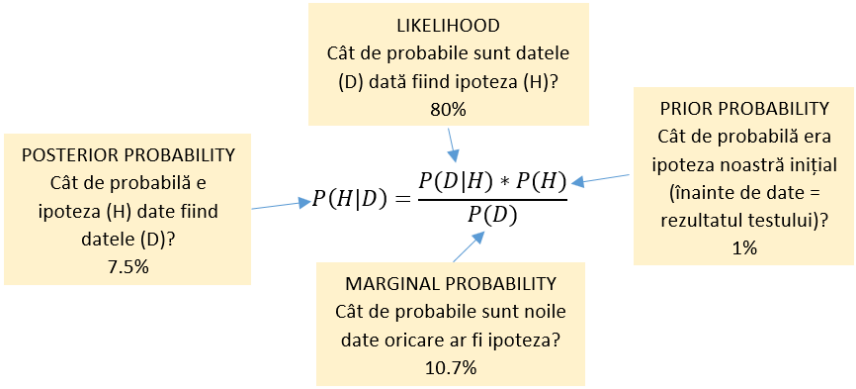
Tabelul 6.1-3. Probabilitatea de a avea cancer în urma unui test pozitiv (date fictive)

True +	Probabilitatea de a avea cancer la nivel de populație x Probabilitatea ca testul să indice prezența cancerului	1% x 80%	**= 0.008**
False +	Probabilitatea de a nu avea cancer la nivel de populație x Probabilitatea ca testul să indice prezența cancerului	99% x 10%	**= 0.099**
	Verosimilitatea ca Ion să primească un răspuns pozitiv (că are cancer), indiferent dacă este adevărat sau nu		**= 0.107**
	Probabilitatea ca Ion să aibă cancer	$\frac{0.008}{0.107}$	**= 7.5%**



Dacă luăm în considerare doar incidența cancerului la nivelul populației, probabilitatea ca Ion să aibă cancer este de doar 1%. Dacă ținem cont doar de rezultatul testului, probabilitatea ca Ion să aibă cancer este 80%. Ambele perspective, considerate separat, sunt incorecte. Ar trebui să ținem cont de ambele. Putem face asta cu ajutorul teoremei lui Bayes (vezi Figura 6.1-2). Notațiile utilizate sunt ușor diferite față de cele utilizate în mod obișnuit[6] deoarece am dorit să păstrăm convențiile din manualele de statistică destinate studenților din științele sociale.

Figura 6.1-2. Teorema lui Bayes



unde,

- H este ipoteza de interes, iar D reprezintă datele (valorile luate de unul sau mai multe atribute / predictori, adică un vector).
- P(H) și P(D) sunt probabilitățile de adevăr asociate ipotezei, respectiv datelor, adică
  - P(H), probabilitatea inițială (**prior probability**), numită și probabilitate generală / globală, este probabilitatea ca ipoteza să fie adevărată independent de date;

[6] Formula sub care este cunoscută această teoremă este $P(A|B) = \frac{P(B|A) * P(A)}{P(B)}$, unde P(A|B) reprezintă probabilitatea evenimentului A condiționată de evenimentul B (probabilitatea ca A să aibă loc atunci când B are loc), P(B|A) probabilitatea evenimentului B condiționată de evenimentul A, P(A) și P(B) probabilitatea evenimentului A, respectiv B. De exemplu, în locul lui A putem pune părăsirea companiei, iar în locul lui B putem pune departamentul Producție sau o combinație de clase a mai multor atribute, de exemplu Departament: Producție + Poziție: Execuție.

- □ P(D), probabilitatea marginală (**marginal probability**), numită și probabilitatea evidenței, este probabilitatea ca datele să fie adevărate independent de ipoteză.

– P(H|D) și P(D|H) sunt probabilități condiționate, adică

- □ P(H|D), probabilitatea posterioară (**posterior probability**), este probabilitatea ca ipoteza (H) să fie adevărată date fiind datele (D), adică acea valoare a atributului / combinație de valori ale atributelor;
- □ P(D|H), verosimilitatea (**likelihood**), este probabilitatea ca datele să fie adevărate dată fiind ipoteza; dacă ipoteza (H) este adevărată, care e probabilitatea să observăm acele date (D), adică acea valoare a atributului / combinație de valori ale atributelor.

Dacă în locul lui H punem „are cancer", iar în locul lui D punem „testul este pozitiv", rezultă că șansele ca Ion să aibă cancer dat fiind faptul că testul său a fost pozitiv sunt egale cu 7.5% (adică exact valoarea obținută anterior).

$$P(H|D) = \frac{80\% * 1\%}{10.7\%} = \frac{0.008}{0.107} = 0.075 = 7.5\%$$

Deoarece numitorul este același indiferent de care este clasa atributului de interes (de exemplu, „are cancer" vs „nu are cancer"), teorema lui Bayes poate fi simplificată astfel (aceasta este și forma sub care se regăsește adesea) (Moreira et al., 2019, pp. 209–210):

$$P(H|D) = P(D|H) * P(H)$$

Aplicând teorema lui Bayes putem calcula probabilitățile asociate celor patru combinații posibile în cazul exemplului nostru (Tabelul 6.1-4). Observăm că probabilitatea de cancer înainte de primul test este 1%, crește la 7.5% după un prim test pozitiv, apoi crește la 39.3% dacă și testul secund este pozitiv.

Tabelul 6.1-4. Calcularea probabilităților posterioare asociate celor patru situații posibile

TEST 1	Prior	Likelihood		Joint		Posterior	
Cancer		Test +	Test –	Test +	Test –	Test +	Test –
Da	0.01	0.80	0.20	0.008	0.002	0.075	0.002
Nu	0.99	0.10	0.90	0.099	0.891	0.925	0.998
				0.107	0.893	1.000	1.000

TEST 2 Cancer	Prior	Likelihood		Joint		Posterior	
		Test +	Test –	Test +	Test –	Test +	Test –
Da	0.075	0.80	0.20	0.060	0.015	0.393	0.018
Nu	0.965	0.10	0.90	0.093	0.833	0.607	0.982
				0.152	0.848	1.000	1.000

Cu privire la formula de mai sus, dacă D se referă la o serie de atribute (evenimente) independente între ele (D_1, D_2, ..., D_n), atunci egalitatea următoare este adevărată:

$$P(D|H) = P(D_1|H) * P(D_2|H) * ... * P(D_n|H)$$

În realitate, cel mai adesea atributele nu sunt independente, deci această egalitatea este rareori adevărată. Dat fiind faptul că algoritmul bayesian asumă că atributele sunt independente, a primit denumirea de **Naive** Bayes. Adesea, acest lucru nu reprezintă neapărat o problemă deoarece algoritmul bayesian își propune să obțină o ordonare în termeni relativi, nu absoluți, a șanselor asociate claselor.

Exact Bayes

Pentru a înțelege algoritmul Naive Bayes e nevoie să înțelegem prima dată versiunea completă / exactă a acestuia, și anume Exact Bayes (Shmueli et al., 2017). Să presupunem că o companie a avut de-a lungul timpului mai mulți angajați și că unii dintre aceștia nu mai lucrează în prezent în acea companie. Cu privire la toți angajații, actuali și foști, cunoaștem câteva lucruri. Astfel, relativ la fiecare (fost) angajat știm următoarele (atributele și clasele acestora):

- Dacă a părăsit compania: da / nu;
- Departamentul: producție / vânzări;
- Poziția ocupată: conducere / execuție;
- Sex: bărbat / femeie;
- Are copii (0-18 ani): da / nu;
- Face naveta: da / nu.

Atributul Id conține codul de identificare al angajatului, „Părăsit” este atributul de interes, cel pe care dorim să-l prezicem, iar restul atributelor sunt predictorii. Setul de date aferent acestei situații ipotetice are 20 de cazuri și este prezentat în Tabelul 6.1-5.

Tabelul 6.1-5. Un set de date fictive cu (foști) angajați ai unei companii (20 cazuri)

Id	Părăsit	Departament	Poziție	Sex	Copii<18	Navetă
1	da	Producție	conducere	bărbat	da	da
2	nu	Producție	conducere	bărbat	nu	nu
3	nu	Producție	conducere	femeie	da	nu
4	da	Producție	execuție	bărbat	da	nu
5	nu	Producție	execuție	bărbat	da	nu
6	nu	Producție	execuție	bărbat	da	nu
7	nu	Producție	execuție	bărbat	da	nu
8	nu	Producție	execuție	bărbat	da	nu
9	nu	Producție	execuție	bărbat	nu	nu
10	nu	Producție	execuție	bărbat	nu	nu
11	nu	Producție	execuție	bărbat	nu	nu
12	da	Producție	execuție	femeie	da	da
13	nu	Producție	execuție	femeie	da	nu
14	nu	Producție	execuție	femeie	nu	nu
15	da	Vânzări	conducere	bărbat	da	nu
16	nu	Vânzări	conducere	bărbat	da	nu
17	da	Vânzări	execuție	bărbat	da	nu
18	nu	Vânzări	execuție	bărbat	da	nu
19	nu	Vânzări	execuție	bărbat	nu	nu
20	da	Vânzări	execuție	femeie	da	da

Să presupunem că dorim să angajăm o persoană pe o poziție de execuție în domeniul producție. Firesc, ne dorim ca angajatul să rămână cât mai mult timp în companie (mai ales dacă angajatul are o valoare relativ mai mare pentru companie). Unul dintre candidați este bărbat, are copii și locuiește în aceeași localitate în care se află și unitatea de producție **(situația A)**. Dorim să estimăm șansele ca acest candidat să nu plece ulterior din companie. Pentru aceasta, prima dată ne uităm la setul de date și selectăm angajații care au aceleași caracteristici, cu excepția informației relativ la părăsirea companiei. Atenție, considerăm atât angajații actuali cât și foștii angajați. Observăm că setul de date conține cinci astfel de angajați (Tabelul 6.1-5, cazurile 4-8). Dintre aceștia, doar unul a părăsit compania (Id=4), iar patru au rămas. În concluzie, predicția noastră va fi că un candidat cu aceste caracteristici nu va părăsi compania (probabilitatea să rămână este 4/5=80%; să plece 1/5=20%).

Similar, ne poate interesa să selectăm un candidat pentru o poziție de conducere în departamentul vânzări. Unul dintre candidații selectați este bărbat, are copii și nu trebuie să facă naveta **(situația B)**. Setul de date conține doi (foști) angajați cu aceste

caracteristici (Tabelul 6.1-5, cazurile 15 și 16), unul dintre aceștia nemaifiind angajat în prezent. Pentru această situație, predicția noastră va fi 50-50% (indecis). Ambele exemple au implicat atribute categoriale. Însă, raționamentul se aplică și în cazul unor atribute de tip metric.

Cele două predicții realizate anterior pot fi evaluate în relație cu două praguri, unul absolut și unul relativ. Dat fiind faptul că, în acest caz, atributul de interes (variabila dependentă) are două categorii, pragul absolut este 50%. Pragul relativ este egal cu proporția clasei de referință la nivelul întregului set de date. În exemplul de față, 6 angajați din 20 au plecat (14 au rămas), prin urmare pragul relativ asociat clasei „pleacă" este 6/20 (30%). Pentru situația A, predicția va fi aceeași, „nu pleacă", indiferent de tipul de prag ales (20% e mai mic decât pragul absolut de 50% sau cel relativ de 30%). Pentru situația B, predicția va fi „pleacă" dacă ne raportăm la pragul relativ, respectiv indecis dacă avem în vedere pragul absolut.

Pornind de la cele două exemple prezentate anterior putem caracteriza pașii algoritmului Exact Bayes astfel (Shmueli et al., 2017, p. 199):

(1) aflăm numărul de cazuri care, raportat la seria de predictori, au același profil; simplu spus, numărăm cazurile care iau aceleași valori relativ la fiecare predictor;
(2) pentru fiecare profil (combinație de predictori), aflăm numărul de cazuri care aparțin fiecărei clase a atributului de interes (variabila dependentă);
(3) fiecare caz pe care dorim să-l prezicem are un anumit profil; raportat la cazurile cu același profil, acest caz va fi alocat la clasa cu cea mai mare frecvență.

Atunci când numărul de predictori este mare și avem relativ puține cazuri, șansele ca setul de date să conțină cazuri relativ la toate profilele posibile (combinațiile de atribute) sunt foarte mici. De exemplu, în cazul setului nostru de date combinația „vânzări +conducere + bărbat + copii + navetă" lipsește. Prin urmare, dacă folosim algoritmul Exact Bayes nu vom putea prezice clasa la care aparține un caz cu aceste caracteristici. Pentru a rezolva această problemă Naive Bayes adaugă câte două pseudo-cazuri,[7] câte unul pentru fiecare clasă a atributului de interes, la fiecare clasă a fiecărui atribut obișnuit. Procedând astfel, fiecare combinație dintre atributul de interes (label / dependentă) și oricare atribut obișnuit va avea cel puțin un caz, deci niciuna dintre probabilitățile asociate nu va mai putea lua valoarea zero.

[7] Dacă atributul de interes (label) are mai multe clase, algoritmul va adăuga câte un pseudo-caz pentru fiecare clasă.

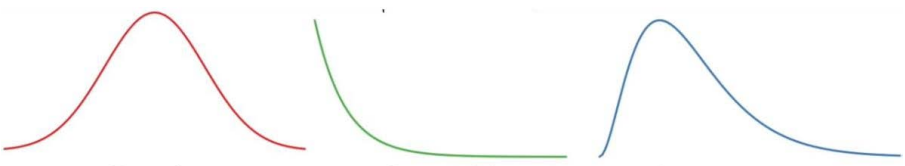
O altă problemă relativ la Exact Bayes este reprezentată de instabilitatea estimărilor ca urmare a faptului că unele profile (combinații de atribute) au un număr mic de cazuri. Naive Bayes rezolvă această problemă prin faptul că, atunci când calculează probabilitățile (respectiv face predicțiile), ține cont de toate cazurile, nu doar de cazurile care au același profil cu cazul pe care dorește să-l prezică.

6.2. Logica și pașii algoritmului Naive Bayes

Descriere generală

Naive Bayes este un algoritm simplu, rapid și puternic (RapidMiner, 2022, pp. 451–452) ce poate fi folosit pentru a prezice o variabilă de tip categorial (apartenența cazurilor la clasele / categoriile unei variabile dependente). Naive Bayes este utilizat adesea pentru a clasifica texte, cu scopul de a detecta automat email-urile de tip spam (spam detection), a analiza sentimentele / emoțiile (sentiment analysis) sau a face recomandări (recommender systems). Dincolo de aceste domenii, Naive Bayes poate fi folosit, în general cu succes, indiferent de domeniu, oricând dorim să prezicem un atribut de tip categorial. Predictorii pot fi de tip categorial sau metric. În cazul predictorilor categoriali Naive Bayes ia forma multinominală, iar în cazul celor metrici ia forma unei distribuții de tip Gaussian (cel mai adesea, dar nu neapărat; funcție de tipul datelor, putem folosi, de exemplu, o funcție de tip Exponențial sau una Gamma).[8]

Naive Bayes este un algoritm simplu pentru că are doar un parametru care trebuie setat (și acesta, tot simplu).[9] Este rapid deoarece citește setul de date o singură dată, îl sintetizează (calculează numărul de cazuri pentru fiecare clasă a atributului de interes, în total și relativ la fiecare atribut obișnuit), apoi calculează și compară probabilitățile asociate diferitelor combinații de atribute. Simplu spus, costurile asociate realizării calculelor sunt relativ mici chiar și atunci când setul de date are foarte multe cazuri și atribute. Este puternic deoarece poate gestiona diferite tipuri



8

9 Corecția Laplace, cu două opțiuni (da / nu). În cazul algoritmului Naive Bayes Kernel, mai apare cel puțin încă un parametru (patru în RapidMiner Studio).

de probleme și date, respectiv poate lucra și cu seturi mici de date, producând cel mai adesea modele cu o calitate relativ ridicată a predicției.

Pentru a oferi toate aceste avantaje, Naive Bayes face două asumpții majore, și anume (Attewell & Monaghan, 2015, p. 142; James et al., 2021, p. 155; Kotu & Deshpande, 2019, p. 112):

(1) consideră că toți predictorii sunt la fel de importanți;
(2) consideră că, pentru oricare dintre clasele variabilei dependente, valoarea unui predictor nu depinde de valoarea altui predictor; simplu spus, relativ la fiecare clasă a variabilei dependente, consideră că predictorii sunt independenți între ei.

În baza acestor asumpții, calculele sunt simplificate foarte mult. În realitate, asumpțiile respective sunt rareori adevărate, de unde și prezența cuvântului „naiv" în denumirea algoritmului. Practic, asumpțiile fac ca algoritmul să fie oarecum distorsionat (biased), adică să obțină estimări care diferă sistematic (erorile de predicție au același sens) față de valorile reale.[10] Pe de altă parte, algoritmul are o varianță redusă (estimările obținute diferă relativ puțin între ele). Cu toate aceste limite, aproximările obținute de Naive Bayes sunt suficient de bune pentru ca algoritmul să performeze cel mai adesea relativ bine (James et al., 2021, p. 155).

Așa cum este implementat în RapidMiner Studio, algoritmul Naive Bayes poate gestiona atribute dependente și independente cu valori lipsă (unele implementări din alte softuri nu acceptă valori lipsă). Desigur, dacă dorim, înainte de a construi un model de predicție, putem înlocui valorile lipsă din setul de date.[11]

Un exemplu didactic – Multinominal Naive Bayes

Pentru a înțelege modul în care (Multinominal) Naive Bayes produce predicțiile, vom folosi același exemplu simplu prezentat anterior.[12] Reamintim faptul că ne interesează să prezicem părăsirea locului de muncă în cazul angajaților unei

[10] Unele studii arată că, într-adevăr, Naive Bayes este adesea cel mai puțin performant algoritm (Vanacore et al., 2022).

[11] Pentru lucrul cu valori lipsă se poate consulta primul volum, capitolul 6.3 (Valorile lipsă).

[12] Pentru un alt exemplu ilustrativ, se poate consulta și textul lui Ingo Mierswa „Naive Bayes – Not so naive after all!". O prezentare foarte simplă și intuitivă a algoritmului este realizată de Josh Starmer în videoul „Naive Bayes, Clearly Explained!!!", respectiv în cartea sa „The StatQuest Illustrated Guide to Machine Learning!!!" (Joshua Starmer, 2022, Chapter 7). Un alt video util este „Naive Bayes classifier: A friendly approach".

companii mici, iar setul de date conține informații relativ la 20 de (foști și actuali) angajați (Tabelul 6.1-5). Pentru acest exemplu presupunem că știm doar următoarele: dacă angajatul a părăsit sau nu compania (atributul pe care dorim să-l prezicem), departamentul din care face parte angajatul și poziția ocupată de acesta (predictorii). Fiecare predictor are doar două clase: departamentul poate fi producție sau vânzări, iar poziția poate fi conducere sau execuție.

Pentru a estima modelul de predicție – faza de modelare (modeling), adică pașii 1 și 2 din lista de mai jos – și a realiza predicțiile – faza de calculare și alocare a probabilităților (scoring), adică pașii 3, 4 și 5 –, algoritmul Naive Bayes (varianta multinominală) procedează după cum urmează (vezi și Exemplul 6.2-1):[13]

(1) calculează numărul de cazuri (counts) pentru fiecare clasă a atributului de interes în relație cu fiecare predictor (clase / categorii ale acestuia, dacă predictorul este de tip categorial);
(2) calculează probabilitățile condiționate (conditional probabilities) pentru fiecare clasă a atributului prezis în relație cu fiecare predictor (clase ale acestuia, dacă predictorul este de tip categorial), respectiv probabilitățile generale asociate claselor atributului prezis;
(3) calculează produsul dintre probabilitățile condiționate și probabilitatea generală (likelihood) pentru toate combinațiile de clase ale predictorilor (dacă atributele sunt categoriale, se calculează mediile și abaterile standard condiționate);
(4) calculează probabilitățile prezise (predicted probabilities) asociate diferitelor combinații de clase ale atributelor;
(5) fiecare caz pe care dorim să-l prezicem are un anumit profil; raportat la cazurile cu același profil, acest caz va fi alocat la clasa cu cea mai mare probabilitate prezisă a atributului de interes.

Primul pas constă în numărarea cazurilor care fac parte din fiecare clasă a atributului de interes în cadrul fiecărei clase a celorlalte atribute (Exemplul 6.2-1, Pasul 1). De exemplu, în cazul setului nostru de date (Tabelul 6.1-5), relativ la atributul Departament există 3 angajați din producție care au părăsit compania și 11 care nu au părăsit-o, respectiv 3 angajați din vânzări care au rămas în companie și 3 care nu au rămas. Similar, relativ la atributul binominal Poziție, există 4 angajați pe poziție de execuție care au părăsit compania și 11 care nu au părăsit-o, respectiv 3 angajați pe poziție de conducere care au rămas în companie și 2 care nu au rămas. În cazul atributelor numerice, algoritmul va calcula media și abaterea standard pentru fiecare dintre clasele atributului de interes (a părăsit sau nu compania) (vezi

[13] Uneori, pașii algoritmului sunt descriși parțial diferit în literatură, dar, în esență, sunt aceiași (Shmueli et al., 2017, p. 199).

secțiunea următoare: Un exemplu didactic – Gaussian Naive Bayes). Tot la acest pas, algoritmul însumează numărul de cazuri pentru fiecare clasă a atributului prezis: 14 angajați care au rămas și 6 care au plecat.

La pasul doi sunt calculate probabilitățile condiționate pentru fiecare clasă a atributului de interes în relație cu fiecare predictor (Exemplul 6.2-1, Pasul 2). De exemplu, în cazul atributului Departament, din cei 6 angajați care au părăsit compania, 3 sunt din producție și 3 din vânzări, deci probabilitățile condiționate sunt 3/6 (50%), respectiv 3/6 (50%). Raportat la atributul Poziție, din cei 6 angajați care au părăsit compania, 4 sunt pe poziție de execuție și 2 de conducere, deci probabilitățile condiționate sunt 4/6 (66.7%), respectiv 2/6 (33.3%). În cazul atributului de interes (label) sunt calculate probabilitățile generale ale claselor și anume 14/20 (70%), respectiv 6/20 (30%). Prin urmare, în acest caz, probabilitatea ca un angajat oarecare să plece din companie este 30%. În cazul atributelor numerice, algoritmul calculează probabilitățile folosind valorile medii și abaterile standard calculate la pasul 1 (în baza asumpției că distribuțiile respective sunt de tip Gaussian, adică normale).[14]

La pasul trei sunt calculate verosimilitățile (likelihoods) asociate claselor atributului prezis pentru fiecare combinație de atribute obișnuite (clasele acestora) prin înmulțirea probabilităților calculate la pasul doi. Pentru acest exemplu vom considera situația unui angajat pe poziție de execuție din departamentul de producție. Verosimilitatea ca un angajat care pleacă din companie să aibă această combinație de caracteristici (producție + execuție) este: 3/6 * 4/6 * 6/20 = 0.1 (Exemplul 6.2-1, Pasul 3). Primii doi termeni sunt probabilitățile condiționate (probabilitățile ca un angajat care pleacă să fie din producție, respectiv să ocupe o poziție de execuție), iar ultimul termen este probabilitatea generală ca un angajat să părăsească firma indiferent de caracteristicile lui. Similar, calculăm verosimilitatea de a nu pleca din companie în cazul aceluiași tip de angajat (producție + execuție): 11/14 * 11/14 * 14/20 = 0.4321 (Exemplul 6.2-1, Pasul 3). Observăm că raportul verosimilităților (likelihood ratio) este 4.3 (0.4321/0.1), adică verosimilitatea ca un angajat din producție pe poziție de execuție să nu plece este de aproximativ patru ori mai mari comparativ cu verosimilitatea să plece.

La final, algoritmul transformă verosimilitățile calculate anterior în probabilități și alocă cazurile pe care dorim să le prezicem la clasele cu probabilitățile cele mai mari (raportat la cazurile care au același profil). Continuând exemplul anterior, în cazul unui angajat pe poziție de execuție în departamentul de producție, probabilitatea de a părăsi compania este 18.8% (0.1 / (0.1 + 0.4321)), iar probabilitatea de a

[14] Alternativ, atributele numerice pot fi discretizate.

rămâne în companie este 81.2% (0.4321 / (0.1 + 0.4321)). Dat fiind faptul că probabilitatea de a rămâne în companie este mai mare decât probabilitate de a pleca, vom prezice că un angajat din producție pe poziție de execuție nu va părăsi compania (raportat la ambele praguri, relativ și absolut). Observăm că rezultatele obținute aici sunt identice cu predicțiile făcute de RapidMiner (vezi și procesul nb_didactic_1 din folderul Exemple).[15]

Exemplul 6.2-1. Pașii algoritmului Naive Bayes

Pasul 1: Calcularea frecvențelor absolute / numărului de cazuri (counts)

	A plecat			A plecat			A plecat	
Departament	**Nu**	**Da**	**Poziție**	**Nu**	**Da**	**Părăsit**	**Nu**	**Da**
Producție	11	3	conducere	3	2		14	6
Vânzări	3	3	execuție	11	4			

Exemple: setul de date conține (vezi și Tabelul 6.1-5):

3 angajați în departamentul producție care au părăsit compania;
2 angajați pe poziție de execuție care au părăsit compania.

Pasul 2: Calcularea probabilităților condiționate (conditional probabilities)

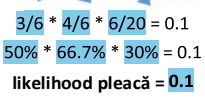
	A plecat			A plecat			A plecat	
Departament	**Nu**	**Da**	**Poziție**	**Nu**	**Da**	**Părăsit**	**Nu**	**Da**
Producție	11/14	3/6	conducere	3/14	2/6		14/20	6/20
Vânzări	3/14	3/6	execuție	11/14	4/6			

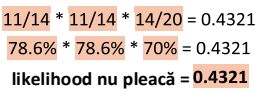
Exemple de calculare a probabilităților condiționate:

probabilitatea ca un angajat care pleacă să fie din producție este 3/6 (50%);
probabilitatea ca un angajat care pleacă să fie pe poziție de execuție este 4/6 (66.7%).

Pasul 3: Calcularea verosimilităților (likelihoods)

Departament	**Nu**	**Da**	**Poziție**	**Nu**	**Da**	**Părăsit**	**Nu**	**Da**
Producție	11/14	3/6	conducere	3/14	2/6		14/20	6/20
Vânzări	3/14	3/6	execuție	11/14	4/6			





[15] Rezultatele sunt identice doar dacă parametrul „laplace correction" nu este selectat. Dacă îl selectăm, rezultatele se schimbă foarte puțin.

Pasul 4: Calcularea probabilităților prezise (predicted probabilities)

Exemplu de calculare a șanselor în cazul unui angajat din producție, pe poziție de execuție.
probabilitatea de a părăsi compania: 0.1 / (0.1 + 0.4321) = 18.8%
probabilitatea de nu părăsi compania: 0.4321 / (0.1 + 0.4321) = 81.2%

Pasul 5: Realizarea predicțiilor

Exemplul 1: dorim să angajăm o persoană în producție, pe o poziție de execuție;
probabilitatea prezisă ca un angajat cu aceste caracteristici să nu părăsească firma este 81.2%;
dat fiind faptul că această valoare este peste pragul de 50%, predicția va fi „rămâne”.

Exemplul 2: dorim să angajăm o persoană în vânzări, pe o poziție de conducere;
probabilitatea prezisă ca un angajat cu aceste caracteristici să părăsească firma este 60.9%;[16]
dat fiind faptul că această valoare este peste pragul de 50%, predicția va fi „pleacă”.

Exemplul 3: dorim să angajăm o persoană în vânzări, pe o poziție de execuție;
probabilitatea prezisă ca un angajat cu aceste caracteristici să părăsească firma este 45.9%;[17]
dat fiind faptul că această valoare este sub pragul de 50%, predicția va fi „rămâne”;
dacă ne raportăm la un prag mai mic, de exemplu 40%, predicția va fi „pleacă”.

Rezultate (fără corecția Laplace):

		Likelihood		Probabilitate	
Departament	**Poziție**	**Nu pleacă**	**Pleacă**	**Nu pleacă**	**Pleacă**
producție	conducere	0.118	0.050	70.2%	29.8%
producție	execuție	0.432	0.100	81.2%	18.8%
vânzări	conducere	0.032	0.050	39.1%	60.9%
vânzări	execuție	0.118	0.100	54.1%	45.9%

Uneori, este posibil ca setul de date să nu conțină nicio observație pentru una dintre combinațiile dintre clasele atributului de interes și clasele unui predictor. Dacă se întâmplă acest lucru, probabilitatea condiționată respectivă va fi zero, deci, atunci când acest termen face parte dintr-un produs de probabilități condiționate, rezultatul va fi întotdeauna zero, indiferent care sunt restul termenilor. Prin urmare, și probabilitatea prezisă respectivă va fi întotdeauna zero. Pentru a evita apariția unor astfel de situații, se folosește corecția Laplace (numită și „Laplace Smoothing”).

[16] Faceți calculele necesare și verificați dacă ajungeți la aceeași estimare.
[17] Faceți calculele necesare și verificați dacă ajungeți la aceeași estimare.

Să presupunem că setul de date include atributul de interes (de prezis) cu valorile pleacă / rămâne, respectiv o serie de atribute obișnuite, unul dintre acestea luând valorile da / nu. Dacă setul de date nu conține niciun caz pentru una (oricare) dintre combinațiile celor două atribute (pleacă + da, pleacă + nu, rămâne + da, rămâne + nu), probabilitatea condiționată asociată acelei combinații este zero. Când înmulțim probabilitatea zero cu alte probabilități (oricare ar fi valorile acestora), rezultatul este zero, prin urmare probabilitatea prezisă asociată acelei combinații va fi întotdeauna zero.

În cazul unor softuri (Weka de exemplu), corecția Laplace constă în adăugarea unui caz la fiecare combinație dintre clasele unui atribut obișnuit și clasele atributul de interes. În RapidMiner corecția Laplace adaugă o fracțiune de caz la numărul de cazuri asociat fiecărei combinații dintre atributul de interes (label) și predictori. Formula utilizată pentru a calcula fracția de caz de adăugat este

$$P_{lap,k}(B|A) = \frac{c(A_i, B_j) + k}{c(A) + k * n}$$

unde:

$c(A_i,B_j)$ = numărul de cazuri care aparțin simultan claselor i a atributului A și j a atributului B;
$c(A_j)$ = numărul de cazuri care aparțin clasei j a atributului A;
k = valoarea parametrului de corecție (smoothing parameter);
n = numărul claselor atributului B.

Așa se explică faptul că la fiecare atribut categorial apare suplimentar o clasă necunoscută (unknown). Procedând astfel, fiecare combinație va avea un număr non-nul de cazuri. În majoritatea situațiilor, aplicarea acestei corecții are un efect neglijabil asupra probabilităților estimate, deci și a rezultatelor modelului. În RapidMiner corecția poate fi aplicată prin selectarea parametrului „laplace correction" asociat operatorului Naive Bayes (parametrul este selectat implicit). Pentru exemplul anterior, dacă aplicăm corecția Laplace, rezultatele se schimbă foarte puțin Exemplul 6.2-2.

Exemplul 6.2-2. Rezultatele analizei anterioare în urma aplicării corecției Laplace

		Likelihood		Probabilitate	
Departament	**Poziție**	**Nu pleacă**	**Pleacă**	**Nu pleacă**	**Pleacă**
producție	conducere	0.118	0.050	70.4%	29.6%
producție	execuție	0.427	0.098	81.3%	18.7%
vânzări	conducere	0.033	0.050	39.6%	60.4%
vânzări	execuție	0.118	0.098	54.6%	45.4%

Dat fiind faptul că rata (anuală) de părăsire a unei companii este în general mică, probabilitatea ca un angajat oarecare să plece din companie este tot mică. Probabilitățile asociate diferitelor sub-categorii de angajați trebuie evaluate (și) în relație cu această probabilitate generală, nu doar ca valori individuale. Prin urmare, în cazul exemplului nostru, comparăm probabilitatea de părăsire a companiei în cazul unui angajat pe poziție de execuție din producție (18.8%) cu probabilitatea generală de părăsire a companiei (6/20, adică 30%). Observăm că prima este semnificativ mai mică, deci, cei care pleacă au mai degrabă alte combinații de caracteristici decât „poziție de execuție în departamentul de producție".

Un exemplu didactic – Gaussian Naive Bayes

În exemplul anterior cei doi predictori erau de tip binominal. Firesc, ne întrebăm cum putem estima un model care conține și unul sau mai mulți predictori de tip metric. Să presupunem că setul nostru de date include, pe lângă atributele Departament și Poziție, și un atribut numit Vârsta care indică vârsta în ani împliniți a angajaților. Setul de date utilizat pentru exemplele din acest manual conține o astfel de variabilă – Age. La nivelul eșantionului de instruire variabila respectivă are media 37 și abaterea standard 9. Am observat anterior (Graficul 1.3-6) că distribuția acestei variabile este relativ Normală, atât la nivelul întregului set de date cât și la nivelul sub-seturilor definite de atributul Pleacă, deci vom asuma că aceasta este și distribuția teoretică a variabilei Vârsta.[18] O distribuție Normală se mai numește și Gaussiană, de unde și numele de Gaussian Naive Bayes. Funcția de densitate a probabilității (probability density function = PDF) asociată unei distribuții statistice de tip Gaussian are forma:

$$f(x) = \frac{e^{\frac{-(x-\mu)^2}{2\sigma^2}}}{\sqrt{2\pi} * \sigma}$$

unde,
x = valoarea variabilei X
μ = media variabilei X
σ = abaterea standard a variabilei X
σ^2 = varianța variabilei X
e și π = constante matematice

Considerând valorile observate ale variabilei Vârsta algoritmul calculează media și abaterea standard pentru fiecare dintre clasele Pleacă și Rămâne. Firesc, media de vârstă este puțin mai mică în cazul celor care părăsesc compania față de cei care

[18] Dacă predictorii nu au o distribuție normală e de preferat să folosim operatorul Naive Bayes Kernel. Acesta nu asumă că distribuția teoretică este Gausiană ci încearcă să estimeze o curbă netezită (smoothed) a valorilor observate relativ la fiecare predictor. Se obține astfel un model mai bun, deci predicții mai bune. Pe de altă parte, estimarea modelului durează mai mult iar șansele de supra-ajustare (overfitting) cresc.

rămân (33.35 vs 37.97), iar abaterile standard sunt similare (9.60 vs 9.07). Folosind aceste valori și asumând că valorile variabilei (vârsta) sunt distribuite normal, Naive Bayes definește două funcții de densitate a probabilității, câte una pentru fiecare grup (Pleacă vs Rămâne) (Graficul 6.2-1).[19] Pe baza funcțiilor calculăm verosimilitatea (likelihood) ca un angajat să plece, respectiv să nu plece în funcție de vârsta lui. De exemplu, dacă angajatul are 30 de ani, verosimilitatea să plece este 0.0391, iar verosimilitatea să rămână este 0.0299.[20] Am obținut aceste valori înlocuind termenii cunoscuți în funcția de densitate a probabilității asociată unei distribuții de tip Gaussian:

$$f(x) = \frac{e^{\frac{-(x-\mu)^2}{2\sigma^2}}}{\sqrt{2\pi} * \sigma}$$

Rămâne + Vârsta = 30 ani

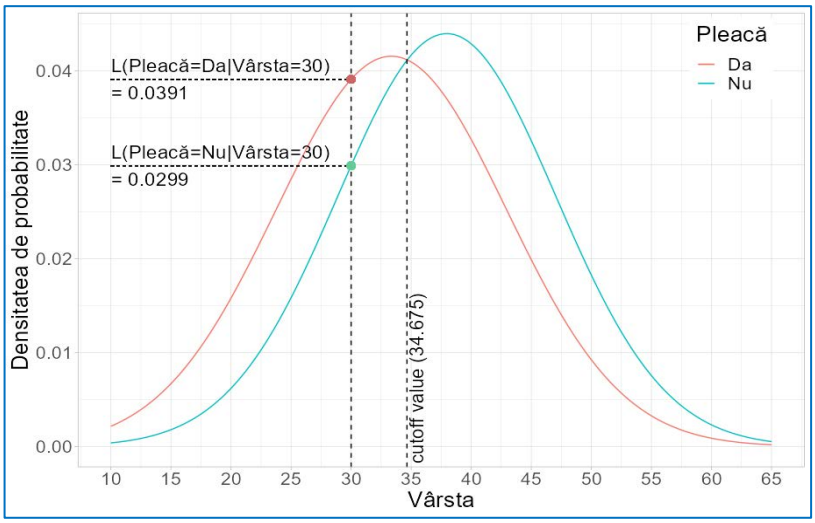
$$f(30) = \frac{e^{\frac{-(30-37.97)^2}{2*9.07^2}}}{\sqrt{2\pi} * 9.07} = 0.0299$$

Pleacă + Vârsta = 30 ani

$$f(30) = \frac{e^{\frac{-(30-33.35)^2}{2*9.6^2}}}{\sqrt{2\pi} * 9.6} = 0.0391$$

Deoarece verosimilitatea este mai mare în cazul clasei Pleacă, prezicem că angajatul va pleca. Pe grafic am indicat și valoarea vârstei (cutoff value) care separă cele două distribuții. Astfel, considerând doar vârsta, prezicem că angajații care au mai puțin de 34.675 ani vor pleca, iar cei care au 34.675 ani sau mai mult vor rămâne.

Graficul 6.2-1. Funcțiile de densitate a probabilității asociate variabilei Vârsta



[19] Graficul e realizat în R. RapidMiner (operatorul Naive Bayes) produce implicit graficele asociate celor două distribuții, separat pentru fiecare predictor, dar nu permite editarea acestora (vezi Results – SimpleDistribution – Simple Charts).

[20] Dacă angajatul are 30 de ani, care sunt șansele (likelihood) ca distribuția normală asociată clasei Pleacă, să aibă media 33.35 și abaterea standard 9.6, respectiv 37.97 și 9.07 pentru clasa Rămâne.

Pentru a calcula probabilitatea de a pleca, respectiv a rămâne, a unui angajat cu vârsta de 30 ani procedăm astfel: înmulțim probabilitatea clasei (pleacă, respectiv nu pleacă) cu valoarea likelihood asociată unui angajat care aparține acelei clase și are vârsta de 30 ani (vezi valorile calculate mai sus); obținem astfel câte o nouă valoare likelihood pentru fiecare clasă (L_{Da} / L_{Nu}); pentru a obține probabilitățile, împărțim fiecare dintre valorile likelihood finale la suma lor (Tabelul 6.2-1).

Tabelul 6.2-1. Calcularea valorilor likelihood și a probabilităților în cazul unui angajat de 30 ani

Valori likelihood pentru **clasa Da (Pleacă)**	
L(Vârstă=30|Pleacă=Da)	0.0391
P(Pleacă=Da)	0.1613
Likelihood final (L_{Da})	0.0063

]×

Valori likelihood pentru **clasa Nu (Rămâne)**	
L(Vârstă=30|Pleacă=Nu)	0.0299
P(Pleacă=Nu)	0.8387
Likelihood final (L_{Nu})	0.0251

]×

$Prob(Da) = LL_{Da} / (LL_{Nu} + LL_{Da})$
$Prob(Da) = 0.0063 / (0.0251 + 0.0063)$
$Prob(Da) = 0.201$ sau 20.1%

$Prob(Nu) = LL_{Nu} / (LL_{Nu} + LL_{Da})$
$Prob(Nu) = 0.0063 / (0.0251 + 0.0063)$
$Prob(Nu) = 0.799$ sau 79.9%

Dacă avem mai mulți predictori de tip metric, aplicăm metoda descrisă anterior în cazul fiecărui predictor, apoi înmulțim între ele valorile likelihood asociate fiecărei clase, apoi înmulțim rezultatul cu probabilitatea generală asociată acelei clase. Să presupunem că pe lângă Vârstă (Age), mai avem două variabile metrice, Salariu (MonthlyIncome) și Vechimea (YearsAtCompany). Procesul prin care am ajuns la aceste probabilități este descris pas cu pas în Tabelul 6.2-2.

Prima dată calculăm media și abaterea standard pentru fiecare predictor, separat pe clasele atributului de interes (Pleacă vs Nu pleacă). Apoi calculăm funcțiile teoretice asociate fiecărui predictor pentru fiecare clasă. În continuare, pentru fiecare caz din setul de date ne uităm la valoarea pe care o ia primul predictor și clasa la care aparține acel caz (Vârsta + Nu pleacă, de exemplu), alegem funcția corespunzătoare și calculăm valoarea likelihood asociată acestei combinații. De exemplu, dacă angajatul are o vechime de un an și nu a plecat din companie, valoarea likelihood asociată este 0.03796; dacă o vechime de un an și a plecat, valoarea likelihood asociată este 0.05515. Procedăm la fel pentru toate cazurile și toți predictorii. În final vom avea pentru fiecare caz șase valori likelihood, câte trei pentru fiecare clasă, plus cele două probabilități generale.

Tabelul 6.2-2. Un alt exemplu de calculare a valorilor likelihood și a probabilităților

	Rămâne		Pleacă	
Variabila	**Media (μ)**	**Std (σ)**	**Media (μ)**	**Std (σ)**
Vârstă	37.97	9.07	33.35	9.60
Venit	6988.32	4941.15	4791.55	3783.46
Vechime	7.37	6.24	5.04	5.56

Std = abaterea standard

		Likelihood (L)	
Variabila	**Valori***	**Nu pleacă**	**Pleacă**
Vârstă	30	0.02991	0.03911
Venit	1,274	0.00004	0.00007
Vechime	1	0.03796	0.05515
Probabilitate generală		0.83868	0.16132
Likelihood final		0.000000039	0.000000024
Probabilitatea prezisă		0.623 62.3%	0.377 37.7%

** valorile asociate unui caz ipotetic*

Likelihood (Vechime=1|Pleacă) =

$$f(x) = \frac{e^{\frac{-(x-\mu)^2}{2\sigma^2}}}{\sqrt{2\pi} * \sigma} \rightarrow f(1) = \frac{e^{\frac{-(1-5.04)^2}{2*5.56^2}}}{\sqrt{2\pi} * 5.56} = 0.05515$$

Likelihood final (Pleacă) = L(Pleacă) * L(Vârstă=30|Pleacă) * L(Venit=1274|Pleacă) * L(Vechime=1|Pleacă) = 0.03911 * 0.00007 * 0.05515 * 0.16132 = 0.000000024

Probabilitatea prezisă (Pleacă) = L(Pleacă) / (L(Pleacă) + L(Nu pleacă)) = $39*10^{-9}$ / ($24*10^{-9}$ + $39*10^{-9}$) = 0.377 = 37.7%

Uneori valorile likelihood sunt extrem de mici (vezi de exemplu valorile asociate vârstei de 60 ani din Graficul 6.2-1) ceea ce, în cazul înmulțirii, pune probleme de calcul computerelor. Pentru a evita această problemă se procedează astfel: pentru fiecare clasă separat se logaritmează produsul termenilor și apoi se adună logaritmii termenilor obținându-se astfel valorile Log Likelihood (LL). Valoarea LL cea mai mare indică clasa la care aparține angajatul cu acele caracteristici. Deoarece LL asociat clasei Rămâne este mai mare, vom prezice că angajatul rămâne. În cazul de față, pentru clasa Pleacă, valoarea LL este

ln(0.03911 * 0.00007 * 0.05515 * 0.16132) =
ln(0.03911) + ln(0.00007) + ln(0.05515) + ln(0.16132) = -17.53045625

iar pentru clasa Rămâne LL este

ln(0.02991 * 0.00004 * 0.03796 * 0.83868) =
ln(0.02991) + ln(0.00004) + ln(0.03796) + ln(0. 83868) = -17.08334187

Avantaje și dezavantaje

La finalul acestei sub-capitol prezentăm un tabel sintetic cu principalele avantaje și dezavantaje ale algoritmului Naive Bayes (Tabelul 6.2-3):

Tabelul 6.2-3. Avantajele și dezavantajele algoritmului Naive Bayes

Avantaje	Dezavantaje
Performanță predictivă bună chiar și atunci când predictorii nu sunt independenți între ei	Nu ia în calcul relațiile dintre predictori (asumă independența predictorilor)
Acceptă orice număr de predictori, indiferent de tipul acestora (categoriali și metrici); performează relativ mai bine atunci când numărul predictorilor este mare	Nu poate învăța situațiile în care doi sau mai mulți predictori interacționează; de exemplu, un angajat care pleacă de la locul de muncă atunci când sunt îndeplinite simultan două condiții (nașterea unui copil + program fix obligatoriu)
Modelul de predicție este calculat rapid	
Predicția este rapidă	
Modelele de predicție sunt ușor interpretabile	
Nu are hiper-parametri	Lucrează mai greu cu predictori metrici
Robustețe (estimările sunt puțin influențate de predictori irelevanți și date cu erori)	Probabilitățile estimate sunt uneori relativ distorsionate (biased)

Sursa: (Moreira et al., 2019, p. 214; Nisbet et al., 2018, p. 184)

6.3. Exemple de utilizare a Naive Bayes în RapidMiner

Un model simplu

În cele ce urmează vom prezenta și discuta un exemplu simplu de analiză în care este folosit operatorul Naive Bayes. Pentru a facilita înțelegerea, vom prezenta calculele în paralel, manual vs. RapidMiner Studio.[21] Setul de date utilizat este „employee_attrition", rolul atributelor fiind definit direct în acesta. Pentru acest exemplu am păstrat doar atributul special (label) de interes (Attrition) și trei atribute obișnuite (Department, JobSatisfaction și WorkLifeBalance). Numărul de cazuri pentru fiecare combinație de clase apare în Tabelul 6.3-1 (operatorul Naive Bayes din RapidMiner calculează aceste frecvențe, dar nu le prezintă la rezultate).

