Mircea Comșa

Data mining pentru științele sociale

Volumul 1.
Pregătirea datelor în RapidMiner Studio

Presa Universitară Clujeană

MIRCEA COMȘA

•

DATA MINING PENTRU ȘTIINȚELE SOCIALE

VOLUMUL I
PREGĂTIREA DATELOR ÎN RAPIDMINER STUDIO

Volum finanțat de Universitatea Babeș-Bolyai
prin Fondul de Dezvoltare UBB 2021,
grant de tip seed (cod GS-UBB-SOCASIS-MIRCEACOMSA)

MIRCEA COMȘA

DATA MINING
PENTRU ȘTIINȚELE SOCIALE

VOLUMUL I

PREGĂTIREA DATELOR ÎN RAPIDMINER STUDIO

PRESA UNIVERSITARĂ CLUJEANĂ

2022

Referenţi ştiinţifici:

Prof. Univ. Dr. Dan Chiribucă,
Universitatea „Babeș-Bolyai" din Cluj-Napoca

Prof. Univ. Dr. Bogdan Voicu,
Universitatea „Lucian Blaga" din Sibiu

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. I: 978-606-37-1497-9

Universitatea Babeş-Bolyai
Presa Universitară Clujeană
Director: Codruța Săcelean
Str. Hașdeu nr. 51
400371 Cluj-Napoca, România
Tel./fax: (+40)-264-597.401
E-mail: editura@ubbcluj.ro
http://www.editura.ubbcluj.ro/

Cuprins

LISTA TABELELOR

LISTA FIGURILOR

LISTA GIFURILOR[1]

[1] Gifurile animate sunt vizibile doar în varianta epub a cărții, disponibilă pe pagina web a editurii.

1. Introducere

1.1. De ce acest manual?

În acest sub-capitol ne propunem să răspundem la patru întrebări cu privire la alegerile făcute în cadrul manualului. De la general la particular, întrebările sunt:

(1) De ce „Data Mining”?

(2) De ce „O analiză a datelor bazată pe proces”?

(3) De ce „Pregătirea datelor”?

(4) De ce „RapidMiner Studio”?

Înainte de a răspunde pe rând la fiecare dintre aceste întrebări, vom descrie pe scurt pașii unui proiect de data mining. Scopul este de a contextualiza locul manualului și mai ales locul acestui volum în contextul mai general al unui proiect de data mining.

Pașii unui proiect de data mining: modelul CRISP-DM

Dacă dorim ca proiectele noastre de analiză să producă o diferență, este necesar să înțelegem rolul lor într-un context mai general definit de paradigma CRISP-DM (Cross-Industry Standard Process for Data Mining). Conform acesteia, orice proiect de data mining parcurge următorii șase pași (Figura 1.1-1):[2]

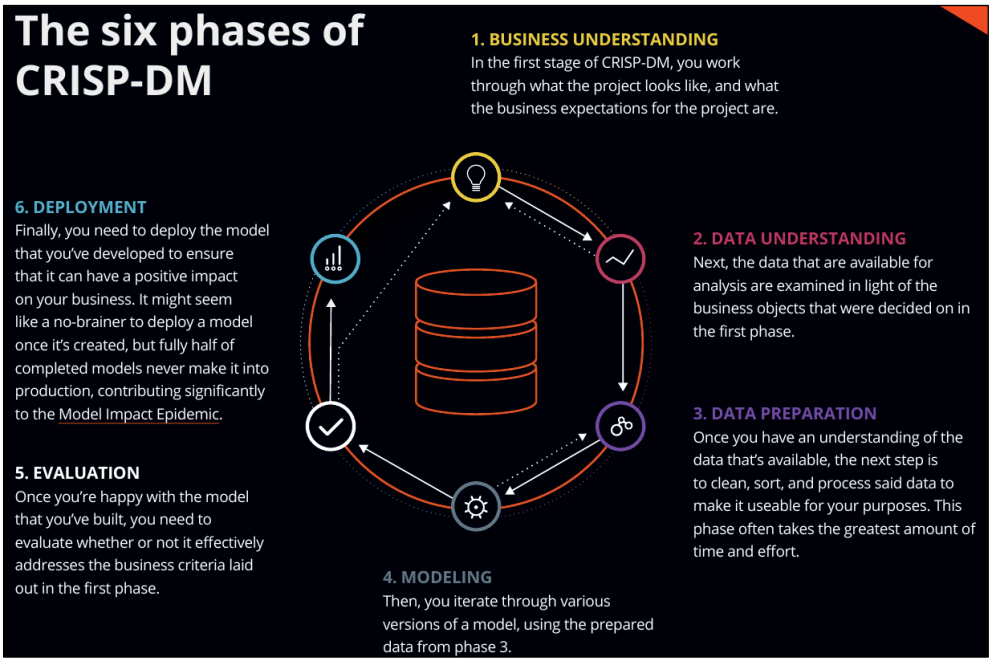
[2] O serie de alți autori folosesc acest model pentru a structura discuția cu privire la realizarea unui proiect de data mining (Chisholm, 2013; North, 2018).

(1) înțelegerea problemei și/sau oportunității (de afacere, intervenție etc.) (**business understanding**); în această fază e foarte important să ne familiarizăm cu problema / oportunitatea și să avem o reprezentare clară cu privire la rezultatele așteptate;
(2) familiarizarea cu și înțelegerea datelor disponibile, respectiv identificarea surselor utile de date și/sau producerea unor date (**data understanding**);
(3) pregătirea datelor pentru analiză (**data preparation**); aici intră activități precum curățarea datelor, transformarea acestora, analize simple etc.;
(4) analiza / modelarea datelor, realizarea modelului / modelelor (**modeling**);
(5) evaluarea modelului / modelelor, și, dacă sunt mai multe, compararea acestora, respectiv combinarea lor (**evaluation**); important, trebuie să verificăm dacă analiza (rezultatele acesteia) răspunde nevoilor de beneficiarilor proiectului de data mining;
(6) implementarea modelului / modelelor (**deployment**), punerea acestora în producție (multe dintre modelele produse nu ajung în această fază[3]).

Niciunul dintre pașii prezentați nu poate fi realizat corect dacă pașii anteriori nu au fost implementați cât mai bine posibil. De exemplu, dacă nu avem o înțelegere deplină a problemei / oportunității, nu vom putea să identificăm datele necesare, să le pregătim pentru analiză, respectiv analizăm.

[3] Un exemplu celebru în acest sens este Netflix Prize. Compania Netflix a lansat în 2006 o competiție deschisă cu scopul de a îmbunătăți algoritmul de predicție a evaluărilor filmelor de către utilizatori. Algoritmul trebuia să se bazeze doar pe evaluările anterioare (nu putea să folosească informații despre utilizatori sau filme). Premiul cel mare (1 milion $) a fost câștigat în 2009 de echipa "BellKor's Pragmatic Chaos". Algoritmul câștigător a fost semnificativ mai bun comparativ cu algoritmul Netflix (calitatea predicției, măsurată ca RMSE - root mean square error, a fost cu 10.09% mai mare, adică RMSE a scăzut de la 0.9525 la 0.8567). Cu toate acestea, algoritmul câștigător nu a fost pus niciodată în producție.

Figura 1.1-1. Cele șase faze ale CRISP-DM



Sursa: Schmitz, Martin. A Human's Guide to Machine Learning Projects. RapidMiner Withepaper (*https://rapidminer.com/resource/humans-guide-machine-learning-projects/)*

Primul volum al acestui manual tratează pasul 3, pregătirea datelor pentru analiză, iar următoarele volume vor trata pașii 4-6 (modelare, evaluare, implementare).

De ce „Data Mining"?

Data mining este o disciplină compozită în sensul că are la bază sau reprezintă o combinație de discipline precum: matematica, statistica, informatica, învățarea automată (machine learning), recunoașterea paternurilor, căutarea și extragerea informațiilor (information retrieval), inteligența artificială și managementul bazelor de date (Kotu & Deshpande, 2015, p. xvi; North, 2018; Roiger, 2017, p. 5).

Data mining este probabil unul dintre conceptele în cazul cărora numărul de definiții aproape îl egalează pe acela al cărților care discută despre acest subiect. La începuturi a fost folosit interșanjabil cu conceptul „knowledge

discovery in databases" (KDD). Potrivit acestor concepte, cunoașterea poate fi obținută prin extragerea ei din date (Roiger, 2017, p. 5). Adesea, conceptul de data mining a avut o conotație negativă, fiind asociat cu practicile de interogare extremă și neghidată teoretic a datelor cu scopul de a le face să „mărturisească" ceva, orice (oarecum similar cu „data snooping") (Bunge & Judson, 2005). Majoritatea definițiilor pun accentul pe acest proces de extragere (automată) a informațiilor, cunoașterii dintr-un volum mare de date, de unde și denumirea inițială de „knowledge discovery in databases":

> *„Data mining refers to a set of approaches and techniques that permit 'nuggets' of valuable information to be extracted from vast and loosely structured multiple data bases." (Olkin & Sampson, 2001).*
> *„DM takes a relative mountain of information (data) and attempts to extract a few gems or nuggets of knowledge" (Tretter, 2003).*
> *„… there is a large quantity, a mountain, of data, and this mountain is mined for nuggets of valuable information" (Bunge & Judson, 2005).*
> *„Data mining is the extraction of implicit, previously unknown, and potentially useful information from data"* (Hofmann & Klinkenberg, 2016, p. xxiv).
> *„We define data mining as the process of finding interesting structure in data" (Roiger, 2017, p. 5). „Structure" poate lua diferite forme precum: set de reguli, grafic, rețea, arbore decizional, una sau mai multe ecuații.*
> *„The science that analyze crude data to extract useful knowledge (patterns) from them. ... can also include data collection, organization, pre-processing, transformation, modeling and interpretation" (Moreira et al., 2019). (Definiția este pentru conceptul de data analytics. Însă, după cum se observă, este foarte similară cu definițiile conceptului de data mining.)*

Unele definiții merg dincolo de identificarea paternurilor, relațiilor și adaugă nevoia de a transforma informația rezultată în cunoaștere:

> *„Data mining is the name given to a variety of computer-intensive techniques for discovering structure and for analyzing patterns in data. (...) finding structures in the data (...) turning data into information" (Attewell & Monaghan, 2015, p. 3).*
> *„Data mining helps to discover underlying structures in the data, to turn data into information, and information into knowledge"* (Hofmann & Klinkenberg, 2016, p. xxiv).

Alte definiții specifică tipul de analize (corelații, anomalii, paternuri) și scopul general al acestora (obținerea unor predicții):

> *„Data mining is the process of finding anomalies, patterns and correlations within large data sets to predict outcomes" (SAS[4])*
> *„Using those patterns, data mining can create predictive models, or classify things, or identify different groups or clusters of cases within data" (Attewell & Monaghan, 2015, p. 3).*
> *„The use of machine-learning algorithms to find patterns of relationship between data elements in large, noisy, and messy data sets, which can lead to actions to increase benefit in some form (diagnosis, profit, detection etc.)" (Nisbet et al., 2018, p. 22).*

Rezultatul direct al unei analize de data mining ia forma unui model al datelor. Scopul ultim este de a folosi acest model în cazul unor situații noi, de același tip, cel mai adesea pentru a clasifica sau prezice noile cazuri / instanțe / observații.

Data mining reprezintă o varietate de proceduri de analiză a datelor (Bunge & Judson, 2005), o clasă de tehnici (Kotu & Deshpande, 2015, p. xvi), o varietate de metode și tehnici folosite pentru a analiza date cu scopul de a răspunde unor nevoi organizaționale. Toate aceste tehnici au în comun învățarea bazată pe inducție, adică, pornind de la o regulă observată pe un set de cazuri se trece la validarea și generalizarea acesteia (Roiger, 2017, p. 5).

Activitățile de data mining pot fi grupate în două categorii majore: pre-procesarea datelor și post-procesarea datelor. Fiecare dintre acestea include o serie de sub-activități:

- **pre-procesarea datelor** (discutată în primul volum al acestui manual):
  - accesarea și preluarea datelor: identificarea datelor utile, realizarea conexiunilor necesare, importarea datelor, unirea datelor;
  - explorarea datelor: statistici și vizualizări la nivel univariat și bivariat;
  - curățarea datelor: identificarea erorilor de preluare și introducere a datelor, eliminarea cazurilor duplicate, anonimizarea datelor;
  - transformarea datelor – relativ la cazuri: tratarea datelor lipsă, identificarea și tratarea outlierilor (cazurile neobișnuite), selectarea și eșantionarea cazurilor, respectiv generarea unor cazuri noi;

[4] https://www.sas.com/en_au/insights/analytics/data-mining.html

  - transformarea datelor – relativ la atribute: transformarea atributelor (recodare, normalizare, aplicarea unei funcții), generarea unor atribute noi, selectarea atributelor (feature selection), reducerea datelor (feature extraction); toate aceste activități sunt realizate în directă relație cu scopul și specificul analizei; o parte dintre activitățile incluse în această sub-etapă sunt realizate uneori (și) în faza de post-procesare a datelor;

- **post-procesarea datelor** (discutate în volumele următoare):
  - analiza descriptivă: statistici și vizualizări la nivel univariat și bivariat;
  - analiza predictivă: antrenarea / instruirea (training), validarea, testarea, implementarea și monitorizarea performanței unui model;
  - analiza prescriptivă: utilizarea unui model de predicție cu scopul de a identifica căile prin care realitatea viitoare posibilă poate fi modificată într-un sens dorit.

Statistică vs. Data Mining

Probabil principala diferență generală dintre cele două concepte, și totodată cea mai vizibilă, constă în faptul că tehnicile de data mining sunt mai potrivite pentru analiza datelor mari (Big Data). O comparație între statistică și data mining, în baza unor criterii relevante (obiectiv, testarea semnificației, eșantionare, relații non-liniare, interacțiuni), este prezentată în Tabelul 1.1-1.

Spre deosebire de statistică, tehnicile de data mining permit, cel puțin parțial, automatizarea etapelor de analiză care altfel ar necesita mult mai mult timp, precum (Attewell & Monaghan, 2015, p. 5):

- identificarea celor mai importanți predictori dintr-un număr mare de predictori potențiali,
- transformarea distribuției statistice a unor predictori,
- detectarea interacțiunilor complexe dintre predictori,
- descoperirea tipurilor de heterogenitate asociate predictorilor,
- compararea mai multor alternative,
- identificarea paternurilor,

- compararea și validarea modelelor,
- combinarea modelelor pentru a maximiza acuratețea predicției.

Tabelul 1.1-1. O comparație între statistică și data mining

Problemă	Statistică	Data mining
Puterea predictivă	Predicția nu este elementul central. Se acceptă valori mici ale lui R^2.	Scopul analizei este adesea predicția. Se doresc valori mari ale lui R^2.
Testele de semnificație	Reprezintă baza pentru generalizabilitate. Au o importanță critică pentru evaluarea ipotezelor și interpretarea mecanismelor. Unele practici de testare sunt problematice (testarea multiplă & simultană a semnificației; p-hacking). Heteroscedasticitatea reprezintă uneori o problemă.	Generalizabilitatea se obține cu ajutorul validării încrucișate (cross-validation: testează stabilitatea rezultatelor la nivelul mai multor subeșantioane). Unele tehnici sunt de tip „cutie-neagră" (nu au parametri interni interpretabili). Poate elimina / reduce, respectiv „păcăli" heteroscedasticitatea; multe dintre tehnicile folosite sunt non-parametrice.
Eșantionarea	Testele de semnificație sunt legate de asumpțiile eșantionării. Toate eșantioanele sunt aleatoare (simple sau complexe).	Folosește validarea încrucișată, bootstrap, teste de permutare. Eșantioanele de convenienț sunt acceptate.
Relațiile non-liniare între X și Y	Adesea, netestate sau ignorate.	Identificarea parțial automatizată a relațiilor non-liniare.
Interacțiunile dintre predictori	Adesea, netestate sau ignorate. Interesează efectele principale.	Identificarea parțial automatizată a efectelor de interacțiune și a celor eterogene.

Sursa: (Attewell & Monaghan, 2015, p. 27)

De ce „O analiză a datelor bazată pe proces"?

Analiza datelor sau, mai general, știința datelor (data science), poate fi realizată în moduri destul de diferite. La o extremă, avem o abordare bazată pe codare / programare (code-based), iar la cealaltă extremă o abordare bazată pe contactul direct și continuu cu datele (data-centric). Abordarea dominantă pare să fie cea bazată pe codare (Python, de exemplu), în principal datorită faptul că este extrem de puternică, flexibilă și ușor de refolosit. Pe de altă parte, este dificil de învățat. Abordarea de tip data-centric (Excel) pornește de la datele analizate: ne uităm la date, le modificăm și le editățm. Simplu spus, lucrăm direct și constant cu datele, ceea ce este extrem de

intuitiv pentru că observăm rapid care este impactul fiecărei schimbări. Această abordare este „ascunsă" (nu știm cum anume au fost modificate datele, realizate analizele) și, de aici, dificil de reluat și reutilizat.

Pornind de la câteva dimensiuni pe care le apreciem ca relevante (de exemplu, capacitatea de a rezolva probleme diverse și complicate, rapiditatea și reproductibilitatea analizelor etc.), putem identifica o serie de avantaje și limite relativ la fiecare dintre cele două abordări (Tabelul 1.1-2). Preferăm una sau alta dintre abordări în funcție de obiectivele de cunoaștere, domeniul de activitate, nevoile de analiză, nevoile practice, experiență, comoditate și/sau obișnuință.

Tabelul 1.1-2. Abordări posibile relativ la analiza / știința datelor

Tip abordare (data science approach)	**Cod / programare (code-based)**	**Proces (process-based)**	**Date (data-centric)**
Probleme diverse	+++	+++	+
Probleme simple	+	+++	+++
Probleme complexe	++	+++	+
Rapiditate	++	+++	+
Eficiență	++	+++	+
Flexibilitate	+++	+++	+
Repetabilitate	+++	+++	+
Reproductibilitate	+++	+++	+
Transferabilitate	++	+++	+
Intuitivitate	+	++	+++
Dificultate	+	++	+++
Durată învățare	+	++	+++
Număr specialiști	+	+	++
Softuri (exemple)	R, Python, JavaScript, Julia SAS →	RapidMiner, Knime, SPSS Modeler, Weka	SPSS, Excel Stata

Printr-o simplă schimbare de logică (Gif 1.1-1), softul de data mining RapidMiner Studio, alături de alte softuri cu o filosofie similară, combină avantajele oferite de cele două abordări „extreme", code-based și data-centric, și propune o perspectivă nouă, bazată pe proces. Simplu spus,

RapidMiner Studio oferă posibilitatea de a face lucruri foarte complicate într-o manieră simplă, ușor repetabilă, fără nevoia de programare / codare (deși acest lucru este posibil), utilizatorul fiind în același timp foarte aproape de date (putem observa relativ repede care este impactul comenzilor asupra datelor).

Gif 1.1-1. Analiza datelor în viziunea RapidMiner (process-based data science)

Sursa: Ingo Mierswa. 2018. Data Preparation: Time consuming and tedious? (https://rapidminer.com/blog/data-prep-time-consuming-tedious/)

Prin modulele Turbo Prep, Auto Model și Deployment, RapidMiner Studio merge și mai mult în această direcție de democratizare a lucrului cu datele în condițiile păstrării tuturor celorlalte avantaje. Turbo Prep este prezentat în ultimul capitol al acestui volum, celelalte module urmând să fie discutate în volumul II.

În concluzie, abordarea bazată pe proces este de preferat deoarece permite utilizarea ambelor căi (code-based, data-centric), este mai eficientă (reduce timpul necesar pentru realizarea unei analize, mai ales relativ la etapa de pregătire a datelor) și mai ușor de învățat, de aici și mai ieftină de implementat. În această nouă logică, nu este nevoie ca analistul să scrie cod pentru a fi un bun analist, respectiv pentru a construi un model util. Din

perspectiva recrutării resurselor, nu mai este nevoie să depunem eforturi supra-umane pentru a căuta „unicorni”[5], a-i recruta și apoi supra-plăti.

De ce „RapidMiner Studio”?

În societatea actuală tot mai multe date sunt produse și stocate. Companiile și instituțiile resimt tot mai acut nevoia de a analiza acestea date și a lua decizii informat. Pentru a realiza această tranziție e nevoie de tot mai mulți analiști de date (data scientist). În prezent, cererea pentru astfel de expertiză depășește mult oferta. În acest context, misiunea asumată a companiei RapidMiner și a fondatorului acesteia, Ingo Mierswa, este de a facilita accesul tuturor celor interesați la analizele de tip data mining:

> *„We want to empower anybody to do super-innovative analytics.”*[6]

RapidMiner Studio și softurile conexe (RapidMiner AI Hub, RapidMiner Radoop) au fost construite cu acest scop în minte. Aceste softuri reprezintă un tot unitar care permite tuturor celor interesați, mai mult sau mai puțin specializați, să realizeze ușor și rapid o analiză de tip data mining. RapidMiner Studio are o interfață grafică (GUI) cu ajutorul căreia întreg procesul de data mining poate fi definit interactiv, simplu, doar prin realizarea unor serii de operații de tip drag&drop, adică prin selectarea și conectarea unor operatori / comenzi predefinite. Niciunul dintre pași nu necesitată cunoașterea unor elemente de programare, deși, cei care doresc pot face acest lucru. Simplu spus, RapidMiner Studio oferă posibilitatea de a utiliza în paralel cele două abordări (code-based and code-free / procesed-based). Mai mult, dacă dorim, etapele de pregătire a datelor și de modelare pot fi realizate automatizat (în cea mai mare parte).

---

[5] În acest context folosim termenul de unicorn pentru a ne referi la o persoană care stăpânește simultan o serie de domenii de specialitate, adesea foarte diferite între ele, precum diferite limbaje de programare, baze de date, matematică și statistică avansate, softuri de prezentare și vizualizare a datelor etc. Am preluat metafora de la Ingo Mierswa (https://rapidminer.com/blog/what-is-data-science/). Cerințele menționate în anunțurile de angajare sunt o dovadă clară a faptului că multe companii caută astfel de angajați.

[6] Sau, spus în alte forme, “empower nonexperts to get to the same findings as data scientists” sau “democratization of advanced analytics”. Mierswa, Ingo. 2015. The RapidMiner Philosophy. (https://rapidminer.com/blog/rapidminer-philosophy/).

Softul RapidMiner Studio are două versiuni principale, Free și Enterprise. Ambele pot fi descărcat pentru platforma preferată de la adresa https://rapidminer.com/. Varianta „free" are aceleași funcționalități, dar setul de date analizat este limitat la cel mult 10.000 de cazuri.[7] Pentru cele mai multe proiecte mici (personale sau instituționale) această constrângere nu pune probleme. Pentru mediul academic există posibilitatea obținerii unei licențe educaționale (trebuie reînnoită în fiecare an). Aceasta permite lucrul cu seturi de date care au un număr nelimitat de cazuri.

Indiferent de variantă, RapidMiner Studio este un soft extrem de intuitiv și ușor de utilizat, fiind astfel potrivit pentru toți cei mai puțin orientați spre programare și comenzi bazate pe sintaxe. „Sintaxa vizuală" propusă reprezintă o soluție care facilitează învățarea și înțelegerea. La puțin timp după contactul cu softul, aproape oricine poate realiza o analiză simplă alegând operatorii necesari și conectându-i între ei într-o succesiune logică. Acolo unde e cazul, preferințele relativ la fiecare operator pot fi alese rapid, utilizatorul având posibilitatea să folosească setările implicite sau pe cele preferate de majoritatea celorlalți utilizatori. Mai mult, exemplele tipice de analiză oferite de soft pot fi ușor adaptate pentru a răspunde propriilor nevoi și întrebări de cercetare.

Toate proiectele realizate în RapidMiner Studio pot fi salvate, asigurându-se astfel reproductibilitatea, reutilizarea și auditul lor. În acest context, foarte important, RapidMiner Studio asigură compatibilitatea între versiuni. Simplu spus, procesele (operatorii) care au funcționat într-o versiune anterioară vor funcționa și în versiunile ulterioare chiar dacă unii operatori au suferit schimbări între timp (denumire, opțiuni).

În același timp, RapidMiner Studio este un soft profesionist, destinat și utilizatorilor experimentați. Aceștia pot analiza date structurate și nestructurate, respectiv date de tip numeric, text și imagine. Diversitatea și numărul de modele ce pot fi rulate sunt impresionante, acoperind practic toate nevoile. Utilizatorii au posibilitatea să automatizeze toți pașii unui

[7] Alte diferențe vizează numărul de procesoare (putem folosi doar un procesor în varianta Free), respectiv opțiunile suplimentare Turbo Prep, Auto Model, Models Ops (Deployment), executarea unui proces în background (nu sunt disponibile în varianta Free).

proiect de data mining, de la pregătirea datelor, la construirea și testarea modelelor, respectiv la punerea lor în producție. Utilizatorii pot interacționa cu și rula comenzi scrise în alte programe / limbaje (Python, R, SQL). Soluțiile conexe oferite, RapidMiner Go, RapidMiner Notebooks, RapidMiner AI Hub, RapidMiner Radoop fac posibilă implementarea unor proiecte de data mining la cele mai înalte standarde de calitate și eficiență. Pe scurt, ecosistemul RapidMiner este unul complex, indiferent de nivelul de interes, puternic integrat cu alte platforme:[8]

- **Integrarea datelor (Data pipeline)**: RapidMiner poate prelua date de diferite tipuri, dintr-o mulțime de surse: Hadoop/Spark, Excel, fișiere, documente, social media, emailuri, baze de date, servicii cloud, Web, HDFS și Hive, diferite softuri dedicate gestionării, analizei și prezentării datelor (Tableau, Qlik, PowerBI, Grafana);
- **Implementarea modelelor (Deployment)**: modelele RapidMiner pot fi rulate (implementate) folosind diferite soluții: Docker și Kubernetes, platforme cloud (AWS Azure, Google Cloud, Microsoft Azure) sau sisteme de operare (Mac, Linux, Windows);
- **Învățare automată (Machine Learning)**: RapidMiner permite integrarea cu diferite limbaje de programare, librării de învățare automată și „deep learning": Python, R, Java, Groovy, PySpark, SparkR, SQL Scripting, H2O, Deep Learning (Keras) – DL4J, Weka, Keras, Tensorflow, Theano, Microsoft CNTK, NVIDIA CUDA.

Platforma RapidMiner este una completă, în sensul că acoperă toți pașii unui proiect de „Data Science":

- **Pregătirea datelor (Data engineering)**: acțiuni precum conectarea, achiziția, explorarea, pregătirea datelor sunt realizate simplu, cu un efort cât mai redus;
- **Construirea modelului (Model building):** se poate realiza simplu, în diferite moduri (automat, vizual, cod), indiferent de nivelul de specializare;

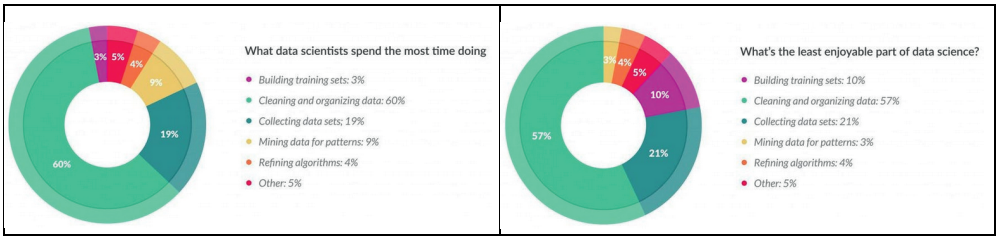
[8] https://rapidminer.com/platform/integrations/

- **Lucrul cu modele (Model ops):** permite o serie de acțiuni relativ la modele precum implementare, evaluare, comparare, monitorizare, management, înlocuire;
- **Construirea unor aplicații inteligente (AI app building):** cu ajutorul aplicațiilor inteligente, modelele și rezultatele acestora devin ușor accesibile factorilor de decizie, nefiind nevoie ca aceștia să fie specialiști în analiză și programare;
- **Colaborare & Guvernare (Collaboration & Governance):** facilitează comunicarea deschisă, lucrul în echipă, reutilizarea proiectelor anterioare;
- **Încredere & Transparență (Trust & Transparency):** facilitează înțelegerea modulului în care funcționează un model, cum produce predicțiile (explainable AI), respectiv anticipează beneficiile care vor fi cel mai probabil obținute în urma aplicării unui model.

De ce „Pregătirea datelor”?

Cea mai mare parte a timpului necesar pentru realizarea unui proiect de data mining este în fapt destinat pregătirii datelor pentru analiză (Figura 1.1-2). Aproximativ 80% din timpul total este alocat pentru realizarea unor activități precum importarea, curățarea, vizualizarea, restructurarea și sumarizarea datelor (Chisholm, 2013, p. 1). Construirea și testarea unui model, evaluarea și vizualizarea rezultatelor necesită cel mai adesea o perioadă de timp semnificativ mai redusă în comparație cu pre-procesarea datelor (Hofmann & Klinkenberg, 2016, p. xx).

Figura 1.1-2. Pregătirea datelor pentru analiză: durată mare, plăcere redusă



Sursa: Press, Gill. 2016. Cleaning Big Data: Most Time-Consuming, Least Enjoyable Data Science Task, Survey Says. https://www.forbes.com/sites/gilpress/2016/03/23/data-preparation-most-time-consuming-least-enjoyable-data-science-task-survey-says/?sh=540bab466f63

Pe lângă faptul că pregătirea datelor consumă foarte mult timp, este considerată a fi și etapa cea mai plictisitoare și, uneori, etapa cea mai puțin importantă. Prin urmare există riscul ca pregătirea datelor să fie realizată „în fugă" și/sau atribuită către persoane mai puțin calificate. Lucrurile nu ar trebui să stea deloc așa, pregătirea bazei reprezentând fundamentul pe care urmează să fie construiți pașii următori. Fără date de calitate, analizele realizate pot fi lipsite de utilitate, respectiv pot direcționa greșit resursele instituționale. Un model de data mining construit pe date greșite va produce predicții greșite, respectiv va duce la concluzii și decizii greșite. Pe scurt, în loc să ajute la economisirea resurselor, utilizarea unui model greșit va crește costurile (bani, timp, resurse umane).

1.2. Data Mining și „rudele" sale

Comparativ cu alte domenii de studiu, domeniul stocării, analizei și utilizării datelor este unul în care se intersectează relativ mai multe discipline și tradiții de cercetare. Firesc, specialiștii care lucrează în această zonă provin din discipline destul de diferite. Comparativ cu alte domenii, acesta este unul relativ nou. Toți acești factori determină o oarecare imaturitate conceptuală a domeniului, probabil chiar o „junglă conceptuală". Sunt utilizate foarte multe concepte, uneori același concept este folosit cu sensuri diferite, alteori două concepte sunt folosite interșanjabil deși sunt distincte (Kotu & Deshpande, 2019, p. 2). Secțiunea de față își propune să prezinte pe scurt principalele concepte utilizate în acest domeniu alături de relațiile dintre ele (în principal relația cu conceptul data mining). Pentru aceasta, vom folosi și câteva diagrame conceptuale sugestive.

Big Data

Recunoscând caracterul imprecis al conceptului de Big Data, uneori acesta este definit în literatura de specialitate atât de larg încât include toate activitățile realizate în relație cu datele:

> *„The term 'Big Data' is an imprecise description of a rich and complicated set of characteristics, practices, techniques, ethical issues, and outcomes all associated with data"* (Japec et al., 2015, p. 5).

Chiar dacă această definiție pare să fie preferată de unii autori (Foster et al., 2021, p. 3), considerăm că este incorectă pentru simplul motiv că prezintă suprapuneri mari cu definițiile altor concepte. Alteori, conceptul de Big Data este definit prin referire la procesul de colectare și analiză a datelor cu scopul de fundamenta luarea deciziilor:

> *„Big data refers to the collection, analysis, and use of massive amounts of digital information for decision making and operational applications" (McNeely & Schintler, 2022, p. 81).*

În contextul acestui manual considerăm că Big Data (date foarte mari, extrem de voluminoase) se referă la datele care conțin foarte multe informații relativ la foarte multe cazuri, uneori și relativ la foarte multe momente de timp. Informațiile pot lua forme diferite, funcție de tipul datelor: atribute / variabile în cazul datelor structurate, respectiv cuvinte, imagini, sunete sau combinații ale acestora în cazul datelor nestructurate. Cel mai adesea, datele mari sunt produse automat, cu ajutorul tehnologiei (senzori incluși în diferite obiecte – ceasuri și brățări inteligente, camere de luat vederi etc. – programe de monitorizare a traficului pe internet sau a utilizării unor softuri etc.), fără o implicare directă a factorului uman. Dincolo de acestea aspecte generale, e necesar să identificăm caracteristicile pe care datele trebuie să le îndeplinească pentru a intra în categoria Big Data.[9]

Tradițional, se consideră că Big Data include datele care îndeplinesc următoarele trei condiții, respectiv au aceste caracteristici (Balusamy et al., 2021; McNeely & Schintler, 2022; Moreira et al., 2019):

- **Volum** mare – adică sunt prea mari pentru a fi stocate și analizate cu ajutorul tehnologiilor convenționale, fiind nevoie de nevoie de tehnici și instrumente noi precum MapReduce, Hadoop, Spark, Storm **(VOLUME)**;

[9] Pentru o scurtă prezentare în limba română a conceptului de Big Data se poate vedea și volumul "R cu aplicații în statistică" (Dușa et al., 2015, pp. 191–193).

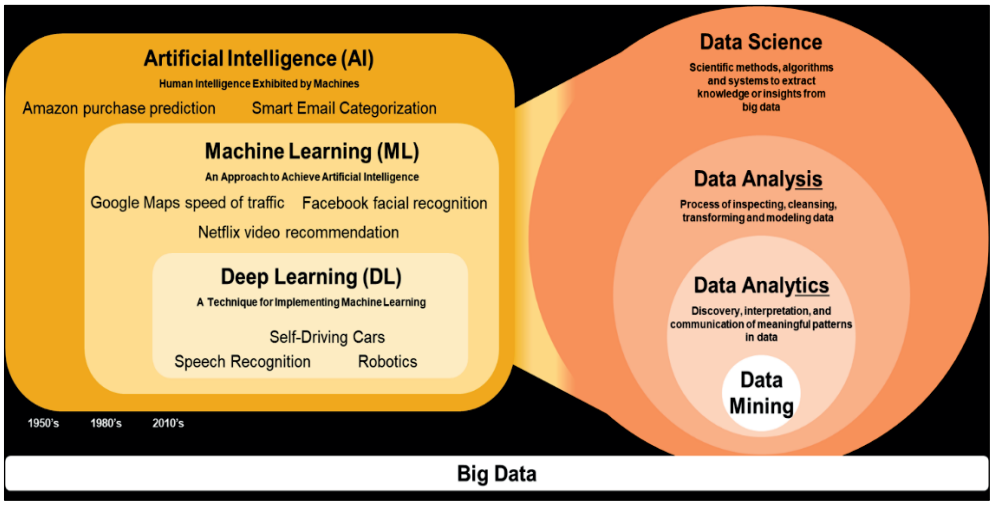
- **Varietate** mare – sunt diverse relativ la diferite criterii; după gradul de structurare pot fi structurate, semistructurate și nestructurate; după tip: imagine, audio, video, text, numere; de asemenea, astfel de date provin adesea dintr-un număr mare de surse **(VARIETY)**;
- **Velocitate** mare – sunt produse și puse la dispoziție într-un ritm foarte alert, deci impun utilizarea unor soluții tehnice care să răspundă nevoilor de integrare și prelucrare rapidă a acestor date **(VELOCITY)**.

Alături de aceste caracteristici și/sau provocări, ulterior au fost adăugate și altele, precum (McNeely & Schintler, 2022, p. 80):

- **Variabilitate** – este reflectată în inconsistențele relativ la fluxurile de date ce însoțesc varietatea și complexitatea datelor mari **(VARIABILITY)**;
- **Validitate și fidelitate** – se referă la calitatea și integritatea datelor, la gradul în care putem avea încredere în acestea; adesea, datele mari sunt incomplete, redundante și distorsionate, deci trebuie să acordăm o atenție specială metodelor de verificare și validare a acestora **(VERACITY)**;
- **Vulnerabilitate** – se referă la provocările cu privire la securitatea datelor, respectiv la nevoia de a asigura caracterul privat al acestora; în acest context, o problemă majoră o reprezintă transferul, distribuția și inter-conectarea datelor **(VULNERABILITY)**;
- **Valoare** – se referă la capacitatea datelor de a fi valoroase, respectiv la capacitatea utilizatorilor de a extrage valoare din acestea **(VALUE)**.

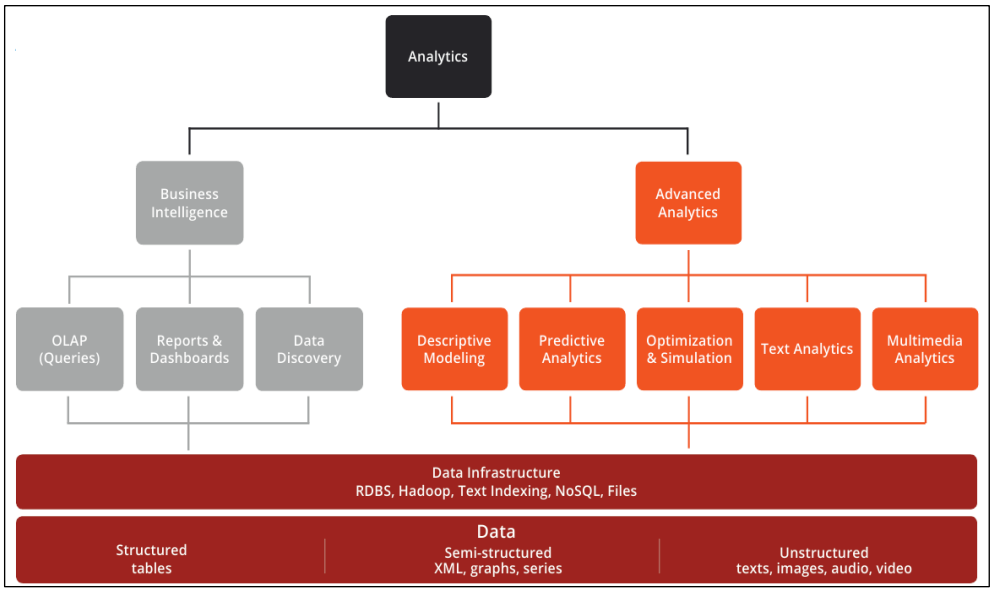
Problemele semnalate în relație cu caracteristicile Big Data au în comun faptul că pot fi rezolvate cu ajutorul tehnologiei. Acesta este și motivul pentru care în diferite reprezentări vizuale ale relațiilor dintre conceptele majore din acest domeniu, conceptul de Big Data și tehnologia asociată gestionării acestor date mari apar la bază, semnalând rolul de fundament al acestora pentru activitățile de analiză, respectiv pentru aplicațiile practice (Figura 1.2-1, Figura 1.2-2). Simplu spus, Big Data colectează și gestionează date, Data Science folosește diferite abordări și tehnici pentru a analiza datele mari cu scopul de a descoperi informații noi și utile, iar AI materializează sub formă de aplicații practice toate aceste date și modelele asociate lor.

Figura 1.2-1. Big Data, Artificial Intelligence și Data Science



Vollmer, Marcell. 2020. How to make it simple to explain AI, ML, DL together with Data Science, Data Analysis & Analytics and Data Mining? (https://www.linkedin.com/pulse/how-make-simple-explain-ai-ml-dl-together-data-science-vollmer)

Figura 1.2-2. Tipuri de date, infrastructură de date și analiza datelor



Sursa: RapidMiner. An Introduction to Advanced Analytics. https://rapidminer.com/wp-content/uploads/2014/04/advanced-analytics-introduction.pdf

Data Science

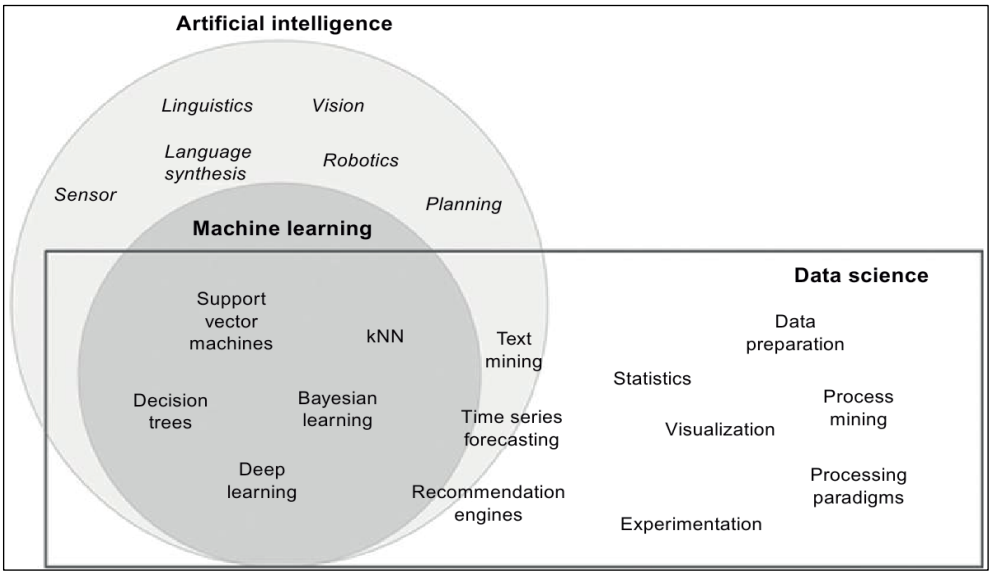
Relativ nou, conceptul de Data Science (știința datelor) este definit adesea prin referire la domeniul larg al disciplinelor interesate de studiul datelor. Simplu spus, Data Science nu este o știință în sensul strict al termenului ci o disciplină care integrează mai multe științe. O persoană care activează în cadrul acestei discipline se numește „data scientist". E necesar ca o astfel de persoană să posede cunoștințe din domenii multiple precum informatică, matematică, statistică, științe sociale și, foarte important, domeniul de specialitate pe care îl analizează (cel la care se referă datele pe care le analizează).

Activitățile și tehnologiile dezvoltate în cadrul disciplinelor grupate sub umbrela Data Science au rolul de a răspunde unei nevoi tot mai presante, aceea de a căuta sensuri în munții de date produși de activitățile umane, dar nu numai. Printre disciplinele incluse sub această umbrelă sunt enumerate adesea următoarele: data mining, machine learning, deep learning, artificial intelligence. Uneori, conceptul de Data Science este restrâns strict la zona de analiză a datelor, deci excluzând sfera AI (Figura 1.2-1), incluzând aici activități precum stocarea și accesarea datelor mari, pregătirea datelor pentru analiză, modelarea și prezentarea datelor. Mergând și mai mult în această direcție, unii autori restrâng conceptul de Data Science la aplicațiile din domeniul economic ale acelorași discipline, accentul fiind pus pe extragerea valorii economice din date. În această viziune, Data Science și Data Mining sunt oarecum echivalente (sau se suprapun în mare parte):

> *„Data science is the business application of machine learning, artificial intelligence, and other quantitative fields like statistics, visualization, and mathematics. It is an interdisciplinary field that extracts value from data. In the context of how data science is used today, it relies heavily on machine learning and is sometimes called data mining" (Kotu & Deshpande, 2019, p. 4).*
>
> *„Data science or data analytics is defined as the process of extracting meaningful knowledge from data" (Roiger, 2017, p. 4).*

Dintr-o perspectivă diferită (Figura 1.2-3), conceptul de Data Science se intersectează cu cel de inteligență artificială, mai ales cu sub-domeniul acesteia numit Machine Learning (Kotu & Deshpande, 2019, p. 3).

Figura 1.2-3. Relația dintre AI, Machine Learning și Data Science



Sursa: (Kotu & Deshpande, 2019, p. 3)

În contextual acestui manual, folosim conceptul de Data Science într-un sens general, referindu-ne la orice activitate de analiză a datelor, incluzând aici vizualizarea datelor, analiza statistică, data mining, machine learning și inteligența artificială.

Data Analytics

Diverși autori consideră că Data Science și Data Analytics sunt echivalente. Diferența apare relativ la tipurile de activități care compun cele două concepte. Unii autori includ în acestea o serie largă de activități precum statistică, data mining, machine learning și vizualizarea datelor:

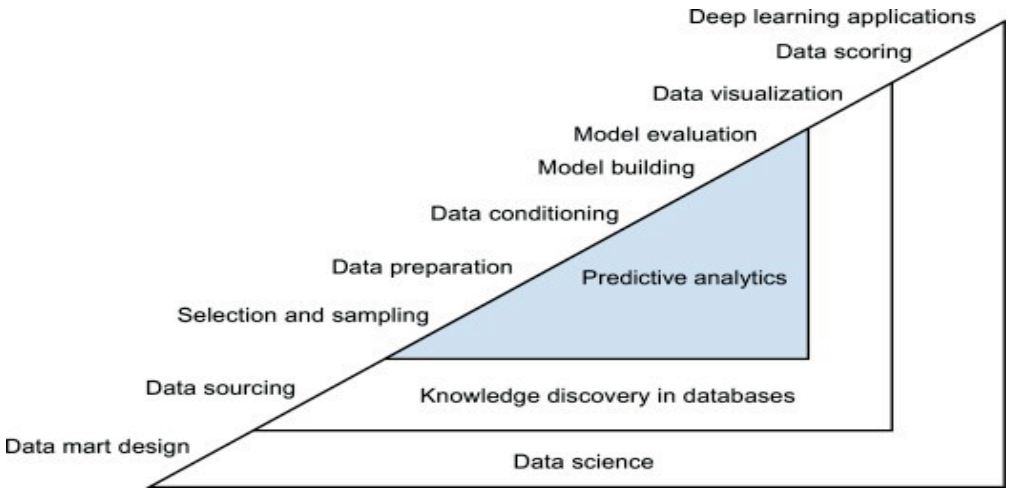
> *„… the term analytics is used to describe different techniques and methods that extract new knowledge from data and communicate it. The term comprises statistics, data mining, machine learning, operations research, data visualization, and many other areas"* (Pagans, 2015).

Alți autori includ doar partea de analiză a datelor, aceasta fiind echivalată cu data mining (Roiger, 2017, p. 4), deci, în viziunea acestor, conceptele de data science, data analytics și data mining sunt echivalente.

Alți autori pun semnul de egalitate între Data Analytics, mai exact Predictive Analytics (analiză predictivă) și Data Mining (Nisbet et al., 2018, p. 23). Conform acestei reprezentări, Data Science este conceptul general care îl înglobează pe cel de Knowledge Discovery, acesta din urmă incluzând analizele de tip Data Mining / Predictive Analytics (Figura 1.2-4).

> *„Knowledge discovery: The entire process of data access, data exploration, data preparation, modeling, model deployment, and model monitoring. This broad process includes data mining activities" (Nisbet et al., 2018, p. 22).*
> *„Data science: The extension of knowledge discovery into data architecture of analytic data marts on one hand and complex image, speech, and textual analysis on the other hand with highly evolved machine-learning algorithms" (Nisbet et al., 2018, p. 22).*

Figura 1.2-4. Data science, knowledge discovery și predictive analytics



Sursa: (Nisbet et al., 2018, p. 23)

Machine Learning și Deep Learning

Machine Learning (învățarea automată) se referă la algoritmii (learners) care au capacitatea de învăța din date (Kotu & Deshpande, 2019, p. 3). Rezultatul învățării este exprimat sub forma unui set de reguli formulate matematic și/sau logic (if-then). Privit din această perspectivă, conceptul de Machine Learning pare a se suprapune parțial cu conceptul de data mining (partea de post-procesare a datelor: antrenarea, validarea, testarea și implementarea unui model). Perspectiva adoptată în acest manual este că Machine

Learning[10] se referă la rezultatul unei analize de date mining, adică la setul de reguli extrase din date și implementarea acestora într-un program care realizează automat, fără intervenția unei persoane, sarcina pentru care a fost antrenat.

Să presupunem că dorim să identificăm candidații care au prezentat informații exagerate sau chiar false în CV. Cum putem învăța un algoritm (machine) să identice CV-urile care conțin astfel de informații? Cel mai simplu e să îi oferim algoritmului cât mai multe exemple de CV-uri din fiecare categorie, precizând totodată categoria din care fac parte. Pornind de la acestea, algoritmii vor identifica paternuri bazate pe prezența sau absența anumitor cuvinte, asocieri de cuvinte, și vor produce un set de reguli în baza cărora pot fi clasificate CV-uri noi, despre care nu știm din ce categorie fac parte. În continuare, algoritmul va exclude automat CV-urile care conțin elemente nedorite.

Conceptul de Deep Learning (învățare profundă) este definit cel mai adesea ca un sub-domeniu al Machine Learning. Machine Learning poate folosi algoritmi de diferite tipuri. Pentru a învăța din date, Deep Learning folosește algoritmii grupați sub eticheta generică de rețele neuronale. Acestea sunt inspirate din modul în care funcționează creierul uman și încearcă să reproducă tipul de învățare specific acestuia.

Artificial Intelligence

Inteligența artificială (AI) se referă la capacitatea „mașinilor" (într-un sens larg, roboți, aplicații informatice / software, sau orice alte sisteme de interacțiune și asistare umană: Siri, Alexa, Google Assistant, Cortana, Bixby) de a reproduce comportamentul și funcțiile umane. Cele mai cunoscute exemple sunt probabil sistemele de recunoaștere facială, conducere automată, roboții umani, chat bots sau trading bots. De exemplu, un sistem AI poate fi un soft care atunci când îi arătăm o imagine poate identifica elementele care apar în aceasta, ne poate oferi numele acestora, sugera

[10] În aplicațiile de deep learning poate exista și ceea ce se numește învățare „online", adică continuă în raport cu inputurile în timp real (în funcție de datele noi de la intrare, parametrii modelului se pot schimba de la un moment la altul și, implicit, predicțiile).

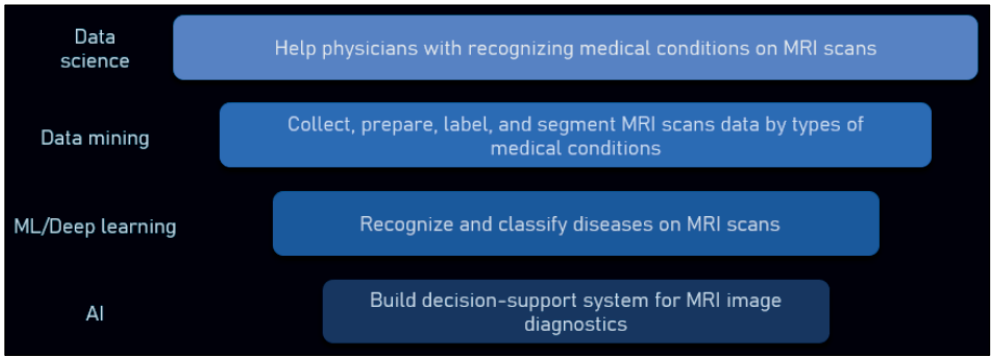
locurile de unde le putem cumpăra, oferi informații cu privire la utilitatea obiectelor etc. Pentru construirea unei inteligențe artificiale folosim Data Mining, Machine Learning și, uneori, Deep Learning.

> *„Artificial intelligence is about giving machines the capability of mimicking human behavior, particularly cognitive functions" (Kotu & Deshpande, 2019, p. 2).*

Un exemplu și o diagramă conceptuală

Să presupunem că dorim să asistăm doctorii în procesul de citire și interpretare a unei imagini a creierului obținute prin rezonanță magnetică (Figura 1.2-5). Pentru aceasta vom colecta mai multe imagini și le vom categoriza în funcție de tipul de diagnostic (medicii specialiști vor face acest lucru). Pornind de la acest set de imagini etichetate (labeled), folosim unul sau mai mulți algoritmi de machine learning, deep learning cel mai probabil, pentru a recunoaște și clasifica imaginile pe tipuri de diagnostic. Obținem astfel un set de reguli ce pot fi folosite pentru a clasifica alte imagini. În continuare, implementăm aceste reguli într-un program care să asiste medicul în procesul de diagnosticare a stării de sănătate / boală.

Figura 1.2-5. O ilustrare practică a relației dintre concepte

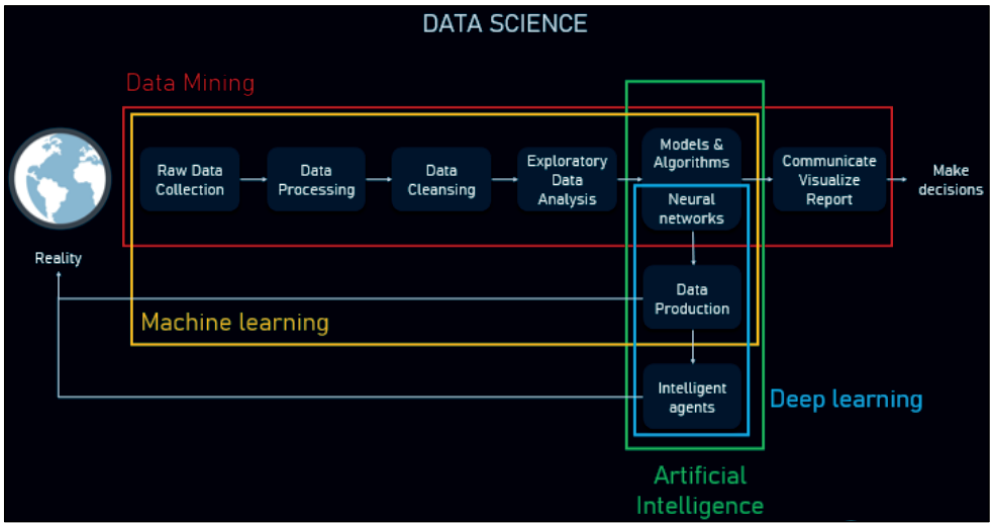


Sursa:
https://www.altexsoft.com/blog/data-science-artificial-intelligence-machine-learning-deep-learning-data-mining/

Plecând de la acest exemplu și discuțiile anterioare ale conceptelor putem încerca să construim o diagramă conceptuală care să ilustreze relațiile dintre toate aceste concepte (Figura 1.2-6). Putem reprezenta acest spațiu pornind

de la cel mai general concept, cel care le include pe toate celelalte, Data Science. În interiorul Data Science distingem diferite zone, cu activități specifice. Decupaje diferite relativ la aceste zone (activități) produc concepte diferite. Fiecare dintre aceste concepte include activități specifice, întâlnite doar în cazul acelui concept, respectiv activități care sunt comune și altor concepte. De exemplu, conceptul de Data Mining, cel mai cuprinzător după Data Science, include toate activitățile de colectare, pregătire, analiză și raportare a datelor. Cu excepția ultimului pas (comunicarea, vizualizarea), Data Mining se suprapune cu Machine Learning. Partea de modelare a datelor este comună cu AI, iar dacă avem în vedere doar modelarea cu ajutorul rețelelor neuronale ajungem în domeniul Deep Learning. Aceasta este totodată și schema conceptuală adoptată în cadrul acestui manual, cu o singură modificare. În viziunea noastră, Machine Learning se referă doar la două elemente: (1) setul de reguli rezultat în urma unei analize de Data Mining și (2) implementarea acestora într-un program care realizează automat, fără intervenția unei persoane, sarcina pentru care a fost antrenat / instruit.

Figura 1.2-6. O ilustrare teoretică a relației dintre concepte



Sursa:
https://www.altexsoft.com/blog/data-science-artificial-intelligence-machine-learning-deep-learning-data-mining/

Business Intelligence vs. Advanced Analytics

Organizațiile colectează tot mai multe informații cu privire la activitățile și angajații lor, apoi folosesc aceste informații pentru a descrie și înțelege rezultatele obținute. Simplu spus, organizațiile încearcă să descrie și să înțeleagă trecutul. În cadrul acestei abordări, întrebările de interes sunt:

- Ce s-a întâmplat?
- Când s-a întâmplat?
- De ce s-a întâmplat?,

iar uneltele folosite pentru a răspunde la acestea iau forme precum: OLAP (online analytical processing) și dashboards (raport vizual cu privire la o serie de indicatori). Cunoașterea este produsă cel mai adesea manual, foarte puțini pași fiind automatizați. Abordările de acest tip formează ceea ce în literatura de specialitate se numește „Business Intelligence".

Orientarea spre trecut este necesară, dar nu și suficientă pentru a construi o organizație performantă. Aceasta se întâmplă pentru simplul motiv că o parte a întrebărilor importante nu sunt puse, deci nici nu primesc un răspuns. Adesea, ne dorim să aflăm răspunsuri la întrebări precum:[11]

- Ce se va întâmpla în continuare?
- În acest moment, care sunt variantele posibile de urmate, cât de bună este fiecare, ce rezultate este mai probabil să obțin dacă aleg o anumită variantă?
- Care sunt lucrurile la care ar trebui să fiu atent, să le planific, respectiv să mă pregătesc?

Toate aceste întrebări au în comun faptul că vizează viitorul (focusul e viitorul și nu trecutul). Răspunsurile la astfel de întrebări încearcă să anticipeze cum vor arăta cel mai probabil realitățile viitoare, pentru a putea identifica apoi căile prin care putem ajunge la realitatea preferată. Analizele sunt cel mai adese automatizate, rezultatele lor fiind disponibile în timp real. Se urmărește astfel trecerea dintr-o sferă acțională reactivă în una proactivă. Abordările de tip proactiv sunt grupate sub termenul de „Advanced

[11] Mierswa, Ingo. 2014. Moving from "What Happened?" to "What Happens Next?" (https://rapidminer.com/blog/moving-happened-happens-next/).

Analytics". O comparație sistematică a celor două abordări este prezentată în Figura 1.2-7.

Figura 1.2-7. Business Intelligence vs. Advanced Analytics

	Business Intelligence	Advanced Analytics
Orientation	Rearview	Future
Types of questions	What happened When, who, how many	What will happen? What will happen if we change this one thing? What's next?
Methods	Reporting (KPIs, metrics) Automated Monitoring/Alerting (thresholds) Dashboards Scorecards OLAP (Cubes, Slice & Dice, Drilling) Ad hoc query	Predictive Modeling Data Mining Text Mining Multimedia Mining Descriptive Modeling Statistical / Quantitative Analysis Simulation & Optimization
Big Data	Yes	Yes
Data types	Structured, some unstructured	Structured and Unstructured
Knowledge Generation	Manual	Automatic
Users	Business Users	Data scientists, Business analysts, IT, Business Users
Business Initiatives	Reactive	Proactive

Sursa: RapidMiner. An Introduction to Advanced Analytics.
https://rapidminer.com/wp-content/uploads/2014/04/advanced-analytics-introduction.pdf

1.3. Structura și logica manualului

Manualul acoperă pașii 3-6 din modelul CRISP-DM, adică de la pregătirea datelor pentru analiză până la implementarea în producție a unui model de data mining. Manualul va avea mai multe volume, primul dintre acestea, cel de față, fiind destinat prezentării pasului 3, pregătirea datelor pentru analiză.

Pentru fiecare temă vom prezenta principalii operatori, vom discuta principalele opțiuni asociate acestora, respectiv vom ilustra cu imagini statice și dinamice utilizarea operatorului în cadrul unui proces. Pentru a facilita reluarea analizelor prezentate, oferim fișierele cu comenzile și seturile de date utilizate în cadrul tuturor exemplelor.

Cum poate fi folosit acest manual?

Cei interesați de analiza de data mining ar trebui să înceapă procesul de învățare cu citirea atentă a textului, a exemplelor comentate, în paralel cu rularea analizelor, de preferat în ordinea prezentării lor. Ordinea e importantă deoarece termenii de specialitate specifici (limbajul RapidMiner) sunt discutați mai degrabă în prima parte a manualului, apoi doar îi folosim, fără să mai revenim asupra lor. Similar, modul în care performăm diferite acțiuni este descris cu detalii doar în cazul primelor procese prezentate, cele mai simple.

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. Toate deciziile și comenzile ar trebui 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 manual?

În mare măsură, 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", „Comunicare, Societate și Mass-Media" și „Cercetare Sociologică Avansată". În consecință și prin extensie, acest manual 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 în primul volum

În capitolul introductiv argumentăm nevoia și utilitatea unui astfel de manual, discutăm comparativ conceptele majore, prezentăm structura, logica și resursele asociate primului volum al manualului. În capitolul 2, „O lume a datelor", prezentăm câteva concepte majore legate de date: tipuri de date, set de date, tabel de date, baze de date, sisteme de management a bazelor de date și tipuri de fișiere pentru Big Data. În capitolul 3, „Programul RapidMiner Studio: Data mining pe înțelesul tuturor", introducem softul RapidMiner Studio și descriem componentele majore ale acestuia (perspectivele și panelurile). În capitolul 4, „Accesarea datelor (Data Access)", ilustrăm

diferite modalități în care putem lucra cu diferite tipuri de date și aplicații în RapidMiner Studio. Capitolul 5, „Lucrul cu atribute, cazuri, tabele și valori (Blending)", prezintă, în succesiune logică, operatorii care fac posibile acțiuni asupra variabilelor, cazurilor, tabelelor și valorilor. Capitolul 6, „<<Curățarea>> și transformarea datelor (Cleansing)", este dedicat descrierii următoarelor teme: normalizarea datelor, gruparea valorilor, valorile lipsă (missing values), valorile neobișnuite (outlierii) și reducerea dimensionalității datelor. În capitolul 7, „Utilitare (Utility)", prezentăm câțiva operatori mai importanți care au în comun faptul că ușurează diferiți pași ai procesului de pregătire a datelor. În ultimul capitol (8), „Pregătirea asistată a datelor (Turbo Prep)", ilustrăm modul în care procesul de pregătire a datelor poate fi automatizat.

1.4. Resursele asociate manualului

Aproape toate comenzile descrise în acest manual sunt ilustrate și practic, cu ajutorul programului RapidMiner. Toate datele (seturi și baze de date, conexiuni) și procesele asociate acestor exemple sunt disponibile direct tuturor celor interesați. Folderul (Depozitul de date / Repository) care conține toate aceste materiale poate fi descărcat de la adresa:
https://storage.rcs-rds.ro/links/e54d61b1-f684-4e08-490e-3cc0824d6524

parola de acces: 848005

Ca resursă generală de învățare, un bun punct de plecare îl constituie site-ul companiei care a produs softul RapidMiner Studio (https://rapidminer.com/). Aici putem accesa 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.

Manualele RapidMiner Studio

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 majoră de informații pentru scrierea acestui manual.

Rapoartele, studiile de caz, blogul și comunitatea RapidMiner

Pentru nevoi relativ mai specifice de informare, foarte utile sunt rapoartele (Figura 1.4-1), studiile de caz (Figura 1.4-2) și blogul RapidMiner (Figura 1.4-3). Temele discutate aici sunt extrem de diverse, oferă sfaturi și împărtășesc experiențe ale unor practicieni ai domeniului data mining.

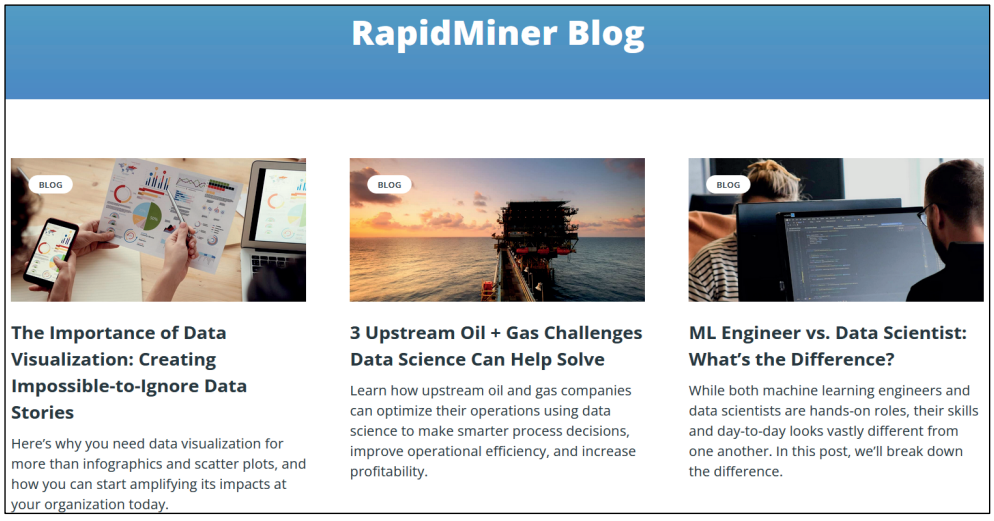
Figura 1.4-1. Site-ul RapidMiner: Rapoarte

Sursa: https://rapidminer.com/reports-tools/

Figura 1.4-2. Site-ul RapidMiner: Studii de caz

Sursa: https://rapidminer.com/case-studies/

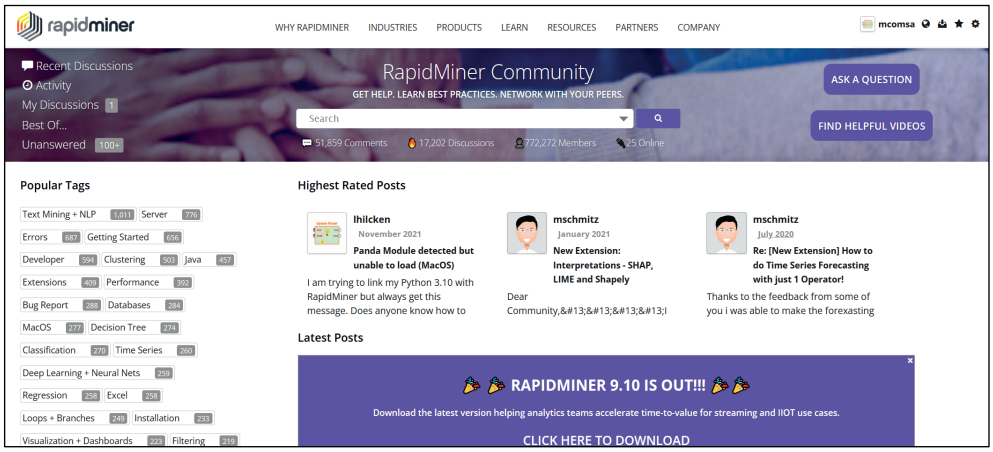
Figura 1.4-3. Site-ul RapidMiner: Blog



Sursa: https://rapidminer.com/blog/

Secțiunea RapidMiner Community (Figura 1.4-4) 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.

Figura 1.4-4. RapidMiner Community

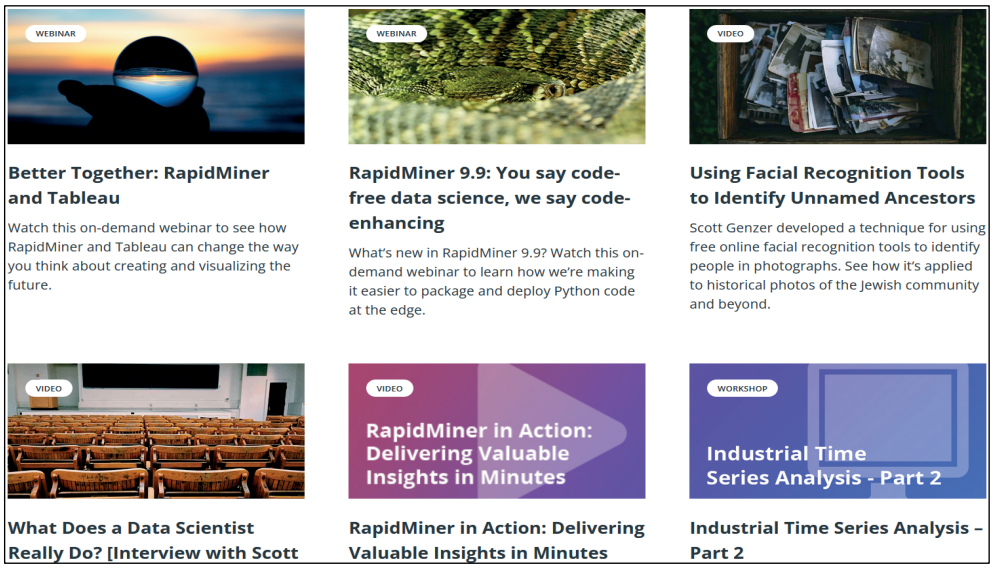


Sursa: https://community.rapidminer.com/

Webinariile și videourile RapidMiner

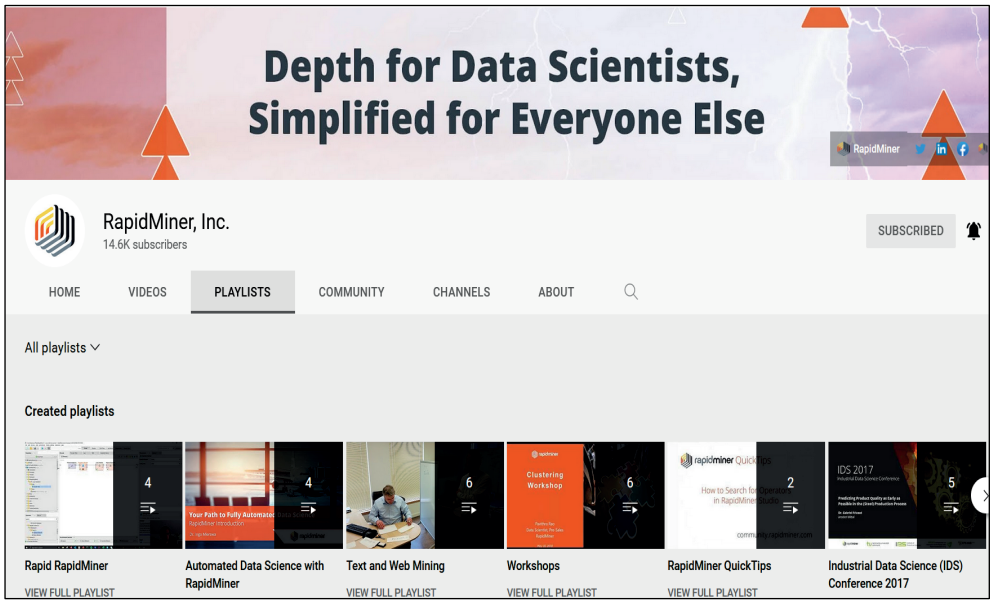
Pentru a accesa diferite prezentări video putem apela la site-ul RapidMiner (Figura 1.4-5) sau la pagina RapidMiner de pe YouTube (Figura 1.4-6). Dintre toate prezentările video de pe YouTube, foarte util pentru faza de început a învățării este playlist-ul „Getting Started with RapidMiner” (Figura 1.4-7).

Figura 1.4-5. Site-ul RapidMiner (secțiunea webinars & videos)



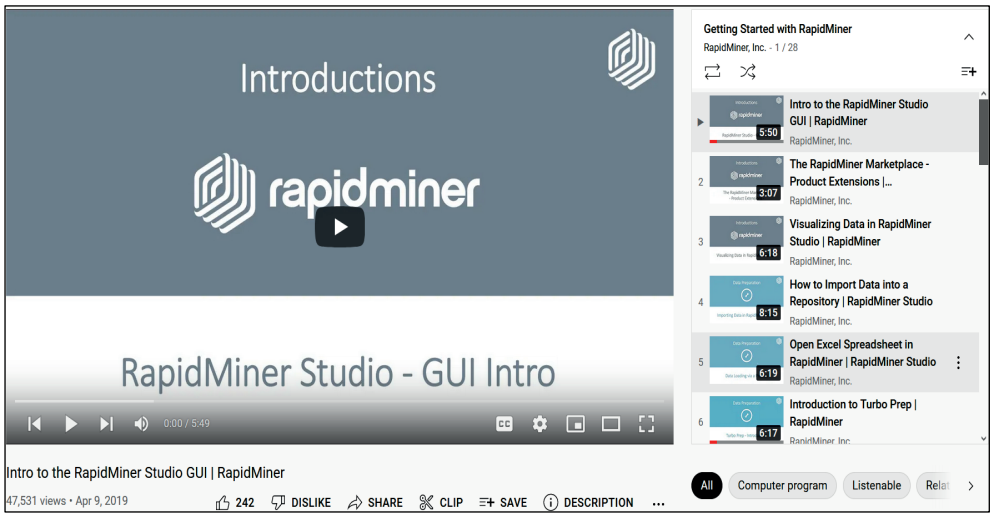
Sursa: https://rapidminer.com/webinars-videos/

Figura 1.4-6. Pagina RapidMiner pe YouTube



Sursa: https://www.youtube.com/channel/UCxneJBWWNLs-A6ckls1Rrug

Figura 1.4-7. Pagina RapidMiner pe YouTube (Getting Started with RapidMiner)



Sursa: https://www.youtube.com/watch?v=Gg01mmR3j-g&list=PLssWC2d9JhOZLbQNZ80uOxLypglgWqbJA)

Cărți care folosesc softul RapidMiner Studio

O altă resursă utilă pentru învățare o reprezintă cărțile care folosesc softul RapidMiner Studio pentru a realiza diferite analize punctuale sau proiecte de data mining pe teme specifice. În toate aceste cărți sunt folosite versiuni mai vechi ale RapidMiner Studio (v.5-7; versiunea cea mai recentă la acest moment este 9.1), dar analizele și procesele prezentate pot fi reluate fără probleme în versiunile mai noi. Câteva astfel de cărți, în ordinea publicării, sunt menționate în Tabelul 1.4-1.

Tabelul 1.4-1. Cărți în care este prezentat / folosit softul RapidMiner Studio

Cartea și autorii / coordonatorii	Tip carte	Tematică nivel	Nivel
Exploring Data with RapidMiner (Chisholm, 2013)	autor	generală, teme multiple	introductiv
Predictive analytics and data mining: concepts and practice with RapidMiner (Kotu & Deshpande, 2015)	autor	generală, teme multiple	introductiv, mediu
RapidMiner. Data Mining Use Cases and Business Analytics Applications. (Hofmann & Klinkenberg, 2016)	coordonare	specifică, teme și domenii multiple	mediu
Data Mining for the Masses, with Implementations in RapidMiner and R (North, 2018)	autor	generală, teme multiple	introductiv, mediu
Data science: concepts and practice (Kotu & Deshpande, 2019)	autor	generală, teme multiple	introductiv, mediu

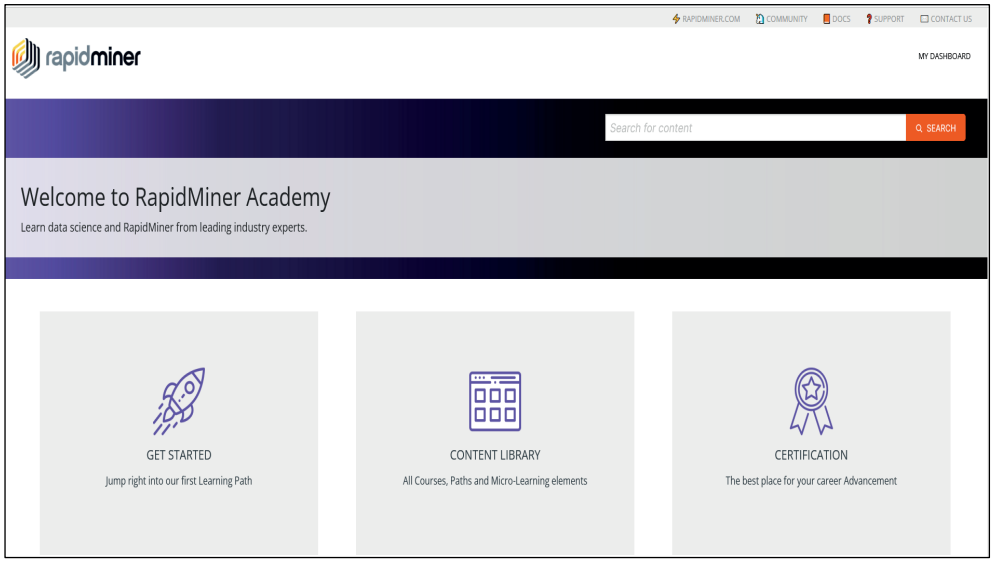
Probabil cea mai completă și bine organizată resursă pentru învățare de pe site-ul RapidMiner este secțiunea RapidMiner Academy, motiv pentru care îi acordăm o prezentare mai extinsă.

1.5. RapidMiner Academy

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 și la adresa https://rapidminer.com/blog/academy-data-science-training/.

Figura 1.5-1. RapidMiner Academy

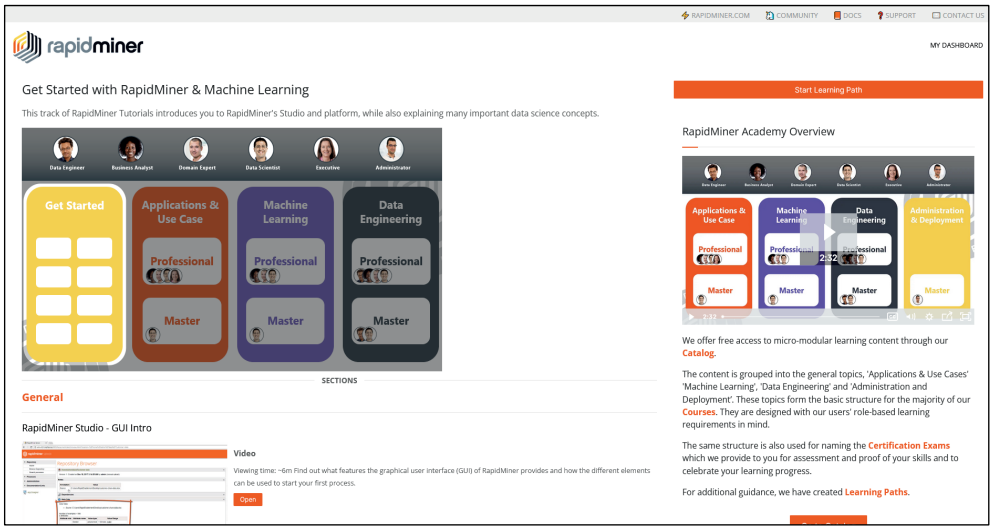


Sursa: https://academy.rapidminer.com/

Cursul „Get Started" și direcțiile de specializare

Secțiunea „Get Started" (Figura 1.5-2) trebuie parcursă de către toți cei care sunt în faza de familiarizare cu softul. Pe parcursul secțiunii sunt abordate teme generale (interfața RapidMiner, modalități de vizualizare a datelor, instalarea extensiilor), accesarea și pregătirea datelor pentru analiză (în cadrul unui proces, respectiv prin intermediul perspectivei TurboPrep), realizarea, validarea și testarea unor modele predictive (în cadrul unui proces, respectiv prin intermediul perspectivei AutoModel), implementarea și managementul modelelor. Pentru fiecare temă este realizată o prezentare video și poate fi descărcat procesul RapidMiner aferent, respectiv datele utilizate (dacă nu sunt deja incluse în soft).

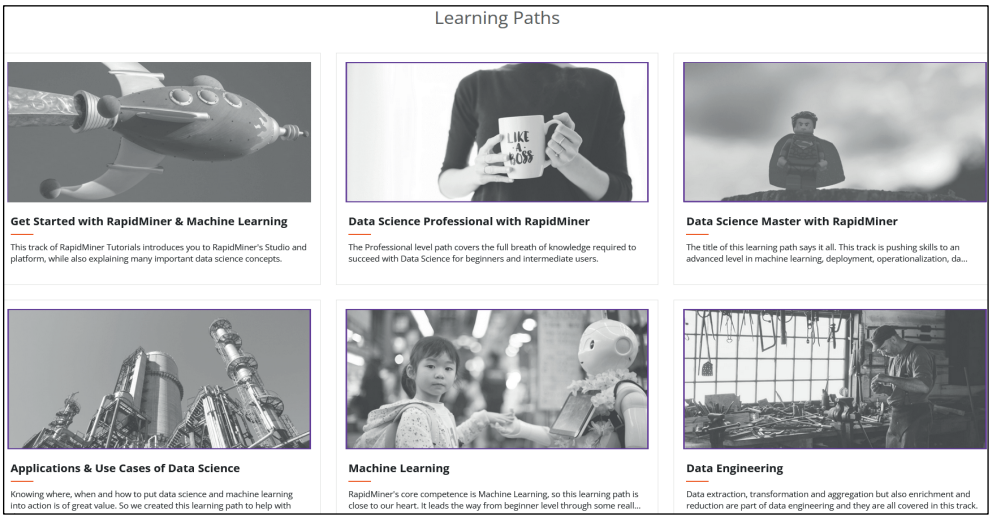
Figura 1.5-2. RapidMiner Academy: Get Started



Sursa: https://academy.rapidminer.com/learning-paths/get-started-with-rapidminer-and-machine-learning

Pentru a aprofunda într-un mod ghidat cunoașterea, se poate accesa secțiunea dorită din ghidul „Learning Paths" (Figura 1.5-3). Secțiunile disponibile sunt: Get Started with RapidMiner & Machine Learning, Data Science Professional with RapidMiner, Data Science Master with RapidMiner, Applications & Use Cases of Data Science, Machine Learning, Data Engineering.

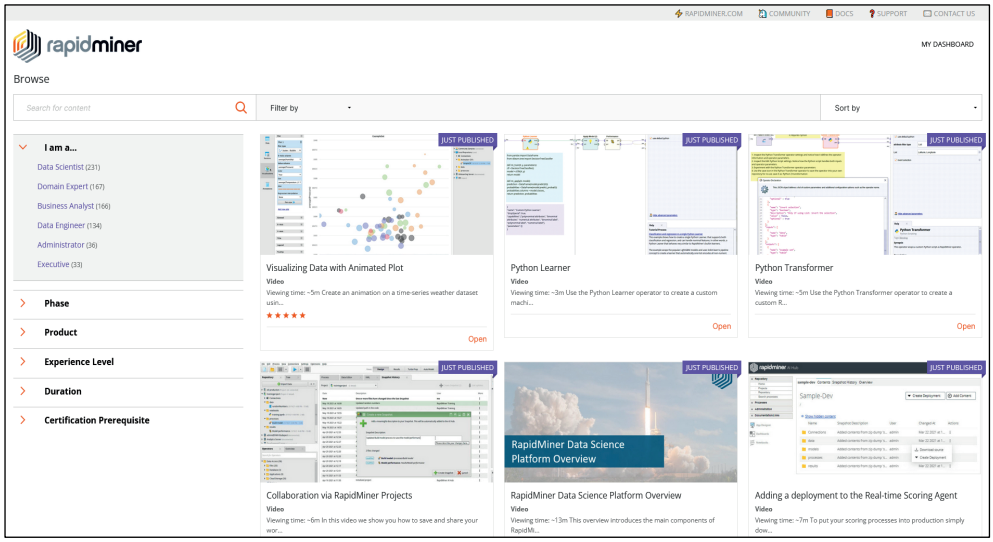
Figura 1.5-3. RapidMiner Academy: Learning Paths



Sursa: https://academy.rapidminer.com/pages/content-library

Resursele disponibile pot fi accesate și prin căutarea lor în Catalog (Figura 1.5-4). Căutarea se poate realiza în funcție de criterii precum poziția ocupată sau dorită, etapa de analiză, produs (RapidMiner Studio sau alt soft), nivelul de experiență, durată și tipul de certificat dorit.

Figura 1.5-4. RapidMiner Academy: Catalog



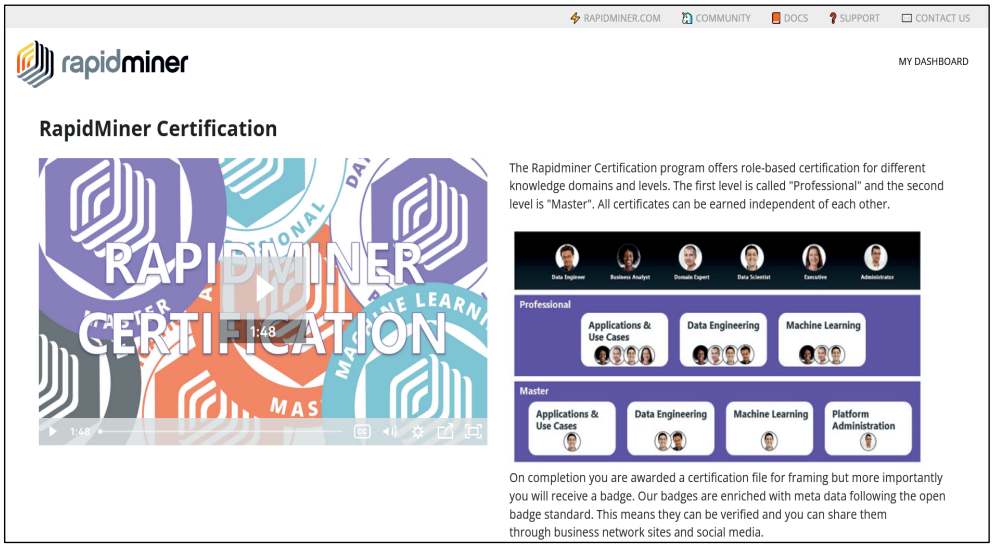
Sursa: https://academy.rapidminer.com/catalog

Pregătirea pentru examene și certificarea

Certificatele RapidMiner sunt organizate pe două nivele: Professional și Master. Nivelul Professional include trei domenii: „Applications & Use Cases", „Data Engineering" și „Machine Learning" (Figura 1.5-5). Nivelul Master include patru domenii: „Applications & Use Cases", „Data Engineering", „Machine Learning" și „Platform Administration". Fiecare certificare poate fi obținută independent de oricare alta. Pe aceeași pagină web pot fi accesate cursurile de certificare. Pentru fiecare curs (Figura 1.5-6) sunt oferite diferite resurse, organizate tematic, iar progresul este vizualizat cu ajutorul unei bare de status. Suplimentar, se oferă un spațiu pentru notițe și unul unde se pot pune întrebări. Fiecare curs poate fi finalizat cu o examinare online care durează aproximativ o oră (20 întrebări care trebuie

completate în ordine, fără posibilitatea de revenire) (Figura 1.5-7). Un examen poate fi dat de mai multe ori (Figura 1.5-8).

Figura 1.5-5. RapidMiner Academy: Certification

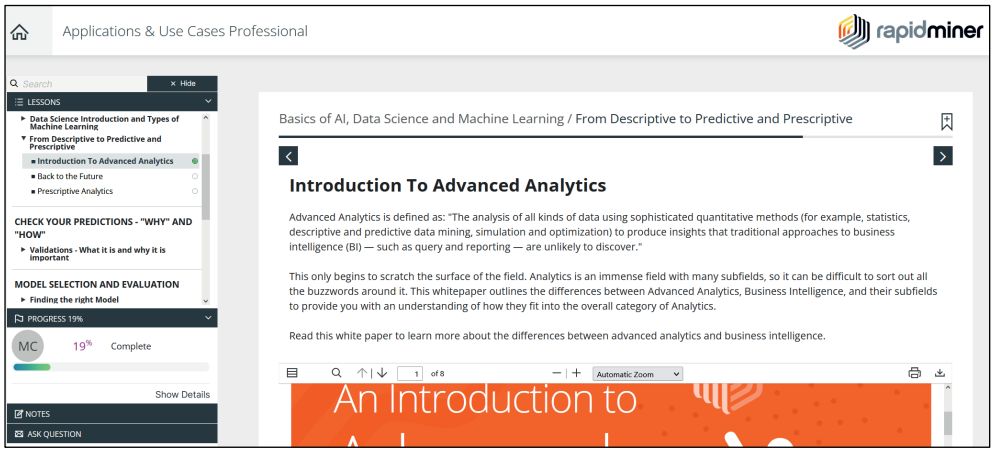


Sursa: https://academy.rapidminer.com/pages/certification

Figura 1.5-6. RapidMiner Academy: Courses for Exam Preparation

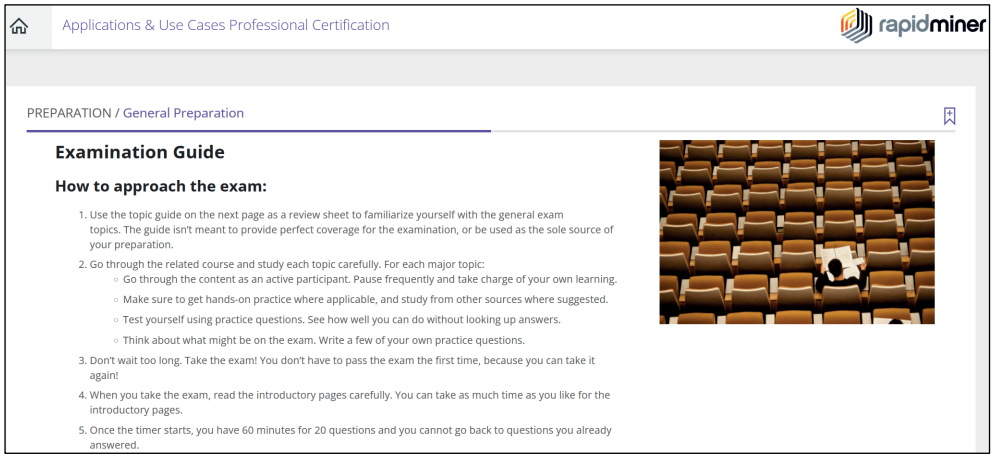
Sursa: https://academy.rapidminer.com/pages/certification

Figura 1.5-7. Extract din pagina unui curs RapidMiner



Sursa: https://academy.rapidminer.com/learn/course/applications-use-cases-professional/basics-of-ai-data-science-and-machine-learning/from-descriptive-to-predictive-and-prescriptive

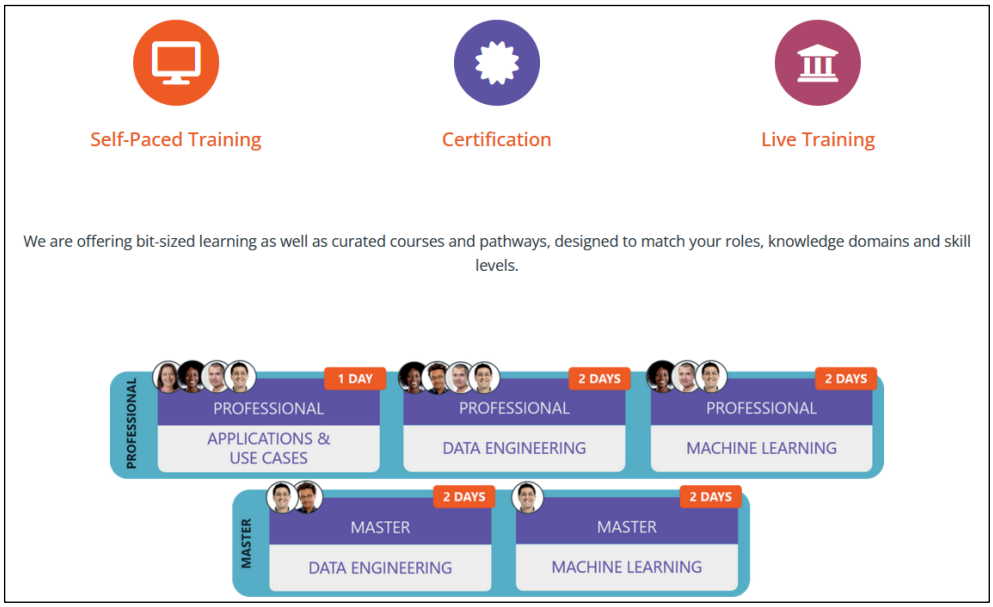
Figura 1.5-8. Extract din Ghidul de Examinare RapidMiner



Sursa: https://academy.rapidminer.com/learn/course/applications-use-cases-professional-certification/preparation/general-preparation

Resursele oferite la training și certificare pot fi accesate și în cadrul unei secțiuni separate dedicate training-ului (Figura 1.5-9). Aici avem sub-secțiunile Self-Paced Training (ne direcționează la RapidMiner Academy), Certification (ne direcționează la RapidMiner Academy: Certification) și Live Training (conține o serie de cursuri plătite, organizate de diferiți specialiști, direct sau prin intermediul unei instituții).

Figura 1.5-9. RapidMiner Training



Sursa: https://rapidminer.com/learn/training/

2. O LUME A DATELOR

Mai mult decât oricând, lumea de azi este o lume a datelor. Multe dintre activitățile realizate de oameni (direct sau indirect, prin intermediul mașinilor) produc date, o tot mai mare parte a acestora fiind stocate și analizate. Creșterea volumului datelor din societate actuală este în mare măsură și rezultatul multiplicării exponențiale a surselor care produc continuu date (datafication): largi colecții de documente care sunt digitalizate, social media, aplicații web, motoare de căutare, comerț electronic, platforme profesionale, platforme de crowdsourcing, colecții de informații cu privire la schimburile social-economice și interacțiunile dintre oameni, locuri și organizații, sateliți, drone, camere de supraveghere, senzori, dispozitive, aparate și servicii de uz cotidian interconectate (IoT = Internet of Things, adică Internetul Obiectelor) etc. Tot mai multe aspecte ale vieții oamenilor sunt digitalizate. Deoarece costurile asociate acestui proces continuă să scadă, e foarte probabil ca tendința să continue.

În acest capitol vom discuta despre patru teme majore legate de date. Începem discuția cu câteva statistici sugestive, statistici ce ilustrează o parte dintre provocările majore legate de date, și anume volumul, velocitatea și varietatea acestora. Apoi, prezentăm o clasificare a tipurilor de date (structurate, semistructurate și nestructurate) și descriem pe scurt fiecare tip. Înainte de a analiza datele, e necesar ca acestea să fie stocate, să poată fi accesate ușor și rapid, respectiv să fie integrate (mai ales atunci când sunt de tipuri foarte diferite). Ultimele două teme ale capitolului abordează tocmai aceste aspecte. Începem discuția cu o prezentare a conceptului de bază de date și a sistemului de management al unei baze de date. La final prezentăm și comparăm câteva formate de fișiere utilizate în cazul Big Data.

2.1. Volumul, velocitatea și varietatea datelor

Numărul de utilizatori ai unei tehnologii utilizate pentru a produce și schimba date este enorm și în creștere rapidă. Același lucru se întâmplă și în cazul obiectelor conectate, respectiv a companiile private și a instituțiilor guvernamentale. Astfel, conform Statista, la nivelul anului 2021, în lume erau

- 5,3 miliarde utilizatori unici de telefonie mobilă (mai mult de două treimi din populația globului),
- 4,7 miliarde utilizatori activi ai Internetului (aproape 60% din populația globului),
- 4,3 miliarde utilizatori ai platformelor de social media.
- Potrivit aceleiași surse, în 2021, existau pe glob
- 14,9 miliarde telefoane mobile,
- 10,1 miliarde dispozitive conectate la Internet (IoT).

Zilnic, în lume sunt produse și consumate o mulțime de date online. Conform site-ului https://www.internetlivestats.com/, în lume există la ora actuală un număr de ordinul miliardelor de

- website-uri (1.95),
- utilizatori activi Facebook (3,1),
- utilizatori activi Google+ (1,1),
- utilizatori activi Twitter (0,4).

Aceeași sursă estimează că azi (2022.05.13)

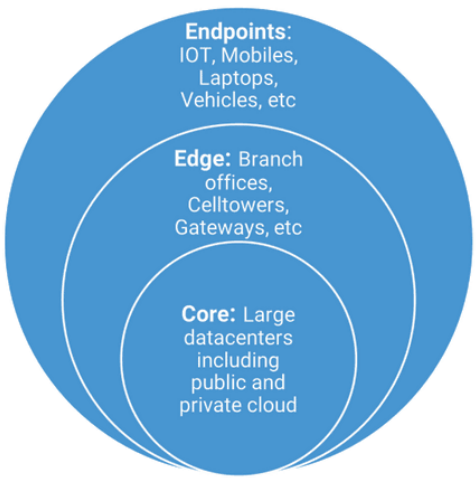
- au fost trimise 121 miliarde emailuri,
- realizate 3,8 miliarde căutări pe Google,
- vizualizate 3,5 milioane videouri pe YouTube,
- scrise 3,7 milioane postări pe bloguri,
- făcute 370 milioane postări pe Twitter,
- postate 43 milioane fotografii noi pe Instagram.

Firesc, date fiind estimările prezentate anterior, volumul de date produs este enorm și cu o rată de creștere accelerată. Conform raportului IDC, Global

DataSphere Forecast[12], volumul de date produse în 2021 la nivelul globului a fost de 79 zetabiți[13], urmând să ajungă la 181 zetabiți în 2025 (creștere anuală de 23%). Pentru același interval, capacitatea de stocare va crește de la 10 la 16 zetabiți (nu toate datele produse sunt și stocate).

Creșterea volumului datelor este însoțită de schimbări majore la nivelul naturii și locației datelor produse. Pe lângă datele structurate (relaționale și tranzacționale) stocate în baze de date (SQL), a crescut și va continua să crească volumul datelor nestructurate. Conform aceluiași raport, în 2025, 80% din date vor fi nestructurate. În loc să fie stocate în locații fixe și cunoscute, ușor de controlat și gestionat, aceste date vor fi răspândite peste tot. O reprezentare sugestivă a locațiilor și volumelor stocate în acestea apare în Figura 2.1-1. Categoria „endpoints" include toate dispozitivele precum telefoanele mobile, senzorii și IoT (obiectele conectate: casnice, dispozitive de monitorizare personală, mașini, computere etc.). Categoria „edge" include datele din serverele companiilor din birouri și mici centre de date. Categoria „core" include centrele de date ale companiilor și centrele globale din cloud.

Figura 2.1-1. Locația datelor și volumul acestora



Sursa: Hack, Ulrike. 2021. What's the real story behind the explosive growth of data?
https://www.red-gate.com/blog/database-development/whats-the-real-story-behind-the-explosive-growth-of-data

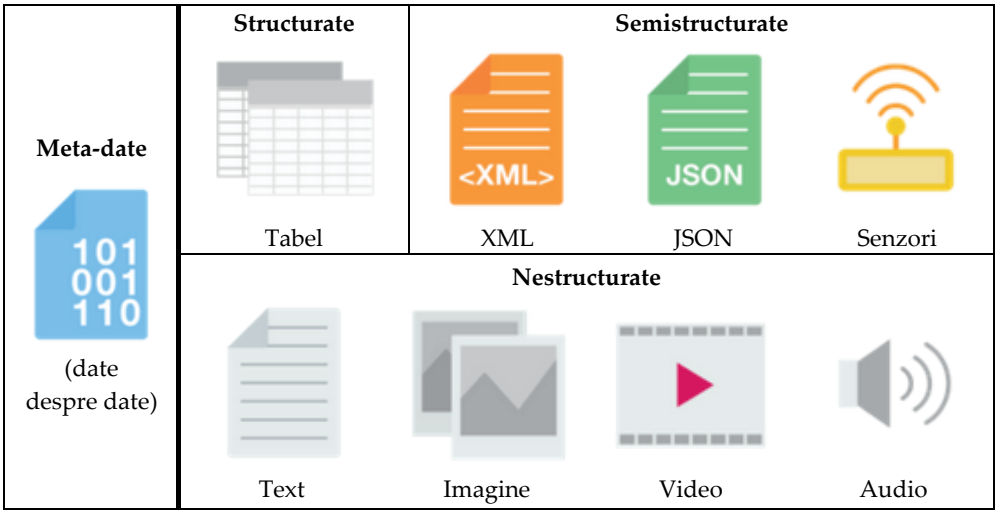
[12] Reinsel, David. Rydning, John & Gantz, F. John. 2021. Worldwide Global DataSphere Forecast, 2021-2025: The World Keeps Creating More Data - Now, What Do We Do with It All?

[13] Un zetabit este egal cu 10^{21} biți sau aproximativ 250 miliarde de DVD-uri.

2.2. Tipuri de date

În relație cu domeniul IT, datele sunt unități de informație formatate și păstrate sub formă de fișiere în conformitate cu scopuri specifice (pentru a fi redate / citite, modificate / actualizate, analizate, transformate, transferate etc.) Datele pot fi grupate în funcție de diferite criterii, cele mai des utilizat fiind gradul de structurare. În funcție de acest criteriu distingem între date structurate, semistructurate și nestructurate (Figura 2.2-1).

Figura 2.2-1. O tipologie a datelor



Datele de tip structurat sunt cele care iau forma unor tabele / structuri matriceale sau, mai simplu spus o serie de linii și coloane. De obicei, într-un astfel de tabel, cazurile, numite și exemple sau observații, sunt poziționate pe linii, iar informațiile asociate acestora pe coloane (Figura 2.2-2). Coloanele sunt denumite cel mai adesea atribute / variabile / caracteristici / trăsături (features) / înregistrări (Mierswa, 2016b, p. 11). În zona științelor sociale se folosesc mai degrabă termenii de variabilă și atribut, respectiv caz sau observație. Structura unui tabel, mai exact informațiile care apar pe coloane, este stabilită înainte de colectarea datelor (formatul este pre-definit). Adesea, nu doar formatul tabelar este pre-definit ci și categoriile / variantele de

răspuns / înregistrare a informațiilor colectate. Pentru fiecare atribut se specifică în avans și formatul în care se înregistrează datele. Formatele de date care apar într-un astfel de tabel sunt cel mai adesea de tip nominal / categorial și numeric și mai rar de tip dată sau text nestructurat (caracter / string). În acest exemplu, datele afișate ca text („masculin, „liceu" sau „muncitor") sunt de fapt nominale / categoriale. Ele pot fi ușor codate ca variabile / atribute cu coduri numerice (de exemplu, 1=muncitor, 2=maistru, 3=inginer, 4=director).

Figura 2.2-2. Formatul de date structurat de tip tabelar

Atributul 1 Id	Atributul 2 Sex	Atributul 3 Vârstă	Atributul 4 Educație	Atributul 5 Vechime	Atributul 6 Poziție
Caz 1	masculin	18	liceu	2	muncitor
Caz 2	masculin	45	post-liceală	3	maistru
Caz 3	feminin	64	facultate	1	inginer
Caz 4	feminin	30	doctorat	5	director

Datele de tip semistructurat pot avea o structură internă și diferite marcaje (tags) cu ajutorul cărora sunt identificate și separate elementele componente, plus relația dintre ele (poate fi ierarhică). Schema respectivă nu constrânge tipul de informație care este stocată. Un exemplu tipic de date de tip semistructurat este emailul. Un email conține câmpuri (atribute) precum emailul destinatarului, al expeditorului, data și subiectul mesajului sau corpul mesajului. Toate aceste elemente componente structurează parțial un email, în sensul că unele câmpuri sunt standardizate (de exemplu, în câmpul corespunzător datei este obligatoriu să apară data în format standardizat de tip dată, câmpurile cu adrese de email acceptă doar un format standard de tip string care necesită prezența caracterului @ înainte de numele domeniului), pe când alte câmpuri sunt nereglementate (de exemplu, în corpul mesajului, forma textului nu are limitări stricte). De asemenea, toate emailurile au același format (aceleași câmpuri, aceeași structură generală), indiferent cine le expediază sau cine le primește.

Toate datele care nu intră în una dintre cele două categorii prezentate anterior sunt de tip nestructurat. De obicei iau forma unui text mai lung care poate include sau nu și alte tipuri de informații sub forma unor imagini, filmulețe

sau mesaje audio. Acest tip de date este relativ mai dificil de stocat, citit și analizat de către programele tradiționale. Majoritatea datelor produse în prezent sunt de tip nestructurat, prin urmare e foarte util să știm să le stocăm și mai ales să extragem din ele informațiile relevante pentru scopurile noastre. În ultimii ani, tehnologia potrivită pentru stocarea și analiza datelor nestructurate a avansat foarte mult.

Datele sunt păstrate (stocate) în computere sau unități externe, respectiv transportate de la un computer la altul, sub formă de fișiere. Există o multitudine de tipuri (formate) de fișiere, create pentru a înmagazina tipuri specifice de date. Unele formate de fișiere au ca scop stocarea imaginilor, altele a textelor, altele a instrucțiunilor pe care să le execute procesorul unui computer, altele a sunetelor ș.a.m.d. Așadar, ținând cont de specificitatea tipurilor de fișiere, formatul de fișier poate fi folosit drept criteriu de clasificare a datelor. Există o suprapunere foarte mare între clasele rezultate în urma aplicării criteriului gradul de structurare și formatul de fișier. Simplu spus, datele cu același grad de structurare sunt stocate în general în aceleași formate de fișiere. Astfel, datele structurate pot fi stocate sub forma unor fișiere de tip:

- **text** (delimited text file): fiecare caz apare pe o linie iar atributele (coloanele) sunt separate cu ajutorul unor caractere specifice; formatele cele mai des utilizate sunt **csv** (comma-separated file) și **tsv** (tab-separated file);
- **xlsx**: tabel Microsoft Excel;
- **sav**: set de date SPSS;
- **dta**: set de date Stata;
- **sas7bdat**: set de date SAS;
- **rdata**: set de date R;
- **rmhdf5table**: set de date RapidMiner;
- etc.

Formatele în care apar cel mai adesea datele semistructurate sunt JSON (JavaScript Object Notation) și XML (eXtensible Markup Language). Formatul JSON reprezintă datele sub forma unei structuri de perechi nume + valoare. Valorile pot fi de tip text (string), numeric, boolean (true/false), null (marchează absența intenționată a unei valori), matrice (array) sau chiar alte

obiecte de tip JSON (inclusiv matrice de obiecte care pot conține alte matrice care la rândul lor pot conține obiecte, ș.a.m.d.). Folosind câteva reguli simple de marcare, formatul XML este utilizat pentru a codifica documentele într-o formă ce poate fi citită atât de către oameni cât și de programe. Pentru a stoca și reprezenta datele, ambele limbaje de marcare folosesc formatul text (codare Unicode). De asemenea, ambele formate pot reprezenta date situate la nivele multiple (nested). Reprezentarea datelor în formatul JSON este mai puțin redundantă. Un scurt exemplu de codificare a acelorași date în cele două formate este prezentat mai jos (Tabelul 2.2-1).

Tabelul 2.2-1. Formatele de date structurate XML și JSON

XML:	JSON:

<angajat>
<nume> xulescu </nume>
<vechime> 10 </vechime>
<varsta> 44 </varsta>
<educatie> master </educatie>
</angajat>
	{
	"nume": "xulescu",
	"vechime": 10,
	"varsta": 44,
	"educatie": "master"
	}

Datele nestructurate pot fi stocate în diferite formate, funcție de tipul lor. Datele nestructurate de tip textual pot fi păstrate în fișiere de tip txt, docx, rtf, odt, pdf etc., datele de tip imagine apar sub forma unor fișiere de tip jpg, gif, png, tiff etc., cele video în fișiere de tip avi, mkv, mov, mp4 etc., iar cele audio în fișiere de tip mp3, wav etc.

Un criteriu de clasificare utilizat relativ mai rar, dar probabil tot mai important este autoreferențialitatea. Mai exact, distingem între date care se referă la fenomenul de interes în sine și date care se referă la datele despre fenomen. Simplu spus, datele se pot referi sau nu la alte date. În literatura de specialitate, datele relativ la alte date se numesc meta-date. De exemplu, cu privire la datele colectate în cadrul unei anchete, setul de date rezultat poate avea asociate diferite alte informații de interes precum: perioada colectării, tipul de anchetă, operatorul care a colectat datele, caracteristicile socio-demografice ale acestuia, tipul de întrebare, durata de aplicare a întrebării, timpul scurs între două întrebări, durata totală de aplicare a chestionarului etc. În relație cu o imagine, meta-datele se pot referi la coordonatele

geografice asociate acesteia, data și ora la care imaginea a fost realizată, rezoluția, tipul de aparat folosit și caracteristicile acestuia etc.

În RapidMiner Studio putem lucra cu toate aceste tipuri de fișiere. Trebuie doar să identificăm formatul fișierului, operatorul aferent, apoi să comparăm avantajele și limitele diferitelor tipuri de fișiere relativ situația concretă în care ne aflăm, scopul procesului: realizarea unei analize în RapidMiner, stocarea datelor, transportarea și/sau transferul lor în alte formate / sisteme.

2.3. Baze de date și sisteme de management al bazelor de date

Set de date (dataset) și tabel de date (datatable)

Terminologia utilizată de diferite discipline pentru a se referi la aceleași concepte este uneori diferită. De exemplu, în științele sociale, conceptul de bază de date (database) este folosit uneori interșanjabil cu cel de fișier / set de date (datafile / dataset) și se referă la ceea ce specialiștii din alte domenii precum IT sau analiza datelor (data mining) numesc tabele de date (table sau dataframe). Aceștia din urmă, folosesc conceptul de bază de date pentru a se referi la unul sau mai multe tabele de date relaționale (legate între ele prin una sau mai multe variabile cheie), la stocări de tip obiecte, depozite de documente, grafuri sau alte tipuri de reprezentare a datelor.

O bază de date este găzduită de un server de date (poate fi și unul local), conține cel mai adesea mai multe fișiere, formate și compresii specifice (softurile au nevoie de drivere pentru conectare, interacțiune cu datele), au sintaxe de interogare specifice, iar volumele datelor sunt foarte mari. Fișierele de date au volume relativ mai mici de date, incluse într-un fișier unic, adeseori flat-file, sunt ușor de accesat și interogat de softuri diverse, respectiv ușor de transportat.

Un set de date (dataset) este o colecție de elemente de același tip pentru care dispunem de o serie de informații. Setul de date ia cel mai adesea forma unui

tabel[14] (unei matrice), adică un număr de linii și coloane, cazurile (indivizii statistici sau elementele) aparând de obicei pe linii, iar atributele (variabilele) pe coloane.[15] De obicei, un set de date nu se schimbă în timp și este utilizat pentru analize statistice. O mică parte a unui astfel de set de date este prezentat în Figura 2.3-1.

Figura 2.3-1. Un exemplu de set de date în format tabelar (dataset)

id_angajat	nume_ang	prenume_ang	telefon_ang	email_ang	...	...
1	aaa	bbb	0745xxxxxx	a.b@em.ro	...	...
2	ccc	ddd	0746xxxxxx	c.d@em.ro	...	...
...	...	...	...	...	...	...

În RapidMiner Studio, conceptul de tabel (table) se referă la orice set de informații organizate sub forma de linii și coloane, deci și la un tabel cu rezultatele unei analize. Prin urmare, sensul cu care este folosit conceptul în RapidMiner Studio este unul ceva mai general comparativ cu conceptul de table / dataset descris mai sus.

Strategii de management și analiză a datelor

Tehnologiile de management și analiză a datelor pot fi grupate în patru mari categorii (Tabelul 2.3-1). O primă dimensiune a clasificării distinge între bazele de date și alte tipuri de fișiere (text, foi de calcul, formate specifice diferitelor softuri de analiză statistică). Bazele de date sunt utile mai ales atunci când volumul datelor este foarte mare. În cazul datelor structurate, în majoritatea situațiilor se folosesc baze de date relaționale (SQL), iar în cazul datelor nestructurate bazele sunt non-relaționale (NoSQL) (acestea din urmă

[14] O perspectivă similară este adoptată și în pachetul de analiză statistică R, un tabel / set de date fiind numit aici dataframe (Dușa et al., 2015, p. 170).

[15] Setul de date poate lua și alte forme decât tabelare. Un astfel de format este json. Un mic exemplu (4 variabile și un caz) de set de date în format json este următorul:
{
"numele": "Pop",
"prenumele": "Ioan",
"vârsta": 47,
"poziția": "manager"
}

pot lua diferite forme, funcție de tipul datelor). Dacă volumul datelor este redus și datele sunt de tip structurat, putem folosi foile de calcul sau diferite programe de analiză statistică.

Tabelul 2.3-1. Tehnologii de management și analiză a datelor

Fișiere text, foi de calcul	**Programe de analiză statistică**
- volum mic de date - analiză simplă - analiza nu va fi repetată	- volum mic/mediu de date - tipul de analiză este potrivit pentru programul statistic ales
Bază de date relațională (SQL)	**Bază de date non-relațională (NoSQL)**
- date structurate - volum mare de date - analiza va fi repetată în timp, pe versiunile noi ale bazei de date - dorim să împărtășim datele și analizele cu alte persoane	- date nestructurate - volum foarte mare de date - analiza datelor se face în principal înafara bazei de date, folosind un limbaj de programare

Sursa: (Foster & Heus, 2021, p. 69)

În domeniul științelor sociale, managementul, analiza și prezentarea datelor, cel mai adesea, sunt realizate folosind softuri statistice precum SPSS, Stata, SAS și R. Pe măsură ce complexitatea și mărimea datelor din societatea actuală crește, o astfel de abordare se confruntă cu multiple probleme. Să considerăm următoarele situații:

Situația	**Date**	**# cazuri**	**# atribute**	**mărime (biți)**
A	mici	10^3	200-300	10^6 (1 MB)
B	medii	10^6	100-200	10^9 (1 GB)
C	mari	10^9	50-100	10^{12} (1 TB)

Care sunt soluțiile posibile pentru managementul și analiza unor astfel de date? Programele de analiză statistică, precum SPSS și Stata, încarcă tot setul de date în memoria de lucru, deci pot lucra fără probleme cu datele din situația A (date mici). Funcție de memoria disponibilă, sunt șanse mari ca aceste programe să nu poată încărca și analiza datele din situația B (sau durata acestor operații să fie foarte mare) și, aproape sigur nu vor putea

încărca datele mari (situația C). În plus, de fiecare dată când vom dori să analizăm aceste date, va trebui să le încărcăm din nou.

Baze de date (databases)

O bază de date (database) este constituită dintr-un set de seturi de date (datasets). Fiecare set este centrat pe o anumită categorie de informații. În general, seturile de date includ una sau mai multe variabile „cheie", de identificare (id). Cu ajutorul acestor variabile putem combina diferite informații din două sau mai multe seturi și produce astfel un nou set de date. De obicei, o bază de date se schimbă în timp (se adaugă cazuri, uneori chiar și noi tipuri de informații) și este utilizată pentru a produce date pentru diferite rapoarte.

Structura unei posibile baze de date relaționale din domeniul resurselor umane este prezentată în Figura 2.3-2. În practică, o astfel de bază de date integrează mult mai multe surse de informații, așa cum rezultă și din Figura 2.3-3.

Figura 2.3-2. Un exemplu de bază de date relațională (relational database)

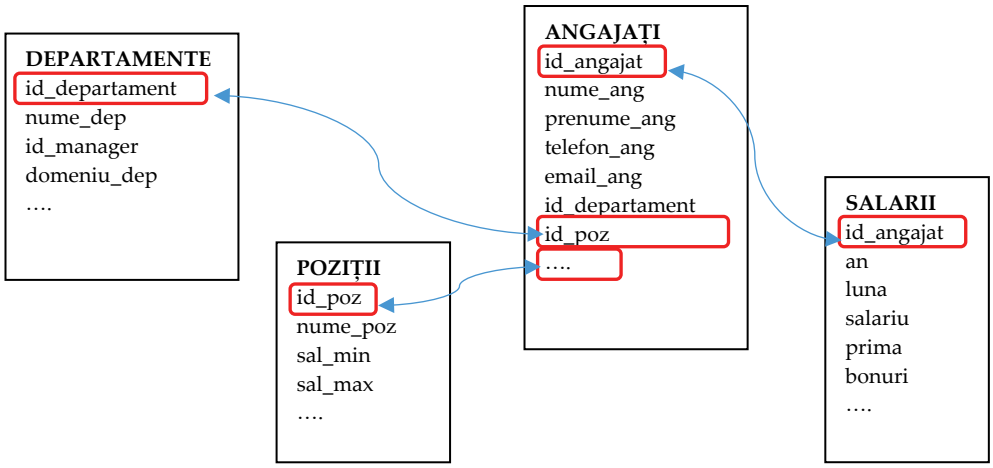
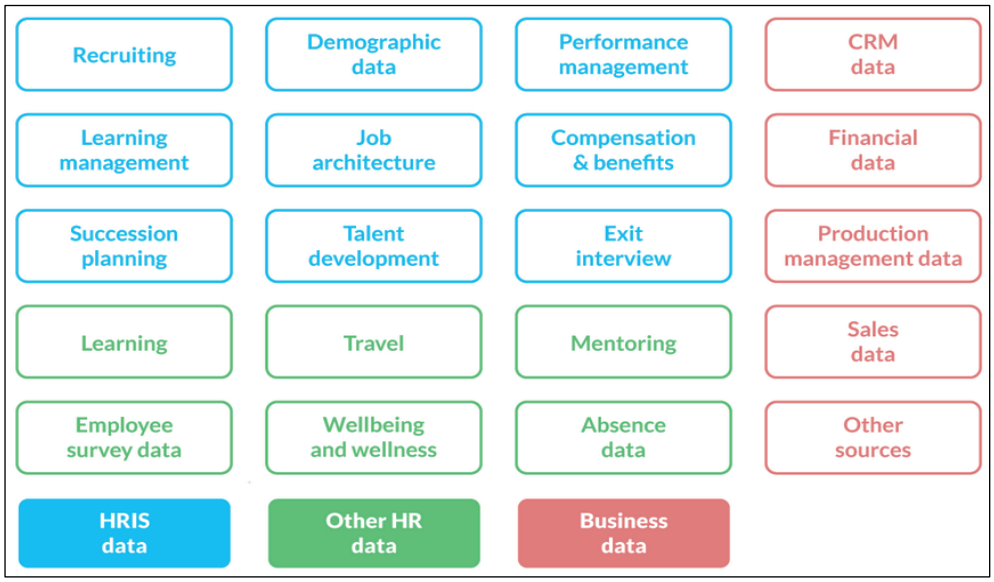


Figura 2.3-3. Structura unei baze de date relaționale din domeniul resurselor umane



Sursa: https://www.aihr.com/blog/hr-data-sources/

Principala dimensiune în funcție de care pot fi clasificate bazele de date constă în utilizarea sau nu a unui limbaj de interogare structurat (Structured Query Language, adică SQL). Bazele care folosesc un astfel de limbaj sunt numite și relaționale sau SQL, iar cele care nu-l folosesc sunt numite non-relaționale sau NoSQL. Fiecare dintre aceste tipuri de baze sunt potrivite într-o măsură mai mare pentru anumite tipuri de situații și sarcini. O schemă sintetică este prezentată în Tabelul 2.3-2.

Tabelul 2.3-2. Tipuri de baze de date: relaționale (SQL) și nerelaționale (NoSQL)

Tip bază	Exemple	Avantaje	Dezavantaje	Utilizări
Relațională	MySQL, PostgreSQL, Oracle, SQL Server, Teradata	ACID (atomizare, consistență, izolare, durabilitate)[16]	Schemă fixă, dificil de scalat	Sisteme tranzacționale: procesarea comenzilor, vânzări, spitale

[16] Atomizarea se referă la faptul că fiecare tranzacție (acțiune / comandă) este tratată ca o unitate singulară care fie reușește complet, fie eșuează complet. Consistența se referă la faptul că o tranzacție modifică baza de date dintr-o stare validă în alta, tot validă. Izolarea se referă la faptul că executarea concurentă a tranzacțiilor produce o stare identică a bazei precum execuția secvențială. Durabilitatea se referă la faptul că o dată ce o tranzacție a fost comisă, ea va rămâne comisă chiar dacă sistemul „cade".

Tip bază	Exemple	Avantaje	Dezavantaje	Utilizări
NoSQL: cheie-valoare	Dynamo, Redis	Schemă dinamică; ușor scalabilă; debit mare al datelor	Consistența nu este imediată; interogarea nivelurilor superioare nu este posibilă	Aplicații Web
NoSQL: coloană	Cassandra, HBase	Similar cu baza cheie-valoare; distribuită; compresie mai bună la nivel de coloane	Consistența nu este imediată; utilizarea tuturor coloanelor este ineficientă	Analiza la scară largă
NoSQL: document	CouchDB, MongoDB	Indexează tot documentul (JSON)	Consistența nu este imediată; interogarea nivelurilor superioare nu este posibilă	Aplicații Web
NoSQL: graf (rețea)	Neo4j, InfiniteGraph	Interogările de tip graf sunt rapide	Puțin eficiente pentru analize de alt tip	Sisteme de recomandare, rețele sociale, rute transport

Sursa: (Foster & Heus, 2021, p. 72)

Bazele de date relaționale sunt utilizate pe scară largă și reprezintă o soluție potrivită în foarte multe contexte de cercetare din științele sociale. Ele permit organizarea, stocarea și analiza eficientă a unui volum mare de date structurate (liniile reprezintă cazurile iar coloanele atributele). Adoptate relativ mai recent, dar într-un ritm tot mai alert, bazele non-relaționale pot fi utile în anumite contexte specifice de cercetare și analiză. Ele sunt mult mai flexibile, pot conține multe tipuri de date, foarte diferite, au o schemă dinamică (aceasta este fixă în cazul bazelor SQL), datele sunt organizate pe linii (vs. tabele în SQL), sunt scalabile pe orizontală[17] (vs. verticală în SQL), sunt mai potrivite în cazul datelor flexibile, fără relații rigide. Suportul de specialitate în cazul bazelor NoSQL este relativ mai limitat, dar în creștere.

Sistemul de management al unei baze de date (DBMS)

În general fiecare bază de date are asociat un program cunoscut sub numele de sistem de management al bazei de date (DataBase Management System - DBMS). DBMS poate fi definit simplu astfel:

[17] Pentru a mări performanța și capacitatea unei baze SQL trebuie să-i creștem puterea de calcul, deci, teoretic, acest tip de bază poate crește doar până la un anumit nivel (determinat de limita componentelor hardware).

> *„Un DBMS este un sistem care interacționează cu utilizatorii, cu alte aplicații și cu baza de date pentru a captura și analiza datele" (Foster & Heus, 2021, p. 71).*

Deci, DBMS are rolul de interfață între baza de date și utilizatorii acesteia (persoane sau programe). DBMS definește drepturile de accesare și modificare a datelor, respectiv asigură accesarea și modificarea datelor. De asemenea, DBMS supraveghează și controlează baza de date, respectiv realizează diferite funcții administrative precum monitorizarea performanței, reglarea, realizarea unei copii de rezervă (backup) și recuperarea datelor.[18]

DBMS are trei componente (Foster & Heus, 2021, p. 71), acestea lipsind sau fiind limitate în cazul seturilor de date folosite de programele de analiză statistică:

- **un model al datelor:** elementele, proprietățile acestora, respectiv relațiile dintre elemente; simplu spus, schema bazei de date, adică tabelele, relațiile dintre ele, respectiv coloanele fiecărui tabel și tipul lor;
- **un limbaj de interogare:** modalitatea prin care comunicăm cu baza de date pentru a o modifica (adăugăm, actualizăm, ștergem elemente), producem noi tabele, extragem informații din bază; majoritatea bazelor de date relaționale suportă SQL; limbajul asigură accesul eficient la date și oarecum optimizează interogările astfel încât rezultatele să fie accesibile cât mai repede; cel puțin la fel de important, limbajul asigură accesul simultan la date pentru mai mulți utilizatori (persoane și programe);
- **un suport pentru tranzacții și recuperarea datelor:** o bază de date protejează accesul la date, respectiv integritatea acestora; de exemplu, în cazul în care PC-ul nu mai funcționează brusc în timpul realizării unei operațiuni, conținutul bazei să nu fie corupt.

DBMS simplifică foarte mult managementul datelor mari.[19] Acestea pot fi organizate în diferite modalități care permit explorarea rapidă și eficientă,

[18] https://www.oracle.com/ro/database/what-is-database/#WhatIsDBMS

[19] Unele versiuni recente de baze de date simplifică și mai mult tot procesul, automatizând majoritatea acțiunilor legate de acestea. Un exemplu în acest sens îl reprezintă bazele de date

stocarea de durată, consistența și integritatea, analize intuitive. Toate aceste acțiuni pot fi realizate în cadrul DBMS și/sau prin intermediul aplicațiilor și softurilor cu care aceasta este conectată (Foster & Heus, 2021, p. 68).

În concluzie, oricine este interesat de analiza datelor sociale trebuie să înțeleagă cel puțin la nivel de începător ce este și cum funcționează o bază de date și un sistem de management al acesteia. Așa cum concluzionează și alți autori (Foster & Heus, 2021, p. 67), dacă ar trebui să alegem o singură recomandare din acest capitol, aceasta ar fi: „Atunci când ai date mari, folosește o bază de date!". Mai nuanțat, preferăm să folosim o bază de date mai ales în situațiile în care cel puțin una dintre afirmațiile următoare este adevărată:

- setul de date este actualizat relativ constant de către diferiți utilizatori,
- ne dorim ca utilizatorii să aibă drepturi diferite relativ la accesarea și folosirea datelor,
- analiza este realizată automat pe un server care trimite ulterior rezultatele către un browser sau alt tip de client (aplicație informatică, dispozitiv inteligent etc.).

2.4. Formate de fișiere pentru Big Data

Probabil cele mai relevante criterii în funcție de care putem descrie și evalua formatele de fișiere pentru bazele mari de date sunt următoarele:

- text vs. binar, respectiv datele pot fi citite / înțelese de oameni sau doar de softuri specifice (human-readable vs. machine-readable);
- stocare pe linii vs. coloane (cazuri vs. atribute);
- schema bazei de date: inclusă sau nu, respectiv modificabilă sau nu;
- divizibilitatea: posibilă sau nu;
- gradul de compresie.[20]

autonome de la Oracle. O astfel de bază (autonomous database) este situată în cloud și folosește algoritmi de învățare automată pentru a automatiza procese precum securizarea, producerea și gestionarea copiilor de siguranță, actualizarea bazei sau alte operații rutiniere de management realizate în mod obișnuit de către administratorul unei baze de date (https://www.oracle.com/autonomous-database/what-is-autonomous-database).

[20] Pentru o listă extinsă se poate consulta Tabelul 2.4-3.

Text vs. binar

La nivel de bază, toate tipurile de fișiere stochează datele în format binar. Ambele tipuri de formate, text și binar, stochează datele în serii de biți (valorile 0 și 1). Dincolo de această trăsătură comună, cele două formate prezintă o serie de diferențe.

Formatul text este mai restrictiv, conține doar text (textul și numerele sunt toate stocate ca text) (biții reprezintă caractere). Formatul binar poate stoca orice tip de informații / date, text, număr, imagine, audio și video, în orice combinație (biții reprezintă orice tip de date).

Formatul text este ușor de citit și înțeles de utilizatori, respectiv ușor de folosit și modificat (e nevoie de un simplu editor de text). Formatul binar este cel mai adesea imposibil de citit și înțeles de utilizatori, fiind nevoie de un soft special pentru asta.

Formatul binar poate fi corupt mai ușor, erorile sunt dificil de identificat, iar o mică eroare poate face ca fișierul să nu mai poată fi citit (corect) de computer. Erorile din formatele text pot fi identificate și corectate ușor, iar fișierul se deschide fără probleme chiar dacă include erori.

Foarte multe din datele cu care lucrăm sunt de tip numeric. Astfel de date pot fi reprezentate atât în formă de text, cât și în formă numerică. Formatul binar este mai eficient pentru stocarea numerelor. De exemplu, pentru a stoca un număr cel mult egal cu 65536, formatul binar are nevoie de doi biți, în timp ce formatul text consumă cinci biți.

Stocare pe linii vs. coloane (cazuri vs. atribute)

Datele structurate pot fi stocate în două moduri: în relație cu liniile / cazurile sau în relație cu coloanele / atributele. Fiecare dintre aceste variante prezintă avantaje și dezavantaje, funcție de situația concretă în care ne aflăm. În imaginile de mai jos (Figura 2.4-1, Figura 2.4-2) am ilustrat situația generală, respectiv un exemplu. În fiecare situație tabelul conține trei cazuri (linii) și patru atribute (coloane). Stocarea datelor în funcție de linii ordonează datele astfel: valorile asociate primului caz, în ordinea atributelor, apoi valorile asociate cazului secund, în ordinea atributelor, și tot așa. Stocarea datelor în

funcție de coloane ordonează datele astfel: valorile asociate primului atribut, în ordinea cazurilor, apoi valorile asociate atributului secund, în ordinea cazurilor, și tot așa.

Stocarea în funcție de cazuri / linii este de preferat atunci când dorim să adăugăm, eliminăm și modificăm ușor și rapid cazuri, respectiv să accesăm și procesăm un caz / o linie în întregime. Aceste avantaje o fac utilă în cazul OLTP (OnLine Transactional Processing). Stocarea în funcție de atribute / coloane este utilă mai ales atunci când dorim să interogăm (analytic queries) rapid seturi mari de date cu privire la anumite atribute (OLAP = OnLine Analytical Processing). Ignorând toate atributele care nu apar în interogarea dorită, acest tip de stocare reduce semnificativ durata procesării.