[21] Exemplul realizat în RapidMiner Studio este inclus în arhiva atașată acestui volum.

Tabelul 6.3-1. Numărul angajaților care (nu) au părăsit compania în funcție de departament, satisfacția în muncă și echilibrul muncă – viață personală (setul de date de instruire)

Depart.	Nu	Da	Satisfacția în muncă	Nu	Da	Echilibrul muncă - viață	Nu	Da	Nu	Da
R&D	579	96	Mică	158	46	Prost	39	17	863	166
Vânzări	248	63	Medie	164	31	Bun	201	38		
HR	36	7	Mare	254	56	Foarte bun	528	93		
			Foarte mare	287	33	Excelent	95	18		

Exemplu: Dintre angajații departamentului Vânzări (Sales), 248 au rămas în companie și 63 au plecat.

Pe baza tabelului de frecvențe, sunt calculate și prezentate în formă grafică și tabelară probabilitățile condiționate („Dacă un angajat pleacă, care e probabilitatea ca acesta să fie dintr-o anumită categorie / clasă a unui anumit atribut?" și „Dacă un angajat nu pleacă, care e probabilitatea ca acesta să fie dintr-o anumită categorie / clasă a unui anumit atribut?"), respectiv probabilitățile claselor atributului de interes Attrition (Figura 6.3-1). Astfel, probabilitatea de a pleca din companie a unui angajat oarecare este 0.161 (166/1029), iar probabilitatea să rămână este 0.839 (863/1029). Dacă angajatul pleacă, probabilitatea să fie din departamentul Vânzări e 0.380 (63/166, unde 63 e numărul de angajați din departamentul Vânzări care au plecat, iar 166 e numărul total de angajați care au plecat), din departamentul R&D 0.578 (96/166), din departamentul HR 0.042 (7/166). Dacă angajatul pleacă, probabilitatea să aibă un nivel Low al JobSatisfaction este 0.277, 0.187 Medium, 0.337 High și 0.199 Very high. Relativ la atributul WorkLifeBalance probabilitățile de a pleca sunt 0.102 (Bad), 0.229 (Good), 0.560 (Very good) și 0.108 (Excellent). Desigur, aceste probabilități condiționate (modelul) sunt calculate pe setul de date de instruire. Ulterior, modelul este aplicat altui set de date (de validare), adică sunt realizate predicțiile și comparate cu realitatea.[22] Pentru calcularea predicțiilor, algoritmul folosește probabilitățile condiționate calculate pe setul de date de instruire. Prima dată calculează valorile likelihood înmulțind probabilitățile condiționate între ele, separat pentru fiecare clasă a atributului de interes, și apoi, pe baza acestora calculează probabilitățile prezise. Setul de date rezultat va conține suplimentar trei[23] atribute speciale noi: predicția și încrederea în aceasta relativ la

[22] Predicțiile pot fi estimate și pentru același set de date (setul de instruire). Dacă facem acest lucru, calitatea predicției / modelului va fi supra-estimată, adică, atunci când vom aplica modelul unor angajați noi, despre care nu știm anterior dacă vor pleca sau nu, vom face predicții incorecte într-o pondere mai mare decât cea așteptată conform modelului estimat.

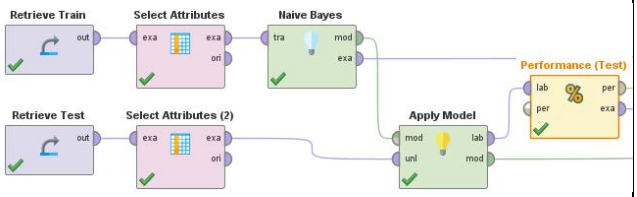
[23] Dacă numărul claselor prezise e mai mare, vom avea mai multe atribute denumite confidence (tot atâtea câte clase sunt).

fiecare clasă (da/nu în acest caz). Valorile care apar la încredere (confidence) reprezintă probabilitățile prezise. Clasa care are asociată probabilitatea cea mai mare va fi și clasa prezisă (atributul predicted(Attrition)).

Figura 6.3-1. Naive Bayes: Un model simplu în RapidMiner

Pasul 1:

Realizăm procesul din imaginea alăturată. Setul de date este „employee_attrition" (subseturile training și validation). La Naive Bayes lăsăm bifat[24] parametrul „laplace correction". Rulăm procesul.



Rezultate (setul de instruire):

Seturile de date conțin doar atributele selectate anterior (în imagine apar primele 5 cazuri din setul de instruire). Distribuția statistică a atributului de interes – Attrition – este 0.161 pentru clasa Yes (a plecat) și 0.839 pentru clasa No (nu a plecat).

Probabilitățile condiționate apar sub formă grafică (Simple Charts) și tabelară (Distribution Table). De exemplu, raportat la angajații care au plecat (166), probabilitatea ca aceștia să fie din departamentul Sales este 0.380 (63/166, unde 63 e numărul de angajați din departamentul Sales care au plecat); similar, raportat la angajații care nu au plecat (801), probabilitatea să fie din Sales este 0.287 (248/863).[25]

Id	Attrition	Department	JobSatisfa...	WorkLifeB
3	Yes	Research & ...	High	Very good
4	No	Research & ...	High	Very good
5	No	Research & ...	Medium	Very good
7	No	Research & ...	Low	Good
11	No	Research & ...	Medium	Very good

SimpleDistribution

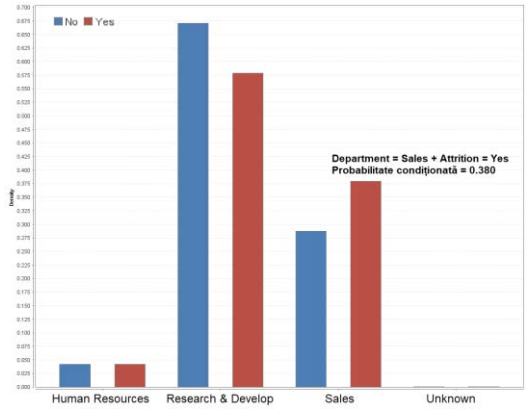
```
Distribution model for label attribute Attrit

Class Yes (0.161)
3 distributions

Class No (0.839)
3 distributions
```

[24] Chiar dacă setul de date de instruire nu conține combinații de atribute fără cazuri, bifarea acestui parametru aplică „laplace correction", prin urmare vor apărea diferențe foarte mici între calculele realizate de soft și calculele manuale fără corecție (rotunjirile pot contribui și ele la aceste diferențe).

[25] Când realizează calculele, algoritmul adaugă automat la fiecare frecvență (count) o fracțiune de caz. Aceste cazuri vor forma o clasă nouă numită „unknown". Probabilitățile condiționate asociate acestei clase vor fi diferite de zero doar în situația în care aplicăm corecția Laplace. În cazul de față aceste probabilități au valori mai mici de 10^{-5}, deci influențează estimările extrem de puțin.



Attribute	Parameter	No	Yes
Department	value=Sales	0.287	0.380
Department	value=Research &	0.671	0.578
Department	value=Human Res	0.042	0.042
Department	value=unknown	0.000	0.000
WorkLife	value=Bad	0.045	0.102
WorkLife	value=Very good	0.612	0.560
WorkLife	value=Good	0.233	0.229
WorkLife	value=Excellent	0.110	0.108
WorkLife	value=unknown	0.000	0.000
JobSat	value=Very high	0.333	0.199
JobSat	value=Medium	0.190	0.187
JobSat	value=High	0.294	0.337
JobSat	value=Low	0.183	0.277
JobSat	value=unknown	0.000	0.000

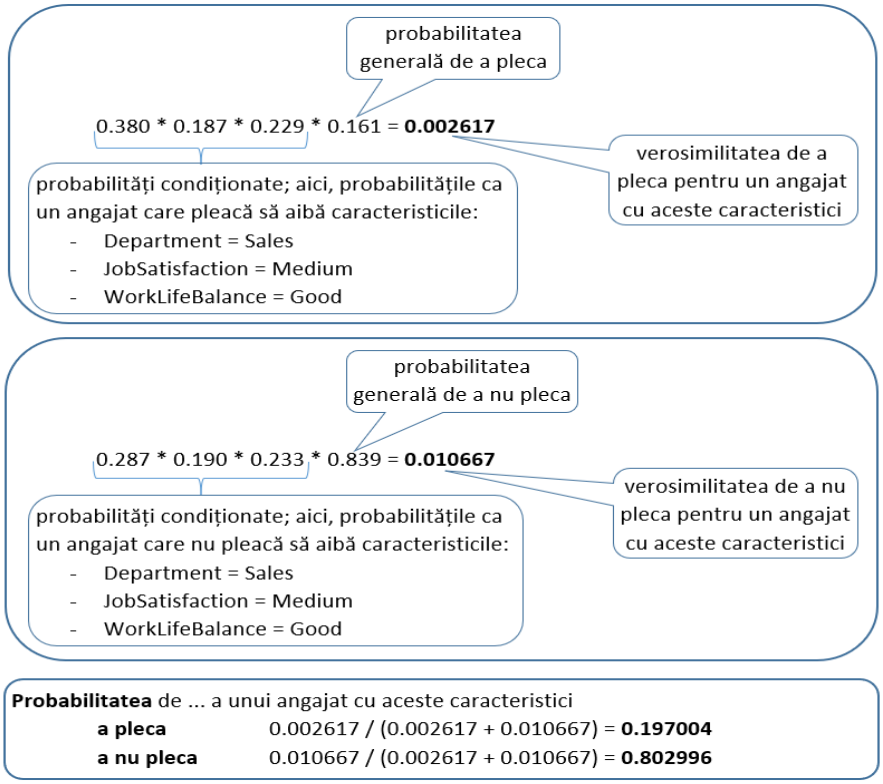
Rezultate (setul de validare):

Modelul produce predicțiile din imagine (primele 6 cazuri din setul de validare și două cazuri de interes în contextul acestui exemplu). Coloana Attrition prezintă situația reală, iar coloana „prediction(Attrition)" predicția realizată de model. Observăm că unele predicții sunt greșite (în imagine, cele cu Id 1, 248 și 650). Probabilitățile asociate predicțiilor apar pe coloanele „confidence" (șansele sunt calculate, dar nu sunt prezentate). Pentru fiecare caz este selectată ca predicție clasa care, pe coloana confidence, are cea mai mare probabilitate. Deoarece toate probabilitățile asociate coloanei „confidence(No)" sunt mai mari de 0.5 (50%), toate predicțiile sunt No (angajatul nu pleacă). Prin urmare modelul are o acuratețe a clasificării identică cu modelul de șansă (83.9%). Aceasta nu înseamnă însă că toți angajații au asociate aceleași probabilități de a nu pleca. Cea mai mare probabilitate estimată de a rămâne este 91.7% (R&D + Very good + Very high), iar cea mai mică 53.4% (Sales + Bad + Low). Desigur, dacă scădem pragul sub 50%, modelul va prezice că o parte din angajați pleacă.

Id	Attrition	prediction(Attrition)	confidence(No)	confidence(Yes)	Department	WorkLifeBal...	JobSatisfac...
1	Yes	No	0.744	0.256	Sales	Bad	Very high
2	No	No	0.870	0.130	Research & ...	Very good	Medium
8	No	No	0.911	0.089	Research & ...	Good	Very high
11	No	No	0.852	0.148	Research & ...	Very good	High
12	No	No	0.852	0.148	Research & ...	Very good	High
13	No	No	0.843	0.157	Research & ...	Good	High
...							
248	Yes	No	0.803	0.197	Sales	Good	Medium
...							
650	Yes	No	0.534	0.466	Sales	Bad	Low
1560	No	No	0.534	0.466	Sales	Bad	Low
...							
2009	No	No	0.917	0.083	Research & ...	Very good	Very high

Softul nu prezintă, dar putem calcula manual verosimilitatea ca un anumit tip de angajat să plece înmulțind probabilitățile condiționate între ele și rezultatul cu probabilitatea generală de a pleca. De exemplu, pentru un angajat din departamentul Sales, cu un grad mediu de satisfacție relativ la muncă și care apreciază că în cazul lui echilibrul muncă-viață este bun (vezi cazul cu Id 248), verosimilitatea de a pleca este egală cu 0.002617; pentru același tip de caracteristici, verosimilitatea de a nu pleca este 0.010667. Calculele necesare pentru a ajunge la aceste valori sunt prezentate în Figura 6.3-2. Folosind aceste valori likelihood, putem calcula manual probabilitățile de a (nu) pleca pentru un angajat cu caracteristicile menționate anterior. Acestea se calculează prin raportarea verosimilității asociate unei clase la suma verosimilităților celor două clase. Astfel, probabilitatea ca un astfel de angajat să plece este 19.7%, iar să rămână 80.3% (Figura 6.3-2). Observăm că probabilitățile calculate manual sunt aproape identice cu probabilitățile calculate de algoritm (diferențele sunt mai mici de 0.1%; ele apar ca urmare a aplicării corecției Laplace și a rotunjirilor). În final, observăm că probabilitatea de a părăsi compania a unui angajat cu aceste caracteristici (Sales, grad mediu de satisfacție relativ la muncă, echilibrul muncă-viață bun) este puțin mai mare comparativ cu probabilitatea generală de părăsire a companiei (16.1%).

Figura 6.3-2. Naive Bayes: Un exemplu de calcul a verosimilităților și probabilităților

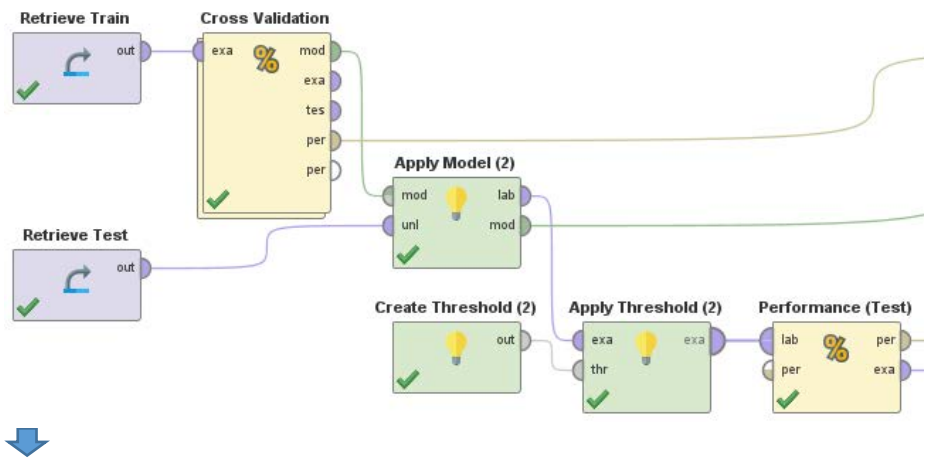


Un model relativ mai complex

În acest exemplu folosim același set de date dar, de această dată, includem toate atributele obișnuite disponibile. Și în acest caz folosim două seturi de date, unul de instruire (pe acesta îl folosim pentru a construi / estima modelul) și unul de testare. Simplu spus, construim modelul de predicție pe setul de instruire (această fază include și validarea încrucișată) și apoi îl testăm pe setul de test. Calitatea estimată a modelului / predicției va fi cea observată în cazul setului de test. Atunci când vom pune modelul de predicție în producție (vom realiza predicții în cazul angajaților noi, cei despre care nu știm dacă vor rămâne sau pleca), ne așteptăm să aibă o performanță similară cu cea observată în cazul setului de test. Procesul și rezultatele obținute apar în Figura 6.3-3.

Figura 6.3-3. Naive Bayes: Un model relativ mai complex în RapidMiner

Pasul 1:
Realizăm procesul din imaginea de mai jos. În folderul asociat volumului am inclus două versiuni ale acestui proces: o versiune mai simplă, fără setarea pragului de probabilitate (acesta este stabilit implicit la 0.5) și o versiune care oferă posibilitatea setării pragului de probabilitate. Setul de date utilizat pentru instruirea modelului este „employee_attrition_training_validation”. Setul utilizat pentru testarea modelului este „employee_attrition_testing”. Includem operatorul Naive Bayes cu setările originale. În cazul modelului 2 includem operatorii care ne ajută să setăm pragul de probabilitate dorit (Create Threshold și Apply Threshold). Includem operatorul necesar pentru aplicarea modelului pe setul de date de testare, respectiv operatorul care calculează măsurile de performanță. Realizăm conexiunile și rulăm procesul. La unul dintre modele pragul de probabilitate este setat implicit la 0.5 (prima clasă este No, a doua Yes), prin urmare modelul va prezice că angajații în cazul cărora valoarea atributului „confidence(Yes)” va fi mai mare de 0.5 vor pleca, iar restul vor rămâne. La celălalt model pragul este stabilit manual la 0.3.

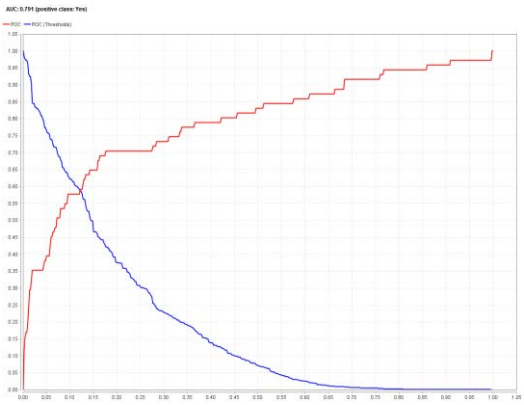


Rezultate (setul de testare):
Relativ la setul de testare, modelul produce predicțiile din imagine (am inclus primele 5 cazuri și alte trei de interes în contextul acestui exemplu). Coloana Attrition prezintă situația reală, iar coloana „prediction(Attrition)" predicția realizată de model. Observăm că unele predicții sunt greșite (în imagine, doar cea cu Id 11). Probabilitățile asociate predicțiilor apar pe coloanele „confidence". Pentru fiecare caz este selectată ca predicție clasa care are cea mai mare probabilitate (confidence).

Row No.	Id	Attrition	prediction(Attrition)	confidence(No)	confidence(Yes)
1	1	Yes	Yes	0.284	0.716
2	2	No	No	0.991	0.009
3	8	No	No	0.641	0.359
4	11	No	No	0.594	0.406
5	12	No	No	0.900	0.100
...					
11	39	No	Yes	0.412	0.588
...					
57	259	No	No	1.000	0.000
...					
41	167	Yes	Yes	0.002	0.998
...					

Probabilitățile condiționate asociate fiecărui predictor sunt prezentate în secțiunea de rezultate „SimpleDistribution (Naive Bayes)", tabul „Distribution Table" (aici am prezentat doar o selecție din acest tabel; observăm că în cazul atributelor numerice apar mediile și abaterile standard condiționate, nu probabilitățile condiționate). Tabul Description prezintă probabilitățile asociate celor două clase ale atributului de interes (No = 0.839, Yes = 0.161). Observăm că modelul de predicție are o performanță generală foarte bună (AUC = 0.791).

Atribut	Parametru	Nu	Da
Gender	value=Female	0.403244	0.367469
Gender	value=Male	0.596755	0.632525
Gender	value=unknown	0.000001	0.000006
Age	mean	37.966396	33.349398
Age	standard deviation	9.072793	9.597896
Education	value=College	0.183082	0.216866
Education	value=Below College	0.120510	0.102412
Education	value=Master	0.289686	0.240961
Education	value=Bachelor	0.373116	0.421678
Education	value=Doctor	0.033605	0.018078
Education	value=unknown	0.000001	0.000006
...	...	...	...



Secțiunea PerformanceVector conține matricea de confuzie și diferite măsuri ale performanței clasificării (cele selectate), toate calculate relativ la ambele seturi de date (instruire și testare). Observăm că acuratețea predicției este de fiecare dată peste 80%, ceea ce indică șanse mari de generalizare a modelului de predicție. Chiar dacă de obicei calitatea clasificării este mai bună în cazul setului de instruire comparativ cu cel de test, de această dată este invers. Comparativ cu modelul care a inclus doar doi predictori, deși modelul cu toți predictorii are o acuratețe a clasificării puțin mai mică, reușește că clasifice corect o parte relativ mai mare a cazurilor de angajați care pleacă (ceea ce e foarte util din punct de vedere practic). În continuare, modelul prezice mai bine situația în care angajații nu pleacă (precizia = 93%, reamintirea = 86/82%) comparativ cu situația în care angajații pleacă (precizia = 47/42%, reamintirea = 65/69%).

accuracy: 77.65% +/- 4.29% (micro average: 77.65%)

	true No	true Yes	class precision
pred. No	693	60	92.03%
pred. Yes	170	106	38.41%
class recall	80.30%	63.86%	

instruire, prag 50%

accuracy: 77.26% +/- 4.63% (micro average: 77.26%)

	true No	true Yes	class precision
pred. No	681	52	92.91%
pred. Yes	182	114	38.51%
class recall	78.91%	68.67%	

instruire, prag 30%

accuracy: 82.54%

	true No	true Yes	class precision
pred. No	318	25	92.71%
pred. Yes	52	46	46.94%
class recall	85.95%	64.79%	

testare, prag 50%

accuracy: 79.82%

	true No	true Yes	class precision
pred. No	303	22	93.23%
pred. Yes	67	49	42.24%
class recall	81.89%	69.01%	

testare, prag 30%

7. CEI MAI APROPIAȚI VECINI (K NEAREST NEIGHBOR, K-NN)

Nume operator	**k-NN**
Categoria majoră	învățare asistată (supervised learning)
Tip model	clasificare (classification)[1]
Grup	algoritm „leneș”
Atribut dependent	categorial
Atribute independente	numerice, categoriale
Gestionează valorile lipsă	parțial (ignoră cazurile cu cel puțin o valoare lipsă la un atribut)
Distorsiune (bias)	mică pentru k mic mare pentru k mare
Varianță (variance)	mare pentru k mic mică pentru k mare

Bibliografie utilizată și recomandată:

- Data mining: a tutorial-based primer (Roiger, 2017, Chapter 5.6)
- Data Science: Concepts and Practice (Kotu & Deshpande, 2019, Chapter 4.3)
- Machine Learning for Business Analytics: Concepts, Techniques and Applications in RapidMiner (Shmueli et al., 2023, Chapter 7)

k-NN este unul dintre cei mai simpli algoritmi de clasificare. Acronimul k-NN vine de la „k-Nearest Neighbors” sugerând faptul că algoritmul caută „cei mai apropiați k vecini” ai unui caz sau, altfel spus, caută „k observații similare” relativ la cazul de interes (cel pentru care dorim să prezicem clasa la care aparține; de exemplu, pleacă / nu pleacă din companie în următorul an). k-NN clasifică observațiile / cazurile noi în funcție de clasele la care aparțin vecinii acelor cazuri (cazurile din proximitate). Simplu spus, pentru a prezice clasa din care face parte un caz X ne vom uita la ce clasă aparțin cele mai multe dintre cazurile din proximitatea lui X (vecinii cazului X).

[1] Poate fi folosit și pentru probleme de regresie, adică atunci când variabila dependentă este metrică.

Clasa dominantă va fi și clasa prezisă a lui X (Attewell & Monaghan, 2015, p. 134). Proximitatea sau distanța dintre cazuri este stabilită în funcție de unul sau mai multe atribute asociate cazurilor. k-NN poate fi folosit pentru probleme de clasificare (binară sau multiplă), respectiv regresie (predicția unor atribute metrice).[2] În contextul acestui volum vom discuta doar despre predicția unor atribute categoriale de tip binominal.

7.1. Logica și pașii algoritmului

În faza de instruire, algoritmul k-NN doar încarcă setul de date în memorie, fără să realizeze niciun fel de calcule, adică fără să estimeze un model de predicție (faza de instruire, adică estimarea unui model, lipsește). Toate calculele (distanțele dintre cazuri și ordinea lor relativă) sunt realizate în faza de predicție, adică în momentul în care algoritmul stabilește care sunt grupurile din care fac parte, cel mai probabil, noile cazuri. Datorită acestei abordări, k-NN este etichetat ca un algoritm „leneș". În procesul de predicție, algoritmul k-NN parcurge următorii trei pași (vezi și Figura 7.1-1):

1. calcularea distanței dintre cazul pe care dorim să-l prezicem și fiecare dintre cazurile pentru care știm valoarea luată în cazul variabilei dependente (clasa la care aparține);
2. identificarea vecinilor cazului care trebuie prezis; numărul vecinilor este egal cu k (valoarea lui k este aleasă de analist; desigur, de obicei, sunt testate și comparate modele cu diferite valori ale lui k);
3. stabilirea clasei la care aparține cazul de prezis în urma identificării clasei majoritare la care aparțin vecinii acestuia (avem posibilitatea să ponderăm votul în funcție de distanța dintre fiecare vecin și cazul de prezis).[3]

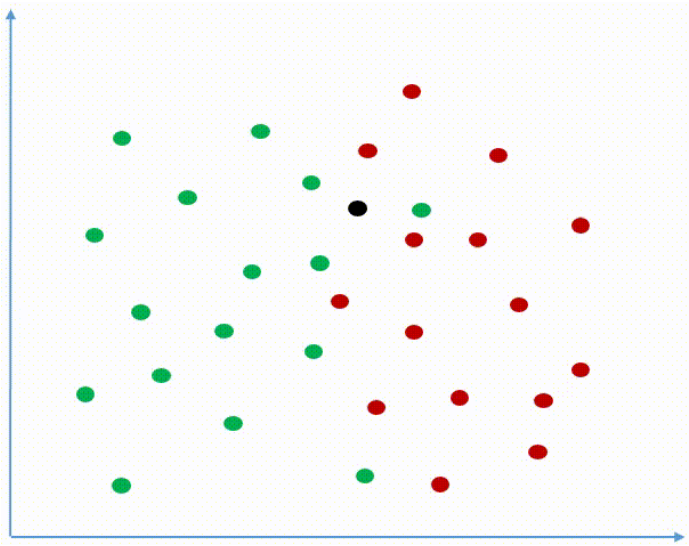
[2] Există o mulțime de variante ale k-NN, fiecare dintre acestea încercând să rezolve anume probleme ale algoritmului original: A-KNN (Adaptative), LA-KNN (Locally adaptative), F-KNN (Fuzzy), kM-KNN (k-means), W-KNN (Weighted), H-KNN (Hassanat), GDM-KNN (Generalised Mean Distance), M-KNN (Mutual), EA-KNN (Ensemble Approach) (Uddin et al., 2022).

[3] În cazul în care variabila dependentă este metrică, predicția fi egală cu media valorilor „vecinilor". Se poate calcula media aritmetică sau media ponderată cu distanța dintre cazuri.

Figura 7.1-1. Pașii algoritmului k-NN

Exemplul 1

Exemplul 2



Sursa: K Nearest Neighbor Classification – Animated Explanation for Beginners

Înainte de a utiliza algoritmul k-NN pentru realizarea unei predicții, trebuie să răspundem la următoarele patru întrebări (Attewell & Monaghan, 2015; Kotu & Deshpande, 2019; Moreira et al., 2019; Shmueli et al., 2017; Wendler & Gröttrup, 2021):

- Ce predictori / atribute utilizăm pentru calcularea distanțelor dintre cazuri?
- Câți vecini avem în vedere (valoarea lui k)?
- Cum calculăm distanța dintre cazuri (ce măsură folosim)?
- Cum stabilim care este clasa prezisă?

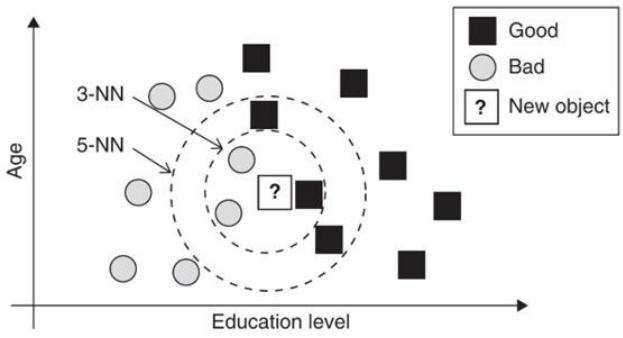
Ce atribute utilizăm pentru calcularea distanțelor dintre cazuri?

Pentru o calitate bună a predicției, este necesar ca predictorii incluși să fie asociați cât mai puternic cu variabila dependentă (prezisă). De asemenea, numărul predictorilor contează mult. Relația dintre numărul predictorilor și calitatea predicției este una în formă de clopot. Simplu spus, dacă numărul predictorilor este prea mic sau prea mare, calitatea predicției va fi, cel mai probabil, relativ mai redusă. La prima vedere, relația negativă dintre numărul predictorilor și calitatea predicției este contraintuitivă, dar poate fi explicată relativ simplu. Dacă includem prea puțini predictori, riscăm să pierdem o parte din informația relevantă deoarece nu includem (unii dintre) predictorii relativ mai puternici, deci calitatea predicției va fi mai scăzută. Dacă includem prea mulți predictori (creștem numărul dimensiunilor), cazurile vor deveni mai similare între ele (distanțele dintre cazuri vor varia mai puțin), informația relevantă, cea care ne ajută să distingem între cazuri, pierzându-se în totalul informației disponibile („the curse of dimensionality"). Prin urmare, numărul cazurilor care pot fi considerate a fi vecini ai unui caz va crește, iar calitatea predicției va scădea (Attewell & Monaghan, 2015, p. 135).

Care este numărul de „vecini" (cazuri similare)?

La acest pas urmărim să stabilim valoarea lui k (numărul vecinilor pe care îi avem în vedere). Valoarea parametrului k determină în mare măsură ce predicție facem într-un anumit caz, care este modelul final și cât de bun este acesta. După cum se poate observa în exemplul din Figura 7.1-2, pentru k = 3 predicția este „bad", iar pentru k = 5 „good".

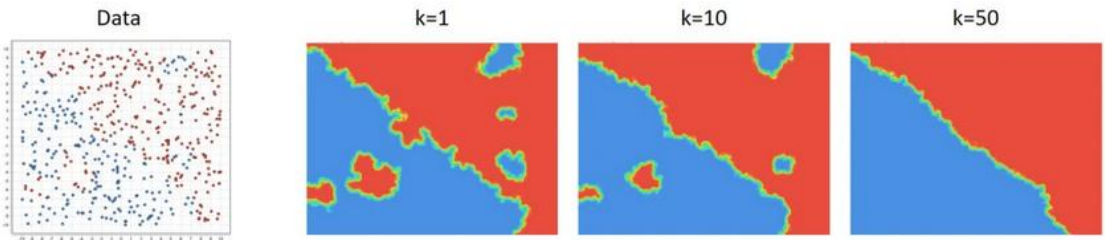
Figura 7.1-2. Modificarea predicției în funcție de valoarea parametrului k



Sursa: (Moreira et al., 2019, p. 206); cercurile reprezintă distanțele euclidiene care determină cei mai apropiați 3, respectiv 5 vecini ai cazului nou, cel pe care dorim să-l prezicem.

Mergând un pas mai departe, în exemplul din Figura 7.1-3 se poate observa că, pornind de la același set de date, diferite valori ale parametrului k duc la soluții care se suprapun în mare parte, dar sunt departe de a fi identice. Valori mici ale lui k produc modele „locale", cel mai adesea non-liniare (granițele dintre grupurile de cazuri similare sunt variabile), iar valori mari duc la modele mai generale, mai simple, cu granițe între grupuri mai bine definite, uneori chiar aproape liniare.

Figura 7.1-3. Soluția obținută de algoritmul k-NN în funcție de diferite valori ale parametrului k

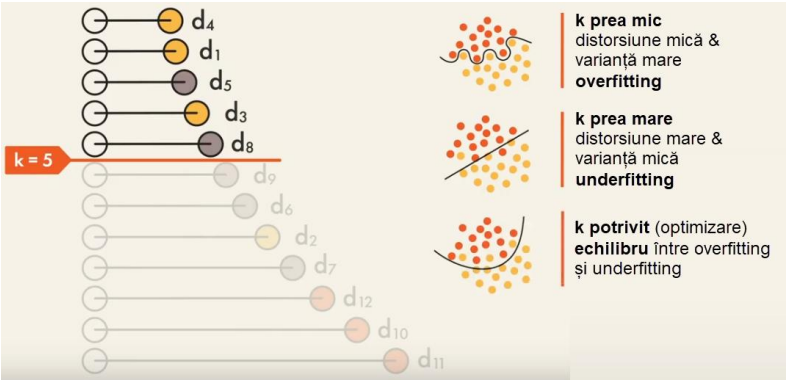


Sursa: Mierswa, Ingo. 2017. K-Nearest Neighbors: A Simple Machine Learning Algorithm.

Un k prea mic va tinde să producă modele de predicție cu distorsiune (bias) mică, dar cu varianță mare, deci va duce la modele de clasificare care „memorează" setul de date utilizat pentru construirea lor (overfitting) și care, în consecință, vor produce predicții relativ mai proaste pe alte seturi de date (Figura 7.1-4). Un k prea mare va avea un efect opus, deci modele cu distorsiune (bias) mare și varianță mică, deci modele care nu profită la maximum de informația din setul de date de instruire (underfitting). Simplu spus, primul tip de modele confundă „zgomotul" cu informația („semnalul"), iar al doilea ignoră o parte din informația disponibilă (semnal) pentru a reduce „zgomotul". Ce înseamnă un k mic sau mare depinde foarte mult de tipul de date, numărul cazuri, numărul de atribute și relațiile dintre ele. Înainte de a face o alegere, este necesar să testăm diferite valori, de preferat automat (optimizare[4]). În urma acestui proces, avem șanse mai mari să ajungem la valoarea optimă a lui k, cea care va asigura un echilibru între nevoia de a obține un model care să îndeplinească simultan două condiții importante: (1) o capacitate predictivă crescută și (2) generalizabilitate (să producă predicții la fel de bune pe alte seturi de date decât cel folosit pentru construirea lui).

[4] Vom discuta despre optimizare în volumul următor al acestui manual.

Figura 7.1-4. Soluția obținută de algoritmul k-NN în funcție de diferite valori ale parametrului k



În general e de preferat să alegem o valoare impară pentru k. O valoare pară ar duce cu o mai mare probabilitate la apariția unor situații de egalitate a numărului de voturi, deci la imposibilitatea predicției, mai ales atunci când atributul prezis are doar două clase.

Ce fel de măsură a distanței folosim?

Două aspecte sunt importante în acest context: (1) să folosim măsuri potrivite pentru tipul de atribute incluse în model (nivelul de măsurare) și (2) să standardizăm predictorii înainte de a realiza predicția (astfel, distanțele dintre cazuri nu vor mai depinde de intervalul de variație al predictorilor (mărimea unității de măsură), deci fiecare predictor va avea o contribuție similară la rezultatul final).

Pentru a calcula distanțele dintre cazuri putem folosi una dintre măsurile disponibile pentru tipul de atribute incluse în analiză.[5] Astfel, RapidMiner Studio prezintă patru categorii de măsuri, fiecare incluzând una sau mai multe măsuri, după cum urmează:

- Mixte:
  - □ Mixed Euclidian Distance: pentru atributele numerice calculează distanța euclidiană; pentru atributele nominale, dacă cele două valori sunt identice distanța va fi 0, iar dacă sunt diferite distanța va fi 1;

[5] O parte dintre distanțele prezentate mai jos sunt ilustrate grafic în volumul 1 al manualului (Comșa, 2022, pp. 226–227). Pentru o prezentare mai extinsă a măsurilor se pot consulta și alte surse (Kotu & Deshpande, 2019, pp. 103–109).

- Nominale:

  □ Nominal Distance: similar cu mai sus (0/1);
  □ Dice Similarity: 2*e/(2*e+u);
  □ Jaccard Similarity: e/(e+u);
  □ Kulczynski Similarity: e/u;
  □ Rogers Tanimoto Similarity: (e+z)/(e+2*u+z);
  □ Russell Rao Similarity: e/(e+u+z);
  □ Simple Matching Similarity: (e+z)/(e+u+z); unde

  - e reprezintă numărul de atribute în cazul cărora cele două cazuri au valori egale și diferite de zero;
  - u reprezintă numărul de atribute în cazul cărora cele două cazuri iau valori diferite;
  - z: reprezintă numărul de atribute în cazul cărora cele două cazuri iau valoarea 0.

- Numerice:

  □ Euclidean Distance: Sqrt(Sum_(j=1) [y(1,j)-y(2,j)]^2);
  □ Canberra Distance: Sum_(j=1) |y(1,j)-y(2,j)| / (|y(1,j)|+|y(2,j)|);
  □ Chebychev Distance: max_(j=1) (|y(1,j)-y(2,j)|);
  □ Correlation Similarity: valoarea coeficientului de corelație dintre valorile celor două cazuri relativ la toate atributele;
  □ Cosine Similarity: cosinusul unghiului definit de vectorii atributelor celor două cazuri;
  □ Dice Similarity: 2*Y1Y2/(Y1+Y2);
  □ Jaccard Similarity: Y1Y2/(Y1+Y2-Y1Y2);
  □ Dynamic Time Warping Distance: folosită în cazul seriilor de timp pentru a măsura distanța dintre două secvențe temporale;
  □ Inner Product Similarity: -Similarity = -Sum_(j=1) y(1,j)*y(2,j);
  □ Kernel Euclidean Distance: distanța euclidiană calculată relativ la un spațiu transformat;
  □ Manhattan Distance: Sum_(j=1) |y(1,j)-y(2,j)|;
  □ Max Product Similarity: -Similarity = -max_(j=1) (y(1,j)*y(2,j));
  □ Overlap Similarity: minY1Y2 / min(Y1,Y2); unde

  - y(i,j) reprezintă valoarea observată în cazul atributului j, cazul i; de exemplu, prin „y(1,3) - y(2,3)” ne referim la diferența dintre valorile luate la atributul 3 de către cazurile 1 și 2;

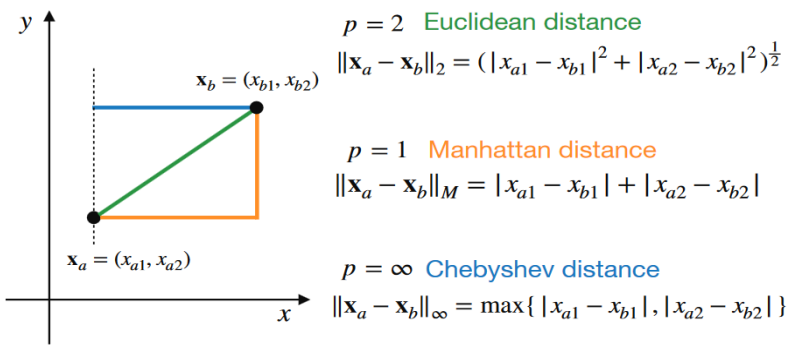
- Y1Y2 = Sum_(j=1) y(1,j)*y(2,j);
- Y1 = Sum_(j=1) y(1,j);
- Y2 = Sum_(j=1) y(2,j);
- minY1Y2 = Sum_(j=1) min [y(1,j),y(2,j)].

– Bregman Divergences:

□ Generalized Divergence: Sum1 Sum2, unde Sum1 = Sum_(j=1) y(1,j)*ln[y(1,j)/y(2,j)] Sum2 = Sum_(j=1) [y(1,j)-y(2,j)]; nu se poate calcula dacă valoarea unuia dintre atribute este ≤ 0;
□ Itakura Saito Distance: y(1,1)/y(2,1)-ln[y(1,1)/y(2,1)]-1; se poate calcula doar atunci când avem un singur atribut cu valori > 0;
□ KL Divergence: Sum_(j=1) [y(1,j)*log_2(y(1,j)/y(2,j))];
□ Logarithmic Loss: y(1,1)*ln[y(1,1)/y(2,1)]-(y(1,1)-y(2,1)); se poate calcula doar atunci când avem un singur atribut cu valori > 0;
□ Logistic Loss: y(1,1)*ln[y(1,1)/y(2,1)]+(1-y(1,1))*ln[(1-y(1,1))/(1-y(2,1))]; se poate calcula doar atunci când avem un singur atribut cu valori > 0;
□ Mahalanobis Distance: Sqrt [(vecY1-vecY2) S (vecY1-vecY2)], unde vecY1 este vectorului valorilor cazului 1, vecY2 ale cazului 2, iar S matricea de covarianță;
□ Squared Euclidean Distance: Sum_(j=1) [y(1,j)-y(2,j)]^2;
□ Squared Loss: [y(1,1)-y(2,1)]^2; poate fi calculată doar dacă avem un singur atribut.

Doar pentru ilustrare, prezentăm aici trei distanțe Minkowski și anume: Euclidiană, Manhattan și Chebyshev (Figura 7.1-5). În general, acestea produc rezultate similare.