Figura 2.4-1. Stocare pe linii vs. coloane (cazuri vs. atribute)

	Col A	Col B	Col C	Col D
Linia 1	A1	B1	C1	D1
Linia 2	A2	B2	C2	D2
Linia 3	A3	B3	C3	D3

Stocare pe linii

A1 B1 C1 D1 A2 B2 C2 D2 A3 B3 C3 D3

Stocare pe coloane

A1 A2 A3 B1 B2 B3 C1 C2 C3 D1 D2 D3

Stocare pe linii	
Cazul / linia 1	A1
	B1
	C1
	D1
Cazul / linia 2	A2
	B2
	C2
	D2
Cazul / linia 3	A3
	B3
	C3
	D3

Stocare pe coloane	
Atributul / coloana 1 (id)	A1
	A2
	A3
Atributul / coloana 2 (firmă)	B1
	B2
	B3
Atributul / coloana 3 (poziție)	C1
	C2
	C3
Atributul / coloana 4 (salariu)	D1
	D2
	D3

Figura 2.4-2. Stocare pe linii vs. coloane (cazuri vs. atribute) (exemplu)

id	companie	poziție	salariu
1	A	conducere	10000
2	B	execuție	5000
3	C	execuție	6000

Stocare pe linii

1 A conducere 10000 2 B execuție 5000 3 C execuție 6000

Stocare pe coloane

1 2 3 A B C conducere execuție execuție 10000 5000 6000

Stocare pe linii	
Cazul / linia 1	1
	A
	conducere
	10000
Cazul / linia 2	2
	B
	execuție
	5000
Cazul / linia 3	3
	C
	execuție
	6000

Stocare pe coloane	
Atributul / coloana 1 (id)	1
	2
	3
Atributul / coloana 2 (firmă)	A
	B
	C
Atributul / coloana 3 (poziție)	conducere
	execuție
	execuție
Atributul / coloana 4 (salariu)	10000
	5000
	6000

Schema unei baze de date

Fiecare bază de date are asociată o schemă. Schema (structura) conține informații cu privire la definițiile atributelor, tipurile și formatele acestora, uneori și la rolul atributelor. Cu privire la o schemă, ne interesează cel puțin următoarele:

- Poate fi modificată (adăugarea / eliminarea / redenumirea unui atribut) și cât de ușor se poate face asta?
- Păstrează compatibilitatea între diferitele versiuni ale schemei? Cum se realizează acest lucru?
- Poate fi citită de oameni (human-readable) sau doar de un soft? E necesar să fie?
- Cât de repede poate fi procesată?
- Ce impact are asupra mărimii datelor (spațiului necesar pentru stocare)?

O schemă poate fi fixă, dar cel mai adesea aceasta suferă modificări în timp (ani). Modificările pot lua diferite forme precum schimbarea definiției unor atribute, a tipului și/sau variantelor de răspuns asociate acestora, respectiv redenumirea unor atribute, eliminarea și/sau adăugarea unor atribute noi. Modificarea schemei, inclusiv păstrarea compatibilității cu versiunile anterioare, este posibilă doar în cazul unora dintre formatele de bază de date. O schemă poate fi inclusă în baza de date sau furnizată separat. Suplimentar, în cazul fișierelor binare, schemele sunt utilizate pentru a cripta și decripta conținutul.

Divizarea unei baze de date

Bazele de date conțin adesea un volum foarte mare de date (număr mare de linii și coloane). Distincția dintre o bază de date mică și una mare este dificil de făcut, depinzând foarte mult de context. Se consideră că o bază mică este una care poate fi încărcată în memoria unui PC obișnuit (o bază mare nu poate fi). În context instituțional, o companie poate folosi următoarea clasificare, funcție de numărul de linii din baza de date: mică (< 1 mil.), medie (1<10 mil.), mare (10<100 mil.), foarte mare (100+ mil.). Cel puțin în cazul bazelor mari și foarte mari, e important ca formatul folosit să permită

divizarea (împărțirea) bazei în baze mai mici (chunks of data) pentru a putea paraleliza operațiile (rularea în paralel a comenzilor, folosirea simultană a mai multor procesoare). Formatele csv, xml și json sunt dificil de divizat.

Comprimarea unei baze de date

Rata de compresie reprezintă o altă caracteristică foarte importantă. Stocarea, transferul și accesarea unei baze de date reprezintă operațiuni care necesită foarte multe resurse (timp și bani), deci e util ca aceasta să fie cât mai mică. Rata de compresie a unei baze de date depinde foarte mult de formatul acesteia. Bazele de date organizate pe coloane au rate de compresie semnificativ mai mari, deci o bază de date organizată pe coloane va ocupa un spațiu semnificativ mai mic comparativ cu aceeași bază organizată pe linii (compresia este mai eficientă atunci când datele sunt de același tip, cum e cazul organizării pe coloane / atribute). Această diferență se observă foarte ușor în Tabelul 2.4-1 - datele în format CSV au o mărime de aproximativ șapte ori mai mare comparativ cu cele în format Parquet.

Tabelul 2.4-1. Mărimea unei baze de date în formatul CSV vs. Parquet

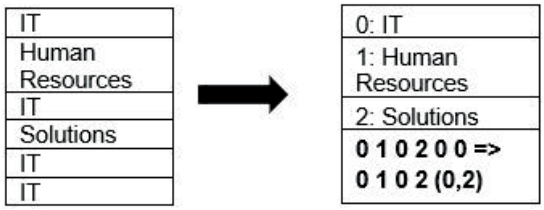
Tip bază de date	Numele bazei de date	# linii	# coloane	Mărime (GB)	
				CSV	Parquet
x-small	claim_history_day	317,617	201	0.3	0.04
small	claim_history_month	5,548,609	202	4.8	0.7
medium	claim_history_year	66,001,292	201	57.3	7.5
large	claim_history	408,197,137	201	351.5	45.1

Sursa: Garnett, Ryan; Wong, Ray & Reed, Dan. 2022. Speed Up Data Analytics and Wrangling With Parquet Files. https://www.r-bloggers.com/2022/04/speed-up-data-analytics-and-wrangling-with-parquet-files/

Pentru a ilustra modul în care funcționează compresia în cazul fișierelor Parquet, prezentăm un exemplu simplu de comprimare prin codificare de tip dicționar (dictionary compression) (Figura 2.4-3). În acest exemplu, atributul Department are trei categorii (IT, Human Resources, Solutions), categoria IT fiind prezentă de patru ori. Fiecărei categorii îi este asociată o valoare numerică (0: IT, 1: Human Resources, 2: Solutions), codificarea devenind parte din dicționarul bazei de date (valorile numerice sunt stocate binar). Seria de date inițială devine „0 1 0 2 0 0", ulterior fiind comprimată suplimentar sub forma „0 1 0 2 (0,2)", unde (0,2) semnifică faptul că valoarea

0 (IT) este luată de două ori consecutiv. Astfel, aceeași bază de date în format Parquet va ocupa un spațiu semnificativ mai mic comparativ cu formatele CSV, JSON, sau XML.

Figura 2.4-3. Formatul Parquet: un exemplu de comprimare a dicționarului



Sursa: Butterfly Thoughts. 2021. Probably The Best Alternative to CSV Storage: Parquet Data. https://geekflare.com/parquet-csv-data-storage/

Mai mult, timpul de procesare necesar pentru o bază de date Parquet va fi semnificativ mai redus (la fel și costurile asociate). Astfel, în cadrul unui experiment în cadrul căruia au fost comparate formatele CSV și Parquet (relativ la durata de procesare a unor sarcini precum join, grupare, numărare, sumarizare în cazul unor baze de mărimi diferite), timpul de procesare a variat în cazul CSV de la 10.9 secunde la 40 minute, iar în cazul Parquet de la 0.3 la 16.4 secunde.[21] O concluzie similară a rezultat și în urma altui experiment (Tabelul 2.4-2).

Tabelul 2.4-2. Performanța CSV vs. Parquet

Dataset	Mărimea bazei (Amazon S3) (TB)	Timp execuție (secunde)	Volum date scanate (TB)	Cost ($)
CSV	1,00	236,00	1,15	5,750
Parquet*	0,13	6,78	2,51	0,013
Economii (timp și bani)	87% mai puțin Parquet	34x mai rapid Parquet	99% mai puțin Parquet	99.7% mai puțin Parquet

*(*comprimare Snappy)*

Sursa: Sinha, Abhishek & Mukerje, Neil. 2016. Analyzing Data in S3 using Amazon Athena. https://aws.amazon.com/blogs/big-data/analyzing-data-in-s3-using-amazon-athena/

O comparație sintetică

În Tabelul 2.4-3 am caracterizat sintetic principalele formate de baze de date folosind dimensiunile discutate anterior dar și altele. Cei interesați de mai

[21] Caracteristici PC: Ubuntu 20, 16 cores, 2.30GHz CPU, 1TB RAM.

multe detalii pot consulta sursele online menționate sau diferite texte de specialitate.

Tabelul 2.4-3. O comparație a formatelor de fișiere utilizate în cazul Big Data

	1978	1996	2001	2008	2009	2013	2016
Format[22]	**CSV**	**XML**	**JSON**	**Protocol Buffers**	**AVRO**	**Parquet**	**ORC**
Date definite ca ...	text	text	text	text	binar	binar	binar
Tipul datelor[23] **este definit?**	nu	nu	da	da	da	da	da
Poate fi citit de oameni?	da	da	da	nu	nu	nu	nu
Suportă structuri ierarhice?	nu	da	da	da	da	da	da
Schema este asociată datelor?	nu	oarecum	oarecum	da	da	da	da
Evoluția schemei	+	+	+	++	+++	++	++
Tip stocare	linie	linie	linie	linie	linie	coloană	coloană
Divizibilitate?	da	nu	da (JSON lines)	nu	da ++	da ++	da +++
Rată compresie	+	+	+	++	+++	+++	+++
Viteză scriere	+	+	+	+	+++	++	+++
Viteză citire	+	+	+	+	++	+++	++
OLAP/OLTP[24]	OLTP	OLTP	OLTP	OLTP	OLTP	OLAP	OLAP
Batch[25]	da	da	da	da	da	da	da
Stream[26]	da	nu	da	da	da	nu	nu
Ecosisteme	Relativ universal	Enterprise	API & Web	RPC & Kubernetes	Big Data & Streaming	Big Data & BI	Big Data & BI

Sursa: Pentru realizarea acestui tabel au fost folosite o serie de resurse online.[27]

[22] CSV (comma-separated value), XML (eXtensible Markup Language), JSON (JavaScript object notation), AVRO, ORC (Optimized Row Columnar).

[23] Dacă definim tipul de atribut / valoare, avem cel puțin două avantaje: putem identifica rapid și corect informația (de exemplu, putem distinge un număr într-un text sau o valoare nulă de textul "null"); codificarea în formatul binar reduce spațiul de stocare (de exemplu, pentru a stoca valoarea "1234" avem nevoie de 4 octeți în timp ce valoarea 1234 consumă doar 2 octeți).

[24] OLTP = OnLine Transactional Processing; OLAP = OnLine Analytical Processing.

[25] Procesarea (citirea, transformarea, analiza) simultană a mai multor cazuri (grupuri de cazuri sau toate cazurile) dintr-o bază de date.

[26] Procesarea (citirea, transformarea, analiza) unui caz în timp real, imediat ce acesta a fost inclus în baza de date.

[27] *Ngom, Aida. 2020. Comparison of different file formats in Big Data. https://www.adaltas.com/en/2020/07/23/benchmark-study-of-different-file-format Nexla. 2018. An Introduction to Big Data*

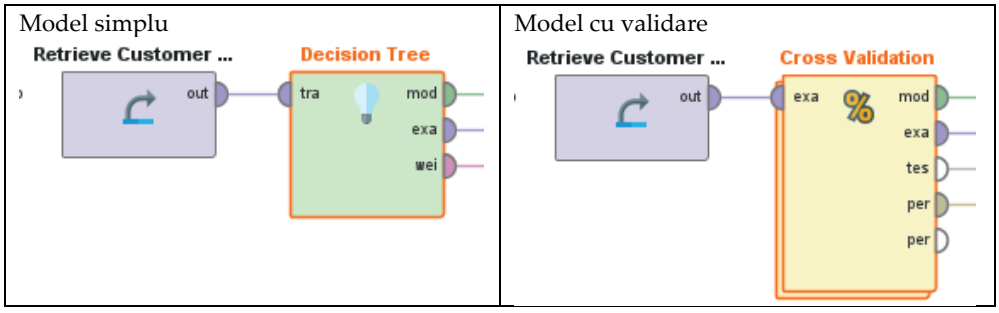
Alegerea formatului de fișier pentru o bază de date depinde de tipul de sarcină și de criteriile de calitate urmărite cu prioritate. Astfel, dacă ne interesează rata compresiei și viteza de recuperare a datelor în vederea raportării, vom alege formatele Parquet sau ORC; dacă ne interesează viteza de scriere și/sau posibilitatea de a modifica schema bazei de date, vom prefera formatul AVRO. Simplu spus, atunci când ne interesează să analizăm câteva atribute, alegem un format de bază ordonat în funcție de coloane; când vrem să accesăm cazuri, folosim formate bazate pe linii. Astfel, formatul AVRO este mai potrivit atunci când procesăm o bază de date, iar Parquet atunci când o analizăm.

Formats. Understanding Avro, Parquet, and ORC. *https://www.nexla.com/resource/introduction-big-data-formats-understanding-avro-parquet-orc* *Luminousmen. 2019. Big Data File Formats.* *https://luminousmen.com/post/big-data-file-formats* *Bhatia, Rahul. 2021. Big Data File Formats.* *https://www.clairvoyant.ai/blog/big-data-file-formats* *Kumari, Rashmi. 2022. Big Data File Formats.* *https://www.hcltech.com/blogs/big-data-file-formats*

3. Programul RapidMiner Studio: Data mining pe înțelesul tuturor

Dacă ar trebui să descriem programul RapidMiner Studio în cât mai puține cuvinte, am putea spune simplu „sintaxă vizuală". Pentru a rula o analiză, tot ce trebuie să facem este să alegem comenzile dorite, să le conectăm între ele și să indicăm opțiunile preferate relativ la fiecare comandă (unde este cazul). De exemplu, pentru a realiza un model de predicție folosind clasificatorul arbore decizional (decision tree) trebuie să conectăm doar doi operatori, setul de date și clasificatorul (Figura 3-1, stânga). Pentru a indica ce rezultate dorim să obținem, conectăm output-urile operatorului „Decision Tree" (mod, exa și wei) la butoanele rezultate (res). Obținem astfel modelul de predicție, setul de date și ponderările asociate atributelor / variabilelor. Suplimentar, dacă dorim să validăm modelul, folosim operatorul (comanda) „Cross Validation" și includem clasificatorul „Decision Tree" în interiorul acestuia (Figura 3-1, dreapta). Pentru a vedea rezultatele, conectăm fiecare dintre output-urile mod, exa, tes și per la câte un buton res. Obținem astfel modelul de predicție, setul de date folosit pentru antrenarea modelului, setul de date folosit pentru testarea modelului și măsurile de performanță a clasificării.

Figura 3-1. Cum arată un model de predicție în RapidMiner Studio?



RapidMiner Studio îndeplinește câteva condiții majore ale unui soft de analiză de data mining: (1) este intuitiv și ușor de utilizat, (2) face posibile

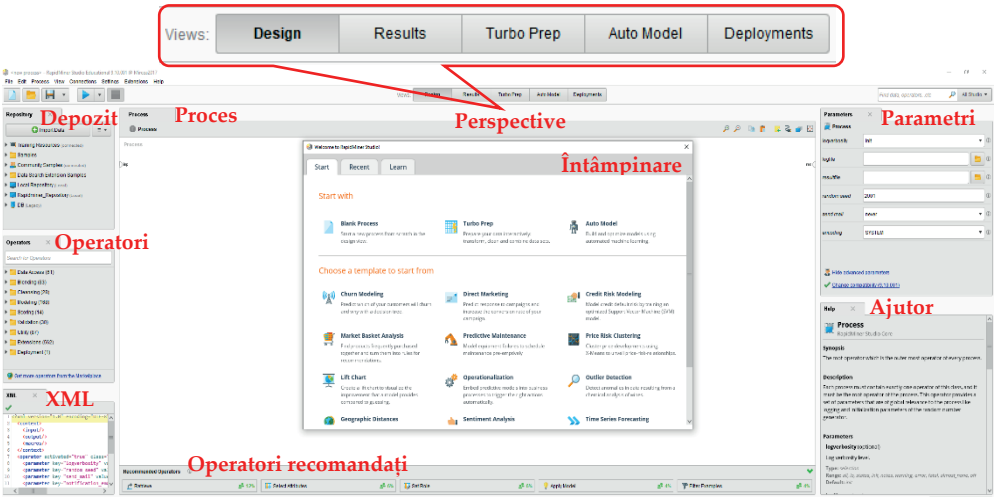
reproductibilitatea și reutilizarea analizelor, (3) poate analiza date de diferite tipuri (structurate și nestructurate; date, text și imagine), (4) poate rula foarte multe tipuri de modele, (5) oferă posibilitatea de automatizare a proceselor de pregătire și modelare a datelor, (6) poate interacționa cu și rula comenzi scrise în alte programe / limbaje (Python, R, SQL).

În acest capitol vom discuta despre perspectivele RapidMiner Studio (Views), ecranul de întâmpinare (Welcome), ferestrele și meniul RapidMiner Studio (Chisholm, 2013; Mierswa, 2016a; RapidMiner, 2014).

3.1. Perspectivele RapidMiner Studio (Views)

RapidMiner Studio oferă posibilitatea de a trece rapid de la o categorie majoră de informații și/sau acțiune la alta, funcție de intențiile utilizatorului. Alegerea între aceste categorii, numite generic Perspective (Views), se face simplu, alegând unul dintre taburile poziționate în zona de sus-centru, pe o singură linie. Astfel putem alege una dintre perspectivele Design, Results, Turbo Prep, Auto Model sau Deployments. La start, programul afișează automat perspectiva Design și ferestrele implicite asociate acesteia (Figura 3.1-1), dar utilizatorul poate schimba rapid pe oricare dintre celelalte perspective.

Figura 3.1-1. O imagine de ansamblu asupra programului RapidMiner Studio



Notă: Perspectiva Design la start, varianta implicită, ușor modificată

Fiecare perspectivă include ferestre (Panels) specifice tipului de informație sugerat de denumirea acelei perspective. Astfel, perspectiva Design include ferestre care conțin informații utile pentru definirea procesului de data mining, în perspectiva Results apar rezultatele analizei, Turbo Prep ne ajută să pregătim rapid datele pentru analiză, perspectiva Auto Model poate fi folosită pentru a automatiza modelarea datelor, iar perspectiva Deployments ne ajută să implementăm proiectele de analiză dezvoltate (să le punem în producție).

Organizarea perspectivelor este de tip modular, pe ferestre ce pot fi afișate implicit sau la cerere. Dacă dorim să nu afișăm o fereastră, o putem elimina simplu apăsând pe semnul x poziționat în dreptul numelui acesteia. Același lucru se poate face și din meniul View + „Show Panel" (suplimentar, aici putem alege să afișăm o serie de alte ferestre).

Fiecare fereastră poate fi personalizată în sensul că putem să-i schimbăm poziția, forma și mărimea. Pentru a schimba forma și mărimea unei ferestre poziționăm cursorul pe una dintre marginile acesteia astfel încât să apară o săgeată dublă, ținem apăsat butonul din stânga și tragem marginea în poziția dorită. Pentru a muta o fereastră, poziționăm cursorul în zona numelui, ținem apăsat butonul din stânga și o mutăm în noua poziție.

Pentru exemplificare, descriem aici pe scurt perspectiva Design. Aceasta include în mod obișnuit ferestrele Repository (depozit de date), Operators (operatori), Process (proces), Parameters (parametri) și Help (ajutor). Utilizatorul poate adăuga și alte ferestre posibil utile, funcție de utilizator. În cazul de față am adăugat în colțul din stânga-jos fereastra „XML". Pentru a realiza acest lucru, am micșorat puțin cele două ferestre poziționate deasupra ferestrei XML. Alternativ, puteam pune două ferestre în același spațiu (spațiul / zona respectivă va conține două taburi diferite, unul pentru fiecare panel / fereastră).

3.2. Ecranul de întâmpinare (Welcome)

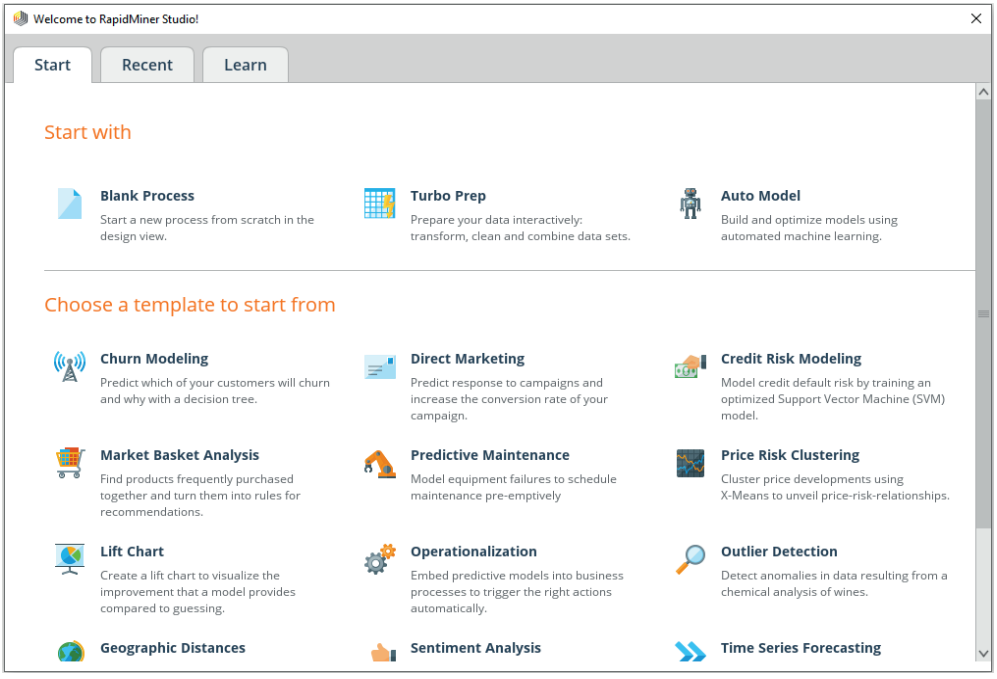
Deși imaginea de mai sus (Figura 3.1-1) oferă o orientare generală utilă, ferestrele și conținutul acestora sunt dificil de vizualizat, motiv pentru care le vom prezenta în continuare pe rând. Ecranul de întâmpinare (Welcome) prezintă trei taburi: Start, Recent și Learn.

Tabul Start (Figura 3.2-1) ne oferă posibilitatea să începem un nou proces (analiză), să pregătim datele pentru analiză folosind „Turbo Prep", respectiv să analizăm datele într-o manieră relativ automatizată (Auto Model).

Alternativ, dacă știm (sau cel puțin avem o idee despre) tipul de problemă de data mining pe care trebuie să o rezolvăm în cadrul proiectului nostru, putem alege unul dintre proiectele deja construite: modelarea renunțării / părăsirii (churn modeling), marketing direct (direct marketing), modelarea riscului de credit (credit risk modeling) etc.

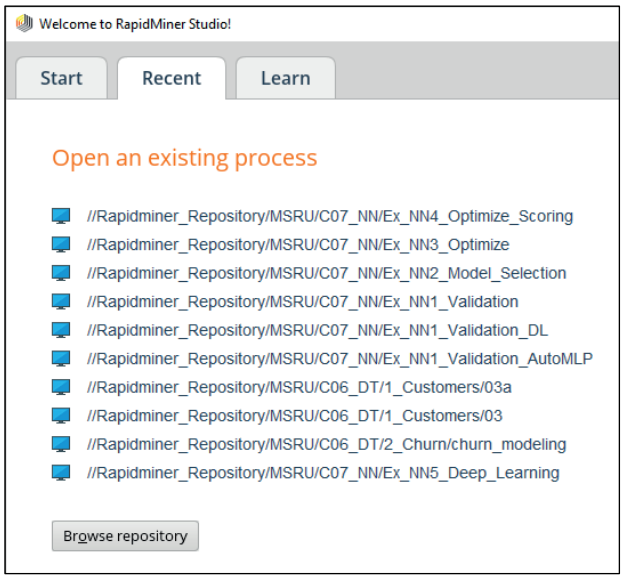
Inspectând aceste exemple ne putem da seama care este cel mai potrivit model pentru analiza dorită de noi, îl putem analiza mai atent și apoi îl putem adapta la problema noastră specifică. Adaptarea poate presupune una sau mai multe acțiuni precum înlocuirea setului de date, definirea rolului variabilelor (numite atribute în RapidMiner), transformarea și selecția unor variabile, setarea parametrilor în cazul unora dintre operatori, schimbarea tehnicii de modelare (de exemplu, putem înlocui operatorul de clasificare arbore decizional cu operatorul kNN) etc.

Figura 3.2-1. Ecranul de întâmpinare: Start



Tabul Recent (Figura 3.2-2) ne oferă posibilitatea de a deschide rapid unul dintre ultimele analize (procese) la care am lucrat, respectiv să inspectăm depozitul de date (Browse Repository).

Figura 3.2-2. Ecranul de întâmpinare: Recent

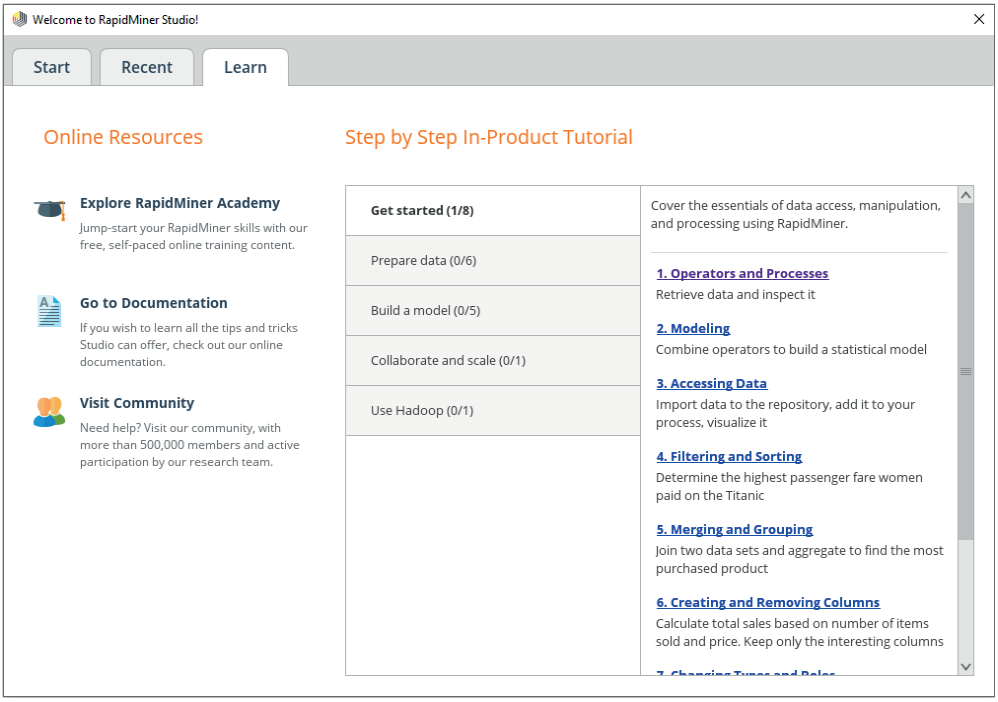


Tabul Learn (Figura 3.2-3) oferă acces rapid la o serie de resurse utile pentru învățarea programului RapidMiner Studio. Resursele disponibile online sunt grupate în trei categorii majore: Academia, Documentația și Comunitatea RapidMiner. Local, în interiorul softului, ne este oferit un tutorial care reproduce pas cu pas principalele tipuri de analize necesare pentru realizarea unui proiect de data mining.

Activitățile de învățare propuse sunt grupate în câteva categorii majore, ordonate logic. Trei dintre acestea sunt relativ mai utile în contextul de față:

- familiarizarea cu programul (accesarea și manipularea datelor, elemente introductive de analiză),
- pregătirea datelor pentru analiză (valorile lipsă, normalizarea variabilelor, identificarea valorilor extreme etc.) și
- obținerea unui model (construirea modelului, validarea modelului, calcularea predicțiilor, compararea modelelor).

Figura 3.2-3. Ecranul de întâmpinare: Learn



3.3. Panelurile RapidMiner Studio (Panels)

Panelurile (ferestrele) RapidMiner sunt folosite pentru a organiza ecranul de lucru în funcție de tipul de sarcină pe care dorim să o realizăm. Principalele paneluri sunt:

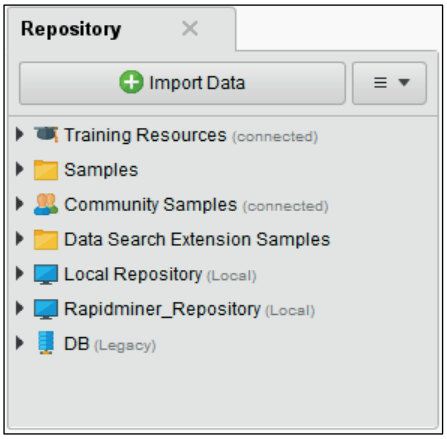
- **Repository**: depozitul de date într-un sens larg, adică seturile de date la care avem acces, procesele, conexiunile, exemplele de analize, resursele pentru învățare);
- **Operators**: operatorii sau comenzile disponibile;
- **Parameters**: opțiunile / setările aferente operatorilor;
- **Recommended operators**: operatorii recomandați;
- **Help**: informații despre comenzi.

Există multe alte ferestre, unele potențial importante, funcție de tipul de informație pe care dorim să-l accesăm rapid. Panelurile pot fi afișate selectându-le în meniul View – „Show Panel”.

Panelul Depozitul de date (Repository)

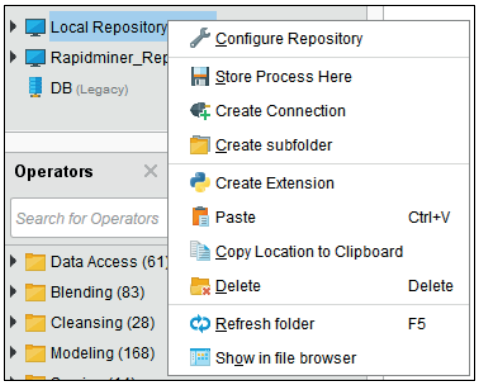
Fereastra (panelul) Repository (depozit de date) conține o serie de foldere în care sunt stocate bazele de date și procesele (sintaxele vizuale) aferente diferitelor proiecte de analiză (Figura 3.3-1). La instalare, RapidMiner propune utilizatorului crearea unui depozit local (numit implicit „Local Repository") situat într-o locație specifică (aceasta poate fi schimbată), respectiv creează câteva foldere care conțin o serie de resurse utile pentru învățarea programului. Astfel, folderul „Training Resources" conține resursele utilizate în cadrul cursurilor RapidMiner Academy, „Samples" include o serie de seturi de date, exemplele de analiză (templates) care apar la fereastra de întâmpinare (tabul Start), respectiv exemplele utilizate în fereastra ajutor (Help), „Community Samples" conține seturi de date și procese realizate de diferiți utilizatori ai programului, „Data Search Extension Samples" conține seturi de date și procese utilizate de diferite extensii instalate de utilizator, iar „DB" include conexiunile la diferite baze de date și aplicații (email, cloud, Tableau, Twitter etc.) create de utilizator. Tabul „Import Data" poate fi utilizat pentru importul datelor (seturi și/sau baze de date) în RapidMiner. În acest caz, importul este realizat pas cu pas, manual, fără producerea și salvarea procesului (seriei de comenzi).

Figura 3.3-1. Panelul Depozitul de date (Repository)



Pentru fiecare element din depozitul de date avem posibilitatea să accesăm comenzile aferente apăsând butonul din dreapta al mouse-lui. Astfel, în cazul unui depozit de date, putem apela rapid comenzi precum configurare, stocare, creare sub-folder, copiere, ștergere etc. (Figura 3.3-2).

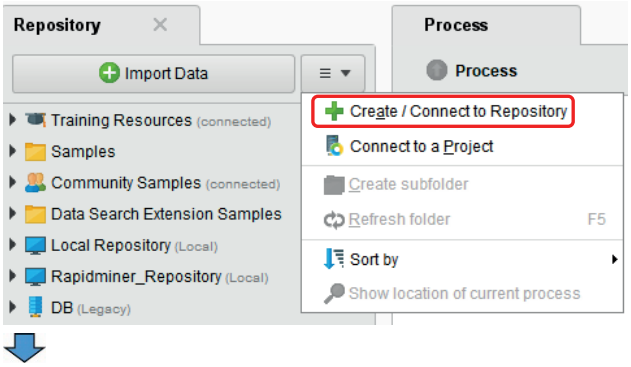
Figura 3.3-2. Lista comenzilor aferente unui depozit de date



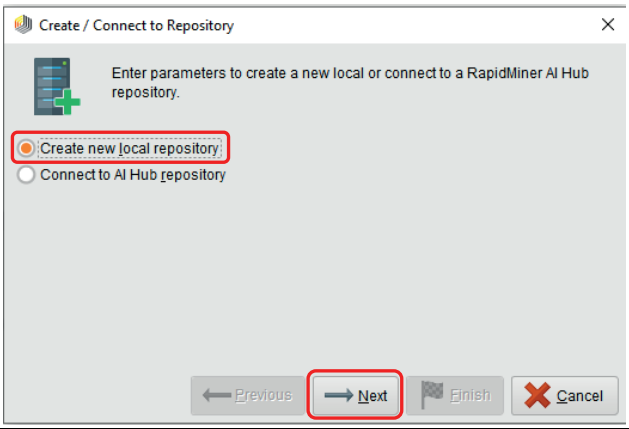
Pentru a putea utiliza exemplele de analiză (fișierele de date și procesele) distribuite împreună cu acest manual trebuie să adăugăm în fereastra Repository folderul DMSS1 (depozitul de date) urmând pașii descriși în Figura 3.3-3.

Figura 3.3-3. Crearea unui depozit de date local

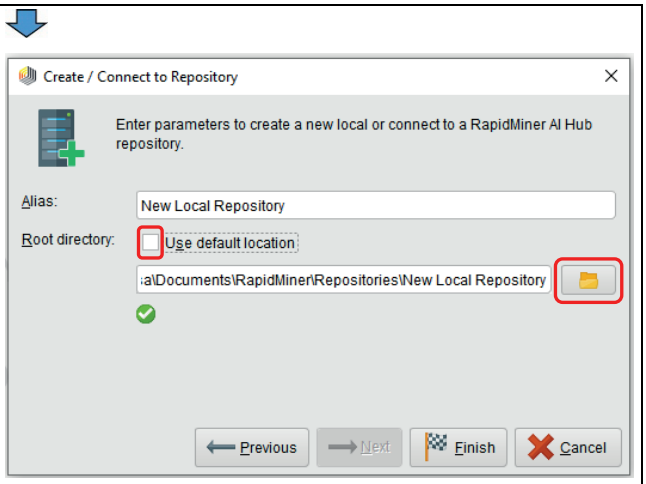
Pasul 1:
În fereastra depozit de date (Repository) se alege opțiunea „Creează / Conectează-te la un depozit de date" („Create / Connect to Repository").



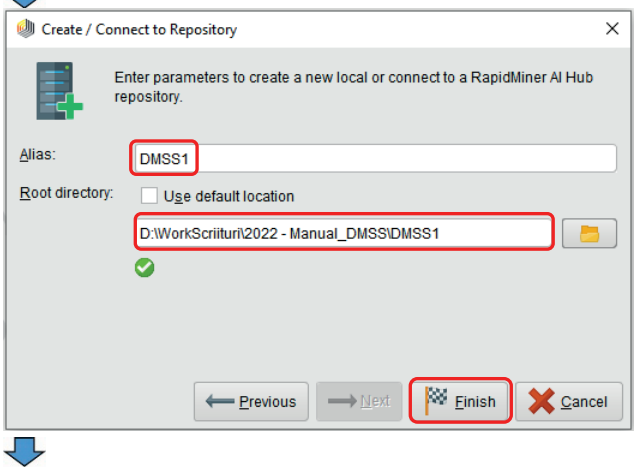
Pasul 2:
Alegem să creăm un nou depozit local de date (Create new local repository).
Dăm click pe butonul Next (pasul următor).



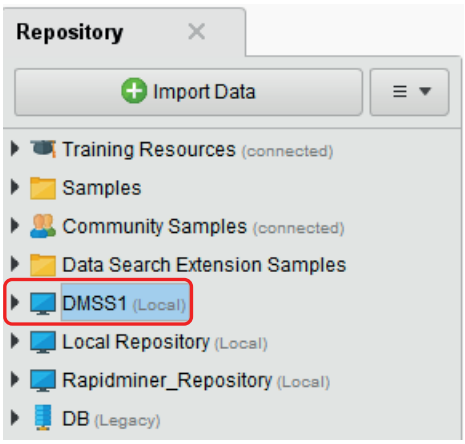
Pasul 3:
Debifăm opțiunea „folosește locația implicită” (Use default location).
Alegem numele și locația folderului care va constitui depozitul de date (de la semnul open).



Pasul 4:
Numele (Alias) depozitului de date este DMSS1 și este situat în folderul „2022 – Manual_DMSS” (locația folderului „2022 – Manual_DMSS” poate diferi de la un computer la altul).
Dăm click pe butonul „Finish” (finalizare).

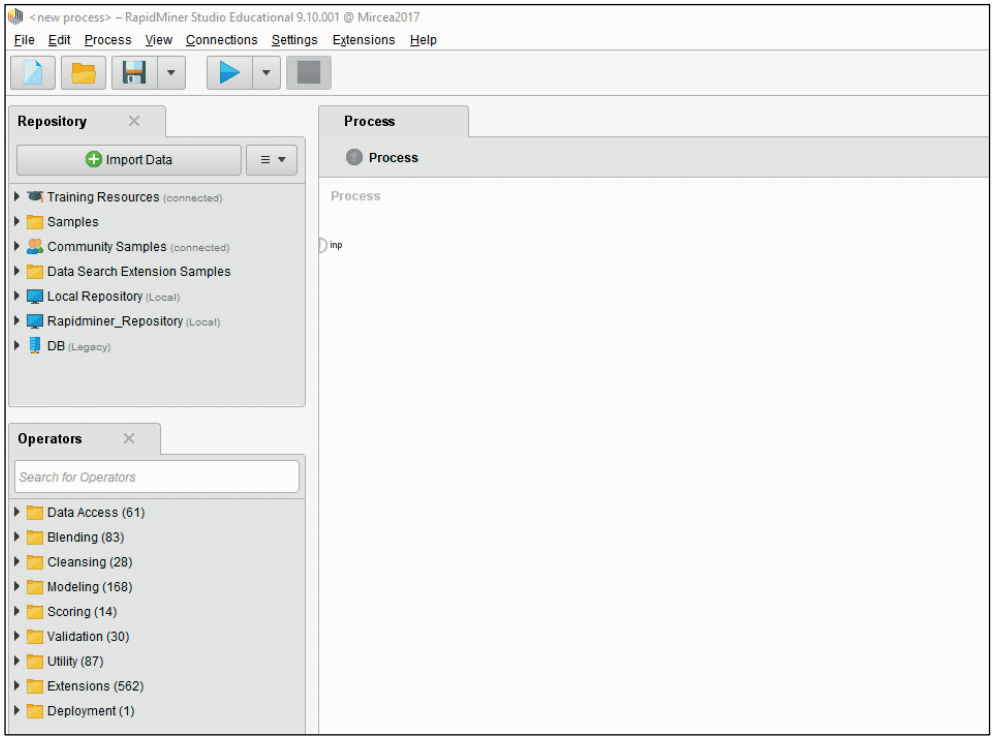


Pasul 5:
Depozitul local nou creat apare în fereastra depozitul de date (Repository).



Pașii necesari pentru a crea un depozit local de date, în format video, apar și în Gif 3.3-1.

Gif 3.3-1. Crearea unui depozit de date local



Înainte de a trece la următorul panel, trebuie să adăugăm o notă importantă cu privire la panelul Repository. E de preferat ca manipularea fișierelor și folderelor (redenumire, mutare, copiere, ștergere) să fie făcută prin intermediul ferestrei Repository și nu în programul „File Explorer" (sau altul similar). Procedând astfel, modificările vor fi automat vizibile în programul RapidMiner. Dacă nu respectăm această regulă, modificările vor fi vizibile în RapidMiner doar după ce am dat click dreapta pe folderul respectiv din Repository și apoi click pe „Refresh Folder".

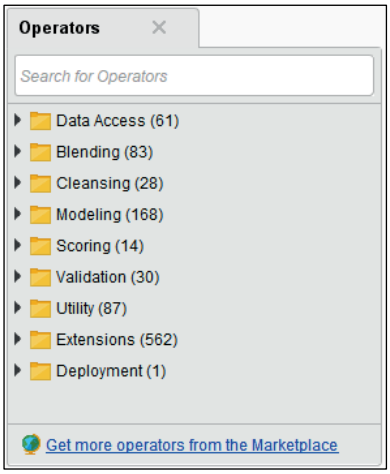
Panelul Operatori (Operators)

Fereastra Operators conține toți operatorii disponibili, inclusiv cei care aparțin de extensiile instalate de utilizator. Operatorii reprezintă comenzile ce pot fi folosite pentru a crea un proces (set de comenzi care compun o

analiză) în RapidMiner. De exemplu, cu operatorul Read CSV putem citi un fișier de date CSV, cu operatorul Store putem salva un set de date în format RapidMiner, cu operatorul Retrieve putem citi aceste date, cu operatorul Decision Tree putem produce un model de predicție de tip arbore decizional, iar cu operatorul Cross-Validation putem valida un model; acești operatori, interconectați într-o ordine specifică, formează împreună un proces particular (o analiză specifică).

Operatorii sunt grupați pe câteva categorii relevante, funcție de obiectivul major urmărit (Figura 3.3-4): „Data Access" (accesarea datelor), Blending (definirea, combinarea și selecția datelor), Cleansing (transformarea datelor), Modeling (modelarea datelor), Scoring (calcularea predicțiilor), Validation (validarea modelului), Utility (asistență pentru realizarea unor acțiuni specifice), Extensions (extensii / module), Deployment (implementarea modelului). Putem adăuga alți operatori folosind opțiunea „Get more operators from the Marketplace". Numărul total de operatori din fiecare categorie este indicat între paranteze.

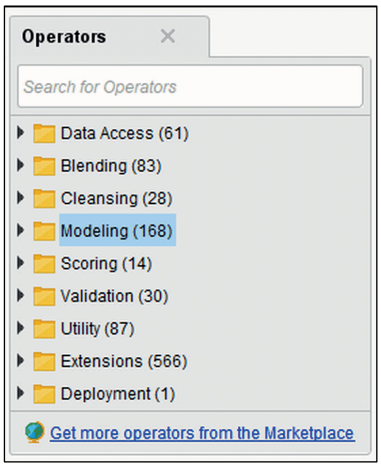
Figura 3.3-4. Panelul Operatori (Operators)



Dacă dorim să căutăm rapid un anumit operator și știm numele acestuia (o parte a numelui sau prima literă din cuvântul / cuvintele care compun denumirea), putem scrie acel nume în fereastra de căutare (Search for Operators), iar fereastra Operators va afișa doar operatorii care conțin acele cuvinte / succesiuni de litere (Gif 3.3-2). Alternativ, dacă nu știm numele

operatorului dar avem o idee cu privire la ceea ce ar trebui să facă, putem căuta operatorul de interes dând click pe categoria la care acesta ar trebui să aparțină din punct de vedere logic.

Gif 3.3-2. Căutarea unui operator în fereastra Operators



Panelul Parametri (Parameters)

În fereastra Parametri sunt afișate setările / opțiunile asociate operatorului selectat. Funcție de setările disponibile și alese de utilizator, respectiv de valorile stabilite pentru aceste setări, operatorul va ține cont sau nu de anumite lucruri, respectiv va realiza acțiunile într-un anumit fel.

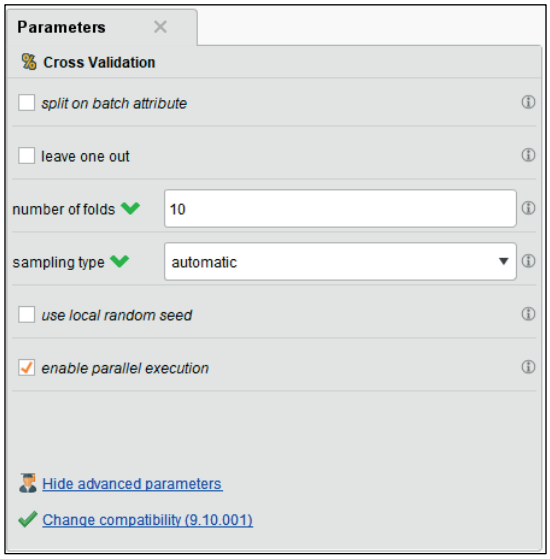
Elementele afișate în panelul Parameters se modifică de fiecare dată când selectăm un operator (indiferent dacă facem selecția în fereastra Operators sau Process), fiind populată cu parametrii specifici acelui operator. De exemplu, dacă selectăm operatorul „Set Role" (definirea rolului variabilelor / atributelor), ne vor apărea parametrii:

- **attribute name** – numele atributului selectat;
- **target role** – rolul definit pentru acel atribut în cadrul analizei respective;
- **set additional roles** – definirea rolurilor pentru o serie de atribute, similar cu opțiunea anterioară cu diferența că în acest caz putem defini rolurile în cazul mai multor atribute (variabile).

Dacă operatorul selectat este „Cross Validation” (Figura 3.3-5), putem defini o serie de parametri precum:

- **split on batch attribute** – folosește pentru divizarea bazei de date atributul special cu rolul batch (definește loturi / sub-seturi de cazuri); în acest caz alegerea altor parametri pentru divizarea bazei nu mai este posibilă;
- **leave one out** – setul de date de test va include pe rând fiecare caz / exemplu din setul de date utilizat pentru antrenarea modelului (training dataset);
- **number of folds** – numărul de sub-seturi de date în care este împărțit setul de date utilizat pentru antrenarea modelului;
- **sampling type** – tipul de eșantion dorit;
- **use local number seed** – indicarea unui număr aleator; opțiunea este utilă în cazul în care dorim să obținem exact aceleași rezultate atunci când reluăm analiza respectivă;
- **enable parallel execution** – execuția în paralel a comenzii, adică utilizarea simultană a mai multor procesoare.

Figura 3.3-5. Fereastra Parametri (Parameters)



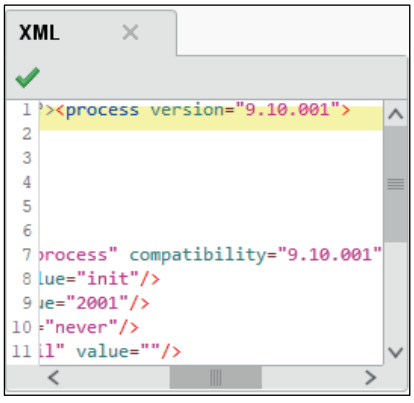
Fereastra Parameters în cazul operatorului Cross Validation

În cazul operatorilor care au mai multe versiuni, putem păstra compatibilitatea cu versiunile anterioare ale programului folosind opțiunea „Change compatibility" (apare doar în cazul operatorilor care au mai multe versiuni). De asemenea, putem alege dacă să fie afișate sau nu opțiunile avansate. Fiecare opțiune este însoțită de un mic semn sugestiv (ⓘ). Atunci când trecem cu cursorul peste acesta, se deschide o fereastră temporară care conține informațiile de bază cu privire la acel parametru. Foarte util, în cazul parametrilor numerici, se prezintă și intervalul de variație teoretic al valorilor posibile. Dacă dăm click pe semnul ⌞, putem vedea distribuția statistică a setărilor alese de către ceilalți utilizatori.

Panelul XML (XML)

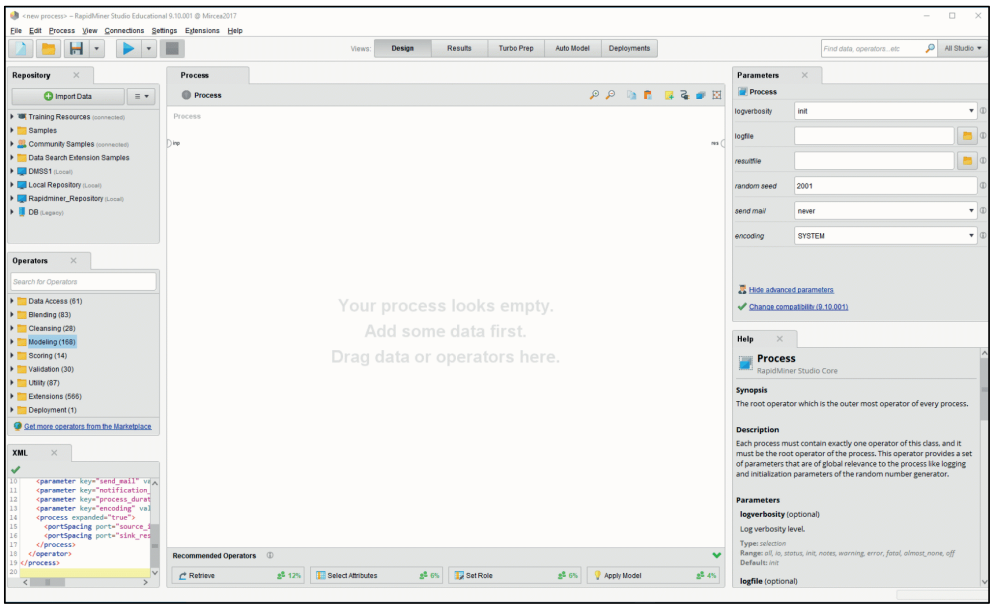
Fereastra XML (Figura 3.3-6) nu este inclusă automat la instalarea programului, dar poate fi afișată de la meniul View - „Show Panel". Această fereastră conține comenzile sau sintaxa analizei în format XML (eXtensible Markup Language). XML este un meta-limbaj utilizat pentru marcarea (adnotarea) documentelor. Prin marcare ne referim la etichete personalizate (tag-uri) care conțin instrucțiuni utilizate de computer pentru a structura un document. Astfel, un document în format XML poate fi transferat între diferite programe, respectiv poate fi citit atât de către oameni, cât și de computere. De exemplu, în RapidMiner, ultima secțiune a primei linii din fereastra XML (<?xml version="1.0" encoding="UTF-8"?><process version="9.10.001">) indică faptul că pentru scrierea acelui proces am folosit versiunea RapidMiner 9.10.

Figura 3.3-6. Panelul XML



Fereastra XML poate fi utilă atunci când dorim să importăm un proces realizat de altcineva, dar avem la dispoziție doar formatul XML al acestuia. Succesiunea de imagini de mai jos ilustrează tocmai această situație (Gif 3.3-3): copiem conținutul procesului de la sursă în fereastra noastră XML, dăm click pe bifa verde și observăm cum fereastra Process se populează cu o serie de operatori interconectați. Alternativ, putem folosi comanda „Import Process" din meniul File (alegem formatul XML). Desigur, procesul va rula doar dacă avem și setul de date utilizat (în acest exemplu, setul de date este unul dintre cele incluse în program).

Gif 3.3-3. Importul manual al unui proces XML

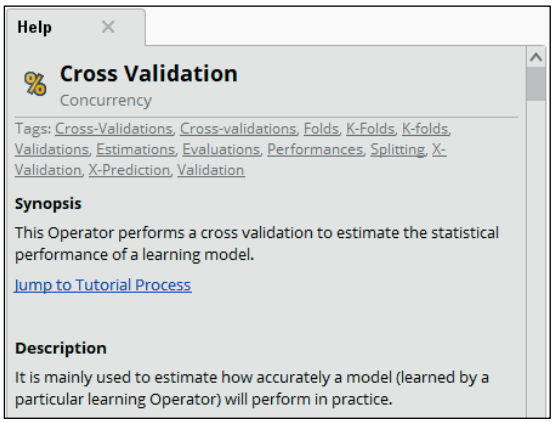


Panelul Ajutor (Help)

În fereastra Help (Ajutor) sunt afișate informațiile disponibile relativ la operatorul selectat (Figura 3.3-7). Conținutul ferestrei Help se modifică de fiecare dată când selectăm un operator, indiferent dacă acesta este selectat (click pe numele sau imaginea operatorului) în fereastra Operators sau Process. Informațiile teoretice prezentate vizează descrierea utilizării generale a operatorului, rolul, parametrii și opțiunile aferente acestuia. De

asemenea, sunt prezentate exemple de utilizare (tutoriale) și se oferă posibilitatea de a încărca în program procesele aferente acestor exemple (click „Jump to Tutorial Process" apoi alegem „Tutorial Process").

Figura 3.3-7. Panelul Ajutor (Help)



Fereastra Help în cazul operatorului Cross Validation

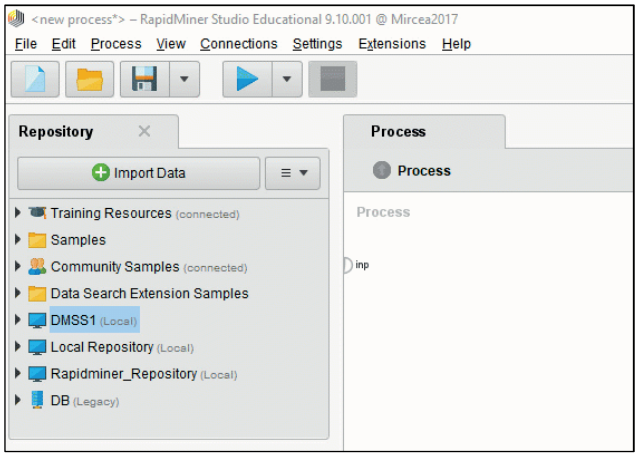
Panelul Proces (Process)

Fereastra centrală, cea mai mare, se numește Process. Aici definim analizele aferente unui proces (serie de comenzi legate între ele). Fereastra poate fi populată foarte simplu cu diferite comenzi, folosind mouse-ul și comanda „drag&drop". În partea de jos a ferestrei Process, putem alege să fie afișați sau nu o serie de operatori recomandați. Recomandările sunt realizate automat de către program în funcție de două tipuri de informații disponibile la un moment dat: operatorii incluși în acel moment în fereastra Process, respectiv operatorii incluși în cadrul unor analize similare realizate de alți utilizatori ai programului. În baza acestor criterii, pentru fiecare dintre operatorii existenți în program se calculează probabilitatea de a fi util în acel context. Programul afișează automat operatorii cu probabilitatea cea mai mare de fi utili în cazul acelei analize, ordonându-i descrescător de la stânga la dreapta.

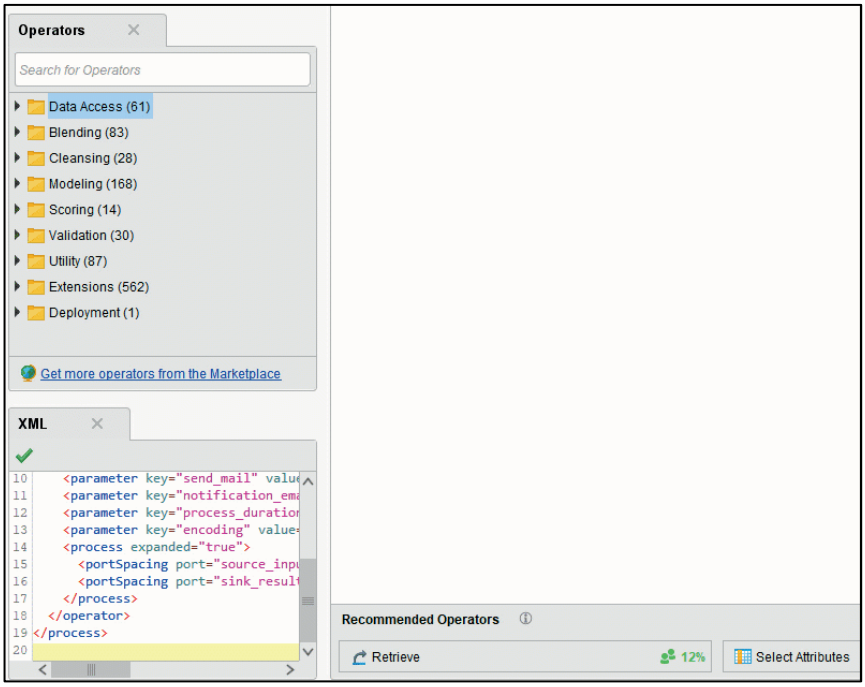
Pentru încărcarea unui set de date sau a unui operator în fereastra Process folosim simplu „drag&drop". În exemplul din Gif 3.3-4 am ilustrat încărcarea directă a setului de date Deals din folderul Samples/Data (fereastra Repository). În exemplul din Gif 3.3-5 am făcut același lucru în doi pași: (1)

drag&drop operatorul Retrieve în fereastra Process și (2) definirea locației bazei de date. Operatorul Retrieve poate fi găsit în folderul „Data Access" din fereastra Operators sau putem să-l căutăm în căsuța Search (trebuie să-i știm numele, măcar aproximativ) sau putem să-l alegem direct din lista operatorilor recomandați.

Gif 3.3-4. Încărcarea unui set de date în fereastra Process



Gif 3.3-5. Încărcarea unui operator în fereastra Process

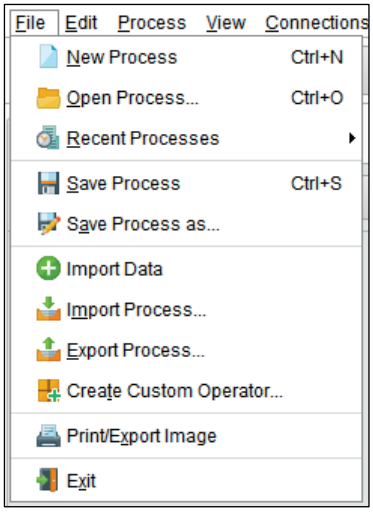


3.4. Meniul RapidMiner Studio

Meniul de comenzi RapidMiner Studio conține câteva categorii majore, denumite cât se poate de informativ: File, Edit, Process, View, Connections, Settings, Extensions și Help.

Meniul File include o serie de comenzi în principal relativ la procese, rolul acestora rezultând clar din denumirea lor (Figura 3.4-1). Folosind aceste comenzi putem deschide un proces nou, încărca un proces existent, inspecta procesele utilizate recent, salva un proces (cu același nume sau cu nume diferit), respectiv importa sau exporta un proces. Tot aici avem posibilitatea de a importa un set de date sau un tabel dintr-o bază de date (Import Data), să printăm sau exportăm o imagine, respectiv să închidem programul.

Figura 3.4-1. Meniul File



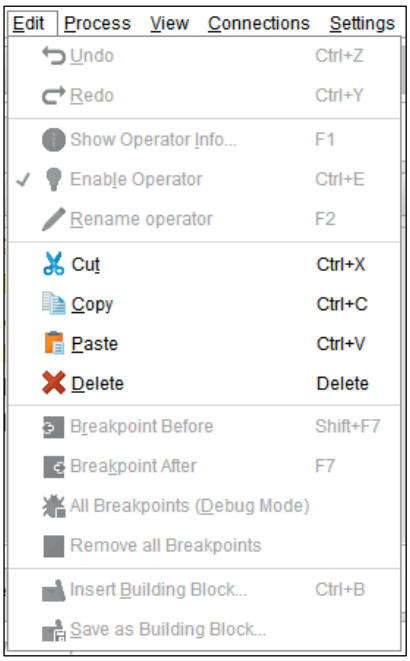
Meniul Edit (Figura 3.4-2) include o serie de comenzi relativ la ...

- modificările realizate anterior: Undo și Redo pentru a reveni pas cu pas la stadiile anterioare, respectiv a relua acțiunile anterioare,
- operatori: informații, activare, redenumire, eliminare, copiere, lipire, ștergere,
- procese: includerea unei pauze – breakpoint – înainte sau după un anumit operator, după fiecare operator, respectiv eliminarea tuturor pauzelor și

- combinații de operatori: includerea sau salvarea unei combinații de operatori – „Building Block”.

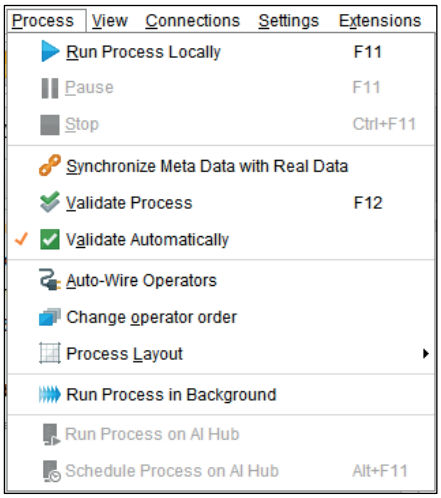
Fiecare dintre aceste comenzi este activată sau nu funcție de ce operatori sunt incluși și/sau selectați în fereastra Process. De exemplu, în imaginea din Figura 3.4-2, comenzile referitoare la operatori sau pauze (breakpoint) nu sunt activate deoarece procesul nu conține niciun operator. Pentru același motiv, nici comanda de inserare a unei combinații de operatori nu este activă. Inserarea unei combinații predefinite de operatori (Insert Building Block) într-un proces nu este posibilă nici dacă am selectat (dat click pe) un operator. Pentru a putea insera o combinație de operatori trebuie să nu avem selectat niciun operator. Fiecare dintre aceste comenzi poate fi apelată folosind o combinație specifică de taste (indicată în dreapta comenzii).

Figura 3.4-2. Meniul Edit



Meniul Process include o serie de comenzi referitoare la procese și anume: rularea locală a unui proces (Run, Pause, Stop, Background), validarea, gestionarea (Auto-Wire, Change operator order, Process layout), respectiv rularea unui proces în AI Hub (putem face acest ultim lucru doar dacă avem definită o conexiune la un depozit online de date) (Figura 3.4-3).

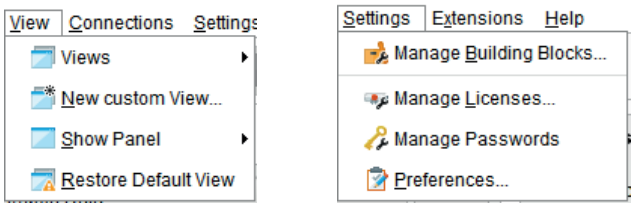
Figura 3.4-3. Meniul Process



Meniul View (Figura 3.4-4) ne oferă acces la perspectivele RapidMiner (Views), ne permite să definim o perspectivă personalizată (New Custom View), ne arată ferestrele posibile pe care le putem afișa (Show Panel) și ne oferă posibilitatea să revenim la starea inițială a ferestrelor, cea definită automat la instalarea programului (Restore Default View).

Comenzile incluse în meniul Settings (Figura 3.4-4) ne ajută să gestionăm blocurile (combinațiile) de operatori (Manage Bulding Blocks), licențele și parolele, respectiv să setăm opțiunile preferate cu privire la o serie foarte mare de aspecte (generale, start, sistem, unelte, actualizări, softurile R și Python etc.). De exemplu, dacă dorim să rulăm în RapidMiner comenzi din R sau Python, aici vom defini locația fișierelor care rulează aceste programe, respectiv putem testa comunicarea dintre RapidMiner și aceste programe.

Figura 3.4-4. Meniurile View și Settings



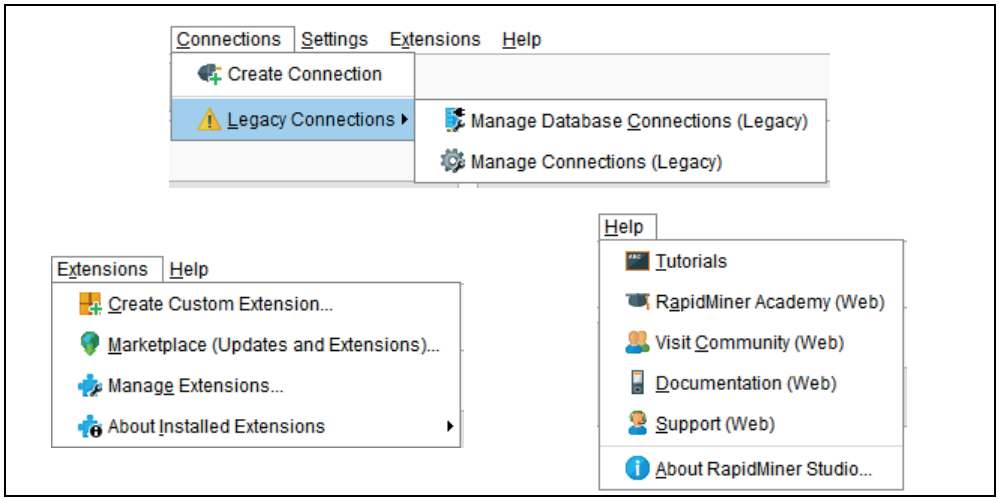
În meniul Connections (Figura 3.4-5) putem defini și gestiona legăturile dintre RapidMiner și diferite tipuri de servicii oferite în cloud (precum

Amazon S3, Azure), baze de date (MySQL, Oracle, PostgreSQL), softuri (Tableau) sau aplicații (Twitter). Conexiunile definite de utilizator în relație cu un anumit Repository (DMSS1 de exemplu) vor fi stocate în folderul Connections din acel Repository.