Figura 7.1-5. Ilustrarea grafică a trei distanțe Minkowski



Sursa: (Fu & Yang, 2021, p. 4)

Dat fiind faptul că majoritatea măsurilor produc rezultate diferite în funcție de mărimea intervalului de variație (unitatea de măsură) asociat unui atribut, este necesar să normalizăm atributele metrice înainte de a le introduce în analiză. Pe scurt, prin normalizare se înțelege compatibilizarea scalelor, a intervalelor lor de variație, deci creșterea similarității dintre acestea. În urma normalizării, unitățile de măsură originale ale atributelor dispar, fiind înlocuite cu unele relative. Normalizarea poate consta în translatarea intervalelor de variație asociate atributelor metrice pe un interval de variație unic (de exemplu [0; 1]) sau în standardizarea atributelor (atributele rezultate vor avea media 0 și abaterea standard 1; în acest caz, unitatea de măsură asociată unui atribut este în fapt abaterea standard a atributului original).[6]

Pentru a ilustra necesitatea acestui pas, vom considera un exemplu foarte simplu. Setul de date din Tabelul 7.1-1 include cinci cazuri relativ la care avem informații cu privire la trei atribute: doi predictori metrici (vechimea în companie și salariul) și o variabilă dependentă cu două clase (părăsește sau nu compania). Ne interesează să prezicem clasa la care aparține cazul „?”, deci dacă acest angajat va părăsi sau nu compania. Relativ la cazul „?” avem informații cu privire la cei doi predictori. Dat fiind faptul că predictorii sunt de tip metric, pentru a calcula distanțele dintre cazuri și cazul necunoscut, folosim distanța euclidiană. Observăm faptul că intervalele de variație ale celor doi predictori sunt foarte diferite (2-6, respectiv 2500-5000). Ca urmare a acestei diferențe, distanța dintre două cazuri va fi dominată de distanța corespunzătoare atributului cu valorile cele mai mari (atributul salariu, în acest caz). Altfel spus, distanțele obținute sunt distorsionate (biased) înspre atributul dominant (cu valori numerice mari). Concret, pentru exemplul de față, dacă avem în vedere cazul cinci și cazul de prezis, distanța Euclidiană dintre acestea va fi egală cu

$$\sqrt{(6-3)^2+(5000-2500)^2} = \sqrt{3^2+2500^2} = \sqrt{6250009} \cong 2500.002$$

Observăm că distanța în funcție de salariu determină în proporție de aproape 100% distanța dintre cele două cazuri în funcție de ambele atribute (valorile celor două distanțe – salariu și totală – sunt aproape identice). Indiferent care ar fi fost diferența (realist vorbind) dintre cele două cazuri în funcție de vechime, am fi ajuns la aceeași concluzie. Simplu spus, distanța dintre cazuri depinde aproape în

[6] Pentru a standardiza o serie de valori (calcularea scorurilor z) folosim formula $x_{si} = \frac{x_i - \bar{x}}{sd_x}$, unde x_{si} este valoarea standardizată pentru cazul i, x_i este valoarea originală pentru cazul i, x este media valorilor variabilei *X*, iar sd_x este abaterea standard a variabilei *X*. normalizarea se poate face folosind și una dintre formulele $x_{ni} = \frac{x_i - \bar{x}}{max_x - min_x}$ sau $x_{ni} = \frac{x_i - min_x}{max_x - min_x}$, unde x_{ni} reprezintă valoarea normalizată pentru cazul i, max_x este valoarea maximă a variabilei *X*, iar min_x valoarea minimă.

totalitate de distanța (diferența) dintre salarii. Această situație apare nu pentru că salariul contează neapărat mai mult comparativ cu vechimea, ci în principal pentru că valorile atributului salariul au un interval de variație disproporționat mai mare. O astfel de situație este absolut de nedorit. Impactul unui predictor asupra soluției finale ar trebui să depindă de intensitatea relației dintre acesta și dependentă, nu de un criteriu arbitrar precum intervalul de variație al predictorului. Mai mult, distanțele rezultate vor fi foarte diferite (variază de la 1 la 2500). Observăm că valorile standardizate sunt mult mai apropiate (similare) între cei doi predictori. Prin urmare, atunci când calculăm distanțele euclidiene folosind datele standardizate, valorile rezultate nu mai sunt dominate de unul dintre atribute (în plus, sunt relativ mai similare), după cum se poate observa și din calculele de mai jos:

$$\sqrt{(1.006+0.671)^2+(1.664+0.936)^2}=\sqrt{1.68^2+2.6^2}=\sqrt{2.81+6.76}=\sqrt{9.57}\cong 3.09$$

În acest caz, contribuția diferenței dintre salarii la distanța dintre cele două cazuri va fi semnificativ mai mică: 70.6% (6.76/9.57).

Mai important, primii trei vecini ai cazului de prezis diferă parțial în cele două situații ceea ce, în acest caz (dar și în general), are un impact important asupra predicției. Astfel, atunci când calculăm distanțele folosind valorile originale, nestandardizate, predicția este „nu pleacă” (două din trei voturi sunt nu), în timp ce, în cazul valorilor standardizate predicția este „pleacă” (două din trei voturi sunt da).

Tabelul 7.1-1. Impactul standardizării asupra distanțelor și predicției (un exemplu simplu)

Valorile originale			Valorile standardizate		Distanța Euclidiană cu valorile	
Pleacă	Vechime	Salariu	Vechime	Salariu	Originale	Standardizate
da	2	2500	-1.230	-0.936	1	0.56
nu	3	3000	-0.671	-0.416	500	0.52
nu	6	3000	1.006	-0.416	500	1.76
da	4	3500	-0.112	0.104	1000	1.18
nu	6	5000	1.006	1.664	2500	3.09
?	3	2500	-0.671	-0.936		

Cazul cu fundal de culoare gri este cazul de referință, cel relativ la care calculăm distanțele. Valorile cu fundal de culoare verde indică distanțele cele mai mici relativ la cazul de referință, „vecinii” acestuia (k=3).

Cum stabilim care este clasa prezisă?

După cum am ilustrat în exemplul anterior, predicția cu privire la apartenența cazului de prezis la una din clasele variabilei dependente se realizează prin vot

majoritar. Simplu spus, clasa „câștigătoare” este clasa la care aparțin cei mai mulți dintre vecinii cazului de prezis. Adesea, vecinii cazului de prezis se află la distanțe destul de diferite de acesta. Într-o situație de acest tip, votul majoritar simplu poate să nu producă cea mai bună predicție. Prin urmare, adesea, votul majoritar este ponderat cu distanța dintre fiecare vecin și cazul de prezis. În concluzie, trebuie să stabilim dacă alegem ca fiecare vecin să aibă aceeași importanță în stabilirea predicției sau preferăm ca importanța unui vecin să fie invers proporțională cu distanța față de cazul prezis. Cele mai multe implementări ale algoritmului k-NN oferă posibilitatea acestei alegeri. În cazul RapidMiner, opțiunea „weighted vote” asociată operatorului k-NN specifică preferința pentru un vot ponderat cu distanța. Prin urmare, dacă lăsăm marcată această opțiune (așa este implicit), votul va fi ponderat cu distanța (importanța fiecărui vot va fi invers proporțională cu distanța dintre cazuri). În general preferăm votul ponderat.

Avantaje și dezavantaje

k-NN prezintă o serie de avantaje relativ la ceilalți algoritmi de clasificare, precum următoarele:

- nu are nevoie de etapa de instruire, deci poate trece direct la etapa de testare;
- poate modela atât date liniare cât și non-liniare (nu face nicio asumpție cu privire la modelul care se află în spatele datelor, adică la procesul de generare a datelor);
- are un număr redus de parametri, deci optimizarea (identificarea valorilor potrivite ale parametrilor) este relativ ușor de realizat;
- pentru valori suficient de mari ale lui k, algoritmul k-NN este robust la outlieri (nu și pentru valori prea mici);
- este simplu și intuitiv.

Principalele dezavantaje ale k-NN includ următoarele:

- calitatea modelului de clasificare scade pe măsură ce crește numărul predictorilor; fenomenul, cunoscut sub numele de „curse of dimensionality”, poate fi explicat astfel: pe măsură ce numărul de dimensiuni (atribute) crește, distanțele dintre cazuri se măresc și devin tot mai similare, ceea ce face tot mai dificil procesul de identificare a unui set clar definit de cazuri pe care să le definim ca „vecini” ai unui caz; mai mult, numărul predictorilor corelează pozitiv cu necesarul de resurse de calcul;

ambele probleme pot fi reduse dacă includem în model doar predictorii relevanți și/sau reducem numărul dimensiunilor folosind analiza factorială, analiza componentelor principale etc.;
- dacă setul de date include atribute cu intervale de variație și/sau unități de măsură foarte diferite, modelul de predicție va fi dominat de atributele cu variație mare; soluția este să normalizăm datele înainte de calcularea distanțelor; o altă soluție este transformarea atributelor metrice în atribute categoriale (binning);
- în faza de instruire este necesară stocarea setului de date în memorie;
- faza de testare necesită timp și putere de calcul, mai ales în cazul datelor mari (multe cazuri și atribute) (unele versiuni ale k-NN rezolvă parțial aceste probleme);[7]
- în cazul unor date cu foarte multe atribute sau cu atribute care iau adesea valoarea zero (sparse data), calitatea predicțiilor este adesea relativ mai redusă comparativ cu alți algoritmi; în acest caz este util să reținem în analiză doar predictorii relativ mai importanți și/sau să reducem datele / dimensiunile folosind, de exemplu, PCA (Principal Component Analysis);
- dacă datele sunt nebalansate (una dintre clase are o pondere foarte mică / mare), acuratețea predicției tinde să scadă (mai ales pentru valori mari ale lui k); putem balansa datele folosind diferite proceduri (vom discuta despre acest subiect în volumul următor).

7.2. Un exemplu didactic

Să presupunem că o companie a avut de-a lungul timpului mai mulți angajați și că unii dintre aceștia nu mai lucrează în prezent în acea companie. Cu privire la toți angajații avem doar următoarele informații: dacă a plecat din companie (da / nu), vechimea (numărul de ani ca angajat al companiei) și salariul mediu din ultimul an (RON). Setul de date aferent acestei situații ipotetice are 20 de cazuri și este prezentat în Tabelul 7.2-1.

[7] Această problemă este semnalată adesea, mai ales în cazul seturilor de date foarte mari. Într-o analiză care analizează comparativ calitatea predicțiilor și timpul necesar pentru obținerea acestora, folosind un set de date de variabil (1-500 mii de produse) și două valori ale lui k (10 și 100), autorul a arătat că ANN (Aproximate Nearest Neighbors, implementat în Python sub numele „Hierarchical Navigable Small World", pe scurt HNSW), comparativ cu k-NN, produce predicții aproape la fel de bune (94-99% din situații) mult mai rapid (de 10 până la 380 ori mai rapid) (Marie Stephen Leo. 2020. KNN (K-Nearest Neighbors) is Dead!).

Tabelul 7.2-1. Un set de date fictive relativ la (foști) angajați ai unei companii (20 cazuri)

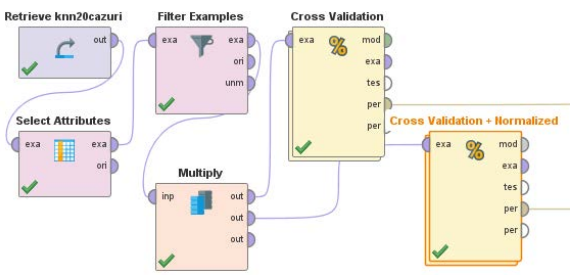
Id	Atriție	Vechime (ani)	Salariu (RON)
1	da	1	4000
2	da	4	5000
3	da	1	2000
4	da	1	3000
5	da	2	2500
6	nu	2	3000
7	nu	6	5000
8	nu	3	4000
9	nu	3	3000
10	nu	7	7000

Id	Atriție	Vechime (ani)	Salariu (RON)
11	nu	5	5000
12	nu	2	1000
13	nu	1	1500
14	nu	3	2000
15	nu	4	3000
16	nu	2	4000
17	nu	2	2000
18	nu	5	4000
19	nu	4	4000
20	nu	3	5000

Prima dată vom estima și compara calitatea predicțiilor a două modele de clasificare foarte simple care se deosebesc doar prin faptul că în cazul unuia atributele obișnuite (predictorii) au fost standardizate (Figura 7.2-1). Procesul aferent (knn_didactic_1) și setul de date sunt incluse în folderul Exemple.

Figura 7.2-1. Impactul standardizării atributelor asupra calității modelului de predicție

Pasul 1:
Setul de date este „knn20cazuri". Setările operatorilor sunt vizibile în procesul inclus. Operatorul „Cross validation" estimează și validează modelul de predicție, folosind atributele nestandardizate, respectiv standardizate. Ne interesează să comparăm performanța celor două modele (calitatea generală a predicției = accuracy).



Rezultate:
Observăm că acuratețea predicției este semnificativ mai mare în cazul modelului construit pe datele standardizate (86% vs. 70%). Creșterea calității predicției se datorează faptului că modelul cu datele standardizate face mai puține erori de predicție, mai ales în cazul categoriei „da" (pleacă). Pentru această clasă, indicatorii precizie și reamintire se dublează sau chiar triplează ca urmare a standardizării atributelor.

date nestandardizate

accuracy: 69.84% +/- 2.75% (micro average: 70.00%)

	true da	true nu	class precision
pred. da	1	2	33.33%
pred. nu	4	13	76.47%
class recall	20.00%	86.67%	

date standardizate

accuracy: 85.71% +/- 14.29% (micro average: 85.00%)

	true da	true nu	class precision
pred. da	3	1	75.00%
pred. nu	2	14	87.50%
class recall	60.00%	93.33%	

Sursa: procesul „knn_didactic_1.rmp" (folderul „Exemple").

Pe lângă impactul asupra calității modelului de clasificare, standardizarea duce uneori și la schimbarea predicțiilor. Considerând același set de date și un model k-NN cu trei vecini (Tabelul 7.2-2, Figura 7.2-2), observăm că predicția relativ la cazul nou (cazul 21) se schimbă în funcție standardizarea sau nu a predictorilor. Dacă predictorii nu sunt standardizați, cele mai mici prime trei distanțe sunt cele observate la cazurile 2, 10 și 20. Deoarece două dintre aceste cazuri aparțin clasei NU și unul clasei DA, modelul prezice că noul caz este de tip NU. Dacă standardizăm predictorii, cele mai mici distanțe aparțin cazurilor 1, 2 și 20. În acest caz, deoarece unul dintre aceste cazuri aparține clasei NU și două clasei DA, modelul prezice că noul caz este de tip DA.

Tabelul 7.2-2. Impactul standardizării atributelor asupra predicției relativ la un caz nou

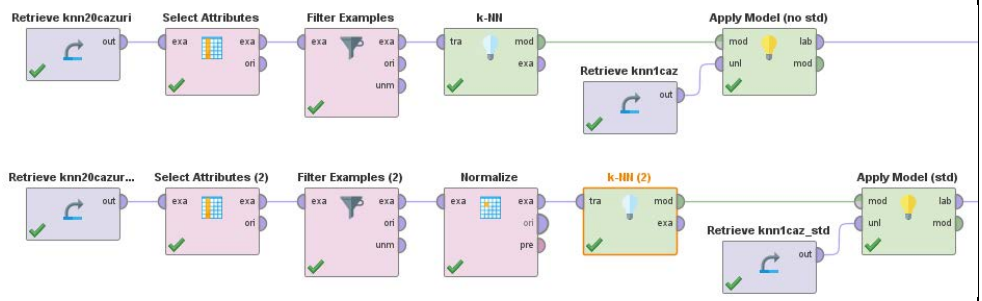
Id	Părăsit	Vechime	Salariu	Distanța	Vechime (std)	Salariu (std)	Distanța
1	da	1	4,000	4,000	-0.95	0.10	2.7
2	da	4	5,000	3,000	0.47	0.77	2.5
3	da	1	2,000	6,000	-0.95	-1.24	4.0
4	da	1	3,000	5,000	-0.95	-0.57	3.4
5	da	2	2,500	5,500	-0.47	-0.91	3.7
6	nu	2	3,000	5,000	-0.47	-0.57	3.4
7	nu	6	5,000	3,000	1.42	0.77	3.1
8	nu	3	4,000	4,000	0.00	0.10	2.8
9	nu	3	3,000	5,000	0.00	-0.57	3.5
10	nu	7	7,000	1,000	1.90	2.11	2.9
11	nu	5	5,000	3,000	0.95	0.77	2.8

Id	Părăsit	Vechime	Salariu	Distanța	Vechime (std)	Salariu (std)	Distanța
12	nu	2	1,000	7,000	-0.47	-1.91	4.7
13	nu	1	1,500	6,500	-0.95	-1.58	4.4
14	nu	3	2,000	6,000	0.00	-1.24	4.1
15	nu	4	3,000	5,000	0.47	-0.57	3.6
16	nu	2	4,000	4,000	-0.47	0.10	2.7
17	nu	2	2,000	6,000	-0.47	-1.24	4.1
18	nu	5	4,000	4,000	0.95	0.10	3.3
19	nu	4	4,000	4,000	0.47	0.10	3.0
20	nu	3	5,000	3,000	0.00	0.77	2.2
21	**?**	**1**	**8,000**	**2NU + 1DA → NU**	**-0.95**	**2.78**	**1NU + 2DA → DA**

Figura 7.2-2. Impactul standardizării atributelor asupra predicției relativ la un caz nou

Pasul 1:

Utilizăm seturile de date knn20cazuri, knn1caz și knn1caz_std. Setările operatorilor sunt vizibile în procesul inclus. Construim două modele de predicție simple, identice cu excepția faptului că în modelul std atributele sunt standardizate. Ne interesează să comparăm predicțiile celor două modele pe același caz. Setările operatorului k-NN sunt următoarele: număr vecini / k = 3, nu selectăm Weighted vote, măsură de tip numeric Distanța Euclidiană.



Rezultate:

Atunci când datele nu sunt standardizate, modelul prezice că noul caz este de tip NU. Probabilitățile asociate sunt 0.333 pentru DA și 0.667 pentru NU, deci, din cei trei vecini ai cazului de prezis, doi au fost de tip NU. Atunci când datele sunt standardizate, modelul prezice că noul caz este de tip DA. Probabilitățile asociate sunt 0. 667 pentru DA și 0.333 pentru NU, deci, din cei trei vecini ai cazului de prezis, doi au fost de tip DA.

prediction(Atriție)	confidence(da)	confidence(nu)	Vechime	Salariu_RON
nu	0.333	0.667	1	8000
da	0.667	0.333	-0.949	2.782

Sursa: procesul „knn_didactic_2.rmp" (folderul „Exemple").

7.3. Exemple de utilizare a k-NN în RapidMiner

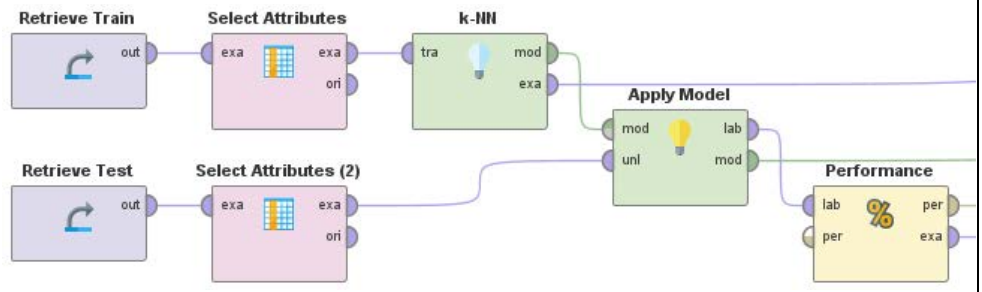
În această secțiune vom prezenta două exemple de analiză în care am folosit operatorul k-NN din RapidMiner (Modeling > Predictive > Lazy).

Un model simplu

Pentru acest exemplu vom folosi aceleași trei atribute din setul de date „employee_attrition" și vom realiza un model de clasificare de tip k-NN. Scopul analizei este predicția situațiilor de părăsire a companiei în funcție de vechimea în muncă și munca peste program. Firesc, ne așteptăm ca o vechime mai mare să reducă șansele de plecare, iar munca peste program să le crească. Pașii realizării acestui model în RapidMiner, rezultatele obținute și interpretarea lor apar în Figura 7.3-1.

Figura 7.3-1. k-NN: un model simplu în RapidMiner

Pasul 1:
Realizăm procesul din imaginea de mai jos (procesul și datele utilizate sunt incluse în folderul asociat volumului). Seturile de date utilizate sunt „employee_attrition_training_validation" (instruire), respectiv „employee_attrition_testing" (testare). Relativ la ambele seturi de date am păstrat doar atributele Attrition, OverTime și YearsAtCompany (operatorul „Select Attributes"). Includem operatorul k-NN cu setările originale apoi restul operatorilor din imagine. Rulăm modelul.



Rezultate (setul de instruire):
Seturile de date conțin doar atributele selectate anterior (în imagine apar primele 5 cazuri din setul de instruire). Distribuția statistică a atributului de interes – Attrition – este 0.161 pentru clasa Yes (pleacă) și 0.839 pentru clasa No (nu pleacă).

Row No.	Id	Attrition	OverTime	YearsAtCompany
1	4	Yes	Yes	0
2	5	No	Yes	8
3	7	No	No	2
4	10	No	Yes	1
5	14	No	No	5

Nominal values

Ind...	Nominal value	Absolute count	Fraction
1	No	863	0.839
2	Yes	166	0.161

Close

Rezultate (setul de testare):

Relativ la setul de testare, modelul produce predicțiile din imagine (am inclus primele 5 cazuri și alte trei de interes în contextul acestui exemplu). Coloana Attrition prezintă situația reală, iar coloana „prediction(Attrition)" predicția realizată de model. Observăm că o parte dintre predicții sunt greșite (de exemplu cea cu Id 1). Probabilitățile asociate predicțiilor apar pe coloanele „confidence". Relativ la fiecare caz, clasa care are asociată cea mai mare probabilitate (confidence) prezisă, determină clasa la care modelul prezice că aparține acel caz. Pragul de probabilitate implicit în funcție de care un caz este prezis a aparține unei clase sau alteia este 50%. Astfel, dacă probabilitatea este sub 50% modelul prezice că acel caz aparține clasei No (angajatul nu pleacă). Dacă probabilitatea este cel puțin 50%, clasa prezisă va fi Yes (angajatul pleacă). Cazurile au asociate diferite probabilități de a pleca. Cea mai mică probabilitate estimată este 0% – cea mai mare parte a angajaților. Cea mai mare probabilitate, 0.6 (60%), apare doar în cazul angajaților 167 și 811 (amândoi au o vechime mai mică de un an și muncesc peste program). Dat fiind faptul că pragul de probabilitate implicit este 0.5, modelul prezice că angajații 167 și 811 vor pleca.

Row No.	Id	Attrition	prediction(Attrition)	confidence(No)	confidence(Yes)	OverTime	YearsAtCompany
1	1	Yes	No	0.600	0.400	Yes	6
2	2	No	No	1	0	No	10
3	8	No	No	1	0	No	7
4	11	No	No	0.800	0.200	No	1
5	12	No	No	0.800	0.200	No	9
...							
41	167	Yes	Yes	0.400	0.600	Yes	0
178	811	Yes	Yes	0.400	0.600	Yes	0
...							
293	1352	No	No	0.823	0.177	Yes	33

...

k-NN prezintă foarte puține informații relativ la model (numărul cazurilor și claselor, respectiv valoarea lui k). Secțiunea PerformanceVector conține matricea de confuzie și diferite măsuri ale performanței clasificării (cele selectate) calculate relativ la setul de date de testare. Acuratețea predicției este mare (84%) doar pentru că majoritatea angajaților nu pleacă. Relativ la clasa de interes pentru noi, cei care pleacă, calitatea predicției e destul de slabă: chiar dacă precizia clasificării acestei clase este perfectă (100%), adică toate cazurile prezise de model că pleacă chiar pleacă (sunt doar două cazuri), reamintirea este aproape zero (3%), adică modelul identifică doar 3% dintre angajații care pleacă efectiv. Pragul implicit al clasificării este 50%. Pentru a crește calitatea modelului relativ la clasa de interes, putem reduce aceste prag (Threshold) la 30% de exemplu și să clasificăm toate cazurile cu probabilitatea prezisă de a pleca de cel puțin 30% ca angajați care pleacă. Desigur, în acest caz ponderea erorilor de tip I va crește, dar vom identifica o parte mai mare dintre angajații care pleacă (de exemplu cazul cu Id 1). Dacă standardizăm atributele, acuratețea predicției nu se schimbă, însă scade precizia și crește reamintirea clasei Da, ceea ce ne ajută deoarece ne dorim să identificăm mai mulți angajați care au șanse mari să părăsească firma.

accuracy: 84.35%

	true No	true Yes	class precision
pred. No	370	69	84.28%
pred. Yes	0	2	100.00%
class recall	100.00%	2.82%	

set testare, prag implicit 50%, atribute nestandardizate

accuracy: 84.81%

	true No	true Yes	class precision
pred. No	363	60	85.82%
pred. Yes	7	11	61.11%
class recall	98.11%	15.49%	

set testare, prag implicit 50%, atribute standardizate

Un model relativ mai complex

k-NN are doar patru parametri (Figura 7.3-2): numărul de vecini (k), vot ponderat în funcție de distanță (weighted vote), nivelul de măsurare (measure types) și măsura efectivă. Cel mai adesea seturile de date conțin ambele tipuri de atribute, metrice și non-metrice, prin urmare vom folosi măsuri de tip mixt, mai exact distanța euclidiană mixtă. De asemenea, preferăm ca votul să fie ponderat (să ia în calcul distanța dintre cazuri). Prin urmare, singurul parametru pe care trebuie să-l alegem este k (numărul de vecini). Însă, după cum am văzut anterior, această alegere este foarte importantă.

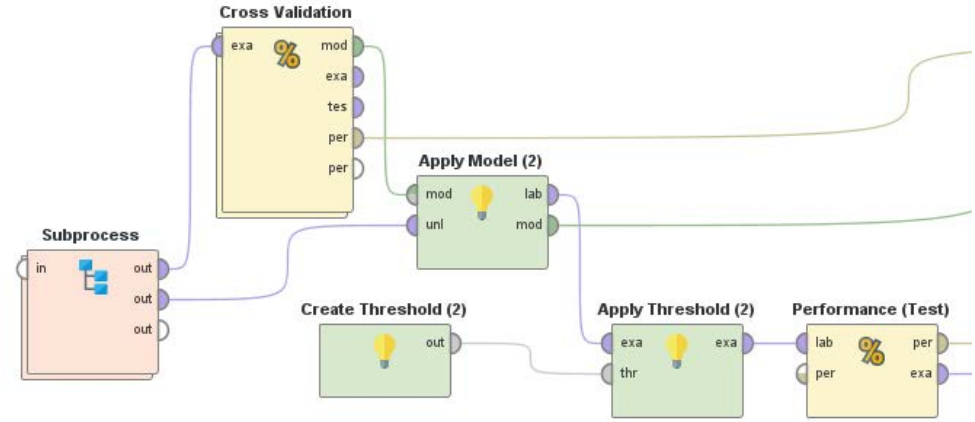
Figura 7.3-2. Parametrii operatorului k-NN: setări implicite (stânga) vs model (dreapta)

Pentru acest exemplu vom folosi atributele toate din setul de date „employee_attrition”. Anterior analizei, fiecare atribut numeric a fost standardizat (Normalize, Z-tranformation). Ne propunem să realizăm un model de clasificare

bazat pe k-NN cu scopul de a prezice situațiile de părăsire a companiei. Pașii realizării acestui model în RapidMiner, rezultatele obținute și interpretarea lor apar în Figura 7.3-3.

Figura 7.3-3. k-NN: un model relativ mai complex în RapidMiner

Pasul 1:
Realizăm procesul din imaginea de mai jos. În folderul asociat volumului am inclus două versiuni ale acestui proces: o versiune mai simplă, fără setarea pragului de probabilitate (acesta este stabilit implicit la 0.5) și o versiune care oferă posibilitatea setării pragului de probabilitate. Toate datele necesare sunt pregătite anterior în interiorul operatorului Subprocess. Setul de date utilizat pentru instruirea modelului este „employee_attrition_training_validation". Setul utilizat pentru testarea modelului este „employee_attrition_testing". Includem operatorul k-NN cu setările originale parțial modificate (k=3 în loc de 5). În cazul modelului 2 am inclus operatorii care ne ajută să setăm pragul de probabilitate dorit (Create Threshold și Apply Threshold). Includem operatorul necesar pentru aplicarea modelului pe setul de date de testare, respectiv operatorul care calculează măsurile de performanță. Realizăm conexiunile și rulăm procesul. La modelul 1, pragul de probabilitate este setat implicit la 0.5 (prima clasă este No, a doua Yes). Prin urmare, modelul va prezice că angajații în cazul cărora valoarea atributului „confidence(Yes)" va fi mai mare de 0.5 vor pleca, iar restul vor rămâne. La modelul 2, pragul este stabilit manual la 0.3.



Rezultate (setul de testare):
Relativ la setul de testare, modelul produce predicțiile din imagine (am inclus primele 5 cazuri și alte trei de interes în contextul acestui exemplu). Coloana Attrition prezintă situația reală, iar coloana „prediction(Attrition)" predicția realizată de model. Observăm că unele predicții sunt greșite (în imagine, cele cu Id 1, 11, 54 și 328). Probabilitățile asociate predicțiilor apar pe coloanele „confidence". Pentru fiecare caz este selectată ca predicție clasa care are cea mai mare probabilitate (confidence).

Row No.	Id	Attrition	prediction(Attrition)	confidence(No)	confidence(Yes)
1	1	Yes	No	0.667	0.333
2	2	No	No	0.658	0.342
3	8	No	No	1	0
4	11	No	Yes	0.336	0.664
5	12	No	No	1	0
...					
14	54	No	Yes	0.326	0.674
...					
74	328	Yes	No	0.653	0.347
...					
108	485	Yes	Yes	0	1
...					

Secțiunea PerformanceVector conține matricea de confuzie și diferite măsuri ale performanței clasificării (cele selectate), toate calculate relativ la ambele seturi de date (instruire și testare). Observăm că deși acuratețea predicției este peste 80% la faza de instruire, la test valorile sunt semnificativ mai mici. Diferența este un semn al prezenței supra-ajustării (overfitting) și indică faptul că modelul are șanse relativ mai mici de generalizare. Foarte probabil problema poate fi rezolvată dacă reducem numărul predictorilor (fie eliminăm o parte din predictori, cei mai puțin importanți, fie folosim o metodă de reducere a numărului de dimensiuni, PCA de exemplu). Comparativ cu modelul care a inclus doar doi predictori, modelul cu toți predictorii are, surprinzător, o acuratețe a clasificării puțin mai mică (am discutat mai sus despre acest aparent paradox), respectiv o precizie mai mică (o pondere mai mică a cazurilor prezise ca plecări chiar sunt plecări). În continuare, modelul prezice mai bine situația în care angajații nu pleacă (precizia = 85/88%, reamintirea = 92/69%) comparativ cu situația în care angajații pleacă (precizia = 30/23%, reamintirea = 17/48%). Observăm că scăderea pragului duce la identificare mai multor cazuri de plecare, ceea ce e dezirabil, chiar dacă erorile de predicție cresc.

accuracy: 83.09% +/- 2.48% (micro average: 83.09%)

	true No	true Yes	class precision
pred. No	824	135	85.92%
pred. Yes	39	31	44.29%
class recall	95.48%	18.67%	

instruire, prag 50%

accuracy: 67.93% +/- 4.67% (micro average: 67.93%)

	true No	true Yes	class precision
pred. No	617	84	88.02%
pred. Yes	246	82	25.00%
class recall	71.49%	49.40%	

instruire, prag 30%

accuracy: 80.27%

	true No	true Yes	class precision
pred. No	342	59	85.29%
pred. Yes	28	12	30.00%
class recall	92.43%	16.90%	

testare, prag 50%

accuracy: 65.31%

	true No	true Yes	class precision
pred. No	254	37	87.29%
pred. Yes	116	34	22.67%
class recall	68.65%	47.89%	

testare, prag 30%

Calitatea clasificării în funcție de numărul „vecinilor” și al predictorilor

Impactul parametrului k

Diferite valori ale lui k duc adesea la o calitate diferită a modelelor de clasificare. Acest lucru se întâmplă și în cazul modelului nostru. Pentru această analiză am considerat modelul prezentat în secțiunea anterioară, varianta cu pragul de 30% (considerăm că vor pleca efectiv toți angajații în cazul cărora probabilitatea prezisă de plecare este de cel puțin 30%). Procesul este inclus în folderul asociat volumului. Aici prezentăm doar rezultatele, adică valorile indicatorilor de calitate a clasificării (acuratețea, respectiv precizia și reamintirea în cazul clasei „Pleacă”) pentru cele două seturi de date și trei valori ale lui k (Tabelul 7.3-1). Observăm că scorul măsurii reamintire scade pe măsură ce crește k, în timp ce scorul preciziei crește. Dacă ne interesează în primul rând să identificăm cât mai multe dintre situațiile de plecare, vom alege modelul în care k = 3 (REC = 48 la setul de test). Dacă vrem în primul rând să facem predicții cât mai corecte relativ la clasa „Pleacă”, alegem modelul cu k = 7 (PRE = 31 la setul de test).

Tabelul 7.3-1. Valoarea lui k și calitatea clasificării

	k=3			k=5			k=7		
Set de date	**ACC**	**PRE**	**REC**	**ACC**	**PRE**	**REC**	**ACC**	**PRE**	**REC**
Instruire	68	25	49	81	38	33	83	46	24
Testare	65	23	48	77	27	25	81	31	17

ACC = acuratețea; PRE = precizia și REC = reamintirea, ambele în cazul clasei „Pleacă”

Impactul eliminării predictorilor relativ mai puțin importanți

Ce se întâmplă dacă includem în model mai puțini predictori, doar predictorii relativ mai importanți? După cum am văzut mai devreme, paradoxal, în cazul k-NN sunt șanse mai mari să obținem un model relativ mai bun. Pentru această analiză considerăm același model (vezi în secțiunea anterioară, varianta cu pragul de 30%). Procesul este inclus în folderul asociat volumului. În Tabelul 7.3-2 am prezentat comparativ valorile indicatorilor de calitate a clasificării (acuratețea, respectiv precizia și reamintirea în cazul clasei „Pleacă") pentru cele două seturi de date (instruire vs testare) și două seturi de predictori (toți vs. o selecție). Relativ la setul de test, observăm că măsurile de calitate sunt evident mai bune în cazul modelului cu mai puțini predictori (dar relativ mai performanți): acuratețea predicției e puțin mai mare (82 vs 77), precizia e mult mai mare (42 vs 27), la fel și reamintirea (38 vs 25).

Tabelul 7.3-2. Numărul predictorilor și calitatea clasificării

	Toți predictorii			Un subset al predictorilor		
Set de date	**ACC**	**PRE**	**REC**	**ACC**	**PRE**	**REC**
Instruire	81	38	33	80	40	47
Testare	77	27	25	82	42	38

ACC = acuratețea; PRE = precizia și REC = reamintirea, ambele în cazul clasei „Pleacă"
Subsetul predictorilor: YearsWithCurrManager, Age, OverTime, TotalWorkingYears, StockOptionLevel.

8. ARBORELE DECIZIONAL (DECISION TREE)

Nume operator	Decision Tree
Categoria majoră	învățare asistată (supervised learning)
Tip model	clasificare (classification)[1]
Grup	arbori decizionali
Atribut dependent	categorial[2]
Atribute independente	numerice, categoriale
Gestionează valorile lipsă	da
Distorsiune (bias)	mică
Varianță (variance)	mare

Bibliografie utilizată și recomandată:

- Data Science: Concepts and Practice (Kotu & Deshpande, 2019, Chapter 4.1)
- Data Analytics for the Social Sciences: Applications in R (Garson, 2021, Chapter 4)
- Machine Learning for Social and Behavioral Research (Jacobucci et al., 2023, Chapter 5)
- Machine Learning for Business Analytics: Concepts, Techniques and Applications in RapidMiner (Shmueli et al., 2023, Chapter 9)

[1] Există și o variantă potrivită în cazul în care dependenta este metrică (problema / sarcina este de tip regresie): arbore de regresie (regression tree). O parte dintre operatorii din categoria Trees din RapidMiner sunt potriviți pentru ambele tipuri de sarcini (clasificare și regresie): Decision Tree, Decision Stump, Random Forest și Gradient Boosted Trees. Ultimii doi construiesc mai multe modele și apoi combină predicțiile (ensembles). Alți operatori potriviți sunt W-RepTree și W-M5P (incluși în extensia Weka).

[2] Poate fi și numeric (vezi nota de subsol anterioară).

8.1. Logica și pașii algoritmului

Terminologie

În științele sociale arborii decizionali sunt priviți tot mai adesea ca o alternativă la algoritmii bazați pe regresie (Robette, 2022). Înainte de a prezenta și ilustra pașii realizării unui arbore decizional, este necesar să discutăm pe scurt despre câteva concepte specifice acestui clasificator: nod (și tipurile de noduri: root, decision, terminal), splitting și pruning (divizare și eliminare), respectiv depth (adâncime / nivele).

Primul tip de nod este **root node**, adică nodul principal sau trunchi. Acesta conține toate cazurile și le împarte în două sau mai multe clase în funcție de atributul care contribuie cel mai mult la creșterea omogenității claselor rezultate (altfel spus, la reducerea entropiei / eterogenității). Omogenitatea este mai mare atunci când o clasă este formată din cât mai multe cazuri de același tip (reciproc, eterogenitatea / entropia este mai mare atunci când o clasă conține aproximativ același număr de cazuri din fiecare tip). În Figura 8.1-1, nodul principal este cel marcat cu fundal verde.

Cel de al doilea tip de nod, **decision node**, adică nod decizional sau ramură, este oricare dintre nodurile situate între nodul principal și nodurile terminale. Simplu spus, un nod decizional este un sub-nod care este divizat în alte noduri (cu excepția nodului principal – și acesta poate fi considerat a fi nod decizional). În Figura 8.1-1, nodurile decizionale au fundal albastru.

Nodul de tip **leaf** (frunză) sau **terminal** este orice nod care nu mai este divizat în alte noduri. Gradul de omogenitate al unui nod terminal este de obicei mai mare comparativ cu nodurile decizionale. În Figura 8.1-1, nodurile decizionale au fundal portocaliu.

Parent node vs child node (nod părinte vs nod copil). Un nod care este divizat în sub-noduri se numește părinte, iar fiecare nod rezultat se numește copil. Evident, un nod poate fi simultan și părinte și copil, funcție de nodul în relație cu care îl evaluăm.

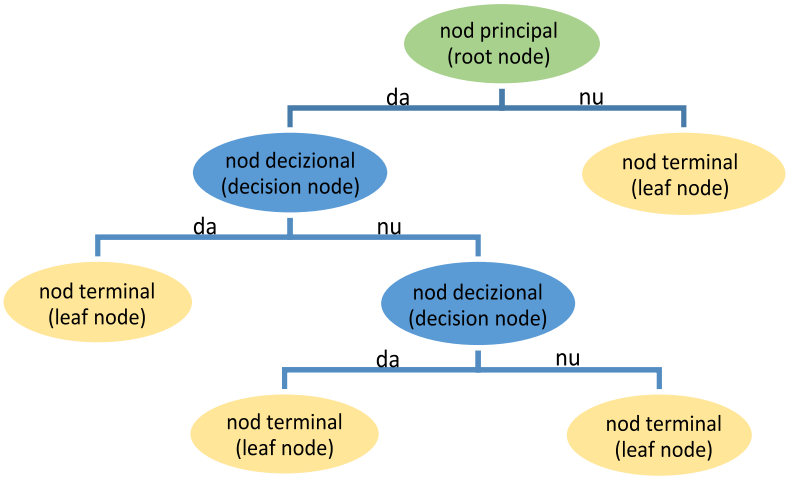
Splitting / partitioning, adică divizare, se referă la procesul de împărțire a unui nod în două sau mai multe sub-noduri. Divizarea are rolul de a reduce sub-potrivirea (underfitting), adică de a crea modele de clasificare / predicție mai performante.

Pruning, adică reducere / simplificare, se referă la procesul de eliminare a unuia sau mai multor noduri (decizionale sau finale) dintr-un arbore decizional. Este opusul

procesului de divizare și are rolul de a reduce supra-ajustarea (overfitting), adică de a produce modele de clasificare / predicție care au șanse mai mari de generalizare.

Maximum depth (adâncime maximă) se referă la numărul maxim de nivele pe care îl poate avea un arbore decizional. Este unul dintre principalii parametri care trebuie definiți anterior construirii unui arbore decizional. Arborele decizional din Figura 8.1-1 are 4 nivele.

Figura 8.1-1. Un exemplu general de arbore decizional



Construirea unui arbore decizional

În această secțiune vom descrie la modul general pașii urmați de algoritmii de construcție a unor arbori decizionali. Scopul urmărit de fiecare algoritm este de a reduce entropia / eterogenitatea la nivelul datelor. La nivelul întregului set de date, entropia asociată variabilei dependente este maximă. Divizând cazurile în sub-grupuri tot mai omogene, funcție de anumite atribute, algoritmul urmărește reducerea entropiei.[3]

Funcție de criteriul de diviziune utilizat, există mai mulți algoritmi de construcție a unor arbori decizionali potriviți pentru probleme de clasificare: algoritmul CART folosește Indexul Gini; algoritmii ID3, C4.5 și C5 folosesc Information Gain și Gain

[3] Un exemplu vizual și dinamic extrem de intuitiv de construcție a unui arbore decizional se poate consulta aici: A visual introduction to machine learning.

Ratio; algoritmul CHAID folosește chi-pătrat. Logica generală de construcție este însă similară și poate fi descrisă sintetic astfel:

1. **calculăm entropia totală** (H), adică entropia asociată atributului de interes (prezis / label) pe tot setul de date de instruire; entropia este o măsură a impurității; alternativ, putem calcula o altă măsură a impurității, respectiv a reversului acesteia, puritatea / omogenitatea, cea mai cunoscută fiind Indexul Gini; acuratețea clasificării (Accuracy) este de asemenea o alternativă;
2. **calculăm entropia asociată fiecăruia dintre predictori** ($H(p_i)$); alternativ, putem folosi alte măsuri;
3. **calculăm câștigul informațional** (Information Gain), adică diferența dintre entropia totală (calculată la pasul 1) și entropia asociată fiecăruia dintre predictori (calculată la pasul 2); alternativ, putem folosi ca măsură câștigul relativ (Gain Ratio); adesea, Gain Ratio e o opțiune mai bună comparativ cu Information Gain;[4]
4. **selectăm predictorul care produce cel mai mare câștig informațional** și împărțim cazurile în funcție de clasele / valorile acestuia; acesta este atributul utilizat la nodul principal;
5. pentru fiecare nod rezultat (determinat la pasul 4), **reluăm procesul** (pașii 1-4);
6. **procesul de dividere se oprește atunci când este satisfăcută cel puțin una dintre condițiile specificate de analist**; condițiile posibile sunt: (1) arborele de decizie atinge adâncimea maximă specificată, (2) nodul obținut conține doar cazuri de același tip (omogenitate maximă), (3) nodul decizional are un număr de cazuri mai mic decât numărul specificat, (4) nodul terminal are un număr de cazuri mai mic decât numărul specificat, (5) creșterea de omogenitate obținută în urma unei divizări este mai mică decât valoarea minimă specificată.

Pentru a înțelege mai bine acest proces prezentăm un exemplu vizual simplu construit pe baza unui set de date cu 10 cazuri și 3 atribute (același set de date utilizat anterior; exemplul complet este descris în sub-capitolul 8.2). Reprezentarea grafică schematică a pașilor parcurși și arborele decizional rezultat apar în Figura 8.1-2 (cele cinci panele).

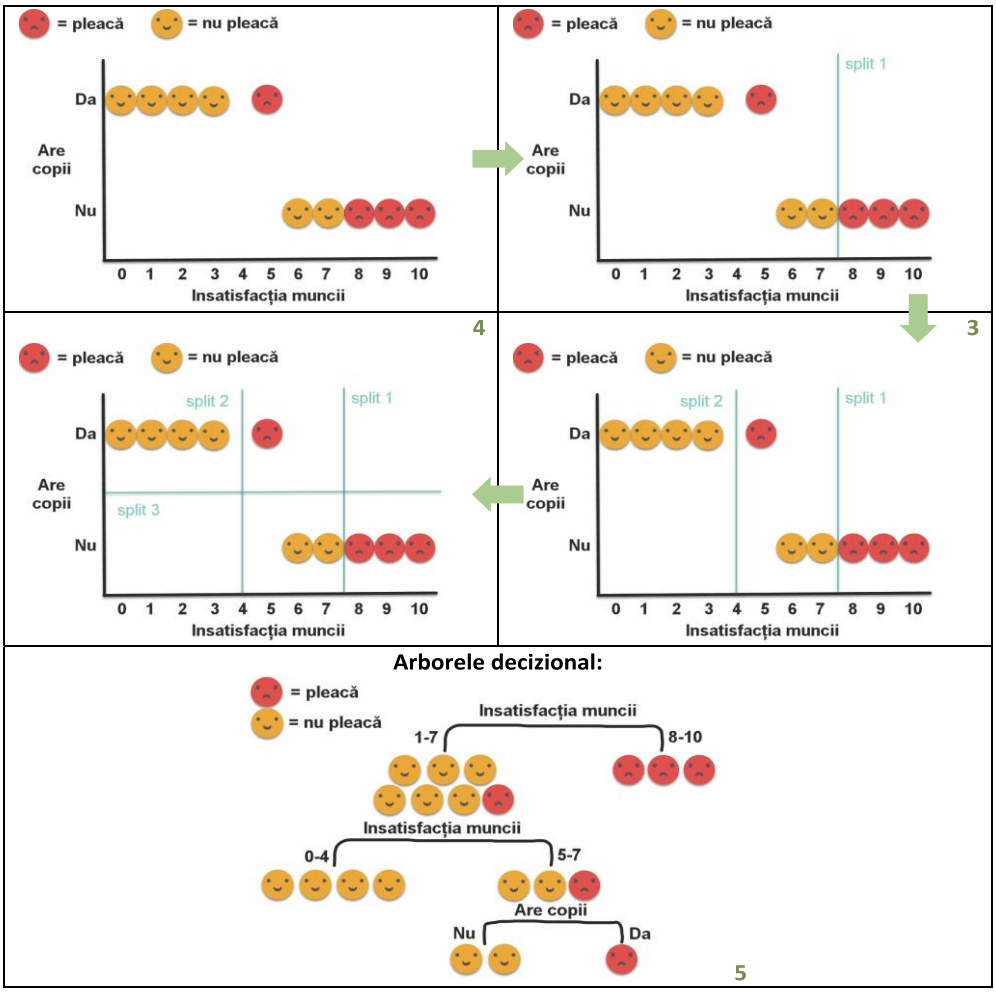
[4] Information Gain favorizează atributele cu mai multe clase, respectiv valori.