În meniul Extensions (Figura 3.4-5) apar comenzile referitoare la producerea unei extensii (aplicație, modul), instalarea și actualizarea extensiilor, gestionarea acestora (aici putem dezactiva, respectiv dezinstala o extensie, putem vedea versiunea instalată) precum și informații despre extensiile instalate. Extensiile instalate cel mai frecvent de către utilizatori sunt „Text Mining", „Web Mining" și „Operator Toolbox". O listă a tuturor extensiilor poate fi găsită online[28], acestea fiind organizate pe categorii, respectiv tipuri de analiză și număr de descărcări.

Din meniul Help (Figura 3.4-5) putem accesa tutorialele locale RapidMiner (cele care apar și în fereastra de întâmpinare Learn), resursele online de învățare (RapidMiner Academy), resursele oferite tot online de către comunitate, documentația oficială a programului, respectiv putem primi asistență online din partea angajaților companiei.

Figura 3.4-5. Meniurile Connections, Extensions și Help



[28] https://marketplace.rapidminer.com/UpdateServer/faces/index.xhtml

4. Accesarea datelor (Data Access)

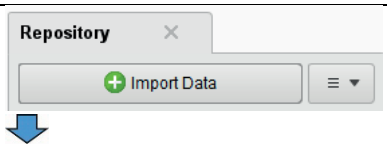
Accesarea datelor (într-un sens larg) se poate face cu ajutorul operatorilor grupați în folderul Data Access. Funcție de tipul și locația datelor, putem folosi o categorie sau alta a acestor operatori. Astfel, operatorii Files sunt utili pentru a lucra cu diferite tipuri de fișiere stocate pe PC (încărcare / citire și salvare / scriere), operatorii Database fac același lucru dar pentru baze de date, operatorii Applications permit accesarea unor date oferite de diferite aplicații precum Twitter sau Tableau, iar operatorii Cloud Storage ne ajută să ne conectăm la date stocate în cloud. Pe lângă aceste categorii, tot aici putem accesa și câțiva operatori relativ la Repository (redenumire, copiere, ștergere, mutare a depozitului de date), respectiv la încărcarea și salvarea unui set de date în format RapidMiner (Retrieve și Store).

4.1. Depozitul de date (Repository)

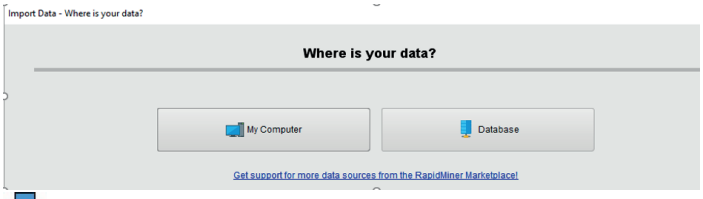
Înainte de a utiliza un set de date care nu este în format RapidMiner (rmhdf5table) trebuie să-l importăm. Putem să importăm manual un set folosind butonul „Import Data" din panelul Repository. Alternativ, putem folosi drag&drop (tragem fișierul respectiv în panelul Process). Putem alege să importăm un set de date sau un tabel (o selecție) dintr-o bază de date. În Figura 4.1-1 am prezentat pas cu pas un exemplu de importare manuală a unui set de date Excel. După cum se poate observa, procesul de importare este asistat de RapidMiner și este relativ simplu. Însă, dacă ar trebui să reluăm acești pași de mai multe ori, foarte probabil am prefera să lucrăm cu un proces (sintaxă). Astfel, nu ar mai trebui de fiecare dată să definim calea spre fișierul dorit, să indicăm numele fișierelor sau să definim nivelele de măsurare și rolurile atributelor.

Figura 4.1-1. Importarea manuală a unui set de date

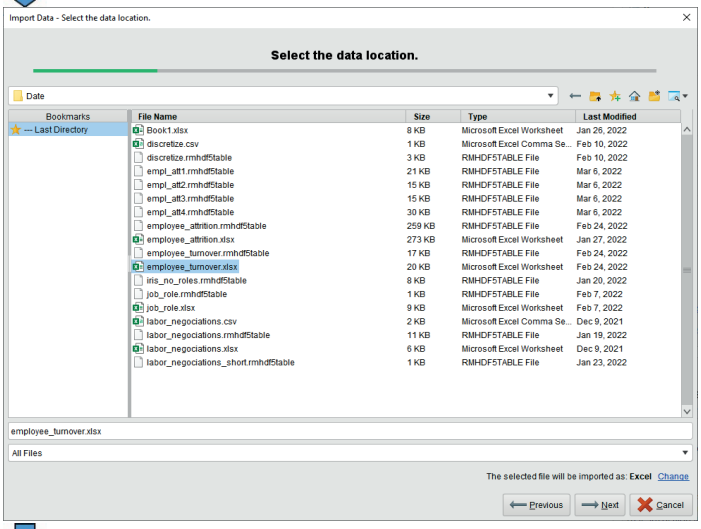
Pasul 1:
Import Data



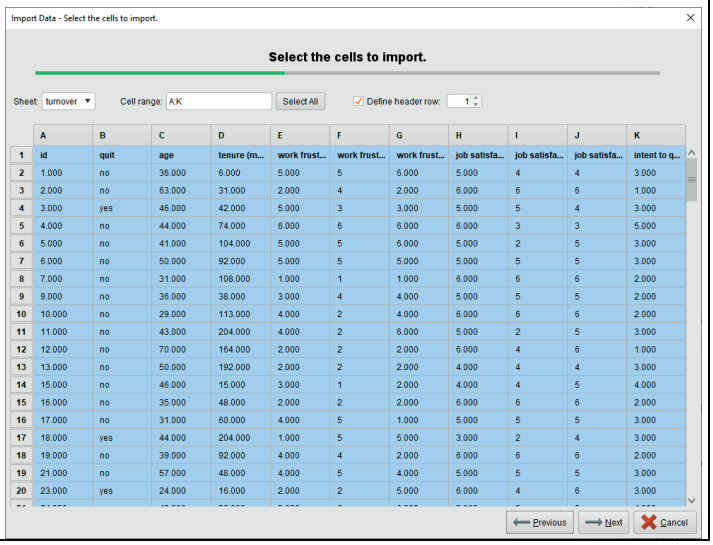
Pasul 2:
Alegem locația datelor (My Computer în acest caz) și categoria majoră: set de date (în acest exemplu) sau bază de date.



Pasul 3:
Alegem setul de date, apoi Next.
Selectăm fișierul Excel „employee_turnover.xlsx".

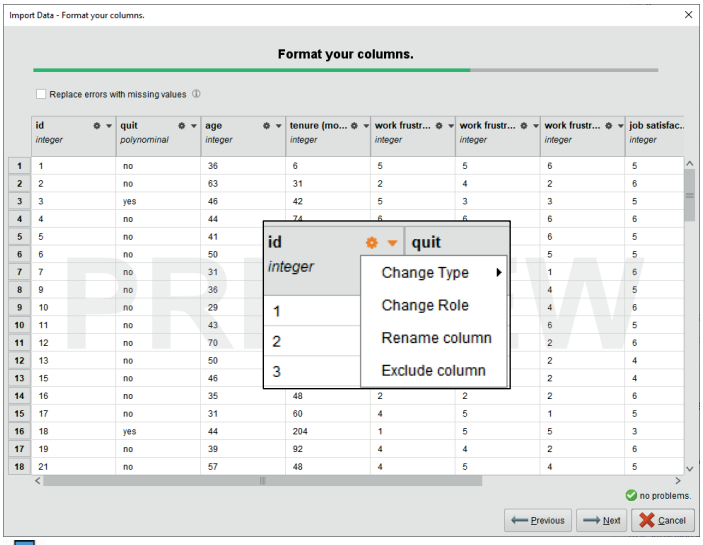


Pasul 4:
Facem selecțiile dorite, apoi Next.
RapidMiner a identificat automat că dorim să importăm toate coloanele din foaia turnover și că numele atributelor apare pe prima linie.

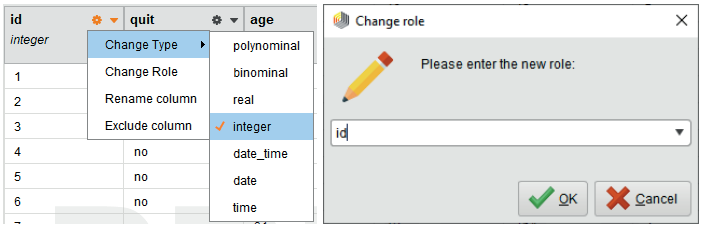


	A	B	C	D	E	F	G	H	I	J	K
1	id	quit	age	tenure (m...	work frust...	work frust...	work frust...	job satisfa...	job satisfa...	job satisfa...	intent to q...
2	1.000	no	36.000	6.000	5.000	5	6.000	5.000	4	4	3.000
3	2.000	no	63.000	31.000	2.000	4	2.000	6.000	6	6	1.000
4	3.000	yes	46.000	42.000	5.000	3	3.000	5.000	5	4	3.000
5	4.000	no	44.000	74.000	6.000	6	6.000	6.000	3	3	5.000
6	5.000	no	41.000	104.000	5.000	5	6.000	5.000	2	5	3.000
7	6.000	no	50.000	92.000	5.000	5	5.000	5.000	5	5	3.000
8	7.000	no	31.000	108.000	1.000	1	1.000	6.000	6	6	2.000
9	9.000	no	36.000	38.000	3.000	4	4.000	5.000	5	5	2.000
10	10.000	no	29.000	113.000	4.000	2	4.000	6.000	6	6	2.000
11	11.000	no	43.000	204.000	4.000	2	6.000	5.000	2	5	3.000
12	12.000	no	70.000	164.000	2.000	2	2.000	6.000	4	6	1.000
13	13.000	no	50.000	192.000	2.000	2	2.000	4.000	4	4	3.000
14	15.000	no	46.000	15.000	3.000	1	2.000	4.000	4	5	4.000
15	16.000	no	35.000	48.000	2.000	2	2.000	6.000	6	6	2.000
16	17.000	no	31.000	60.000	4.000	5	1.000	5.000	5	5	3.000
17	18.000	yes	44.000	204.000	1.000	5	5.000	3.000	2	4	3.000
18	19.000	no	39.000	92.000	4.000	4	2.000	6.000	6	6	2.000
19	21.000	no	57.000	48.000	4.000	5	4.000	5.000	5	5	3.000
20	23.000	yes	24.000	16.000	2.000	2	5.000	6.000	4	6	3.000

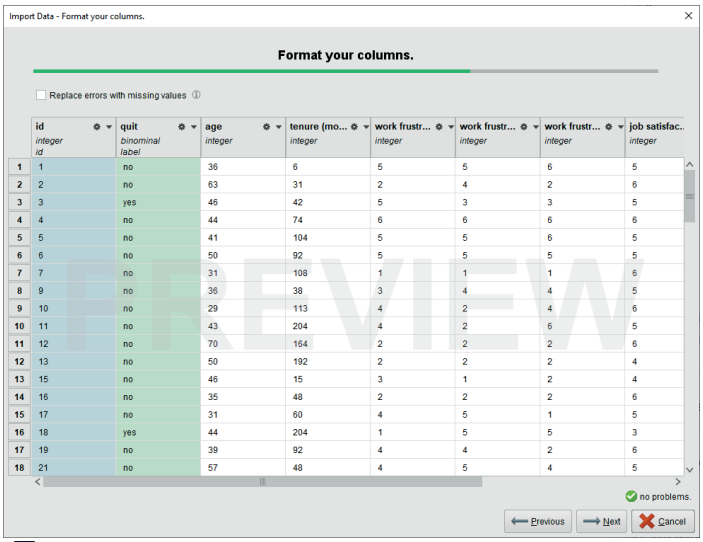
Pasul 5:
Definim atributele (coloanele), apoi alegem Next.
RapidMiner ghicește nivelul de măsurare al atributelor. Putem alege să schimbăm nivelul, să redenumim o coloană, să o excludem, să-i atribuim un rol.



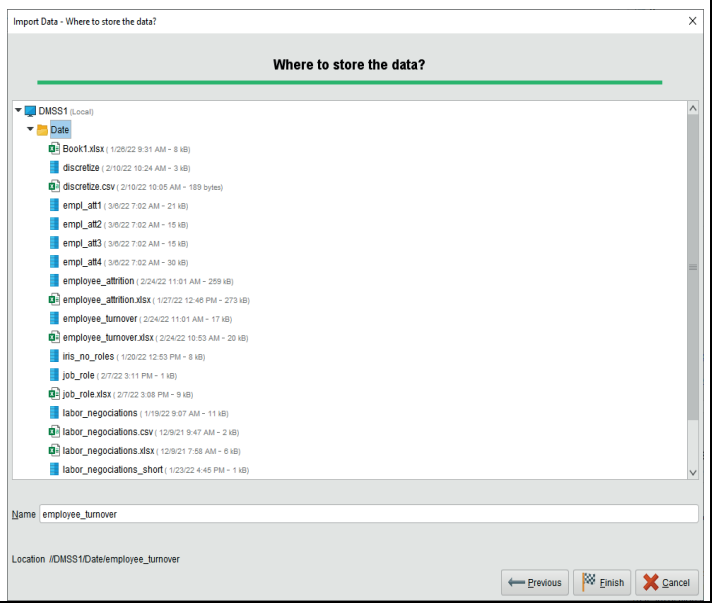
Pasul 6:
Atributul id este de tip număr întreg (integer). Îi atribuim rolul de id (identificare).



Pasul 7:
Atributul quit era inițial de tip polinominal, fără un rol special. Acum e binominal și are rolul de label (variabilă dependentă).
Atributele speciale (id, label) au un fundal colorat.



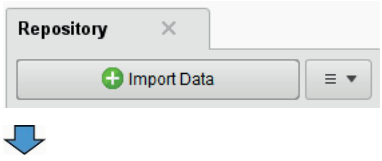
Pasul 8:
Denumim fișierul și alegem locația, apoi apăsăm butonul Finish.



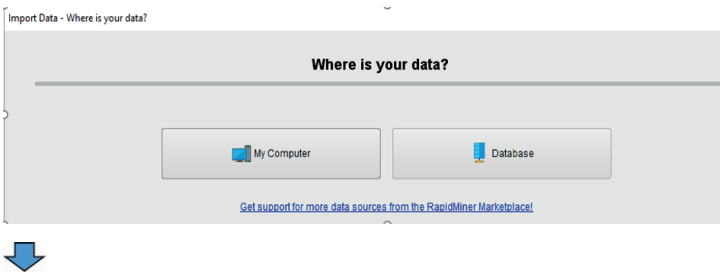
Pentru a importa manual un tabel sau o selecție (atribute și cazuri din unul sau mai multe tabele) dintr-o bază de date urmăm pașii din Figura 4.1-2.

Figura 4.1-2. Importarea manuală a unor date dintr-o bază de date

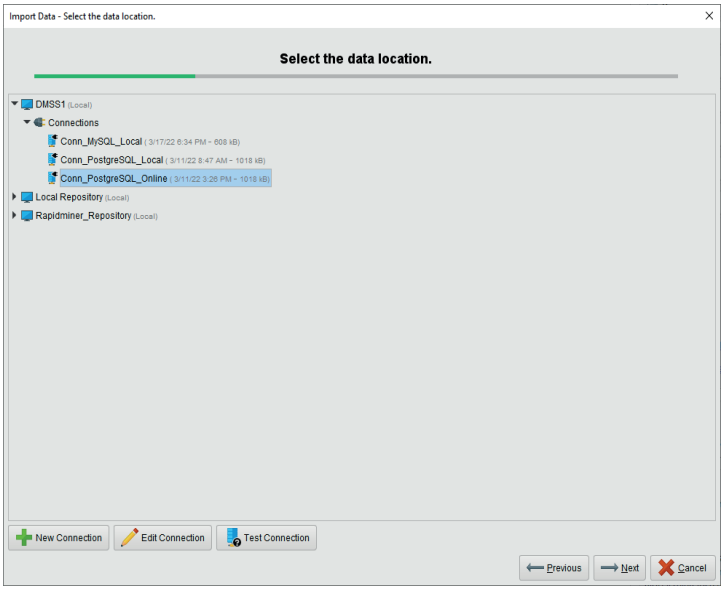
Pasul 1:
Import Data



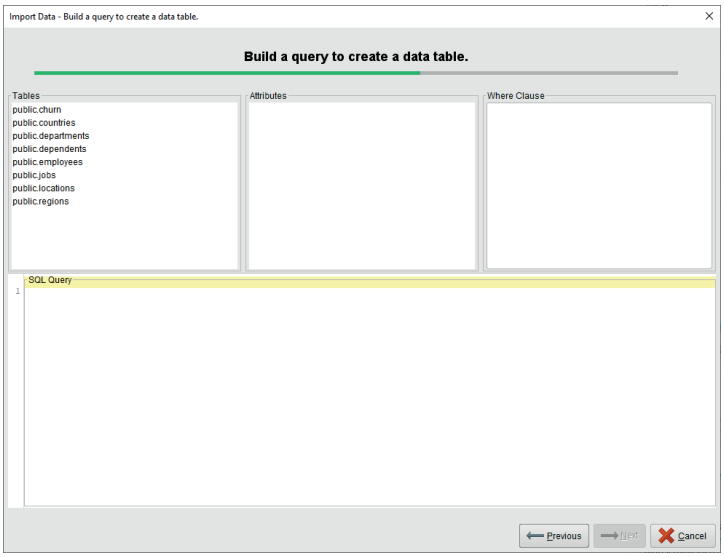
Pasul 2:
Alegem Database (bază de date).



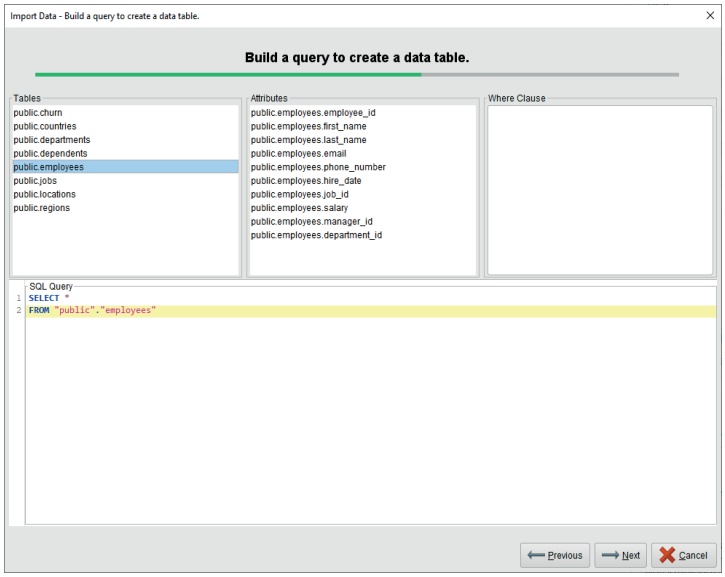
Pasul 3:
Alegem conexiunea (aceasta trebuie definită anterior) apoi Next.



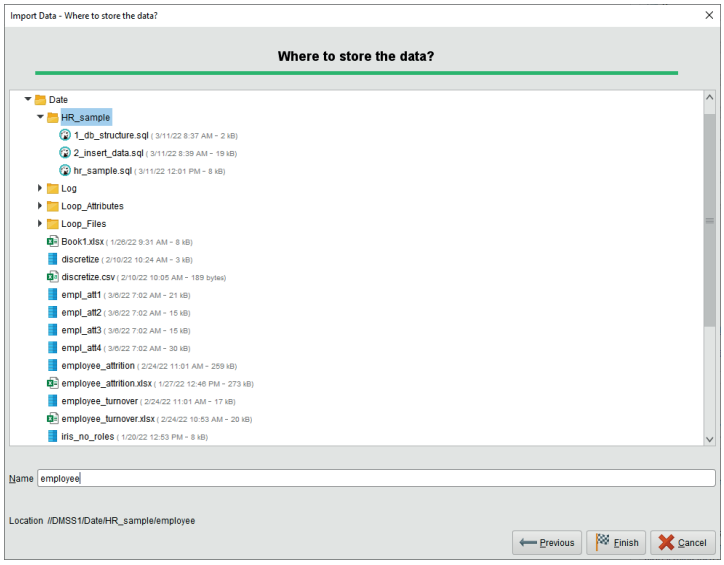
Pasul 4:
Observăm că tabele incluse în baza de date apar în prima fereastră. Celelalte ferestre sunt Attributes, Where Clause și SQL Query. Ele ne ajută să facem selecția dorită a datelor.



Pasul 5:
Facem selecțiile dorite, apoi Next.
În acest caz am selectat tabelul employees (toate atributele și cazurile).
Pentru a realiza acest lucru am selectat cu cursorul tabelul respectiv, iar atributele și formatul SQL al interogării au apărut automat.



Pasul 6:
Indicăm locația în care dorim să salvăm setul de date și numele acestuia apoi alegem Finish.
Setul de date va fi salvat în format RapidMiner (rmhdf5table).

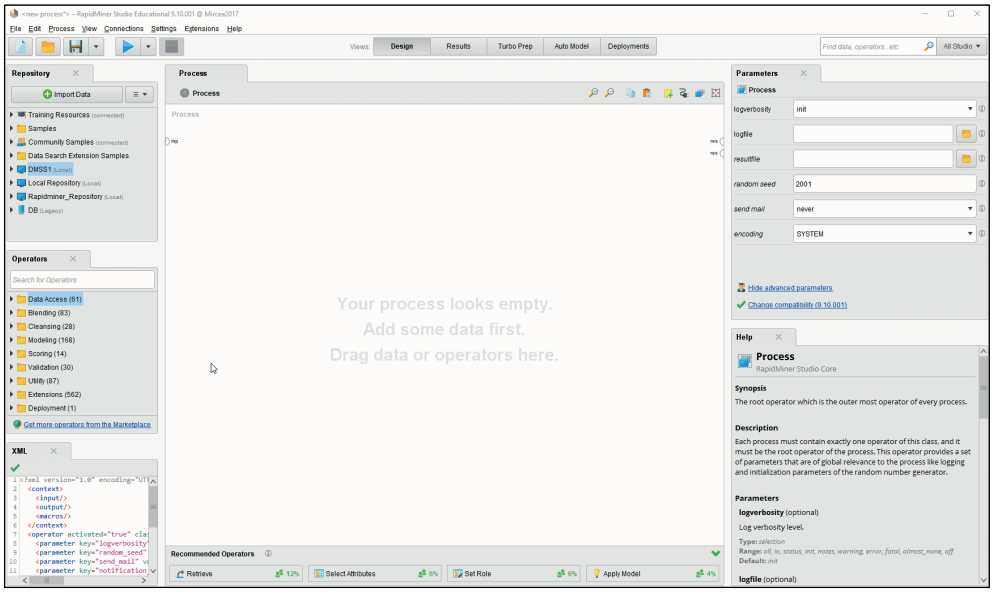


4.2. Încărcarea și salvarea unui set de date RapidMiner (Retrieve & Store)

Pentru a încărca un set de date RapidMiner folosim operatorul Retrieve. Aducem acest operator în fereastra de lucru și realizăm conexiunile (unim porturile out=output și res=results folosind mouse-ul). Operatorul Retrieve are asociat un singur parametru, locația fișierului (repository entry). Cu ajutorul acestui parametru setăm calea spre setul de date pe care dorim să-l folosim în analiză. Pentru a vizualiza setul de date trebuie să rulăm procesul apăsând butonul de Play (procesul va rula local).

Pentru a salva un set de date folosim operatorul Store. Acesta trebuie să fie conectat cu operatorul Retrieve, respectiv cu portul res. Conexiunile se realizează simplu, cu mouse-ul, trasând o linie între cele două puncte de legătură. Store are un singur parametru ce trebuie definit, locația (repository entry), adică numele fișierului și folderul în care acesta urmează să fie salvat.

Gif 4.2-1. Încărcarea și stocarea unui set de date de tip RapidMiner (Retrieve & Store)



În imaginile dinamice din Gif 4.2-1 am ilustrat încărcarea unui set de date (Iris) din folderul „Samples/data” și salvarea lui în folderul „Local Repository/data”. În ambele situații, setul de date este de tip RapidMiner,

adică are extensia rmhdf5table. Setul de date Iris este inclus automat în program, la instalare. Dat fiind faptul că locația (calea; path în engleză) acestui set de date este întotdeauna aceeași, operatorul Retrieve recunoaște versiunea scurtă a acesteia, și anume „//Samples/data/Iris", unde // indică faptul că este vorba de o cale relativă (este definită în relație cu folderul în care este situat setul de date). Similar, operatorul Store recunoaște calea scurtă „//Local Repository/data/Iris". Setul de date este vizualizat doar în urma rulării procesului (butonul Play).

Utilizarea unor căi de tip relativ este utilă în cazul în care dorim să mutăm procesul în alt folder și/sau să reluăm analiza pe alt PC. Dacă toate căile sunt de tip relativ, procesul va rula indiferent de locația în care este mutat (folder și/sau PC). Dacă procesul conține căi de tip absolut[29], după mutarea acestuia în altă locație, vor apărea erori la rulare. Pentru a elimina aceste erori, va trebui să redefinim manual toate căile de tip absolut. Pentru a evita acest lucru, soluția cea mai simplă este să salvăm procesul înainte de includerea unui operator. În acest caz, RapidMiner va salva automat locațiile (căile / paths) în forma lor relativă (în cazul tuturor operatorilor care au un parametru care permite definirea unei locații).

4.3. Lucrul cu fișiere de date (Files)

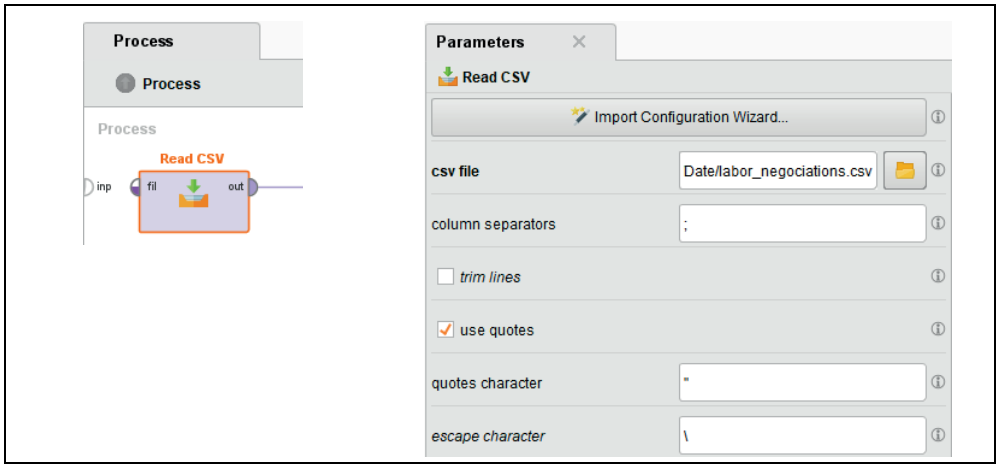
Cel mai adesea, seturile de date pe care dorim să le analizăm sunt în alte formate decât RapidMiner. Formatele utilizate frecvent în context organizațional sunt csv (comma separated values) și xls/xlsx (Microsoft Excel). Relativ mai rar sunt utilizate și formate precum sav (SPSS), dat (Stata), dbf (Dbase) sau accdb (Microsoft Access). RapidMiner poate citi și salva date din / în toate aceste formate și multe altele (arff, sparse, xml, url etc.). Operatorii care fac acest lucru sunt localizați în fereastra Operators, folderul „Data Access/Files".

[29] De exemplu „C:\Users\Mircea Comsa\.RapidMiner\repositories\Local Repository\data".

Pentru a încărca un fișier de date de tip csv (pentru alte formate de fișiere logica este similară) parcurgem următorii pași (Figura 4.3-1):

- drag&drop operatorul „Read CSV”;
- realizarea conexiunii dintre output (portul out) și rezultate (portul res);
- configurarea parametrilor: specificarea locației fișierului pe care dorim să-l încărcăm (parametrul csv file; definirea acestui parametru este obligatorie), respectiv a altor parametri relativ la acest fișier (specificarea caracterului utilizat pentru separarea coloanelor, caracterul utilizat pentru citate – quotes caracter, caracterul utilizat pentru a ignora caracterul quotes – escape caracter, linia de start, caracterul utilizat pentru delimitarea zecimalelor etc.);
- rularea procesului.

Figura 4.3-1. Încărcarea unui set de date de tip csv (Read CSV)

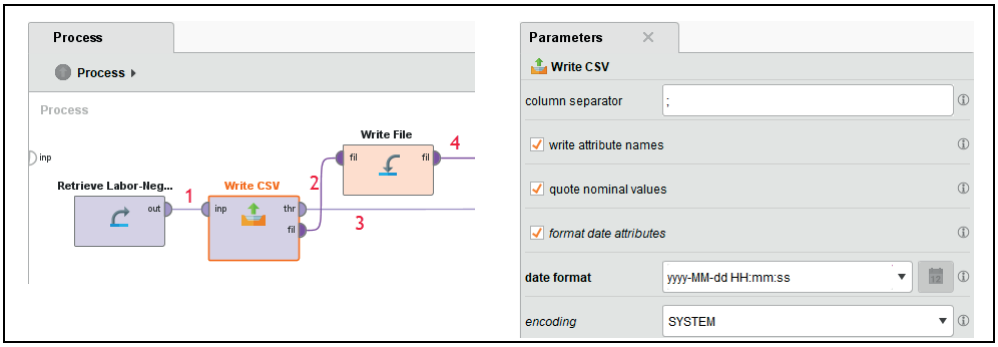


Salvarea unui fișier de date în format csv se realizează cu ajutorul operatorului „Write CSV”. Pentru a ilustra acest proces considerăm că avem un set de date în format RapidMiner (datele pot fi în orice alt format) și urmăm pașii din Figura 4.3-2:

- drag&drop operatorii „Write CSV” și „Write File” (e nevoie de acest operator în toate situațiile în care dorim să salvăm efectiv un fișier pe hard-disk);

- realizarea conexiunilor: (1) pentru a scrie un obiect de tip csv; (2) pentru a trimite obiectul creat spre operatorul „Write File”; (3) pentru a vizualiza setul de date inițial; (4) pentru a salva pe hard-disk fișierul Excel;
- configurarea parametrilor aferenți fiecărui operator: în cazul operatorului „Write CSV” putem seta caracterul utilizat pentru separarea coloanelor, dacă să includem sau nu numele atributelor, formatul pentru variabilele de tip dată etc.;
- rularea procesului.

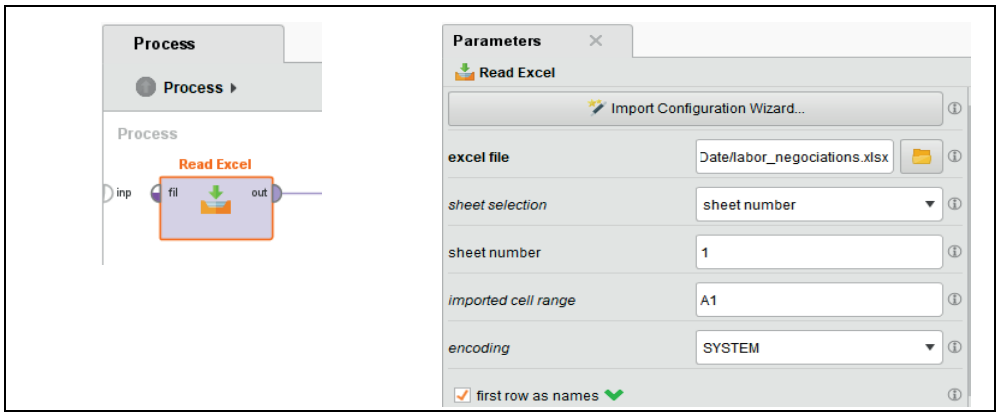
Figura 4.3-2. Salvarea unui set de date în format csv (Write CSV)



Pentru a încărca un fișier de date Excel parcurgem următorii pași (Figura 4.3-3):

- drag&drop operatorul „Read Excel”;
- realizarea conexiunii dintre output (portul out) și rezultate (portul res);
- configurarea parametrilor: specificarea locației fișierului pe care dorim să-l încărcăm (parametrul excel file; definirea acestui parametru este obligatorie), respectiv a altor parametri relativ la acest fișier (numele sau numărul de ordine a foii (sheet) care conține datele, zona în care apar datele, tipul de encoding, preluarea numelor variabilelor de pe prima linie etc.);
- rularea procesului.

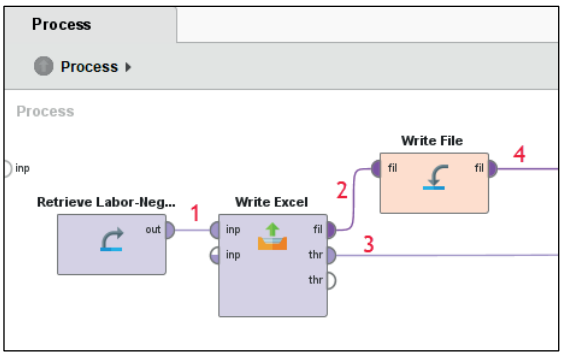
Figura 4.3-3. Încărcarea unui set de date de tip Excel (Read Excel)



Salvarea unui fișier de date în format excel se poate face cu ajutorul operatorului Write Excel. Pentru a ilustra acest proces presupunem că avem un set de date în format RapidMiner (datele pot fi în orice alt format) și urmăm pașii din Figura 4.3-4:

- drag&drop operatorii „Write Excel" și „Write File" (e nevoie de acest operator în toate situațiile în care dorim să salvăm efectiv un fișier pe hard-disk);
- realizarea conexiunilor: (1) pentru a scrie un obiect de tip excel; (2) pentru a trimite obiectul creat spre operatorul „Write File"; (3) pentru a vizualiza setul de date inițial; (4) pentru a salva pe hard-disk fișierul excel;
- configurarea parametrilor aferenți fiecărui operator;
- rularea procesului.

Figura 4.3-4. Salvarea unui set de date în format Excel (Write Excel)



4.4. Lucrul cu baze de date (Database)

Pentru a putea lucra cu o bază de date în RapidMiner e nevoie să setăm o conexiune cu aceasta (Connection). Pentru fiecare tip de bază de date trebuie setată o conexiune specifică (de exemplu, SQL dacă baza este SQL). De asemenea, pentru fiecare bază de date dintr-un anumit tip trebuie să setăm o conexiune separată (de exemplu, o conexiune SQL pentru baza X și altă conexiune SQL pentru baza Y).

Operatorii disponibili în RapidMiner Studio permit citirea, scrierea și actualizarea unei baze de date (Read Database, Write Database, Update Database). Aici vom prezenta doar operatorul „Read Database". Anterior oricărei comenzi relativ la o bază de date, trebuie să definim conexiunea asociată acesteia.

Setarea unei conexiuni (Connection)

O conexiune este un obiect salvat de obicei în folderul Connections asociat unui Repository. Fiecare conexiune conține cel puțin următoarele informații: numele bazei de date, driverul asociat acesteia, numele utilizatorului, parola, adresa serverului (host) și portul. Serverul poate fi local sau online. Tipurile de conexiuni ce pot fi setate sunt următoarele:

- baze de date SQL (JDBC);
- baze de date No SQL și Cloud;
- servicii în Cloud;
- servere email;
- servere securizate Shell.

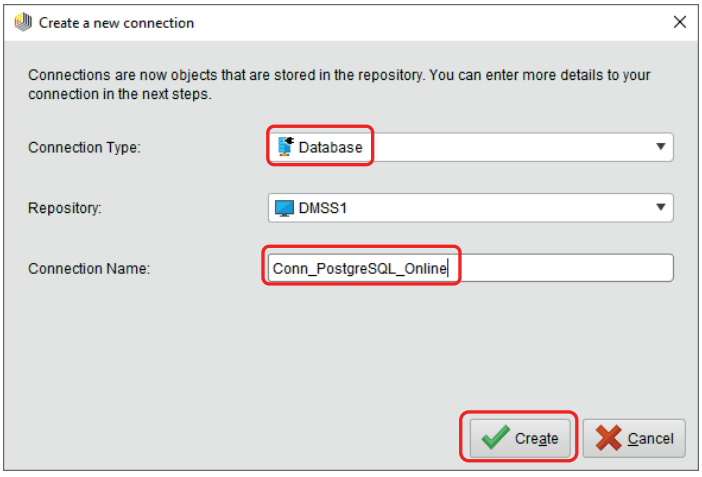
În acest context discutăm doar despre setarea unei conexiuni la o bază de date SQL (PostgreSQL) stocată pe un server online. Pentru acest exemplu am postat o bază de date pe un server online. Toate informațiile necesare pentru a realiza o conexiune la această bază de date au fost salvate în conexiune, deci pot fi folosite de oricine are acest fișier. Desigur, conexiunea va putea fi utilizată atât timp cât baza de date și serverul sunt funcționale. Informațiile

necesare pentru setarea unei astfel de conexiuni sunt relativ standard și pot fi aflate de la administratorul bazei.

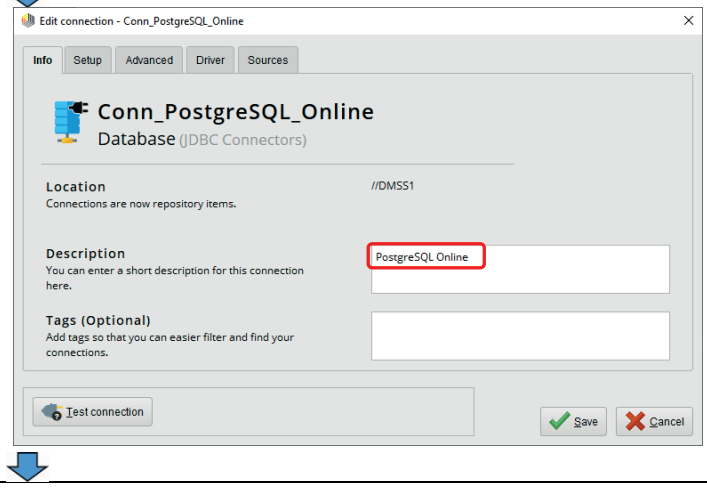
Primul pas în setarea unei conexiuni la o bază de date constă în alegerea unui nume pentru conexiune și a tipului acesteia (Database în acest caz). În continuare oferim o descriere și câteva cuvinte cheie, apoi introducem informațiile cu privire la server, bază, utilizator, locația driverului specific acelei baze. La final testăm conexiunea și o salvăm (Figura 4.4-1).[30]

Figura 4.4-1. Setarea unei conexiuni cu o bază de date PostgreSQL (server online)

Pasul 1:
În Repository DMSS1 apăsăm click dreapta pe „Connections" și alegem „Create Connection".
La „Connection Type" alegem Database.
La „Connection Name" dăm un nume conexiunii respective.
Apăsăm „Create".

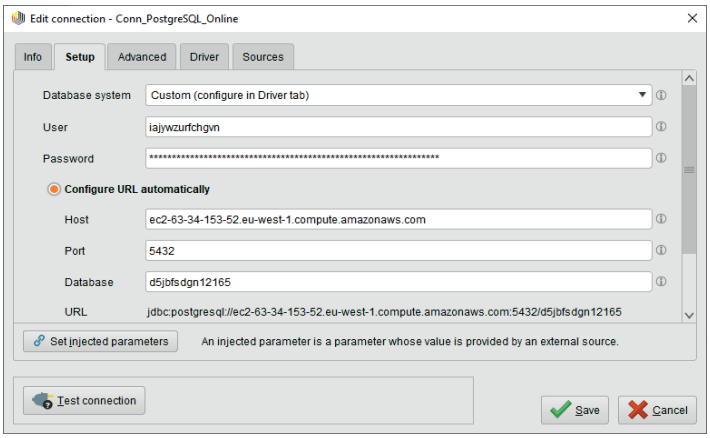


Pasul 2:
La tabul Info putem introduce o descriere a conexiunii și câteva cuvinte cheie (tags).

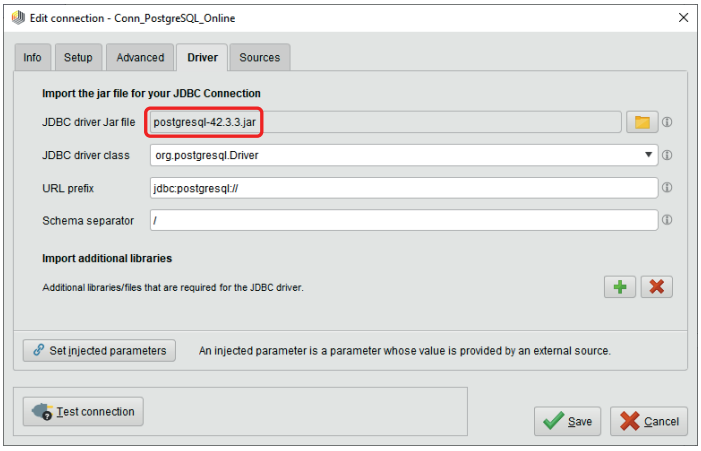


[30] Prezentare video: https://academy.rapidminer.com/learn/video/connecting-to-databases.

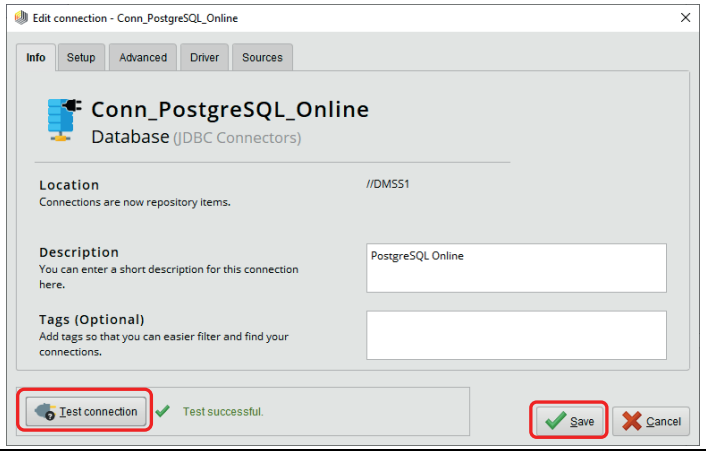
Pasul 3:
La tabul Setup introducem datele cerute. În cazul de față conexiunea conține deja toate aceste informații (ele sunt salvate în conexiune și pot fi utilizate de oricine atât timp cât baza de date și serverul sunt funcționale).



Pasul 4:
La tabul Driver indicăm locația driverului JDBC (în acest caz am inclus fișierul jar respectiv în folderul Connections).



Pasul 5:
Testăm conexiunea (Test connection), observăm că funcționează, apoi o salvăm (Save).



În folderul Connections am inclus și o conexiune la o bază de date PostgreSQL postată pe un server local. Acesta nu va fi funcțională pe alte PC-uri. Cei interesați pot realiza o conexiune similară funcțională urmând pașii următori (desigur, sunt posibile și alte soluții):

- descărcăm fișierul de instalare a softului postgresql „interactive installer postgresql" (https://www.postgresql.org/download/windows/);
- instalăm softul[31]; atenție, alegem versiunea potrivită pentru sistemul de operare (aici am ales Windows);
- creăm un server[32];
- deschidem softul pdAdmin și încărcăm baza de date hr_sample; am inclus în folderul Database baza de date respectivă; alternativ, putem genera structura (schema) bazei de date și să o populăm cu date folosind scripturile incluse în același folder;
- descărcăm versiunea de driver JDBC potrivită pentru versiunea Java instalată pe PC (https://jdbc.postgresql.org/download.html);
- definim conexiunea urmând pașii specificați în Figura 4.4-1; atenție, informațiile relativ la bază, utilizator, parolă etc. vor fi altele în acest caz.

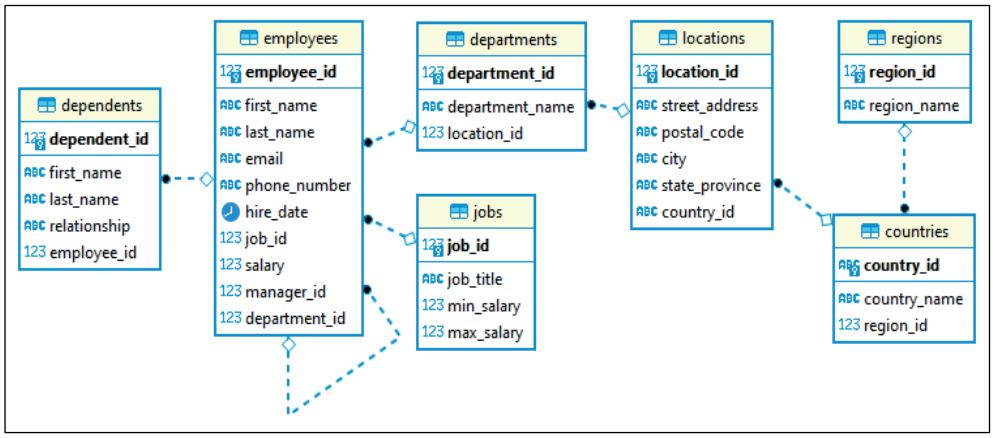
Citirea unei baze de date (Read Database)

Baza de date folosită în cadrul acestui exemplu este doar una de test. Este o bază SQL, PostgreSQL mai exact. Schema acesteia, adică tabelele, conexiunile dintre ele și atributele din fiecare tabel sunt prezentate în Figura 4.4-2. Baza este găzduită pe un server online (procedăm la fel și dacă serverul este local, doar setările conexiunii vor fi diferite).

[31] O descriere a instrucțiunilor de instalare se poate consulta la adresa: https://www.enterprisedb.com/docs/supported-open-source/postgresql/installer/02_installing_postgresql_with_the_graphical_installation_wizard/01_invoking_the_graphical_installer/

[32] O serie de instrucțiuni cu privire la realizarea acestui pas pot fi consultate la adresa: https://docs.rapidminer.com/9.8/legacy/install/database_setup/creating_postgres_db.html

Figura 4.4-2. Schema bazei de date „hr_sample”

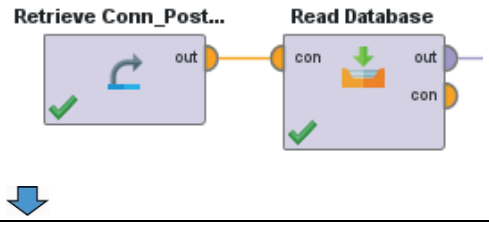


Pentru a citi în RapidMiner o bază de date PostgreSQL postată pe un server online avem nevoie de o conexiune la acea bază („Conn_PostgreSQL_Online”, folderul Connections) și operatorul „Read Database”. Pentru a defini informațiile pe care dorim să le citim folosim parametrul „define query”. Aici putem alege între:

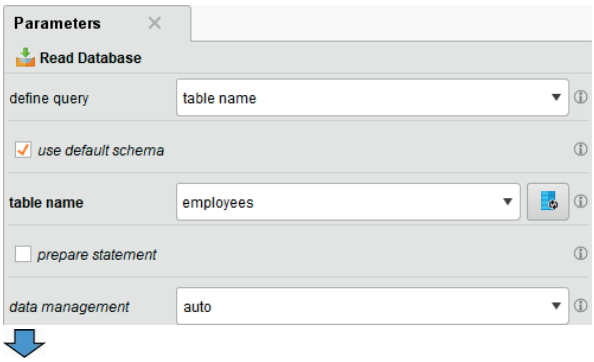
- **query**: interogarea SQL - trebuie să o definim, adică să scriem comenzile specifice SQL; putem selecta astfel o parte a atributelor, cazurilor, din unul sau mai multe tabele;
- **query file**: fișierul care definește interogarea - trebuie să indicăm fișierul care conține comenzile specifice SQL;
- **table name**: numele tabelului; alegem tabelul pe care dorim să-l citim (în general, o bază de date conține mai multe tabele).

Figura 4.4-3. Citirea unui tabel dintr-o bază de date PostgreSQL (server online)

Pasul 1:
Din Repository DMSS1, folderul „Connections”, încărcăm conexiunea setată anterior „Conn_PostgreSQL_Online” apoi o conectăm cu operatorul „Read Database”.



Pasul 2:
La parametrul „define query" alegem „table name". Specificăm astfel că dorim să citim unul dintre tabelele (seturile de date) din baza de date la care ne-am conectat.
La parametrul „table name" alegem employees (numele tabelului / setului de date).



Pasul 3:
Observăm că tabelul respectiv a fost încărcat în RapidMiner.

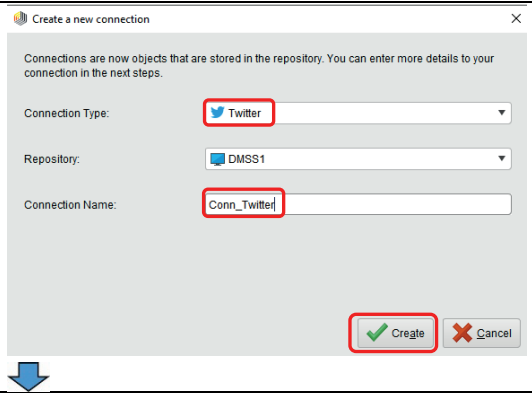
employee_id	first_name	last_name	email	phone_num...
100	Steven	King	steven.kin...	515.123.4567
101	Neena	Kochhar	neena.koc...	515.123.4568
102	Lex	De Haan	lex.de haa...	515.123.4569
103	Alexander	Hunold	alexander....	590.423.4567
104	Bruce	Ernst	bruce.erns...	590.423.4568
105	David	Austin	david.austi...	590.423.4569

4.5. Lucrul cu aplicații (Applications)

Pentru a putea folosi în RapidMiner Studio date din alte aplicații (precum Twitter), primul pas constă în realizarea unei conexiuni între RapidMiner Studio și aplicația respectivă. Succesiunea de imagini de mai jos (Figura 4.5-1) ilustrează pașii necesari pentru a realiza o conexiune cu aplicația Twitter.

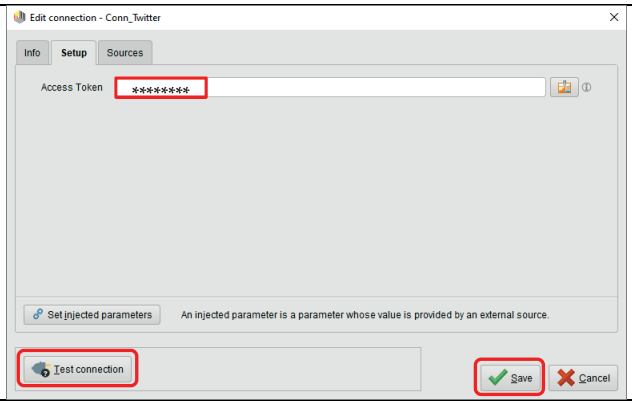
Figura 4.5-1. Conectarea la Twitter

Pasul 1:
În Repository DMSS1 apăsăm click dreapta pe „Connections" și alegem „Create Connection".
La „Connection Type" alegem Twitter.
La „Connection Name" dăm un nume conexiunii respective.
Apăsăm „Create".



Pasul 2:

Introducem tokenul de acces pentru aplicația Twitter (trebuie să aveți un cont obișnuit de Twitter, respectiv unul de dezvoltator Twitter[33]). Testăm și salvăm conexiunea.

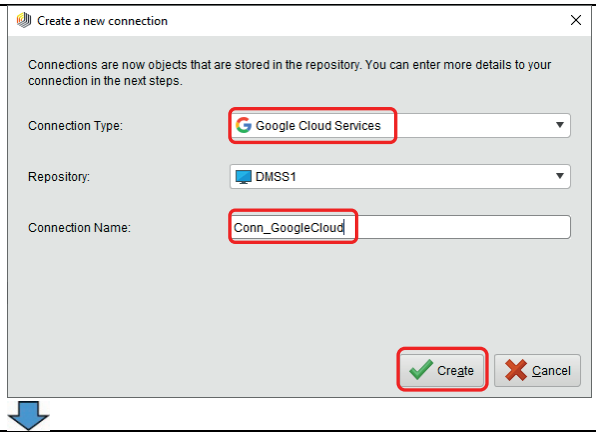


4.6. Accesarea datelor stocate în cloud (Cloud Storage)

Pentru a accesa date stocate în cloud, prima dată trebuie să definim o conexiune cu aplicația care oferă acest serviciu. Pașii sunt similari cu cei descriși în secțiunea anterioară. Îi ilustrăm aici în cazul aplicațiilor Google Cloud și Dropbox. În ambele situații, este necesar să obținem tokenul de acces înainte de a ne conecta la serviciile / aplicațiile respective.

Figura 4.6-1. Conectarea la Google Cloud

Pasul 1:
În Repository DMSS1 apăsăm click dreapta pe „Connections" și alegem „Create Connection".
La „Connection Type" alegem „Google Cloud Services".
La „Connection Name" dăm un nume conexiunii respective.
Apăsăm „Create".



[33] https://developer.twitter.com/en/docs/twitter-api/getting-started/getting-access-to-the-twitter-api

Pasul 2:
Introducem datele necesare (ID proiect și tokenul de acces pentru Google Cloud).
Testăm conexiunea (Test connection).
Salvăm conexiunea (Save).

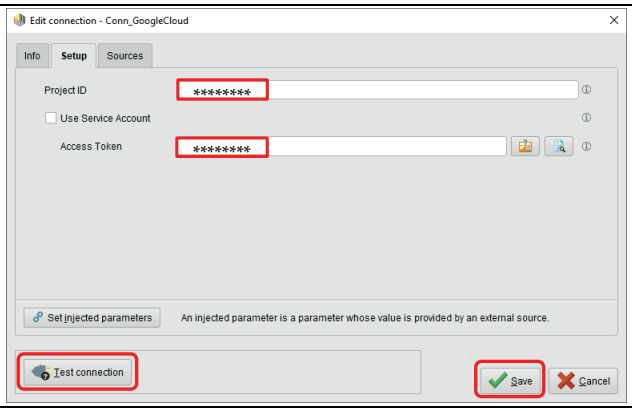
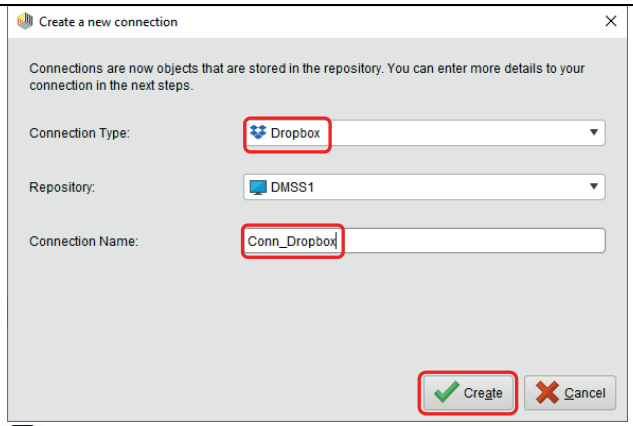


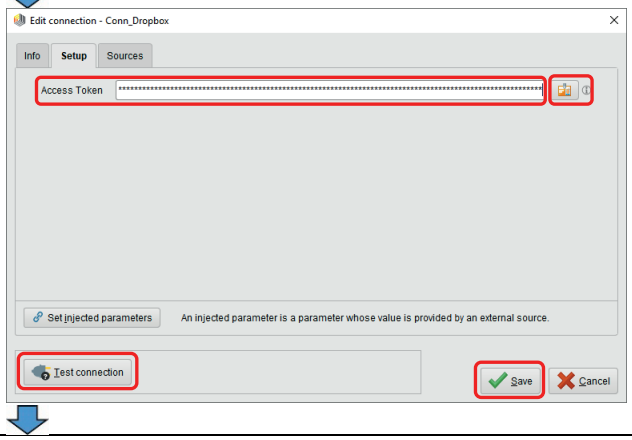
Figura 4.6-2. Conectarea la Dropbox și citirea unui fișier cu date

Pasul 1:
În Repository DMSS1 apăsăm click dreapta pe „Connections" și alegem „Create Connection".
La „Connection Type" alegem Dropbox.
La „Connection Name" dăm un nume conexiunii respective.
Apăsăm „Create".



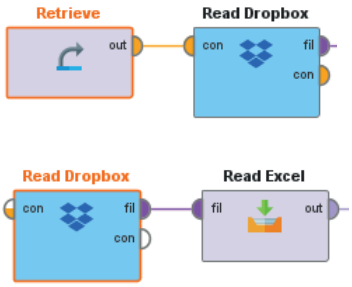
Pasul 2:

Introducem tokenul de acces.[34]
Apăsăm butonul mic din dreapta tokenului (Request access token) și urmăm cei doi pași.
Testăm conexiunea (Test connection).
Salvăm conexiunea (Save).



[34] https://developers.dropbox.com/oauth-guide

Pasul 3:
Încărcăm operatorul Retrieve și indicăm Dropbox la numele conexiunii.
Conectăm operatorul „Read Dropbox" și indicăm numele fișierului RapidMiner (în acest caz, labor_negociations).
Pentru a citi fișiere de alt tip, folosim operatorul „Read Dropbox" (indicăm numele conexiunii și al fișierului), apoi conectăm operatorul care poate citi acel tip de fișier (xlsx în acest caz).



5. Lucrul cu atribute, cazuri, tabele și valori (Blending)

Operatorii din categoria Blending sunt grupați în patru sub-categorii:

- **Attributes**: include operatorii care modifică atributele;
- **Examples**: include operatorii care modifică exemplele / cazurile;
- **Tables**: acești operatori facilitează lucrul cu tabele;
- **Values**: acești operatori facilitează lucrul cu valori.

5.1. Lucrul cu atribute (Attributes)

Informațiile relativ la cazurile statistice (acestea apar pe linii în tabele / fișierele cu date) sunt denumite generic variabile (apar pe coloane) în manualele de specialitate. În literatura pe tema metodologiei colectării datelor cantitative, variabilele sunt denumite uneori atribute. Tehnic, denumirea generală corectă este cea de variabilă, atributul fiind un tip particular de variabilă, și anume o variabilă non-metrică sau calitativă. Simplu spus, o astfel de variabilă, mai exact însușirea pe care această o evaluează („măsoară”[35]), nu are asociată o unitate de măsură. Astfel de variabile pot fi de tip nominal sau ordinal. O variabilă nominală este orice clasificare (gruparea cazurilor în clase în funcție de un criteriu). De exemplu, variabila sex este una nominală deoarece în funcție de acest criteriu distingem între două clase (categorii), bărbat și femeie, fiecare om aparținând uneia

[35] Termenul de măsoară apare aici între ghilimele pentru a indica faptul că nu este sinonim cu termenul de evaluare, fiind un tip specific de evaluare, unul care are asociat o unitate de măsură (Rotariu, 1991). În general, în literatură, această distincție nu apare sau nu este explicită, termenul de măsurare fiind folosit pentru toate tipurile de scale, non-metrice (fără unitate de măsură: nominală și ordinală) și metrice (cu unitate de măsură: interval și raport).

dintre acestea (nu se poate să nu aparțină niciunei clase, nici să aparțină mai multor clase). Dacă clasele rezultate în urma unei clasificări pot fi ordonate în funcție de intensitatea percepută a însușirii, variabila respectivă este de tip ordinal. În domeniul metodologiei măsurării din științele sociale, folosim denumirea de atribut pentru a ne referi la o variabilă de tip nominal sau ordinal, respectiv la una dintre clasele unei variabile de tip nominal, la o însușire a unui caz (bărbat, femeie, tânăr, bătrân, rezidență în urban etc.). Oarecum contrar acestor convenții, în RapidMiner Studio variabilele sunt numite atribute, prin urmare se referă la toate caracteristicile / însușirile cazurilor, indiferent de nivelul lor de evaluare / măsurare. În acest manual vom folosi interșanjabil cei doi termeni (variabilă și atribut[36]), dar vom avea în vedere sensul larg, cel definit de conceptul de variabilă.

În RapidMiner Studio comenzile relativ la atribute se regăsesc în fereastra Operators, secțiunea Blending/Attributes. Acești operatori pot fi folosiți pentru a defini numele atributelor, rolurile, tipurile, a selecta anumite atribute, a genera atribute, respectiv a le reordona (schimba ordinea lor în setul de date). În continuare vom prezenta și ilustra principalii operatori din fiecare categorie.

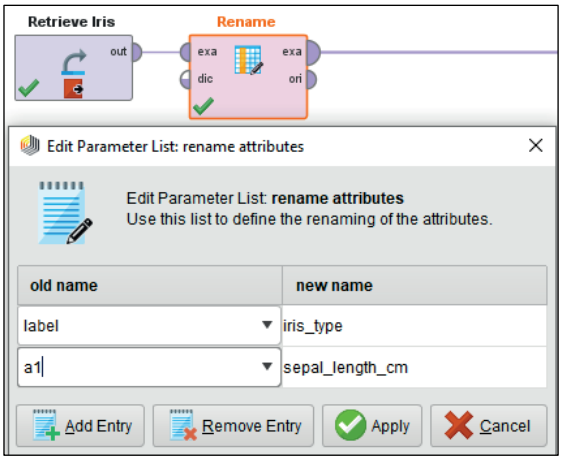
Numele și rolul atributelor (Names & Roles)

Operatorii Rename pot fi folosiți pentru redenumirea atributelor. Uneori acest lucru poate fi util, mai ales atunci când denumirile originale ale atributelor constau în litere și cifre, sunt neclare sau insuficient de informative. În imaginea din Figura 5.1-1 am ilustrat modalitatea de redenumire a unei variabile oferind un exemplu în care am schimbat numele a două atribute. Astfel, am redenumit atributul vechi „label" cu numele „iris_type", respectiv „a1" cu „sepal_length_cm". La start, operatorul Rename oferă posibilitatea de a redenumi o singură variabilă, dar putem indica mai multe variabile prin apăsarea butonului „Add Entry". Eliminarea unei redenumiri se face folosind „Remove Entry", iar finalizarea setărilor cu „Apply" (renunțarea cu „Cancel"). Pentru a observa impactul acestei

[36] În machine learning variabilele / atributele sunt numite features (trăsături, proprietăți măsurabile ale unui fenomen sau ale unei entități).

comenzi, procesul prezentat ca exemplu ne arată prima dată (după prima rulare = apăsare a butonului Play/Run ▶) setul de date cu denumirile inițiale, apoi, după încă o apăsare a aceluiași buton, setul cu denumirile schimbate. Rularea în doi pași a procesului a fost posibilă ca urmare a utilizării comenzii „Breakpoint after" (această comandă poate fi accesată rapid folosind click dreapta pe operatorul relativ la care dorim să inserăm o pauză; pentru a continua rularea apăsăm din nou butonul Play/Run ▶).