În **panelul 1** apar cazurile poziționate în funcție de valorile predictorilor: 5 angajați care pleacă și 5 care nu pleacă, fiecare având un anumit nivel al insatisfacției muncii (scală de la 0 la 10); 4 angajați pleacă (marcați cu roșu), iar 6 rămân (portocaliu). În **panelul 2** observăm că apare prima împărțire a cazurilor, aceasta fiind realizată în funcție de atributul Insatisfacția muncii: cazurile cu un nivel ridicat al insatisfacției (8-10) sunt separate de restul (0-7). Toți cei trei angajați cu insatisfacție ridicată pleacă. Din ceilalți șapte, unul pleacă și șase nu. Calculele care au stat la baza selectării acestui atribut și a acelui prag (7.5) sunt descrise pas cu pas în sub-capitolul 8.2. În **panelul 3** împărțirea se face tot în funcție de insatisfacție, de această dată pragul fiind valoarea 4.5. Rezultă două grupuri: unul cu un nivel redus al insatisfacției (0-4) format din patru angajați care nu pleacă, celălalt cu insatisfacție medie (5-7) format din trei angajați, unul care pleacă și doi care nu pleacă. În **panelul 4** apare a treia împărțire, de această dată în funcție de prezența sau nu a copiilor. Rezultă două noduri terminale omogene, unul format din doi angajați (nu au copii și nu pleacă), altul din unul (are copii și pleacă). Observăm că am plecat de la un grup care conținea aproape același număr de cazuri din fiecare clasă (4 pleacă, 6 nu pleacă) și am ajuns în final la patru grupuri, fiecare grup fiind compus doar din cazuri de același tip. Deci, am plecat de la o eterogenitate mare și am ajuns la o omogenitate maximă.

Panelul 5 prezintă arborele decizional final. Observăm că atributul insatisfacție apare de două ori, o dată ca nod principal și apoi ca nod decizional, iar atributul copii apare pe poziția unui nod decizional. Arborele decizional are patru noduri finale: primul e format din trei angajați care pleacă, al doilea din patru angajați care nu pleacă, al treilea din doi angajați care nu pleacă și ultimul dintr-un angajat care pleacă. Deoarece nu am limitat adâncimea arborelui (numărul de nivele / divizări) fiecare nod final este perfect omogen. În practică, limităm numărul de nivele (sau impunem alte restricții), deci nodurile terminale vor fi foarte rar perfect omogene.

Putem să folosim modelul construit pentru a prezice dacă și alți angajați vor pleca sau nu. De exemplu, vom prezice că un angajat cu un nivel ridicat al insatisfacției (8+) va pleca, că un angajat cu un nivel al insatisfacției de cel mult 4 puncte nu va pleca sau că un angajat cu un scor de insatisfacție 5-7 și cu copii va pleca. Observăm că toate aceste predicții, ultima în special, se bazează pe un număr foarte mic de cazuri, deci ne așteptăm ca generalizarea acestor reguli să fie limitată. Altfel spus, chiar dacă modelul construit de noi are o acuratețe a predicției de 100%, acuratețea reală, calculată pe cazuri noi, va fi mult mai mică. De exemplu, dacă am fi avut mai multe cazuri, am fi observat că angajații cu copii, chiar dacă au un nivel mediu sau ridicat al insatisfacției, vor tinde într-o măsură relativ mai mare să nu plece din companie, ori modelul nostru actual prezice că toți aceștia pleacă.

Figura 8.1-2. Un exemplu simplu de construcție a unui arbore decizional



Cum măsurăm omogenitatea unui nod?

Raportat la atributul de interes, fiecare nod are un anumit nivel al entropiei (omogenității, respectiv eterogenității). Nivelul de omogenitate este măsurat cel mai adesea folosind una dintre măsurile Entropie (H) sau Indicele Gini (G). În cele ce urmează vom prezenta și ilustra aceste două măsuri.

Formula Entropiei pentru situația cu două clase:[5]

$$H = -p_1 * log_2(p_1) - p_2 * log_2(p_2)$$

[5] Formula generală: $H = -\sum_{i=1}^{m} p_i * log_2(p_i)$, unde H = entropia și p_i = probabilitatea clasei i.

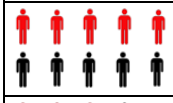
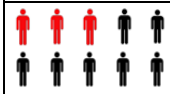
unde H = Entropia, p_1 = probabilitatea clasei 1, p_2 = probabilitatea clasei 2 (1-p_1). Entropia maximă este atinsă atunci când clasele au aceeași pondere. Valoarea teoretică maximă a entropiei crește pe măsură ce numărul claselor crește. Pentru situația cu două clase, valoarea maximă este 1 (pentru 3 clase 1.585, pentru 4 clase 2 etc.). Pentru situația cu două clase, entropia unui nod poate lua valori în intervalul [0;1], unde 0 indică entropia minimă (omogenitate maximă = toate cazurile din acel nod aparțin aceleiași clase), iar 1 entropia maximă (eterogenitate maximă = jumătate dintre cazurile din acel nod aparțin unei clase, jumătate celeilalte).

Formula de calcul a indicelui Gini (Gini Index / Impurity) pentru situația cu două clase este:[6]

$$G = 1 - p_1^2 - p_2^2$$

unde G = indicele Gini, p_1 = probabilitatea clasei 1, p_2 = probabilitatea clasei 2 (1-p_1). Indicele Gini ia valoarea maximă (0.5) atunci când clasele au aceeași pondere, respectiv valoarea minimă (0) atunci când una dintre clase are ponderea de 100%. Valoarea teoretică maximă a indicelui Gini crește pe măsură ce numărul claselor crește. Pentru situația cu două clase, valoarea maximă este 0.5 (pentru 3 clase 0.667, pentru 4 clase 0.75 etc.).

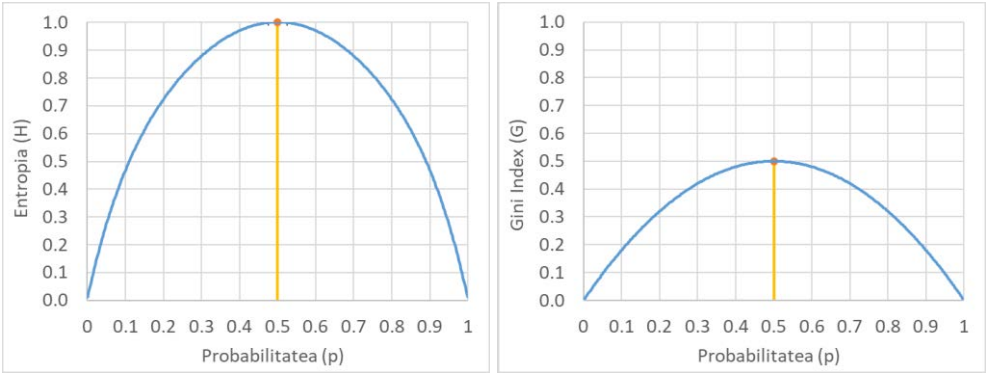
Tabelul 8.1-1. Câteva exemple de calcul a Entropiei și a Indicelui Gini

Observații/Cazuri	Probabilitate	Entropie $H = -p_r * log_2(p_r) - p_n * log_2(p_n)$	Indicele Gini $G = 1 - p_r^2 - p_n^2$
	p(r)=0.5 p(n)=0.5	$H = -0.5 * log_2(0.5) - 0.5 * log_2(0.5)$ $H = -0.5 * (-1) - 0.5 * (-1) = \mathbf{1}$	$G = 1 - 0.5^2 - 0.5^2$ $G = 1 - 0.25 - 0.25 = \mathbf{0.50}$
	p(r)=0.3 p(n)=0.7	$H = -0.3 * log_2(0.3) - 0.7 * log_2(0.7)$ $H = -0.3 * (-1.74) - 0.7 * (-0.51) = \mathbf{0.88}$	$G = 1 - 0.3^2 - 0.7^2$ $G = 1 - 0.09 - 0.49 = \mathbf{0.42}$
	p(r)=0.1 p(n)=0.9	$H = -0.1 * log_2(0.1) - 0.9 * log_2(0.9)$ $H = -0.1 * (-3.33) - 0.9 * (-0.15) = \mathbf{0.47}$	$G = 1 - 0.1^2 - 0.9^2$ $G = 1 - 0.01 - 0.81 = \mathbf{0.18}$
	p(r)=0.0 p(n)=1.0	$H = -0 * log_2(0) - 1 * log_2(1)$ $H = -1 * log_2(1) = -1 * 0 = \mathbf{0}$	$G = 1 - 0^2 - 1^2$ $G = 1 - 0 - 1 = \mathbf{0}$

** două clase: roșu și negru; de la omogenitate minimă (primul exemplu) la omogenitate maximă (ultimul exemplu); formulări alternative: de la eterogenitate maximă (primul exemplu) la eterogenitate minimă (ultimul exemplu); de la entropie maximă (primul exemplu) la entropie minimă (ultimul exemplu).*

[6] Formula generală: $G = 1 - \sum_{i=1}^{m} p_i^2$, unde G = indicele Gini și p_i = probabilitatea clasei i.

Figura 8.1-3. Distribuția valorilor Entropiei și Indicelui Gini în funcție de probabilitatea de apartenență la una dintre clase (situația cu două clase)



Cum putem crește șansele de generalizare a unui arbore decizional?

Dacă setul de date conține atribute metrice și/sau suficiente atribute categoriale și/sau clase definite de unul sau mai multe atribute categoriale, construcția unui arbore decizional se termină în momentul în care fiecare nod terminal este complet omogen. Procedând astfel, relativ la setul de date pe care a fost construit, modelul de predicție va fi unul perfect (acuratețea modelului va fi 100%). Desigur, multe dintre acest noduri vor conține foarte puține cazuri, respectiv vor conține combinații particulare de caracteristici, deci predicțiile făcute în baza unui astfel de model vor avea șanse relativ mai mici să atingă același grad de acuratețe.

Pentru a crește șansele de generalizare a unui arbore decizional, avem la îndemână câteva soluții: (1) reducem adâncimea (numărul de nivele / divizări), (2) permitem ca un nod decizional să se dividă doar dacă numărul de cazuri depășește un anumit prag (pre-pruning), (3) specificăm un număr minim de cazuri pe care trebuie să-l aibă un nod terminal (pre-pruning), (4) setăm un prag minim al încrederii (pruning) și (5) setăm un prag minim al câștigului informațional (pre-pruning).

Avantaje și dezavantaje

Arborii decizionali prezintă o serie de avantaje și dezavantaje. Printre **avantaje**, enumerăm următoarele:

- este simplu de înțeles, intuitiv, ușor de implementat;
- nu este necesar ca predictorii folosiți să aibă o distribuție normală;

- gestionează situațiile în care setul conține valori lipsă, respectiv valori foarte diferite (outliers); ignoră atributele care contribuie prea puțin la calitatea predicției;
- gestionează ușor și automat relații de tip non-liniar, respectiv efecte de interacțiune între variabile;
- predicția cazurilor noi, necunoscute, este rapidă.

Dezavantajele și posibilele soluții relativ la acestea sunt listate în continuare:

- are șanse relativ mai mari de supra-ajustare (overfitting), deci arborele de decizie construit nu va produce predicții la fel de bune pe un set de date nou (probleme de generalizare la date noi); soluția la îndemână constă în reducerea complexității modelului folosind una sau mai multe dintre următoarele: pre-pruning și post-pruning, impunerea unui număr minim de cazuri asociat unui nod intermediar, respectiv final și impunerea unui număr maxim de divizări;
- varianța este relativ mai mare, adică, modificări relativ mici ale setului de date pot produce rezultate diferite; vezi soluțiile propuse la punctul anterior;
- faza de instruire este relativ mai costisitoare (timp și resurse); soluții: reducerea datelor și identificarea predictorilor relevanți, rezultatul fiind reducerea complexității modelului;
- uneori arborii decizionali sunt foarte stufoși, deci e dificil să-i interpretăm; vezi soluțiile propuse la punctul anterior.

8.2. Un exemplu didactic

Pentru acest exemplu vom folosi același set de date cu trei atribute și 10 cazuri (Figura 8.2-1). Atributul de interes este „Pleacă”, iar predictorii sunt „Insatisfacția muncii” și „Are copii” (Figura 8.2-1). Modelul include doar doi operatori: Retrieve (încarcă setul de date) și Decision Tree (clasificatorul). Acest model este realizat în procesul dt_didactic_1 din folderul Exemple. Criteriul ales pentru construcția arborelui este Gain Ratio. Reamintim faptul că obiectivul urmărit este reducerea entropiei (creșterea omogenității grupurilor, adică producerea unor grupuri care să conțină cât mai multe cazuri din aceeași clasă a variabilei dependente) prin divizarea cazurilor.

Observăm că prima separare (nivelul 1 de adâncime) este realizată în funcție de atributul Insatisfacție. Cei 10 angajați din setul de date sunt împărțiți în două noduri:

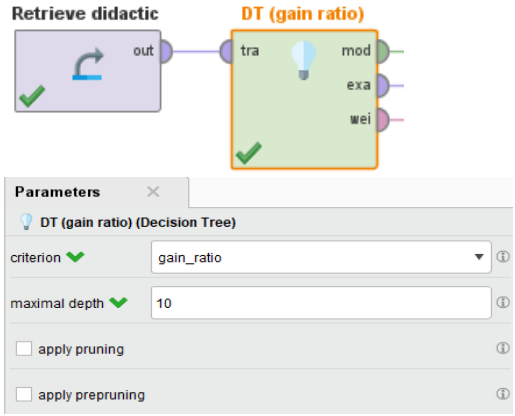
un nod este format din trei angajați cu un nivel al insatisfacției ridicat (8-10) care pleacă (deci avem un nod terminal perfect omogen), celălalt nod, decizional, include restul de șapte angajați (unul care pleacă și 6 care nu pleacă). La nivelul doi de adâncime, separarea se face tot în funcție de Insatisfacție, rezultând două noduri, unul decizional (trei angajați, unul pleacă și doi rămân) și unul final (patru angajați cu insatisfacția în intervalul 0-4 care nu pleacă). Nodul decizional rezultat anterior este divizat mai departe în funcție de atributul Copii, rezultând două noduri finale perfect omogene: unul include un angajat cu copii care pleacă, iar celălalt doi angajați care nu au copii și nu pleacă.

Figura 8.2-1. Un exemplu didactic de arbore decizional în RapidMiner

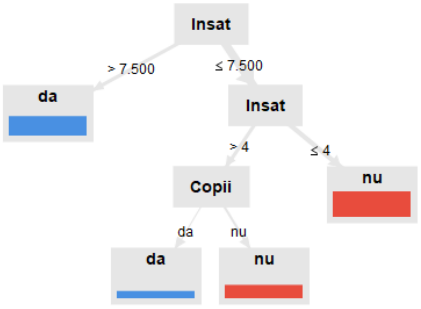
Setul de date:

Row No.	Id	Pleacă	Copii	Insat
1	1	da	nu	10
2	2	da	nu	9
3	3	da	nu	8
4	4	nu	nu	7
5	5	nu	nu	6
6	6	da	da	5
7	7	nu	da	3
8	8	nu	da	2
9	9	nu	da	1
10	10	nu	da	0

Modelul:



Arborele decizional: Grafic



Arborele decizional: Descriere (text)

Insat > 7.500: da {da=3, nu=0}
Insat ≤ 7.500
| Insat > 4
| | Copii = da: da {da=1, nu=0}
| | Copii = nu: nu {da=0, nu=2}
| Insat ≤ 4: nu {da=0, nu=4}

Cei interesați de modul în care algoritmul a ales atributul în funcție de care a împărțit cazurile pot consulta următoarele două secțiuni. Aici reamintim doar faptul că algoritmul urmărește creșterea omogenității cazurilor prin divizarea lor în grupuri folosind diferite criterii / măsuri precum Entropia (H), câștigul informațional (Information Gain - IG), câștigul relativ (Gain Ratio - GR) și Indicele Gini (G).

Entropia (H), câștigul informațional (IG) și câștigul relativ (GR)

În cazul atributelor categoriale, calcularea măsurilor este relativ mai simplă (Tabelul 1.2-2). Primul pas constă în calcularea entropiei totale, la nivelul întregului set de date. Pentru cazul nostru entropia totală (H) este aproape maximă (0.97). În continuare, calculăm entropia asociată cu fiecare dintre clasele predictorului categorial și obținem valoarea 0.72 pentru clasa „cu copii", respectiv 0.97 pentru clasa „fără copii".[7] Entropia ponderată asociată atributului Copii este calculată ca medie ponderată (cu proporția claselor) a celor două valori obținute anterior și este egală cu 0.85. Observăm că această valoare este mai mică comparativ cu valoarea de start - entropia finală este mai mică cu 0.12. Aceasta este de altfel și valoarea măsurii câștig informațional (Information Gain - IG). Dat fiind faptul că măsura câștig informațional depinde de numărul de clase / valori ale predictorului (corelație pozitivă; IG este distorsionat = biased), pentru a obține o măsură mai corectă împărțim câștigul informațional la entropia predictorului. Obținem astfel o măsură comparabilă între atribute, câștigul informațional relativ (Gain Ratio - GR). În cazul de față, deoarece Entropia atributului Copii este 1, valorile Gain Ratio și Information Gain sunt identice (0.12).

Tabelul 8.2-1. Exemplu de calcul a Entropiei în cazul unui atribut binominal

Eșantion	**Observații / Cazuri**	**Probabilitate**	**Entropia** $H = -p_r * log_2(p_r) - p_n * log_2(p_n)$
Total		p(r)=0.4 p(n)=0.6	$H = -0.4 * log_2(0.4) - 0.6 * log_2(0.6)$ $H = 0.4 * 1.32 + 0.6 * 0.74 = \mathbf{0.97}$
Cu copii (W=5/10=**0.5**)		p(r)=0.2 p(n)=0.8	$H = -0.2 * log_2(0.2) - 0.8 * log_2(0.8)$ $H = 0.2 * 2.32 + 0.8 * 0.32 = \mathbf{0.72}$
Fără copii (W=5/10=**0.5**)		p(r)=0.6 p(n)=0.4	$H = -0.6 * log_2(0.6) - 0.4 * log_2(0.4)$ $H = 0.6 * 0.74 + 0.4 * 1.32 = \mathbf{0.97}$
Entropia condiționată de atributul „Are copii" =			$H = W_{copii} * G_{copii} + W_{fără\ copii} * G_{fără\ copii}$ $G = 0.5 * 0.72 + 0.5 * 0.97 = 0.16 + 0.24 =$

[7] Dacă predictorul categorial are mai mult de două clase, măsurile de omogenitate vor fi calculate pentru fiecare combinație posibilă de clase (subseturi). De exemplu, pentru un predictor cu clasele {a, b și c}, algoritmul calculează diferența dintre {a} și {b, c}, {b} și {a, c}, respectiv {c} și {a, b}.

Eșantion	Observații / Cazuri	Probabilitate	Entropia $H = -p_r * log_2(p_r) - p_n * log_2(p_n)$
Câștig informațional (Information Gain) asociat atributului „Are copii" =			$IG = H_{total} - H_{cu\ predictor}$ $IG = 0.97 - 0.85 = \mathbf{0.12}$
Câștig informațional relativ (Gain Ratio) asociat atributului „Are copii"[8] =			$GR = IG/H_{predictor}$ $GR = 0.12/1 = \mathbf{0.12}$

În cazul unui predictor metric procedăm la fel (Tabelul 8.2-2), dar, suplimentar, testăm toate posibilele divizări. De exemplu, în cazul predictorului Insatisfacție, vom testa pragurile 9.5, 8.5, 7.5 etc. (mediile a două valori adiacente). Pentru fiecare prag vom calcula valorile entropiei și măsurilor derivate din aceasta. La final vom alege pragul care produce cea mai mare reducere a entropiei. În cazul de față, pragul 9.5 are o Entropie de 0.83, deci produce un câștig informațional egal cu 0.14, respectiv un câștig informațional relativ egal cu 0.31; pentru pragul de 8.5 aceleași măsuri iau valorile 0.65, 0.32 și 0.45; pentru pragul 7.5, Entropia este 0.41, deci produce un câștig informațional egal cu 0.56, respectiv un câștig informațional relativ egal cu 0.63. Similar, calculăm Entropia (restul măsurilor) și pentru celelalte praguri[9] și observăm că cea mai mică valoare este 0.41. Această valoare este obținută pentru pragul 7.5, deci decidem să divizăm cazurile după atributul Insatisfacție în două clase, una cu valorile 8-10, cealaltă cu valorile 0-7.

Tabelul 8.2-2. Exemplu de calcul a Entropiei în cazul unui atribut metric

Prag insatisfacție = 9.5	Probabilitate	Entropia
Insatisfacție = Da (W=1/10=**0.1**)	p(r)=1.0 p(n)=0.0	$H = -1 * log_2(1) - 0 * log_2(0) = 0$
Insatisfacție = Nu (W=9/10=**0.9**)	p(r)=0.33 p(n)=0.67	$H = -0.33 * log_2(0.33) - 0.67 * log_2(0.67) = 0.92$
Entropia (prag 9.5) =		$H = 0.1 * 0 + 0.9 * 0.92 = \mathbf{0.83}$
Câștig informațional (Information Gain)		$IG = 0.97 - 0.83 = \mathbf{0.14}$
Câștig informațional relativ (Gain Ratio)[10]		$GR = 0.14/0.47 = \mathbf{0.31}$

[8] Dat fiind faptul că avem același număr de angajați cu / fără copii, entropia atributului Copii este 1.

[9] Restul valorilor sunt în ordine 0.71, 0.97, 0.55, 0.69, 0.80 și 0.89.

[10] Pentru acest prag atributul Insatisfacție are două clase, una cu un caz, alta cu nouă cazuri, deci entropia atributului Insatisfacție este 0.47.

Prag insatisfacție = 8.5	**Probabilitate**	**Entropia**
Insatisfacție = Da (W=2/10=**0.2**)	p(r)=1.0 p(n)=0.0	$H = -1 * log_2(1) - 0 * log_2(0) = 0$
Insatisfacție = Nu (W=8/10=**0.8**)	p(r)=0.25 p(n)=0.75	$H = -0.25 * log_2(0.25) - 0.75 * log_2(0.75) = 0.81$
Entropia (prag 8.5) =		$H = 0.2 * 0 + 0.8 * 0.81 = \mathbf{0.65}$
Câștig informațional (Information Gain)		$IG = 0.97 - 0.65 = \mathbf{0.32}$
Câștig informațional relativ (Gain Ratio)[11]		$GR = 0.32/0.72 = \mathbf{0.45}$

Prag insatisfacție = 7.5	**Probabilitate**	**Entropia**
Insatisfacție = Da (W=3/10=**0.3**)	p(r)=1.0 p(n)=0.0	$H = -1 * log_2(1) - 0 * log_2(0) = 0$
Insatisfacție = Nu (W=7/10=**0.7**)	p(r)=0.14 p(n)=0.86	$H = -0.14 * log_2(0.14) - 0.86 * log_2(0.86) = 0.59$
Entropia (prag 8.5) =		$H = 0.3 * 0 + 0.7 * 0.59 = \mathbf{0.41}$
Câștig informațional (Information Gain)		$IG = 0.97 - 0.41 = \mathbf{0.56}$
Câștig informațional relativ (Gain Ratio)[12]		$GR = 0.56/0.88 = \mathbf{0.63}$

Dintre cei doi predictori considerați, Insatisfacția are cea mai mică valoare a Entropiei (0.41 vs 0.85 pentru atributul Copii). Câștigul informațional este mai mare în cazul Insatisfacției (0.56 vs 0.12 pentru Copii). Câștigul informațional relativ este de asemenea mai mare în cazul Insatisfacției (0.63 vs 0.12 pentru Copii). Prin urmare, cazurile vor fi divizate prima dată în funcție de predictorul Insatisfacție.

Pentru a vedea care este următorul atribut în funcție de care divizăm cazurile, reluăm procesul și observăm că cea mai mare reducere a Entropiei este produsă tot de atributul Insatisfacție, de această dată pentru pragul 4. În final, mai divizăm o dată cazurile rămase în funcție de atributul Copii (nu am mai prezentat aici calculele).

[11] Pentru acest prag atributul Insatisfacție are două clase, una cu două cazuri, alta cu opt cazuri, deci entropia atributului Insatisfacție este 0.72.

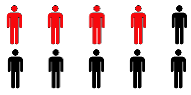
[12] Pentru acest prag atributul Insatisfacție are două clase, una cu trei cazuri, alta cu șapte cazuri, deci entropia atributului Insatisfacție este 0.88.

Indicele Gini (G)

Procedura descrisă în cazul Entropiei se aplică și în cazul Indicelui Gini. Valoarea indicelui Gini la nivelul întregului set de date este 0.48 (aproape de maximul de 0.5). Dacă luăm în calcul doar atributul Copii, indicele Gini scade la 0.40 (Tabelul 8.2-3). Dacă luăm în calcul doar atributul Insatisfacție, valoarea indicelui Gini diferă în funcție de prag. Pentru pragul 9.5 Gini ia valoarea 0.4, pentru 8.5 0.3, iar pentru 7.5 valoarea 0.17 (Tabelul 8.2-4). Similar, calculăm indexul Gini și pentru celelalte praguri[13] și observăm că cea mai mică valoare este 0.17. Această valoare este obținută pentru pragul 7.5, deci decidem să împărțim cazurile după atributul Insatisfacție în două clase, una cu valorile 8-10, cealaltă cu valorile 0-7.

Deoarece cea mai mare scădere este observată în cazul atributului Insatisfacție (0.17 vs 0.4 pentru atributul Copii), acesta va fi și primul atribut utilizat pentru dividerea cazurilor. Următoarea dividere va fi realizată tot în funcție de Insatisfacție (dar cu un prag <= 4), iar ultima în funcție de atributul Copii.

Tabelul 8.2-3. Exemplu de calcul a indicelui Gini în cazul unui atribut binominal

Eșantion	**Observații / Cazuri**	**Probabilitate**	**Indicele Gini** $G = 1 - p_r^2 - p_n^2$
Total		p(r)=0.4 p(n)=0.6	$G = 1 - 0.4^2 - 0.6^2$ $G = 1 - 0.16 - 0.36 = \mathbf{0.48}$
Cu copii (W=5/10=**0.5**)		p(r)=0.2 p(n)=0.8	$G = 1 - 0.2^2 - 0.8^2$ $G = 1 - 0.04 - 0.64 = \mathbf{0.32}$
Fără copii (W=5/10=**0.5**)		p(r)=0.6 p(n)=0.4	$G = 1 - 0.6^2 - 0.4^2$ $G = 1 - 0.36 - 0.16 = \mathbf{0.48}$
Index Gini asociat atributului "Are copii" =			$G = W_{copii} * G_{copii} + W_{fără\ copii} * G_{fără\ copii}$ $G = 0.5 * 0.32 + 0.5 * 0.48 = 0.16 + 0.24 = \mathbf{0.40}$

[13] Restul valorilor sunt în ordine 0.32, 0.48, 0.27, 0.34, 0.40 și 0.44.

Tabelul 8.2-4. Exemplu de calcul a indicelui Gini în cazul unui atribut metric

Prag insatisfacție = 9.5	**Probabilitate**	**Indicele Gini**
Insatisfacție = Da (W=1/10=**0.1**)	p(r)=1.0 p(n)=0.0	$G = 1 - 1^2 - 0^2 = 0$
Insatisfacție = Nu (W=9/10=**0.9**)	p(r)=0.33 p(n)=0.67	$G = 1 - 0.33^2 - .067^2 = 0.44$
	Indicele Gini (prag 9.5) =	$G = 0.1 * 0 + 0.9 * 0.44 = \mathbf{0.40}$

Prag insatisfacție = 8.5	**Probabilitate**	**Indicele Gini**
Insatisfacție = Da (W=2/10=**0.2**)	p(r)=1.0 p(n)=0.0	$G = 1 - 1^2 - 0^2 = 0$
Insatisfacție = Nu (W=8/10=**0.8**)	p(r)=0.25 p(n)=0.75	$G = 1 - 0.25^2 - 0.75^2 = 0.38$
	Indicele Gini (prag 8.5) =	$G = 0.2 * 0 + 0.8 * 0.38 = \mathbf{0.30}$

Prag insatisfacție = 7.5	**Probabilitate**	**Indicele Gini**
Insatisfacție = Da (W=3/10=**0.3**)	p(r)=1.0 p(n)=0.0	$G = 1 - 1^2 - 0^2 = 0$
Insatisfacție = Nu (W=7/10=**0.7**)	p(r)=0.14 p(n)=0.86	$G = 1 - 0.14^2 - 0.86^2 = 0.24$
	Indicele Gini (prag 7.5) =	$G = 0.3 * 0 + 0.7 * 0.24 = \mathbf{0.17}$

.....

Reducerea entropiei vs. importanța unui predictor

În Tabelul 8.2-5 am prezentat sintetic rezultatele obținute cu privire la alegerea primului atribut în funcție de care să fie făcută împărțirea cazurilor. În acest caz, decizia este aceeași indiferent de tipul de măsură utilizată. Toate aceste măsuri, plus acuratețea clasificării, la nivelul întregului model (cel cu toate divizările), pot fi calculate în RapidMiner folosind procesul dt_didactic_2 din folderul Exemple. După cum se observă, valorile sunt identice.

Tabelul 8.2-5. Sinteza rezultatelor: reducerea entropiei / impurității

Atribut	**Entropie**	**Câștig Informațional**	**Câștig Relativ**	**Indicele Gini**	**Diferență Gini**
Calcul manual					
Total	0.97	-	-	0.48	-
Copii	0.85	0.12	0.12	0.40	0.08
Insat	0.41	0.56	0.63	0.17	0.31

Atribut	Entropie	Câștig Informațional	Câștig Relativ	Indicele Gini	Diferență Gini
Calcul automat cu operatorul Weight by					
Weight by	**-**	**Info Gain**	**Gain Ratio**		**Gini Index**
Copii	-	0.12	0.12		0.08
Insat	-	0.56	0.63		0.31

Faptul că un atribut a fost utilizat la un moment dat pentru a divide cazurile nu înseamnă că restul atributelor nu ar putea contribui la calitatea predicției. Dacă un atribut X produce un câștig informațional doar puțin mai mare comparativ cu alt atribut, divizarea se va face în funcție de atributul X, însă acest lucru nu înseamnă că celălalt atribut nu poate avea o importanță comparabilă relativ la predicția variabilei dependente. Aceasta pare să fie și situația în cazul acestui exemplu.

Chiar dacă atributul Insatisfacție a fost folosit pentru primele două divizări, iar atributul Copii a fost utilizat doar în cazul ultimei divizări, cele două atribute au o importanță apropiată din punct de vedere al capacității lor predictive (Copii chiar puțin mai mare) (Tabelul 8.2-6). Logica importanței este simplă: care ar fi fost contribuția unui atribut la reducerea entropiei dacă acel atribut ar fi fost utilizat pentru dividerea cazurilor. Pentru a calcula automat importanța atributelor, e suficient să conectăm portul wei (weights = importanță / ponderări) asociat operatorului Decision Tree cu portul res (rezultate). În Tabelul 8.2-6 am calculat importanța pentru toate cele patru criterii de dividere disponibile în RapidMiner. Observăm că valorile sunt identice (în general sunt apropiate), respectiv că suma celor două ponderi este 1 (nu este întâmplător; datorită metodei de calcul, indiferent de numărul predictorilor și relațiile dintre aceștia, suma va fi întotdeauna 1).

Situațiile de acest tip (un atribut este utilizat mai frecvent pentru construirea sub-grupurilor, dar importanța lui e apropiată de importanța altor atribute) apar din cauza multicoliniarității (corelației mari) dintre predictori. Într-adevăr, pentru acest set de date didactic, corelația dintre atributele Copii și Insatisfacție este foarte mare (-0.88).

Tabelul 8.2-6. Importanța predictorilor

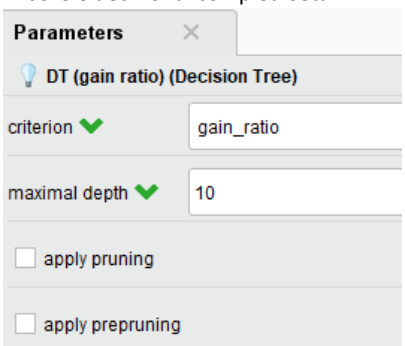
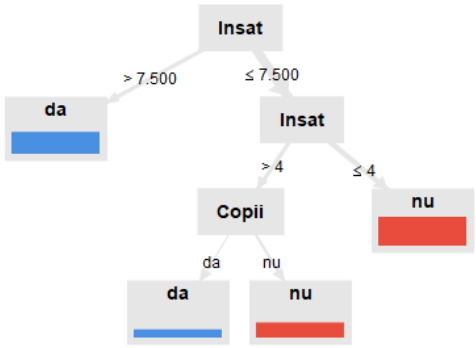
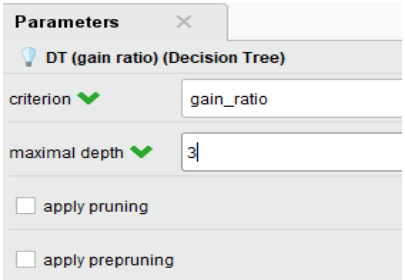
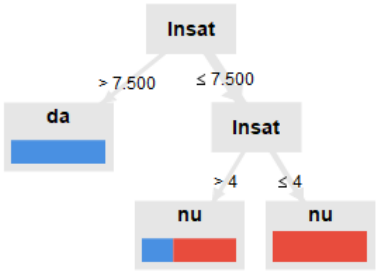
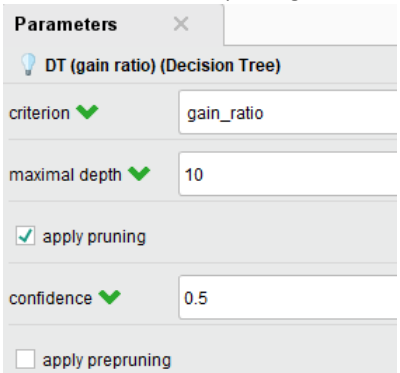
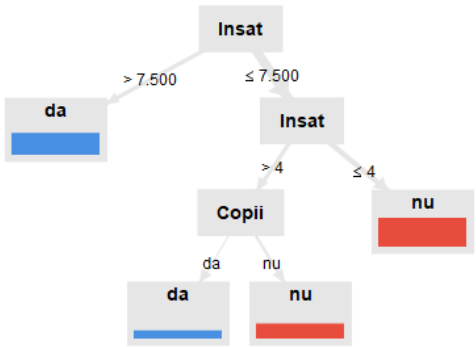
Atribut	Accuracy	Info Gain	Gain Ratio	Gini Index
Copii	0.55	0.55	0.55	0.55
Insat	0.45	0.45	0.45	0.45

Simplificarea arborelui decizional

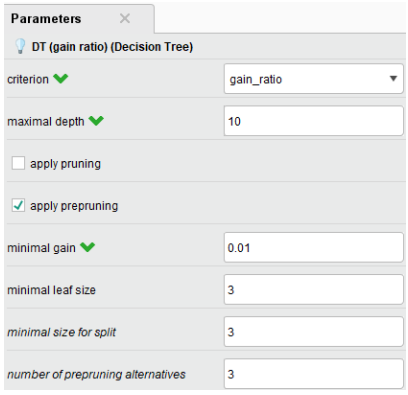
Dat fiind faptul că exemplul nostru conține doar 10 cazuri și trei atribute, arborele decizional nu se schimbă prea mult indiferent de tipul de modificare ales (Figura

8.2-2). Singura diferență care apare în cazul unora dintre setări este dispariția nivelului patru, adică divizarea în funcție de atributul Copii. Fără această divizare, unul dintre nodurile terminale, cel compus din trei angajați cu un nivel al insatisfacției de 5-7 puncte, nu mai este perfect omogen (doi angajați pleacă și unul rămâne). Simplificarea are rolul de a reduce diferența dintre predicția obținută pe setul de date de instruire și predicția obținută pe setul de date de test. Simplu spus, ne dorim un model de predicție care să nu fie afectat de supra-ajustare (overfitting).

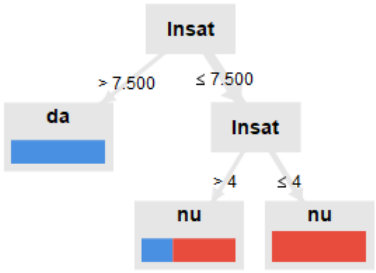
Figura 8.2-2. Simplificarea arborelui decizional

<table>
<tr><td>Arborele decizional complet: setări</td><td>Arborele decizional complet: grafic</td></tr>
<tr><td> 
</td><td> 
</td></tr>
<tr><td>Arborele decizional cu 3 nivele: setări</td><td>Arborele decizional cu 3 nivele: grafic</td></tr>
<tr><td> 
</td><td> 
</td></tr>
<tr><td>Arborele decizional cu pruning: setări</td><td>Arborele decizional cu pruning: grafic</td></tr>
<tr><td> 
</td><td> 
</td></tr>
</table>

Arborele decizional cu prepruning: setări



Arborele decizional cu prepruning: grafic



8.3. Exemple de utilizare a arborilor decizionali în RapidMiner

În această secțiune vom prezenta două exemple de analiză în care am folosit operatorul Decision Tree din RapidMiner (Modeling > Predictive > Trees).[14] Pe lângă acest operator, RapidMiner include o serie de alți operatori care funcționează într-o logică similară: CHAID, ID3, Random Trees, Random Forest, Gradient Boosted Trees.[15] Extensia Weka include o serie de alți operatori.

Un model simplu

Pentru acest exemplu vom folosi aceleași trei atribute din setul de date „employee_attrition" și vom realiza un model de clasificare de tip arbore decizional. Scopul analizei este predicția situațiilor de părăsire a companiei în funcție de

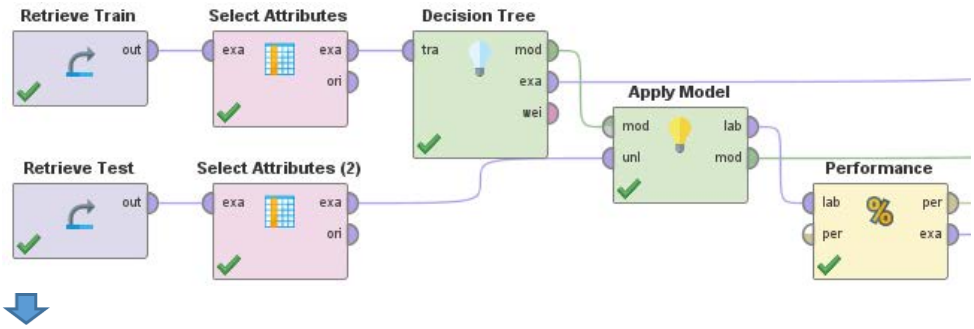
[14] Pentru alte exemple se pot consulta diferite texte de specialitate (Kotu & Deshpande, 2019, Chapter 4; Shmueli et al., 2023, Chapter 9).

[15] Operatorii CHAID și ID3 pot fi folosiți doar în cazul atributelor de tip categorial (criteriul de divizare utilizat de CHAID este chi-pătrat; ID3 nu folosește pruning). Operatorul Random Trees diferă de Decision Tree doar prin faptul că la fiecare divizare ia în considerare doar o parte dintre atribute, selectate aleator. Operatorul Random Forest produce o serie de arbori decizionali construiți pe diferite sub-seturi de date selectate aleator din setul analizat, arborele decizional final fiind ales prin vot, prin urmare este mai puțin afectat de supra-ajustare. Operatorul Gradient Boosted Trees, un meta-algoritm, construiește succesiv o serie de arbori decizionali, fiecare arbore nou urmărind reducerea erorilor de predicție făcute de arborele anterior.

vechimea în muncă și munca peste program. Firesc, ne așteptăm ca o vechime mai mare să reducă șansele de plecare, iar munca peste program să le crească. Pașii realizării acestui model în RapidMiner, rezultatele obținute și interpretarea lor apar în Figura 8.3-1.

Figura 8.3-1. Arbore decizional: un model simplu în RapidMiner

Pasul 1:
Realizăm procesul din imaginea de mai jos (procesul și datele utilizate sunt incluse în folderul asociat volumului). Seturile de date utilizate sunt „employee_attrition_training_validation" (instruire), respectiv „employee_attrition_testing" (testare). Relativ la ambele seturi de date am păstrat doar atributele Attrition, OverTime și YearsAtCompany (operatorul „Select Attributes"). Includem operatorul Decision Tree cu setările originale apoi restul operatorilor din imagine. Rulăm modelul.



Rezultate (setul de instruire):
Seturile de date conțin doar atributele selectate anterior (în imagine apar primele 5 cazuri din setul de instruire). Distribuția statistică a atributului de interes – Attrition – este 0.161 pentru clasa Yes (pleacă) și 0.839 pentru clasa No (nu pleacă).

Row No.	Id	Attrition	OverTime	YearsAtCompany
1	4	Yes	Yes	0
2	5	No	Yes	8
3	7	No	No	2
4	10	No	Yes	1
5	14	No	No	5

Nominal values

Ind...	Nominal value	Absolute count	Fraction
1	No	863	0.839
2	Yes	166	0.161

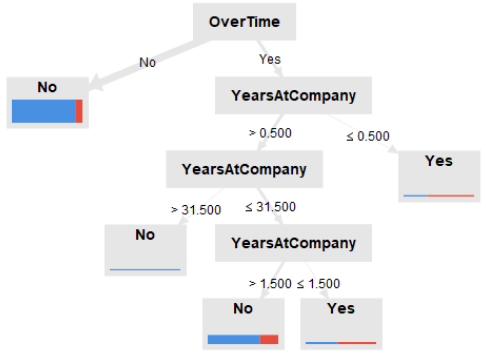
Close

Rezultate (setul de testare):
Relativ la setul de testare, modelul produce predicțiile din imagine (am inclus primele 5 cazuri și alte trei de interes în contextul acestui exemplu). Coloana Attrition prezintă situația reală, iar coloana „prediction(Attrition)" predicția realizată de model. Observăm că o parte dintre predicții sunt greșite (de exemplu cele cu Id 1, 22, 27). Probabilitățile asociate predicțiilor apar pe coloanele „confidence". Relativ la fiecare caz, clasa care are asociată cea mai mare probabilitate (confidence) prezisă, determină clasa la care modelul prezice că aparține acel caz. Pragul de probabilitate implicit în funcție de care un caz este prezis a aparține unei clase sau alteia este 50%. Astfel, dacă

probabilitatea este sub 50% modelul prezice că acel caz aparține clasei No (angajatul nu pleacă). Dacă probabilitatea este cel puțin 50%, clasa prezisă va fi Yes (angajatul pleacă). Cazurile au asociate diferite probabilități de a pleca. Cea mai mică probabilitate estimată este 0% – angajatul 1352 – are o vechime de 33 ani și muncește peste program. Cea mai mare probabilitate, 0.667 (66.7%), apare în cazul angajaților 167 și 811 (amândoi au o vechime mai mică de un an și muncesc peste program). Dat fiind faptul că pragul de probabilitate implicit este 0.5, modelul prezice că angajații 167 și 811 vor pleca.