Figura 5.1-1. Redenumirea unui atribut (Rename)



Setarea rolului variabilelor dintr-un set de date este un pas esențial care trebuie realizat înaintea oricărei analize. Prin stabilirea rolurilor, comunicăm programului care sunt variabilele pe care trebuie să le includă în analiză și care sunt rolurile pe care aceste variabile le vor avea în analiza respectivă fără să fim nevoiți să facem acest lucru de fiecare dată când folosim acel set de date. Simplu spus, rolul fiecărei variabile este definit direct în setul de date. În RapidMiner Studio avem două tipuri principale de roluri: obișnuit (regular) și special (special = toate atributele care nu sunt de tip obișnuit). Mai detaliat, rolurile posibile pe care le poate avea un atribut sunt următoarele (Mierswa, 2016b, p. 14; RapidMiner, 2022, pp. 160–161):

- **regular** = independentă / obișnuită / predictor – reprezintă o variabilă care nu are un rol special; frecvent, o astfel de variabilă este utilizată

pentru a prezice valorile / apartenența la clasele unei variabile label / dependente;

- **id** = de identificare – este o variabilă specială care ajută la identificarea cazurilor; valorile pot fi formate din litere, numere, caractere speciale sau combinații ale acestora; fiecare caz are asociat un id unic; o variabilă de tip id este exclusă automat din orice model de predicție; o astfel de variabilă este absolut necesară atunci când dorim să unim două seturi de date, respectiv poate fi necesară, funcție de situația concretă, atunci când dorim să agregăm datele dintr-un set de date, să transpunem un set de date etc.;
- **label** = dependentă / prezisă / țintă / de interes (target / class) – este principala variabilă specială care ne interesează într-o analiză de predicție; scopul analizei constă în prezicerea cât mai corectă a valorilor acestei variabile în cazul unei variabile metrice, respectiv a apartenenței la clasele acestei variabile în cazul unei variabile non-metrice;
- **prediction** = predicție – variabilă specială care conține valorile prezise (relativ la variabila label / dependentă) de un model de predicție; dacă variabila label este metrică, predicția acesteia va fi tot o variabilă metrică; dacă variabila label este non-metrică (categorială), variabila prezisă va fi la fel și va avea asociate suplimentar două sau mai multe variabile prezise care indică probabilitatea apartenenței la fiecare dintre clasele variabilei label; o astfel de variabilă este obligatorie (alături de o variabilă de tip label) atunci când dorim să estimăm performanța unui model de predicție;
- **cluster** = grupare – variabilă specială care indică apartenența cazurilor la anumite grupuri rezultate în urma realizării unei analize cluster / de grupare;
- **weight** = ponderare – variabilă specială; conține valori numerice care indică ponderarea / importanța unui caz; de exemplu, dacă un caz are asociată valoarea 2 în cazul variabilei weight, în analiză acest caz va conta ca două cazuri (ca și cum setul de date ar conține două astfel de cazuri identice, nu doar unul); în acest context, o variabilă de tip weight este folosită pentru a aduce structura eșantionului la structura populației (sau la o altă structură dorită); de exemplu, dacă eșantionul are o pondere a persoanelor educate mai mare comparativ cu populația

de referință, utilizarea unei variabile de ponderare va reduce ponderea persoanelor educate (prin atribuirea unor valori sub-unitare) și va crește ponderea persoanelor mai puțin educate (au asociate valori supra-unitare); o astfel de variabilă poate fi folosită și atunci când evaluăm performanța unui model prin faptul că ne ajută să definim costuri diferite pentru erorile de clasificare; de exemplu, putem atribui un cost mai mare în cazul erorilor de tip 1 (fals negativ: situația reală este „X" dar modelul prezice că nu este „X", unde „X" se referă la categoria de interes – fraudă, părăsirea companiei, bolnav etc.);

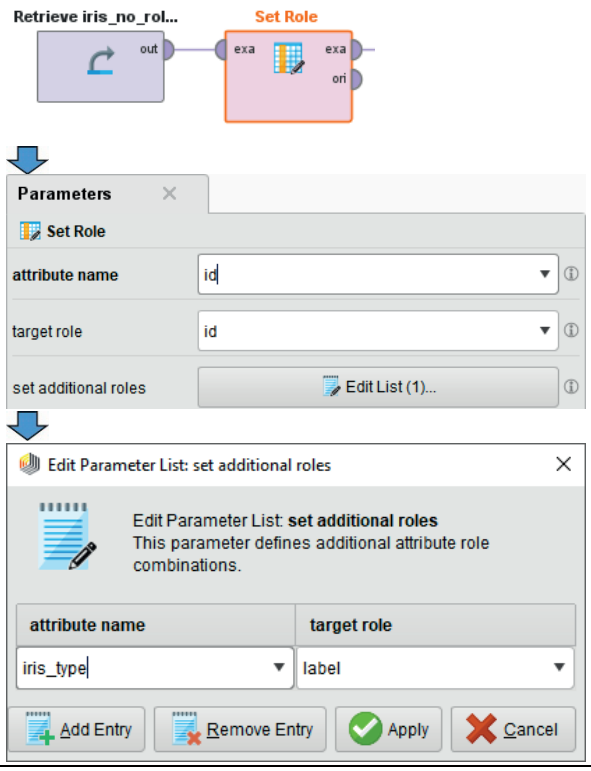
- **batch** = variabilă specială care definește loturi / sub-seturi de cazuri; este utilă în cadrul unei analize de validare încrucișată (cross-validation), în situația în care analistul dorește să definească el, nu softul, numărul grupurilor și apartenența cazurilor la aceste grupuri.

Figura 5.1-2. Setarea rolului unui atribut (Set Role)

Pasul 1:
Încărcăm setul de date RapidMiner „iris_no_roles". Conectăm operatorul „Set Role".

Pasul 2:
Setăm rolul de id pentru atributul numit id.

Pasul 3:
Setăm alte roluri – în acest caz setăm rolul de label pentru atributul numit „iris_type".
Putem adăuga alte variabile cu „Add Entry".



Tipuri de atribute (Types)

Prin tipuri de atribute înțelegem niveluri de evaluare / măsurare. Literatura de specialitate distinge între două mari tipuri metric și non-metric (variabila / atributul are sau nu asociată o unitate de măsură). Variabilele metrice pot fi de tip interval (valoarea zero absolut este stabilită convențional) și raport (zero absolut este natural, corespunde absenței depline a însușirii). Variabilele non-metrice pot fi de tip nominal (orice clasificare) și ordinal (o variabilă nominală cu clasele ordonate în funcție de intensitatea însușirii). Denumirile folosite în cadrul RapidMiner Studio pentru a identifica tipurile de atribute se suprapun parțial cu clasificarea prezentată anterior. În RapidMiner Studio, atributele de tip non-metric (fără unitate de măsură) sunt numite nominale, iar cele metrice sunt numite numerice. Valorile atributelor nominale iau forma unui text, iar cele numerice iau forma unor numere.[37] Pentru fiecare dintre aceste tipuri, RapidMiner distinge diferite sub-tipuri. Astfel, variabilele de tip **nominal** pot fi de tip:

- **text**: orice text (răspunsuri elaborate de către subiecții cercetării; de obicei astfel de fragmente de discurs sunt unice fiecărui caz / subiect),
- **binominal**: variabile nominale cu două categorii / clase (sex: masculin și feminin),
- **polynominal**: variabile nominale cu mai mult de două categorii / clase (statut marital: căsătorit / parteneriat, necăsătorit, divorțat, văduv),
- iar cele **numerice** pot fi de tip
- **integer**: numere întregi (numărul de angajați),
- **real**: numere reale (zecimale) (suprafața biroului în m^2, măsurată cu o precizie de o zecimală),
- **date_time**, **date**, **time**: data, timpul, respectiv data și timpul.

[37] Pentru a fi cu adevărat variabile metrice, numerele respective trebuie să indice intensitatea prezenței însușirii exprimată în unitățile de măsură specifice acelei variabile. O variabilă care are ca variante de răspuns codurile 1 și 2, unde 1 înseamnă masculin și 2 feminin, nu este o variabilă metrică pentru că: nu avem o unitate de măsură, deci nu are sens să facem operații matematice cu valorile acestei variabile (dacă adunăm doi bărbați nu obținem o femeie); clasele nu sunt nici măcar ordonate (am fi putut să atribuim codul 1 clasei feminin și codul 2 clasei masculin). O astfel de variabilă este una de tip nominal, binominal mai exact.

Operatorii grupați în categoria Types pot fi utilizați pentru a defini tipul de date asociat unei variabile, mai exact pentru a transforma dintr-un tip de date în altul. Atunci când importăm un set de date în RapidMiner Studio, softul recunoaște (prezice) tipul de date asociat fiecărei variabile. Uneori, softul ghicește greșit tipul unei variabile (această eroare se întâmplă de fiecare dată în cazul în care o variabilă nominală are asociate variantelor de răspuns coduri numerice, nu etichete / text) deci trebuie să-l corectăm. Alteori, poate fi necesar să recodăm o variabilă și în situația în care aceasta este nominală dar dorim să o includem într-o analiză care necesită doar variabile codate numeric. De exemplu, dacă avem o variabilă binominală cu variantele de răspuns nu/da și dorim să o prezicem folosind clasificatorul neural networks, va trebui să o recodăm 0/1 deoarece acest clasificator poate prezice doar variabile (codate ca) numerice.

Funcție de situația concretă în care ne aflăm, RapidMiner oferă operatori care transformă o variabilă de tip numeric în una binominală / polinominală / reală / dată, o variabilă de tip nominal în una binominală / numerică / dată, reală în numere întregi, text în nominală, respectiv dată în numerică / nominală. Nu are sens să ilustrăm aici toți acești operatori deoarece opțiunile sunt similare. Vom exemplifica utilizarea acestor operatori în două tipuri de situații și anume transformarea unei variabile numerice în una binominală, respectiv a uneia nominale în binominală.

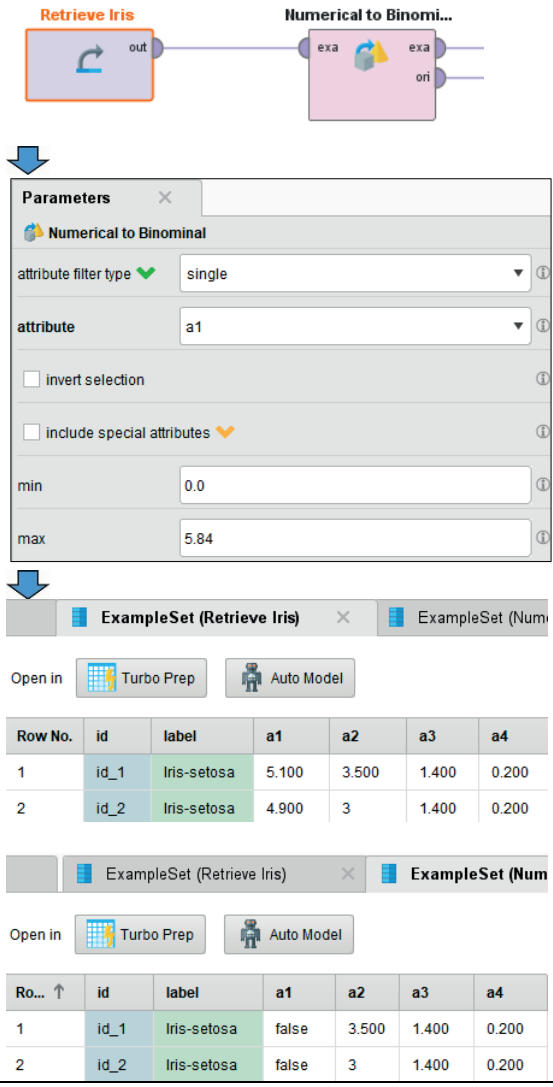
Transformarea unei variabile numerice într-o variabilă binominală se realizează cu ajutorul operatorului „Numerical to Binominal”. La parametrul „attribute filter type” indicăm dacă dorim să transformăm un singur atribut, câteva sau toate (avem și alte opțiuni, utilizate relativ mai rar). La parametrul „attribute” alegem atributul / atributele, iar la min și max indicăm intervalul valorilor care vor fi transformate în categoria false (restul valorilor vor forma categoria true). Dacă dorim să selectăm celelalte atribute decât cele indicate la parametrul „attribute”, bifăm opțiunea „invert selection”. Dacă dorim să includem și atributele speciale din setul de date, bifăm opțiunea „include special attributes”.

Figura 5.1-3. Transformarea unei variabile numerice într-o variabilă binominală (Numerical to Binominal)

Pasul 1:
Încărcăm setul de date RapidMiner „Iris".
Conectăm operatorul „Numerical to Binominal".

Pasul 2:
Alegem single la parametrul „attribute filter type".
Alegem atributul a1 la parametrul „attribute".
La min și max indicăm intervalul de valori care vor fi transformate în categoria false (restul vor fi true). 5.84 este media variabilei a1 (putem alege orice altă valoare).

Pasul 3:
Rulăm procesul și comparăm cele două seturi de date (original și modificat).
Observăm că valorile din intervalul [0; 5.84] ale atributului a1 au acum valoarea false, iar restul (peste 5.84) valoarea true.

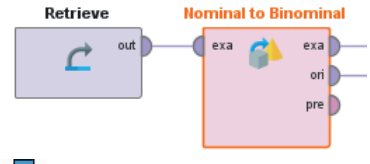


Transformarea unei variabile nominale într-o variabilă binominală se realizează cu ajutorul operatorului „Nominal to Binominal". Parametrii acestui operator sunt similari cu cei ai operatorului prezentat anterior. Rolul acestui operator este de a construi câte o nouă variabilă pentru fiecare dintre variantele de răspuns ale variabilei(lor) selectate. Acest tip de transformare poate fi util în situațiile în care (1) dorim să folosim mai puține categorii în

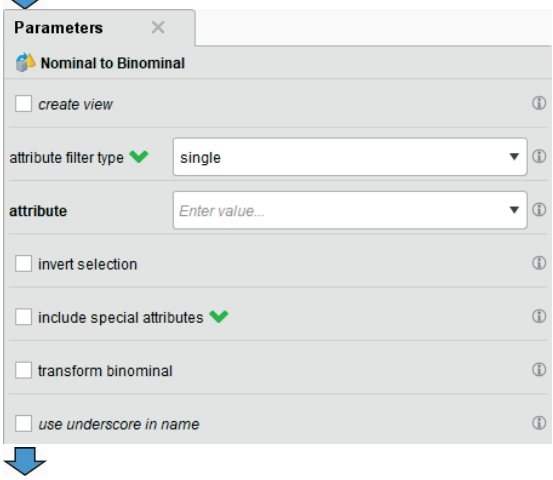
modelul de predicție cu scopul de a reduce complexitatea modelului (în relație și cu numărul de cazuri disponibile), (2) analiza / clasificatorul permite doar utilizarea unor variabile de tip numeric (putem transforma variabilele binominale în variabile numerice), respectiv (3) softul a citit greșit nivelul de măsurare al unei variabile atunci când am importat setul de date.

Figura 5.1-4. Transformarea unei variabile nominale într-o variabilă binominală (Nominal to Binominal)

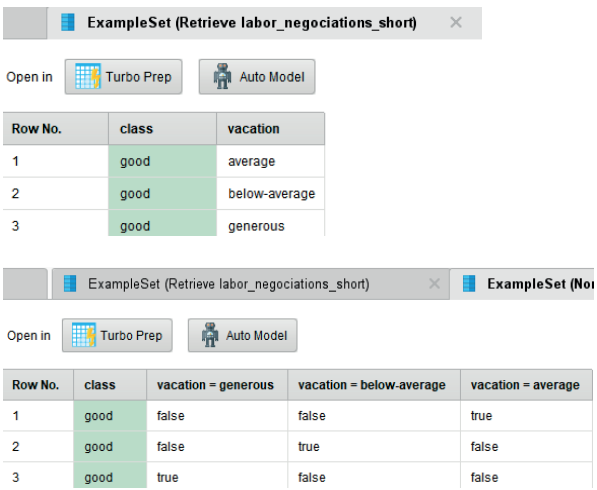
Pasul 1:
Încărcăm setul de date RapidMiner „labor_negociations_short". Conectăm operatorul „Nominal to Binominal".



Pasul 2:
Alegem single la parametrul „attribute filter type".



Pasul 3:
Rulăm procesul și comparăm cele două seturi de date (original și modificat).
Observăm că în locul atributului vacation cu trei categorii de răspuns (average, below-average și generous), noua bază are trei atribute, câte unul pentru fiecare dintre categoriile originale. Noile variante de răspuns sunt true / false.



Alături de operatorii prezentați anterior, categoria Types include o serie de alți operatori precum:

- **parse numbers**: similar cu operatorul „Nominal to Numerical" cu mențiunea că în acest caz, deși valorile originale sunt în realitate de tip numeric, ele sunt definite (stocate) ca nominale în setul de date;
- **format numbers**: formatarea numerelor (de exemplu afișarea valorilor ca procent sau adăugarea unui simbol care specifică moneda);
- **guess types**: ghicirea tipului de variabile;
- **one-hot encoding**: elimină variabilele nominale care au prea multe categorii (putem specifica pragul), respectiv codifică variabilele rămase ca numerice;
- **target encoding**: similar cu operatorul anterior;
- **set positive values**: indicarea variantei de răspuns care este privită ca pozitivă (de interes / de dorit) din punctul de vedere al analizei.

Selectarea atributelor (Selection)

Operatorii incluși în această categorie au în comun faptul că, relativ la un set de date, rețin în analiză doar atributele care îndeplinesc anumite condiții. Selecția poate fi realizată în baza a diferite criterii, funcție de obiectivele urmărite. Astfel, putem:

- specifica direct atributele pe care le dorim (**Select Attributes**);
- selecta atributele care au o anumită importanță (**Select by Weights**), de obicei mai mare de un anumit prag, dar putem realiza selecția și în funcție de alte criterii (primele x atribute care au scorul importanței cel mai mare, primele x% care au scorul importanței cel mai mare etc.);
- alege aleator un anumit număr de atribute (**Select by Random**);
- elimina atributele dintr-un anumit interval, de exemplu toate atributele din intervalul 3-5, adică atributele 3, 4 și 5 (**Remove Attribute Range**);
- elimina atributele inutile (**Remove Useless Attributes**); ne ajută să eliminăm atributele numerice care nu depășesc un anumit prag de variația (valorile sunt relativ constante), atributele nominale care sunt mai degrabă constante (o mare parte a cazurilor au aceeași valoare; putem defini proporția cazurilor), care au foarte multe valori diferite

(foarte multe valori sunt luate de o parte mică a cazurilor; putem defini proporția) sau care arată ca o variabilă de tip id (valorile sunt toate diferite între ele),

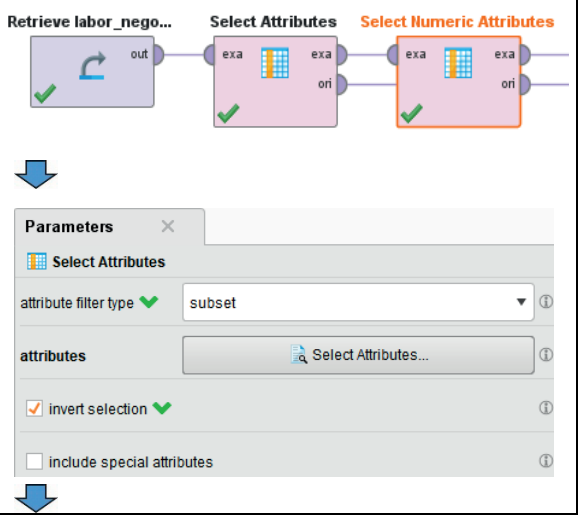
- elimina atributele care corelează mai mult decât un anumit prag (pragul poate fi indicat de utilizator, respectiv definit ca valoare absolută sau nu) (**Remove Correlated Attributes**); dacă două atribute corelează foarte puternic, informația utilă pentru predicție adusă de ele se suprapune în cea mai mare parte, deci utilitatea marginală a unuia dintre atribute este foarte mică;
- aplica unul sau mai mulți operatori (un sub-proces = set de comenzi) doar unora dintre atributele care apar în setul de date (**Work on Subset**).

Operatorul utilizat cel mai frecvent pentru a selecta atributele pe care dorim să le utilizăm este „Select Attributes". În Figura 5.1-5 ilustrăm folosirea acestui operator în două dintre situațiile posibile (selectarea anumitor atribute și selectarea atributelor de tip numeric).

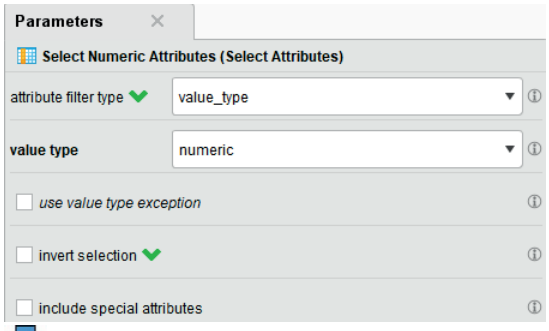
Figura 5.1-5. Selecția atributelor (Select Attributes)

Pasul 1:
Încărcăm setul de date „labor_negociations".
Conectăm operatorul „Select Attributes" (de două ori; al 2-lea îl numim „Select Numeric Attributes").

Pasul 2:
Alegem subset la parametrul „attribute filter type", iar la attributes alegem variabilele contribution-to-dental-plan și contribution-to-health-plan.
Bifăm „invert selection" pentru a indica faptul că dorim să reținem restul variabilelor.



Pasul 3:
Alegem value_type la parametrul „attribute filter type".
La parametrul „value type" alegem numeric.
Scopul este de a selecta toate variabilele de tip numeric.



Pasul 4:
Rulăm procesul și comparăm cele două seturi de date.
Observăm că în setul nou de date nu mai apar cele două variabile excluse la pasul 2 și nici variabilele care în setul original nu erau numerice (cu excepția variabilei numite class care este o variabilă specială de tip label; dacă dorim să aplicăm comanda respectivă și acestei variabile, trebuie să bifăm parametrul „include special attributes").

Name	Type	Name	Type
class (Label)	Nominal	class (Label)	Nominal
duration	Integer	duration	Integer
wage-inc-1st	Real	wage-inc-1st	Real
wage-inc-2nd	Real	wage-inc-2nd	Real
wage-inc-3rd	Real	wage-inc-3rd	Real
col-adj	Nominal	working-hours	Integer
working-hours	Integer	standby-pay	Integer
pension	Nominal	shift-differential	Integer
standby-pay	Integer	statutory-holidays	Integer

Generarea atributelor (Generation)

Operatorii incluși în categoria Generation pot fi utilizați pentru a construi atribute pornind de la atributele deja existente în setul de date (doar operatorul „generate ID" generează un atribut de tip ID = cod unic de identificare pentru fiecare caz din setul de date). Posibilitățile de a genera atribute sunt numeroase, după cum se poate vedea din scurta descriere a operatorilor disponibili:

- **Generate Attributes**: generează atribute folosind expresii (matematice și logice) definite de utilizator;
- **Generate Empty Attribute**: generează atribute fără valori (putem denumi atributul și specifica nivelul de măsurare);
- **Generate Copy**: generează un atribut identic cu un alt atribut existent în setul de date;
- **Generate Concatenation**: unește două atribute într-un nou atribut de tip nominal cu valori egale cu valorile unite ale celor două atribute (putem adăuga sau nu diferite caractere separatoare între valori); de exemplu, putem uni două atribute care conțin numele, respectiv prenumele într-un nou atribut care le conține pe ambele;
- **Generate Aggregation**: generează un atribut nou care ia valorile indicate de funcția de agregare specificată; de exemplu, putem genera un atribut care ia valoarea medie a unei variabile existente în setul de date sau media acelei variabile în cadrul claselor altei variabile;
- **Generate Absolutes**: înlocuiește toate valorile numerice cu modulul lor (valorile negative devin pozitive);
- **Generate Products**: generează câte un atribut nou pentru fiecare produs de două atribute indicate în două liste de atribute;
- **Generate Gaussians**: generează o variabilă care ia valori definite de atributul, media și abaterea standard specificate;
- **Generate Function Set**: generează o serie de atribute pentru fiecare dintre atributele din setul de date în urma aplicării uneia sau mai multora dintre funcțiile predefinite selectate;
- **Generate TFIDF**: calculează importanța cuvintelor relativ la fiecare dintre documentele analizate;
- **Generate Item Set Indicators**: creează atribute numerice de tip binar (0/1) unde 1 indică faptul că textul respectiv include cuvintele specificate de utilizator;
- **Generate Weight (Stratification)**: calculează o variabilă de tip ponderare (weight) egală cu numărul de cazuri indicat de utilizator astfel încât numărul de cazuri să fie același între categoriile / clasele variabilei label;

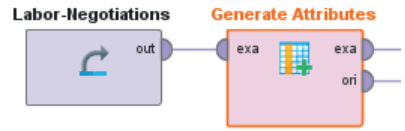
- **Generate Weight (LPR)**: generează o variabilă de tip weight folosind Local Polynomial Regression;
- **Generate Batch**: produce un atribut special de tip batch (divide setul de date în numărul specificat de grupuri);
- **Text Vectorization**: util pentru extragerea informațiilor din variabile de tip text (de exemplu extrage informații cu privire la prezența anumitor cuvinte, sentimentul general al textului (sentiment analysis), detectează limba folosită).

Foarte probabil, cel mai frecvent utilizat operator din această categorie este „Generate Attributes", motiv pentru care îl ilustrăm în continuare (Figura 5.1-6).

Figura 5.1-6. Generarea atributelor (Generate Attributes)

Pasul 1:
Încărcăm setul de date „labor_negociations".
Conectăm operatorii „Set Macros" și „Generate Attributes".

Pasul 2:
Folosind operatorul „Generate Attributes" definim numele noilor atribute și funcțiile asociate lor. Imaginea prezintă doar o parte dintre atributele generate (pentru alte exemple se pot vedea procesul și Tabelul 5.1-1).



attribute name	function expressions
average wage-inc	([wage-inc-1st] + [wage-inc-2nd] + [wage-inc-3rd]) / 3
neglected worker bool	([working-hours] >= 35) && ([education-allowance] != "yes")
logarithmic attribute	log([shift-differential]) + ln([standby-pay])
trigno attribute	sin([wage-inc-1st]) + cos([wage-inc-1st]) + tan([wage-inc-1s
rounded average wage-inc	round(avg([wage-inc-1st],[wage-inc-2nd],[wage-inc-3rd]))
vacations	replaceAll(vacation,"generous","above-average")
deadline	if(class=="good", date_add(date_now(),25,DATE_UNIT_DA
shift complete	if(missing([shift-differential])==true, floor(rand()*25), [shift-di

Operatorul „Generate Attributes" generează atribute folosind expresii matematice și logice definite de utilizator (expresiile pot include una sau mai multe funcții). În cele ce urmează (vezi și fișierul cu exemplul de proces, respectiv Tabelul 5.1-1), ilustrăm și explicăm câteva astfel de expresii. Desigur, nu este posibil să acoperim nici măcar toate tipurile de situații posibile. Prima coloană reprezintă numele atributului pe care dorim să-l generăm, iar a doua expresia asociată.

Tabelul 5.1-1. Exemple de funcții utilizate pentru generarea unor atribute

attribute name	function expressions
average wage-inc	[wage-inc-1st] + [wage-inc-2nd] + [wage-inc-3rd]) / 3
neglected worker bool	([working-hours] >= 35) && ([education-allowance] != "yes") && (vacation == "average" || vacation == "below-average")
logarithmic attribute	log([shift-differential]) + ln([standby-pay])
trigno attribute	sin([wage-inc-1st]) + cos([wage-inc-1st]) + tan([wage-inc-1st])
rounded average wage-inc	round(avg([wage-inc-1st],[wage-inc-2nd],[wage-inc-3rd]))
vacations	replaceAll(vacation,"generous","above-average")
deadline	if(class=="good", date_add(date_now(),25,DATE_UNIT_DAY), date_add(date_now(),10,DATE_UNIT_DAY))
shift complete	if(missing([shift-differential])==true, floor(rand()*25), [shift-differential])
remaining_holidays	15 - [statutory-holidays]
remaining_holidays_perc	remaining_holidays / 15 * 100
constants	PI * e
cut	cut(class,0,2) + " "+ cut("class",0,2)
index	index(class,"o")
date constants	date_str(deadline,DATE_FULL,DATE_SHOW_TIME_ONLY)

** Sursa: Help RapidMiner*

- Primul exemplu (Tabelul 5.1-1) ilustrează generarea unei variabile numite „average wage-inc" ca medie aritmetică a variabilelor wage-inc-1st, wage-inc-2nd și wage-inc-3rd. Fiecare dintre numele acestor atribute este pus între paranteze drepte deoarece numele respective conțin caractere „ilegale". Un caracter „ilegal" este un caracter care are asociată o funcție, deci nu ar trebui să fie folosit în denumirea unui atribut. În acest caz, caracterul „-" are asociată operația de scădere, deci softul ar înțelege greșit că atributul „wage-inc-1st" este o diferență între atributul „wage" și atributele „inc" și „1st". În absența parantezelor drepte care să delimiteze fiecare atribut și deoarece setul de date nu conține un atribut cu numele „wage" (sau „inc" sau „1st"), softul va semnala faptul că expresia respectivă nu este corectă și nu o va executa (procesul nu va rula). Desigur, dacă numele unui atribut nu conține caractere „ilegale", nu e nevoie să-l punem între paranteze drepte.

- Exemplul secund ilustrează generarea unei variabile binominale (numită „neglected worker bool") care ia valorile fals și true, funcție de statutul de adevăr al expresiei logice indicate.
- Atributul „rounded average wage-inc" este similar cu atributul din primul exemplu doar că în acest caz valorile sunt rotunjite, media fiind calculată folosind funcția avg (average) din RapidMiner.
- Următoarele două atribute generate („logarithmic attribute" și „trigno attribute") exemplifică generarea cu ajutorul funcțiilor logaritmice și trigonometrice.
- Atributul nou „vacations" ia valorile atributului vechi „vacation", dar în locul variantei de răspuns „generous" pune „above-average".
- Atributul nou „deadline" indică, raportat la momentul rulării procesului, a 25-a zi pentru cazurile care la variabila class au valoarea „good", respectiv a 10-a pentru cazurile „bad".
- Atributul „shift complete" copiază valorile atributului „shift-differential" dar înlocuiește valorile lipsă (missing values) cu valori aleatoare mai mici de 25.
- Atributul nou „remaining_holidays" este egal cu valoarea 15 (numărul total de zile de concediu) minus valoarea atributului „statutory-holidays" (numărul de zile de concediu avute). Atributul „remaining_holidays_perc" exprimă valorile calculate la atributul „remaining_holidays" ca procent din total zile de concediu. Se poate observa că în acest caz atributul „remaining_holidays" nu este pus între paranteze drepte deoarece caracterul underscore (_) poate fi folosit în denumirea unui atribut.
- Atributul „constants" este egal cu o constantă (produsul dintre valorile constantelor PI și e).
- Atributul denumit „cut" unește (concatenează) trei fragmente de text și anume: literele ba sau go (primele două litere de la răspunsul observat la atributul class), un spațiu și literele cl (primele două litere ale cuvântului class).
- Atributul numit „index" ia valorile 1 dacă răspunsul la atributul class conține litera o și -1 dacă nu conține această literă.
- Atributul denumit „date constants" copiază valorile atributului „deadline" (indică ziua și ora până la care trebuie finalizată sarcina; e o

variabilă de tip „Date time") dar afișează doar timpul ca text (variabilă de tip nominal).

5.2. Lucrul cu cazuri (Examples)

În RapidMiner Studio, conceptul de exemplu se referă la cazul statistic, liniile dintr-un tabel cu date, unitatea relativ la care au fost colectate datele (atributele). Operatorii grupați în secțiunea „Examples" au în comun tocmai faptul că realizează acțiuni în relație cu cazurile din setul de date. Distingem între trei categorii mari de acțiuni și anume filtrarea / selecția (**Filter**), eșantionarea (**Sampling**) și sortarea cazurilor (**Sort**).

Filtrarea cazurilor (Filter)

Prin filtrare, RapidMiner înțelege selecție, deci operatorii Filter sunt folosiți pentru a reține dintr-un set de date doar cazurile care îndeplinesc condițiile specificate de analist. Filtrarea cazurilor se poate realiza folosind doi operatori: „Filter Examples" și „Filter Example by Range". Ultimul operator este utilizat pentru a selecta cazurile în funcție de poziția lor în setul de date, utilizatorul indicând doar intervalul de cazuri pe care îl dorește. Primul operator este mai complex și mai util, deci merită o atenție sporită.

Operatorul „Filter Examples" are un parametru principal, „condition class", acesta indicând opțiunile pe care le avem la dispoziție atunci când dorim să definim condițiile pe care trebuie să le respecte cazurile care vor rămâne în setul de date:

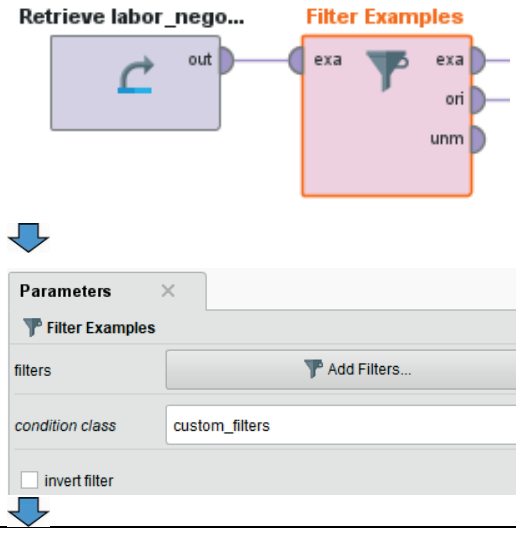
- **all**: păstrează toate cazurile;
- **correct_predictions**: păstrează cazurile cu predicții corecte (setul de date trebuie să conțină două variabile speciale, una de tip label și una prediction);
- **wrong_predictions**: păstrează cazurile cu predicții incorecte (setul de date trebuie să conțină două variabile speciale, una de tip label și una prediction);

- **no_missing_attributes**: păstrează cazurile care nu au valori lipsă la niciun atribut (indicate cu „?” în RapidMiner);
- **missing_attributes**: păstrează cazurile care au valori lipsă la toate atributele;
- **no_missing_labels**: păstrează cazurile care nu au valori lipsă la atributul special de tip label;
- **missing_labels**: păstrează cazurile care au valori lipsă la atributul special de tip label;
- **attribute_value_filter**: păstrează cazurile care respectă condiția indicată la parametrul string (acest parametru este afișat doar în cazul acestei opțiuni);
- **expression**: păstrează cazurile care respectă expresia indicată la parametrul „parameter expression” (acest parametru este afișat doar în cazul acestei opțiuni);
- **custom_filters**: păstrează doar cazurile care respectă condițiile definite de utilizator; condițiile sunt definite în relație cu unul sau mai multe atribute prin accesarea parametrul „filters” (apare doar în cazul acestei opțiuni); utilizarea acestei opțiuni este ilustrată în Figura 5.2-1.

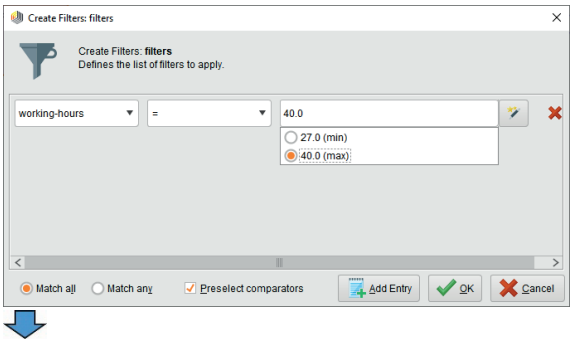
Figura 5.2-1. Selecția cazurilor (Filter Examples)

Pasul 1:
Încărcăm setul de date „labor_negociations”. Conectăm operatorul „Filter Examples”. Conectăm porturile de output exa (setul de date nou) și ori (setul de date original) la porturile cu rezultate (res).

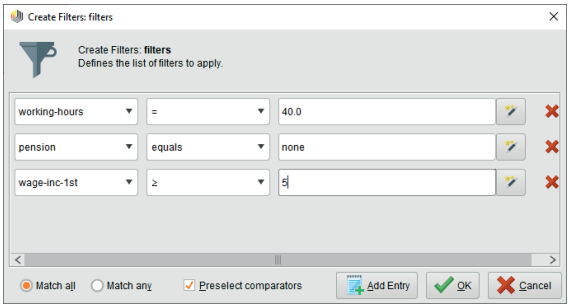
Pasul 2:
Alegem opțiunea custom_filters la parametrul „condition class”. Observăm că apare posibilitatea de a defini filtrele dorite.



Pasul 3:
Alegem atributul relativ la care dorim să definim condiția (working-hours), relația (=) și valoarea (40). Punând această condiție, în setul de date vor rămâne doar angajații care lucrează 40 ore pe săptămână.



Pasul 4:
Definim și alte condiții, funcție de obiectivele urmărite. În acest exemplu am adăugat două condiții: angajatul să nu aibă un plan de pensie și ultima creștere salarială să fie de cel puțin 5%. Alegem una dintre opțiunile „Match all" (cazurile selectate vor respecta toate condițiile specificate) sau „Match any" (cel puțin una dintre condiții).



Dacă alegem să definim noi condițiile („custom_filters"), putem face asta folosind diferite funcții. Astfel, cu ajutorul funcțiilor „is missing" / „is not missing", putem selecta cazurile care au valori la un anumit atribut (numeric sau nominal), respectiv nu au valori. Unele funcții (=, ≠, >, <, ≥, ≤) pot fi utilizate doar în cazul atributelor numerice, altele doar în cazul celor de tip nominal. Denumirile ultimelor sunt indicative relativ la rolul lor:

- **equals / does not equal**: valoarea nominală luată de un caz este aceeași cu valoarea indicată, respectiv nu este aceeași;
- **is in / is not in**: valoarea nominală apare în listă, respectiv nu apare; putem alege una sau mai multe valori nominale / categorii de răspuns (trebuie separate prin semnul „;");
- **contains / does not contain**: valoarea nominală conține, respectiv nu conține caracterele menționate;
- **starts with / ends with**: valoarea nominală începe, respectiv se termină cu caracterele menționate;
- **matches**: valoarea nominală este identică cu valoarea indicată.

Eșantionarea cazurilor (Sampling)

Pentru a extrage diferite eșantioane din setul de date folosim seria de operatori incluși în categoria „Sampling". Aceștia ne ajută să realizăm diferite tipuri de eșantioane, de la cele mai simple (simplu aleator) la unele mai complicate precum stratificat aleator sau bootstrap. Operatorii „Sample" și „Sample (Stratification)" au în comun faptul că selectează din setul de date original o parte dintre cazuri (cel mult toate cazurile), fiecare caz putând fi selectat o singură dată într-un eșantion (sampling without replacement). Operatorul „Sample (Bootstrapping)" poate selecta un caz de mai multe ori (sampling with replacement), deci poate ajunge la un eșantion oricât de mare dorim (adică inclusiv mai mare decât eșantionul original). În continuare, vom ilustra aceste trei modalități prin care putem extrage eșantioane.

Cea mai simplă selecție se poate realiza cu operatorul „Sample" (Figura 5.2-2, Figura 5.2-3). Parametrul „sample" asociat acestui operator permite alegerea tipului de valori numerice în funcție de care va fi realizat eșantionul. Putem alege între:

- valori absolute (**absolute**) – adică un anumit număr de cazuri (Figura 5.2-2),
- valori relative (**relative**) – adică o anumită pondere a cazurilor (Figura 5.2-3), și
- probabilități (**probability**) – selecția este realizată conform probabilităților specificate (Figura 5.2-4).

Pentru fiecare dintre aceste opțiuni va trebui să indicăm numărul de cazuri / proporția / probabilitatea relativ la numărul inițial de cazuri. O reducere a numărului de cazuri[38] poate fi utilă mai ales în situația în care numărul de cazuri din setul de date este foarte mare (milioane) și/sau e nevoie să pregătim setul de date și/sau dorim să testăm un model complex (multe atribute, multe clase pentru atributul label etc.). Rularea unor procese poate

[38] Uneori dorim, dimpotrivă, să creștem numărul de cazuri, în total sau doar în cazul clasei cu o incidență foarte scăzută (de exemplu mărim numărul de cazuri de fraudă dintr-un set de date) sau a unei combinații de atribute. Procedura bootstrapping este utilă tocmai pentru astfel de situații.

dura exponențial mai mult dacă setul de date are milioane de cazuri. Pentru a vedea dacă procesele realizate sunt corecte, ce rezultate produc, compara diferite modele de predicție, e mai simplu dacă folosim inițial un eșantion mai mic. Astfel, reducem foarte mult din timpul alocat pentru pregătirea și analiza datelor.

Dacă dorim să realizăm selecția diferit pentru cel puțin una dintre clasele atributului label, bifăm opțiunea „balance data" și apoi indicăm numărul de cazuri / ponderea / probabilitățile relativ la fiecare dintre clasele atributului label. Trebuie să fim atenți când scriem denumirile claselor. Acestea trebuie să fie identice cu cele din setul de date, altfel operatorul va produce un eșantion fără cazuri. Astfel, pentru cele două exemple de mai jos - denumirea corectă a claselor este Yes și No (denumirile încep cu majuscule) - dacă trecem yes și no (denumiri fără majuscule), procesul va rula, dar eșantionul rezultat nu va conține niciun caz.

Dacă bifăm opțiunea „use local random seed", respectiv trecem un număr în căsuță, ne asigurăm că procesul va produce exact aceleași rezultate la fiecare rulare.

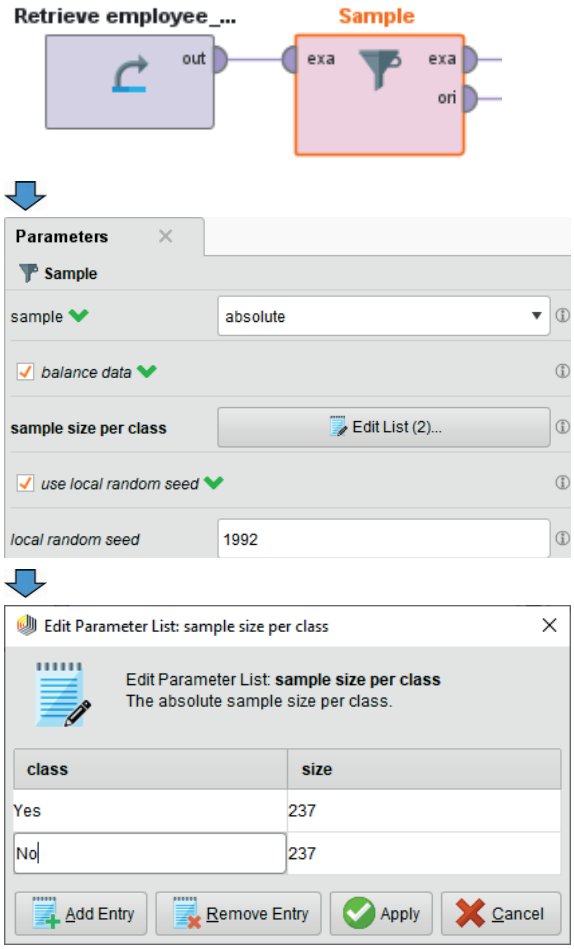
Exemplul din Figura 5.2-2 ilustrează selecția unui eșantion care va avea un număr precizat de cazuri (237) pentru fiecare dintre cele două clase ale variabilei de tip label numite Attrition. Observăm că setul original conține în total 1470 cazuri, din care 237 Yes (angajați care au părăsit compania) și 1233 No (angajați care lucrează în continuare în companie). Firesc, având în vedere tipul de comportament studiat, numărul de cazuri este destul de dezechilibrat (unbalanced) între cele două clase. Un astfel de set de date pune o serie de probleme atunci când dorim să realizăm un model de predicție. Pentru a echilibra setul de date (clasele variabilei label să fie relativ apropiate ca număr), putem apela la diferite strategii. Una dintre strategii constă în reducerea numărului de cazuri aferent categoriei dominante. În cazul de față, am selectat aleator 237 de cazuri No din cele 1233.

Figura 5.2-2. Extragerea unui eșantion (Sample) – valori absolute

Pasul 1:
Încărcăm setul de date „employee_attrition" și conectăm operatorul „Sample". Conectăm porturile de output exa și ori la porturile cu rezultate (res).

Pasul 2:
Alegem opțiunea „absolute" la parametrul „sample".
Bifăm opțiunea „balance data" (observăm că apare un nou parametru - „sample size per class").
Bifăm opțiunea „use local random seed", respectiv trecem în căsuță 1992 (sau alt număr).

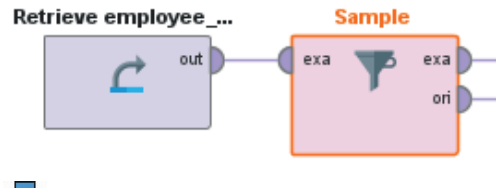
Pasul 3:
La parametrul „sample size per class" trecem numărul de cazuri pe care le vom selecta relativ la fiecare clasă. Aici, selectăm același număr de cazuri (237) pentru ambele clase (reținem toate cazurile Yes și selectăm aleator 237 dintre cele 1233 cazuri No).



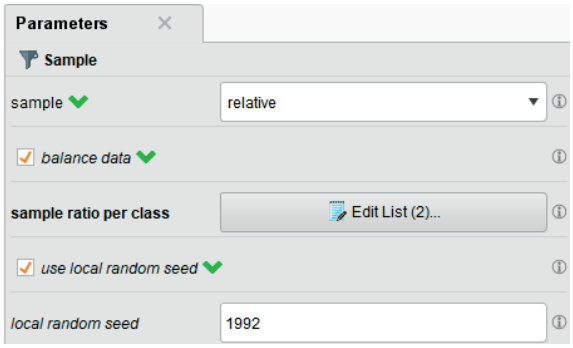
Selecția cazurilor se poate face și relativ, nu doar absolut. Simplu spus, putem selecta aleator o anumită proporție a cazurilor (nu doar un anumit număr de cazuri). În Figura 5.2-3 reluăm exemplul prezentat anterior folosind de această dată selecția relativă. Astfel, din totalul angajaților care au părăsit compania (răspuns No la variabila label) (1233), alegem să selectăm 20% și rămânem cu 246 cazuri No. Pentru că avem puțin cazuri Yes (237), alegem să le păstrăm pe toate. Setul de date rezultat va avea 483 cazuri în total (246 + 237).

Figura 5.2-3. Extragerea unui eșantion (Sample) – valori relative

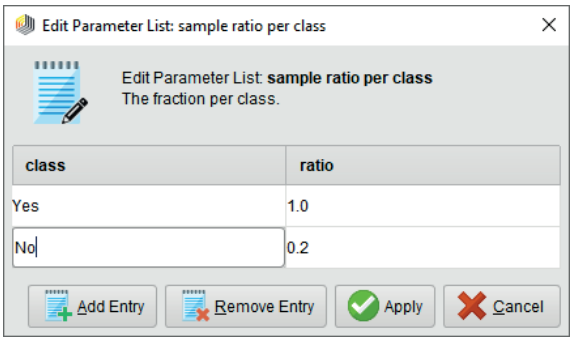
Pasul 1:
Încărcăm setul de date „employee_attrition" și conectăm operatorul „Sample". Conectăm porturile de output exa și ori la porturile cu rezultate (res).



Pasul 2:
Alegem opțiunea „relative" la parametrul „sample".
Bifăm opțiunea „balance data" (observăm că apare un nou parametru - „sample ratio per class").
Bifăm opțiunea „use local random seed", respectiv trecem în căsuță 1992 (sau alt număr).



Pasul 3:
La parametrul „sample ratio per class" definim ponderea cazurilor care urmează să fie selectate relativ la fiecare dintre cele două clase. Pentru că avem puține cazuri „Yes", le selectăm pe toate. Pentru clasa „No" selectăm doar o 20% dintre cazuri, astfel încât ponderile celor două clase să fie cât mai apropiate (~50%).



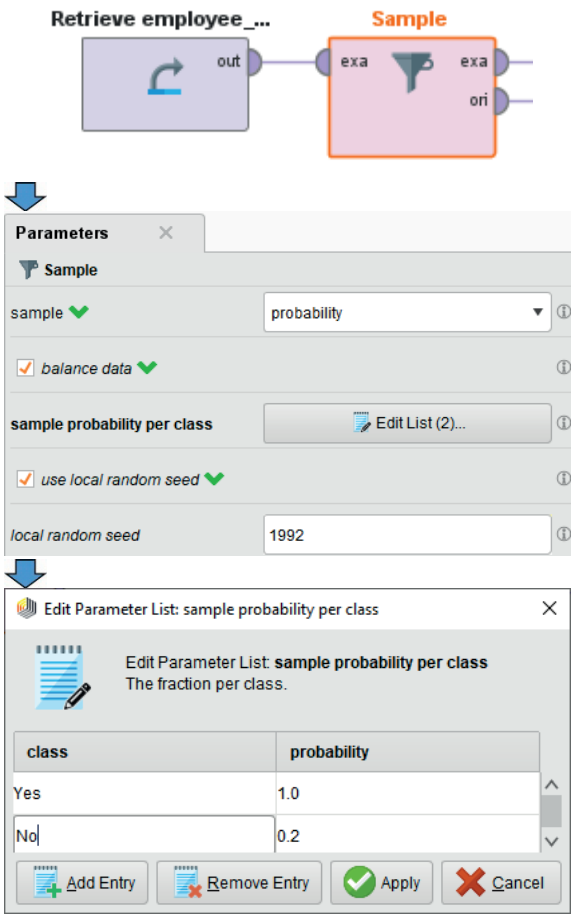
Selecția probabilistă este similară cu cea relativă. În Figura 5.2-4 reluăm exemplul prezentat anterior folosind de această dată selecția probabilistă. Cazurile No (1233 în setul de date original) au asociată probabilitatea de selecție 0.2, deci vor rămâne 231. Cazurile Yes au asociată probabilitatea de selecție 1, deci vor rămâne toate (237). Setul de date final va avea 468 cazuri în total (231 + 237).

Figura 5.2-4. Extragerea unui eșantion (Sample) – valori probabiliste

Pasul 1:
Încărcăm setul de date „employee_attrition" și conectăm operatorul „Sample". Conectăm porturile de output exa și ori la porturile cu rezultate (res).

Pasul 2:
Alegem opțiunea „probability" la parametrul „sample".
Bifăm opțiunea „balance data" (observăm că apare un nou parametru - „sample probability per class").
Bifăm opțiunea „use local random seed", respectiv trecem în căsuța 1992 (sau alt număr).

Pasul 3:
La parametrul „sample probability per class" definim probabilitățile cu care vor fi selectate cazurile relativ la fiecare dintre cele două clase. Pentru că avem puține cazuri „Yes", le selectăm pe toate (probabilitatea e 1). Pentru clasa „No" selectăm cazuri cu probabilitatea 0.2, astfel încât ponderile celor două clase să fie cât mai apropiate (~50%).

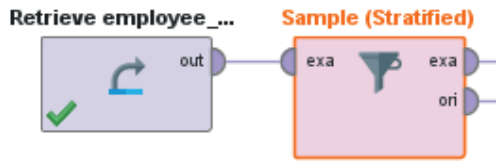


Operatorul „Sample (Stratified)" (Figura 5.2-5) extrage dintr-un set de date un eșantion de tip aleator stratificat. Prima data setul de date este divizat în sub-seturi de date, fiecare sub-set fiind compus din cazurile care formează o anumită clasă de răspunsuri relativ la variabila label. Astfel, variabila label din exemplele anterioare, Attrition, are două clase, Yes și No, deci vor rezulta două sub-seturi de date. Din fiecare sub-set sunt selectate simplu aleator un anumit număr de cazuri, cele două eșantioane rezultate fiind apoi unite rezultând astfel eșantionul final. Simplu spus, un eșantion aleator stratificat este unul care reproduce proporția claselor variabilei label. Acest operator

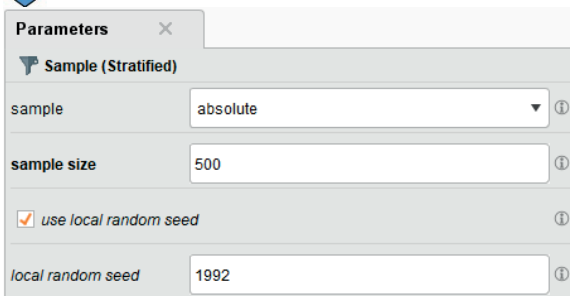
poate fi folosit doar dacă setul de date include un atribut label de tip nominal. Mărimea dorită a eșantionului poate fi specificată în termeni absoluți (număr de cazuri) sau relativi (proporții din cazuri).

Figura 5.2-5. Extragerea unui eșantion de tip stratificat (Sample - Stratified)

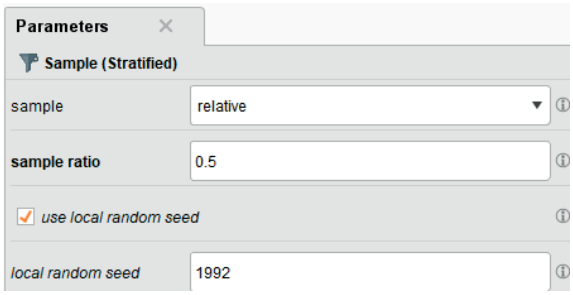
Pasul 1:
Încărcăm setul de date „employee_attrition" și conectăm operatorul „Sample (Stratified)". Conectăm porturile de output exa și ori la porturile cu rezultate (res).



Pasul 2 (absolute):
Alegem opțiunea „absolute" la parametrul „sample", iar la „sample size" trecem numărul de cazuri dorit (500).
Eșantionul rezultat va avea 500 de cazuri, iar proporțiile claselor Yes și No vor fi similare cu proporțiile din setul original.



Pasul 2 (relative):
Alegem opțiunea „relative" la parametrul „sample". La „sample ratio" trecem proporția dorită a cazurilor care dorim să rămână (0.5).
Eșantionul rezultat va avea jumătate din numărul de cazuri din setul original, iar proporțiile claselor Yes și No vor fi similare cu proporțiile din setul original.

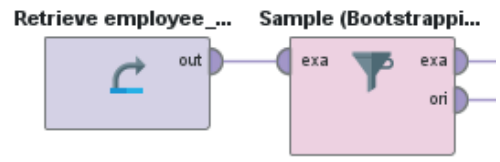


Operatorul „Sample (Bootstrapping)" poate produce un eșantion cu un număr de cazuri mai mic sau mai mare decât numărul de cazuri din setul de date original, funcție de intențiile analistului. În ambele situații, un caz poate intra o dată, de mai multe ori sau niciodată în eșantionul rezultat. Tehnic spus, acest tip de eșantionare este una cu înlocuire (sampling with replacement), adică la fiecare moment al selecției, fiecare caz are aceeași probabilitate de a fi selectat (nu contează dacă a mai fost selectat anterior sau

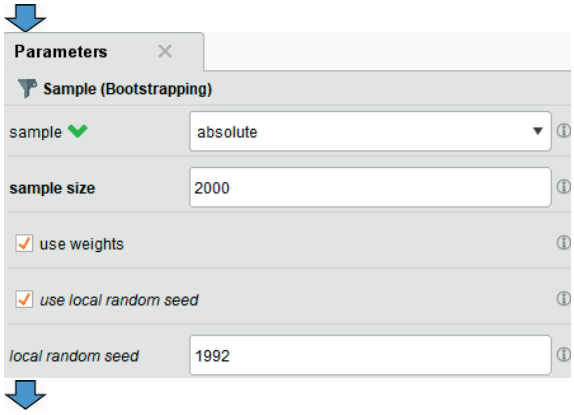
nu). Mărimea dorită a eșantionului poate fi definită la modul absolut sau relativ. Figura 5.2-6 prezintă ambele tipuri de situații. Dacă setul de date original include un atribut special de tip weight (ponderare), putem ține cont de această informație atunci când realizăm un eșantion folosind bootstrapping. Simplu spus, la fiecare pas al selecției, probabilitatea de selecție a unui caz va fi determinată și de valoarea de ponderare asociată acelui caz. Pentru a ține cont de variabila de ponderare, în cazul în care există una, bifăm parametrul „use weights".

Figura 5.2-6. Realizarea unui eșantion folosind procedura bootstrapping (Sample - Bootstrapping)

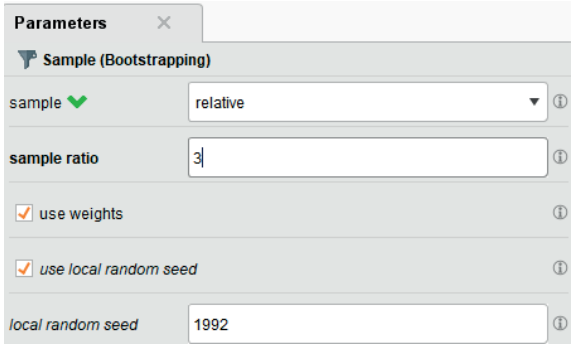
Pasul 1:
Încărcăm setul de date „employee_attrition" și conectăm operatorul „Sample (Bootstrapping)". Conectăm porturile de output exa și ori la porturile cu rezultate (res).



Pasul 2 (absolute):
Alegem opțiunea „absolute" la parametrul „sample", iar la „sample size" trecem numărul de cazuri dorit (2000).
Eșantionul rezultat va avea 2000 de cazuri, iar proporțiile claselor Yes și No vor fi similare cu proporțiile din setul original.



Pasul 2 (relative):
Alegem opțiunea „relative" la parametrul „sample". La „sample ratio" trecem valoarea cu care multiplicăm numărul de cazuri din setul original (3).
Eșantionul rezultat va avea de trei ori numărul de cazuri din setul original (1470 x 3 = 4410), iar proporțiile claselor Yes și No vor fi similare cu proporțiile din setul original.



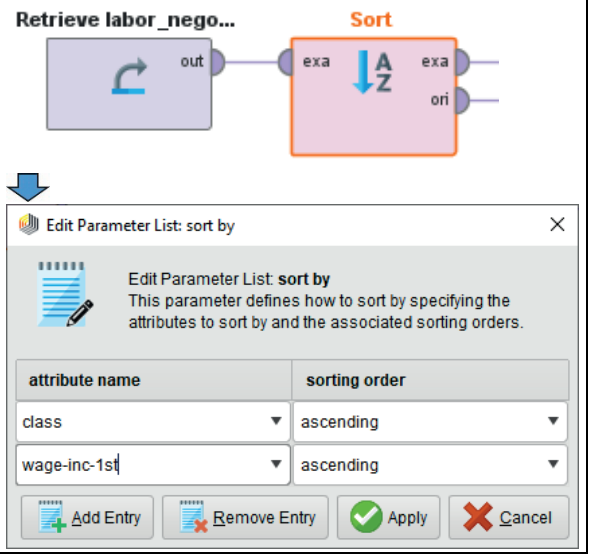
Sortarea cazurilor (Sort)

Grupul de operatori „Sort" conține operatorii: Sort, Shuffle și Sort by Pareto Rank. Sort poate fi utilizat pentru a sorta, după unul sau mai multe atribute, ascendent sau descendent, cazurile dintr-un set de date (Figura 5.2-7). Shuffle poate fi utilizat pentru a amesteca cazurile (a le ordona la întâmplare).

Figura 5.2-7. Sortarea cazurilor (Sort)

Pasul 1:
Încărcăm setul de date „labor_negociations" și conectăm operatorul „Sort". Conectăm porturile de output exa și ori la porturile cu rezultate (res).

Pasul 2:
Alegem variabilele în funcție de care dorim să facem sortarea și indicăm sensul acesteia.



5.3. Lucrul cu tabele (Tables)

Operatorii grupați în categoria Tables pot fi utilizați pentru trei tipuri mari de sarcini: gruparea / agregarea datelor dintr-un tabel (Grouping), rotația unui tabel (Rotation) și unirea a două sau mai multe tabele (Joins).

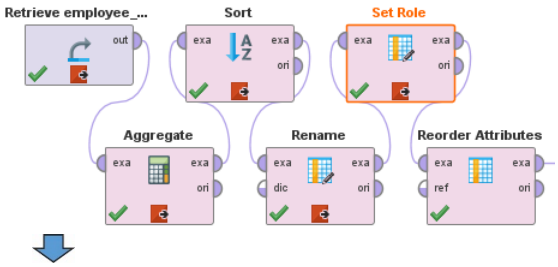
Agregarea datelor dintr-un tabel (Aggregate)

Pentru a grupa datele dintr-un tabel folosim operatorul Aggregate (Figura 5.3-1). Prima dată trebuie să specificăm atributul(ele) în funcție de care dorim să facem agregarea datelor. Un astfel de atribut trebuie să fie unul de tip nominal (binominal sau polinominal), adică să aibă categorii de răspuns. Apoi alegem ce atribute dorim să agregăm, respectiv indicăm ce măsuri

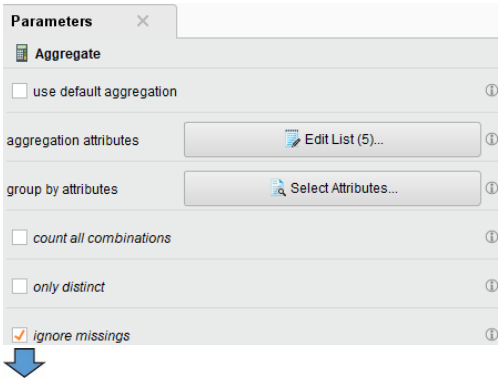
dorim să calculăm. Să presupunem că dorim să comparăm vârsta medie a celor două tipuri de angajați, cei care au plecat din companie, respectiv cei care au rămas (atributul Attrition). În acest caz, atributul în funcție de care facem agregarea este Attrition, atributul agregat este Age, iar funcția de agregare folosită este media (average). Operatorul va produce un tabel care include două valori, media vârstei celor care au plecat, respectiv a celor care au rămas.

Figura 5.3-1. Agregarea datelor dintr-un tabel (Aggregate)

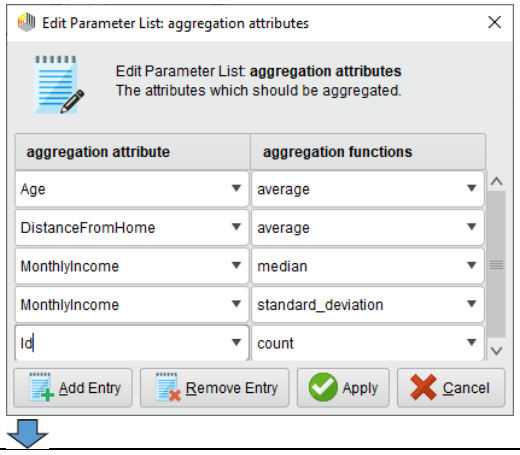
Pasul 1:
Încărcăm setul de date „employee_attrition" și conectăm operatorul Aggregate. Ceilalți operatori ne ajută să producem un tabel personalizat.



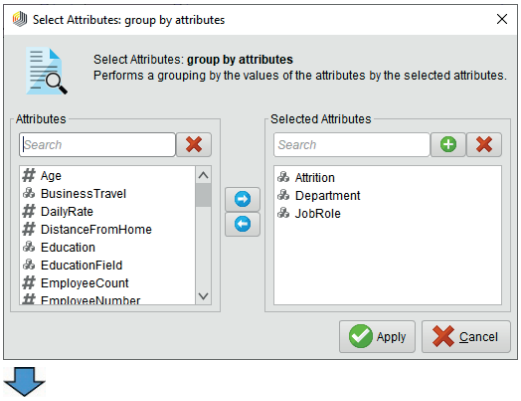
Pasul 2:
Operatorul Aggregate are doi parametri principali: „aggregation attributes" (atributele pe care dorim să le agregăm și funcțiile de agregare) și „group by attributes" (atributele în funcție de care dorim să facem agregarea).



Pasul 3:
La parametrul „aggregation attributes" indicăm funcția și atributul căruia dorim să-i aplicăm această funcție. Funcția aleasă trebuie să aibă sens în relație cu nivelul de măsurare al atributului.
În exemplul alăturat, cerem calcularea mediei pentru atributele Age și DistanceFromHome, mediana și abaterea standard pentru MontlyIncome și numărul de cazuri.



Pasul 4:
La parametrul „group by attributes" alegem atributele în funcție de care dorim să facem agregarea.
Toate aceste atribute trebuie să fie de tip nominal (au clase / categorii de răspuns). Putem alege unul sau mai multe atribute.



Rezultat:

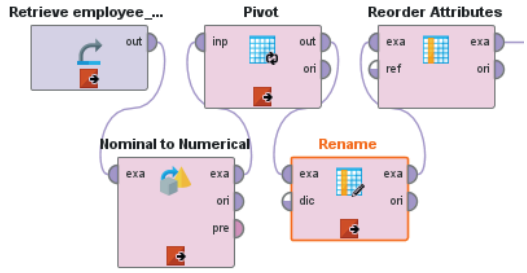
Department	JobRole	Attrition	Employees (#)	Age (mean)	DistanceFro...	MonthlyInco...	MonthlyInco...
Human Reso...	Human Reso...	No	40	37.125	6.600	3600	2241.002
Human Reso...	Human Reso...	Yes	12	30.083	13.417	2741	3063.978
Human Reso...	Manager	No	11	48.727	11.182	18844	1785.629
Research & ...	Healthcare R...	No	122	39.877	9.205	6755	2560.465
Research & ...	Healthcare R...	Yes	9	38.889	17.667	8722	2152.779
Research & ...	Laboratory Te...	No	197	34.944	9.330	3068	1172.986

Pivotarea unui tabel (Pivot)

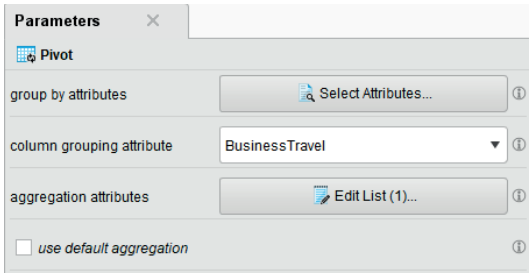
Pentru a pivota datele dintr-un tabel folosim operatorul Pivot (Figura 5.3-2). Prima dată specificăm atributele în funcție de care dorim să facem gruparea datelor. Pentru aceasta, indicăm la parametrul „group by attributes" unul sau mai multe atribute (acestea vor apărea pe linii în tabelul rezultat la final), iar la parametrul „column grouping attribute" alegem un singur atribut. Toate aceste atribute trebuie să fie de tip nominal. La parametrul „aggregation attributes" indicăm atributele pe care vrem să le agregăm împreună cu măsurile dorite. Să presupunem că dorim să aflăm dacă rata plecărilor (Attrition = Yes) variază în funcție de departament și intensitatea deplasărilor în interes de serviciu (atributul BusinessTravel). Atributele în funcție de care facem gruparea sunt exact acestea două, iar atributul agregat prin calcularea mediei este „Attrition = Yes". Aceste comenzi vor produce un tabel care va conține probabilitățile de plecare pentru fiecare dintre combinațiile claselor atributelor BusinessTravel și Departament.