Row No.	Id	Attrition	prediction(Attrition)	confidence(No)	confidence(Yes)	OverTime	YearsAtCompany
1	1	Yes	No	0.743	0.257	Yes	6
2	2	No	No	0.898	0.102	No	10
3	8	No	No	0.898	0.102	No	7
4	11	No	No	0.898	0.102	No	1
5	12	No	No	0.898	0.102	No	9
...							
41	167	Yes	Yes	0.333	0.667	Yes	0
...							
293	1352	No	No	1	0	Yes	33
...							

Arborele decizional apare la secțiunea Tree(Decision Tree). La tabul Graph este prezentată varianta grafică a regulilor, iar la Description varianta text. Observăm că prima divizare a cazurilor este în funcție de atributul OverTime, iar următoarele în funcție de YearsAtCompany. Modelul are patru nivele, nodurile rezultate fiind cel mai adesea dominate de una dintre cele două categorii (pleacă / nu pleacă). Dacă un angajat nu muncește peste programul, predicția este că angajatul nu pleacă (din cei 719 angajați care nu muncesc peste program, 646 nu pleacă). Dacă angajatul muncește peste program, predicția variază în funcție de vechimea acestuia în companie. Astfel, angajații foarte recenți (un an sau mai puțin) tind mai degrabă să plece, iar cei cu vechime de cel puțin doi ani tind să rămână.



Tree

```
OverTime = No: No {No=646, Yes=73}
OverTime = Yes
|   YearsAtCompany > 0.500
|   |   YearsAtCompany > 31.500: No {No=4, Yes=0}
|   |   YearsAtCompany ≤ 31.500
|   |   |   YearsAtCompany > 1.500: No {No=194, Yes=67}
|   |   |   YearsAtCompany ≤ 1.500: Yes {No=15, Yes=18}
|   YearsAtCompany ≤ 0.500: Yes {No=4, Yes=8}
```

Secțiunea PerformanceVector conține matricea de confuzie și diferite măsuri ale performanței clasificării (cele selectate) calculate relativ la setul de date de testare. Acuratețea predicției este mare (84%) doar pentru că majoritatea angajaților nu pleacă. Relativ la clasa de interes pentru noi, cei care pleacă, calitatea predicției e destul de slabă: precizia clasificării acestei clase este 50%

(jumătate dintre cazurile prezise că pleacă chiar pleacă), iar reamintirea doar 17% (modelul identifică doar 17% dintre angajații care pleacă efectiv). Pragul implicit al clasificării este 50%. Pentru a crește calitatea modelului relativ la clasa de interes, putem reduce aceste prag (Threshold) la 30% de exemplu și să clasificăm toate cazurile cu probabilitatea prezisă de a pleca de cel puțin 30% ca angajați care pleacă. Desigur, în acest caz ponderea erorilor de tip I va crește.

accuracy: 83.90%

	true No	true Yes	class precision
pred. No	358	59	85.85%
pred. Yes	12	12	50.00%
class recall	96.76%	16.90%	

set testare, prag implicit 50%

Un model relativ mai complex

Înainte de a trece la prezentarea și interpretarea acestui model, e necesar să descriem pe scurt parametrii asociați operatorului Decision Tree (Figura 8.3-2):

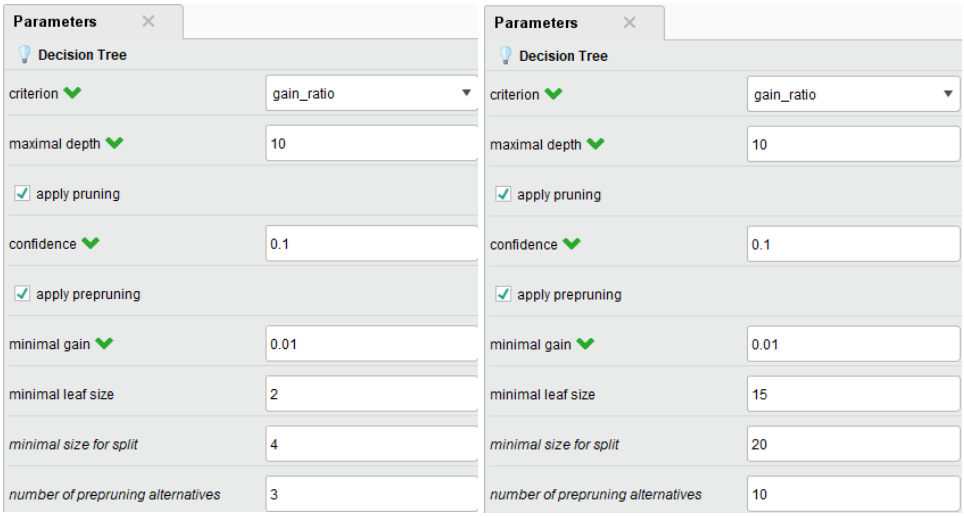
- Parametrul **criterion** se referă la criteriul utilizat pentru selecția atributelor și divizarea cazurilor. Criteriul recomandat este Gain Ratio, dar putem alege și altul: Information Gain, Gini Index sau Accuracy (Least Square e potrivit doar în cazul în care atributul prezis e de tip metric).
- **Maximal depth** permite setarea numărului maxim de nivele permis. Cu cât acest număr este mai mare, cu atât arborele decizional rezultat va fi mai complex și, probabil, mai performant dar cu șanse mai mari să fie afectat de supra-ajustare.
- Dacă selectăm parametrul **pruning**, modelul final va fi o versiune simplificată a modelului rezultat în urma analizei (reducerea complexității se realizează după ce arborele decizional a fost crescut maximal). RapidMiner folosește tipul de pruning numit pesimist:[16] consideră că eroarea reală e întotdeauna mai mare decât cea estimată; prin urmare eroarea reală va fi egală cu eroarea estimată plus o penalitate; penalitatea

[16] Simplificarea de tip pesimist parcurge următorii pași: (1) pornim de la primul atribut utilizat pentru divizarea cazurilor; (2) eliminăm nodul din stânga; dacă eroarea scade eliminăm nodul, altfel îl păstrăm; (3) procedăm la fel cu nodul din dreapta; (4) reluăm pașii 2 și 3 în cazul nodurilor decizionale de la nivelul următor; ne oprim când procedura nu mai duce la eliminarea altor noduri; am ajuns astfel la cel mai simplu arbore decizional posibil. Există și alți algoritmi de pruning. Unul dintre aceștia folosește drept criteriu costul complexității: eroarea pe setul de instruire scade pe măsură ce crește numărul de noduri, dar eroarea pe setul de test scade doar până la un punct (minimum error tree), apoi crește; punctul respectiv e locul în care se aplică pruning.

asociată unui sub-arbore este egală cu numărul de noduri terminale împărțit la dublul numărului de cazuri. Dacă am selectat pruning, putem seta nivelul de încredere (**confidence**) care va fi folosit pentru calcularea erorii pesimiste.

- Similar, selectarea parametrului **prepruning** va avea ca efect tot simplificarea modelului final doar că, de această dată, se previne creșterea în complexitate a arborelui decizional. Pentru a defini modul în care se realizează prevenirea creșterii în complexitate avem la dispoziție o serie de parametrii: câștigul minim necesar a fi obținut în urma unei divizări (**minimal gain**); numărul minim de cazuri pe care trebuie să-l aibă un nod terminal pentru a putea fi divizat (**minimal leaf size**); numărul minim de cazuri pe care trebuie să-l aibă un nod decizional pentru a putea fi divizat (**minimal size for split**); numărul de soluții alternative testate atunci când un nod este divizat (**number of prepruning alternatives**). Complexitatea unui arbore decizional crește atunci când parametrii minimal gain, minimal leaf size și minimal size for split iau valori relativ mai mici, respectiv parametrul number of prepruning alternatives ia valori mai mari.

Figura 8.3-2. Parametrii operatorului Decision Tree: setări implicite (stânga) vs model (dreapta)

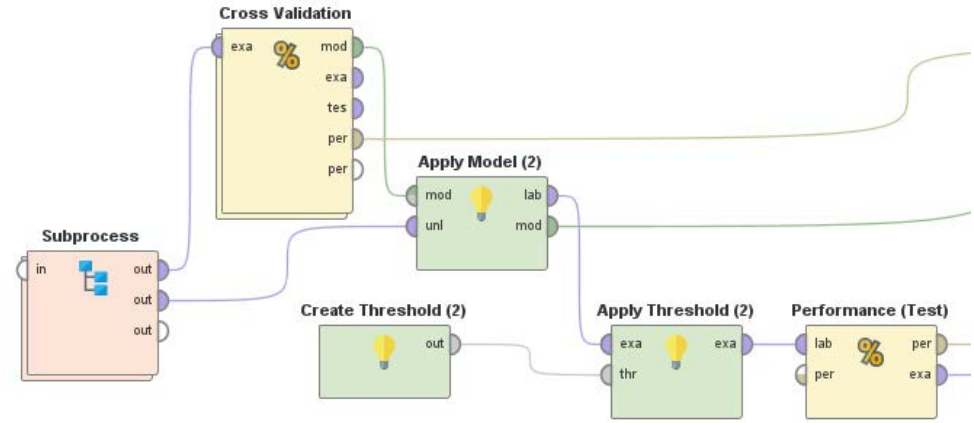


Pentru acest exemplu vom folosi majoritatea atributelor din setul de date „employee_attrition”. Anterior analizei, fiecare atribut numeric a fost recodificat într-un atribut cu trei categorii (Discretize by Binning). Ne propunem să realizăm un

model de clasificare bazat pe arborele decizional cu scopul de a prezice situațiile de părăsire a companiei. Pașii realizării acestui model în RapidMiner, rezultatele obținute și interpretarea lor apar în Figura 8.3-3.

Figura 8.3-3. Arbore decizional: un model relativ mai complex în RapidMiner

Pasul 1:
Realizăm procesul din imaginea de mai jos. În folderul asociat volumului am inclus două versiuni ale acestui proces: o versiune mai simplă, fără setarea pragului de probabilitate (acesta este stabilit implicit la 0.5) și o versiune care oferă posibilitatea setării pragului de probabilitate. Toate datele necesare sunt pregătite anterior în interiorul operatorului Subprocess. Setul de date utilizat pentru instruirea modelului este „employee_attrition_training_validation". Setul utilizat pentru testarea modelului este „employee_attrition_testing". Includem operatorul Decision Tree cu setările originale parțial modificate (am crescut numărul minim de cazuri necesar pentru divizarea unui nod, respectiv asociat unui nod terminal; am crescut numărul de alternative la pre-pruning). În cazul modelului 2 am inclus operatorii care ne ajută să setăm pragul de probabilitate dorit (Create Threshold și Apply Threshold). Includem operatorul necesar pentru aplicarea modelului pe setul de date de testare, respectiv operatorul care calculează măsurile de performanță. Realizăm conexiunile și rulăm procesul. La modelul 1, pragul de probabilitate este setat implicit la 0.5 (prima clasă este No, a doua Yes). Prin urmare, modelul va prezice că angajații în cazul cărora valoarea atributului „confidence(Yes)" va fi mai mare de 0.5 vor pleca, iar restul vor rămâne. La modelul 2, pragul este stabilit manual la 0.3.

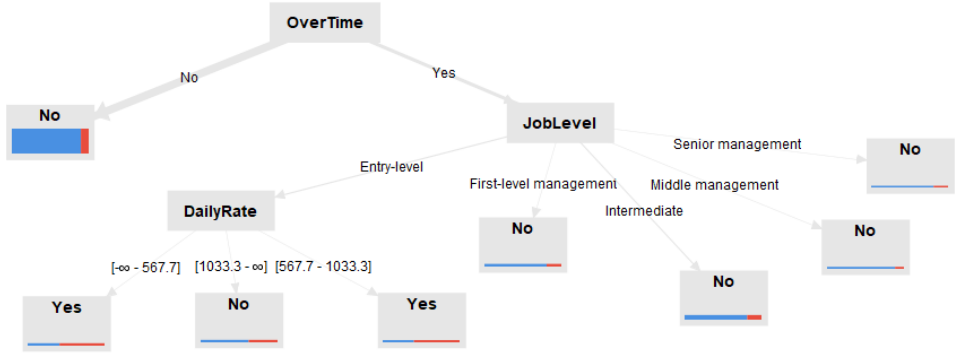


Rezultate (setul de testare):
Relativ la setul de testare, modelul produce predicțiile din imagine (am inclus primele 5 cazuri și alte trei de interes în contextul acestui exemplu). Coloana Attrition prezintă situația reală, iar coloana „prediction(Attrition)" predicția realizată de model. Observăm că unele predicții sunt greșite (în imagine, cele cu Id 1, 21 și 65). Probabilitățile asociate predicțiilor apar pe coloanele „confidence". Pentru fiecare caz este selectată ca predicție clasa care are cea mai mare probabilitate (confidence).

Row No.	Id	Attrition	prediction(Attrition)	confidence(No)	confidence(Yes)
1	1	Yes	No	0.815	0.185
2	2	No	No	0.898	0.102
3	8	No	No	0.898	0.102
4	11	No	No	0.898	0.102
5	12	No	No	0.898	0.102
...					
8	21	No	Yes	0.415	0.585
...					
17	65	Yes	No	0.625	0.375
...					
33	137	Yes	Yes	0.406	0.594

...

Modelul de predicție estimat pe setul de date de instruire este unul relativ simplu, compus din doar trei atribute: OverTime, JobLevel și DailyRate. Concluzia acestui model este clară: angajații pe poziții de intrare, plătiți relativ mai prost și nevoiți să muncească peste program tindă să părăsească firma într-o măsură mult mai mare. Desigur, acesta nu este singurul model de predicție posibil. Dacă schimbăm valorile parametrilor sau atributele incluse în setul de date, sunt șanse să obținem un model parțial diferit de acesta.



Secțiunea PerformanceVector conține matricea de confuzie și diferite măsuri ale performanței clasificării (cele selectate), toate calculate relativ la ambele seturi de date (instruire și testare). Observăm că acuratețea predicției este de fiecare dată peste 80%, ceea ce indică șanse mari de generalizare a modelului de predicție. Chiar dacă de obicei calitatea clasificării este mai bună în cazul setului de instruire comparativ cu cel de test, de această dată este invers. Comparativ cu modelul care a inclus doar doi predictori, modelul cu toți predictorii are o acuratețe a clasificării similară (e doar puțin mai mare), respectiv reușește că clasifice corect o pondere puțin mai mare a cazurilor de angajați care pleacă (ceea ce e foarte util din punct de vedere practic). În continuare, modelul prezice mai bine situația în care angajații nu pleacă (precizia = 86/88%, reamintirea = 97/95%) comparativ cu situația în care angajații pleacă (precizia = 54/56%, reamintirea = 20/34%).

accuracy: 82.61% +/- 2.97% (micro average: 82.60%)

	true No	true Yes	class precision
pred. No	814	130	86.23%
pred. Yes	49	36	42.35%
class recall	94.32%	21.69%	

instruire, prag 50%

accuracy: 81.15% +/- 5.77% (micro average: 81.15%)

	true No	true Yes	class precision
pred. No	774	105	88.05%
pred. Yes	89	61	40.67%
class recall	89.69%	36.75%	

instruire, prag 30%

accuracy: 84.35%

	true No	true Yes	class precision
pred. No	358	57	86.27%
pred. Yes	12	14	53.85%
class recall	96.76%	19.72%	

testare, prag 50%

accuracy: 85.03%

	true No	true Yes	class precision
pred. No	351	47	88.19%
pred. Yes	19	24	55.81%
class recall	94.86%	33.80%	

testare, prag 30%

9. Rețea neuronală (Neural Net)

Nume operator	Rețea neuronală (Neural Net)
Categoria majoră	învățare asistată (supervised learning)
Tip model	clasificare (classification)[1]
Grup	rețele neuronale
Atribut dependent	categorial[2]
Atribute independente	numerice, categoriale[3]
Gestionează valorile lipsă	nu
Distorsiune (bias)	mică/mare
Varianță (variance)	mare/mică[4]

Bibliografie utilizată și recomandată:

- Data mining: a tutorial-based primer (Roiger, 2017, Chapters 8, 10)
- Data Science: Concepts and Practice (Kotu & Deshpande, 2019, Chapter 4.5)
- Data Analytics for the Social Sciences: Applications in R (Garson, 2021, Chapter 7)
- Machine Learning for Business Analytics: Concepts, Techniques and Applications in RapidMiner (Shmueli et al., 2023, Chapter 11)

[1] Poate prezice și variabile dependente de tip metric.

[2] Poate fi și numeric.

[3] Anterior rulării modelului, atributele categoriale trebuie convertite în atribute numerice de tip 0/1, unde 0 indică absența însușirii, iar 1 prezența însușirii. De exemplu, atributul sex cu două categorii, bărbat și femeie, poate fi convertit în atributul bărbat cu valorile 0/1 (sau femeie, tot cu valorile 0/1).

[4] Atunci când numărul cazurilor este suficient de mare, rețelele neuronale (cele complexe, de tip deep learning) pot avea simultan bias mic și varianță mică. Fenomenul este cunoscut sub numele de „double descent" (Belkin et al., 2019; Neal et al., 2019). Situația de „double descent" nu apare în cazul tuturor rețelelor instruite pe un set de date cu un număr suficient de mare de cazuri. Apariția acestui fenomen depinde în esență de complexitatea rețelei, mai exact de numărul de atribute de input (numărul predictorilor), numărul de clase de output, adâncimea rețelei (numărul de straturi), deci de numărul total al coeficienților care trebuie estimați (Bubeck & Sellke, 2021).

În cazul unora dintre algoritmii de clasificare, regresia logistică de exemplu, forma generală a relației dintre atribute este specificată direct în structura algoritmului, prin urmare datele sunt citite și interpretate prin intermediul acestei lentile. În realitate, relațiile dintre atribute nu apar doar sub astfel de forme relativ simple, ci și sub forme relativ mai complexe. Mai mult, cel mai adesea nu știm care e forma pe care o iau relațiile analizate. Prin urmare, pentru a modela relații complexe, adesea necunoscute, e nevoie de algoritmi mai flexibili, de algoritmi care să aibă capacitatea de a descoperi relativ automat formele cele mai probabile ale relațiilor. Rețelele neuronale (neural networks) intră tocmai în această categorie. Ideea pe care se bazează acestea este relativ simplă: combinarea într-o manieră foarte flexibilă a informațiilor (atributelor) disponibile cu scopul de a modela relațiile dintre ele și mai ales dintre ele și atributul de interes. Similar capitolelor anterioare, și acesta este structurat în trei secțiuni principale: prezentarea logicii și a pașilor algoritmului, prezentarea și discutarea unui exemplu didactic, respectiv prezentarea a două exemple de analiză în RapidMiner Studio. Pentru cei relativ mai interesați de acest algoritm, am adăugat la final un exemplu de modelare în care am folosit un algoritm mai complex, învățarea profundă (Deep Learning).

9.1. Logica și pașii algoritmului

Câteva prezentări video la nivel introductiv despre rețelele neuronale sunt următoarele: Neural Nets intro, Neural Networks Pt. 1: Inside the Black Box, Neural Networks Pt. 2: Backpropagation Main Ideas, Neural Network Full Course | Neural Network Tutorial For Beginners. În contextul unui manual, probabil cea mai intuitivă formă de introducere a conceptului de rețea neuronală este prin intermediul unor reprezentări grafice. Câteva exemple simple de rețele neuronale apar în Figura 9.1-1. Prima rețea (stânga sus) e formată din două straturi (layers), unul de input și altul de output (deci nu avem un strat invizibil / ascuns); stratul de input conține trei neuroni / noduri (atribute), iar cel de output doi neuroni / noduri corespunzătoare celor două clase ale atributului de interes (este binominal);[5] săgețile reprezintă coeficienții de ponderare / ponderările (weights). Această rețea este echivalentă cu un model de regresie logistică, ambele având aceeași expresie matematică (x_1-x_3

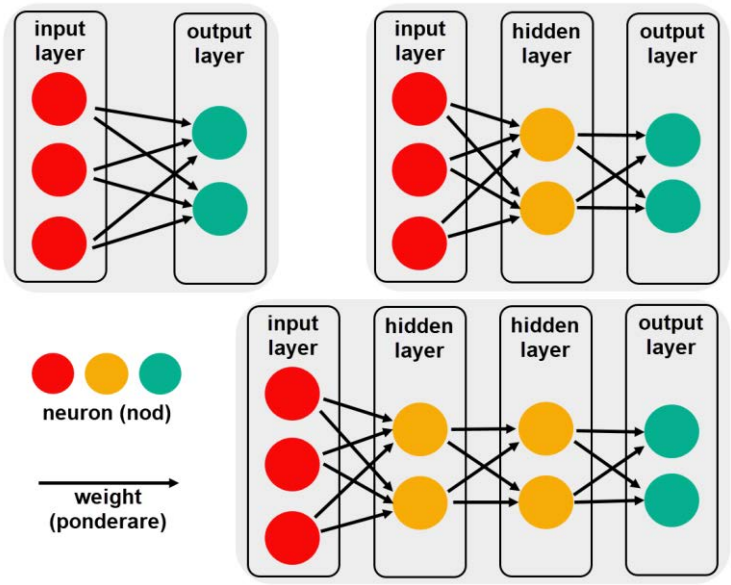
[5] În cazul atributelor de interes categoriale, în general, stratul de output include un număr de neuroni egal cu numărul de clase (unele softuri includ un număr de neuroni egal cu numărul de clase minus unu). Dacă atributul de interes ar fi fost metric, stratul de output ar fi inclus un singur neuron.

reprezintă atributele de input adică variabilele independente, w_1-w_3 sunt ponderările, respectiv coeficienții de regresie, iar bias este constanta):

$$valoare\ nod_{clasa\ i} = x_1 * w_{1i} + x_2 * w_{2i} + x_3 * w_{3i} + bias_{clasa\ i}$$

Rețeaua neuronală din dreapta sus include suplimentar un strat ascuns / invizibil format din doi neuroni, toate conexiunile dintre straturile de input și output trecând prin aceștia. Rețeaua neuronală din partea de jos are două straturi ascunse, fiecare format din câte doi neuroni. Pe lângă ponderări, fiecare neuron care face parte dintr-un nod ascuns sau de output include și o valoare constantă specifică fiecărui neuron (nu este vizibilă în imagine) numită distorsiune (bias) sau prag (threshold).

Figura 9.1-1. Câteva exemple simple de rețele neuronale (dependentă binominală)



Toate aceste exemple au în comun faptul că datele (informația) urmează un drum strict definit. Astfel, informația circulă de la stânga la dreapta, adică de la stratul de input la cel de output, trecând prin stratul(rile) ascunse (dacă e cazul). Pe parcursul acestei faze nu există feedback (loops). De aici și denumirea de „feed-forward neural network" sau „forward propagation". În faza următoare, rețeaua compară predicțiile obținute cu situația reală (valorile corecte) și calculează eroarea de predicție (folosind o așa numită funcție cost). Luând în calcul eroarea obținută și setările definite de utilizator, algoritmul modifică (ajustează) valorile ponderărilor (weights), reface predicțiile și recalculează eroarea. Acest proces este reluat de mai multe ori (de obicei de sute sau mii de ori) până sunt îndeplinite condițiile specificate

de utilizator, iar funcția țintă este aproximată cât mai corect, adică este „învățată”. Această fază poartă numele de propagare inversă (back propagation) și este compusă din două etape: calcularea erorii de clasificare și ajustarea coeficienților (ponderările și constantele). În cadrul acestui capitol ne interesează în principal tipul de rețea neuronală descrisă sumar anterior. Desigur, există și alte tipuri de rețele neuronale.

Construirea unei rețele neuronale

La modul general, similar cu procesul de realizare a unui model de clasificare, procesul de construire a unei rețele neuronale poate fi divizat în următoarea serie de pași:

1. **Identificarea atributelor** de input (variabilele independente sau obișnuite), respectiv a atributului de output (variabila dependenta sau label sau de interes).
2. **Rescalarea atributelor** de input și output pe intervalul de variație [-1;1] sau [0;1].
3. **Definirea rețelei neuronale**, a topologiei acesteia, mai exact includerea sau nu a unui strat (unor straturi) ascuns(e). Cel mai adesea o rețea neuronală conține cel puțin un strat. În continuare, pentru fiecare strat ascuns stabilim un număr de neuroni. La final stabilim valorile parametrilor relevanți (rata de învățare, momentum, numărul ciclurilor de instruire, eroarea minimă etc.).
4. **Instruirea rețelei neuronale** pe un set de date reprezentativ. Simultan cu acest pas sau imediat după folosim setul de date de validare pentru a alege valorile coeficienților (ponderări și constante).
5. **Testarea rețelei neuronale** pe setul de date de test.
6. **Realizarea predicțiilor** pe un set nou de date (relativ la acesta nu știm valorile atributului de interes).

Realizarea efectivă a unei rețele neuronale are loc la pasul major trei. Pașii anteriori sunt și ei importanți, dar sunt oarecum impliciți, comuni tuturor modelelor de clasificare. Prin urmare, o dată ce am ales atributele de input și atributul de output, definirea unei rețele neuronale se reduce, în esență, la două lucruri: (1) stabilirea configurației „cutiei negre”, adică a configurației straturilor ascunse și (2) setarea valorilor parametrilor asociați rețelei neuronale. În acest context e necesar să răspundem la următoarele întrebări:

Care va fi numărul de straturi ascunse (hidden layers)? Răspunsul depinde în esență de setul de date (numărul de cazuri și de atribute, specificul acestora) și de tipurile de relații (patterns). Cel mai adesea definim un singur strat ascuns, dar, dacă relațiile dintre atribute sunt complexe, e foarte posibil să fie nevoie să definim mai multe straturi.

Care va fi numărul de neuroni din fiecare strat ascuns? Răspunsul inițial este orientativ și îl putem stabili în funcție de criterii precum: (1) valoarea din mijlocul seriei definite de numărul de noduri de output și numărul de noduri de input; de exemplu, dacă atributul de output are două clase, iar numărul de atribute incluse în model este 10, vom testa inițial un strat ascuns cu 5/6 neuroni; (2) mai puțin decât dublul numărului de atribute; (3) două treimi din numărul de noduri de input la care adăugăm numărul de noduri de output. Numărul de neuroni din stratul ascuns influențează foarte mult capacitatea rețelei de a modela relații complexe. Dacă observăm că eroarea finală este prea mare (underfitting) vom crește numărul de neuroni, iar dacă rețeaua „memorează" datele (overfitting), îl reducem.

Ce funcție de transformare folosim? Cel mai adesea valoarea rezultată din însumarea termenilor ce intră într-un nod (produsele dintre ponderări și valorile de input plus constanta aferentă) nu aparține de intervalul [-1;1], respectiv [0;1], prin urmare trebuie transformată. Deoarece relațiile dintre atribute (input și output) sunt adesea complexe și non-liniare, transformarea se realizează cu ajutorul unei funcții non-liniare. Funcțiile utilizate cel mai adesea sunt sigmoid (logistică), ReLU (Rectified Linear Unit), tanh și softmax.[6]

Care va fi rata de învățare? Rata de învățare reprezintă valoarea maximă cu care se pot modifica, de la un ciclu la următorul, valorile de ponderare (weights) și constantele (biases). Prin urmare, acest parametru indică rapiditatea cu care este permisă schimbarea valorilor coeficienților. Rata de învățare poate lua valori în intervalul (0;1]. O rețea cu valori mari ale ratei de învățare va permite coeficienților să se schimbe foarte mult de la un ciclu la următorul. Una cu o rată mică va permite schimbări relativ mai mici. În general este de preferat să începem cu o rată mare de învățare și să-i permitem să scadă pe măsură ce crește numărul ciclurilor. O rată inițială mare sporește șansele ca rețeaua să identifice zona cu valorile cele mai

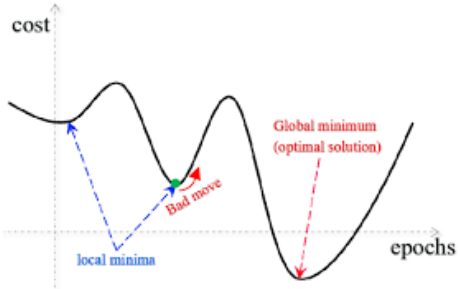
[6] Transformarea valorilor este obligatorie doar în cazul în care sarcina este una de clasificare (precum în cazul acestui capitol, respectiv volum). Dacă sarcina este una de regresie, nu e necesar să transformăm valorile de output. Pe lângă alegerea funcție de transformare, unele softuri permit și alegerea unui algoritm de optimizare (de exemplu Gradient Descent, Adam; Neural Net folosește implicit Gradient Descent), respectiv a unei tehnici de regularizare (L1 sau L2). Spre deosebire de operatorul Neural Net care nu permite astfel de alegeri, operatorul Deep Learning din RapidMiner și extensia Keras permit mult mai multe alegeri.

potrivite ale coeficienților și să nu rămână blocată într-o zonă cu valori relativ ok, dar care nu sunt printre cele mai bune posibil. Pe măsură ce rețeaua ajunge tot mai aproape de soluția optimă, e de preferat ca rata de învățare să scadă tot mai mult astfel încât să nu rateze valorile „adevărate" ale coeficienților.[7] În RapidMiner, parametrul care permite acest lucru este Decay.

Care va fi valoarea parametrului momentum? Momentum ne arată măsura în care valorile actuale ale coeficienților (weights și biases) sunt influențate de valorile anterioare ale acelorași coeficienți. Simplu spus, momentum se referă la ușurința cu care permitem coeficienților să-și schimba direcția în care se îndreaptă. Oarecum coeficienții țin minte care era direcția anterioară (creștere sau scădere) iar momentum încearcă să le mențină aceeași direcție. De exemplu, dacă un coeficient avea anterior tendința de crește, cât de ușor îi va fi să inverseze această tendință ca urmare a valorilor de input asociate noului ciclu. Schimbarea de la un ciclu la următorul e egală cu: rata de învățare * eroarea * valoarea neuronului + momentum * rata de schimbare anterioară. Parametrul momentum poate lua valori în intervalul [0;1]. O rețea cu valori mari ale momentum va tinde să păstreze direcția anterioară de variație a coeficienților. Una cu momentum mic va permite schimbări frecvente ale direcției.

Câte iterații (cicluri / epoci) sunt necesare, respectiv care e eroarea minimă acceptată? Numărul de iterații se referă la numărul maxim al ciclurilor pe care le poate parcurge rețeaua atunci când aproximează coeficienții. Funcție de complexitatea relațiilor și mărimea setului de date (cazuri și atribute) pot fi suficiente câteva zeci, sute sau chiar mii de cicluri. Evident, eroarea minimă

[7] Câteva exemple de probleme care pot să apară atunci când rata de învățare nu este cea adecvată pot fi vizualizate la linkurile următoare: Learning Rate Local Minimum, Learning Rate Too Big. În esență e vorba despre faptul că rețeaua neuronală consideră greșit că valoarea minimă a erorii de clasificare identificată la un anumit moment al procesului de instruire (ciclu / epocă) este și valoarea minimă globală când, de fapt, valoarea respectivă este doar o valoare minimă locală. Prin urmare, dacă procesul de instruire ar fi continuat, s-ar fi obținut o eroare și mai mică, cea mai mică posibilă (valoarea minimă globală), deci rețeaua neuronală ar fi fost cea mai performantă posibil. Imaginea următoare ilustrează simplu diferența dintre cele două vezi:



acceptată se referă la valoarea la care trebuie să ajungă eroarea în urma instruirii. Ambele valori trebuie definite de analist. În general, procesul de instruire încetează atunci când este îndeplinită una dintre următoarele două condiții:

- Se atinge numărul maxim de cicluri de instruire (epoci) specificat de analist.
- Eroarea de clasificare coboară sub valoarea minimă specificată de analist.

În cazul unor programe e posibil să definim și alte două condiții:

- Eroarea de clasificare nu scade după un anumit număr de cicluri de instruire.
- Valorile actuale ale coeficienților diferă foarte puțin de valorile din ciclul anterior.

Pentru a crește șansele de succes (obținerea unui model cu o acuratețe mare a predicției), e necesar să avem în vedere și următoarele două categorii de probleme, prima categorie vizând atributele din setul de date, cealaltă numărul de cazuri.

Distribuția valorilor fiecărui atribut din setul de date ar trebui să fie similară cu distribuția observată la nivel de populație, respectiv cu distribuția teoretică, incluzând aici și intervalul de variație. Dacă distribuția valorilor unui atribut metric este asimetrică, adică are o „coadă lungă" (skewed distribution), putem transforma valorile prin logaritmare, discretizare sau standardizare. Atributele categoriale trebuie transformate în unul sau mai multe atribute care iau valoarea 0 sau 1 funcție de prezența sau absența însușirii respective (transformare dummy). Dacă un atribut are foarte multe variante de răspuns, de exemplu codul companiei sau al localității, putem construi unul sau mai multe atribute care să prezinte diferite valori metrice relevante. De exemplu, în situația în care știm organizația din care a făcut parte un anumit angajat, putem construi diferite variabile noi: una care să indice pentru fiecare angajat mărimea medie a organizației din care a făcut parte (ca cifră de afaceri sau ca număr de angajați), alta care să indice rata atriției din anul anterior etc.

E de preferat ca **numărul de cazuri din setul de instruire** trebuie să fie cât mai mare. Putem aproxima numărul minim necesar de cazuri în funcție de numărul coeficienților (weights și biases) ce compun rețeaua. Să presupunem că avem o rețea cu un număr s de neuroni de input, h neuroni ascunși și un neuron de output. Numărul coeficienților unei rețele poate fi calculat folosind formula: h x (s + 1) + h + 1. De exemplu, dacă avem 5 atribute de input și 3 neuroni ascunși, rețeaua va trebui să estimeze 22 coeficienți (3x6+3+1); pentru 10 atribute de input și 5 neuroni ascunși ajungem la 61 coeficienți. Recomandarea este ca setul de date să conțină minimum 30 de cazuri pentru fiecare coeficient (de preferat 100), deci acesta va trebui să conțină minimum 660 cazuri, respectiv minimum 1830.

Instruirea unei rețele neuronale

Procesul de instruire a unei rețele neuronale începe cu alegerea unor valori aleatoare ale coeficienților (fiecare weight și bias). Apoi, poate începe primul ciclu de instruire / învățare. Acesta poate fi divizat în patru pași și anume:

1. Propagarea informației dinspre primul strat spre ultimul (forward propagation).
2. Calcularea erorii de clasificare.
3. Propagarea informației ultimul strat înspre primul (backward propagation).
4. Ajustarea coeficienților (weights și bias).

În continuare se trece la ciclul doi de învățare și se reiau pașii descriși mai sus. Procesul continuă până se obține o soluție care îndeplinește cel puțin unul dintre criteriile specificate în faza de definire a rețelei neuronale. În Figura 9.1-2 am prezentat o ilustrare grafică simplă și sugestivă a procesului de învățare (chiar dacă stratul de output conține un singur neuron, logica nu se schimbă).[8] În imaginea din stânga (feed forward) observăm cum procesul de învățare începe de la (1) valorile asociate neuronilor din stratul de input (valorile luate de atributele din setul de date), apoi continuă cu (2) însumarea acestor valori ponderate cu valorile selectate aleator (inițial, doar pentru primul ciclu) ale ponderărilor la care se adaugă o constantă, rezultă astfel (3) o valoare pentru fiecare neuron din stratul ascuns; la rândul lor, (4) aceste ultime valori sunt combinate într-o manieră similară rezultând (5) valoarea asociată neuronului de output.

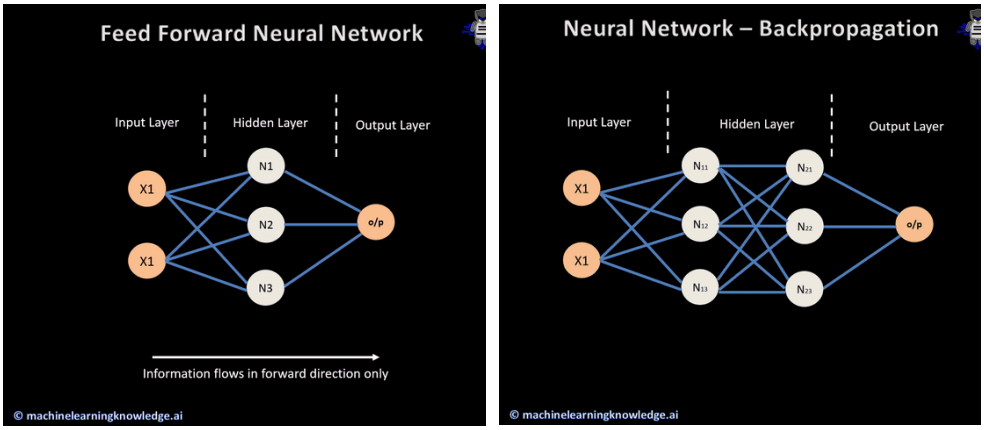
În imaginea din dreapta are loc procesul de propagare inversă: (1) predicția obținută pentru fiecare caz este comparată cu situația reală și se calculează diferența dintre acestea, (2) diferențele sunt însumate obținându-se o măsură a erorii de clasificare, (3) valoarea erorii este trimisă spre fiecare neuron din stratul ascuns, (4) coeficienții (weight și bias) asociați neuronilor ascunși sunt ajustați, (5) iar apoi eroarea este propagată mai departe spre coeficienții neuronilor din stratul de input care sunt la rândul lor ajustați. Astfel se termină primul ciclu de învățare. Procesul descris anterior este apoi reluat din nou și din nou până în momentul în care unul dintre criteriile specificate de analist este îndeplinit.

Diferența dintre predicțiile realizate de rețeaua neuronală și valorile reale, eroarea, este calculată folosind o așa numită funcție pierdere (loss funcțion). În cazul modelelor

[8] Pentru un exemplu simplu de calcul în faza de propagare inversă (backpropagation) se poate consulta linkul Backpropagation Example, respectiv prezentarea video Backpropagation Details Pt. 1: Optimizing 3 parameters simultaneously.

de clasificare (urmărim să estimăm probabilitatea), funcția pierdere folosită cel mai adesea este cross-entropy loss. Scopul instruirii este de a minimiza tocmai această eroare / pierdere (valoarea funcției pierdere), deci de a crește calitatea predicțiilor, procesul numindu-se optimizarea pierderii (loss optimization). Minimizarea pierderii se realizează prin ajustarea coeficienților rețelei. Pentru a afla minimul funcției pot fi folosiți diferiți algoritmi de optimizare, cel mai utilizat fiind „gradient descent" (ajustarea coeficienților se face în direcția negativului gradientului).

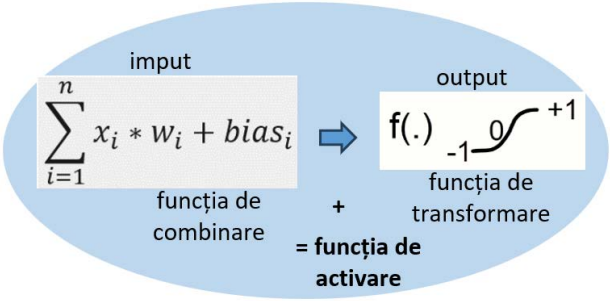
Figura 9.1-2. Procesul de învățare în cadrul unei rețele neuronale: propagare înainte (feed forward) și propagare inversă (backpropagation)



Sursa: https://machinelearningknowledge.ai/

În faza de propagare înainte, la nivelul fiecărui neuron ascuns sau de output, sunt realizate două operații majore (Figura 9.1-3). Prima dată este calculată suma dintre produsul fiecărei valori de input și coeficientul de ponderare (weight) la care se adaugă constanta (bias) (vezi partea din stânga a figurii). Aceasta este pasul de combinare a valorilor. Valoarea nou obținută este apoi transformată pe intervalul [-1;1] folosind o funcție de transformare (logistică în imagine, dar poate fi și de alt tip). Noua valoare rezultată este folosită ca input pentru neuronii din stratul următor. Cele două funcții combinate formează așa numita funcție de activare.