Figura 5.3-2. Pivotarea unui tabel (Pivot)

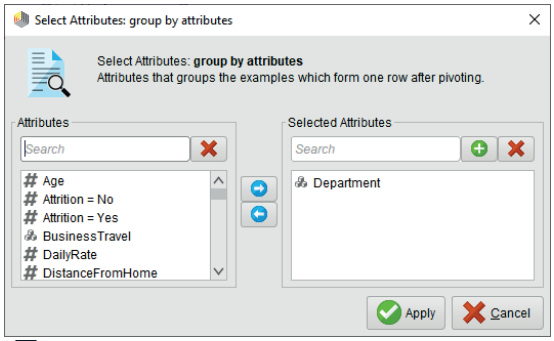
Pasul 1:
Încărcăm setul de date „employee_attrition” și conectăm operatorul Pivot. Ceilalți operatori ne ajută să producem un tabel personalizat.



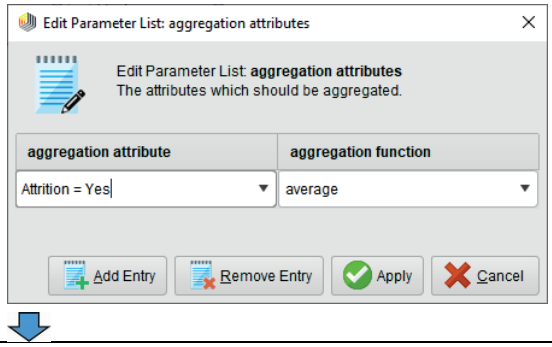
Pasul 2:
Operatorul Pivot are trei parametri principali: „group by attributes” (atributele în funcție de care dorim să facem gruparea) și „column grouping attribute” (atributul în funcție de care dorim să facem gruparea pe coloane) și „aggregation attributes” (atributele pe care dorim să le agregăm și funcțiile de agregare).



Pasul 3:
La parametrul „group by attributes” alegem atributele în funcție de care dorim să facem agregarea.
Toate aceste atribute trebuie să fie de tip nominal (au clase / categorii de răspuns). Putem alege unul sau mai multe atribute.



Pasul 4:
La parametrul „aggregation attributes” indicăm funcția și atributul căruia dorim să-i aplicăm această funcție. Funcția aleasă trebuie să aibă sens în relație cu nivelul de măsurare al atributului. În exemplul alăturat cerem calcularea mediei pentru atributul „Attrition = Yes”.



Rezultat:

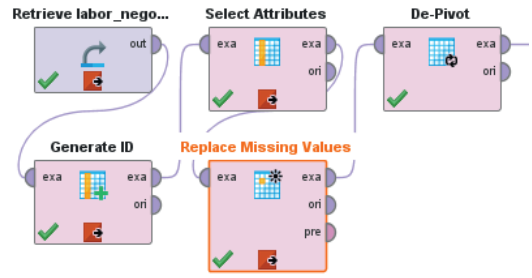
Department	Travel - None	Travel - Rarely	Travel - Frequently
Sales	0.085	0.190	0.333
Research & Development	0.082	0.129	0.203
Human Resources	0	0.174	0.364

De-pivotarea unui tabel (De-Pivot)

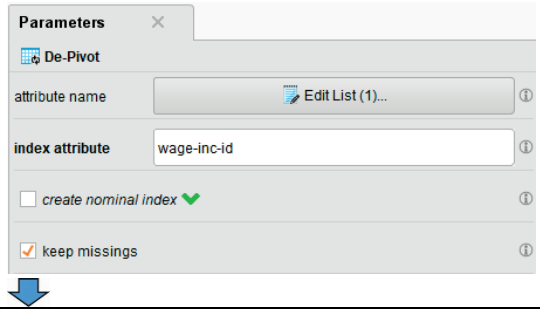
Pentru a ilustra utilizarea operatorului De-Pivot, să presupunem că avem un set de date care conține variabilele id, class/label, wage-inc-1st, wage-inc-2nd și wage-inc-3rd. Dorim să transformăm acest set de date într-un set care să conțină două variabile, wage-inc-id și wage-inc, unde wage-inc-id să indice numărul de ordine al creșterii salariale, iar wage-inc procentul cu care a crescut salariul la acel moment. Pașii descriși în Figura 5.3-3 realizează tocmai acest lucru.

Figura 5.3-3. Depivotarea unui tabel (De-Pivot)

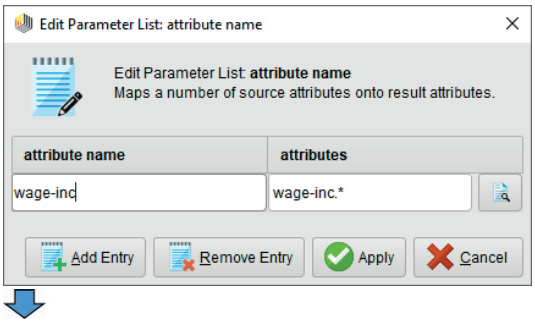
Pasul 1:
Încărcăm setul de date „labor_negociations" și conectăm operatorul De-Pivot. Ceilalți operatori ne ajută să producem un set de date potrivit pentru a ilustra rolul acestui operator.



Pasul 2:
La parametrul „index attribut" denumim atributul generat cu scopul de a identifica atributul sursă din care au fost preluate valorile (prima creștere salarială, a doua, a treia).



Pasul 3:
La parametrul „attribut name" alegem un nume pentru atributul generat și indicăm atributele sursă. Aici, „wage-inc.*" indică faptul că vom include toate atributele care au în nume „wage-inc".



Rezultat:
Pentru fiecare dintre cele trei creșteri salariale, noile atribute indică procentul creșterii (wage-inc) și numărul de ordine al creșterii salariale (1-3).
Observăm că fiecare caz din setul de date original apare acum de trei ori.

class	id	wage-inc-id	wage-inc
good	id_1	1	5
good	id_1	2	0
good	id_1	3	0
good	id_2	1	4.500
good	id_2	2	5.800
good	id_2	3	0

Transpunerea unui tabel (Transpose)

Așa cum indică și numele, operatorul Transpose transpune liniile și coloanele dintr-un set de date. Simplu spus, liniile / cazurile devin coloane / atribute, iar coloanele / atributele devin linii / cazuri. Pentru a ilustra funcționarea acestui operator vom folosi tabelul rezultat în urma analizei Pivot prezentate anterior (Figura 5.3-4).

Figura 5.3-4. Transpunerea unui tabel (Transpose)

Pasul 1:
Așa arată tabelul cu rezultate înainte de aplicarea operatorului Transpose.

Department	Travel - None	Travel - Rar...	Travel - Freq...
Sales	0.085	0.190	0.333
Research & ...	0.082	0.129	0.203
Human Reso...	0	0.174	0.364

Pasul 2:
Același tabel după aplicare operatorului Transpose.
Dat fiind faptul că tabelul inițial conținea o variabilă nominală, toate atributele din noul tabel sunt de tip nominal (chiar dacă ultimele trei conțin numere).

id	att_1	att_2	att_3
Department	Sales	Research & ...	Human Reso...
Travel - None	0.085106382...	0.082474226...	0.0
Travel - Rarely	0.190476190...	0.129032258...	0.173913043...
Travel - Frequ...	0.333333333...	0.203296703...	0.363636363...

Aspecte generale cu privire la unirea tabelelor (Joins)

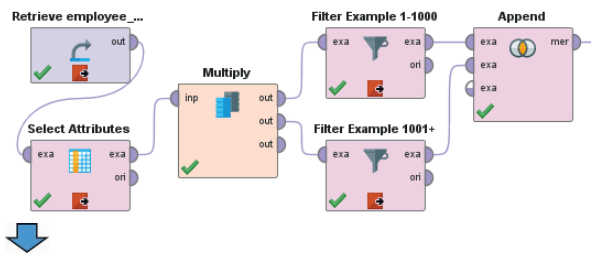
Funcție de situația concretă, unirea a două tabele (seturi de date) poate lua diferite forme. Astfel, putem uni două tabele care au aceleași atribute dar cazuri diferite (operatorii Append și Append (Robust)) sau cazuri și atribute (parțial) diferite (operatorii Join și alții). Append și Join sunt și comenzile utilizate cel mai des. Însă, în RapidMiner Studio sunt disponibile comenzi care unesc seturi de date și după alte reguli (Set Minus, Intersect, Union, Superset, Cartesian Product).

Unirea cazurilor din două tabele (Append)

Operatorul Append este folosit pentru a pune împreună cazurile din două sau mai multe seturi de date care au aceeași structură (aceleași atribute, definite identic). Exemplul prezentat în Figura 5.3-5 ilustrează utilizarea acestui operator.

Figura 5.3-5. Unirea cazurilor din două tabele (Append)

Pasul 1:
Încărcăm setul de date „employee_attrition" și conectăm operatorul Append. Ceilalți operatori ne ajută să producem două seturi de date cu aceleași variabile dar cazuri diferite.



Pasul 2:
Dorim să unim două tabele care au aceleași variabile (aceeași structură) dar cazuri diferite. Primul set are cazurile cu Id 1-1000, al doilea 1001-1470.

Id	Attrition	Age	Department
1	Yes	41	Sales
2	No	49	Research & ...
3	Yes	37	Research & ...
4	No	33	Research & ...

Id	Attrition	Age	Department
1001	No	52	Research & ...
1002	No	37	Research & ...
1003	No	35	Research & ...
1004	No	25	Research & ...

Rezultat:
Tabelul rezultat va include toate cele 1470 cazuri.

Id	Attrition	Age	Department
1	Yes	41	Sales
2	No	49	Research & ...
3	Yes	37	Research & ...
4	No	33	Research & ...
.....			
1001	No	52	Research & ...
1002	No	37	Research & ...
1003	No	35	Research & ...
1004	No	25	Research & ...

RapidMiner include un alt operator extrem de similar cu Append, și anume „Append (Robust)". Diferența dintre cei doi constă în faptul că ultimul reține toate valorile atributelor de tip nominal, chiar dacă acestea apar în doar una dintre bazele care au fost unite.

Unirea cazurilor și atributelor din două tabele (Join)

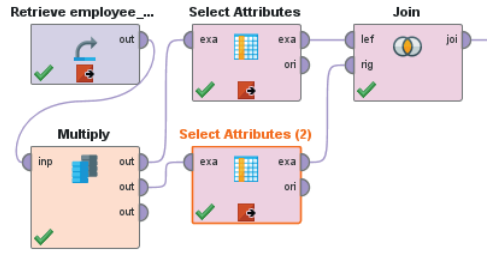
Operatorul Join unește cazurile și atributele din două seturi de date (putem uni mai multe seturi doar dacă folosim operatorul Join de mai multe ori). Pentru a performa această operație, ambele seturi de date trebuie să conțină cel puțin un atribut de tip id, definit identic. Acest atribut este folosit pentru a pune în corespondență cazurile din seturile de date pe care dorim să le unim. Dacă dorim, putem identifica cazurile după mai mult de un atribut. Join este foarte util atunci când dorim să selectăm din câteva seturi de date atributele pe care dorim să le includem într-un set nou de date.

Să presupunem că datele cu privire la salariile angajaților apar într-o bază, istoricul pozițiilor ocupate de fiecare angajat apar în altă bază, vânzările realizate de aceștia în alta, evaluările anuale în alta etc. Dacă dorim să aflăm care sunt factorii care prezic performanța angajaților va trebuie să preluăm diferite variabile din diferite baze sau seturi de date. În exemplul din Figura 5.3-6 am folosit operatorul Join pentru a pune în același tabel atributele Attrition (angajatul a părăsit sau nu compania), Age și Education (date socio-demografice). Intenția este de a vedea dacă rata plecărilor variază în funcție

de vârsta și nivelul de educație al angajaților. Corespondența între cazuri a fost realizată folosind atributul de tip id numit Id.

Figura 5.3-6. Unirea atributelor și cazurilor din două tabele (Join)

Pasul 1:
Încărcăm setul de date „employee_attrition" și conectăm operatorul Join (cu opțiunea inner). Ceilalți operatori ne ajută să producem două seturi de date cu aceleași variabile dar cazuri diferite.



Pasul 2:
Dorim să unim două tabele care conțin atribute diferite cu privire la aceleași cazuri.
Primul set include variabilele Id, Attrition și Department. Setul secund include atributele Id, Age și Education.
Observăm că variabila numită Id este comună și este definită ca variabilă specială de tip id.
Ambele seturi conțin aceleași 1470 cazuri.

Id	Attrition	Department
1	Yes	Sales
2	No	Research & ...
3	Yes	Research & ...
4	No	Research & ...

Id	Age	Education
1	41	College
2	49	Below College
3	37	College
4	33	Master

Rezultat:
Tabelul rezultat va include toate atributele din cele două tabele, pentru toate cele 1470 cazuri.

Id	Attrition	Department	Age	Education
1	Yes	Sales	41	College
2	No	Research & ...	49	Below College
3	Yes	Research & ...	37	College
4	No	Research & ...	33	Master

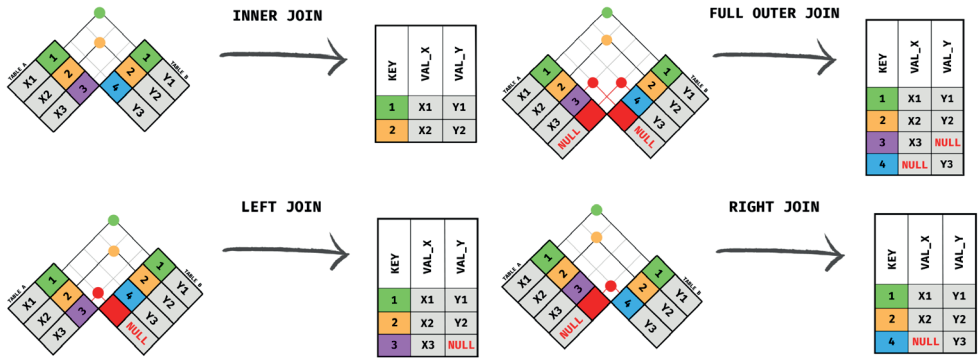
.....

Operatorul Join are un parametru (joint type) care specifică tipul de unire pe care dorim să o realizăm (procesele aferente nu sunt prezentate aici, dar sunt incluse în folderul cu fișiere aferent acestui volum). Putem alege între:

- **inner**: reține cazurile comune, cele care apar în ambele seturile de date, respectiv toate atributele;
- **left**: reține doar cazurile care apar în setul de date conectat la portul left, respectiv toate atributele;

- **right**: reține doar cazurile care apar în setul de date conectat la portul right, respectiv toate atributele;
- **outer**: reține cazurile care apar în cel puțin unul dintre seturile de date, respectiv toate atributele (Figura 5.3-7).

Figura 5.3-7. Tipuri de join: inner, outer, left, right

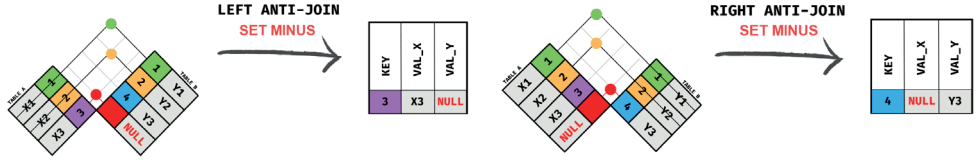


Sursa: (Grolemund & Wickham, 2017)

Unirea tabelelor cu păstrarea cazurilor specifice unui tabel (Set Minus)

Operatorul „Set Minus" identifică și returnează într-un tabel de date cazurile care se regăsesc doar în primul tabel (Figura 5.3-8) (numit și left; în limbaj SQL este vorba de comanda left anti-join, fiind posibil să avem și o comandă right anti-join). Identificarea acestor cazuri se poate realiza doar dacă ambele seturi de date conțin un atribut de tip id identic (același nivel de măsurare). Nu este nevoie ca seturile de date inițiale să aibă aceleași alte atribute. În Figura 5.3-9 am prezentat un proces care ilustrează utilizarea acestui operator.

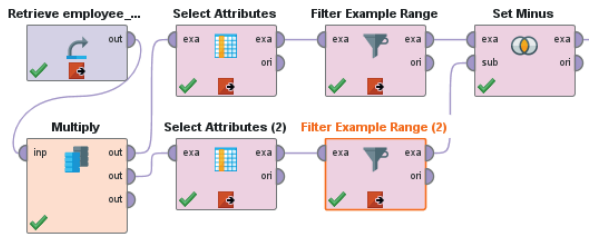
Figura 5.3-8. Tipuri de join: Set Minus



Sursa: (Grolemund & Wickham, 2017)

Figura 5.3-9. Unirea tabelelor cu păstrarea cazurilor specifice unui tabel (Set Minus)

Pasul 1:
Încărcăm setul de date „employee_attrition” și conectăm operatorul „Set Minus”. Ceilalți operatori ne ajută să producem două seturi de date cu variabile diferite și câteva cazuri comune.



Pasul 2:
Dorim să identificăm cazurile care apar într-un tabel dar nu și în altul.
Primul set include variabilele Id, Attrition și Department. Setul secund include atributele Id, Age și Education.
Observăm că variabila numită Id este comună și este definită ca variabilă specială de tip id.
Cazurile cu Id 1-4 apar doar în primul set, 5-10 sunt comune, iar 11-20 apar doar în setul secund.

Id	Attrition	Department
1	Yes	Sales
2	No	Research & ...
3	Yes	Research & ...
4	No	Research & ...
5	No	Research & ...
6	No	Research & ...
7	No	Research & ...
8	No	Research & ...
9	No	Research & ...
10	No	Research & ...

Id	Age	Education
5	27	Below College
6	32	College
7	59	Bachelor
8	30	Below College
9	38	Bachelor
10	36	Bachelor
11	35	Bachelor
12	29	College
13	31	Below College
14	34	College
15	28	Bachelor
16	29	Master
17	32	College
18	22	College
19	53	Master
20	38	Bachelor

Rezultat:
Tabelul rezultat va include doar cazurile care apar în primul set de date (Id 1-4).

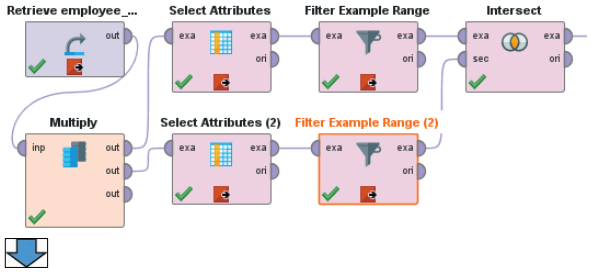
Id	Attrition	Department
1	Yes	Sales
2	No	Research & ...
3	Yes	Research & ...
4	No	Research & ...

Unirea tabelelor cu păstrarea cazurilor comune (Intersect)

Așa cum indică și numele, operatorul Intersect reține doar cazurile comune din două seturi de date (care au același Id). Atributul de tip id trebuie să apară în ambele seturi și să fie definit identic. În rest, cele două seturi de date nu trebuie să aibă aceleași atribute, nici ca celelalte atribute să fie definite la fel (Figura 5.3-10).

Figura 5.3-10. Unirea tabelelor cu păstrarea cazurilor comune (Intersect)

Pasul 1:
Încărcăm setul de date „employee_attrition" și conectăm operatorul Intersect. Ceilalți operatori ne ajută să producem două seturi de date cu variabile diferite și câteva cazuri comune.



Pasul 2:
Dorim să identificăm cazurile comune celor două tabele.
Primul set include variabilele Id, Attrition și Department. Setul secund include atributele Id, Age și Education.
Observăm că variabila numită Id este comună și este definită ca variabilă specială de tip id.
Cazurile cu Id 5-10 sunt comune.

Id	Attrition	Department
1	Yes	Sales
2	No	Research & ...
3	Yes	Research & ...
4	No	Research & ...
5	No	Research & ...
6	No	Research & ...
7	No	Research & ...
8	No	Research & ...
9	No	Research & ...
10	No	Research & ...

Id	Age	Education
5	27	Below College
6	32	College
7	59	Bachelor
8	30	Below College
9	38	Bachelor
10	36	Bachelor
11	35	Bachelor
12	29	College
13	31	Below College
14	34	College
15	28	Bachelor
16	29	Master
17	32	College
18	22	College
19	53	Master
20	38	Bachelor

Rezultat:
Tabelul rezultat include doar cazurile comune (Id 5-10).

Id	Attrition	Department
5	No	Research & Develo...
6	No	Research & Develo...
7	No	Research & Develo...
8	No	Research & Develo...
9	No	Research & Develo...
10	No	Research & Develo...

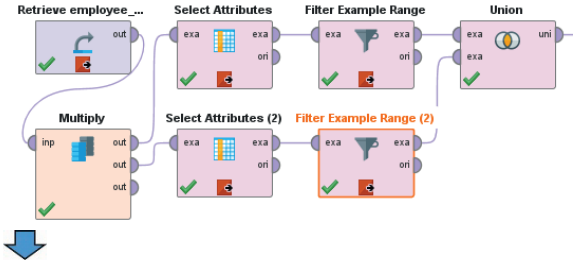
Unirea tabelelor cu păstrarea tuturor cazurilor și atributelor (Union)

Folosim operatorul Union pentru a reține toate cazurile și atributele, comune și necomune, din două seturi de date (Figura 5.3-11). Atributele comune celor două seturi nu vor apărea de două ori. Dacă atributele speciale sunt compatibile, va fi reținut unul, iar acesta va conține valorile ambelor atribute inițiale. Dacă nu sunt compatibile, va fi reținut atributul din primul set de

date. Dacă două atribute au același nume, ele trebuie să fie compatibile, altfel procesul nu va rula.

Figura 5.3-11. Unirea tabelelor cu păstrarea tuturor cazurilor și atributelor (Union)

Pasul 1:
Încărcăm setul de date „employee_attrition" și conectăm operatorul Union. Ceilalți operatori ne ajută să producem două seturi de date necesare pentru a ilustra rolul operatorului Union.



Pasul 2:
Primul set include variabilele Id, Attrition și Department. Setul secund include atributele Id, Age, Department și Education. Observăm că variabila numită Id este comună și este definită ca variabilă specială de tip id. Atributul Department este și el comun. Ambele atribute comune sunt definite identic.
Cazurile cu Id 3-5 sunt comune.

Id	Attrition	Department
1	Yes	Sales
2	No	Research &...
3	Yes	Research &...
4	No	Research &...
5	No	Research &...

Id	Age	Department	Education
3	37	Research &...	College
4	33	Research &...	Master
5	27	Research &...	Below Coll...
6	32	Research &...	College
7	59	Research &...	Bachelor

Rezultat:
Dorim să combinăm toate cazurile și variabilele din cele două tabele. Tabelul produs include toate cazurile și variabilele.
Semnul de întrebare (?) indică faptul că informația respectivă lipsește (nu apărea în niciunul dintre seturile de date inițiale).

Id	Attrition	Department	Age	Education
1	Yes	Sales	?	?
2	No	Research & ...	?	?
3	Yes	Research & ...	?	?
4	No	Research & ...	?	?
5	No	Research & ...	?	?
3	?	Research & ...	37	College
4	?	Research & ...	33	Master
5	?	Research & ...	27	Below Co...
6	?	Research & ...	32	College
7	?	Research & ...	59	Bachelor

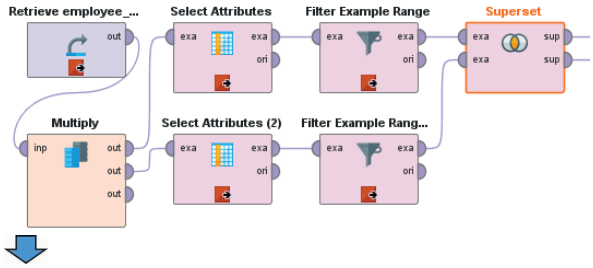
Compatibilizarea structurii a două tabele (Superset)

Operatorul Superset aduce atributele care se regăsesc doar în unul dintre cele două tabele în celălalt tabel (Figura 5.3-12). Astfel, rezultă două tabele noi, fiecare tabel incluzând toate atributele din cele două seturi inițiale. Fiecare dintre tabelele produse poate avea un număr de cazuri cuprins în intervalul definit de numărul de cazuri din tabelul original și suma numărului de cazuri

din cele două tabele inițiale (funcție de numărul de cazuri care apar în ambele tabele inițiale). În cazul noilor atribute introduse într-un tabel, răspunsurile apar ca valori lipsă (?). Un astfel de set de date poate avea un singur atribut special de un anumit tip.

Figura 5.3-12. Compatibilizarea structurii a două tabele (Superset)

Pasul 1:
Încărcăm setul de date „employee_attrition" și conectăm operatorul Superset. Ceilalți operatori ne ajută să producem două seturi de date necesare pentru a ilustra rolul operatorului Superset.



Pasul 2:
Primul set include variabilele Id, Attrition, Department și JobRole. Setul secund include atributele Id, Age, Department și Education.
Observăm că atributele Id și Department sunt comune și definite identic.
Cazurile cu Id 3-5 sunt comune.

Id	Attrition	Department	JobRole
1	Yes	Sales	Sales Exec...
2	No	Research &...	Research ...
3	Yes	Research &...	Laboratory ...
4	No	Research &...	Research ...
5	No	Research &...	Laboratory ...

Id	Age	Department	Education
3	37	Research &...	College
4	33	Research &...	Master
5	27	Research &...	Below Coll...
6	32	Research &...	College
7	59	Research &...	Bachelor

Rezultat:
Dorim să combinăm atributele din cele două seturi de date. După rularea comenzii rezultă două tabele, unul pentru fiecare dintre cele două tabele inițiale. Fiecare dintre tabelele produse include toate atributele și cazurile din tabelul original plus toate atributele diferite identificate în celălalt tabel.
„?" indică faptul că informația respectivă lipsește.

Id	Attrition	Department	Age	Education
1	Yes	Sales	?	?
2	No	Research &...	?	?
3	Yes	Research &...	?	?
4	No	Research &...	?	?
5	No	Research &...	?	?

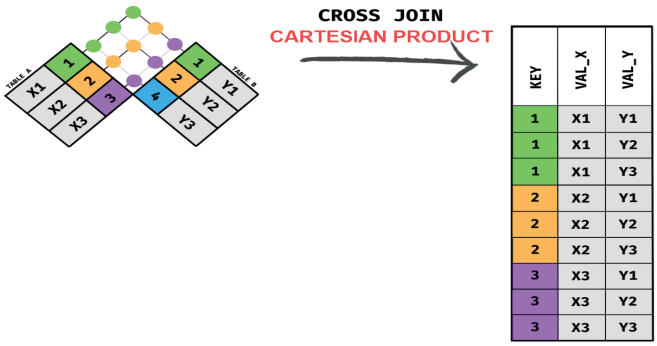
Id	Age	Department	Education	JobRole
3	37	Research &...	College	?
4	33	Research &...	Master	?
5	27	Research &...	Below Col...	?
6	32	Research &...	College	?
7	59	Research &...	Bachelor	?

Produsul cartezian a două tabele (Cartesian Product)

Uneori dorim să producem un set de date ca produs cartezian al altor două seturi de date. Pentru a realiza acest lucru folosim operatorul „Cartesian

Product" (Figura 5.3-13). Un exemplu de utilizare a acestui operator în RapidMiner apare în Figura 5.3-14.

Figura 5.3-13. Tipuri de join: Cartesian Product



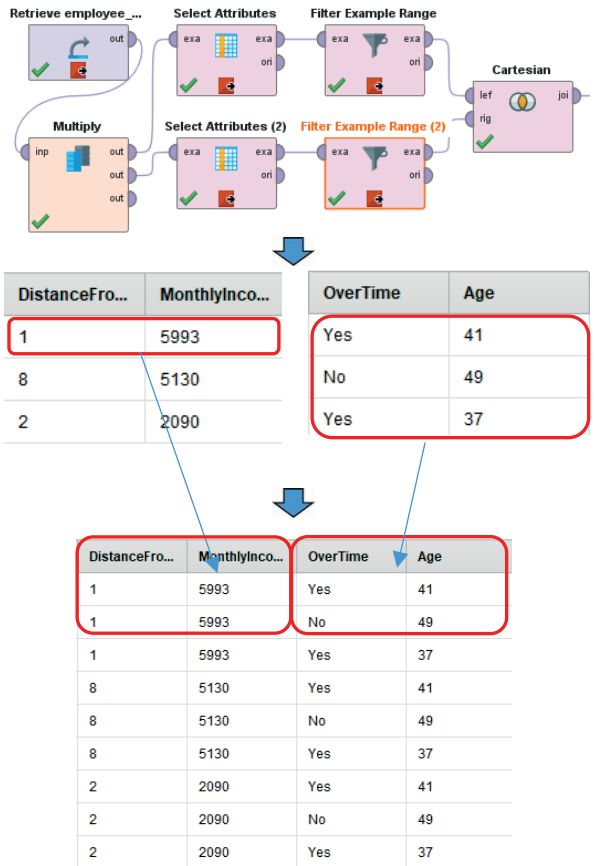
Sursa: (Grolemund & Wickham, 2017)

Figura 5.3-14. Produsul cartezian a două tabele (Cartesian Product)

Pasul 1:
Încărcăm setul de date „employee_attrition" și conectăm operatorul „Cartesian Product". Ceilalți operatori ne ajută să producem două seturi de date necesare pentru a ilustra rolul acestui operator.

Pasul 2:
Primul set include atributele DistanceFromHome și MontlyIncome iar setul secund atributele OverTime și Age. Observăm că toate variabilele cu excepția OverTime sunt numerice.

Rezultat:
După rularea comenzii rezultă un set de date care include toate atributele.
În noul tabel, fiecare caz din primul set este multiplicat de n ori, unde n este egal cu numărul de cazuri din setul secund. Valorile noilor atribute (cele care nu apăreau în primul set) sunt cele din setul secund.



5.4. Lucrul cu valori (Values)

Operatorii utili pentru lucrul cu valori sunt grupați în categoria Values. Folosind acești operatori putem realiza următoarele acțiuni: redenumirea valorilor (**Map**), înlocuirea valorilor (**Replace**), înlocuirea valorilor cu ajutorul unui dicționar (**Replace (Dictionary)**), divizarea variantelor de răspuns asociate atributelor nominale (**Split**), eliminarea unor secțiuni din valori (**Cut**), eliminarea spațiilor din valori (**Trim**), unirea a două sau mai multe valori (**Merge**), adăugarea unei categorii de răspuns la categoriile unui atribut nominal (**Add**), redefinirea valorilor de tip binominal (**Remap Binominals**), setarea valorilor (**Set Data**) și ajustarea valorilor de tip dată (**Adjust Date**).

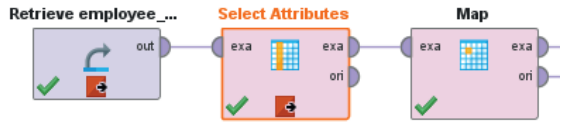
Redenumirea valorilor (Map)

Pentru a redenumi valorile (categoriile de răspuns) unui atribut (de tip nominal sau numeric) folosim operatorul Map. Acesta permite realizarea unor redenumiri simple cu ajutorul parametrului „values mappings". În Figura 5.4-1 am prezentat câteva exemple de redenumire a unor variante de răspuns. De exemplu, am înlocuit varianta de răspuns Male cu bărbat, respectiv „Very high" cu 4.

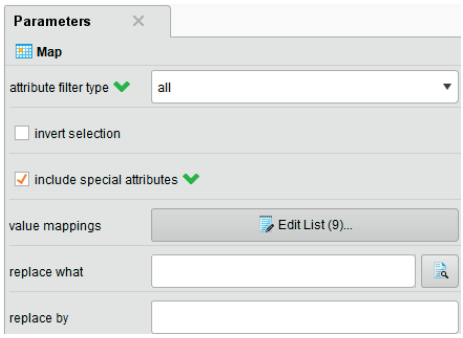
Pentru redenumiri mai complexe, definite general, ca paternuri de litere și cifre folosim parametrul „replace what". Aici definim valorile de interes folosind regex (regular expression). De exemplu, dacă dorim să căutăm toate variantele de răspuns care au în componența lor cel puțin o literă, folosim expresia [a-zA-Z], iar pentru cele care au cel puțin o cifră folosim expresia [0-9].

Figura 5.4-1. Redenumirea valorilor (Map)

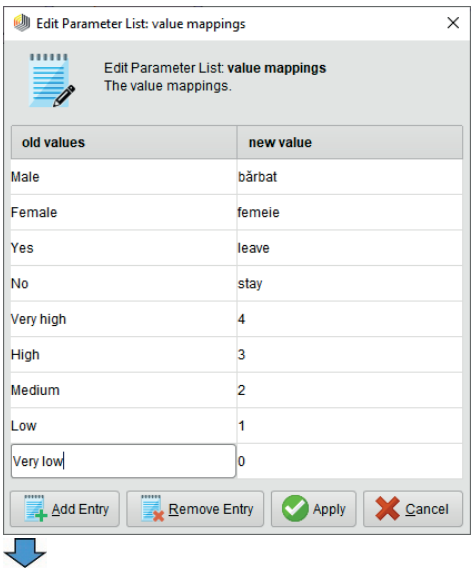
Pasul 1:
Încărcăm setul de date „employee_attrition” și conectăm operatorul „Map”. Operatorul Select ne ajută să producem un set de date mai simplu.



Pasul 2:
Cel mai important parametru este „values mappings”. Aici indicăm valorile de răspuns pe care dorim să le recodăm și care sunt noile valori. Pentru a defini căutări și înlocuiri mai complexe folosim parametrul „replace what” (permite definirea căutărilor folosind formatul regex - regular expression – adică paternuri de litere și cifre).



Pasul 3:
La parametrul „values mappings” atribuim valorilor vechi noile valori. Putem atribui valori noi de tip text sau numerice. Chiar dacă am atribuit valori numerice, atributul respectiv va păstra nivelul de măsurare anterior. De exemplu, atributul JobSatisfaction (Very high – Very low) va rămâne în continuare de tip polinominal, deși noile valori atribuite sunt cifre. Dacă dorim, putem să-l transformăm în numeric (integer în acest caz).



Rezultat:
După rularea comenzii obținem un set de date care are variantele noi de răspuns. JobSatisfaction e definit în continuare ca atribut de tip polinominal chiar dacă răspunsurile sunt numerice.

Attrition	Gender	JobSatisf...
Yes	Female	Very high
No	Male	Medium
Yes	Male	High
No	Female	High

Attrition	Gender	JobSatis...
leave	femeie	4
stay	bărbat	2
leave	bărbat	3
stay	femeie	3

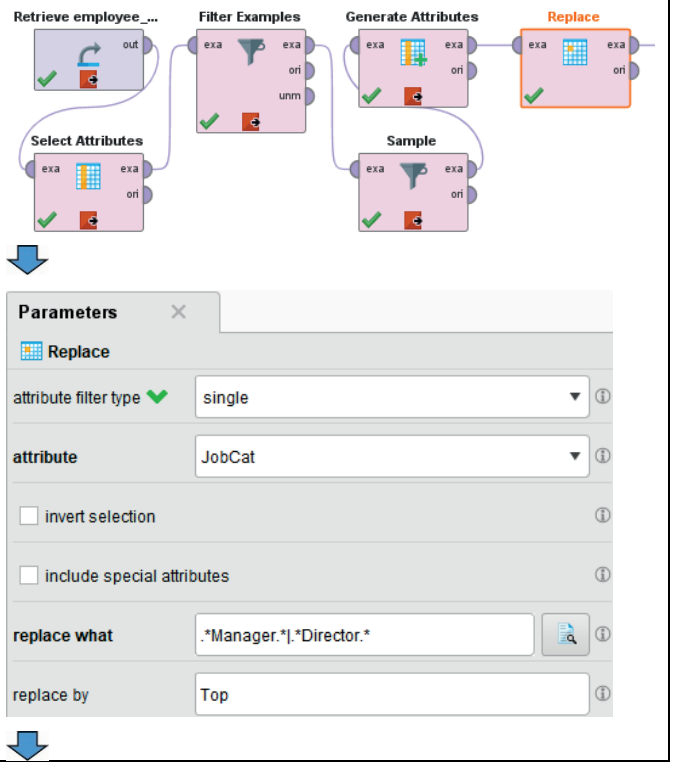
Înlocuirea valorilor (Replace)

Operatorul Replace poate fi folosit pentru a înlocui variantele de răspuns (fragmente din acestea) cu alte variante de răspuns (fragmente). Înlocuirea se poate face pentru unul sau mai multe atribute simultan. În exemplul din Figura 5.4-2 am arătat cum putem înlocui toate variantele de răspuns care conțin cel puțin unul dintre cuvintele indicate (Manager sau Director) cu o altă variantă de răspuns (Top). Definirea fragmentelor sau paternurilor de text care ne interesează (litere, cifre, alte caractere sau combinații ale acestora) se realizează cu ajutorul regex (regular expressions). Pentru mai multe informații relativ la regex și exemple de definire a paternurilor de căutare se pot consulta diferite pagini web.[39]

Figura 5.4-2. Înlocuirea valorilor (Replace)

Pasul 1:
Încărcăm setul de date „employee_attrition" și conectăm operatorul „Replace". Ceilalți operatori ne ajută să producem un set de date potrivit pentru acest exemplu.

Pasul 2:
Prima dată indicăm care este atributul în care dorim să facem înlocuirea.
La parametrul „replace what" definim expresia pe care dorim să o folosim pentru înlocuire.
Aici, variantele de răspuns care conțin oricare dintre cuvintele Manager sau Director, poziționate oriunde, sunt înlocuite cu Top.



[39] https://www.regular-expressions.info/

Rezultat:
După rularea comenzii observăm că atributul JocCat ia valoarea Top pentru toate cazurile care conţin unul dintre cuvintele Manager sau Director la atributul JobRole.

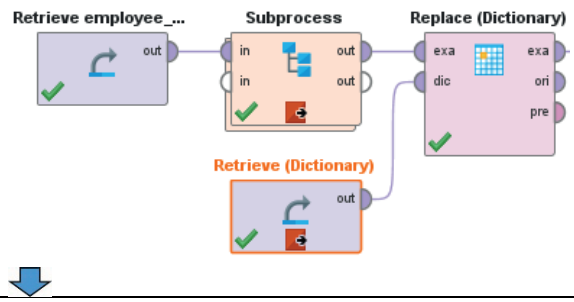
Id	JobCat	JobRole
225	Top	Manufacturing Director
446	Top	Manager
542	Top	Research Director
708	Top	Manufacturing Director
742	Top	Manager
962	Research Scientist	Research Scientist
1015	Top	Research Director
1102	Research Scientist	Research Scientist

Înlocuirea valorilor cu ajutorul unui dicţionar (Replace (Dictionary))

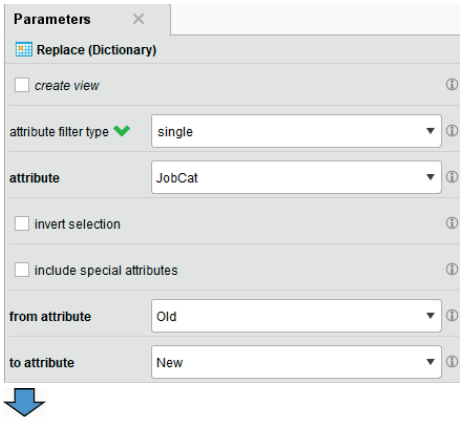
Dacă dorim să înlocuim valorile (părţi ale acestora) asociate unor atribute nominale cu valori predefinite şi stocate într-un dicţionar, folosim operatorul „Replace (Dictionary)". În practică, dicţionarul e de fapt un set de date cu două coloane (atribute), prima asociată valorii vechi, a doua celei noi. Prin urmare, pentru a ilustra utilizarea acestui operator avem nevoie de două seturi de date, cel de lucru, în care dorim să facem înlocuirile, şi setul de date (dicţionarul) care conţine corespondenţa dintre vechile şi noile valori. Pentru exemplul din Figura 5.4-3 am produs un set de date cu atributele nominale Old şi New, unde Old conţine tipurile de poziţii (job-uri) (vezi atributul JobRole din setul de date employee_attrition), iar New specifică noile categorii (valori) asociate acestora (Low, Middle, Top).

Figura 5.4-3. Înlocuirea valorilor folosind un dicţionar (Replace (Dictionary))

Pasul 1:
Încărcăm seturile de date „employee_attrition" şi „job_role" (dicţionarul), apoi le conectăm la operatorul „Replace (Dictionary)". Ceilalţi operatori ne ajută să producem un set de date potrivit pentru acest exemplu.



Pasul 2:
Prima dată indicăm care este atributul pentru care dorim să facem înlocuirea valorilor.
La parametrul „from attribute" alegem Old (numele vechii categorii de răspuns), iar la „to attribute" alegem New (numele noii categorii de răspuns).



Rezultat:
După rularea comenzii observăm că atributul JocCat ia valorile Top, Middle, Low, funcție de valoarea indicată în dicționar. Atributul JobRole conține vechile valori, pentru comparație.

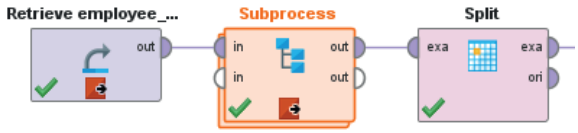
Id	JobCat	JobRole
91	Middle	Healthcare Representative
122	Low	Sales Executive
334	Middle	Healthcare Representative
732	Middle	Research Scientist
733	Low	Laboratory Technician
755	Middle	Sales Representative

Dividerea valorilor (Split)

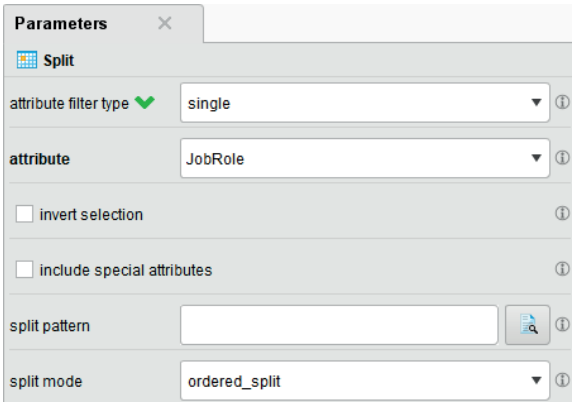
Operatorul Split divide în două sau mai multe părți valorile luate de un atribut de tip nominal rezultând astfel două sau mai multe atribute de tip nominal. Caracterul relativ la care se realizează diviziunea poate fi ales în funcție de situația concretă. Acesta poate fi un spațiu, o virgulă, un punct sau oricare alt caracter. De exemplu, un atribut cu valoarea „Research Scientist", poate fi divizat (raportat la caracterul spațiu) în două atribute, unul cu valoarea „Research", altul „Scientist". Diviziunea valorilor se poate face în două moduri („split pattern"): ordonat (ordered_split) sau neordonat (unordered_split). Diviziunea de tip ordonat produce un număr de atribute egal cu numărul de sub-diviziuni asociate cazului care are cele mai multe sub-diviziuni. Diviziunea de tip neordonat produce un număr de atribute egal cu numărul total de sub-diviziuni diferite posibile (conform specificațiilor).

Figura 5.4-4. Diviziunea valorilor (Split)

Pasul 1:
Încărcăm setul de date „employee_attrition" și conectăm operatorul „Split". Ceilalți ne ajută să producem un set de date potrivit.



Pasul 2:
Indicăm atributul pe care dorim să-l dividem (putem alege mai multe atribute simultan).
La parametrul „split pattern" trecem caracterul în funcție de care dorim să facem diviziunea. Aici e spațiu, deci nu este vizibil în imagine.
La „split pattern" alegem ordered_split (la final reluăm analiza folosind unordered_split).



Rezultat (ordered_split):
După rularea comenzii obținem două atribute nominale noi (deoarece, indiferent de caz, caracterul utilizat pentru separare - spațiu - apare cel mult o dată). Primul atribut conține prima parte a vechiului conținut (de la început până la prima apariție a caracterului spațiu). Al doilea atribut ia valorile celei de a doua secțiuni.

Id	JobName	JobRole_1	JobRole_2
91	Healthcare Representative	Healthcare	Representative
733	Laboratory Technician	Laboratory	Technician
851	Sales Representative	Sales	Representative
908	Manager	Manager	?
999	Research Scientist	Research	Scientist
1026	Sales Executive	Sales	Executive
1105	Research Scientist	Research	Scientist

Rezultat (unordered_split):
Aici numărul de atribute binominale rezultat este egal cu numărul total de sub-diviziuni diferite. Valorile luate de aceste atribute sunt true și false. True apare dacă acel caz conține o anumită sub-diviziune.

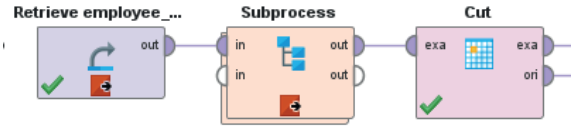
Id	JobName	JobRole_Executive	JobRole_Healthcare
91	Healthcare Representative	false	true
733	Laboratory Technician	false	false
851	Sales Representative	false	false
908	Manager	false	false
999	Research Scientist	false	false
1026	Sales Executive	true	false
1105	Research Scientist	false	false

Eliminarea unei secțiuni (Cut)

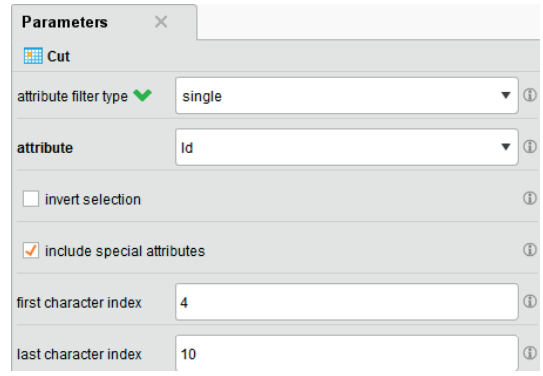
Operatorul Cut e util pentru a elimina părți din variantele de răspuns ale unor atribute de tip nominal. Eliminarea se realizează în funcție de pozițiile caracterelor, utilizatorul având posibilitatea să indice primul și ultimul caracter selectat. Vom obține astfel toate caracterele începând cu primul indicat până la ultimul indicat.

Figura 5.4-5. Eliminarea unei secțiuni (Cut)

Pasul 1:
Încărcăm setul de date „employee_attrition" și conectăm operatorul „Cut". Operatorul Select ne ajută să producem un set de date potrivit pentru acest exemplu.



Pasul 2:
Selectăm atributul de interes (numit Id, în acest caz).
La parametrul „first character index" indicăm poziția primului caracter care dorim să rămână.
La parametrul „last character index" indicăm poziția ultimului caracter care dorim să rămână.



Rezultat:
După rularea comenzii, valorile atributului Id nu mai includ secvența „Id_" (așa cum apărea inițial) (primele trei caractere au fost eliminate).

Id
Id_91
Id_733
Id_908
Id_1026
Id_1105

Id
91
733
908
1026
1105

Eliminarea spațiilor (Trim)

Operatorul Trim elimină caracterele de tip spațiu ce apar înaintea valorilor sau după valorile unui atribut de tip nominal (spațiile care apar în interiorul textului nu sunt eliminate). Astfel, în urma aplicării operatorului Trim,

variantele de răspuns „person 1”, „person 2”, „person 3” devin „person 1”, „person 2”, „person 3”.

Unirea valorilor (Merge)

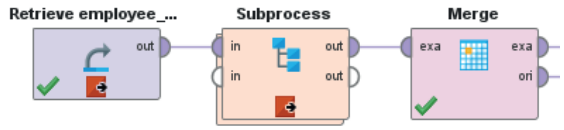
Operatorul Merge este utilizat pentru a uni două valori de tip text. Cel puțin una dintre cele două valori trebuie să apară în atributul selectat. Merge poate fi aplicat doar unor atribute obișnuite. Figura 5.4-6 prezintă un exemplu simplu de utilizare a acestui operator.

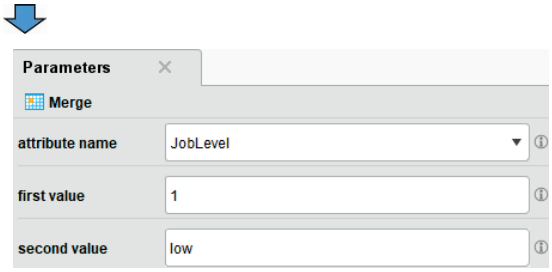
Figura 5.4-6. Unirea valorilor (Merge)

Pasul 1:
Încărcăm setul de date „employee_attrition” și conectăm operatorul „Merge”. Ceilalți operatori ne ajută să producem un set de date potrivit.

Pasul 2:
La „attribute name” indicăm atributul pe care dorim să-l modificăm. La „first value” indicăm valoarea pe care dorim să o modificăm. La „second value” indicăm valoarea pe care dorim să o adăugăm la prima valoare.

Rezultat:
După rularea comenzii obținem observăm că valorile 1 au devenit 1_low.





Id	JobLevel
91	4
733	1
908	5
1026	2
1105	1

Id	JobLevel
91	4
733	1_low
908	5
1026	2
1105	1_low

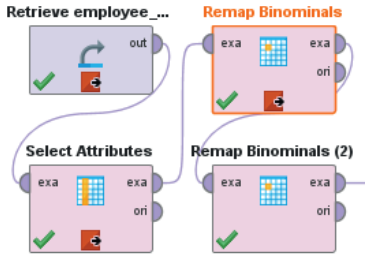
Re-maparea valorilor binominale (Remap Binominals)

Acest operator poate fi aplicat doar unor atribute definite explicit ca binominale. În cazul unor astfel de atribute, uneori dorim să inversăm ordinea celor două variante de răspuns, mai exact să o definim pe alta dintre ele ca varianta pozitivă / de interes. Pentru a face acest lucru, softul va

schimba doar valorile alocate intern celor două variante de răspuns (prin urmare schimbarea nu va fi vizibilă explicit în setul de date). Exemplu din Figura 5.4-7 ilustrează rolul acestui operator.

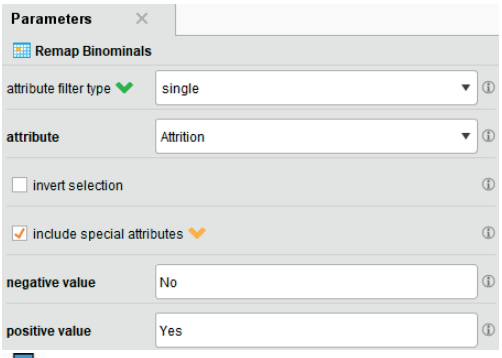
Figura 5.4-7. Re-maparea valorilor binominale (Remap Binominals)

Pasul 1:
Încărcăm setul de date „employee_attrition" și conectăm operatorul „Remap Binominals" de două ori (dorim să remapăm două atribute diferite).
În acest moment valorile pozitive și negative atribuite convențional atributelor Gender și Attrition arată așa:

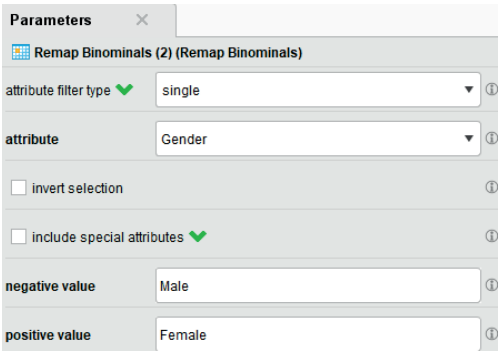


Label Attribution	Binominal	0	Negative Yes	Positive No
Gender	Binominal	0	Negative Female	Positive Male

Pasul 2:
Selectăm atributul ale cărui valori dorim să le remapăm (Attrition).
Pentru că este un atribut special, bifăm parametrul „include special attributes".
La „negative value" și „positive value" indicăm variantele de răspuns pe care dorim să le considerăm pozitive / negative.



Pasul 3:
Selectăm atributul ale cărui valori dorim să le remapăm (Gender).
La „negative value" și „positive value" indicăm variantele de răspuns pe care dorim să le considerăm pozitive / negative.



Rezultat:
După rularea comenzii, cele două atribute, Attrition și Gender, au fiecare valorile pozitive și negative inversate comparativ cu starea inițială. Astfel, valorile considerate pozitive sunt acum Yes și Female (anterior erau No și Male).

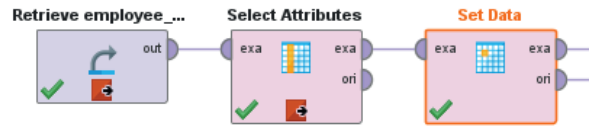
	Label			Negative	Positive
✓	Attrition	Binominal	0	No	Yes
✓	⚠ Gender	Binominal	0	Male	Female

Setarea valorilor (Set Data)

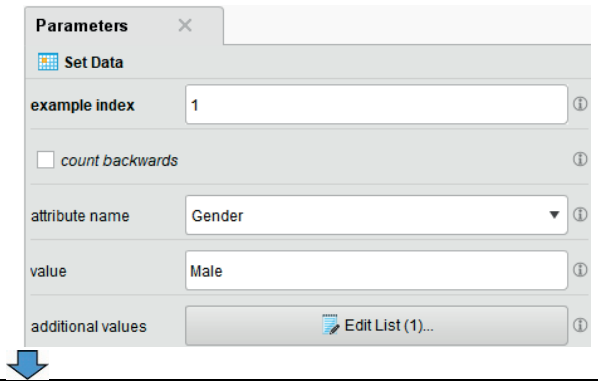
Folosind acest operator putem modifica valorile unor atribute pentru cazurile specificate (Figura 5.4-8). Indicarea cazurilor se realizează folosind indexul intern al softului. Pentru ca înlocuirea să fie realizată, este necesar ca valorile indicate să fie de același tip cu cele ale atributului căruia i se aplică. De exemplu, dacă atributul este numeric, valoarea nouă va trebui să fie un număr.

Figura 5.4-8. Setarea valorilor (Set Data)

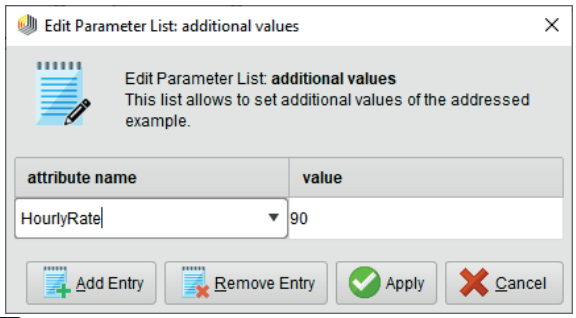
Pasul 1:
Încărcăm setul de date „employee_attrition" și conectăm operatorul „Set Data".



Pasul 2:
La parametrul „example index" indicăm numărul de ordine al cazului pentru care dorim să schimbăm valorile. La „attribute name" alegem atributul iar la „value" valoarea nouă dorită (valoarea veche va fi înlocuită cu aceasta).



Pasul 3:
La parametrul „additional values" putem alege și alte atribute.
Aici am ales un atribut numeric (HourlyRate) și am specificat că dorim să înlocuim vechea valoare cu valoarea 90.



Rezultat:
Observăm că primul caz ia valori diferite la atributele Gender și HourlyRate.

Id	Attrition	Gender	HourlyRate
1	Yes	Female	94
2	No	Male	61
3	Yes	Male	92

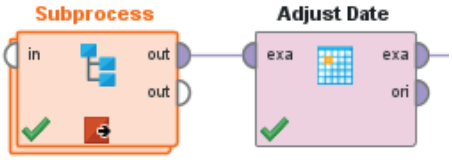
Id	Attrition	Gender	HourlyRate
1	Yes	Male	90
2	No	Male	61
3	Yes	Male	92

Ajustarea valorilor de tip dată (Adjust Date)

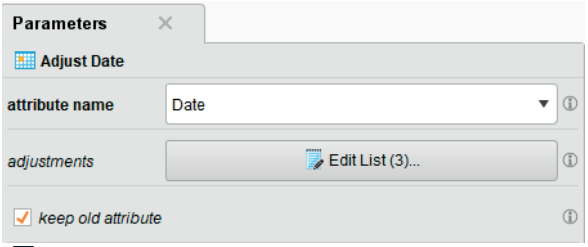
Acest operator este folosit pentru a aduna sau scădea o anumită perioadă de timp (secunde, minute, ore, zile etc.) relativ la un atribut de tip dată. Putem aduna / scădea simultan un anumit număr de ore, zile, luni etc. Dacă dorim, putem păstra și vechiul atribut (Figura 5.4-9).

Figura 5.4-9. Ajustarea valorilor de tip dată (Adjust Date)

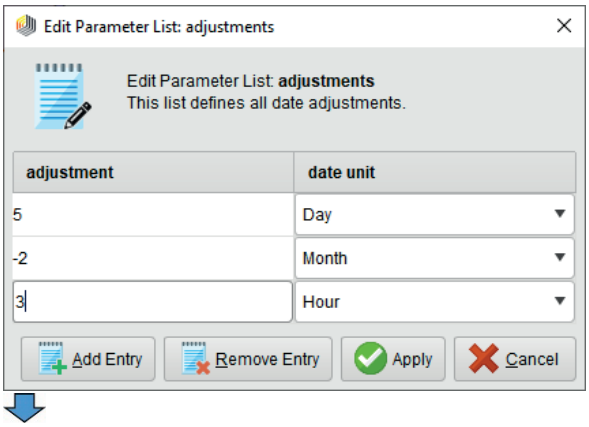
Pasul 1:
Subprocesul produce setul de date necesar. Conectăm operatorul „Adjust Date".



Pasul 2:
La parametrul „attribute name" alegem atributul pe care dorim să-l modificăm. Bifăm faptul că dorim să păstrăm și vechiul atribut.



Pasul 3:
La parametrul „adjustments" specificăm valorile și unitățile de timp dorite în funcție de care va fi realizată modificarea atributului Date. În acest exemplu adunăm 5 zile, scădem 2 luni și adunăm 3 ore.



Rezultat:
După rularea comenzii observăm că noua variabilă, de același tip, conține datele modificate conform instrucțiunilor anterioare.

Row No.	Date	Date_adjusted
1	Oct 30, 2014 11:33:34 AM EET	Sep 4, 2014 2:33:34 PM EEST
2	Apr 3, 2016 11:33:34 AM EEST	Feb 8, 2016 2:33:34 PM EET
3	Apr 20, 2016 11:33:34 AM EEST	Feb 25, 2016 2:33:34 PM EET
4	May 5, 2016 11:33:34 AM EEST	Mar 10, 2016 2:33:34 PM EET
5	Sep 24, 2016 11:33:34 AM EEST	Jul 29, 2016 2:33:34 PM EEST

6. „Curățarea" și transformarea datelor (Cleansing)

Categoria Cleansing include mai multe sub-categorii de operatori:

- **Normalization**: conține comenzile care normalizează atributele;
- **Binning**: conține comenzile care ne ajută să grupăm valorile atributelor;
- **Missing**: conține comenzile care ne ajută să înlocuim valorile lipsă;
- **Duplicates**: conține comenzile care ne ajută să identificăm și eliminăm cazurile duplicate;
- **Outliers**: conține comenzile care ne ajută să identificăm cazurile extreme;
- **Dimensionality Reduction**: conține comenzile care ne ajută să reducem numărul de atribute păstrând în același timp cât mai mult din informația asociată acestora;
- **Statistics** și **Quality Measures**: calculează diferite măsuri statistice și ale calității atributelor.

6.1. Normalizarea variabilelor (Normalization)

Operatorii grupați în categoria Normalization pot fi aplicați doar unor variabile metrice (numerice). Există trei operatori de acest tip, fiecare cu funcții specifice:

- Normalizarea (**Normalize**);
- Denormalizarea (**De-Normalize**);
- Scalarea în funcție de importanță (**Scale by Weights**).

Normalizarea (Normalize)

Atributele metrice cu care lucrăm au adesea diferite unități de măsură, iar valorile au intervale de variație foarte diferite. De exemplu, vârsta unui angajat ia cel mai adesea valori în intervalul 18-65, în timp ce salariul lunar poate lua valori de cel puțin o 100 de ori mai mari; venitul unei persoane poate fi măsurat în RON sau **mii** RON, deci valoarea înregistrată poate fi 3 sau 3000, funcție de unitatea de măsură folosită. În cazul anumitor tipuri de analiză (de exemplu, cele care folosesc distanțe de tip Euclidian), diferențele mari între scalele și unitățile de măsură ale atributelor sunt problematice. O comparație corectă ar presupune scale similare. Operatorul Normalization face tocmai acest lucru (Figura 6.1-1). În esență, normalizarea înseamnă transformarea (translatarea) valorilor unui atribut astfel încât să acopere un interval specific (simultan, unitatea de măsură inițială dispare, fiind înlocuită de o unitate oarecum fictivă, trans-atribut, precum abaterea standard).

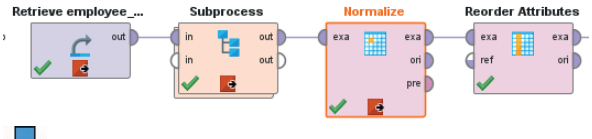
Normalizarea se poate realiza folosind una dintre următoarele soluții:

- **Z-transformation**: transformarea valorilor cu ajutorul scorurilor Z, numită și standardizare; pentru a realiza această transformare, scădem din vechea valoare media atributului respectiv și împărțim rezultatul la abaterea standard a acelui atribut; variabila transformată va avea media 0 (deci va lua valori pozitive și negative) și abaterea standard 1; acest tip de transformare păstrează distribuția originală a atributului și este mai puțin influențată de valorile extreme (outliers);
- **range transformation**: se referă la translatarea valorilor din intervalul original de variație în altul, același pentru toate atributele; de obicei, acest interval este [0;1] sau [-1;1]; de exemplu, pentru a transforma valorile unui atribut în intervalul [0;1], scădem din fiecare valoare valoarea minimă și împărțim rezultatul la valoarea maximă; acest tip de transformare păstrează distribuția originală a atributului, dar este influențată relativ mai mult de valorile extreme;
- **proportion transformation**: normalizarea se obține prin împărțirea valorilor originale la suma acestor valori;

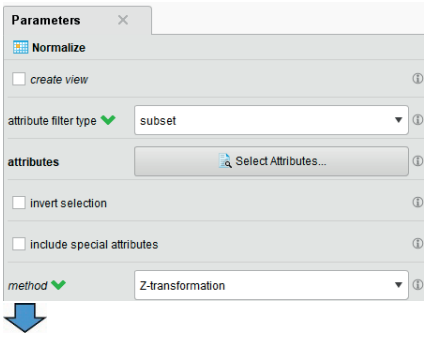
– **interquartile range**: în acest caz normalizarea este realizată în relație cu valoarea mediană (Q2 = quartila 2, adică percentila 50) și abaterea interquartilică (IQR) (diferența dintre quartila 3 și 1, adică percentila 75 și 25) a atributului respectiv; pentru a obține valorile normalizate folosind această metodă, scădem din valoarea originală Q2 și împărțim rezultatul la IQR; ca urmare a faptului că primele și ultimele 25% dintre cazuri (sortate ascendent) sunt ignorate atunci când calculăm Q2 și IQR, rezultatul este foarte puțin influențat de posibilele valori extreme.

Figura 6.1-1. Normalizarea (Normalize)

Pasul 1:
Conectăm datele și operatorii conform imaginii alăturate (a se vedea și procesul).



Pasul 2:
În cazul operatorului Normalize, alegem subset la „attribute filter type” și apoi indicăm cele două atribute pe care dorim să le normalizăm (parametrul attributes).
La parametrul method alegem Z-transformation (scoruri Z).