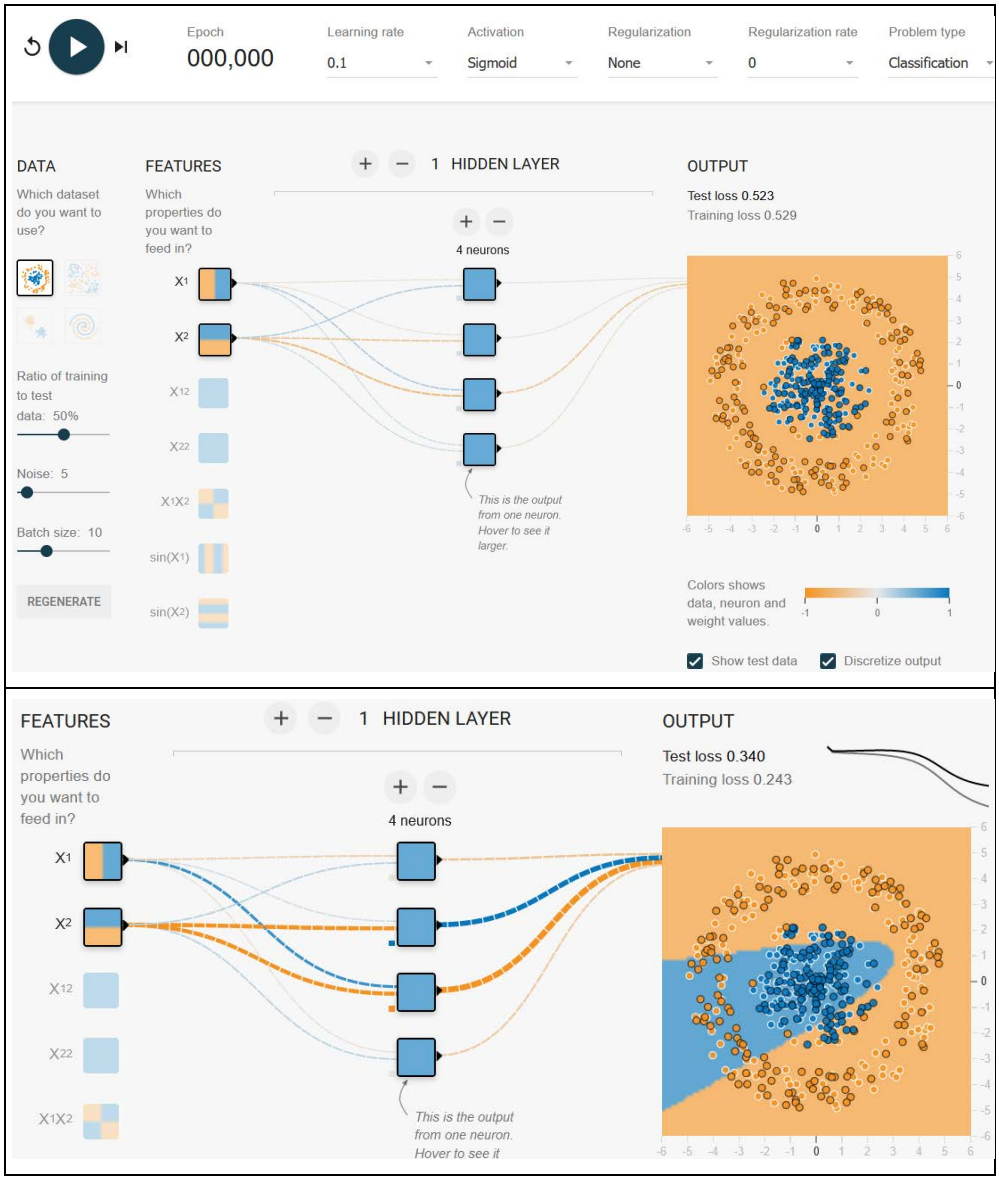
Figura 9.1-3. O prezentare schematică a calculelor realizate în interiorul unui neuron în faza de propagare înainte

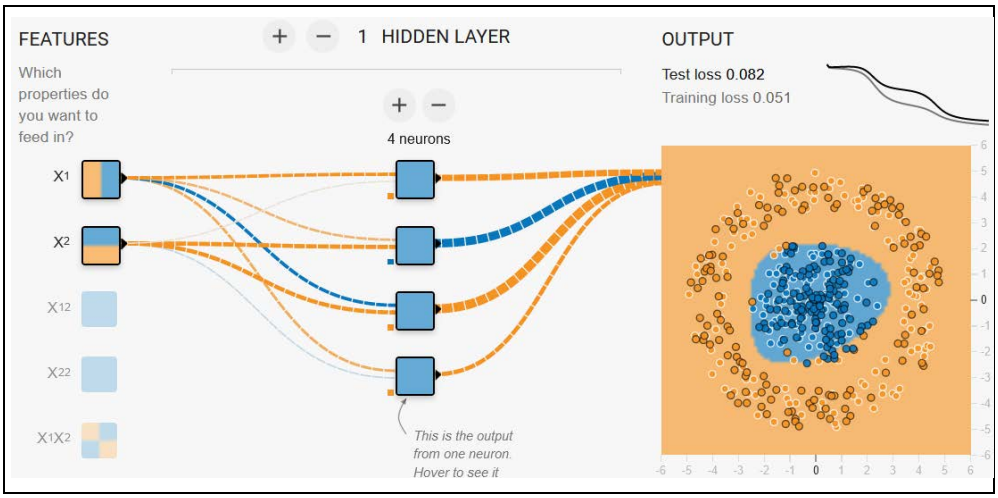


Pentru a avea o imagine și mai clară cu privire la procesul de instruire se poate consulta Figura 9.1-4. Aceasta conține trei imagini: prima descrie punctul de start al unei rețele neuronale simple (două atribute de input, un strat ascuns cu patru neuroni și un atribut de output cu două clase), respectiv două faze intermediare (starea rețelei după 34, respectiv 79 cicluri de instruire). Pentru o ilustrare cât mai veridică, am inclus pe lângă cazurile din setul de date de instruire și cazurile din setul de date de test. Inițial, eroarea de clasificare este maximă și similară relativ la ambele seturi de date (0.53/0.52), iar valorile coeficienților sunt similare. Valorile ponderărilor (weights) sunt ilustrate prin liniile conectoare dintre neuroni. O grosime mai mare indică o valoare absolută mai mare. Culoarea albastră indică o valoare pozitivă, iar culoarea portocalie o valoare negativă. Observăm că la start valorile absolute ale ponderărilor sunt similare (liniile au aproximativ aceeași grosime). Valoarea constantei (bias) este reprezentată prin pătratul mic poziționat în partea stângă-jos a fiecărui neuron din stratul ascuns. Și în cazul acesta albastru indică o valoare pozitivă, iar portocaliu una negativă; cu cât culoarea este mai intensă, cu atât valoarea constantei este mai mare. La start, valorile constantelor sunt relativ similare. La ciclul 34, ca urmare a învățării (ajustării coeficienților), eroarea scade la aproximativ jumătate (0.24/0.34). Observăm că valorile coeficienților diferă (liniile conectoare – weights – au grosimi și culori diferite comparativ cu liniile de la punctul de start; culorile și intensitatea acestora la nivelul pătratelor mici – constantele – diferă și ele). Observăm două lucruri importante: (1) funcția estimată de rețeaua neuronală este doar o aproximare grosieră a funcției reale și (2) eroarea de clasificare este semnificativ mai mare în cazul setului de date de test, deci rețeaua neuronală a „memorat" setul de date de instruire (overfitting). La ciclul 79 eroarea scade și mai mult (0.05/0.08), iar valorile coeficienților se modifică și mai mult, accentuând direcția de schimbare semnalată la ciclul 34. Observăm că: (1) funcția estimată de rețeaua neuronală se suprapune foarte mult cu funcția reală și (2) eroarea de clasificare asociată setului de test este în continuare

mai mare comparativ cu eroarea din testul de instruire. Desigur, procesul de instruire poate continua, erorile scăzând și mai mult. Cei care doresc, pot folosi această aplicație online pentru a seta diferite configurații, relativ la diferite seturi de date, și a observa cum are loc învățarea pas cu pas.

Figura 9.1-4. O rețea neuronală la momentul de start și după 34 / 79 cicluri (epoci) de învățare



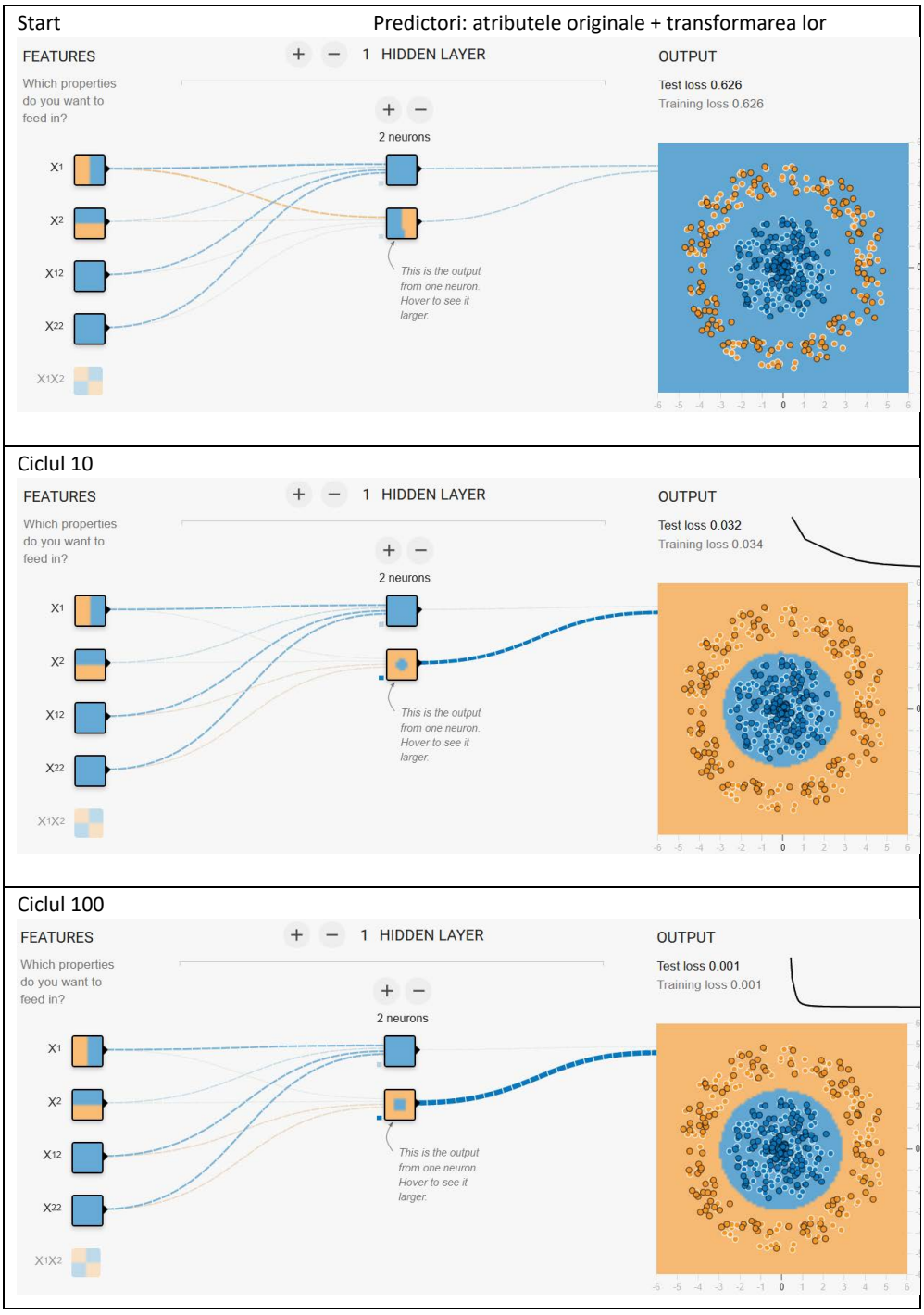


Sursa: https://playground.tensorflow.org/

Învățarea unor relații nonliniare

În exemplul anterior am văzut că rețeaua neuronală a reușit să învețe relații de tip nonliniar. Acest lucru a fost posibil ca urmare a faptului că numărul de neuroni ascunși a fost mai mare decât numărul neuronilor (atributelor) de input. O soluție alternativă constă în creșterea numărului atributelor de input: putem include atribute noi sau transformări ale atributelor inițiale. În Figura 9.1-5 am prezentat comparativ procesul de învățare în două situații care diferă doar prin faptul că într-un caz am folosit ca predictori și transformări ale atributelor originale (pătratul acestora). În cazul în care predictorii sunt doar atributele originale, rețeaua neuronală nu reușește să învețe relațiile din setul de date: după 10 cicluri sunt clasificate corect 51% din cazuri, iar după 100 cicluri tot rămân clasificate corect 73% din cazuri. În cazul în care alături de predictorii originali includem și transformări ale acestora, rețeaua neuronală reușește să învețe relațiile din setul de date: după 10 cicluri sunt clasificate corect doar 97% din cazuri, iar după 100 cicluri 99.9%.

Figura 9.1-5. Învățarea unor relații nonliniare: atribute originale vs. transformate



Sursa: https://playground.tensorflow.org/

Avantaje și dezavantaje

Rețele neuronale prezintă o serie de avantaje și dezavantaje. Printre avantaje se numără cel puțin următoarele:

- Pot modela automat relații nonliniare complicate, precum și efecte de interacțiune.
- Gestionează relativ bine date problematice, afectate de outlieri și erori aleatoare.
- Pot fi folosite pentru a rezolva ambele tipuri de probleme (clasificare și regresie).
- Pot lucra cu atribute de tip metric și categorial. Atributul dependent (label) poate fi metric sau categorial. La fel și atributele independente.
- Performează bine cu date mari (multe cazuri și atribute) din aproape orice domeniu.
- Predicțiile pot fi bune și în cazul subiecților cu valori extreme.

Desigur, toate aceste avantaje sunt însoțite de anumite costuri. Printre dezavantaje se numără și următoarele:

- Procesul de stabilire a topologiei finale a rețelei neuronale este unul dificil, fiind adesea necesară testarea unor configurații multiple. Procesul de optimizare al parametrilor (numărul acestora este foarte mare) este unul îndelungat și necesită o serie de informații precum și experiență.
- Drumul parcurs până la rezultatul obținut (legătura dintre input și output) este mai puțin vizibil, intuitiv. Este un algoritm de tip „black box” (cutie neagră), deci avem nevoie de analize ulterioare pentru a înțelege cum a ajuns algoritmul la acel rezultat.
- Uneori algoritmul nu converge spre soluția optimă, produce o soluție care este bună, dar nu este neapărat cea mai bună.
- Uneori performanța pe setul de testare este mult mai slabă comparativ cu performanța pe setul de instruire (overfitting).
- Funcție de complexitatea modelului și numărul de cazuri, timpul necesar pentru rularea modelului poate fi relativ mai mare.
- Dacă atributele sunt de tip categorial, transformarea lor în atribute numerice va produce un set de date cu mult mai multe atribute, deci complexitatea modelului (numărul atributelor de input, numărul straturilor, numărul nodurilor) va crește exponențial. Prin urmare e nevoie ca setul de date de instruire să includă un număr foarte mare de cazuri.

- Atunci când setul de date are puține cazuri (< 1000, funcție și de numărul de atribute), e posibil ca alți clasificatori să performeze relativ mai bine.

9.2. Un exemplu didactic

Toate exemplele prezentate aici în acest sub-capitol, respectiv în următorul sunt realizate în RapidMiner folosind operatorul Neural Net. Înainte de a trece efectiv la prezentare acestor exemple este necesar să descriem pe scurt parametrii asociați acestui operator chiar dacă o parte a informațiilor prezentate aici se regăsesc sub o formă apropiată și în sub-capitolul anterior.

Parametrii operatorului Neural Net

Parametrul **hidden layers** permite setarea numărului de straturi intermediare, respectiv a numărului de neuroni (noduri) asociat fiecărui strat. Cu cât includem mai multe straturi și mai mulți neuroni, cu atât complexitatea modelului crește, deci, teoretic, rețeaua va putea modela cu succes relații mai complicate. Dezavantajul este că modelul obținut va avea șanse de generalizare mai reduse. Dacă nu definim niciun strat ascuns, modelul va include automat un singur strat cu un număr de noduri egal cu (# atribute + # clase) / 2 + 1. Dacă setăm numărul de noduri la -1, numărul de noduri va fi stabilit folosind aceeași formulă. Întotdeauna numărul de noduri final va fi mai mare cu unu față de numărul de noduri cerut de utilizator. Nodul suplimentar, denumit constantă sau distorsiunea (bias), poate fi recunoscut prin faptul că nu va fi conectat la nodul anterior, respectiv are o culoare diferită.

Parametrul **training cycles** se referă la numărul de cicluri de instruire. După fiecare ciclu, rețeaua neuronală calculează eroarea și, în funcție de mărimea și sensul acesteia, ajustează valorile coeficienților (ponderările și constantele) cu scopul de a reduce eroarea de predicție. Acest proces este repetat de un număr de ori egal cu valoarea setată pentru acest parametru.

Parametrul **learning rate** sau rata de învățare specifică cât de mult se schimbă coeficienții (weights și biases) la fiecare ciclu. Firesc, valoarea parametrului nu poate fi zero. Adesea, o rată mai mică a schimbării crește șansele de a găsi „valorile corecte” (desigur, dacă avem un număr suficient de mare de cicluri). Pentru siguranță e de preferat să folosim inițial o rată relativ mai mare pe care să o reducem pe măsură ce crește numărul de cicluri.

Parametrul **momentum** adaugă o parte din valoarea cu care a fost modificată ponderarea anterioară la ponderarea actuală. Scopul este de reduce șansele ca modelul să considere că valoarea minimă a erorii obținute la un moment dat este și valoarea minimă maximă, respectiv pentru a nu permite schimbări bruște de direcție a sensului în care sunt modificate ponderările.

Parametrul **decay** indică faptul că rata de învățare va fi redusă de la un ciclu la următorul. Parametrul **shuffle** solicită ca ordinea cazurilor din setul de date să fie aleatoare, iar parametrul **normalize** rescalează automat valorile atributelor pe intervalul [-1;1]. Parametrul **epsilon** indică valoarea erorii la care procesul de optimizare (instruire) se oprește. Dacă selectăm parametrul **use local random seed** și indicăm o valoare pentru **local random seed**, vom obține aceeași soluție de fiecare dată când rulăm procesul.

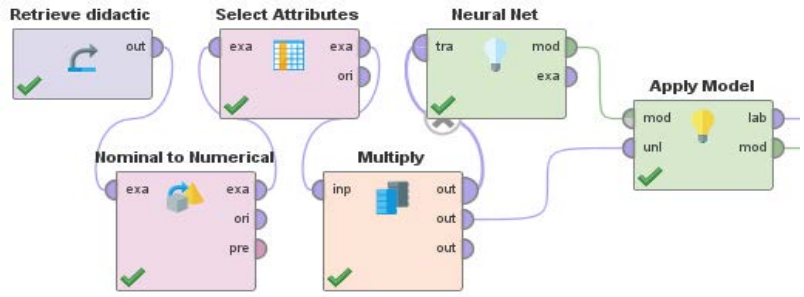
Descrierea unei rețele neuronale simple și a procesului de estimare

În Figura 1.3-1 am prezentat un exemplu simplu de rețea neuronală așa cum am definit-o în RapidMiner, respectiv valorile estimate ale coeficienților și predicțiile obținute. Setul de date este unul foarte simplu, având doar două atribute independente și 10 cazuri (este același set folosit și în capitolele anterioare în cadrul exemplelor didactice). Rețeaua neuronală include un singur strat ascuns, definit automat de RapidMiner. Acesta include, definit tot automat, trei neuroni (formula de calcul este: (2 atribute + 2 clase) / 2 + 1 = 4 / 2 + 1 = 3). În reprezentarea grafică realizată de RapidMiner, ultimul cerculeț din stratul ascuns e de fapt constanta (threshold / bias), nu un neuron propriu-zis (la fel și ultimul cerculeț din stratul de input). Pentru a evita o posibilă confuzie între neuron și constantă, în graficul realizat manual am preferat să nu includ un cerculeț suplimentar pentru fiecare constantă și am sugerat prezența acesteia folosind litera C poziționată în interiorul fiecărui neuron ascuns, respectiv de output.

Setările operatorului Neural Net sunt parțial modificate comparativ cu setările implicite: deoarece am preferat ca instruirea să pornească de la o rată de învățare mare care să scadă în timp, am crescut rata de învățare la 0.9 (aproape valoarea maximă) și am selectat parametrul Decay; pentru a permite schimbarea mai rapidă a direcției în care se schimbă coeficienții am crescut valoarea parametrului momentum de la 0.1 la 0.9 (aproape valoarea maximă); pentru a obține aceleași rezultate de fiecare dată, am setat un număr aleator (random seed).

Figura 9.2-1. Un exemplu didactic de rețea neuronală în RapidMiner

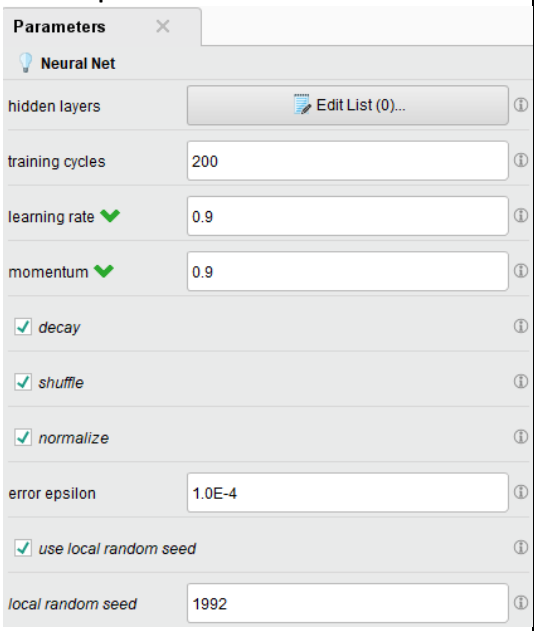
Modelul de predicție:



Setul de date:

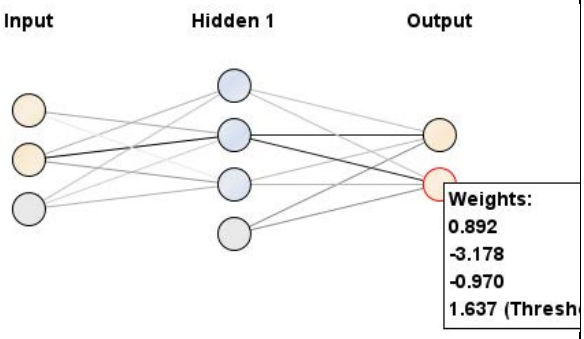
Row No.	Id	Pleacă	Copii	Insat
1	1	da	nu	10
2	2	da	nu	9
3	3	da	nu	8
4	4	nu	nu	7
5	5	nu	nu	6
6	6	da	da	5
7	7	nu	da	3
8	8	nu	da	2
9	9	nu	da	1
10	10	nu	da	0

Setările operatorului Neural Net:



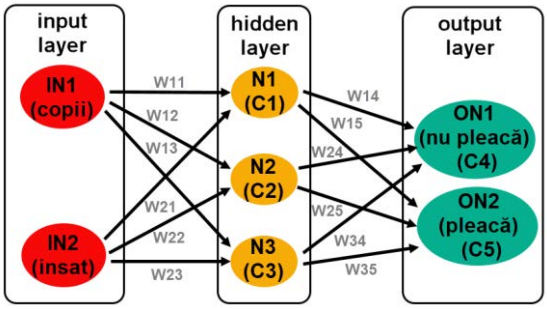
Rețeaua neuronală:
Grafic RapidMiner

Ultimul nod al fiecăruia dintre straturile Input și Hidden 1 este constanta (bias, threshold).
Valoarea absolută a ponderărilor este indicată prin linii relativ mai închise la culoare și mai groase.
Valorile coeficienților asociate unui nod apar doar dacă selectăm acel nod.



Rețeaua neuronală:
Grafic manual cu explicații

IN_1-IN_2 = noduri / neuroni input
N_1-N_3 = noduri / neuroni ascunși
ON_1-ON_2 = noduri / neuroni output
W_{11}-W_{35} = ponderările (weights)
C_1-C_5 = constantele (bias, threshold)



Rețeaua neuronală:
Descriere matematică (RapidMiner)

Valorile finale ale coeficienților (weights și biases) la nivelul fiecărui neuron din stratul ascuns, respectiv de output. Observăm că toate funcție de transformare sunt de tip sigmoid (logistic). De asemenea, constanta este denumită Bias dacă neuronul aparține de stratul ascuns, respectiv Threshold dacă neuronul aparține de stratul de output.

Hidden 1
========

Node 1 (Sigmoid)

Copii = da: 0.011
Insat: -1.077
Bias: -0.899

Node 2 (Sigmoid)

Copii = da: 1.223
Insat: 3.615
Bias: -0.690

Node 3 (Sigmoid)

Copii = da: -0.305
Insat: 1.432
Bias: -1.107

Output
======

Class 'da' (Sigmoid)

Node 1: -0.897
Node 2: 3.152
Node 3: 0.994
Threshold: -1.632

Class 'nu' (Sigmoid)

Node 1: 0.892
Node 2: -3.178
Node 3: -0.970
Threshold: 1.637

Setul de date + predicțiile:

Row No.	Id	Pleacă	prediction(Pleacă)	confidence(da)	confidence(nu)	Copii = da	Insat
1	1	da	da	0.829	0.171	0	10
2	2	da	da	0.752	0.248	0	9
3	3	da	da	0.622	0.378	0	8
4	4	nu	nu	0.459	0.541	0	7
5	5	nu	nu	0.322	0.678	0	6
6	6	da	da	0.573	0.427	1	5
7	7	nu	nu	0.278	0.722	1	3
8	8	nu	nu	0.195	0.805	1	2
9	9	nu	nu	0.150	0.850	1	1
10	10	nu	nu	0.127	0.873	1	0

În Tabelul 9.2-1 am ilustrat pașii parcurși pentru a ajunge de la date la predicții. Obiectivul este de a analiza aceleași date și de a ajunge la predicțiile obținute cu ajutorul RapidMiner (Figura 9.2-1, ultima linie). **Primul pas** constă în rescalarea valorilor atributelor pe intervalul [-1;1]. Îl putem realiza simplu folosind formula:

$$X_{rescalat[-1;1]} = 2 * \frac{X - X_{min}}{X_{max} - X_{min}} - 1$$

De exemplu, relativ la cel de al doilea caz din setul de date, observăm că atributul Insat (insatisfacția muncii) ia valoarea 9. Dorim să rescalăm această valoare pe intervalul [-1;1]. Observăm că valoarea maximă înregistrată pentru același atribut este 10, iar valoarea minimă este 0. Aplicând formula de mai sus, valoarea 9 rescalată devine 0.8 $(2 * \frac{9-0}{10-0} - 1)$. Alternativ, am fi putut rescala datele pe intervalul [0;1] folosind formula:

$$X_{rescalat[0;1]} = \frac{X - X_{min}}{X_{max} - X_{min}}$$

La pasul doi, folosind valorile rescalate ale nodurilor de input și valorile coeficienților estimați (weight și bias), calculăm valorile nodurilor din stratul ascuns folosind formula:

$$valoare\ nod_j = x_1 * w_{1j} + x_2 * w_{2j} + \cdots x_i * w_{ij} + bias_j$$

Valoarea fiecărui nod ascuns este egală cu suma fiecărei valori de input înmulțită cu ponderarea la care adăugăm biasul asociat nodului respectiv. De exemplu, pentru cazul 1, valoarea primului nod ascuns va fi egală cu -1.987. Procedăm la fel pentru toate cazurile și nodurile ascunse.

$$-1 * 0.011 + 1 * (-1.077) - 0.899 = 0.011 - 1.077 - 0.899 = -1.987$$

imput 1 · weight 1 · imput 2 · weight 2 · bias

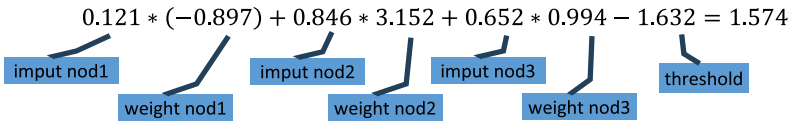
La **pasul trei** transformăm valorile asociate nodurilor invizibile folosind o funcție de tip sigmoid (logistică; pot fi folosite și alte funcții, ReLU[9] de exemplu):

$$valoare\ nod_j\ transformată\ sigmoid = \frac{1}{1 + e^{-valoare\ nod_j}}$$

De exemplu, relativ la primul caz și primul nod, valoarea transformată este 0.121 $(\frac{1}{1+e^{1.987}})$. La **pasul patru** calculăm valoarea de input asociată fiecărei clase a variabilei de interes (Pleacă). Similar cu pasul doi, valoarea fiecărui nod de output

[9] Rectified Linear Unit

este egală cu suma fiecărei valori de input înmulțită cu ponderarea la care adăugăm constanta (threshold) asociată nodului respectiv. De exemplu, pentru primul caz și clasa DA, această valoare este 1.574. Procedăm la fel pentru toate cazurile și nodurile de output.



La **pasul cinci** transformăm valorile obținute la nodurile de output folosind tot o funcție de tip sigmoid. De exemplu, pentru primul caz și clasa DA, valoarea 1.574 transformată devine 0.828, iar pentru primul caz și clasa NU, valoarea transformată este 0.171. Deoarece valoarea asociată clasei DA este mai mare, predicția va fi că primul caz aparține clasei DA. Observăm că predicția este corectă. Procedăm la fel pentru toate cazurile și nodurile de output și observăm că toate predicțiile sunt corecte.

Tabelul 9.2-1. Un exemplu didactic de rețea neuronală: de la date la predicții

Setul de date inițial				Valori noduri input rescalate		Valori noduri invizibile			Valori noduri invizibile transformate			Valori noduri output		Valori noduri output transformate		Predicția modelului	
Id	Pleacă	Copii	Insat	Copii	Insat	N1	N2	N3	N1	N2	N3	Clasa DA	Clasa NU	Clasa DA	Clasa NU	Clasa prezisă	Evaluare predicție
1	da	0	10	-1	1.0	-1.987	1.702	0.630	0.121	0.846	0.652	1.574	-1.576	0.828	0.171	da	corectă
2	da	0	9	-1	0.8	-1.772	0.979	0.344	0.145	0.727	0.585	1.110	-1.111	0.752	0.248	da	corectă
3	da	0	8	-1	0.6	-1.556	0.256	0.057	0.174	0.564	0.514	0.500	-0.498	0.622	0.378	da	corectă
4	nu	0	7	-1	0.4	-1.341	-0.467	-0.229	0.207	0.385	0.443	-0.163	0.168	0.459	0.542	nu	corectă
5	nu	0	6	-1	0.2	-1.125	-1.190	-0.516	0.245	0.233	0.374	-0.745	0.752	0.322	0.680	nu	corectă
6	da	1	5	1	0.0	-0.888	0.533	-1.412	0.292	0.630	0.196	0.288	-0.296	0.571	0.427	da	corectă
7	nu	1	3	1	-0.4	-0.457	-0.913	-1.985	0.388	0.286	0.121	-0.957	0.955	0.277	0.722	nu	corectă
8	nu	1	2	1	-0.6	-0.242	-1.636	-2.271	0.440	0.163	0.094	-1.420	1.421	0.195	0.805	nu	corectă
9	nu	1	1	1	-0.8	-0.026	-2.359	-2.558	0.493	0.086	0.072	-1.731	1.733	0.150	0.850	nu	corectă
10	nu	1	0	1	-1.0	0.189	-3.082	-2.844	0.547	0.044	0.055	-1.930	1.932	0.127	0.874	nu	corectă

În Tabelul 9.2-1 am folosit valorile finale estimate ale coeficienților (weights, biases/thresholds). Evident, la momentul de start al învățării (primul ciclu), valorile coeficienților nu sunt cunoscute, algoritmul alegând aleatoriu valorile de start ale acestora. Desigur, procedând astfel rețeaua neuronală produce cel mai adesea multe predicții greșite. De exemplu, dacă valorile coeficienților ar fi cele din Tabelul 9.2-2, la finalul primului ciclu, respectiv al celui de al doilea, rețeaua neuronală ar

prezice că toate cazurile aparțin clasei NU, deci patru predicții ar fi incorecte (cazurile sunt DA, dar rețeaua prezice că sunt NU) (Tabelul 9.2-3).

Tabelul 9.2-2. Valorile coeficienților (exemplul didactic)

Ciclul 1			
Nod ascuns	Weight Copii	Weight Insat	Bias
N1	-0.322	0.236	-0.283
N2	-0.375	0.320	-0.355
N3	-0.309	0.295	-0.286

Ciclul 2			
Nod ascuns	Weight Copii	Weight Insat	Bias
N1	-0.466	0.281	-0.715
N2	-0.555	0.423	-0.766
N3	-0.455	0.423	-0.707

Ciclul 1				
Nod output	Weight N1	Weight N2	Weight N3	Threshold
DA	-0.295	-0.234	-0.262	-0.646
NU	0.302	0.185	0.281	0.627

Ciclul 2				
Nod output	Weight N1	Weight N2	Weight N3	Threshold
DA	-0.564	-0.420	-0.489	-1.350
NU	0.566	0.365	0.503	1.326

Tabelul 9.2-3. Predicțiile și erorile după fiecare dintre primele două cicluri de instruire (exemplul didactic)

1	Va-lori obs.	Valori input rescalate		Valori output			Valori transformate			Valori output		Valori transformate		Predicția modelului		Eroare absolută	
Id	Pleacă	Copii = da	Insat	N1	N2	N3	N1	N2	N3	Clasa DA	Clasa NU	Clasa DA	Clasa NU	Clasa prezisă	Predicție	Clasa DA	Clasa NU
1	da	-1	1.0	0.275	0.340	0.318	0.568	0.584	0.579	-1.102	1.069	0.249	0.751	nu	incorectă	0.141	0.141
2	da	-1	0.8	0.228	0.276	0.259	0.557	0.569	0.564	-1.091	1.059	0.251	0.749	nu	incorectă	0.141	0.141
3	da	-1	0.6	0.181	0.212	0.200	0.545	0.553	0.550	-1.080	1.048	0.253	0.747	nu	incorectă	0.141	0.141
4	nu	-1	0.4	0.133	0.148	0.141	0.533	0.537	0.535	-1.069	1.038	0.256	0.744	nu	corectă	0.142	0.142
5	nu	-1	0.2	0.086	0.084	0.082	0.522	0.521	0.520	-1.058	1.027	0.258	0.742	nu	corectă	0.142	0.142
6	da	1	0.0	-0.605	-0.730	-0.595	0.353	0.325	0.355	-0.919	0.894	0.285	0.715	nu	incorectă	0.146	0.146
7	nu	1	-0.4	-0.699	-0.858	-0.713	0.332	0.298	0.329	-0.900	0.875	0.289	0.711	nu	corectă	0.146	0.146
8	nu	1	-0.6	-0.747	-0.922	-0.772	0.322	0.285	0.316	-0.890	0.866	0.291	0.709	nu	corectă	0.146	0.146
9	nu	1	-0.8	-0.794	-0.986	-0.831	0.311	0.272	0.303	-0.881	0.857	0.293	0.707	nu	corectă	0.146	0.146
10	nu	1	-1.0	-0.841	-1.050	-0.890	0.301	0.259	0.291	-0.872	0.848	0.295	0.705	nu	corectă	0.147	0.147
															Total	1.437	1.437

2	Valori obs.	Valori input rescalate		Valori output			Valori transformate			Valori output		Valori transformate		Predicția modelului		Eroare absolută	
Id	Pleacă	Copii = da	Insat	N1	N2	N3	N1	N2	N3	Clasa DA	Clasa NU	Clasa DA	Clasa NU	Clasa prezisă	Predicție	Clasa DA	Clasa NU
1	da	-1	1.0	0.032	0.212	0.171	0.508	0.553	0.543	-2.134	2.088	0.106	0.894	nu	incorectă	0.085	0.085
2	da	-1	0.8	-0.024	0.127	0.086	0.494	0.532	0.522	-2.107	2.062	0.108	0.892	nu	incorectă	0.086	0.086
3	da	-1	0.6	-0.080	0.043	0.002	0.480	0.511	0.500	-2.080	2.036	0.111	0.889	nu	incorectă	0.088	0.088
4	nu	-1	0.4	-0.137	-0.04	-0.083	0.466	0.490	0.479	-2.053	2.009	0.114	0.886	nu	corectă	0.089	0.089
5	nu	-1	0.2	-0.193	-0.12	-0.167	0.452	0.468	0.458	-2.026	1.983	0.117	0.883	nu	corectă	0.091	0.091
6	da	1	0.0	-1.181	-1.32	-1.162	0.235	0.211	0.238	-1.687	1.656	0.156	0.844	nu	incorectă	0.111	0.111
7	nu	1	-0.4	-1.293	-1.49	-1.331	0.215	0.184	0.209	-1.651	1.620	0.161	0.839	nu	corectă	0.113	0.113
8	nu	1	-0.6	-1.350	-1.57	-1.416	0.206	0.172	0.195	-1.634	1.603	0.163	0.837	nu	corectă	0.114	0.114
9	nu	1	-0.8	-1.406	-1.65	-1.500	0.197	0.160	0.182	-1.617	1.588	0.166	0.834	nu	corectă	0.115	0.115
10	nu	1	-1.0	-1.462	-1.74	-1.585	0.188	0.149	0.170	-1.602	1.572	0.168	0.832	nu	corectă	0.116	0.116
															Total	1.009	1.009

Erorile din Tabelul 9.2-3 sunt calculate folosind formula:

$$err_k = \hat{y}_k(1 - \hat{y}_k)(y_k - \hat{y}_k)$$

unde k este numărul cazului, y_k este 0 dacă nodul de output se referă la clasa NU, respectiv 1 dacă nodul de output se referă la clasa DA, iar $\hat{y}_k$ este probabilitatea estimată de model ca acel caz să aparțină clasei k (0=NU sau 1=DA în acest caz). În tabel am trecut doar erorile absolute. De exemplu, după primul ciclu de instruire, pentru primul caz din setul de date didactic (aparține de clasa DA) eroarea asociată nodului de output DA este egală cu 0.249 * (1 - 0.249) * (1 - 0.249), adică 0.141, iar eroarea asociată nodului de output NU este egală cu 0.751 * (1 - 0.751) * (0 - 0.751), adică -0.141 (valoarea absolută este 0.141). Tot relativ la primul caz, după ciclul doi eroarea scade la 0.085 pentru fiecare din cele două clase. De asemenea, observăm că suma erorilor relativ la fiecare clasă scade și ea (de la 1.437 la 1.009).

Cu privire la Tabelul 9.2-3 observăm că valorile coeficienților (weights și biases) diferă de la ciclul 1 la ciclul 2. Ajustarea valorilor coeficienților se realizează în așa fel încât eroarea să scadă. Pentru aceasta, e nevoie ca informația cu privire la mărimea erorii să fie transmisă dinspre stratul de output înspre neuronii din stratul ascuns și apoi înspre neuronii din stratul de input (de aici și numele algoritmului: propagare inversă). Noua valoare a unui coeficient se calculează ținând cont de vechea valoare (θ = bias; w = weight), de rata învățării (l), de momentum (m) și de eroare (err). Prima dată calculăm rata schimbării, apoi valoarea nouă a coeficientului. Formulele utilizate sunt cele de mai jos (am exemplificat doar pentru coeficientul weight):

$$\Delta w_{i,j} = l * err_j * n_{i,j}$$

$$w_{i,j}^{new} = w_{i,j}^{old} + \Delta w_{i,j} + m * \Delta_{t-1}$$

unde

Δw = rata schimbării coeficientului $weight_{i,j}$
Δ_{t-1} = rata schimbării la ciclul anterior
w^{new} = valoarea nouă a coeficientului weight
w^{old} = valoarea anterioară a coeficientului weight
l = parametrul rata învățării
m = parametrul momentum
err = eroarea
n = valoarea nodului

Procesul de ajustare începe cu recalcularea coeficienților din stratul de output, apoi a celor din stratul ascuns, după care rețeaua neuronală recalculează eroarea finală folosind valorile noi ale coeficienților și tot așa, iterativ, până când eroarea scade sub pragul indicat sau până se atinge numărul de cicluri specificat. Dacă selectăm opțiunea Decay, rata de învățare va fi tot mai mică pe măsură ce trecem de la un ciclu la următorul. Scăderea treptată a ratei de învățare crește șansele de a găsi cea mai bună soluție.

Ajustarea coeficienților se poate face prin una dintre următoarele două modalități: (1) ajustarea după fiecare caz, deci coeficienții se modifică de fiecare dată ce un caz trece prin rețea (case updating) și (2) ajustarea după ce toate cazurile au trecut prin rețea, procesul reluându-se de numărul de ori specificat de analist (batch updating). În cazul modalității (2), eroarea este egală cu suma erorilor relativ la toate cazurile. Ajustarea coeficienților după fiecare caz duce în final la predicții mai bune, dar procesul de estimare durează mai mult. RapidMiner folosește ajustarea caz cu caz (case updating).

9.3. Exemple de utilizare a rețelelor neuronale în RapidMiner

Pentru o prezentare introductivă cu privire la construirea unei rețele neuronale în RapidMiner se poate vedea filmulețul Neural Nets demo. Pentru o serie de alte exemple se pot consulta cărțile recomandate la începutul acestui capitol.

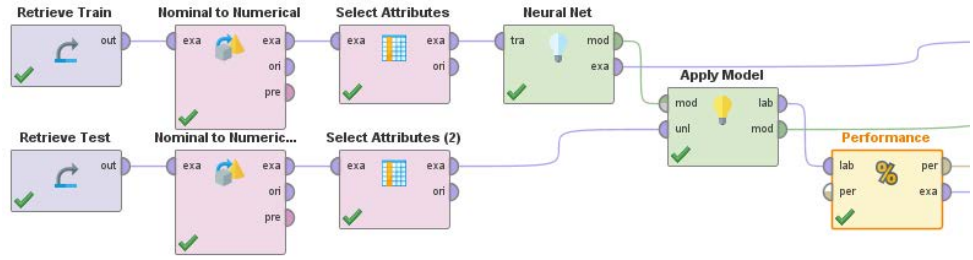
Un model simplu

Pentru acest exemplu vom folosi aceleași trei atribute din setul de date „employee_attrition” și vom realiza un model de clasificare de tip rețea neuronală. Scopul analizei este predicția situațiilor de părăsire a companiei în funcție de vechimea în muncă și munca peste program. Desigur, ne așteptăm ca o vechime mai mare să reducă șansele de plecare, iar munca peste program să le crească. Pașii realizării acestui model în RapidMiner, rezultatele obținute și interpretarea lor apar în Figura 9.3-1.

Figura 9.3-1. Rețea neuronală: un model simplu în RapidMiner

Pasul 1:

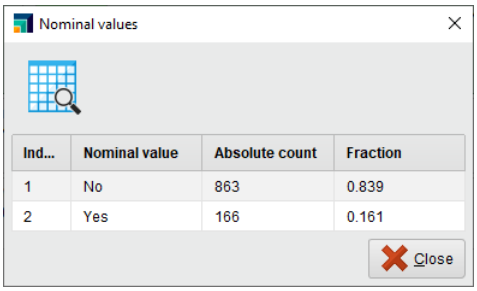
Realizăm procesul din imaginea de mai jos (procesul și datele utilizate sunt incluse în folderul asociat volumului). Seturile de date utilizate sunt „employee_attrition_training_validation" (instruire), respectiv „employee_attrition_testing" (testare). Relativ la ambele seturi de date am păstrat doar atributele Attrition, OverTime și YearsAtCompany (operatorul „Select Attributes"). Includem operatorul Neural Nets cu setările din imagine și restul operatorilor. Rulăm modelul.

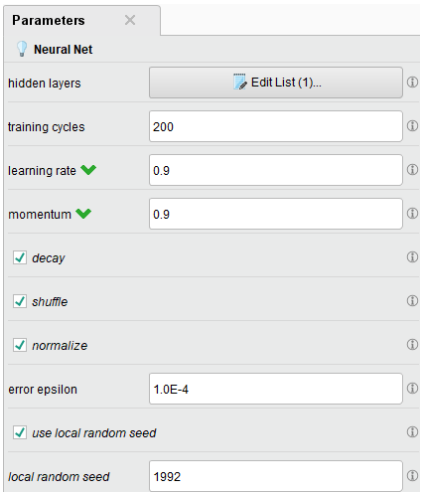


Rezultate (setul de instruire):

Seturile de date conțin doar atributele selectate anterior (în imagine apar primele 5 cazuri din setul de instruire). Distribuția statistică a atributului de interes – Attrition – este 0.161 pentru clasa Yes (pleacă) și 0.839 pentru clasa No (nu pleacă).

Row No.	Id	Attrition	OverTime = Yes	YearsAtCompany
1	4	Yes	1	0
2	5	No	1	8
3	7	No	0	2
4	10	No	1	1
5	14	No	0	5





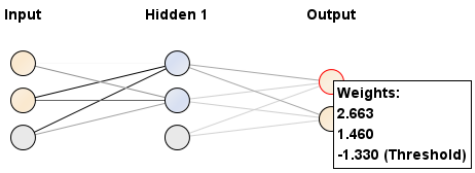
Rezultate (setul de testare):

Relativ la setul de testare, modelul produce predicțiile din imagine (am inclus primele 5 cazuri și alte trei de interes în contextul acestui exemplu). Coloana Attrition prezintă situația reală, iar coloana „prediction(Attrition)" predicția realizată de model. Observăm că obținem și predicții greșite (Id 1 și 165). Probabilitățile asociate predicțiilor apar pe coloanele „confidence". Relativ la fiecare caz, clasa care are asociată cea mai mare probabilitate (confidence) prezisă, determină clasa la care modelul prezice că aparține acel caz. Pragul de probabilitate implicit în funcție de care un caz este prezis a aparține unei clase sau alteia este 50%. Astfel, dacă probabilitatea este sub 50% modelul prezice că acel caz aparține clasei No (angajatul nu pleacă). Dacă probabilitatea este cel puțin 50%, clasa prezisă va fi Yes (angajatul pleacă). Cazurile au asociate diferite probabilități de a pleca. Cea mai mică probabilitate estimată este 6% – angajatul 165 – are o vechime de 40 ani și nu muncește peste program (mai sunt și alte cazuri cu o probabilitate similară). Cea mai mare probabilitate, 0.598 (59.8%), apare în cazul angajaților 167 și 811 (amândoi au o vechime mai mică de un an și muncesc peste program). Dat fiind faptul că pragul de probabilitate implicit este 0.5, modelul prezice că angajații 167 și 811 vor pleca.

Row No.	Id	Attrition	prediction(Attrition)	confidence(No)	confidence(Yes)	OverTime = Yes	YearsAtCompany
1	1	Yes	No	0.741	0.259	1	6
2	2	No	No	0.939	0.061	0	10
3	8	No	No	0.930	0.070	0	7
4	11	No	No	0.795	0.205	0	1
5	12	No	No	0.937	0.063	0	9
...							
41	167	Yes	Yes	0.402	0.598	1	0
...							
40	165	Yes	No	0.942	0.058	0	40

...