Rezultat:
Observăm că atributele ...Z conțin scorurile standardizate ale vechilor atribute.

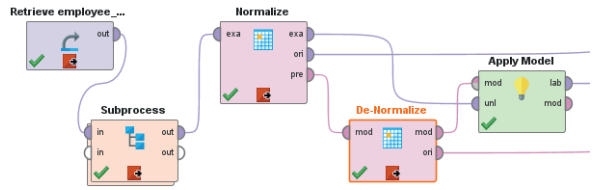
Id	DailyRate	DailyRateZ	DistanceFromHome	DistanceFromHomeZ
91	530	-0.847	1	-0.866
334	1001	0.361	7	-0.014
732	1097	0.608	11	0.553

De-normalizarea (De-Normalize)

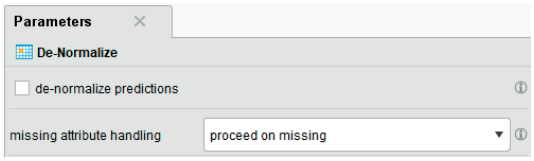
Normalizarea datelor este necesară cel mai adesea, deci majoritatea variabilelor utilizate în analize nu au nici o unitate de măsură, nici valori ușor interpretabile. Pentru a rezolva această problemă, putem denormaliza atributele respective (de exemplu atributele cu predicțiile obținute în urma rulării unui model). Operatorul De-Normalize face exact acest lucru.

Figura 6.1-2. Denormalizarea (De-Normalize)

Pasul 1:
Conectăm datele și operatorii conform imaginii alăturate (a se vedea și procesul).



Pasul 2:
Prima dată normalizăm atributele, apoi le de-normalizăm. În acest caz nu trebuie să schimbăm valoarea parametrilor operatorului De-Normalize.



Rezultat:
Observăm că atributele conțin valorile inițiale (nestandardizate) ale atributelor.

Z-Transformation

```
Normalize 2 attributes to mean 0 and variance 1.
Using
DailyRate --> mean: 860.1, variance: 151922.32222222225
DistanceFromHome --> mean: 7.1, variance: 49.65555555555555
```

Id	DailyRate	Distan...
91	530.000	1
334	1001.000	7.000
732	1097.000	11.000
733	109.000	5.000
851	862.000	2
873	1146.000	25
908	1099.000	5.000
999	683.000	2
1026	1476.000	4.000
1105	598.000	9.000

Scalarea în funcție de importanță (Scale by Weights)

Operatorul Normalize urmărește să crească similitudinea dintre scalele diferitelor atribute dintr-un set de date, să le facă cât mai comparabile între ele. Scopul este de a acorda atributelor aceleași șanse la punctul de start al unei analize. Uneori știm, din alte analize și/sau din literatură, că unele atribute sunt mai importante pentru o anumită analiză (model de predicție) comparativ cu altele. Spre deosebire de Normalize, operatorul „Scale by Weights” ține cont de importanța diferită a atributelor și le scalează în funcție de aceasta. Astfel, în urma procesului de scalare, atributele importante vor primi scoruri relativ mai mari, iar cele cu o importanță mică, scoruri relativ mai mici. Desigur, pentru a calcula ponderările / importanța atributelor, setul de date trebuie să includă un atribut de tip label (variabilă dependentă).

Importanța unui atribut va fi calculată în raport cu puterea predictivă a acestuia relativ la atributul label.

Figura 6.1-3. Scalarea în funcție de importanța atributelor (Scale by Weights)

Pasul 1:
Conectăm datele și operatorii conform imaginii alăturate (a se vedea și procesul).

Pasul 2:
În urma aplicării operatorului „Weights by Chi Square” obținem importanța fiecărui atribut pentru predicția atributului special de tip label (Attrition).
Conform acestui criteriu, atributul Age e cel mai important.

attribute	weight
PercentSalaryHike	4.712
YearsSinceLastPromotion	10.073
DistanceFromHome	15.666
YearsInCurrentRole	50.737
YearsAtCompany	52.797
Age	75.480

Rezultat:
Observăm că atributele cu o importanță mai mare au primit scoruri mărite cu un factor semnificativ mai mare. Factorul este egal cu ponderările (weights) calculate la pasul 2.

Id	Attrition	Age	DistanceFromHome	PercentSalaryHike
1	Yes	41	1	11
2	No	49	8	23
3	Yes	37	2	15
4	No	33	3	11

Id	Attrition	Age	DistanceFromHome	PercentSalaryHike
1	Yes	3094.689	15.666	51.829
2	No	3698.530	125.331	108.370
3	Yes	2792.768	31.333	70.676
4	No	2490.847	46.999	51.829

6.2. Gruparea valorilor (Binning)

În unele situații poate fi necesar și/sau util să transformăm un atribut de tip numeric în unul de tip nominal. De exemplu, valorile atributului vârstă (măsurat ca număr de ani împliniți) pot fi grupate în intervale de ani (de exemplu <18, 18-34, 35-49, 50-64, 65+) sau în funcție de alt criteriu (de exemplu, grupul 1 format din primele 33.3% cazuri cu vârsta cea mai mică,

grupul 2 format din următoarele 33.3% cazuri în ordinea crescătoare a vârstei și grupul 3 format din ultimele 33.3% cazuri, persoanele cele mai în vârstă). În primul caz obținem un atribut nominal cu 5 clase / categorii, în celălalt unul cu 3 clase. Acest proces de grupare, numit și discretizare, poate fi realizat folosind operatorii incluși în categoria Binning. Fiecare dintre acești operatori realizează discretizarea în funcție de anumite criterii, utilizatorul având posibilitatea să seteze parametrii aferenți. În cele ce urmează vom descrie și exemplifica utilizarea acestor operatori. Tabelul 6.2-1 prezintă sintetic diferențele și asemănările dintre operatorii din categoria Binning (Discretizare).

Tabelul 6.2-1. O comparație a tipurilor de discretizare

Denumirea tipului de discretizare în RM	**Size**	**Binning**	**Freq.**	**User**	**Entropy**
Discretizarea se face în funcție de ...	*# cazurilor*	*range + # grupurilor*	*range + # valorilor*	*depinde*	*depinde*
Numărul de grupuri este stabilit de ...	*utilizator (indirect)*	*utilizator (direct)*	*utilizator (direct)*	*utilizator (direct)*	*soft*
Numărul de cazuri în fiecare grup este ...	*(aproape) egal*	*variabil (depinde)*	*(aproape) egal*	*variabil (depinde)*	*variabil (depinde)*
Intervalele de variație asociate grupurilor sunt ...	*inegale (depinde)*	*egale*	*inegale (depinde)*	*inegale (depinde)*	*inegale (depinde)*

Pentru a clarifica discuția anterioară, oferim aici un exemplu simplu de serie de valori numerice transformată folosind pe rând fiecare dintre operatorii de discretizare (cu excepția operatorului bazat pe entropie; pentru aplicarea acestuia, setul de date trebuie să conțină un atribut de tip label; în plus, rezultatul grupării va fi dependent de relația dintre acest atribut și atributul obișnuit). Să presupunem că avem seria de valori 0, 4, 12, 16, 16, 18, 24, 26, 28 și dorim să le grupăm în trei grupuri. În Tabelul 6.2-2 prezentăm comparativ rezultatele teoretice care ar trebui să fie obținute în urma aplicării operatorilor de discretizare. În Tabelul 6.2-3 prezentăm rezultatele furnizate de RapidMiner. Dat fiind faptul că exemplul are puțin cazuri și că am ales trei cazuri pe grup la operatorul „Discretize by Size”, soluțiile produse de discretizările Size și Frequency sunt identice (în practică, chiar dacă setăm valorile parametrilor astfel încât să rezulte același număr de grupuri, soluțiile obținute vor fi similare, dar foarte rar identice).

Tabelul 6.2-2. Rezultatul discretizării în funcție de tipul acesteia (teoretic)

Tip discretizare	Cazuri			Intervale		
	Grupul 1	Grupul 2	Grupul 3	Grupul 1	Grupul 2	Grupul 3
Size	*0, 4, 12*	*16, 16, 18*	*24, 26, 28*	*[0-14)*	*[14-21)*	*[21+)*
Binning	*0, 4*	*12, 16, 16, 18*	*24, 26, 28*	*[0-9.4)*	*[9.4-18.7)*	*[18.7+)*
Freq.	*0, 4, 12*	*16, 16, 18*	*24, 26, 28*	*[0-14)*	*[14-21)*	*[21+)*
User	*0*	*4*	*12, 16, 16, 18, 24, 26, 28*	*[0]*	*[1-10)*	*[10+)*

Tabelul 6.2-3. Rezultatul discretizării în funcție de tipul acesteia (RapidMiner)

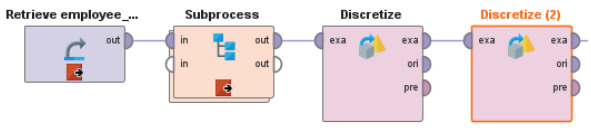
id	orig	d_size	d_binn	d_freq	d_user
1	0	[-∞ - 14.0]	[-∞ - 9.3]	[-∞ - 14.0]	[0]
2	4	[-∞ - 14.0]	[-∞ - 9.3]	[-∞ - 14.0]	[1-10)
3	12	[-∞ - 14.0]	[9.3 - 18.7]	[-∞ - 14.0]	[10-50)
4	16	[14.0 - 21.0]	[9.3 - 18.7]	[14.0 - 21.0]	[10-50)
5	16	[14.0 - 21.0]	[9.3 - 18.7]	[14.0 - 21.0]	[10-50)
6	18	[14.0 - 21.0]	[9.3 - 18.7]	[14.0 - 21.0]	[10-50)
7	24	[21.0 - ∞]	[18.7 - ∞]	[21.0 - ∞]	[10-50)
8	26	[21.0 - ∞]	[18.7 - ∞]	[21.0 - ∞]	[10-50)
9	28	[21.0 - ∞]	[18.7 - ∞]	[21.0 - ∞]	[10-50)

Discretizarea în funcție de numărul cazurilor (Discretize by Size)

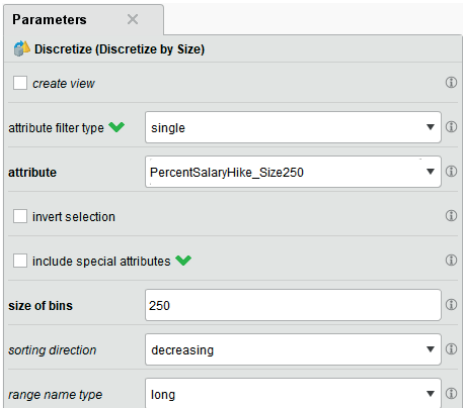
În cazul acestui operator, discretizarea (formarea grupurilor) se realizează în funcție de numărul de cazuri pe care dorim să îl aibă fiecare grup. Numărul de cazuri trebuie indicat de către utilizator. Pornind de la acest număr și ținând cont de numărul total de cazuri din setul de date, operatorul va grupa valorile astfel încât fiecare dintre grupurile rezultate să aibă un număr de cazuri cât mai apropiat de numărul dorit de utilizator. Dacă atributul original are una sau mai multe valori care sunt luate de un număr disproporționat de mare / mic de cazuri, grupurile rezultate vor avea mai multe / puține cazuri decât numărul indicat, deci grupurile vor avea un număr diferit de cazuri. La fel se întâmplă și dacă numărul total de cazuri nu este un multiplu (aproximativ) al numărului de cazuri cerut de utilizator. Lungimea intervalelor de variație (valoarea maximă minus valoarea minimă) ale fiecărui grup poate diferi. Utilizarea acestui operator este ilustrată în Figura 6.2-1.

Figura 6.2-1. Discretizarea în funcție de numărul cazurilor (Discretize by Size)

Pasul 1:
Conectăm datele și operatorii conform imaginii alăturate (a se vedea și procesul).



Pasul 2:
Numărul de cazuri pe care le alocăm la un grup este 250 (500 în exemplul secund). Dat fiind faptul că operatorul încearcă să formeze grupuri de câte 250 cazuri, ne așteptăm să obținem 6 grupuri (numărul total de cazuri este 1470). Deoarece numărul de valori diferite luate de atributul PercentSalaryHike este foarte mic raportat la numărul total de cazuri (mulți angajați au parte de aceeași creștere procentuală), e posibil să obținem mai puține grupuri, respectiv mărimea grupurilor să fie diferită.
La denumirea grupurilor (range name type) am ales long (interval în exemplul secund).



Rezultat:
Numele categoriilor diferă în cele două situații: specificarea intervalului vs. interval + numărul de ordine al intervalului.

Id	PercentSalaryHike_Size500	PercentSalaryHike_Size250	PercentSalaryHike
1	[-∞ - 12.5]	range1 [-∞ - 11.500]	11
2	[15.5 - ∞]	range6 [18.500 - ∞]	23
3	[12.5 - 15.5]	range4 [13.500 - 15.500]	15
4	[-∞ - 12.5]	range1 [-∞ - 11.500]	11
5	[-∞ - 12.5]	range2 [11.500 - 12.500]	12

În primul caz am obținut 6 grupuri. Numărul de cazuri alocat la fiecare grup diferă (198-302). Asta se întâmplă deoarece atributul are multe cazuri cu exact aceleași valori. Aceste tabele apar în perspectiva Results, tabul Statistics.

Index	Nominal value	Absolute count	Fraction
1	range4 [13.500 - 15.500]	302	0.205
2	range6 [18.500 - ∞]	302	0.205
3	range5 [15.500 - 18.500]	249	0.169
4	range1 [-∞ - 11.500]	210	0.143
5	range3 [12.500 - 13.500]	209	0.142
6	range2 [11.500 - 12.500]	198	0.135

Index	Nominal value	Absolute count	Fraction
1	[15.5 - ∞]	551	0.375
2	[12.5 - 15.5]	511	0.348
3	[-∞ - 12.5]	408	0.278

Discretizarea în funcție de numărul grupurilor (Discretize by Binning)[40]

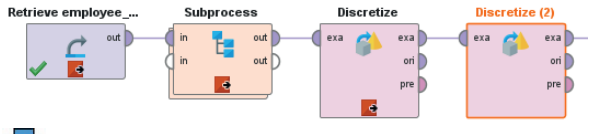
În cazul acestui operator discretizarea se realizează în funcție de numărul de grupuri ales de către utilizator și lungimea intervalului de variație a

[40] Numită în literatura de specialitate „Equal Width Binning”.

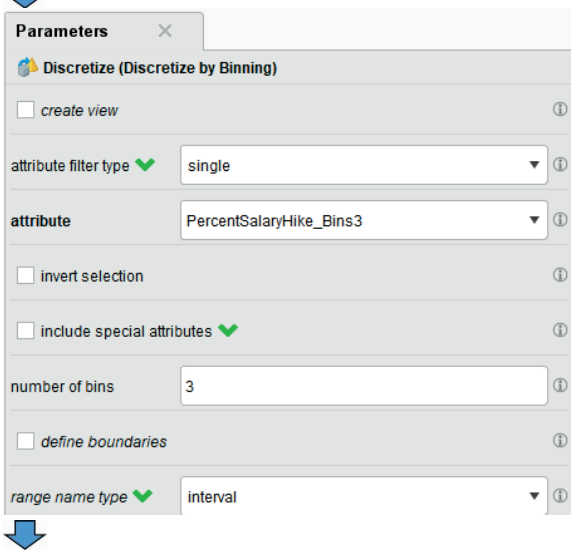
atributului de interes. Algoritmul alocă cazurile la grupuri în funcție de raportul dintre intervalul de variație al atributului și numărul de grupuri indicat. Simplu spus, intervalul de variație al atributului se divide în n intervale, unde n este egal cu numărul dorit de grupuri. Se obține astfel lungimea intervalului. Primul grup va fi format din cazurile cu valori în intervalul [minim, minim + lungime]. Următorul grup va fi format din cazurile cu valori în intervalul [minim + lungime, minim + 2 x lungime] etc. Firesc, intervalele de variație (range) asociate grupurilor vor avea aceeași lungime. Grupurile rezultate pot avea un număr (foarte) diferit de cazuri (funcție de forma distribuției atributului original, diferențele pot fi foarte mari). În Figura 6.2-2 am ilustrat două exemple de discretizare a aceluiași atribut.

Figura 6.2-2. Discretizarea în funcție de numărul grupurilor (Discretize by Binning)

Pasul 1:
Conectăm datele și operatorii conform imaginii alăturate (a se vedea și procesul).



Pasul 2:
Folosim operatorul „Discretize by Binning” de două ori pentru a ilustra diferențele relativ la opțiunile parametrilor. Indicăm faptul că dorim 3 grupuri (5 în celălalt caz). Dorim ca numele claselor să fie de tip interval. Intervalul de variație (range) al atributului PercentSalaryHike este 14 (max-min, adică 25-11). Împărțim 14 la 3 și obținem 4.7, deci primul grup va fi format din cazurile care i-au valori între 11 și 15.7, următorul 15.7 și 20.3 etc.



Rezultat:
Denumirile grupurilor au forma de interval.

Id	PercentSalaryHike_Bins5	PercentSalaryHike_Bins3	PercentSalaryHike
1	[-∞ - 13.8]	[-∞ - 15.7]	11
2	[22.2 - ∞]	[20.3 - ∞]	23
3	[13.8 - 16.6]	[-∞ - 15.7]	15
4	[-∞ - 13.8]	[-∞ - 15.7]	11

Numărul de cazuri alocat fiecărui grup diferă mult (grupul asociat valorilor mici conține semnificativ mai multe cazuri). Intervalele asociate grupurilor au lungimi identice în fiecare caz: 4.6, respectiv 2.8. Aceste tabele apar în perspectiva Results, tabul Statistics.

Index	Nominal value	Absolute count	Fraction
1	[-∞ - 15.7]	919	0.625
2	[15.7 - 20.3]	380	0.259
3	[20.3 - ∞]	171	0.116

Index	Nominal value	Absolute count	Fraction
1	[-∞ - 13.8]	617	0.420
2	[13.8 - 16.6]	380	0.259
3	[16.6 - 19.4]	247	0.168
4	[19.4 - 22.2]	159	0.108
5	[22.2 - ∞]	67	0.046

Discretizarea în funcție de frecvență (Discretize by Frequency)[41]

Discretizarea în funcție de frecvență pare să fie același lucru cu discretizarea în funcție de mărime (numărul cazurilor). Ele produc soluții similare doar în condiții speciale. În cazul „Discretize by Size” alegem numărul de cazuri pe care dorim să-l aibă fiecare din grupurile rezultate iar softul calculează numărul de grupuri. În cazul „Discretize by Frequency” este exact invers: stabilim numărul de grupuri[42] iar softul determină câte cazuri va conține fiecare. Algoritmul distribuie cazurile la grupuri în așa fel încât fiecare grup să conțină aproximativ același număr de cazuri. De exemplu, dacă dorim să grupăm valorile unui atribut în trei grupuri, prima treime dintre cazuri, cele cu valorile cele mai mici, vor fi alocate primului grup, a doua treime grupului secund, iar ultima treime (cazurile cu valorile cele mai mari) grupului cu numărul 3. Dacă atributul în cauză are una sau mai multe valori care sunt luate de un număr disproporționat de mare / mic de cazuri, grupurile rezultate vor avea mai multe / puține cazuri decât numărul de cazuri așteptat (numărul de cazuri nu va fi egal între grupuri). Lungimea intervalelor de variație asociate grupurilor poate diferi.

În Figura 6.2-3 am prezentat două exemple de discretizare de acest tip. În primul exemplu, operatorul încearcă să aloce valorile în mod egal la fiecare dintre cele trei grupuri, așteptarea fiind ca fiecare grup să aibă aproximativ 490 cazuri. Alocarea echilibrată nu este posibilă deoarece

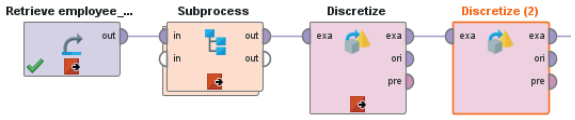
[41] Numită în literatura de specialitate „Equal Frequency Binning”.

[42] Alternativ, putem cere ca numărul de grupuri să fie determinat automat în funcție de numărul de cazuri din setul de date (numărul de grupuri va fi egal cu radical de ordinul doi din numărul de cazuri).

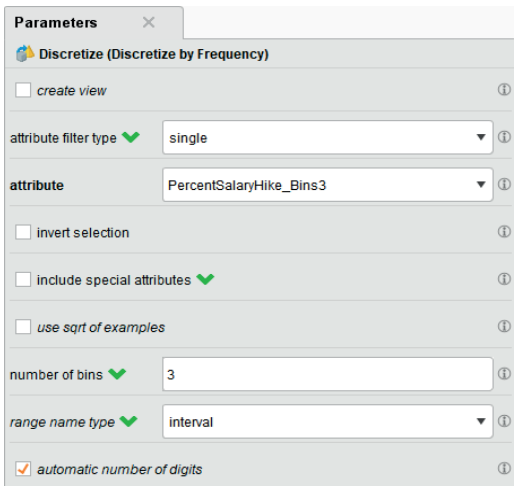
numărul de valori luate de atributul PercentSalaryHike este foarte mic raportat la numărul total de cazuri (mulți angajați au beneficiat de aceeași creștere procentuală relativ mai mică a salariului, respectiv mai puțini angajați au primit o creștere procentuală mare).

Figura 6.2-3. Discretizare în funcție de frecvență (Discretize by Frequency)

Pasul 1:
Conectăm datele și operatorii conform imaginii alăturate (a se vedea și procesul).



Pasul 2:
Folosim operatorul „Discretize by Frequency” de două ori pentru a ilustra diferențele relativ la opțiunile parametrilor. Indicăm faptul că dorim 3 grupuri (5 în celălalt caz), iar numele claselor să apară ca interval.
Operatorul va încerca să producă 3 (5) grupuri, fiecare cu aproximativ același număr de cazuri. În cazul de față grupurile rezultate vor avea cel mai probabil un număr diferit de cazuri.
Alternativ, numărul de grupuri poate fi calculat de soft în funcție de numărul de cazuri din setul de date (opțiunea „use sqrt of examples”).



Rezultat:
Denumirile grupurilor au forma de interval.

Id	PercentSalaryHike_Bins5	PercentSalaryHike_Bins3	PercentSalaryHike
1	[-∞ - 12.5]	[-∞ - 13.5]	11
2	[19.5 - ∞]	[16.5 - ∞]	23
3	[13.5 - 15.5]	[13.5 - 16.5]	15
4	[-∞ - 12.5]	[-∞ - 13.5]	11

Numărul de cazuri alocat la fiecare grup diferă deoarece setul de date conține foarte multe cazuri cu exact aceleași valori.
Observăm că intervalele au lungimi diferite.
Aceste tabele apar în perspectiva Results, tabul Statistics.

Index	Nominal value	Absolute count	Fraction
1	[-∞ - 13.5]	617	0.420
2	[16.5 - ∞]	473	0.322
3	[13.5 - 16.5]	380	0.259

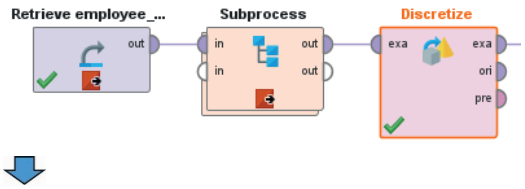
Index	Nominal value	Absolute count	Fraction
1	[-∞ - 12.5]	408	0.278
2	[15.5 - 19.5]	325	0.221
3	[13.5 - 15.5]	302	0.205
4	[19.5 - ∞]	226	0.154
5	[12.5 - 13.5]	209	0.142

Discretizare în funcție de preferințele utilizatorului (Discretize by User Specification)

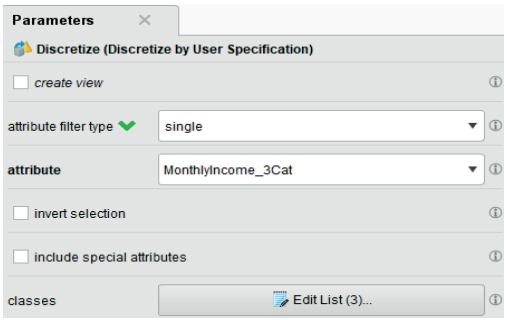
În cazul acestei metode de discretizare, utilizatorul trebuie să specifice atât numărul de grupuri dorite cât și intervalele de valori asociate acestora (Figura 6.2-4). În fapt, operatorul realizează ceea ce în limbajul softurilor de analiză statistică se numește recodarea valorilor.

Figura 6.2-4. Discretizare în funcție de preferințe (Discretize by User Specification)

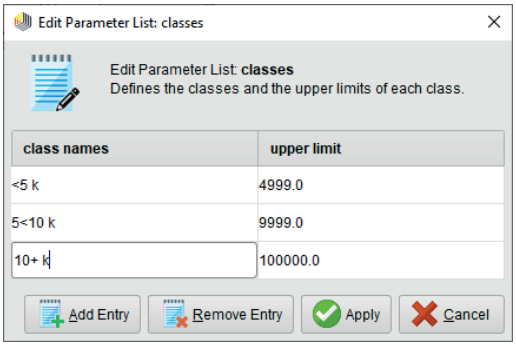
Pasul 1:
Conectăm datele și operatorii conform imaginii alăturate (a se vedea și procesul).



Pasul 2:
La operatorul „Discretize by User Specification” alegem atributul pe care dorim să-l discretizăm. Pot fi alese mai multe atribute simultan. Acest lucru este indicat doar dacă atributele au aceleași scale, altfel recodările propuse nu au sens pentru toate atributele.



La parametrul classes indicăm numele claselor și limita superioară a intervalului de variație asociat clasei respective.



Rezultat:
Observăm că fiecare valoare inițială a fost alocată grupului corespunzător.

Id	MonthlyIncome_3Cat	MonthlyIncome
1	5<10 k	5993
2	5<10 k	5130
3	<5 k	2090

Atributul rezultat are trei clase, fiecare cu un număr variabil de cazuri.

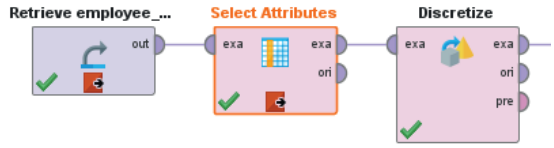
Ind...	Nominal value	Absolute count	Fraction
1	<5 k	749	0.510
2	5<10 k	440	0.299
3	10+ k	281	0.191

Discretizare în funcție de entropie (Discretize by Entropy)

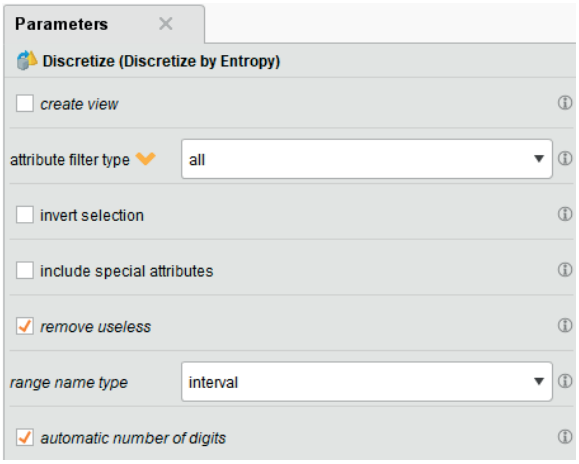
În cazul acestui operator limitele grupurilor sunt stabilite automat astfel încât entropia (dezordinea) asociată fiecărui grup să fie minimă. Entropia este calculată în relație cu un atribut de tip label, prin urmare este obligatoriu ca setul de date să includă un astfel de atribut. Atributele care nu contribuie la predicție pot fi eliminate, caz în care nu vor mai apărea în setul final. Un scurt exemplu este prezentat în Figura 6.2-5.

Figura 6.2-5. Discretizare în funcție de entropie (Discretize by Entropy)

Pasul 1:
Conectăm datele și operatorii conform imaginii alăturate (a se vedea și procesul).



Pasul 2:
Operatorul „Discretize by Entropy” stabilește automat numărul de grupuri alegând soluția care reduce cel mai mult entropia. Pentru a calcula entropia este necesar ca setul de date să includă un atribut de tip label. Reducerea entropiei va fi calculată în relație cu acest atribut.
În acest exemplu am inclus mai multe atribute numerice. Deoarece am marcat parametrul „remove useless”, atributele inutile pentru predicție vor fi eliminate.



Rezultat:
Denumirile grupurilor au forma de interval.
Atributele care au rămas și au fost discretizate sunt MontlyIncome, YearsAtCompany, YearsInCurrentRole, YearsWithCurrManager.
În acest exemplu, numărul de grupuri stabilit automat este 2, indiferent de atribut.
Numărul de cazuri alocat fiecărui grup variază de la un atribut la altul.

Id	Attrition	MonthlyIncome	YearsAtCompany	YearsInCurrentRole	YearsWithCurrManager
1	Yes	[2800.0 - ∞]	[2.0 - ∞]	[2.0 - ∞]	[0.0 - ∞]
2	No	[2800.0 - ∞]	[2.0 - ∞]	[2.0 - ∞]	[0.0 - ∞]
3	Yes	[-∞ - 2800.0]	[-∞ - 2.0]	[-∞ - 2.0]	[-∞ - 0.0]
4	No	[2800.0 - ∞]	[2.0 - ∞]	[2.0 - ∞]	[-∞ - 0.0]
5	No	[2800.0 - ∞]	[-∞ - 2.0]	[-∞ - 2.0]	[0.0 - ∞]

MontlyIncome

Index	Nominal value	Absolute count	Fraction
1	[2800.0 - ∞]	1135	0.772
2	[-∞ - 2800.0]	335	0.228

YearsAtCompany

Index	Nominal value	Absolute count	Fraction
1	[2.0 - ∞]	1128	0.767
2	[-∞ - 2.0]	342	0.233

6.3. Valorile lipsă (Missing)

Conceptul de valoare lipsă (missing value) este unul central în domeniul colectării și analizei datelor. Dat fiind faptul că tratarea acestei teme în literatura de specialitate în limba română este foarte redusă, îi vom acorda un spațiu relativ mai mare comparativ cu restul temelor.[43]

Prin valoare lipsă ne referim la situația în care informația relativ la intersecția dintre un atribut și un caz lipsește (Buuren, 2018, p. 3; Chisholm, 2013, p. 77). De exemplu, în setul de date disponibil, nu apare care este venitul lunar net al angajatului Ion Ion. La extreme, putem avea situații în care pentru unul sau mai multe cazuri lipsesc (aproape) toate valorile, indiferent de atribut, respectiv situații în care pentru unul sau mai multe atribute lipsesc (aproape) toate valorile, indiferent de caz. Desigur, situația comună este cea în care valorile lipsă apar rar, pentru o mică parte a cazurilor și atributelor (Tabelul 6.3-1).

[43] De obicei tema este redusă la descrierea succintă (maximum o pagină) a modului în care definim valorile lipsă în cadrul programului de analiză statistică SPSS (Howitt & Cramer, 2010, pp. 126–131; Opariuc-Dan, 2009, p. 49; Sava, 2011, p. 86; Vasile, 2014, pp. 79–83).

Tabelul 6.3-1. Date complete vs. date cu valori lipsă

Date complete

Id	Venit	Cheltuieli	Angajați
1	1000	1000	10
2	500	400	3
3	400	300	5
4	600	700	8
5	1500	1000	12

Valori lipsă: atribut (integral)

Id	Venit	Cheltuieli	Angajați
1	1000		10
2	500		3
3	400		5
4	600		8
5	1500		12

Valori lipsă: caz (integral)

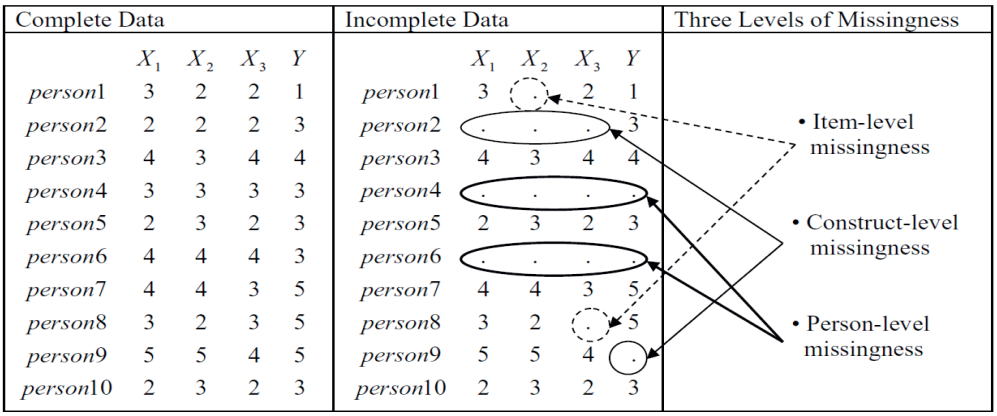
Id	Venit	Cheltuieli	Angajați
1	1000	1000	10
2	500	400	3
3	.	.	.
4	600	700	8
5	1500	1000	12

Valori lipsă: atribute și cazuri (parțial)

Id	Venit	Cheltuieli	Angajați
1	1000	1000	10
2	500	400	3
3	400	.	5
4	600	700	.
5	.	1000	12

O altă clasificare a tipurilor de valori lipsă (Tabelul 6.3-2), relativ similară, distinge între valori lipsă la nivel de item (atribut independent / obișnuit), la nivel de construct (atribut dependent / label), respectiv la nivel de persoană (respondent / caz statistic / exemplu) (Newman, 2014).

Tabelul 6.3-2. Date complete vs. incomplete și tipuri de valori lipsă



Sursa: (Newman, 2014)

De ce apar valorile lipsă?

Datele lipsă pot avea diferite cauze (Tabelul 6.3-3). De exemplu, în cazul datelor de anchetă, cauzele dominante sunt refuzul subiecților de a răspunde la unele întrebări sau chiar la toate, respectiv lipsa informației. Secundar, alte

cauze au legătură cu schema de eșantionare, selecția itemilor (o parte dintre itemi sunt aplicați doar unor sub-grupuri, pentru a reduce durate de aplicare a chestionarului), refuzul participării la una dintre componentele cercetării (subiectul răspunde la chestionarul principal dar nu și la cel opțional), refuzul participării la valurile ulterioare ale unei cercetări de tip panel, întrebările filtru (de exemplu, dacă subiectul nu este șomer, nu va răspunde la întrebările despre ajutorul de șomaj).

În cazul datelor instituționale apar aceleași cauze, ponderea lor fiind diferită. Cel mai adesea angajații răspund la întrebări sau datele relativ la situația acestora sunt preluate din documentele oficiale, dar e posibil ca, în timp, categoriile de informații care sunt colectate să varieze (de exemplu, la început compania a colectat doar date despre pozițiile ocupate de angajat în companie și beneficiile salariale primite, și doar în ultimii ani a început să colecteze și date relativ la satisfacție, recompensele non-financiare, participarea la cursuri de specializare etc.).

Tabelul 6.3-3. O tipologie a cauzelor apariției datelor lipsă

Nonresponse	Intentional	Unintentional
Unit nonresponse	Sampling	Refusal Self-selection
Item nonresponse	Matrix sampling Branching	Skip question Coding error

Sursa: Buuren, 2018, p. 34

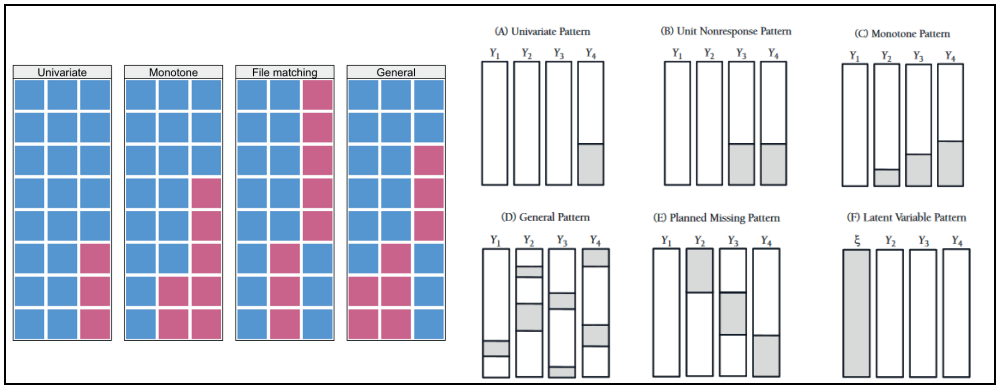
Paternuri de valori lipsă

Dincolo de cauze, datele lipsă dintr-un set de date pot forma diferite paternuri. E important să identificăm paternul deoarece acesta conține indicii cu privire la cauzele apariției valorilor lipsă, respectiv ne direcționează spre tipul adecvat de tratare a valorilor lipsă. Exemplele prezentate în literatură (Figura 6.3-1) disting între paternuri de tip:

- **Univariat**: datele lipsesc în cazul unui atribut, pentru o parte a cazurilor;
- **Monoton**: datele lipsesc pentru câteva atribute și o parte a cazurilor; dacă sortăm atributele și cazurile în funcție de numărul valorilor lipsă observăm că rezultă un patern monoton crescător;

- **General**: datele lipsesc pentru câteva atribute și o parte a cazurilor, fără să avem un patern evident;
- **Potrivire** (file matching): se referă la situația în care unim două seturi de date iar cazurile dintre aceste nu se suprapun în totalitate; unele cazuri vor avea date doar pentru o parte a atributelor;
- **Planificat**: produse intenționat, prin designul cercetării, cel mai adesea cu scopul de a reduce cantitatea de date colectată de la un sub-eșantion (de obicei aleator) de respondenți.

Figura 6.3-1. Paternuri ale valorilor lipsă



Sursa: stânga (Buuren, 2018, p. 106), *dreapta* (Enders, 2010, p. 4)*; roșu / gri = valori lipsă.*

Tipuri de valori lipsă: MCAR, MAR, MNAR

La prima vedere, toate valorile lipsă par să fie la fel. În realitate, ele pot fi foarte diferite, mecanismul de producere a lor poate fi diferit, implicațiile pentru analiză și estimare a statisticilor de interes fiind majore. Cea mai cunoscută și utilizată clasificare a valorilor lipsă distinge între următoarele trei mecanisme (Buuren, 2018, pp. 8–9; Enders, 2010, pp. 6–8; Little & Rubin, 2019, pp. 13–23):

- **MCAR**: missing completely at random; probabilitatea apariției valorilor lipsă nu depinde de niciun alt atribut;
- **MAR**: missing at random; probabilitatea apariției valorilor lipsă depinde de una unul sau mai multe atribute, cu excepția atributului în cauză (cel cu valori lipsă);

- **MNAR**: missing not at random; probabilitatea apariției valorilor lipsă depinde de atributul în cauză (cel cu valori lipsă), mai exact de valoarea lipsă în sine.

Pentru a înțelege mai bine aceste tipuri vom considera un exemplu simplu. Să presupunem că dorim să estimăm venitul mediu al angajaților dintr-o organizație. În situația în care alegem să nu punem întrebarea despre venit unei sub-eșantion aleator al eșantionului investigat, valorile lipsă la venit vor fi de tip MCAR (decizia a fost una externă, a cercetătorului, deci prezența / absența unui răspuns nu depinde de caracteristicile respondenților). Dacă rata de răspuns la întrebarea despre venit este mai mică în cazul bărbaților comparativ cu femeile, valorile lipsă vor fi de tip MAR (prezența / absența răspunsului depinde de gen, dar nu depinde de venit / valoarea acestuia). Dacă respondenții cu venituri relativ mai mari au o rată de răspuns mai mică la întrebarea despre venit, valorile lipsă vor fi de tip MNAR (prezența / absența răspunsului depinde de venit / valoarea acestuia).

În Tabelul 6.3-4 am ilustrat diferența dintre cele trei tipuri de valori lipsă (mecanisme de producere a valorilor lipsă) alături de impactul asupra estimărilor. Să presupunem că avem o populație (sau eșantion) de 8 indivizi. Dorim să aflăm care este venitul mediu al acestei populații, prin urmare încercăm să colectăm date de la toți indivizii (sau un eșantion al acestora). În fiecare situație avem două persoane care nu ne răspund la întrebarea despre venit. În cazul MCAR, probabilitatea ca o persoană să nu declare venitul nu depinde de mediul de rezidență (una dintre persoanele care nu răspund e din urban, cealaltă din rural) și nici de mărimea venitului (ambele au un venit apropiat de medie). În cazul MAR, probabilitatea ca o persoană să nu declare venitul depinde de mediul de rezidență (ambele sunt din rural), dar nu depinde de mărimea venitului. În cazul MNAR, probabilitatea ca o persoană să nu declare venitul depinde de mărimea acestuia (cele două persoane care nu declară venitul au un venit mai mare decât media). Observăm că estimările produse de cei doi indicatori statistici (media și suma veniturilor) sunt destul de diferite de valorile reale, respectiv diferă în funcție de tipul

mecanismul responsabil pentru producerea valorilor lipsă. Cele mai mari erori apar în cazul MNAR.

Tabelul 6.3-4. O ilustrare a tipurilor de valori lipsă și impactului acestora

Id	Populație	MCAR	MAR	MNAR	Mediu
1	1000	1000	1000	.	rural
2	500	500	500	500	rural
3	800	.	.	800	rural
4	700	700	.	600	rural
5	1000	.	1000	1000	urban
6	1200	1200	1200	1200	urban
7	1500	1500	1500	.	urban
8	800	800	800	800	urban
Medie	**938**	**950**	**1000**	**817**	
Sumă	**7500**	**5700**	**6000**	**4900**	
Diferență		**MCAR**	**MAR**	**MNAR**	
Medie		**12**	**62**	**-121**	
Sumă		**-1800**	**-1500**	**-2600**	

Să considerăm un alt exemplu, de această dată mai apropiat de domeniul resurselor umane (Enders, 2010, p. 7). Să presupunem că organizația X a realizat o selecție de personal în baza scorului la un test de inteligență. Un an mai târziu, pentru fiecare nou angajat a fost măsurată performanța în muncă. Datele obținute sunt prezentate în Tabelul 5.1-1, coloana Complete. Observăm că datele sunt sortate ascendent în funcție de atributul IQ. În practică, adesea, datele disponibile sunt incomplete. Celelalte coloane prezintă astfel de posibile situații reale (date incomplete) pentru fiecare dintre cele trei scenarii (MCAR, MAR, MNAR). În fiecare dintre aceste situații, informația cu privire la performanța în muncă lipsește în cazul a cinci angajați. În cazul MCAR, evaluarea performanței lipsește la întâmplare (în sensul că nu depinde de niciunul dintre scorurile la cele două teste, inteligență și performanță în muncă). În cazul MAR, evaluarea performanței lipsește mai degrabă în cazul angajaților cu scoruri mici la testul de inteligență, deci depinde de IQ. În cazul MNAR, evaluarea performanței lipsește mai degrabă în cazul angajaților cu scoruri mici la performanță (și IQ sub medie). În acest caz, probabilitatea ca evaluarea performanței să lipsească depinde de scorul primit la evaluare (chiar și atunci când controlăm scorul la testul de inteligență).

Tabelul 6.3-5. Selecția personalului: scorul la testul de inteligență și performanța în muncă

	Job performance ratings			
IQ	Complete	MCAR	MAR	MNAR
78	9	—	—	9
84	13	13	—	13
84	10	—	—	10
85	8	8	—	—
87	7	7	—	—
91	7	7	7	—
92	9	9	9	9
94	9	9	9	9
94	11	11	11	11
96	7	—	7	—
99	7	7	7	—
105	10	10	10	10
105	11	11	11	11
106	15	15	15	15
108	10	10	10	10
112	10	—	10	10
113	12	12	12	12
115	14	14	14	14
118	16	16	16	16
134	12	—	12	12

Sursa: Enders, 2010, p. 7

Ce putem face atunci când avem valori lipsă?

Să presupunem că am făcut tot ceea ce depinde de noi pentru a colecta date cât mai complete. Cu toate acestea, o parte din date lipsesc. Ce putem face în acest caz?[44] Răspunsul depinde foarte mult de situația concretă în care ne aflăm, mai exact de mecanismul care a produs datele lipsă (MCAR, MAR, MNAR), tipul de date (metrice sau non-metrice, secționale sau longitudinale / serii de timp) și tipuri de măsuri statistice pe care dorim să le calculăm (medie, mediană, coeficient de regresie, sumă etc.).

La modul general, avem la dispoziție trei direcții generale de acțiune: (1) eliminarea valorilor lipsă, sau, mai exact, eliminarea din analiză a cazurilor și/sau atributelor cu valori lipsă, (2) estimarea directă a parametrilor de interes[45] și (3) imputarea valorilor lipsă (înlocuirea lor cu valorile relativ mai

[44] Pentru o discuție teoretică cu privire la tratarea valorilor lipsă în cazul bazelor de date mari, se poate consulta capitolul „Handling missing data in large databases” (Spiess & Augustin, 2021), din volumul „Handbook of Computational Social Science, Volume 2, Data Science, Statistical Modelling, and Machine Learning Methods”. Autorii menționați prezintă două soluții, ponderarea și imputarea multiplă, argumentând că, în cazul bazelor de date mari, folosirea imputării multiple optimizată în raport cu calitatea predicției poate duce la inferențe invalide.

[45] În acest caz valorile lipsă nu sunt înlocuite și nici nu sunt eliminate cazurile, respectiv atributele cu valori lipsă. Nu discutăm aici această categorie, doar menționăm cele două variante: Full Information Maximum Likelihood (FIML) și Bayesian Estimation (folosind un algoritm Markov Chain Monte Carlo, de exemplu Metropolis-Hastings).

probabile, în baza unor asumpții). Pentru situațiile MCAR și MAR putem utiliza toate strategiile, iar pentru MNAR sunt preferate imputarea și FIML, respectiv Bayesian Estimation.

Eliminarea valorilor lipsă se poate face în unul dintre următoarele moduri:

- **Eliminarea cazurilor**: eliminăm din setul de date (analiză) cazurile cu valori lipsă prin una dintre metodele:
  - **Listwise**: numită și analiza cazurilor complete (complet case analysis); elimină toate cazurile care au cel puțin o valoare lipsă la cel puțin unul dintre atributele incluse în analiză / de interes; de exemplu, dacă dorim că calculăm matricea de corelații a trei atribute, toți coeficienții de corelație vor fi calculați pe același număr de cazuri, cele care nu au nicio valoarea lipsă la niciunul dintre cele trei atribute;
  - **Pairwise**: numită și analiza cazurilor disponibile (available case analysis); elimină din analiză un număr variabil de cazuri, funcție de perechea de atribute analizată; de exemplu, dacă dorim că calculăm matricea de corelații a trei atribute, fiecare coeficient de corelație va fi calculat pe un număr posibil diferit de cazuri, numărul respectiv fiind egal cu cel mai mic număr de cazuri observat în cazul unuia dintre cele două atribute care compun perechea respectivă;
- **Eliminarea atributelor**: eliminăm din setul de date (analiză) toate atributele care au o pondere a valorilor lipsă mai mare decât un anumit prag.

Eliminarea cazurilor și/sau a atributelor e ușor de implementat. În plus, „pairwise deletion” are avantajul că ține cont de toată informația disponibilă. Principalele dezavantaje constau în faptul că se pierde o parte a informației și că estimările obținute pot fi distorsionate (mai ales în cazul MCAR). În general, metodele din această categorie nu sunt de preferat (la nivel general, soluția optimă e imputarea).

Imputarea sau înlocuirea valorilor lipsă se poate face folosind o multitudine de metode (Buuren, 2018, pp. 9–18; Enders, 2010, pp. 37–55). Mai mult, putem înlocui o valoare lipsă cu o altă valoare (imputare unică) sau cu mai multe

valori selectate aleator dintr-o distribuție statistică estimată sau dintre valorile unui set de cazuri similare (imputare multiplă). În Tabelul 6.3-6 am prezentat doar câteva metode de imputare cu valori unice și o scurtă descriere, pe câteva dimensiuni relevante. Imputarea multiplă[46] depășește cadrele acestui manual și nici nu este disponibilă în RapidMiner în acest moment, deci nu o vom discuta aici, deși este metoda de preferat în comparație cu imputarea unică și așa zisele metode ad-hoc (Buuren, 2018, p. 20) sau tradiționale (Enders, 2010, pp. 37–55).

Tabelul 6.3-6. Metode de imputare unică a valorilor lipsă (selecție)

Metoda	Înlocuire cu …	Avantaje	Dezavantaje
Măsură a tendinței centrale	media / mediana / valoarea modală	- ușor de implementat;	- distorsionează distribuția; - reduce varianța; - distorsionează estimările;
Regresie deterministă	predicțiile produse de un model de regresie	- ușor de implementat; - folosește toată informația disponibilă;	- distorsionează corelațiile dintre atribute; - sub-estimează variabilitatea; - supra-estimează intensitatea relațiilor dintre atribute; - poate produce predicții neplauzibile
Regresie stocastică	predicțiile produse de un model de regresie + o valoare reziduală	- ușor de implementat; - folosește toată informația disponibilă;	- distorsionează corelațiile dintre atribute; - sub-estimează variabilitatea; - supra-estimează intensitatea relațiilor dintre atribute; - poate produce predicții neplauzibile
Predictive mean matching (PMM)	valorile observate în cazul vecinilor (cazuri similare cu cazul pentru care dorim să înlocuim valorile lipsă)	- ușor de implementat; - potrivită pentru date categoriale sau numerice; - robustă; - distribuția finală va fi similară cu distribuția inițială;	- e necesar un eșantion mare; - probleme dacă distribuțiile sunt asimetrice (skewed) sau dispersate (sparse); - nu poate extrapola dincolo de intervalul de variație inițial al atributului;

[46] La nivel multivariat, funcție de paternul valorilor lipsă, se poate folosi una dintre strategiile de imputare multiplă: Monotone, „Joint Modelling”, „Multivariate Imputation by Chained Equations” (MICE) / „Fully Conditional Specification” (FCS) (Buuren, 2018).

Impactul asupra estimărilor

Alegerile pe care le facem cu privire la tratarea valorilor lipsă influențează calitatea estimărilor statistice obținute. În cele ce urmează vom prezenta câteva exemple care ilustrează câteva probleme tipice.

Să considerăm un exemplu foarte simplu cu privire la utilizarea cazurilor complete versus a cazurilor disponibile pentru a estima media și suma unor atribute (Tabelul 6.3-7). Observăm că în ambele situații estimările obținute sunt diferite de valorile reale. În plus, erorile sunt mult mai mari în cazul sumei.

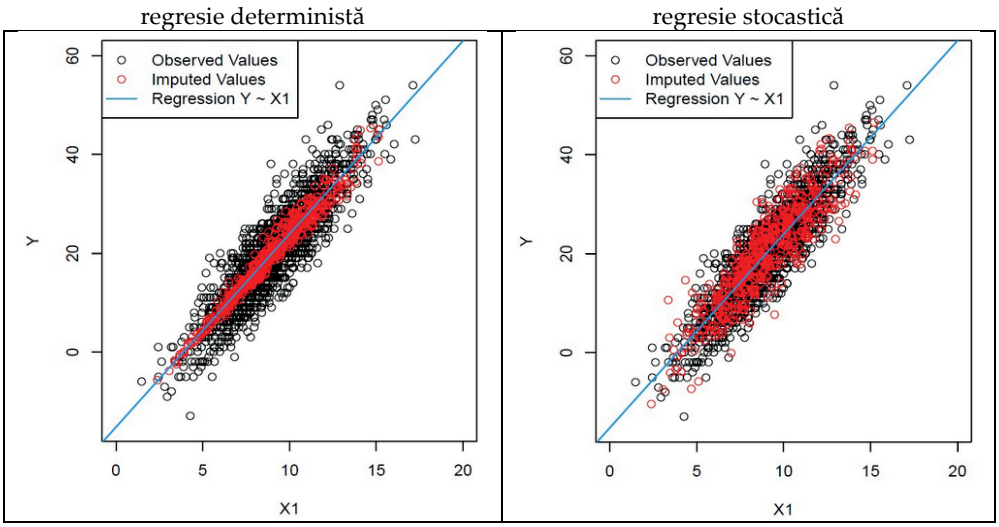
Tabelul 6.3-7. Listwise vs. Pairwise (cazuri complete vs. disponibile)

Id	Venituri	Cheltuieli	Angajați
1	1000	1000	10
2	500	400	3
~~3~~	~~400~~	~~.~~	~~5~~
~~4~~	~~600~~	~~700~~	~~.~~
~~5~~	~~.~~	~~1000~~	~~12~~
Medie	750	700	6.5
Sumă	1500	1400	13
Populație			
Medie	660	680	7.4
Sumă	3300	3400	37
Diferență			
Medie	+90	+20	-0.9
Sumă	-1800	-2000	-24

Id	Venituri	Cheltuieli	Angajați
1	1000	1000	10
2	500	400	3
3	400	~~.~~	5
4	600	700	~~.~~
5	~~.~~	1000	12
Medie	625	775	7.5
Sumă	2500	3100	30
Populație			
Medie	660	680	7.4
Sumă	3300	3400	37
Diferență			
Medie	-35	+95	+0.1
Sumă	-800	-300	-7

Dacă alegem să comparăm două tipuri de imputare bazate pe regresie, deterministă vs. stocastică (Figura 6.3-2), observăm că distribuțiile finale rezultate sunt diferite. Astfel, în cazul regresiei stocastice, valorile imputate se abat mult mai mult de la dreapta de regresie (ca urmare a termenului eroare adăugat). Foarte probabil, valorile obținute cu ajutorul regresiei stocastice sunt relativ mai apropiate de valorile „reale” (deoarece relațiile dintre variabile sunt rareori perfecte).

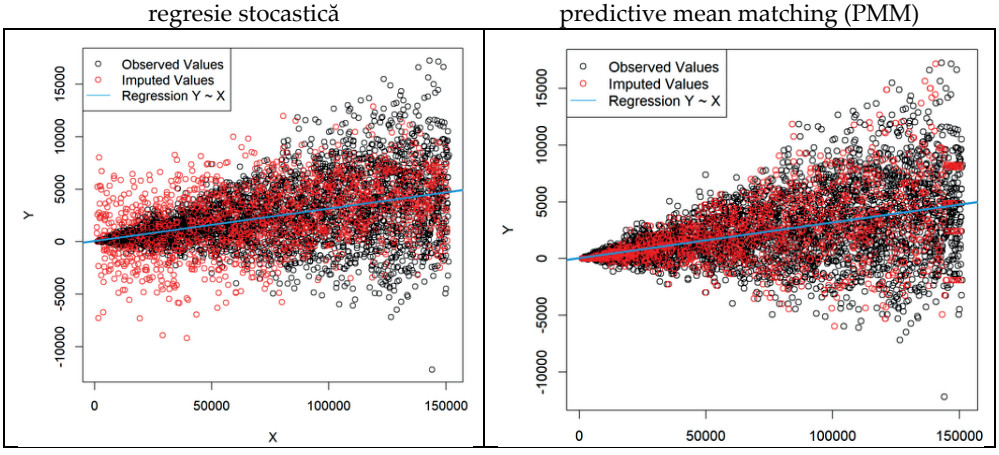
Figura 6.3-2. Date imputate: regresie deterministă vs. stocastică



Sursa: https://statisticsglobe.com/regression-imputation-stochastic-vs-deterministic/

Să considerăm că avem două atribute Y și X și ne interesează să imputăm valorile lipsă în cazul lui Y. Dacă relația dintre cele două atribute nu respectă asumpția de heteroscedasticitate (necesară în cazul regresiei), în urma aplicării unui model de imputare bazat pe regresie (stocastică aici) rezultă o distribuție destul de diferită a datelor. În schimb, imputarea cu ajutorul PMM (Predictive Mean Matching) nu schimbă forma distribuției.

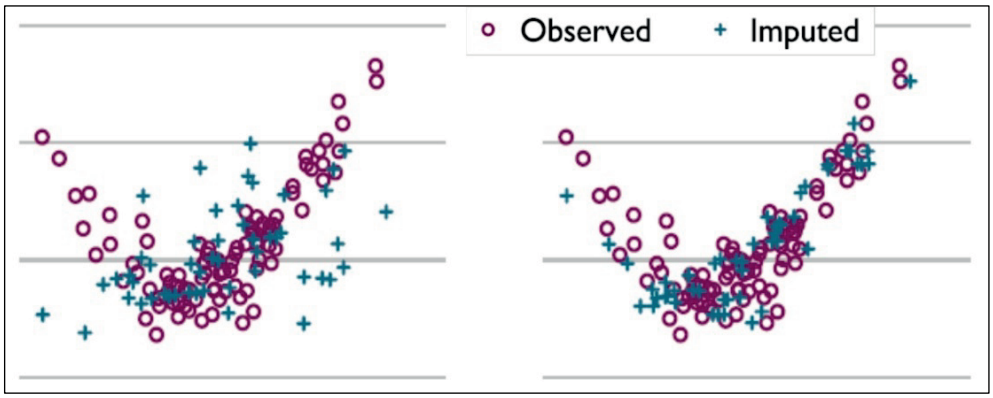
Figura 6.3-3. Date imputate (heteroscedasticitate): regresie stocastică vs. PMM



Sursa: https://statisticsglobe.com/predictive-mean-matching-imputation-method/

În cazul în care relația dintre cele două atribute este non-liniară, valorile imputate folosind regresia liniară nu urmează același patern, schimbă forma relației. Nu același lucru se întâmplă dacă folosim PMM.

Figura 6.3-4. Date imputate (non-liniaritate): regresia liniară vs. PMM



Sursa: https://stefvanbuuren.name/fimd/

Valorile lipsă în RapidMiner Studio

În RapidMiner Studio, caracterul folosit pentru a indica lipsa unei valori este „?”. Acest caracter este folosit indiferent de tipul de atribut (numeric, categorial). După cum se poate observa în Tabelul 6.3-8, numărul de situații cu valori lipsă poate varia foarte mult de la un caz la altul sau de la un atribut la altul. De exemplu, atributul class nu are nicio valoare lipsă, atributul col-adj are toate valorile lipsă (relativ la cele afișate), iar restul atributelor au un număr variabil de valori lipsă. Raportat la cazuri, avem cazuri cu toate valorile lipsă (exceptând atributul special class), cazuri cu valori complete, respectiv situații intermediare. La perspectiva Results, tabul Statistics, coloana Missing este indicat numărul de cazuri lipsă relativ la fiecare atribut.

Tabelul 6.3-8. Exemple de valori lipsă într-un set de date RapidMiner

class	duration	wage-inc-1st	wage-inc-2nd	wage-inc-3rd	col-adj
good	1	5	?	?	?
good	2	4.500	5.800	?	?
good	?	?	?	?	?

class	duration	wage-inc-1st	wage-inc-2nd	wage-inc-3rd	col-adj
bad	3	2	2.500	2.100	tc

class	duration ↓	wage-inc-1st	wage-inc-2nd	wage-inc-3rd	col-adj
good	3	3.700	4	5	tc
good	3	4.500	4.500	5	?
good	3	4	5	5	tc

Name	Type	Missing
Label class	Nominal	0
duration	Integer	1
wage-inc-1st	Real	1
wage-inc-2nd	Real	10
wage-inc-3rd	Real	28
col-adj	Nominal	16

** selecții din setul de date labor-negociations*

Operatorii care ne ajută să lucrăm cu valorile lipsă sunt grupați în folderul Missing. Funcție de obiectivul urmărit, putem folosi unul sau mai mulți dintre operatorii:

- **Replace Missing Values**: înlocuirea valorilor lipsă; ne ajută să înlocuim valorile lipsă folosind metode de predicție simple (de exemplu, înlocuirea lor cu valoarea medie / modală); fiecare valoare lipsă este înlocuită cu o singură valoare prezisă, deci nu luăm în calcul incertitudinea asociată predicției;
- **Impute Missing Values**: imputarea valorilor lipsă; ne ajută să înlocuim valorile lipsă folosind un model de predicție complex (de exemplu, valorile lipsă sunt înlocuite cu valorile prezise de un model bazat pe arbori decizionali); fiecare valoare lipsă este înlocuită cu o serie de valori prezise, deci luăm în calcul incertitudinea asociată predicției; în RapidMiner fiecare valoare lipsă este înlocuită cu o singură valoare prezisă, deci ignorăm incertitudinea asociată predicției;
- **Declare Missing Values**: declararea valorilor lipsă; este folosit pentru a declara că anumite valori sunt valori lipsă, a le defini ca valori lipsă;
- **Replace Infinite Values**: înlocuirea valorilor infinite; îl folosim pentru a înlocui valorile infinite;

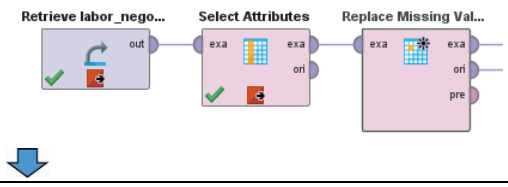
- **Remove Unused Values**: eliminarea valorilor neutilizate; îl folosim pentru a elimina valorile neutilizate;
- **Fill Data Gaps**: completarea golurilor în cazul unui atribut numeric de tip id;
- **Replace All Missings**: înlocuirea tuturor valorilor lipsă; ne ajută să înlocuim toate valorile lipsă;
- **Handle Unknown Values**: gestionarea valorilor necunoscute; îl folosim pentru a comunica softului modalitatea de gestionare a valorilor necunoscute.

Înlocuirea valorilor lipsă (Replace Missing Values)

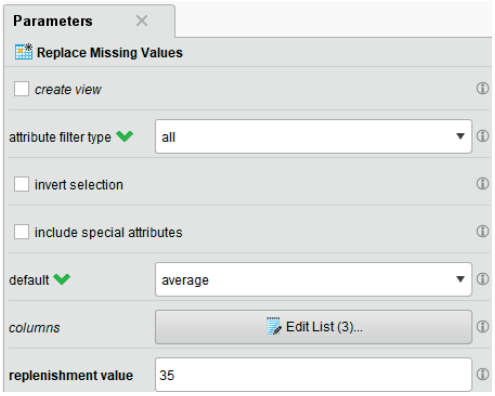
Acest operator implementează cele mai simple modalități de înlocuire a valorilor lipsă (Figura 6.3-5). Valorile lipsă ale unui atribut de tip metric pot fi înlocuite cu valoarea minimă, maximă, medie, zero sau orice altă valoare. În cazul unui atribut nominal, putem înlocui valorile lipsă cu valoarea modală (varianta de răspuns cu cea mai mare frecvență) sau cu altă valoare. Operatorul „Replace Missing Values” are două caracteristici majore: (1) înlocuiește o valoare lipsă cu o singură altă valoare și (2) pentru a realiza înlocuirea folosește informația asociată strict acelui atribut (nu ține cont de relația dintre acest atribut și altul) sau furnizată de utilizator (în baza unei decizii personale bazată sau nu pe informații suplimentare; de exemplu, în cazul angajaților cu o vechime mai mică de un an, valoarea lipsă la atributul „creștere salarială” poate fi înlocuită cu zero).

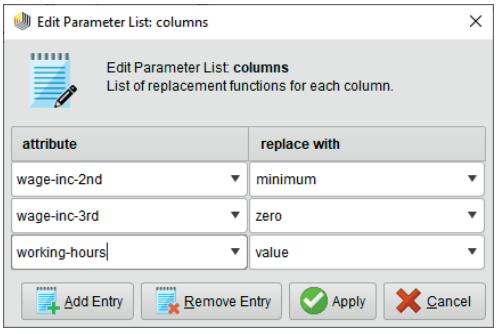
Figura 6.3-5. Înlocuirea valorilor lipsă (Replace Missing Values)

Pasul 1:
Conectăm datele și operatorii conform imaginii alăturate (a se vedea și procesul).



Pasul 2:
Indicăm faptul că dorim să înlocuim valorile lipsă în cazul tuturor atributelor. Dacă nu apar alte specificații, valorile lipsă vor fi înlocuite cu media / valoarea modală (average). Softul știe dacă un atribut este de tip numeric sau nominal, deci va înlocui valorile lipsă cu media în cazul atributelor numerice, respectiv cu valoarea modală în cazul celor nominale.
În cazul unor atribute, ne dorim să facem înlocuirea cu alte valori decât media. Aici am înlocuit valorile lipsă în cazul atributului wage-inc-2nd cu valoarea minimă, la wage-inc-3rd cu zero, iar la working-hours cu „value”, unde value este 35 (replenishment value).





Rezultat:
Compararea celor două seturi de date arată faptul că valorile lipsă (?) au fost înlocuite conform specificațiilor.
De exemplu, valoarea lipsă de la cazul 4 pentru atributul vacation a fost înlocuită cu below-average (cele mai multe cazuri aveau această valoare).