Rețeaua neuronală apare la secțiunea Improuved Neural Net. La tabul Neural Net este prezentată varianta grafică rețelei, iar la Description transpunerea matematică. Observăm că stratul de input conține trei neuroni: cele două atribute independente (OverTime și YearsAtCompany) și Constanta (Bias / Threshold). Stratul invizibil include tot trei neuroni, unul fiind o altă Constantă, iar stratul de output este format din doi neuroni asociați celor două clase ale atributului de interes. Observăm că neuronii asociați constantelor nu au intrări, doar ieșiri. Reprezentarea grafică a legăturilor dintre doi neuroni oarecare sugerează intensitatea legăturii (weight) dintre aceștia. Cu cât linia respectivă este mai închisă la culoare, cu atât legătura este mai intensă în termeni absoluți (intensitatea poate fi negativă sau pozitivă). Observăm că legăturile dintre atributul de input vechime în muncă și neuronii invizibili sunt relativ mai intense (comparativ cu legăturile asociate atributului muncă peste program). Valorile ponderărilor și constantelor pot fi văzute la tabul Description. Niciuna dintre cele două variante de prezentare a modelului nu ne spune nimic despre cum anume ajungem de la valorile de input la cele de output, câți angajați pleacă și câți rămân, respectiv ce caracteristici au aceștia.



Hidden 1
========

Node 1 (Sigmoid)

OverTime = Yes: -0.184
YearsAtCompany: 7.925
Bias: 7.481

Node 2 (Sigmoid)

OverTime = Yes: -3.413
YearsAtCompany: 4.850
Bias: 2.864

Output
======

Class 'No' (Sigmoid)

Node 1: 2.663
Node 2: 1.460
Threshold: -1.330

Class 'Yes' (Sigmoid)

Node 1: -2.663
Node 2: -1.460
Threshold: 1.330

Secțiunea PerformanceVector conține matricea de confuzie și diferite măsuri ale performanței clasificării (cele selectate) calculate relativ la setul de date de testare. Acuratețea predicției este mare (84%) doar pentru că majoritatea angajaților nu pleacă. Relativ la clasa de interes pentru noi, cei care pleacă, calitatea predicției e destul de slabă: precizia clasificării acestei clase este 50% (jumătate dintre cazurile prezise că pleacă chiar pleacă), iar reamintirea doar 17% (modelul identifică doar 17% dintre angajații care pleacă efectiv). Pragul implicit al clasificării este 50%. Pentru a crește calitatea modelului relativ la clasa de interes, putem reduce aceste prag (Threshold) la 30% de exemplu și să clasificăm toate cazurile cu probabilitatea prezisă de a pleca de cel puțin 30% ca angajați care pleacă. Desigur, în acest caz ponderea erorilor de tip I va crește.

accuracy: 83.90%

	true No	true Yes	class precision
pred. No	358	59	85.85%
pred. Yes	12	12	50.00%
class recall	96.76%	16.90%	

set testare, prag implicit 50%

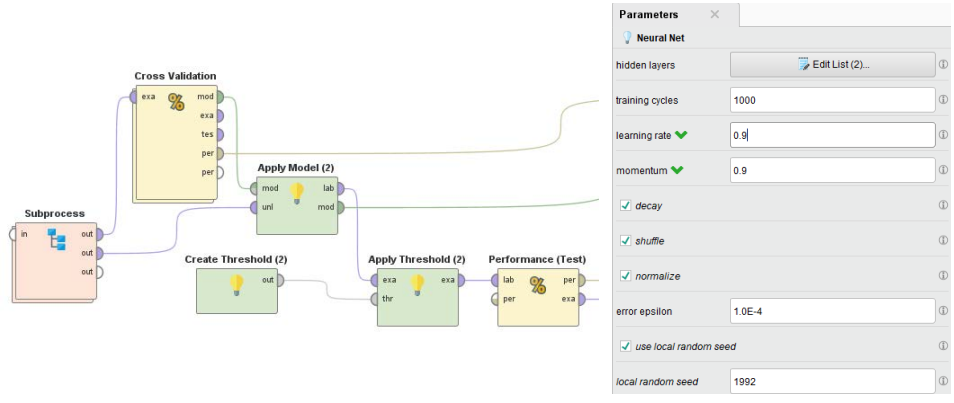
Un model relativ mai complex

Pentru acest exemplu vom folosi setul de date „employee_attrition”. Ne propunem să realizăm un model de clasificare bazat pe o rețea neuronală cu scopul de a prezice situațiile de părăsire a companiei. Pașii realizării acestui model în RapidMiner, rezultatele obținute și interpretarea lor apar în Figura 9.3-2.

Figura 9.3-2. Rețea neuronală: un model relativ mai complex în RapidMiner

Pasul 1:

Realizăm procesul din imaginea de mai jos. În folderul asociat volumului am inclus două versiuni ale acestui proces: o versiune mai simplă, fără setarea pragului de probabilitate (acesta este stabilit implicit la 0.5) și o versiune care oferă posibilitatea setării pragului de probabilitate. Toate datele necesare sunt pregătite anterior în interiorul operatorului Subprocess. Setul de date utilizat pentru instruirea modelului este „employee_attrition_training_validation". Setul utilizat pentru testarea modelului este „employee_attrition_testing". Includem operatorul Neural Net cu setările modificate conform imaginii (modelul include două straturi invizibile, primul cu 10 neuroni, al doilea cu cinci). În cazul modelului 2 includem operatorii care ne ajută să setăm pragul de probabilitate dorit (Create Threshold și Apply Threshold). Includem operatorul necesar pentru aplicarea modelului pe setul de date de testare, respectiv operatorul care calculează măsurile de performanță. Realizăm conexiunile și rulăm procesul. La modelul 1, pragul de probabilitate este setat implicit la 0.5 (prima clasă este No, a doua Yes). Prin urmare, modelul va prezice că angajații în cazul cărora valoarea atributului „confidence(Yes)" va fi mai mare de 0.5 vor pleca, iar restul vor rămâne. La modelul 2, pragul este stabilit manual la 0.3.



Rezultate (setul de testare):

Relativ la setul de testare, modelul produce predicțiile din imagine (am inclus primele 5 cazuri și alte trei de interes în contextul acestui exemplu). Coloana Attrition prezintă situația reală, iar coloana „prediction(Attrition)" predicția realizată de model. Observăm că unele predicții sunt greșite (în imagine, cele cu Id 1, 1038 și 746). Probabilitățile asociate predicțiilor apar pe coloanele „confidence". Pentru fiecare caz este selectată ca predicție clasa care are cea mai mare probabilitate (confidence).

Row No.	Id	Attrition	prediction(Attrition)	confidence(No)	confidence(Yes)
1	1	Yes	Yes	0.000	1.000
2	2	No	No	1.000	0.000
3	8	No	No	0.991	0.009
4	11	No	No	1.000	0.000
5	12	No	Yes	0.480	0.520
...					
225	1038	Yes	No	1.000	0.000
220	1026	No	No	1.000	0.000
...					
168	746	No	Yes	0.413	0.587
41	167	Yes	Yes	0.396	0.604
...					
415	1944	Yes	Yes	0.000	1.000
...					

Secțiunea PerformanceVector conține matricea de confuzie și diferite măsuri ale performanței clasificării (cele selectate), toate calculate relativ la ambele seturi de date (instruire și testare). Observăm că acuratețea predicției este de fiecare dată peste 80%, ceea ce indică șanse mari de generalizare a modelului de predicție. Chiar dacă de obicei calitatea clasificării este mai bună în cazul setului de instruire comparativ cu cel de test, de această dată este invers. Comparativ cu modelul care a inclus doar doi predictori, modelul cu toți predictorii are o acuratețe a clasificării similară (aceasta e doar puțin mai mare), dar reușește că clasifice corect o pondere mai mare a cazurilor de angajați care pleacă (ceea ce e foarte util din punct de vedere practic). În continuare, modelul prezice mai bine situația în care angajații nu pleacă (precizia = 90/91%, reamintirea = 94/92%) comparativ cu situația în care angajații pleacă (precizia = 60/57%, reamintirea = 48/54%).

accuracy: 85.03% +/- 2.50% (micro average: 85.03%)

	true No	true Yes
pred. No	813	104
pred. Yes	50	62
class recall	94.21%	37.35%

instruire, prag 50%

accuracy: 84.26% +/- 1.88% (micro average: 84.26%)

	true No	true Yes
pred. No	800	99
pred. Yes	63	67
class recall	92.70%	40.36%

instruire, prag 30%

accuracy: 86.39%

	true No	true Yes
pred. No	347	37
pred. Yes	23	34
class recall	93.78%	47.89%

testare, prag 50%

accuracy: 85.94%

	true No	true Yes
pred. No	341	33
pred. Yes	29	38
class recall	92.16%	53.52%

testare, prag 30%

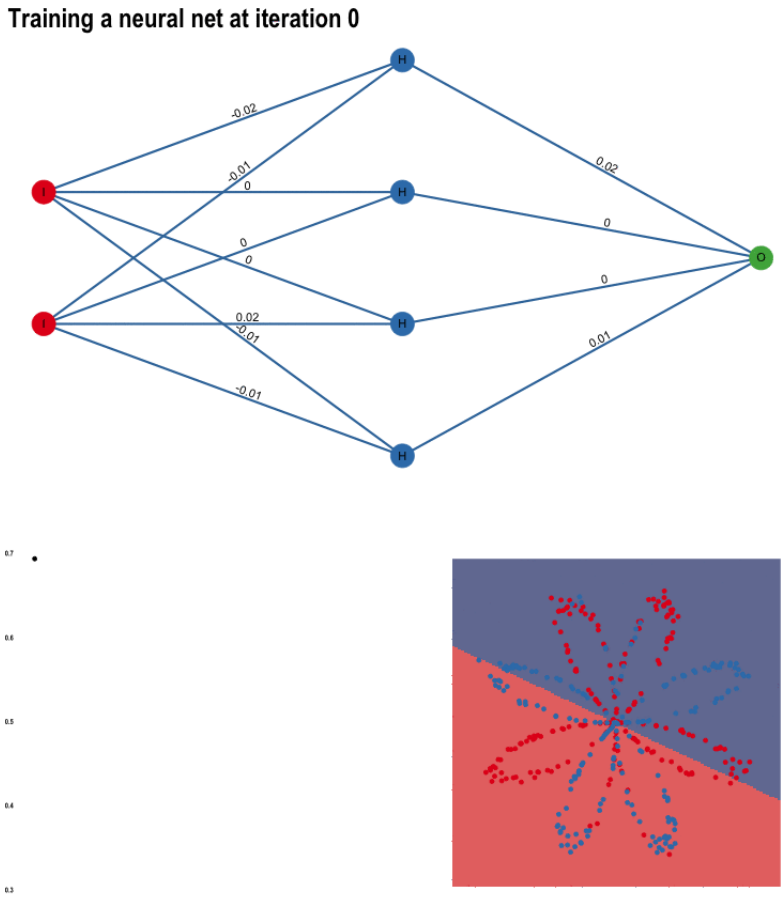
Un exemplu relativ mai complex folosind operatorul Deep Learning

Operatorul Neural Net prezentat anterior permite o definire relativ mai limitată a parametrilor și valorilor asociați unei rețele neuronale. Dacă dorim o flexibilitate mai mare, folosim operatorul Deep Learning (instruire / învățare profundă). Operatorul Deep Learning implementează într-o manieră mult mai flexibilă același tip de rețea neuronală precum Neural Nets (multi-layer feed-forward artificial neural network that is trained with stochastic gradient descent using back-propagation). Operatorul Deeep Learning are în spate de fapt algoritmul companiei H2O. Desigur, utilizarea acestui operator nu duce neapărat la un model mai performant. Câteva prezentări video în care este discutat și folosit operatorul în RapidMiner sunt următoarele: Introduction to Deep Learning, Deep Learning for Classification, An Introduction to Deep Learning | RapidMiner.

Utilitatea unei rețele neuronale cu multe straturi și neuroni, a unei rețele care permite o definire cât mai flexibilă a unui număr cât mai mare a parametrilor, este cu atât mai evidentă cu cât funcția care stă în spatele datelor este una mai complexă. Să considerăm un exemplu simplu precum cel din imaginea de mai jos (Figura 9.3-3). Fără a fi extrem de complicată, imaginea respectivă este una relativ complicată. Cu

toate acestea, o rețea neuronală cu doar două atribute de input și un strat ascuns cu doar patru neuroni, poate învăța relativ repede funcția din spatele imaginii și să clasifice corect majoritatea cazurilor. Observăm că pe măsură ce rețeaua învață, coeficienții se modifică, funcția estimată se apropie tot mai mult de funcția „reală", iar eroarea de clasificare scade.

Figura 9.3-3. O ilustrare a procesului de instruire a unei rețele neuronale care modelează o funcție relativ mai complexă



Sursa: Tamas Szilagyi. 2017. Coding with Data. An animated neural net implementation.

În cele ce urmează vom exemplifica utilizarea operatorului Deep Learning folosind același exemplu (cel prezentat în secțiunea anterioară). Pentru început, prezentăm pe scurt parametrii operatorului Deep Learning:

- **activation**: alegem funcția de activare (transformare) dorită; putem alege una dintre următoarele funcții: Tanh, TanhWithDropout, Rectifier, RectifierWithDropout, Maxout, MaxoutWithDropout, ExpRectifier, ExpRectifierWithDropout; funcția implicită este Rectifier;
- **hidden dropout ratios**: opțiunea devine activă doar dacă am ales o funcție de activare care implică neincluderea unor atribute în etapa de instruire (dropout); valoarea aleasă indică ponderea atributelor de input care vor fi omise relativ la fiecare strat ascuns; scopul este de a crește șansele de generalizare;
- **hidden layer sizes**: aici setăm numărul straturilor ascunse, respectiv numărul neuronilor din care este format fiecare strat;
- **reproducible**: pentru a obține de fiecare dată exact același model, e nevoie să selectăm acest parametru;
- **use local random seed**: numărul aleator poate fi specificat doar dacă am ales anterior reproducible;
- **epochs**: numărul de epoci / cicluri de instruire, adică de câte ori algoritmul parcurge toate cazurile din setul de date; valoarea implicită este 10;
- **compute variable importances**: calculează importanța atributelor (cele mai importante, respectiv cele mai puțin importante);
- **train samples per iteration**: numărul de cazuri care va fi procesat la fiecare iterație (când vor fi calculate scorurile, nu când va fi actualizat modelul; actualizarea se face după fiecare caz); valoarea 0 semnifică o epocă pe iterație, -1 indică numărul maxim de cazuri pe iterație, iar -2 impune ca numărul de cazuri să fie stabilit automat;
- **adaptative rate**: implementează un algoritm de instruire adaptată (ADADELTA); prin ajustarea ratei de învățare, între straturi și de la un ciclu la următorul, algoritmul ajută la obținerea mai rapidă a convergenței, simultan cu evitarea minimelor locale;
- **epsilon**: eroarea maximă admisă; devine activ doar dacă parametrul adaptative rate este selectat;
- **rho**: este echivalentul parametrului momentum al operatorului Neural Net; valorile tipice sunt în intervalul [0.9;0.999]; devine activ doar dacă parametrul adaptative rate este selectat;
- **learning rate**: rata de învățare (similar cu parametrul de la Neural Net); devine activ doar dacă parametrul adaptative rate nu este selectat;
- **rate annealing**: contribuie la evitarea situației în care rata de învățare rămâne blocată într-o zonă de minimum local; devine activ doar dacă parametrul adaptative rate nu este selectat;

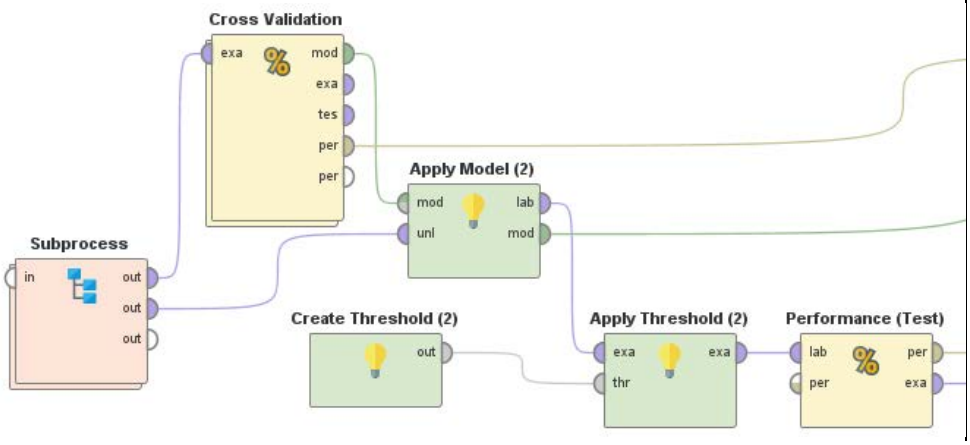
- **rate decay**: rata de învățare scade de la un strat la următorul strat (de exemplu, dacă rata de învățare la stratul 1 este 0.4 iar decay este 0.5, rata de învățare în cazul stratului 2 va fi 0.2); devine activ doar dacă parametrul adaptative rate nu este selectat;
- **momentum start**:; devine activ doar dacă parametrul adaptative rate nu este selectat;
- **momentum ramp**: valoarea momentum poate să crească doar după ce cantitatea de învățare specificată la acest parametru are loc; devine activ doar dacă parametrul adaptative rate nu este selectat;
- **momentum stable**: controlează valoarea finală a parametrului momentum (momentum rămâne același după acest punct); devine activ doar dacă parametrul adaptative rate nu este selectat;
- **nesterov accelerated gradient**: o variantă modificată a funcției gradient clasice; devine activ doar dacă parametrul adaptative rate nu este selectat;
- **standardize**: standardizează atributele;
- **L1**: regularizare de tip L1; conexiunile (weights) cu valori absolute apropiate de zero vor fi ignorate (se consideră că sunt zero); are rolul de a reduce complexitatea modelului și a evita supra-ajustare (overfitting);
- **L2**: regularizare de tip L2; adaugă o anumită cantitate de zgomot (bias) la valorile estimate ale coeficienților cu scopul de a reduce varianța modelului;
- **max w2**: valoarea maximă pe care o poate lua pătratul sumei ponderărilor care intră într-un neuron;
- **loss function**: funcția utilizată pentru calcularea erorii; funcțiile Absolute, Quadratic și Huber sunt potrivite pentru ambele tipuri de probleme (regresie și clasificare); funcția CrossEntropy se folosește doar pentru probleme de clasificare (dintre funcțiile posibile, este cea preferată); funcția Huber ajută în cazul în problemelor de regresie cu outlieri;
- **distribution function**: specifică tipul de distribuție asociat atributelor; pentru atributele nominale considerăm că distribuția este multinominală, iar pentru atributele numerice gausiană; pentru probleme de clasificare a unor atribute binominale distribuția este de tip Bernoulli; pentru probleme de regresie putem alege dintre gaussian, possion, gamma, tweedie, quantile și laplace;
- **early stopping**: instruirea se oprește atunci când valorile specificate ale parametrilor sunt atinse;
- **missing values handling**: în cazul în care setul de date conține cazuri cu valori lipsă, putem alege ca algoritmul să ignore cazurile cu valori lipsă sau să înlocuiască valorile lipsă cu media atributului respectiv (valoarea modală, dacă atributul este categorial);

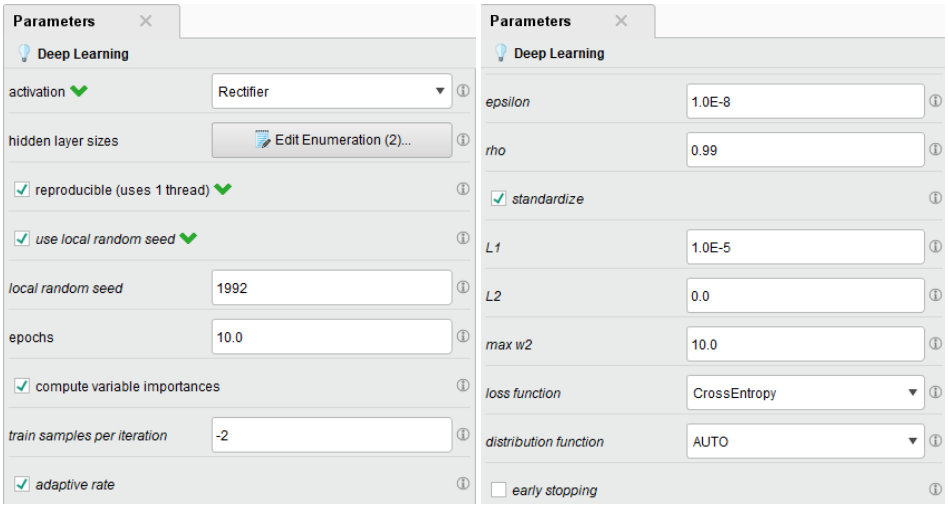
- **max runtime seconds**: durata maximă a instruirii în secunde (0 = oricât);
- **expert parameters**: o serie de alți parametri care ajută la o și mai mare flexibilitate în definirea rețelei neuronale (se poate consulta documentația H2O).

În Figura 9.3-4 am reluat exemplul din secțiunea anterioară, folosind de această dată operatorul Deep Learning. Ca de obicei, am inclus procesul și setările operatorilor, rezultatele obținute și interpretarea lor.

Figura 9.3-4. Rețea neuronală: un model relativ mai complex în RapidMiner – Deep Learning

Pasul 1:
Realizăm procesul din imaginea de mai jos. În folderul asociat volumului am inclus o singură versiune a acestui proces, cea în care pragul de probabilitate este setat la 0.3 (orice alt prag poate fi setat la fel de simplu). Toate datele necesare sunt pregătite anterior în interiorul operatorului Subprocess. Setul de date utilizat pentru instruirea modelului este „employee_attrition_training_validation". Setul utilizat pentru testarea modelului este „employee_attrition_testing". Includem operatorul Deep Learning cu setările modificate conform imaginii (modelul include două straturi ascunse, primul cu 10 neuroni, al doilea cu cinci). Includem operatorii care ne ajută să setăm pragul de probabilitate dorit (Create Threshold și Apply Threshold). Includem operatorul necesar pentru aplicarea modelului pe setul de date de testare, respectiv operatorul care calculează măsurile de performanță. Realizăm conexiunile și rulăm procesul. Pragul de probabilitate este setat la 0.3 (prima clasă este No, a doua Yes). Prin urmare, modelul va prezice că angajații în cazul cărora valoarea atributului „confidence(Yes)" va fi mai mare de 0.3 vor pleca, iar restul vor rămâne.





Rezultate (setul de testare):

Relativ la setul de testare, modelul produce predicțiile din imagine (am inclus primele 5 cazuri și alte trei de interes în contextul acestui exemplu – pentru comparație, am ales cazurile selectate în exemplul anterior, cel cu operatorul Neural Net). Coloana Attrition prezintă situația reală, iar coloana „prediction(Attrition)" predicția realizată de model. Observăm că unele predicții sunt greșite (în imagine, cele cu Id 1 și 1038). Probabilitățile asociate predicțiilor apar pe coloanele „confidence". Pentru fiecare caz este selectată ca predicție clasa care are cea mai mare probabilitate (confidence).

Row No.	Id	Attrition	prediction(Attrition)	confidence(No)	confidence(Yes)
1	1	Yes	No	0.924	0.076
2	2	No	No	0.997	0.003
3	8	No	No	0.923	0.077
4	11	No	No	0.987	0.013
5	12	No	No	0.808	0.192
...					
225	1038	Yes	No	0.999	0.001
220	1026	No	No	0.999	0.001
...					
168	746	No	No	0.862	0.138
41	167	Yes	Yes	0.064	0.936
...					
415	1944	Yes	Yes	0.021	0.979

...

Secțiunea PerformanceVector conține matricea de confuzie și diferite măsuri ale performanței clasificării (cele selectate), toate calculate relativ la ambele seturi de date (instruire și testare). Observăm că acuratețea predicției este 83% la instruire și 85% la testare (de obicei e puțin mai mare la instruire). Important e că modelul are șanse mari de generalizare. Comparativ cu modelul anterior (Neural Net) acuratețea clasificării este similară, ambele modele reușind să clasifice corect puțin peste jumătate dintre cazurile de angajați care pleacă (ceea ce e foarte util din punct de vedere practic). Însă, și acest model prezice relativ mai bine situația în care angajații nu pleacă (precizia = 91%, reamintirea = 92%) comparativ cu situația în care angajații pleacă (precizia = 54%, reamintirea = 52%).

accuracy: 82.90% +/- 3.67% (micro average: 82.90%)

	true No	true Yes	class precision
pred. No	759	72	91.34%
pred. Yes	104	94	47.47%
class recall	87.95%	56.63%	

instruire, prag 30%

accuracy: 85.26%

	true No	true Yes	class precision
pred. No	339	34	90.88%
pred. Yes	31	37	54.41%
class recall	91.62%	52.11%	

testare, prag 30%

Alte măsuri ale calității modelului

Operatorul Deep Learning oferă o serie de măsuri care ne ajută să evaluăm calitatea modelului, să comparăm modelul cu alte modele, respectiv să comparăm performanța clasificării în funcție de diferite praguri de probabilitate (care sunt grupurile de cazuri pe care le selectăm). Deoarece AUC este aproape maxim, R^2 este mare și logloss este mic, considerăm că avem un model foarte bun (în general). Să presupunem că păstrăm pentru intervenție (măsurile luate pentru a preveni plecarea angajaților) doar primele șapte grupuri[10] (primii 15% angajați cu probabilitățile de plecare cele mai mari), adică angajații pentru care am estimat că probabilitatea de plecare este de cel puțin 36%. Considerând doar acest grup de angajați observăm că măsurile de calitate ale modelului sunt foarte bune: modelul nostru, comparativ cu modelul de șansă (selecție aleatoare), are o șansă de 4.8 ori mai mare să identifice care sunt cazurile de angajați care pleacă (Cumulative Lift); 77.4% dintre angajații din acest grup chiar pleacă (Cumulative Response Rate); dacă selectăm acești angajați (primele șapte grupuri), identificăm 72.3% din numărul total al angajaților care pleacă (Cumulative Capture Rate).[11]

[10] Numărul grupurilor este 16, ele fiind definite astfel: cazurile sunt sortate în ordinea descrescătoare a probabilității estimate de a pleca; grupul 1 este format din primele 1% din cazuri, grupul 2 din următoarele 1% din cazuri, grupul 3 din următoarele 1%, grupul 4 din următoarele 1%, grupul 5 din următoarele 1%, grupul 6 din următoarele 5%, grupul 6 din următoarele 5%, grupul 7 din următoarele 5%, grupul 8 din următoarele 10%, grupul 9 din următoarele 10%, și tot așa până la grupul 16 în care intră ultimele 10% dintre cazuri (cutt points: 0.99, 0.98, 0.97, 0.96, 0.95, 0.9, 0.85, 0.8, 0.7, 0.6, 0.5, 0.4, 0.3, 0.2, 0.1, 0).

[11] Mai multe informații cu privire la măsurile calculate în acest tabel se pot afla din sub-capitolul 4.4 (Vizualizarea performanței (Visual): Gain & Lift charts), respectiv documentația H2O.

Tabelul final ne arată care sunt atributele cu cea mai mare capacitate predictivă. Însă, nu apar informații cu privire la sensul relației (dacă atributele respective reduc sau cresc șansele de plecare), nici despre șansele de plecare asociate unei anumite combinații de atribute. Pentru a obține astfel de informații e nevoie să folosim AutoModel sau să includem în proces și alți operatori (Explain Predictions și Model Simulator).

MSE: 0.065 R^2: 0.521 pr_auc: 0.821 mean_per_class_error: 0.157
RMSE: 0.254 AUC: 0.927 logloss: 0.229

Gains/Lift Table (Avg response rate: 16.13%, avg score: 15.69%):

Group	Cumulative Data Fraction	Lower Threshold	Lift	Cumulative Lift	Response Rate	Score	Cumulative Response	Cumulative Score	Capture Rate	Cumulative Capture Rate	Gain	Cumulative Gain
1	0.011	0.923	6.199	6.199	1.000	0.952	1.000	0.952	0.066	0.066	520	520
2	0.020	0.865	6.199	6.199	1.000	0.881	1.000	0.919	0.060	0.127	520	520
3	0.030	0.824	6.199	6.199	1.000	0.846	1.000	0.895	0.060	0.187	520	520
4	0.041	0.805	6.199	6.199	1.000	0.814	1.000	0.874	0.066	0.253	520	520
5	0.051	0.703	6.199	6.199	1.000	0.744	1.000	0.849	0.060	0.313	520	520
6	0.100	0.480	4.862	5.537	0.784	0.583	0.893	0.717	0.241	0.554	386	454
7	0.151	0.360	3.338	4.799	0.538	0.421	0.774	0.618	0.169	0.723	234	380
8	0.200	0.252	1.459	3.972	0.235	0.301	0.641	0.539	0.072	0.795	46	297
9	0.300	0.153	0.782	2.909	0.126	0.198	0.469	0.426	0.078	0.873	-22	191
10	0.400	0.091	0.301	2.257	0.049	0.118	0.364	0.349	0.030	0.904	-70	126
11	0.500	0.059	0.481	1.902	0.078	0.075	0.307	0.294	0.048	0.952	-52	90
12	0.600	0.033	0.243	1.628	0.039	0.045	0.263	0.253	0.024	0.976	-76	63
13	0.700	0.021	0.120	1.412	0.019	0.027	0.228	0.220	0.012	0.988	-88	41
14	0.800	0.012	0.000	1.235	0.000	0.016	0.199	0.195	0.000	0.988	-100	24
15	0.900	0.004	0.060	1.105	0.010	0.008	0.178	0.174	0.006	0.994	-94	10
16	1.000	0.000	0.060	1.000	0.010	0.002	0.161	0.157	0.006	1.000	-94	0

Variable	Relative Importance	Scaled Importance	Percentage
JobLevel = Middle management	1.000	1.000	0.018
OverTime = No	0.992	0.992	0.018
BusinessTravel = Non-Travel	0.989	0.989	0.018
RelationshipSatisfaction = Very high	0.984	0.984	0.018
Education = Below College	0.971	0.971	0.017
Gender = Female	0.965	0.965	0.017
Education = Bachelor	0.952	0.952	0.017
StockOptionLevel = None	0.948	0.948	0.017
StockOptionLevel = Low	0.939	0.939	0.017
JobRole = Laboratory Technician	0.921	0.921	0.016

10. MODELAREA AUTOMATĂ A DATELOR (AUTO MODEL)

Similar cu pregătirea automată a datelor pentru analiză (vezi capitolul despre Turbo Prep din primul volum), RapidMiner oferă posibilitatea automatizării și a procesului de modelare a datelor. Folosind tabul Auto Model, este suficient să indicăm doar setul de date și variabila de interes, iar softul estimează și face predicții pentru o serie de modele, apoi afișează rezultatele. Desigur, dacă dorim, putem interveni și selecta / deselecta o serie de opțiuni, funcție de datele analizate și preferințele noastre. În folderul asociat acestui volum am inclus rezultatele a două modelări automate folosind aceleași setări, cu diferența că o dată am analizat setul de date original (vezi rezultatele prezentate în acest capitol și folderul „0 AutoModel (unbal + cost)"), iar a doua oară am analizat același set de date cu clase echilibrate (folderul „0 AutoModel (bal smote + cost)"; analizele nu sunt prezentate aici).

10.1. Realizarea unei modelări automate, pas cu pas

Pașii necesari pentru realizarea unui proces de modelare automată a datelor sunt ilustrați în Figura 10.1-1.[1] După ce am selectat tabul Auto Model, ne apare o fereastră în care putem alege setul de date pe care dorim să-l modelăm (Select Data for a New Model) sau importăm (Import New Data) (Pasul 0). Tot aici, alternativ, putem alege unul dintre seturile de date utilizate anterior sau putem încărca rezultatele salvate ale unei modelări anterioare. De asemenea, putem indica folderul în care să fie salvate rezultatele unei modelări (Select Results Folder). La pasul 1 selectăm setul de date (employee_attrition), iar la pasul 2 alegem tipul de model (Predict), apoi Next.

La pasul 3 apare un grafic cu numărul de cazuri asociat fiecărei categorii aferente variabilei de interes (label), putem alege clasa de interes, asocia alte valori valorilor originale, respectiv putem defini matricea cost. În acest caz am ales să atribuim un

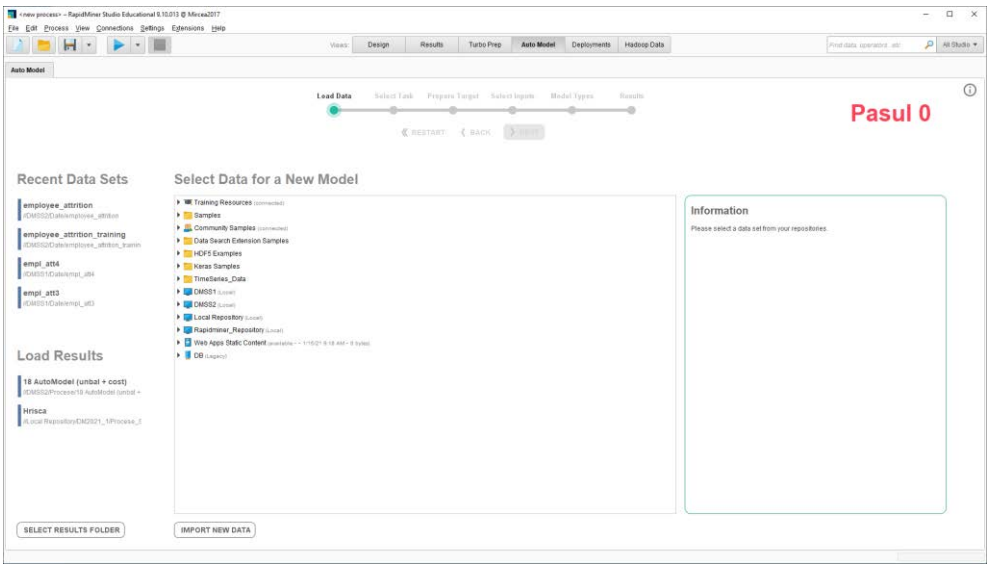
[1] O prezentare video relativ mai extinsă a Auto Model (o versiune anterioară, ceva mai simplă), se poate vedea aici: Auto Model - Classification, iar o prezentare pentru versiunea 9.10 aici: Cost-Sensitive Scoring with RapidMiner Auto Model - Classification Evaluation.

cost de trei ori mai mare erorilor de clasificare de tip II (FN) adică situației în care angajatul părăsește compania, predicția modelului fiind că rămâne.

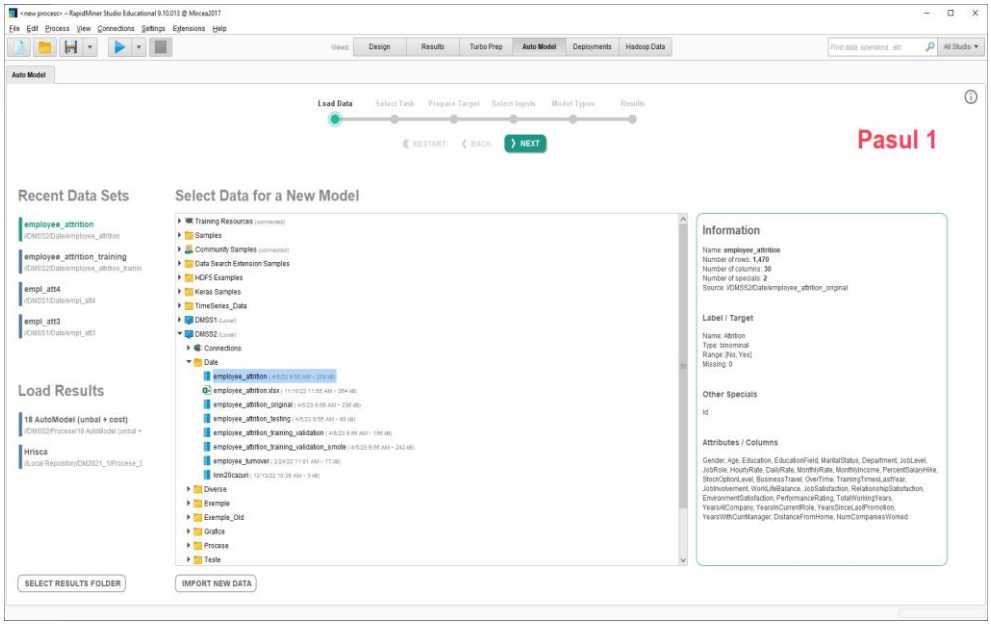
La pasul 4 softul permite selectarea / deselectarea atributelor utilizate ca predictori. Atributele marcate cu roșu, cele inutile pentru predicție, sunt deselectate automat. Aici le-am deselectat și pe cele portocalii. Utilitatea unui atribut este evaluată în funcție de cinci criterii și anume: Correlation, Id-ness, Stability, Missing și Text-ness. Cu excepția corelației (și aici, nu vrem o corelație perfectă, deci valori maxime), pentru toate celelalte criterii ne dorim valori mici (pentru mai multe detalii se poate consulta primul volum).

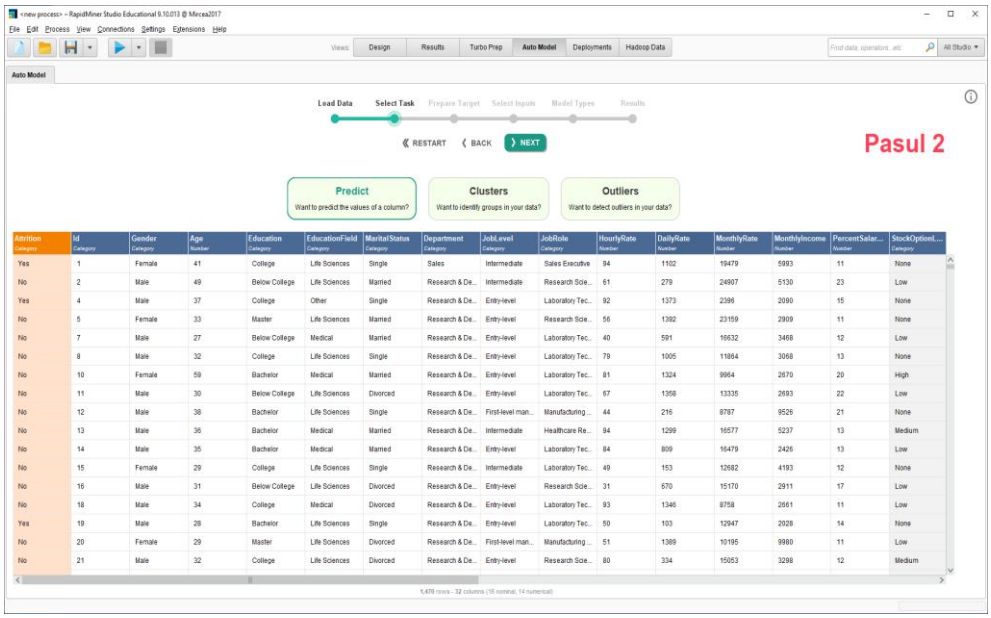
La pasul 5 putem selecta unul sau mai multe modele de clasificare, respectiv indica dacă dorim să optimizăm automat sau nu fiecare model. Sunt disponibile și alte două categorii de opțiuni: pregătirea datelor și analiza atributelor. La pregătirea datelor putem alege una sau mai multe opțiuni dintre următoarele: eliminarea atributelor cu prea multe valori, extragerea unor informații din atributele de tip dată sau text, selectarea automată a atributelor și generarea automată a unor atribute. Aici am lăsat doar prima opțiune (generarea și selectarea atributelor durează foarte mult).[2] La analiza atributelor putem alege să calculăm matricea de corelații pentru toate atributele selectate, să evaluăm importanța acestora pentru predicție, respectiv să explicăm predicțiile.

Figura 10.1-1. Auto Model: Modelarea automată pas cu pas

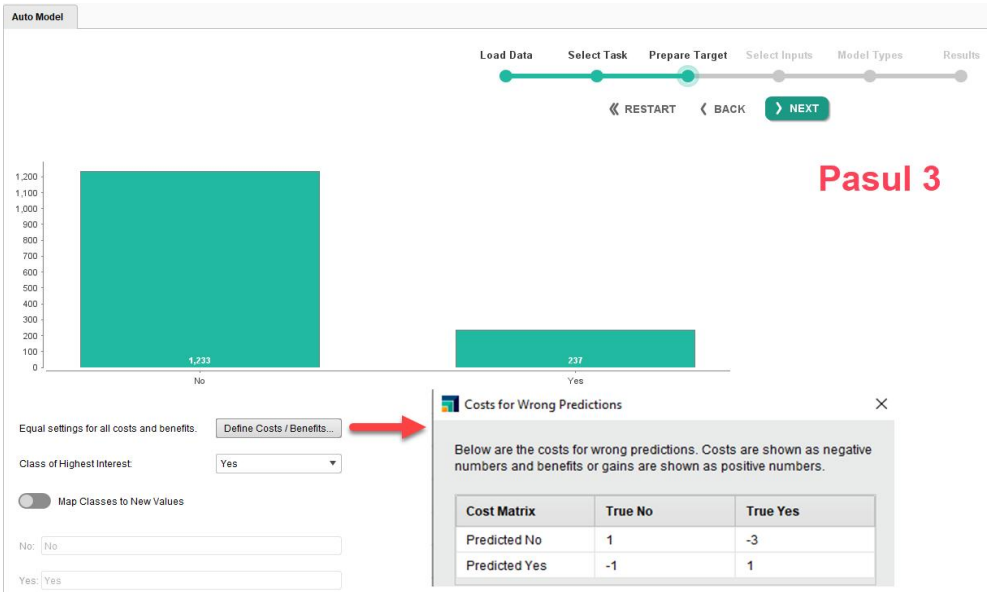


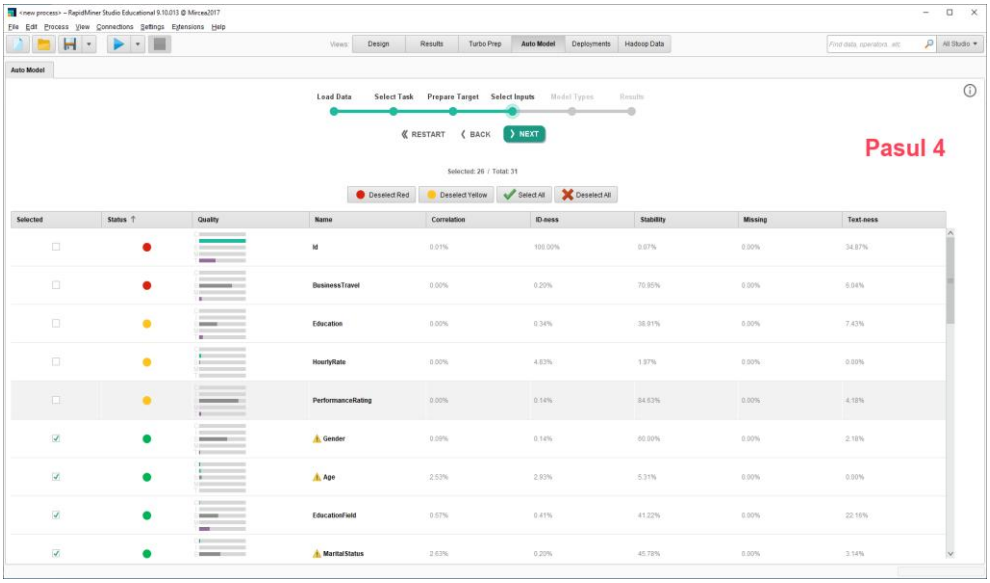
[2] O prezentare video extinsă a acestor funcții poate fi găsită aici: Automatic Feature Engineering with RapidMiner Auto Model.

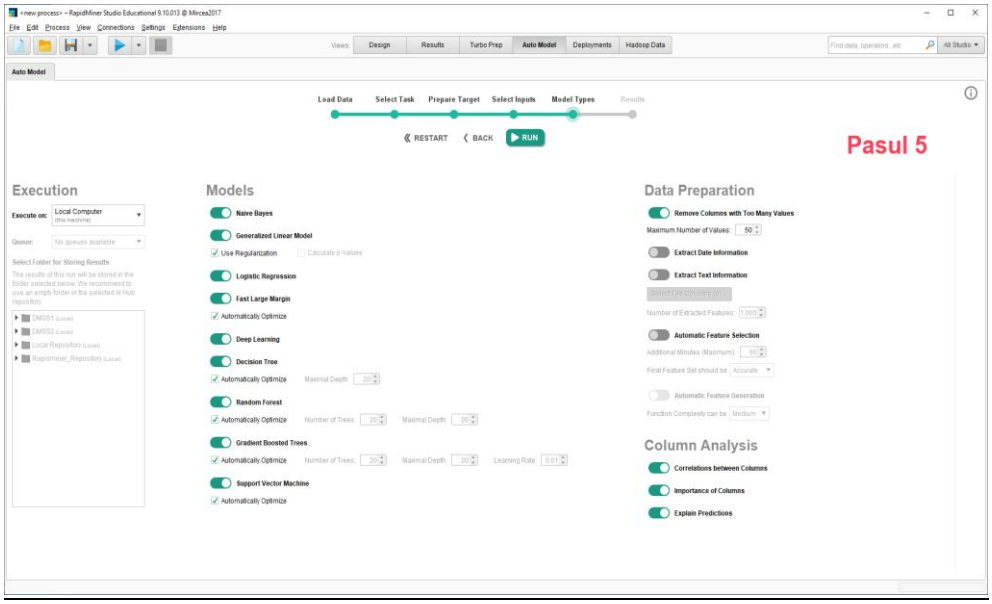




Attrition	Id	Gender	Age	Education	EducationField	MaritalStatus	Department	JobLevel	JobRole	HourlyRate	DailyRate	MonthlyRate	MonthlyIncome	PercentSalar...	StockOption...
Yes	1	Female	41	College	Life Sciences	Single	Sales	Intermediate	Sales Executive	94	1102	19479	5993	11	None
No	2	Male	49	Below College	Life Sciences	Married	Research & De...	Intermediate	Research Scie...	61	279	24907	5130	23	Low
Yes	4	Male	37	College	Other	Single	Research & De...	Entry-level	Laboratory Tec...	92	1373	2396	2090	15	None
No	5	Female	33	Master	Life Sciences	Married	Research & De...	Entry-level	Research Scie...	56	1392	23159	2909	11	None
No	7	Male	27	Below College	Medical	Married	Research & De...	Entry-level	Laboratory Tec...	40	591	16632	3468	12	Low
No	8	Male	32	College	Life Sciences	Single	Research & De...	Entry-level	Laboratory Tec...	79	1005	11864	3068	13	None
No	10	Female	59	Bachelor	Medical	Married	Research & De...	Entry-level	Laboratory Tec...	81	1324	9964	2670	20	High
No	11	Male	30	Below College	Life Sciences	Divorced	Research & De...	Entry-level	Laboratory Tec...	67	1358	13335	2693	22	Low
No	12	Male	38	Bachelor	Life Sciences	Single	Research & De...	First-level man...	Manufacturing ...	44	216	8787	9526	21	None
No	13	Male	36	Bachelor	Medical	Married	Research & De...	Intermediate	Healthcare Re...	94	1299	16577	5237	13	Medium
No	14	Male	35	Bachelor	Medical	Married	Research & De...	Entry-level	Laboratory Tec...	84	809	16479	2426	13	Low
No	15	Female	29	College	Life Sciences	Single	Research & De...	Intermediate	Laboratory Tec...	49	153	12682	4193	12	None
No	16	Male	31	Below College	Life Sciences	Divorced	Research & De...	Entry-level	Research Scie...	31	670	15170	2911	17	Low
No	18	Male	34	College	Medical	Divorced	Research & De...	Entry-level	Laboratory Tec...	93	1346	8758	2661	11	Low
Yes	19	Male	28	Bachelor	Life Sciences	Single	Research & De...	Entry-level	Laboratory Tec...	50	103	12947	2028	14	None
No	20	Female	29	Master	Life Sciences	Divorced	Research & De...	First-level man...	Manufacturing ...	51	1389	10195	9980	11	Low
No	21	Male	32	College	Life Sciences	Divorced	Research & De...	Entry-level	Research Scie...	80	334	15053	3298	12	Medium


Auto Model
Load Data
Select Task
Prepare Target
Select Inputs
Model Types
Results
RESTART
BACK
NEXT
Pasul 3
1,200
1,100
1,000
900
800
700
600
500
400
300
200
100
0
1,233
237
No
Yes
Equal settings for all costs and benefits.
Define Costs / Benefits...
Class of Highest Interest:
Yes
Map Classes to New Values
No:
No
Yes:
Yes
Costs for Wrong Predictions
Below are the costs for wrong predictions. Costs are shown as negative numbers and benefits or gains are shown as positive numbers.
Cost Matrix
True No
True Yes
Predicted No
1
-3
Predicted Yes
-1
1


Auto Model
Load Data
Select Task
Prepare Target
Select Inputs
Model Types
Results
RESTART
BACK
NEXT
Pasul 4
Deselect Red
Deselect Yellow
Select All
Deselect All
Selected
Status
Quality
Name
Correlation
ID-ness
Stability
Missing
Text-ness

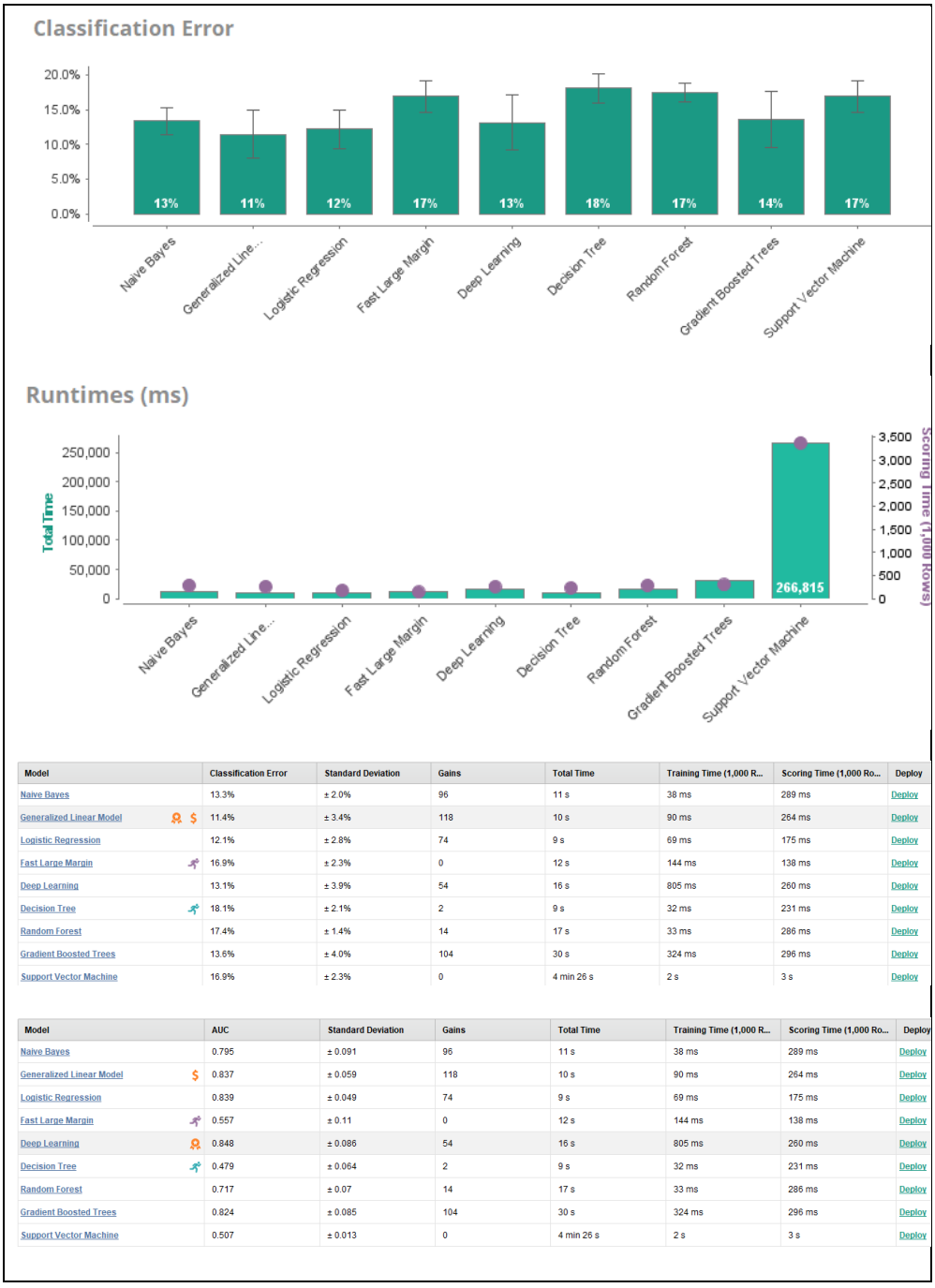


10.2. Rezultatele obținute și utilitatea acestora

Output-ul produs de Auto Model este organizat în trei secțiuni principale: (1) Compararea modelelor (Figura 10.2-1), (2) Rezultatele asociate fiecărui model (Figura 10.2-2) și (3) Analize generale (nu le mai prezentăm aici, dar pot fi vizualizate în procesul asociat acestui capitol).

Pentru fiecare dintre modelele construite, la secțiunea comparativă (comparison - overview) sunt prezentate grafic și tabelar performanța și timpul de rulare. Performanța comparativă poate fi afișată în funcție de o serie de măsuri: eroarea de clasificare, acuratețea, AUC, F, precizia, reamintirea, senzitivitatea și specificitatea. Tot aici este inclus și un grafic comparativ al curbelor ROC (ROC Comparison). Pentru fiecare dintre măsurile de performanță este prezentată și abaterea standard asociată (e utilă pentru a evalua stabilitatea modelului). Timpul de rulare este prezentat ca total, dar și diferențiat ca timp necesar pentru instruirea modelului, respectiv pentru calcularea predicțiilor. Coloana Gains din tabel indică câștigurile obținute suplimentar ca urmare a utilizării modelului comparativ cu utilizarea celei mai bune opțiuni dintre cele existente (în acest caz, pleacă / nu pleacă). Modelul cel mai bun, în funcție de fiecare dintre criteriile performanță, câștig, timpul de calculare a scorurilor și timpul total de rulare, este indicat printr-un semn grafic sugestiv.

Figura 10.2-1. Auto Model: Performanța comparativă a modelelor (Comparison - Overview)



Model	Classification Error	Standard Deviation	Gains	Total Time	Training Time (1,000 R...	Scoring Time (1,000 Ro...	Deploy
Naive Bayes	13.3%	± 2.0%	96	11 s	38 ms	289 ms	Deploy
Generalized Linear Model	11.4%	± 3.4%	118	10 s	90 ms	264 ms	Deploy
Logistic Regression	12.1%	± 2.8%	74	9 s	69 ms	175 ms	Deploy
Fast Large Margin	16.9%	± 2.3%	0	12 s	144 ms	138 ms	Deploy
Deep Learning	13.1%	± 3.9%	54	16 s	805 ms	260 ms	Deploy
Decision Tree	18.1%	± 2.1%	2	9 s	32 ms	231 ms	Deploy
Random Forest	17.4%	± 1.4%	14	17 s	33 ms	286 ms	Deploy
Gradient Boosted Trees	13.6%	± 4.0%	104	30 s	324 ms	296 ms	Deploy
Support Vector Machine	16.9%	± 2.3%	0	4 min 26 s	2 s	3 s	Deploy

Model	AUC	Standard Deviation	Gains	Total Time	Training Time (1,000 R...	Scoring Time (1,000 Ro...	Deploy
Naive Bayes	0.795	± 0.091	96	11 s	38 ms	289 ms	Deploy
Generalized Linear Model	0.837	± 0.059	118	10 s	90 ms	264 ms	Deploy
Logistic Regression	0.839	± 0.049	74	9 s	69 ms	175 ms	Deploy
Fast Large Margin	0.557	± 0.11	0	12 s	144 ms	138 ms	Deploy
Deep Learning	0.848	± 0.086	54	16 s	805 ms	260 ms	Deploy
Decision Tree	0.479	± 0.064	2	9 s	32 ms	231 ms	Deploy
Random Forest	0.717	± 0.07	14	17 s	33 ms	286 ms	Deploy
Gradient Boosted Trees	0.824	± 0.085	104	30 s	324 ms	296 ms	Deploy
Support Vector Machine	0.507	± 0.013	0	4 min 26 s	2 s	3 s	Deploy

Model	F Measure	Standard Deviation	Gains	Total Time	Training Time (1,000 R...	Scoring Time (1,000 Ro...	Deploy
Naive Bayes	52.7%	± 3.7%	96	11 s	38 ms	289 ms	Deploy
Generalized Linear Model	60.1%	± 10.8%	118	10 s	90 ms	264 ms	Deploy
Logistic Regression	44.5%	± 7.2%	74	9 s	69 ms	175 ms	Deploy
Fast Large Margin	?	?	0	12 s	144 ms	138 ms	Deploy
Deep Learning	35.8%	± 9.4%	54	16 s	805 ms	260 ms	Deploy
Decision Tree	?	?	2	9 s	32 ms	231 ms	Deploy
Random Forest	19.5%	± 3.6%	14	17 s	33 ms	286 ms	Deploy
Gradient Boosted Trees	57.5%	± 9.6%	104	30 s	324 ms	296 ms	Deploy
Support Vector Machine	?	?	0	4 min 26 s	2 s	3 s	Deploy

Pentru fiecare model, RapidMiner afișează o serie de rezultate:

- **Model (modelul):** permite vizualizarea relației dintre fiecare predictor și atributul prezis; în imagine observăm că angajații care pleacă stau în medie, firesc, mai puțin timp sub conducerea managerului;
- **Weights (ponderări):** prezintă importanța asociată fiecărui predictor (cât de important este acesta pentru a crește probabilitatea de a face partea din clasa de interes – pleacă, în acest caz); primii trei predictori sunt numărul de ani petrecuți sub conducerea managerului, numărul de ani petrecuți în poziția din prezent și munca peste program;
- **Simulation (simulare):** ne ajută să calculăm probabilitatea clasei de interes în funcție de valorile luate de predictori (putem alege orice combinație de valori); desigur, probabilitățile pot fi comparate între ele, adică putem vedea care e impactul modificării valorii asociate unui predictor asupra probabilității; de exemplu, probabilitatea ca o persoană căsătorită să plece este 15%; probabilitatea crește la 17% în cazul persoanelor singure (celelalte condiții fiind aceleași; valorile atributelor sunt cele specificate în imagine / folderul inclus);
- **Performance (performanța):** prezintă tabelar valorile măsurilor de performanță (alături apare incertitudinea asociată – abaterea standard) și matricea de confuzie; tot aici apar valorile estimate cu privire la profitul obținut ca urmare a aplicării modelului, profitul obținut în cazul în care nu se folosește un model (modelul naiv sau cea mai bună alegere în situația în care nu folosim un model), respectiv câștigul așteptat (diferența dintre cele două profituri); de exemplu, în cazul de față (Naive Bayes), profitul asociat modelului este 224, profitul asociat modelului naiv este 128 (prezicem că nu pleacă niciun angajat), iar câștigul obținut este 96 (224-128); aceste valori sunt calculate folosind datele din cele două matrici (confuzie și cost) astfel:

Matricea de confuzie	Realitate	
Predicție	Nu pleacă	Pleacă
Nu pleacă	333	42
Pleacă	14	31

Matricea cost	Realitate	
Predicție	Nu pleacă	Pleacă
Nu pleacă	1	-3
Pleacă	-1	1

Număr angajați care rămân în realitate: 347 (333 + 14)
Număr angajați care pleacă în realitate: 73 (42+31)
Cost total dacă toți rămân: 128 (347 * 1 + 73 * (-3))
Cost total dacă toți pleacă: -274 (347 * (-1) + 73 * 1)
Cea mai bună alegere fără model: Rămân (deoarece 128 > -274)
Cost atunci când folosim modelul: 224 (333*1 + 42*(-3) + 14*(-1) + 31 * 1)
Câștig (Gain): 94 (224-128)

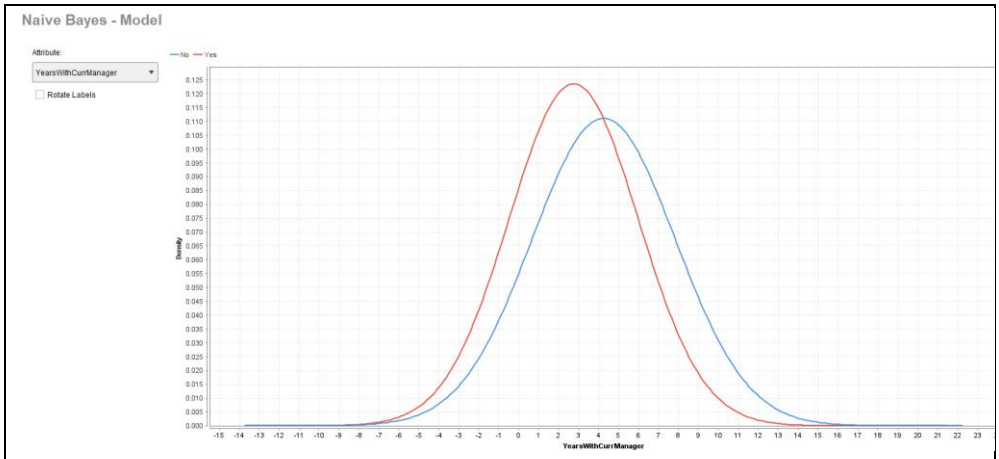
- **Lift Chart:** prezintă graficul Lift (vezi sub-capitolul 4.4 din acest manual); în esență, dacă de raportăm la primii 20% angajați cu probabilitatea cea mai mare de a pleca, folosind modelul, vom identifica 63% din totalul angajaților care părăsesc compania;
- **Predictions (predicții):** prezintă predicțiile la nivel de caz, mai exact probabilitatea ca acel caz să aparțină fiecărei clase și costul asociat cazului (un exemplu de calcul a costului este prezentat mai jos – valorile sunt cele asociate primului caz); valorile asociate predictorilor apar pe un fundal colorat, culorile folosite variind pe o scală continuă, în funcție de gradul în care valoarea respectivă susține sau contrazice predicția clasei de interes (pleacă);

Realitate	Predicție	Confidence No	Confidence Yes	Cost
Yes (Pleacă)	No (Rămâne)	0.722	0.278	-0.113

Cost = Conf.NO * CostNONO + Conf.YES * CostYESNO = 0.722 * 1 + 0.278 * (-3) = -0.113

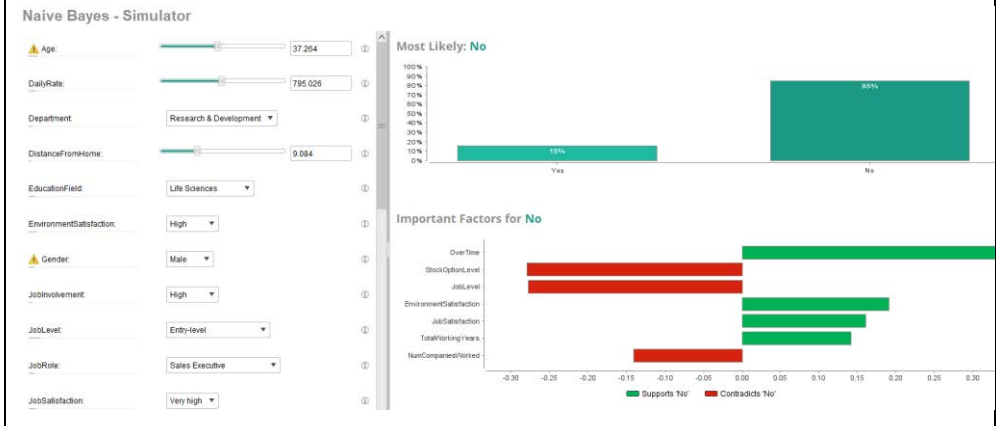
- **Production Model (modelul pentru producție):** permite vizualizarea relației dintre fiecare predictor și atributul prezis în cazul modelului propus a intra în producție; spre deosebire de primul pas (Model), de această dată modelul este calculat folosind tot setul de date (instruire și validare); în imagine observăm că angajații care pleacă stau, în medie, firesc, mai puțin timp sub conducerea managerului.

Figura 10.2-2. Auto Model: Rezultatele asociate modelului Naïve Bayes



Naive Bayes - Weights

Attribute	Weight
YearsWithCurrManager	0.215
YearsInCurrentRole	0.166
OverTime	0.152
YearsAtCompany	0.144
TotalWorkingYears	0.135
MonthlyIncome	0.125
MonthlyRate	0.116
JobLevel	0.106
StockOptionLevel	0.097
Age	0.077
EnvironmentSatisfaction	0.075
MaritalStatus	0.073
PercentSalaryHike	0.070
JobSatisfaction	0.065



Naive Bayes - Performance

Profits

Profits from Model: 224 Profits for Best Option (No): 128 Gain: 96 Show Costs / Benefits...

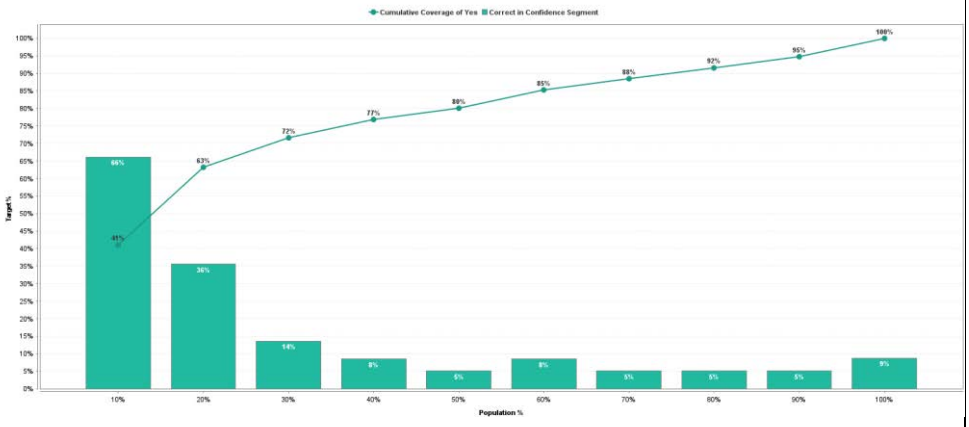
Performances

Criterion	Value	Standard Deviation
Accuracy	86.7%	± 2.0%
Classification Error	13.3%	± 2.0%
AUC	79.5%	± 9.1%
Precision	70.2%	± 10.3%
Recall	43.4%	± 7.7%
F Measure	52.7%	± 3.7%
Sensitivity	43.4%	± 7.7%
Specificity	96.0%	± 1.8%

Confusion Matrix

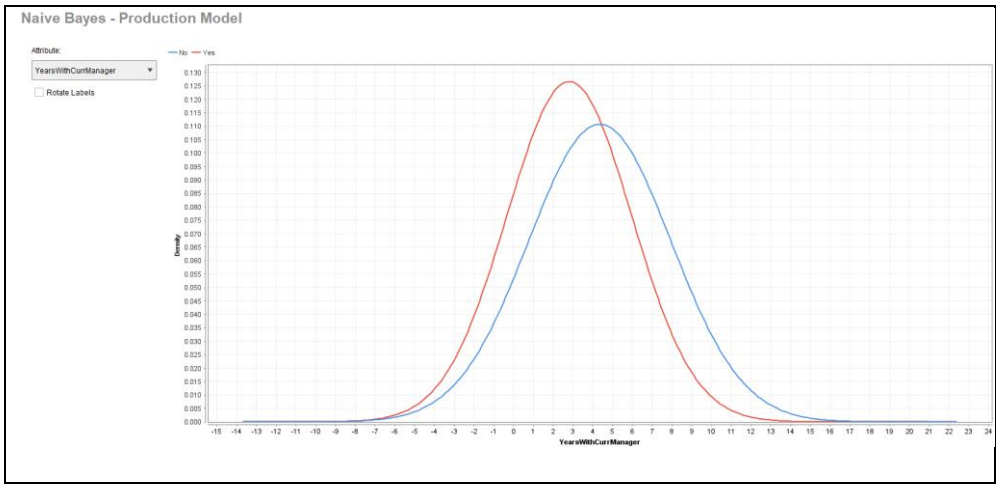
	true No	true Yes	class precision
pred. No	333	42	88.80%
pred. Yes	14	31	68.89%
class recall	95.97%	42.47%	

Naive Bayes - Lift Chart



Naive Bayes - Predictions

Row No.	Attrition	prediction(A...	confidence(...	confidence(...	cost	Gender	EducationFi...	MaritalStatus	Department	JobLevel	JobRole
1	Yes	No	0.722	0.278	-0.113	Female	Life Sciences	Single	Sales	Intermediate	Sales Executi...
2	No	No	0.865	0.135	0.460	Male	Life Sciences	Married	Research & ...	Intermediate	Research Sci...
3	Yes	Yes	0.629	0.371	-0.259	Male	Other	Single	Research & ...	Entry-level	Laboratory Te...
4	No	No	0.805	0.195	0.221	Female	Life Sciences	Married	Research & ...	Entry-level	Research Sci...
5	No	No	0.826	0.174	0.305	Male	Life Sciences	Single	Research & ...	Entry-level	Laboratory Te...
6	No	No	0.739	0.261	-0.045	Male	Life Sciences	Divorced	Research & ...	Entry-level	Laboratory Te...
7	No	No	0.858	0.142	0.433	Male	Life Sciences	Single	Research & ...	First-level ma...	Manufacturin...
8	No	No	0.857	0.143	0.429	Male	Medical	Married	Research & ...	Intermediate	Healthcare R...
9	No	No	0.762	0.238	0.048	Male	Life Sciences	Divorced	Research & ...	Entry-level	Research Sci...
10	No	No	0.865	0.135	0.460	Female	Life Sciences	Divorced	Research & ...	First-level ma...	Manufacturin...


Naive Bayes - Production Model
Attribute:
YearsWithCurrManager
Rotate Labels
No
Yes
Density
YearsWithCurrManager

BIBLIOGRAFIE

An, S.-H. (2019). Employee Voluntary and Involuntary Turnover and Organizational Performance: Revisiting the Hypothesis from Classical Public Administration. *International Public Management Journal*, *22*(3), 444–469. https://doi.org/10.1080/10967494.2018.1549629

Attewell, P., & Monaghan, D. (2015). *Data Mining for the Social Sciences: An Introduction*. University of California Press.

Belkin, M., Hsu, D., Ma, S., & Mandal, S. (2019). Reconciling modern machine-learning practice and the classical bias–variance trade-off. *Proceedings of the National Academy of Sciences of the United States of America*, *116*(32), 15849–15854. https://doi.org/10.1073/PNAS.1903070116

Bubeck, S., & Sellke, M. (2021). A Universal Law of Robustness via Isoperimetry. *Advances in Neural Information Processing Systems*, *34*, 28811–28822. https://arxiv.org/abs/2105.12806v4

Canbek, G., Temizel, T. T., & Sagiroglu, S. (2021). BenchMetrics: a systematic benchmarking method for binary classification performance metrics. *Neural Computing and Applications*. https://doi.org/10.1007/S00521-021-06103-6

Chen, W., Miao, C., Zhang, Z., Fung, C. S. H., Wang, R., Chen, Y., Qian, Y., Cheng, L., Yip, K. Y., Tsui, S. K. W., & Cao, Q. (2024). Commonly used software tools produce conflicting and overly-optimistic AUPRC values. *Genome Biology*, *25*(1), 1–12. https://doi.org/10.1186/S13059-024-03266-Y

Chicco, D., & Jurman, G. (2020). The advantages of the Matthews correlation coefficient (MCC) over F1 score and accuracy in binary classification evaluation. *BMC Genomics*. https://doi.org/10.1186/S12864-019-6413-7

Chicco, D., & Jurman, G. (2022). An Invitation to Greater Use of Matthews Correlation Coefficient in Robotics and Artificial Intelligence. *Frontiers in Robotics and AI*, *9*. https://www.frontiersin.org/articles/10.3389/frobt.2022.876814

Chicco, D., Tötsch, N., & Jurman, G. (2021). The Matthews correlation coefficient (MCC) is more reliable than balanced accuracy, bookmaker informedness, and markedness in two-class confusion matrix evaluation. *BioData Mining*, *14*(1), 13. https://doi.org/10.1186/s13040-021-00244-z

Chicco, D., Warrens, M. J., & Jurman, G. (2021). The Matthews Correlation Coefficient (MCC) is More Informative Than Cohen's Kappa and Brier Score in Binary Classification Assessment. *IEEE Access*. https://doi.org/10.1109/ACCESS.2021.3084050

Comșa, M. (2022). *Data mining pentru științele sociale. Volumul 1: Pregătirea datelor în RapidMiner Studio*. Presa Universitară Clujeană.

Culic, I. (2004). *Metode avansate în cercetarea socială. Analiza multivariată de interdependență*. Polirom.

David, A. (2008). Retaining Talent: A Guide to Analyzing and Managing Employee Turnover. *SHRM Foundation*, 3.

Davis, J., & Goadrich, M. (2006). The Relationship Between Precision-Recall and ROC Curves. *ACM International Conference Proceeding Series*, *23*, 233–240. https://doi.org/10.1145/1143844.1143874

De Diego, I. M., Redondo, A. R., Fernández, R. R., Navarro, J., & Moguerza, J. M. (2022). General Performance Score for classification problems. *Applied Intelligence*, *52*(10), 12049–12063. https://doi.org/10.1007/s10489-021-03041-7

Delgado, R., & Tibau, X.-A. (2019). Why Cohen's Kappa should be avoided as performance measure in classification. *PLOS ONE*, *14*(9). https://doi.org/10.1371/journal.pone.0222916

Domingos, P. (2012). A few useful things to know about machine learning. *Communications of the ACM*, *55*(10), 78–87. https://doi.org/10.1145/2347736.2347755

Domingos, P. (2015). *The master algorithm: how the quest for the ultimate learning machine will remake our world*. Basic books.

Edwards, H. (2016). *How Machines Learn: An Illustrated Guide to Machine Learning*. Koru Ventures, LLC.

Etz, A. (2018). Introduction to the Concept of Likelihood and Its Applications. *Advances in Methods and Practices in Psychological Science*, *1*(1), 60–69. https://doi.org/10.1177/2515245917744314

Field, A. P. (2018). *Discovering statistics using IBM SPSS statistics* (5th ed.). Sage Publications.

Foster, I., Rayid, G., Jarmin, S. R., Kreuter, F., & Lane, J. (2021). *Big data and social science: Data science methods and tools for research and practice* (2nd ed.). CRC Press.

Fu, C., & Yang, J. (2021). Granular classification for imbalanced datasets: A minkowski distance-based method. *Algorithms*, *14*(2). https://doi.org/10.3390/a14020054

Garson, G. D. (2014). *Logistic Regression: Binary and Multinomial*. Statistical Associates Publishers.

Garson, G. D. (2021). *Data Analytics for the Social Sciences: Applications in R*. Routledge. https://doi.org/10.4324/9781003109396

Gösgens, M., Zhiyanov, A., Tikhonov, A., & Prokhorenkova, L. (2022). Good Classification Measures and How to Find Them. *ArXiv*. https://doi.org/10.48550/ARXIV.2201.09044

Hastie, T., Tibshirani, R., & Friedman, J. (2018). *The elements of statistical learning: data mining, inference, and prediction* (2nd ed.). Springer.

Hilbe, J. M. (2016). *Practical Guide to Logistic Regression*. Chapman and Hall/CRC. https://doi.org/10.1201/b18678

Hosmer, D. W., Lemeshow, Stanley., & Sturdivant, R. X. (2013). *Applied logistic regression*. Wiley.

Jacobucci, R., Grimm, J. K., & Zhang, Z. (2023). *Machine Learning for Social and Behavioral Research*. Guilford Press.

James, G., Witten, D., Hastie, T., & Tibshirani, R. (2021). *An Introduction to Statistical Learning: with Applications in R* (2nd ed.). Springer.

Joshua Starmer. (2022). *The StatQuest Illustrated Guide to Machine Learning!!!* StatQuest Publications.

Kotu, V., & Deshpande, B. (2019). *Data science: concepts and practice* (2nd ed.). Morgan Kaufmann.

Long, J. S., & Freese, J. (2014). *Regression models for categorical dependent variables using Stata* (College Station, Ed.; 3rd ed.). Stata Press.

Lovell, D., Miller, D., Capra, J., & Bradley, A. (2022). Never mind the metrics - what about the uncertainty? Visualising confusion matrix metric distributions. *ArXiv*. https://doi.org/10.48550/ARXIV.2206.02157

Luque, A., Carrasco, A., Martín, A., & de las Heras, A. (2019). The impact of class imbalance in classification performance metrics based on the binary confusion matrix. *Pattern Recognition*, *91*, 216–231. https://doi.org/https://doi.org/10.1016/j.patcog.2019.02.023

Moreira, J., Carvalho, A., & Horvath, T. (2019). A General Introduction to Data Analytics. In *A General Introduction to Data Analytics*. Wiley. https://doi.org/10.1002/9781119296294

Neal, B., Mittal, S., Baratin, A., Tantia, V., Scicluna, M., Lacoste-Julien, S., & Mitliagkas, I. (2019). A Modern Take on the Bias-Variance Tradeoff in Neural Networks. *ArXiv*. https://arxiv.org/abs/1810.08591

Nisbet, R., Miner, G. D., & Yale, K. (2018). *Handbook of Statistical Analysis and Data Mining Applications*. Academic Press.

Norton, E. C., & Dowd, B. E. (2018). Log Odds and the Interpretation of Logit Models. *Health Services Research*, *53*(2). https://doi.org/10.1111/1475-6773.12712

Pampel, F. C. (2020). *Logistic Regression: A Primer*. SAGE Publications, Inc. https://doi.org/10.4135/9781071878729

RapidMiner. (2017). *How to Correctly Validate Machine Learning Models Basics of Predictive Models and Validation*.

RapidMiner. (2022). *RapidMiner 9. Operator Reference Manual*. RapidMiner GmbH.

Raza, A., Munir, K., Almutairi, M., Younas, F., & Fareed, M. M. S. (2022). Predicting Employee Attrition Using Machine Learning Approaches. *Applied Sciences*, *12*(13). https://doi.org/10.3390/app12136424

Robette, N. (2022). Trees and forest. Recursive partitioning as an alternative to parametric regression models in social sciences. *Bulletin of Sociological Methodology/Bulletin de Méthodologie Sociologique*, *154*(1), 7–56. https://doi.org/10.1177/07591063221128325

Roiger, R. J. (2017). *Data mining: a tutorial-based primer* (2nd ed.). Chapman and Hall/CRC.

Saito, T., & Rehmsmeier, M. (2015). The precision-recall plot is more informative than the ROC plot when evaluating binary classifiers on imbalanced datasets. *PLoS ONE.*, *10*(3). https://doi.org/10.1371/journal.pone.0118432

Sava, F. A. (2011). *Analiza datelor în cercetarea psihologică*. ASCR.

Shmueli, G., Bruce, P. C., Deokar, A. V., & Patel, N. R. (2023). *Machine Learning for Business Analytics: Concepts, Techniques and Applications in RapidMiner*. Wiley.

Shmueli, G., Bruce, P. C., Yahav, I., Patel, N. R., & Lichtendahl Jr, K. C. (2017). *Data Mining for Business Analytics: Concepts, Techniques, and Applications in R*. Wiley.

Tan, P.-N., Steinbach, M., Karpatne, A., & Kumar, V. (2019). *Introduction to data mining* (2nd ed.). Pearson.

Uddin, S., Haque, I., Lu, H., Moni, M. A., & Gide, E. (2022). Comparative performance analysis of K-nearest neighbour (KNN) algorithm and its different variants for disease prediction. *Scientific Reports*, *12*(1), 1–11. https://doi.org/10.1038/s41598-022-10358-x

Vanacore, A., Pellegrino, M. S., & Ciardiello, A. (2022). Fair evaluation of classifier predictive performance based on binary confusion matrix. *Computational Statistics*. https://doi.org/10.1007/S00180-022-01301-9

Vanacore, A., Pellegrino, M. S., & Ciardiello, A. (2023). Evaluating classifier predictive performance in multi-class problems with balanced and imbalanced data sets. *Quality and Reliability Engineering International*. https://doi.org/10.1002/QRE.3237

Wendler, T., & Gröttrup, S. (2021). *Data mining with SPSS modeler: Theory, exercises and solutions* (2nd ed.). Springer. https://doi.org/10.1007/978-3-319-28709-6

Williams, R. (2012). Using the margins command to estimate and interpret adjusted predictions and marginal effects. *Stata Journal*, *12*(2), 308–331. https://doi.org/10.1177/1536867x1201200209

Williams, R., & Jorgensen, A. (2023). Comparing logit & probit coefficients between nested models. *Social Science Research*, *109*. https://doi.org/10.1016/j.ssresearch.2022.102802

Zhu, Q. (2020). On the performance of Matthews correlation coefficient (MCC) for imbalanced dataset. *Pattern Recognition Letters*, *136*, 71–80. https://doi.org/10.1016/J.PATREC.2020.03.030

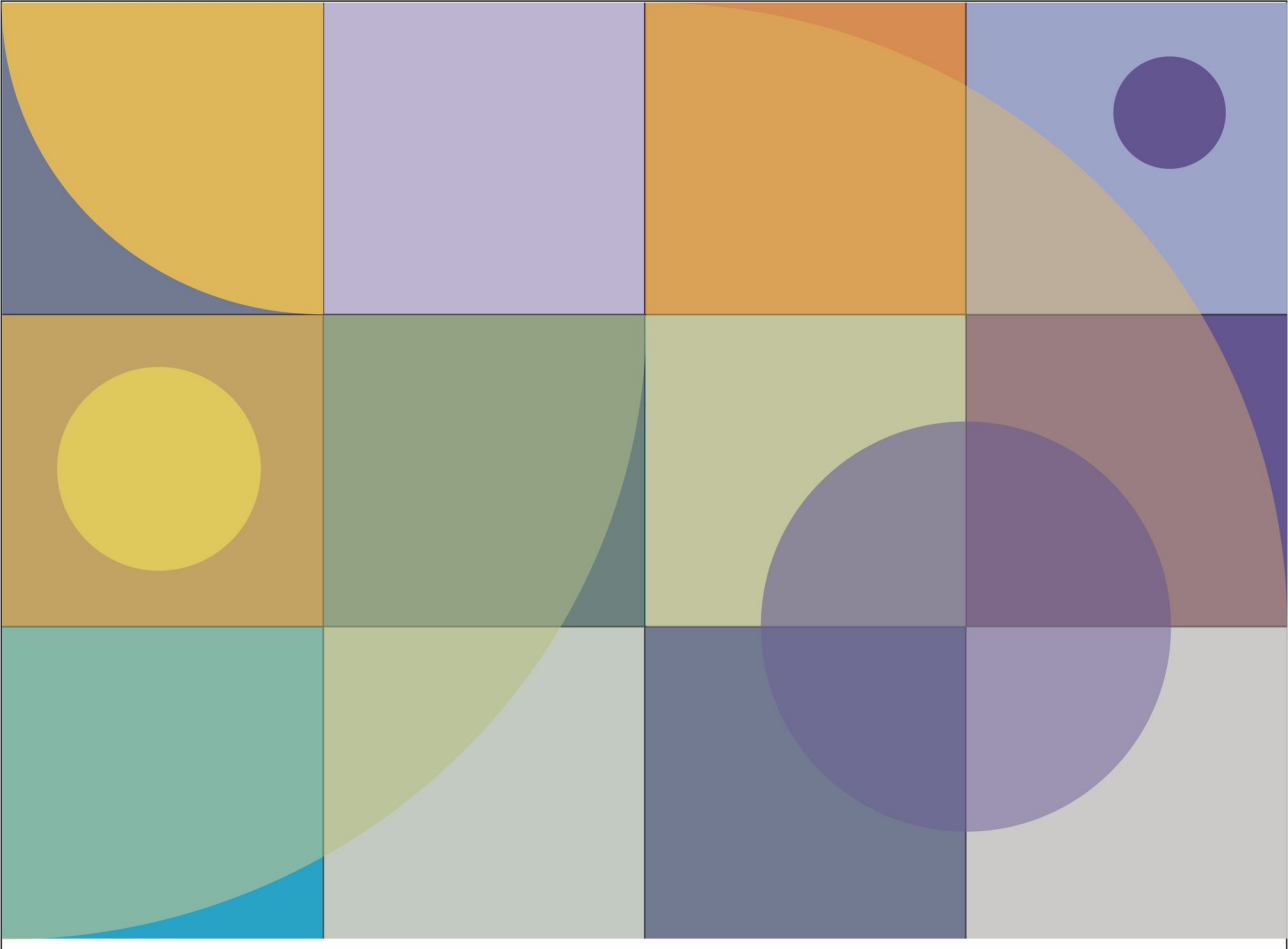
ANEXE

Termeni importanți

EN	RO	Definire
training set (learning)	instruire ~ antrenare ~ învățare	Setul de date utilizat pentru instruirea unui model (estimarea coeficienților modelului). Acest set de date conține valorile aferente atributului dependent și atributelor independente.
validation set	validare	Setul de date utilizat pentru validarea unui model. Poate lua forma unui set de date cu alte cazuri decât cele incluse în setul de instruire sau forma mai multor sub-seturi (de obicei 5/10) ale setului de instruire, deci loturi de cazuri selectate dintre cazurile incluse în setul de date de instruire (validarea încrucișată). Acest set de date conține valorile aferente atributului dependent și atributelor independente.
testing set	testare	Setul de date utilizat pentru testarea unui model. Aceste date nu au fost „văzute" anterior de model, adică nu au fost folosite pentru instruirea și validarea modelului. Acest set de date conține doar valorile aferente atributelor independente. Ulterior testării, rezultă un set de date care conține și valorile variabilei dependente. Performanța modelului se calculează pe baza informațiilor din acest set de date.
confusion matrix	matrice de confuzie	Matrice ce conține numărul de cazuri din fiecare clasă care sunt prezise corect (sunt A și sunt prezise A; sunt B și sunt prezise B), respectiv incorect (sunt A și sunt prezise B; sunt B și sunt prezise A).
accuracy	acurateță	Raportul dintre numărul cazurilor clasificate corect și numărul total de cazuri x 100.
precision	precizie	În relație cu una dintre clasele atributului de interes / țintă, se referă la ponderea cazurilor clasificate corect ca aparținând la acea clasă.
recall	probabilitate de detecție ~ rapel ~ reamintire ~ recuperare	În relație cu una dintre clasele atributului interes / țintă, se referă la ponderea cazurilor din acea clasă care sunt clasificate corect ca aparținând clasei respective.

Echivalențe terminologice

Statistică	Data mining / machine learning
Variabilă dependentă	Output
Variabilă prezisă	Variabilă de interes / țintă (target)
Răspuns	Variabilă label (RapidMiner)
Variabile independente	Variabile de input
Variabile explicative	Trăsături (features)
Predictori	Variabile comune (regular) (RapidMiner)
Covariate	
Observații	Instanțe
Cazuri	Cazuri
Subiecți	Exemple (RapidMiner)



ISBN: 978-606-37-1496-2
ISBN: 978-606-37-2303-2