Row No.	class	wage-inc-1st	wage-inc-2nd	wage-inc-3rd	working-h...	vacation
1	good	5	?	?	40	average
2	good	4.500	5.800	?	35	below-average
3	good	?	?	?	38	generous
4	good	3.700	4	5	?	?

Row No.	class	wage-inc-1st	wage-inc-2nd	wage-inc-3rd	working-h...	vacation
1	good	5	2	0	40	average
2	good	4.500	5.800	0	35	below-average
3	good	3.621	2	0	38	generous
4	good	3.700	4	5	35	below-average

Imputarea valorilor lipsă (Impute Missing Values)

Operatorul „Impute Missing Values” are două caracteristici majore: (1) înlocuiește o valoare lipsă cu o singură altă valoare[47] și (2) pentru a realiza imputarea folosește un model de predicție în care intră toate atributele incluse de utilizator. Modelul de predicție este bazat pe un clasificator ales de utilizator. Clasificatorul trebuie plasat în interiorul operatorului „Impute Missing Values”. Procesul va imputa toate valorile lipsă doar dacă clasificatorul ales poate gestiona astfel de valori.

Operatorul are o serie de opțiuni care determină soluția obținută (Figura 6.3-6):

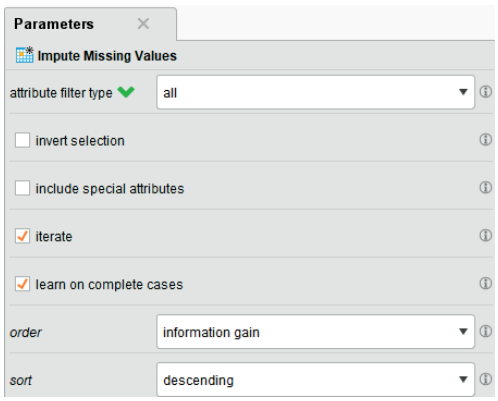
- Putem alege ce atribute să includem în model (parametrul „attribute filter type”), inclusiv dacă să includem sau nu atributele speciale (parametrul „include special attributes”).
- Procesul de predicție poate fi iterativ sau nu. Dacă parametrul este marcat, prima dată sunt imputate valorile unui atribut (în baza modelului de predicție asociat acelui atribut), apoi se trece la următorul atribut etc.
- Predicția se poate face folosind doar cazurile complete (fără valori lipsă) sau nu. Dacă nu bifăm acest parametru, procesul va rula doar dacă clasificatorul are capacitatea de a gestiona valorile lipsă.
- Putem stabili ordinea în care sunt imputate atributele (parametrii order și sort). Criteriul folosit pentru ordonare poate fi: cronologic, aleator, numărul valorilor lipsă și câștigul informațional (information gain). Sortarea atributelor în funcție de criteriul ales poate fi ascendentă sau descendentă.

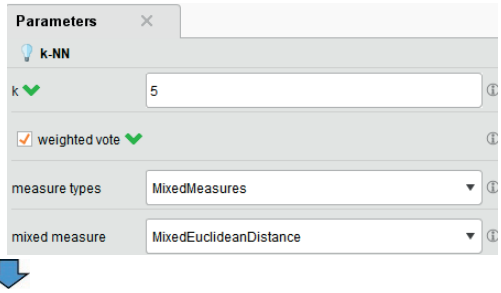
[47] Tehnic, termenul de imputare se referă în literatura de specialitate la situația în care, în baza unui model de predicție, fiecare dintre valorile lipsă este înlocuită cu mai multe valori (o serie de valori). Astfel, fiecare caz cu valori lipsă va fi multiplicat de numărul indicat de ori, iar valoarea lipsă va fi înlocuită pe rând cu una dintre valorile prezise.

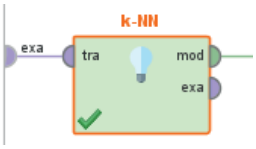
Figura 6.3-6. Imputarea valorilor lipsă (Impute Missing Values)

Pasul 1:
Conectăm datele și operatorii conform imaginii alăturate (a se vedea și procesul).

Pasul 2:
Indicăm faptul că dorim să imputăm valorile lipsă în cazul tuturor atributelor.
Parametrii iterate și „learn on complete cases” sunt marcați implicit.
La order am ales „information gain”, iar la sort ascending.
Pentru a „afla” valorile lipsă, trebuie să includem în proces un clasificator. Aici am folosit k-NN. Pentru a-l include am intrat în interiorul operatorului „Impute Missing Values” cu dublu click și am conectat operatorul k-NN (cu opțiunile implicite).







Rezultat:
Observăm că valorile lipsă (?) au fost înlocuite. La atributul vacation, cazul 4, valoarea imputată este below-average. La atributul wage-inc-1st, cazul 3, valoarea imputată este 0, la working-hours, cazul 4, valoarea imputată este 36.6 etc.

Row No.	class	wage-inc-1st	wage-inc-2nd	wage-inc-3rd	working-h...	vacation
1	good	5	?	?	40	average
2	good	4.500	5.800	?	35	below-average
3	good	?	?	?	38	generous
4	good	3.700	4	5	?	?

Row No.	class	wage-inc-1st	wage-inc-2nd	wage-inc-3rd	working-hou...	vacation
1	good	5	0	3.291	40	average
2	good	4.500	5.800	3.396	35	below-average
3	good	0	0	3.330	38	generous
4	good	3.700	4	5	36.575	below-average

Declararea valorilor lipsă (Declare Missing Values)

Oarecum contra-intuitiv[48] având în vedere denumirea lui, acest operator înlocuiește valorile indicate cu valori lipsă (?). Să presupunem că avem un set de date pe care dorim tocmai l-am importat în RapidMiner; știm că în cazul unui anumit atribut numeric valoarea 99 înseamnă valoare lipsă, prin urmare vrem să o înlocuim cu „?"; similar, în cazul unui atribut de tip nominal, valoarea de răspuns „nu știu" poate fi înlocuită cu „?". Parametrii operatorului permit alegerea și indicarea atributului / atributelor de interes, tipul de valoare (nominală, numerică sau expresie) și valoarea în cauză.

Înlocuirea valorilor infinite (Replace Infinite Values)

Acest operator înlocuiește valorile de tip infinit (negativ sau pozitiv) cu valorile specificate de utilizator. Putem alege dintre variantele: none, zero, max_byte, max_int, max_double și missing. În cazul opțiunilor none, zero și missing valorile infinite sunt înlocuite cu nimic, 0, respectiv nan (not a number). În cazul celor cu max în nume, valorile infinite pozitive sunt înlocuite cu valoarea maximă care nu este infinit pozitiv, iar valorile infinite negative sunt înlocuite cu valoarea minimă care nu este infinit negativ.

Eliminarea valorilor neutilizate (Remove Unused Values)

Cu referire la atributele de tip nominal, acest operator va elimina valorile care nu sunt asociate unui caz măcar.

Umplerea golurilor (Fill Data Gaps)

Dacă setul de date conține o variabilă de tip id de tip numeric (valori întregi) care acoperă un interval dar cu goluri, operatorul va include în setul de date cazurile cu id-urile lipsă. De exemplu, dacă setul de date conține cazurile cu id-urile 1-4 și 6-10, aplicarea acestui operator va include în setul de date un caz nou care va avea valoarea 5 la id și ? la restul atributelor.

[48] Ne-am fi așteptat să păstreze valorile originale, dar să le trateze ca și cum ar fi valori lipsă.

Înlocuirea tuturor valorilor lipsă (Replace All Missings)

Așa cum sugerează și numele, operatorul înlocuiește automat toate valorile lipsă dintr-un set de date. În cazul atributelor nominale valorile lipsă sunt înlocuite cu valoarea specificată de utilizator. În cazul atributelor numerice, valorile lipsă sau infinite sunt înlocuite cu media (cu 0 dacă nu apare nicio valoare). Este mai simplu de aplicat dar este relativ mai puțin flexibil și util.

Gestionarea valorilor necunoscute (Handle Unknown Values)

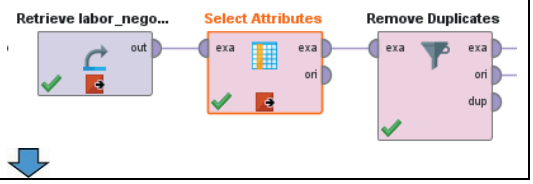
Operatorul identifică toate valorile asociate atributelor de tip nominal și le salvează într-un model care poate fi utilizat pentru pre-procesarea altor seturi de date. Astfel ne asigurăm că, atunci când prelucrăm un nou set de date, vom utiliza doar valorile nominale (categoriile de răspuns) identificate anterior. Dacă un set nou conține și alte valori, acest operator le va transforma în valori lipsă. Operatorul nu schimbă setul de date de input.

6.4. Cazurile identice (Duplicates)

Această categorie conține un singur operator și anume „Remove Duplicates”. Dacă un set de date conține mai multe cazuri identice, operatorul va păstra în setul de date doar un caz din fiecare set de cazuri identice. Utilizatorul poate alege care sunt atributele în funcție de care se stabilește dacă două cazuri sunt identice sau nu. Simplu spus, dacă două cazuri iau exact aceleași valori în cazul atributelor indicate, ele sunt considerate identice și unul dintre ele va fi eliminat. Atributele în funcție de care este realizată comparația pot fi de orice tip. În Figura 6.4-1 am ilustrat aplicarea acestui operator în două situații, o data considerând toate atributele, apoi doar atributele cu valori numerice.

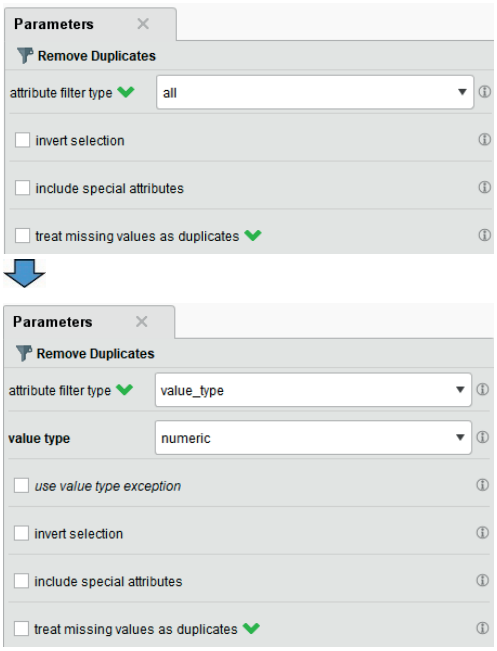
Figura 6.4-1. Eliminarea cazurilor identice (Remove Duplicates)

Pasul 1:
Conectăm datele și operatorii conform imaginii alăturate (a se vedea și procesul).



Pasul 2 (all):
Dorim să eliminăm cazurile identice luând în considerare toate atributele obișnuite din setul de date. Putem ține cont și de atributele speciale dacă bifăm parametrul „include special attributes”. Setul rezultat conține doar 36 cazuri (față de 40 în setul inițial).

Pasul 2 (value_type):
Dorim să eliminăm cazurile care sunt identice luând în considerare toate atributele obișnuite cu valori numerice. Putem ține cont și de atributele speciale numerice dacă bifăm parametrul „include special attributes”. Observăm că setul rezultat conține doar 33 cazuri (față de 40 în setul inițial). Dacă bifăm parametrul „treat missing values as duplicates”, operatorul va considera că două cazuri care au valori lipsă sunt identice, deci va elimina unul din ele.



6.5. Cazurile neobișnuite (Outliers)

Discuțiile cu privire la outlieri și tratarea lor în literatura de specialitate de limbă română este foarte redusă. Pentru acest motiv, alături de importanța temei, considerăm că merită o atenție specială comparativ cu restul temelor.[49]

Ce sunt outlierii?

Termenul de outlier este folosit pentru a caracteriza un caz dintr-un set de date sau o valoare luată de acel caz relativ la un anumit atribut. Adesea, conceptul de outlier este definit ca un caz deviant, o valoare extremă:

> *„An outlier is a deviant case – it is an extreme numeric value in a distribution” (De Vaus, 2002, p. 92).*

[49] Și în acest caz discuțiile sunt reduse (maximum o pagină), limitându-se la prezentarea unor exemple simple (Popa, 2008, pp. 61–63; Sava, 2011, p. 87) sau la menționarea modului în care putem defini și trata outlierii în cadrul programului de analiză statistică SPSS (Millea, 2018, p. 162; Vasile, 2014, p. 121).

> *„Data values that are unusually large or small compared to the other values of the same construct” (Aguinis et al., 2013).*
> *„Outliers are data points that are extremely distant from most of the other data points” (Leys et al., 2019, p. 1).*

Astfel de definiții par să limiteze conceptul de outlier la o singură serie de valori, la o singură măsură / variabilă, deci la nivel univariat. Însă, este posibil să observăm outlieri și la nivel bivariat sau multivariat. Un astfel de outlier este cel mai adesea un caz care are o combinație neobișnuită a valorilor în cazul ambelor (mai multor) variabile (De Vaus, 2002, p. 92). Prin urmare, o definiție completă a conceptului de outlier ar trebui să includă și această posibilitate:

> *„An outlier is a case with such an extreme value on one variable (a univariate outlier) or such a strange combination of scores on two or more variables (multivariate outlier) that it distorts statistics” (Tabachnick & Fidell, 2019, p. 62).*
> *„An object in a data set is usually called an outlier if (1) it deviates from the normal/known behavior of the data, (2) it assumes values that are far away from the expected/average values, or (3) it is not connected/similar to any other object in terms of its characteristics” (Ranga Suri et al., 2019, p. 3).*

Unele definiții semnalează suplimentar faptul că outlierul distorsionează valorile unor măsuri statistice univariate (media) sau multivariate (coeficienții de corelație și regresie):

> *„An outlier ... can have an undue influence on some statistics, especially parametric ones. Outliers can be a problem when using summary statistics to describe a distribution or a relationship between variables” (De Vaus, 2002, p. 92).*
> *„An outlier ... distorts statistics” (Tabachnick & Fidell, 2019, p. 62).*

Conform altei definiții, mai generală și totodată mai scurtă, outlierii sunt:

> *„Data points with large residual values” (Leys et al., 2019).*

Prin valoare reziduală înțelegem diferența dintre valoarea observată și valoarea prezisă de un model statistic. O astfel de definiție este de preferat deoarece nu restrânge numărul de dimensiuni în relație cu care definim un outlier și nici nu impune un anumit model / o anumită măsură statistică (Leys et al., 2019).

În concluzie, prin eticheta de outlier semnalăm că o valoare dintr-o serie de valori, respectiv o combinație de valori asociată unui caz, este foarte diferită / extremă / aberantă / anormală / neobișnuită / ciudată / o anomalie. Probabil, termenul care definește cel mai cuprinzător conceptul de outlier este „neobișnuit” deoarece nu sugerează o valorizare și acoperă toate formele concrete sub care apar outlierii. Traducerea prin sintagma valoare extremă, folosită cel mai adesea, induce ideea de valoare foarte mare / mică raportat la restul valorilor dintr-o serie. Sintagma acoperă multe dintre situațiile comune de outlieri, dar nu pe toate, fiind posibil să avem outlieri care apar la centrul unei distribuții în formă de U sau doar atunci când considerăm simultan două sau mai multe atribute (raportat la fiecare atribut în parte, cazul / valoarea nu este un outlier).

„Neobișnuit” poate lua forme diverse în lumea reală. Neobișnuită poate fi o valoare care este mult mai mică / mare decât celelalte valori din setul de date sau poate fi o valoare luată de foarte puține cazuri. Caracterul neobișnuit se poate manifesta în relație cu un singur atribut, cu două sau mai multe atribute. În consecință, lumea outlierilor este una diversă, după cum se poate observa și din Figura 6.5-1:

- **panel a:** numărul de produse vândute în data de 26 martie 2019 sunt mult mai mari comparativ cu celelalte zile, deci ziua respectivă este un outlier;
- **panel b:** numărul de persoane cu numele Jane este semnificativ mai mare, deci numele respectiv este un outlier (probabil în cazul persoanelor fără nume a fost trecut Jane);
- **panel c:** având în vedere relația observată dintre înălțime și greutate, persoana respectivă are o greutate mai mică pentru înălțimea pe care o are, deci este un outlier;
- **panel d:** tensiunea sanguină în stare de repaus a persoanelor din partea de sus a graficului este mai mare de 120, deci, din punctul de vedere al doctorului, valorile respective sunt outlieri;
- **panel e:** comparativ cu restul cazurilor, cele două cazuri marcate cu roșu iau valori mai mari în cazul ambelor atribute, deci sunt outlieri;
- **panel f:** deși nu ia valori extreme la niciun atribut, dimpotrivă, cazul din centru este foarte diferit de restul cazurilor, deci este outlier;

- **panel g:** cazurile poziționate în exteriorul elipsei sunt outlieri pentru că iau valori diferite de restul cazurilor în funcție de cel puțin unul dintre atribute;
- **panel h:** angajații cu id 2 și 19 sunt outlieri deoarece scorul lor la test nu pare să fie la fel de conectat cu performanța (id 2 are o performanță mai mare decât ne-am aștepta în baza scorului la test, iar id 19 mai mică).

Figura 6.5-1. Exemple de outlieri

Dacă am cere unor copii să indice cazurile neobișnuite din aceste imagini, foarte probabil majoritatea le-ar identifica fără probleme. Dincolo de această subiectivitate împărtășită, este necesar să definim matematic ce condiții trebuie să îndeplinească un caz sau o valoare pentru a fi considerat(ă) outlier. Simplu spus, trebuie să avem o formulă de calcul a distanțelor dintre cazuri / valori, respectiv un „prag” în funcție de care să convenim că un caz / o valoare este sau nu outlier. Dincolo de formalizarea matematică, o doză mare de subiectivitate rămâne cel mai adesea, atât la nivelul alegerii modului în care este calculată distanța (formula), cât și la nivelul stabilirii pragului (de exemplu, relativ la un atribut numeric, de ce o valoare, pentru a fi considerată outlier, trebuie să fie mai mare/mică cu trei abateri standard decât media și nu cu patru sau chiar mai multe).

De ce apar outlierii și de ce e util să-i identificăm?

O investigare a outlierilor este importantă pentru cel puțin două motive: (1) outlierii în sine pot fi de interes pentru analist și (2) outlierii distorsionează rezultatele analizelor statistice (univariate și multivariate); de exemplu, la nivel multivariat, outlierii pot împiedica găsirea unor relații semnificative statistic sau, dimpotrivă, pot contribui la identificarea unor relații care în realitate nu există.

Outlierii conțin adesea informații importante cu privire la fenomenul analizat, respectiv la procesul prin care datele au fost colectate și înregistrate. Înainte de a decide ce facem cu acele valori / cazuri (cum le „tratăm”), e necesar să înțelegem care e sursa lor.

Outlierii apar ca urmare a erorilor asociate procesului de producere a datelor, dar pot corespunde și unor situații reale. O clasificare relativ mai detaliată a cauzelor apariției acestora este prezentată în Tabelul 6.5-1. Pentru a realiza această clasificare am folosit două criterii majore: sursa (naturală / nenaturală) și intenționalitatea (da / nu).

Tabelul 6.5-1. O clasificare a cauzelor apariției outlierilor

Nenaturale + Neintenționate	Nenaturale + Intenționate	Naturale
– erori de design (designul cercetării); – erori de măsurare (instrumentele); – erori de colectare (procesul efectiv de colectare a datelor); – erori de introducere a datelor (asociate unor persoane / unui soft); – erori de procesare a datelor (manipulare, transformare etc.); – erori de eșantionare (cadrul de eșantionare, selecția eșantionului, identificarea cazurilor);	– valori introduse intenționat de analist în setul de date cu scopul de a testa și compara algoritmii de detecție a outlierilor; – valori declarate intenționat greșit de către subiecți;	Nu sunt erori. Valorile corespund situației reale

Dacă datele utilizate conțin erori, rezultatele analizelor statistice vor fi aproape întotdeauna distorsionate. Prezența outlierilor poate indica uneori faptul că datele au o calitate redusă. Prin urmare este necesar să analizăm outlierii pentru a ne asigura că datele corespund cu realitatea (valorile sunt cele corecte). Valorile incorecte pot lua diferite forme precum:

- valori care țin locul valorilor lipsă, dar sunt tratate ca valori reale; de exemplu, codurile numerice 999/-999 care semnifică varianta de răspuns „nu răspund” pot fi ușor considerate valori corecte; dacă aceste valori nu aparțin intervalului de variație al atributului, sunt șanse mari să fie identificate ca outlieri; dacă nu, le identificăm ca outlieri doar dacă avem informații relativ la modul în care au fost codate răspunsurile;
- erori de înregistrare și/sau introducere: cifre care sunt dublate, valori înregistrate în unități de măsură diferite, date cu formate diferite etc.

Măsurile pe care le folosim pentru a descrie datele sunt afectate de prezența outlierilor. De exemplu, o valoare extrem de mare prezentă la atributul salariu modifică valoarea estimată a salariului mediu mult în sus, respectiv crește varianța atributului (inegalitatea estimată va fi mai mare). Mai mult, outlierii influențează chiar și măsurile folosite pentru identificarea lor (Leys

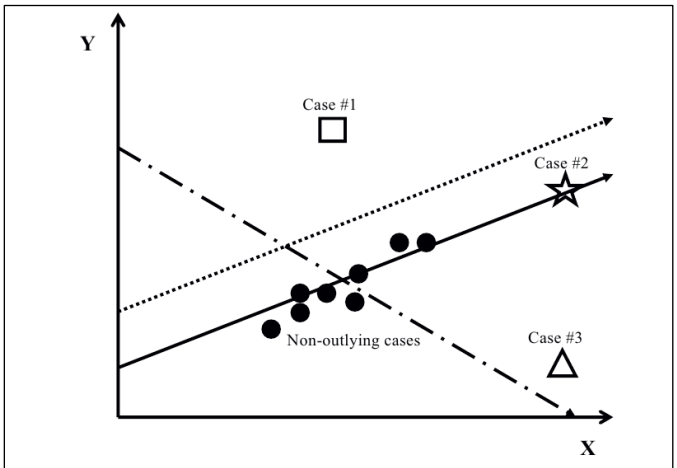
et al., 2019). O parte din procesul de analiză și interpretare a datelor presupune vizualizarea distribuțiilor sub forma unor grafice. Pentru că extind foarte mult axele graficelor, outlierii pot face mai dificil acest proces.

Uneori ne interesează mai degrabă cazurile atipice, nu cele tipice. De exemplu, poate fi de interes să analizăm situația angajaților cu performanță foarte mare, respectiv foarte mică. Uitându-ne la caracteristicile pe care le au în comun fiecare dintre aceste categorii de angajați putem înțelege mult mai bine care sunt factorii care determină performanța.

La nivel multivariat, funcție de situația concretă (număr de cazuri, tip de analiză, complexitatea modelului, tipul outlierilor), prezența unuia sau a mai multor outlieri poate distorsiona estimările coeficienților, semnificația asociată acestora și calitatea modelului. În următoarele două exemple ilustrăm la nivel bivariat, în contextul regresiei liniare, impactul outlierilor asupra variației „explicate”, mărimii coeficienților ecuației de regresie (constanta și panta) și incertitudinii asociate acestora (semnificația statistică).

În Figura 6.5-2 sunt ilustrate trei situații tipice de influență ale unui outlier. Cercurile pline reprezintă cazurile care nu sunt outlieri, iar cazurile 1-3 sunt trei tipuri de outlieri. Cazul 1 este outlier relativ la variabila dependentă (Y) și urmează un patern diferit de restul cazurilor (este situat la distanță mare de dreapta de regresie, deci eroarea de predicție în cazul acestuia este mare). Includerea în analiză a cazului 1 are ca efect scăderea variației explicate și creșterea interceptului / constantei (panta rămâne la fel). Cazul 2 este oarecum outlier relativ la ambele variabile, dar este situat pe dreapta de regresie originală. Includerea în analiză a cazului 2 are ca efect creșterea variației explicate (constanta și panta rămân la fel). Cazul 3 este oarecum outlier relativ la ambele variabile și este la distanță de dreapta de regresie originală (are o valoare reziduală mare). Includerea în analiză a cazului 3 are ca efect scăderea variației explicate și modificarea semnificativă a constantei și a pantei.

Figura 6.5-2. Impactul outlierilor asupra estimării dreptei de regresie (1)

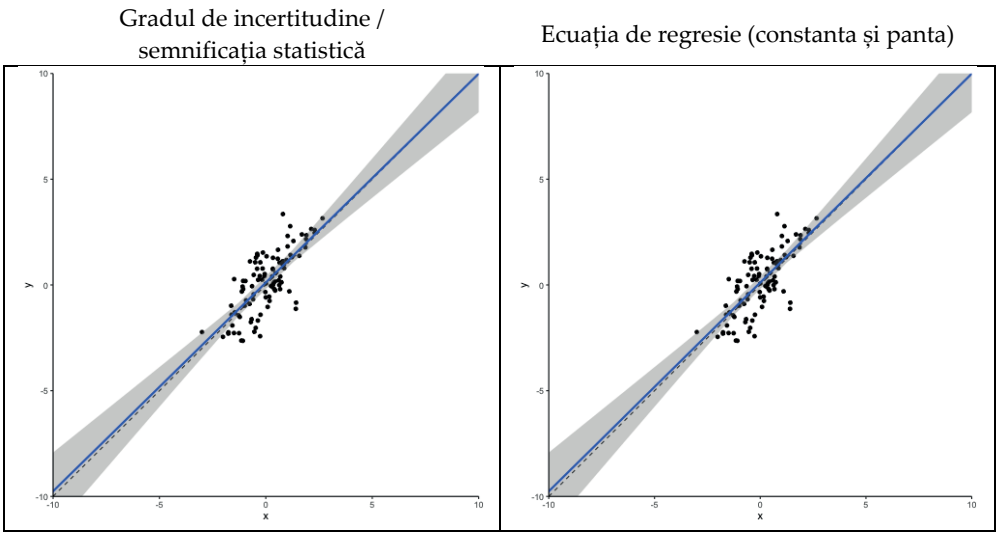


	Regression line	R^2	Slope	Intercept
With ● only	———	.73	.83	5.34
With ● and □	·····	.11	.83	9.13
With ● and ☆	———	.95	.83	5.34
With ● and △	— · — · —	.17	–.22	30.34

Sursa: (Aguinis et al., 2013)

În Figura 6.5-3 este ilustrat impactul unui singur outlier în două situații simple, oarecum extreme: în panelul din stânga, outlierul este relativ la variabila dependentă (y) și ia o valoare centrală în cazul variabilei independente (x); în panelul din dreapta, outlierul este relativ la variabila independentă (x) și ia o valoare centrală în cazul variabilei dependente (y). Se observă cum incertitudinea estimării (poziția estimată a dreptei de regresie) crește pe măsură ce outlierul ia valori tot mai mari în cazul atributului y (panelul din stânga), respectiv coeficienții estimați ai dreptei de regresie se schimbă (constanta crește iar panta scade) pe măsură ce outlierul ia valori tot mai mari în cazul atributului x (panelul din dreapta). Desigur, relativ la acest exemplu sunt posibile o mulțime de situații intermediare. În plus, există și alți factori care contribuie la mărimea și sensul distorsiunii. Astfel, mărimea și sensul distorsiunii depind de numărul total de cazuri, variabilele relativ la care apar outlierii, respectiv numărul și poziția outlierilor.

Figura 6.5-3. Impactul outlierilor asupra estimării dreptei de regresie (2)



Sursa: Gassen, Joachim. 2021. Taking Outlier Treatment to the Next Level. (https://arc.eaa-online.org/blog/taking-outlier-treatment-next-level)

Cum detectăm outlierii?

Pentru a identifica outlierii este necesar să folosim combinat trei surse de informații: (1) informații cu privire la procesul de colectare a datelor (cum au fost colectate datele, când, de către cine etc.); (2) informații cu privire la fenomenul studiat, respectiv atributele măsurate; de exemplu, relativ la fiecare atribut (combinații ale acestora), e util să știm care valori sunt legitime, tipice, neobișnuite, respectiv imposibile; (3) informațiile rezultate din analiza statistică a outlierilor. În general nu este o idee bună să excludem cazuri / valori doar în baza unor teste sau măsuri statistice.

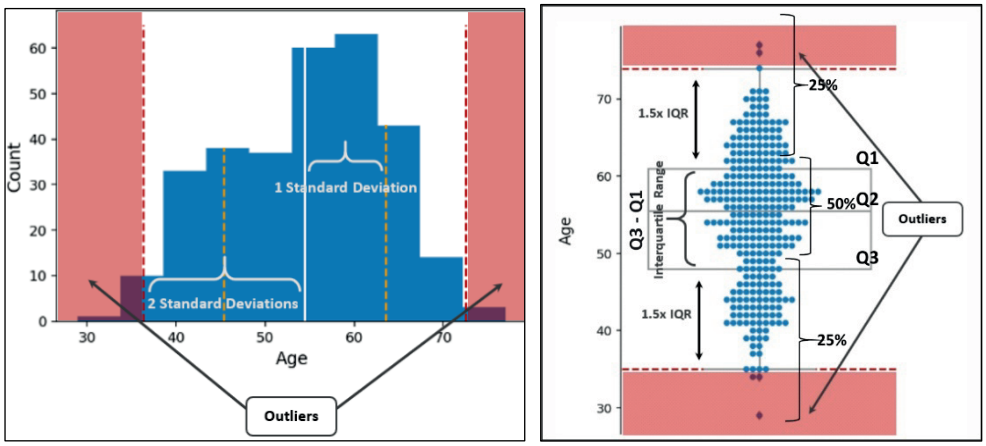
Nu există reguli statistice absolut sigure pentru a identifica outlierii. Unii analiști preferă metodele vizuale, alții diferite proceduri statistice. Vizualizarea cazurilor cu ajutorul unor grafice de tip boxplot, histogramă și scatter este suficientă cel mai adesea. Pentru a identifica outlierii ne putem uită la ...

- setul de date sortat în funcție de atributul de interes, respectiv ne uităm la primele și ultimele valori; orice valoare foarte diferită de valorile din proximitate este suspectă;
- tabele de frecvențe și selecții de cazuri în funcție de două sau mai multe atribute;

- grafice: box plot și histogramă la nivel univariat sau bivariat, scatter plot (cu sau fără linia de regresie) pentru nivel bivariat;
- analize statistice: relativ la analiza univariată, în general, o valoare este etichetată ca outlier dacă se abate cu mai mult de două sau trei abateri standard de la media valorilor (De Vaus, 2002, p. 94); valorile situate la o distanță de cel puțin trei ori IQR (interquartile range) sunt considerate outlieri; erorile de predicție mari asociate unui caz semnalează că acesta are o probabilitate mare să fie outlier.

De exemplu, în cazul unei variabile numerice, la nivel univariat, putem identifica outlierii uitându-ne la devierea relativ la valoarea medie exprimată în abateri standard (potrivită pentru distribuții normale), respectiv la devierea de la valoarea mediană exprimată în abateri intercuartilice (utilă mai ales în cazul distribuțiilor care se abat de la normalitate) (Figura 6.5-4). În aceste exemple, cazurile care se abat cu mai mult de două abateri standard de la medie, respectiv mai mult de trei IQR (abatere intercuartilică) de la mediană, sunt considerate outlieri.

Figura 6.5-4. O definire statistică a outlierilor la nivel univariat



Sursa: What is an Outlier?
(https://dataschool.com/fundamentals-of-analysis/what-is-an-outlier/)

Probabil o alternativă și mai corectă pentru detectarea outlierilor la nivel univariat este MAD (median absolute deviation) (Leys et al., 2013). Pentru a calcula MAD urmăm pașii: (1) calculăm mediana unei serii de valori, (2) calculăm diferențele dintre fiecare valoare a seriei și valoarea mediană calculată anterior, rezultând o nouă serie de valori, (3) calculăm mediana seriei

rezultate și (4) înmulțim noua mediană cu 1.4826. Valorile seriei originale care sunt mai mari de două sau trei ori decât MAD sunt considerate outlieri.

La nivel multivariat putem folosi măsuri clasice precum Mahalanobis Distance, MVE (Minimum Volume Ellipsoid), MCD (Minimum Covariance Determinant), MGV (Minimum Generalized Variance), PBO (Projection-Based Outliers) (Finch, 2012) sau, de preferat, unele robuste precum Mahalanobis-MCD (Leys et al., 2018).

„Tratarea” outlierilor

În general, metodele recomandate pentru „tratarea” outlierilor intră în una dintre următoarele trei categorii: (1) păstrarea outlierilor, (2) eliminarea outlierilor și (3) recodarea outlierilor (Aguinis et al., 2013).

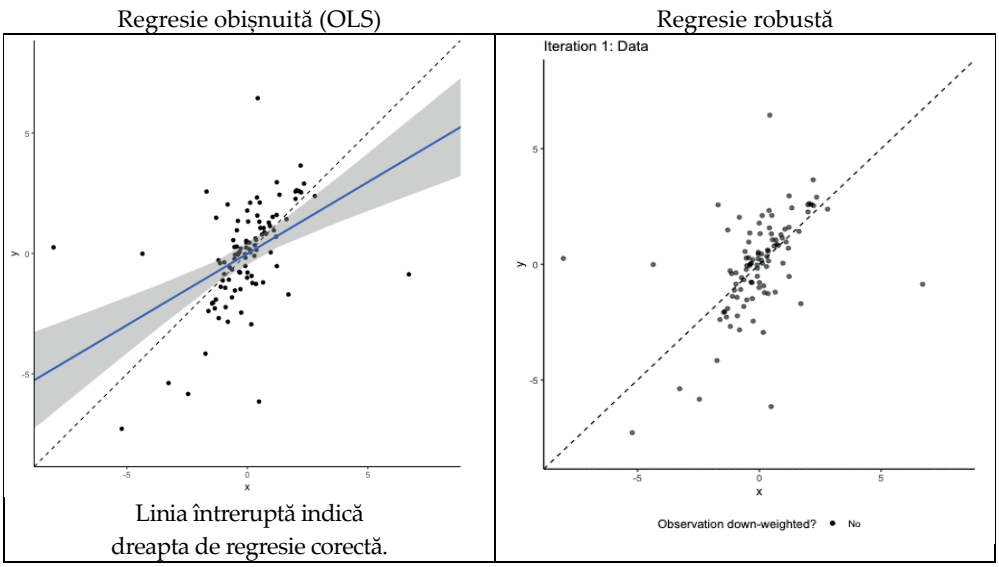
Dacă observăm că un caz din setul nostru de date are valori foarte diferite comparativ cu restul cazurilor (relativ la unul sau mai multe atribute), prima dată verificăm dacă acesta aparține populației investigate. Dacă nu aparține, cazul trebuie eliminat din setul de date. Dacă outlierii au apărut ca urmare a erorilor asociate procesului de măsurare putem încerca să corectăm acele erori și, implicit, valorile. Dacă acest lucru nu este posibil, o soluție poate fi eliminarea cazurilor respective sau doar a valorilor outlier din setul de date. Alternativ, putem înlocui (recoda) valorile outlier folosind una dintre următoarele metode: (1) înlocuim outlierii cu valoarea observată în cazul percentilei 1/5, respectiv 95/99, (2) transformăm distribuția folosind o funcție matematică (log, arcsin, radical, putere) și (3) tratăm outlieri ca valori lipsă și îi înlocuim prin imputare.

Alternativ la soluțiile prezentate anterior, putem păstra outlierii, dar să utilizăm analize statistice care sunt relativ puțin influențate de prezența acestora. Astfel putem folosi metode de tip nonparametric, măsuri și metode de analiză robuste[50], respectiv tehnici de bootstrapping. De exemplu, în locul regresiei simple putem folosi regresia „robustă”. Robustețea poate fi atinsă prin identificarea outlierilor folosind metoda reziduurilor, urmată de scăderea ponderărilor (weights) asociate cazurilor outlier (pentru fiecare caz outlier algoritmul calculează o valoare sub-unitară a ponderării; cazul va fi

[50] De exemplu mediana în locul mediei, respectiv regresia liniară robustă în locul regresiei obișnuite (OLS).

ponderat în jos – down-weighted) (Gassen & Veenman, 2022). În exemplul din Figura 6.5-5 observăm că există mai mulți outlieri, unii relativ la un singur atribut (x sau y), alții relativ la ambele atribute. Dreapta de regresie corectă este cea punctată, iar cea estimată de regresia OLS este cea albastră (panelul din stânga). Evident, cele două drepte de regresie diferă, diferența fiind rezultatul prezenței outlierilor. Dacă aplicăm algoritmul de ponderare în jos (sub-unitară) a outlierilor (panelul din dreapta), dreapta de regresie estimată se apropie, ca poziție și incertitudine a estimării, tot mai mult de dreapta de regresie reală.[51]

Figura 6.5-5. Ilustrarea funcționării unui algoritm de regresie robustă[52]



Sursa: Gassen, Joachim. 2021. Taking Outlier Treatment to the Next Level.
(https://arc.eaa-online.org/blog/taking-outlier-treatment-next-level

Procesul de „tratare” a outlierilor trebuie să răspundă cel puțin la următoarele întrebări: Cum am identificat outlierii? Care este cauza lor cea mai probabilă? Am eliminat cazurile respective și de ce? Am eliminat valorile outlier sau le-am înlocuit (cum?) și de ce? Dacă i-am păstrat în analiză, ce tehnici statistice am folosit (unele care produc estimări mai puțin dependente

[51] În acest caz relația dintre x și y este liniară. Dacă relația ar fi fost non-liniară, algoritmul ar fi produs o estimare distorsionată a poziției dreptei de regresie.

[52] M-estimator using a Huber loss function with a k value of 1.345 where the scale is being adjusted based on the median absolute deviation of the residuals (Gassen & Veenman, 2022).

de prezența outlierilor)? Deciziile relativ la definirea, identificarea și tratarea outlierilor pot avea implicații importante în sensul că pot schimba concluziile obținute (prezența sau absența unui efect, sensul și mărimea acestuia) (Aguinis et al., 2013). Pentru a preveni situațiile de tip p-hacking, se recomandă ca autorii să specifice, anterior implementării studiului, care este metoda ce va fi utilizată pentru detectarea outlierilor (criteriul inclusiv) și cum anume vor trata outlierii (Leys et al., 2019).

Suplimentar, este adesea util să realizăm analizele de interes pe setul de date complet, setul fără outlieri, respectiv setul cu valorile înlocuite, și să comparăm rezultatele obținute. Dacă rezultatele sunt similare, cel mai probabil outlierii nu au un impact, deci putem avea încredere în rezultatele obținute. Dacă apar diferențe, vom analiza atent sursa acestora încercând să identificăm legătura dintre deciziile metodologice luate și sensul diferențelor.

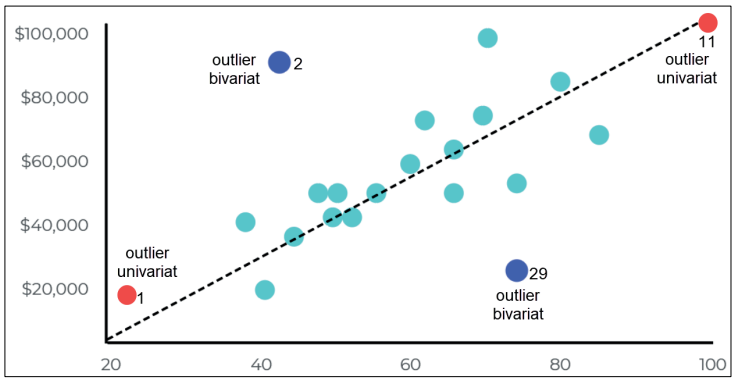
Clasificarea outlierilor

În funcție de numărul atributelor pe care le avem în vedere atunci când decidem dacă un caz este sau nu outlier, distingem între outlieri la nivel univariat (distribuția valorilor este funcție de un singur atribut), bivariat și multivariat. În Figura 6.5-6 avem doi outlieri la nivel univariat (cazurile 1 și 11, marcate cu roșu) și doi la nivel bivariat (cazurile 2 și 29, marcate cu albastru).

Relativ la atributul de pe axa orizontală, scorul la test, cazul 1 ia o valoarea semnificativ mai mică, iar cazul 11 o valoare semnificativ mai mare. Niciunul dintre cele două cazuri nu este outlier relativ la atributul de pe axa verticală (performanța, măsurată prin valoarea produselor vândute) deoarece mai există și alte cazuri care au aproximativ aceleași valori.

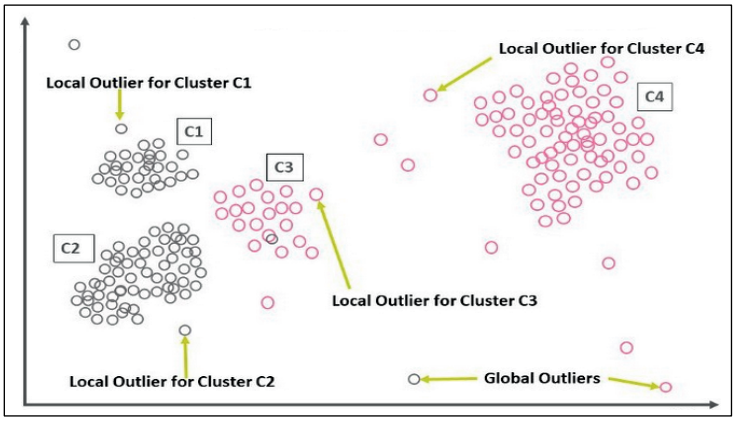
Cazurile 2 și 29 sunt outlieri la nivel bivariat deoarece au valori semnificativ diferite de restul cazurilor atunci când considerăm ambele atribute. După cum indică și dreapta de regresie, corelația dintre scorul la test și performanță este puternic pozitivă, dar cele două cazuri nu par să o urmeze (sunt la mare distanță de linia de regresie). Astfel, cazul 2 are o performanță mult mai mare decât ne-am fi așteptat conform scorului la test, iar cazul 29 mult mai mică.

Figura 6.5-6. Outlier univariat vs. bivariat



Atunci când decidem dacă un caz este sau nu outlier, putem să-l comparăm cu toate celelalte cazuri din setul de date, respectiv doar cu o parte a cazurilor, cele din proximitate. În primul caz vorbim de outlier global, iar în celălalt de outlier local. Un outlier global e foarte diferit de toate celelalte cazuri din setul de date, iar un outlier local e diferit doar relativ la cazurile din proximitate. Outlierul global e imediat vizibil, fără să fie nevoie să avem alte informații. Pentru a identifica un outlier local e nevoie să avem și alte informații. O ilustrare vizuală a diferenței dintre cele două tipuri de outlieri apare în Figura 6.5-7.

Figura 6.5-7. Outlier de tip global vs. local

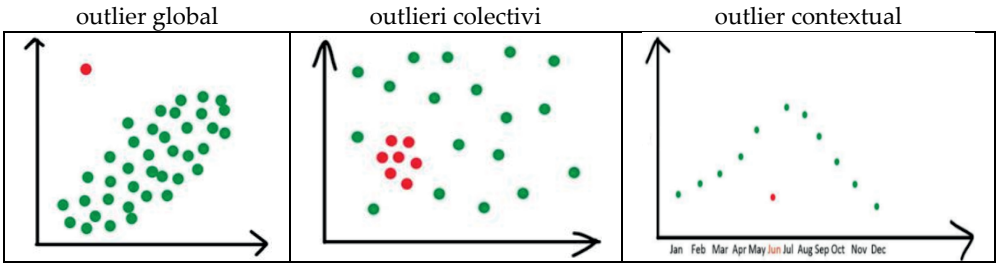


Sursa: Mahto, Paritosh (2020) Local Outlier Factor: A way to Detect Outliers.
https://medium.com/mlpoint/local-outlier-factor-a-way-to-detect-outliers-dde335d77e1a

O altă clasificare (sau mai degrabă tipologie) care apare relativ frecvent în literatură distinge între outlieri de tip global, contextual (condițional) și colectiv. În Figura 6.5-8 am ilustrat vizual aceste tipuri de outlier. Outlierul contextual ia valori apropiate de celelalte valori din serie, dar diferite relativ

la așteptări. În acest caz, temperatura observată în Iunie nu este foarte diferită de temperatura medie anuală, dar este mult mai mică decât ne-am aștepta pentru o lună de Iunie. Outlierii de tip colectiv se referă la un subgrup de cazuri care sunt diferite ca grup de restul cazurilor deși, fiecare caz considerat separat, nu este diferit.

Figura 6.5-8. Outlier global vs. contextual vs. colectiv



Distanța dintre cazuri

Pentru a putea defini un caz ca outlier este nevoie să calculăm distanța / disimilaritatea dintre acesta și celelalte cazuri. Distanța dintre două cazuri poate fi definită / calculată în diferite moduri. Probabil cea mai des utilizată măsură a distanței este cea euclidiană. Aceasta se calculează simplu urmând pașii ilustrați în Tabelul 6.5-2.

Tabelul 6.5-2. Un exemplu simplu de calculare a distanței euclidiene[53]

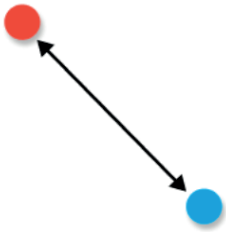
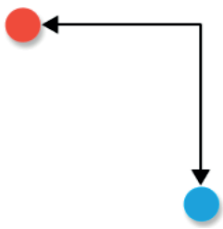
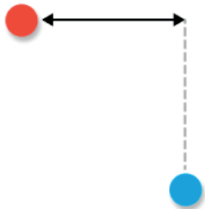
Id caz	Vârstă (ani)	Vechime (ani)	Salariu (mii)
Caz1	30	5	2
Caz2	40	8	3
Diferența	-10	-3	-1
Diferența^2	100	9	1
Suma pătratelor diferențelor	100 + 9 + 1 = 110		
Distanța (radical din sumă)	$\sqrt{110}$=33.3		

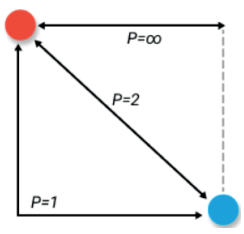
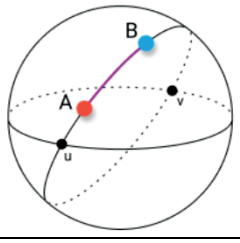
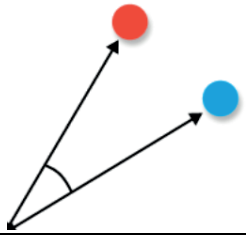
În RapidMiner, în contextul comenzilor utilizate pentru identificarea outlierilor, există și alte funcții ce pot fi folosite pentru calcularea distanțelor:

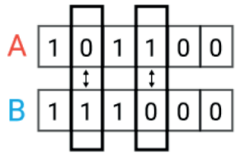
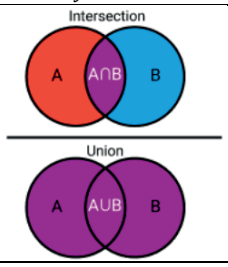
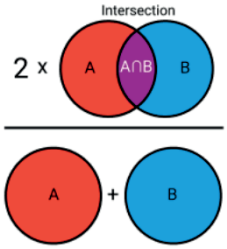
[53] Aici am lăsat valorile exprimate în unitățile originale de măsură. În practică, valorile sunt cel mai adesea normalizate pentru a acorda aceeași importanță tuturor atributelor. În acest exemplu, deoarece atributul vârstă are un interval de variație semnificativ mai mare, va avea un impact disproporționat mai mare asupra scorului obținut (distanței), deci și a stabilirii cazurilor care vor fi considerate outlieri.

squared distance (pătratul distanței euclidiene), cosine distance (distanța cosinus), inverted cosine distance (inversul distanței cosinus) și angle (unghiul). Aceste măsuri, alături de altele, sunt ilustrate și caracterizate simplu în Tabelul 6.5-3. În acest context atragem doar atenția asupra faptului că unele dintre ele sunt mai potrivite în cazul atributelor metrice, altele pentru cele non-metrice (categoriale).

Tabelul 6.5-3. Cum poate fi măsurată distanța dintre două puncte?

Distanța	Euclidian	Manhattan	Chebyshev
Reprezentare grafică			
Formulă	$D(x,y)=\sum_{i=1}^{k}\lvert x_i-y_i\rvert$	$D(x,y)=\sum_{i=1}^{k}\lvert x_i-y_i\rvert$	$D(x,y)=\max_i(\lvert x_i-y_i\rvert)$
Condiții utilizare	- număr mic de dimensiuni - mărimea vectorilor e importantă - atribute standardizate	- oricâte dimensiuni - mărimea vectorilor e importantă - atribute standardizate	- oricâte dimensiuni - ne interesează distanța maximă dintre vectori - atribute (ne)standardizate

Distanța	Minkowski	Haversine	Cosine
Reprezentare grafică			
Formulă	$D(x,y)=\left(\sum_{i=1}^{n}\lvert x_i-y_i\rvert^p\right)^{\frac{1}{p}}$	$d=2r\arcsin\left(\sqrt{\sin^2\left(\frac{\varphi_2-\varphi_1}{2}\right)+\cos(\varphi_1)\cos(\varphi_2)\sin^2\left(\frac{\lambda_2-\lambda_1}{2}\right)}\right)$	$D(x,y)=cos(\theta)=\frac{x\cdot y}{\lVert x\rVert\,\lVert y\rVert}$
Condiții aplicare	- p=1 —> Manhattan - p=2 —> Euclidean - p=∞ —> Chebyshev	- număr mic de dimensiuni - mărimea vectorilor e importantă - atribute standardizate	- oricâte dimensiuni - mărimea vectorilor nu contează - atribute (ne)standardizate

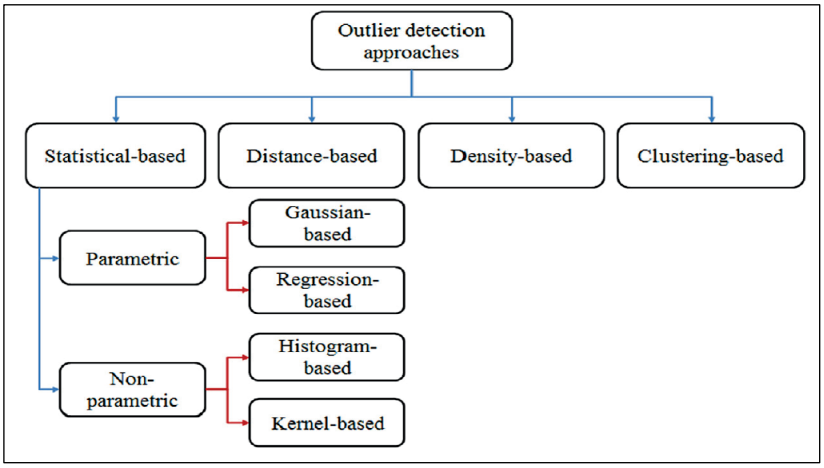
Distanța	Hamming	Jaccard	Sørensen-Dice
Reprezentare grafică			
Formulă	$D(x,y)=\sum_{i=1}^{k}\lvert x_i-y_i\rvert$	$D(x,y)=1-\frac{\lvert x\cap y\rvert}{\lvert y\cup x\rvert}$	$D(x,y)=\frac{2\lvert x\cap y\rvert}{\lvert x\rvert+\lvert y\rvert}$
Condiții aplicare	- oricâte dimensiuni - vectorii au aceeași lungime - atribute categoriale	- oricâte dimensiuni - vectorii au aceeași lungime - atribute categoriale	- oricâte dimensiuni - vectorii au aceeași lungime - atribute categoriale

Sursa reprezentărilor grafice: Grootendorst, Maarten. 2021. 9 Distance Measures in Data Science. https://towardsdatascience.com/9-distance-measures-in-data-science-918109d069fa

Detectarea outlierilor în RapidMiner Studio

Metodele de detectare a outlierilor pot fi clasificate în diferite moduri. Una dintre acestea (Figura 6.5-9), distinge patru mari categorii, funcție de criteriul utilizat pentru identificarea outlierilor: măsuri statistice, distanța dintre cazuri, densitatea cazurilor și gruparea cazurilor.

Figura 6.5-9. O clasificare a metodelor de detectare a outlierilor



Sursa: (Smiti, 2020)

În RapidMiner, operatorii grupați în categoria Outliers identifică outlierii în funcție de tipul acestora. Primii doi operatori, bazați pe distanțe, respectiv

densități, identifică outlieri de tip global, iar operatorii LOF și COF identifică outlieri de tip local, respectiv contextual.

- **Detect Outlier (Distances)**: apropierea dintre cazuri este definită cu ajutorul distanțelor dintre valorile atributelor; sunt considerate cazuri extreme primele n cazuri cu distanța cea mai mare relativ la toate celelalte cazuri;
- **Detect Outlier (Densities)**: cazurile sunt distribuite în n dimensiuni (n = numărul de atribute), unele zone având o densitate semnificativ mai mică de cazuri; un caz este considerat outlier dacă ponderea cazurilor care se află la o distanță mai mare decât o anumită valoare depășește un anumit prag;
- **Detect Outlier (LOF)**: detectează outlieri de tip local analizând variațiile de densitate (concentrare a cazurilor) dintre diferite zone;
- **Detect Outlier (COF)**: detectezi outlieri de tip contextual; atributul de referință (în relație cu care este definit caracterul de outlier) este atributul definit ca label în setul de date.

Algoritmii statistici de identificare a outlierilor presupun adesea ca utilizatorul să specifice numărul de outlieri dorit înainte de realizarea efectivă a analizei. Dat fiind faptul că facem această analiză tocmai pentru că nu știm câți outlieri avem în setul de date, cerința e oarecum ciudată. Dacă alegem un număr prea mare sau prea mic de outlieri (relativ la situația reală, necunoscută), soluția (care sunt outlierii și câți sunt) la care ajunge algoritmul poate fi diferită de situația „reală”.

Detectarea cazurilor extreme prin metoda distanțelor (Detect Outlier (Distances))

Operatorul determină care sunt cazurile extrem folosind conceptul de distanță între cazuri. Distanța poate fi calculată în diferite moduri dar, în esență, este vorba de diferența dintre valorile înregistrate de două cazuri relativ la unul sau mai multe atribute. Parametrii operatorului sunt:

- **number of neighbors**: numărul de „vecini” pe care algoritmul îi are în vedere; prin vecini ne referim la cazurile cele mai apropiate, cele care sunt relativ mai similare, adică, raportat la atributele incluse în analiză, au valori cât mai apropiate;

- **number of outliers**: numărul de cazuri pe care le considerăm că sunt outlieri;
- **distance function**: funcția care indică modalitatea de calcul a distanței dintre două cazuri.

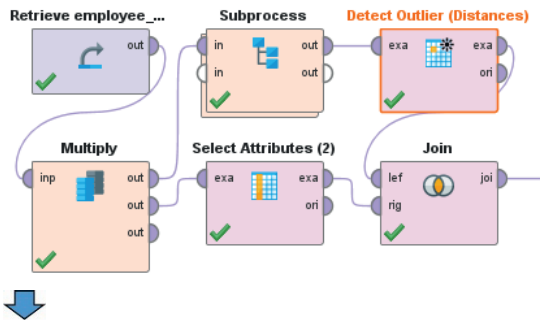
Toți cei trei parametri sunt opționali, în sensul că, dacă nu este specificată nicio valoare, algoritmul va folosi valorile implicite, adică 10, 10 și „euclidian distance” (distanța euclidiană).

După ce am calculat distanța asociată fiecărei perechi de cazuri, pentru fiecare caz calculăm distanța medie relativ la cele mai apropiate k cazuri (unde k este numărul de cazuri / vecini ales de noi). Ordonăm cazurile în funcție de această distanță medie și considerăm că sunt outlieri primele n cazuri cu cea mai mare distanță (n fiind numărul de outlieri ales de noi).

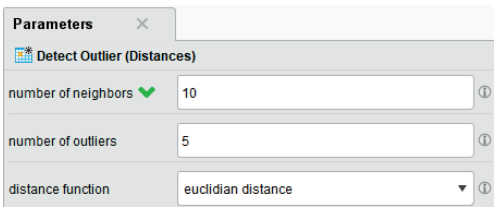
După rularea comenzii, setul de date va conține un atribut nou denumit outlier, cu două valori, true și false, valoarea true indicând faptul că acel caz este considerat a fi outlier. Numărul de outlieri va fi egal cu n, adică numărul cerut la parametrul „number of outliers”.

Figura 6.5-10. Detectarea cazurilor extreme prin metoda distanțelor (Detect Outlier (Distances))

Pasul 1:
Conectăm datele și operatorii conform imaginii alăturate (a se vedea și procesul).
Restul parametrilor ne ajută să păstrăm în setul final ambele versiuni ale atributelor (normalizate și originale).



Pasul 2:
Dorim să identificăm 5 outlieri. Pentru a calcula distanța dintre cazuri folosim funcția „distanța euclidiană”. Pentru fiecare caz alegem să calculăm distanța dintre acesta și cei mai apropiați 10 vecini (cazuri).



Rezultat:
Setul de date rezultat va conține un atribut special denumit outlier. Acesta are două valori, true și false, unde true înseamnă că acel caz este outlier (5 în total). Se observă că outlierii au valori foarte diferite comparativ cu celelalte cazuri. Outlierii sunt relativ mai tineri și au salarii relativ mai mari.

Id	out...	Age_Norm	Income_Norm	Age	Monthlyl
456	true	-0.430	2.056	33	16184
711	true	-0.430	2.324	33	17444
906	true	-0.867	2.044	29	16124
1015	true	-0.648	2.107	31	16422
1056	true	-0.320	2.231	34	17007
1	false	0.446	-0.108	41	5993
2	false	1.322	-0.292	49	5130

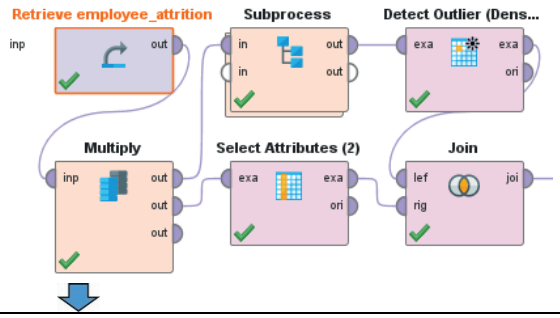
Outlier	Binominal	0	Negative	Positive	Values
outlier			false	true	false (1465), true (5)

Detectarea cazurilor extreme prin metoda densităților (Detect Outlier (Densities))

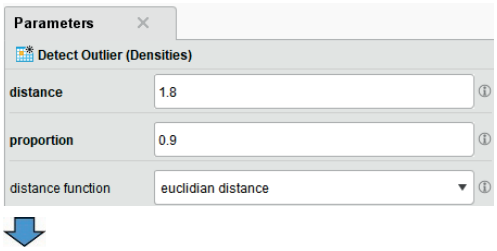
În acest caz outlierii sunt identificați prin metoda densităților. Un caz este definit ca outlier dacă ponderea cazurilor care se află la o distanță mai mare decât valoarea indicată de utilizator depășește un anumit prag ales tot de utilizator. Pe lângă aceste valori (parametrii distance și proportion), utilizatorul trebuie să aleagă și modul de calcul a distanțelor (distance function). Evident, în cazul acestui operator utilizatorul nu are posibilitatea să seteze numărul de outlieri (pot fi oricâți, funcție de valorile alese de utilizator la cei trei parametri). După rularea comenzii, setul de date va conține un atribut nou denumit outlier, cu două valori, true și false, valoarea true indicând faptul că acel caz este considerat a fi outlier.

Figura 6.5-11. Detectarea cazurilor extreme prin metoda densităților (Detect Outlier (Densities))

Pasul 1:
Conectăm datele și operatorii conform imaginii alăturate (a se vedea și procesul).
Restul parametrilor ne ajută să păstrăm în setul final ambele versiuni ale atributelor (normalizate și originale).



Pasul 2:
Distanța este stabilită la 1.8 (putem încerca mai multe valori). Proporția cazurilor care trebuie să fie la o distanță mai mare de 1.8 este setată la 90%. Pentru a măsura distanțele dintre cazuri folosim distanța euclidiană.



Rezultat:

Setul de date rezultat conține un atribut special denumit outlier. Avem 4 valori true, deci 4 outlieri. Se observă că outlierii au valori foarte diferite comparativ cu celelalte cazuri. Outlierii sunt relativ mai în vârstă[54] și au salarii relativ mai mari.

Id	ou...	Age_Norm	Income_Norm	Age	MonthlyIn...
106	true	2.417	2.621	59	18844
412	true	2.526	2.775	60	19566
596	true	2.307	2.707	58	19246
1010	true	2.307	2.803	58	19701
1	false	0.446	-0.108	41	5993
2	false	1.322	-0.292	49	5130

Outlier		Negative	Positive	Values
outlier	Binominal	0	false	true	false (1466), true (4)

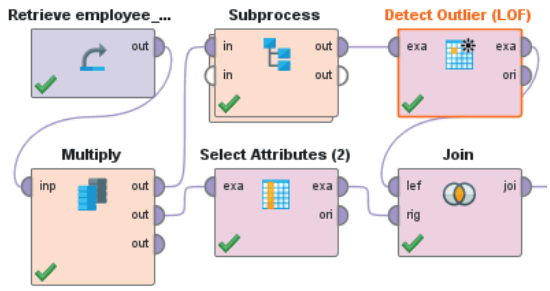
Detectarea cazurilor extreme prin metoda LOF (Detect Outlier (LOF))

Operatorul LOF compară densitatea locală (distanța medie până la cazurile din proximitate) a unui caz cu densitatea locală a vecinilor lui, identificând astfel zone cu densitate similară, respectiv cazuri care au o densitate locală semnificativ mai redusă comparativ cu a vecinilor. Cazurile cu o densitate locală relativ mai redusă sunt definite ca outlieri. Distanța dintre cazuri se poate calcula folosind orice măsură a distanței. Utilizatorul poate specifica numărul de cazuri în raport cu care vor fi calculate densitățile locale („minimal points lower bound” și „minimal points upper bound”). Spre deosebire de ceilalți doi operatori prezentați anterior, acest operator produce un atribut special de tip numeric. Valorile relativ mai mari asociate atributului indică o probabilitate mai mare ca acel caz să fie un outlier local.

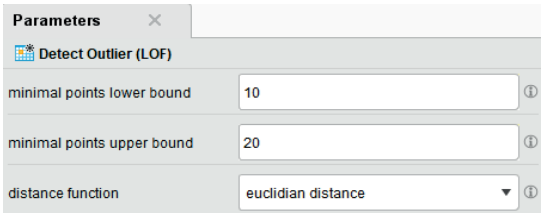
[54] Anterior, folosind metoda bazată pe distanțe, am observat că outlierii sunt relativ mai tineri și că au salarii relativ mai mari. Această inconsistență între soluțiile produse de cele două metode poate fi un semn că există mai multe tipuri de outlieri în setul nostru de date.

Figura 6.5-12. Detectarea cazurilor extreme prin metoda LOF (Detect Outlier (LOF))

Pasul 1:
Conectăm datele și operatorii conform imaginii alăturate (a se vedea și procesul).
Restul parametrilor ne ajută să păstrăm în setul final ambele versiuni ale atributelor (normalizate și originale).



Pasul 2:
Outlierii locali vor fi cei cu limitele setate între 10 și 20 puncte.
Distanța dintre cazuri este calculată folosind funcția „distanța euclidiană”.



Rezultat:
Setul de date rezultat conține un atribut special denumit outlier. Acesta este de tip numeric, valorile mari indicând o probabilitate ridicată ca acel caz să fie outlier local. Aici, outlierii au în comun faptul că sunt tineri cu salarii mici.

Id	out... ↓	Age_Norm	Income_Norm	Age	MonthlyInco...
765	3.581	-0.977	-1.158	28	1052
516	3.217	-0.211	-1.109	35	1281
1070	3.203	-0.977	-1.049	28	1563
1366	3.146	-0.867	-1.150	29	1091
912	2.905	-1.305	-1.144	25	1118

Outlier			Min	Max	Average
outlier	Real	0	0.869	3.581	1.131

Detectarea cazurilor extreme prin metoda COF (Detect Outlier (COF))

Acronimul COF înseamnă „Class Outlier Factors”. Simplu spus, detectarea outlierilor se realizează prin raportare la clasele unui atribut de tip label, deci trebuie să avem un astfel de atribut în setul de date. Algoritmul utilizat este ECODB (Enhanced Class Outlier - Distance Based), scopul fiind ordonarea cazurilor după măsura COF. COF este calculat folosind formula:

COF = PCL(T,K) - norm(deviation(T)) + norm(kDist(T)), unde

- PCL(T,K) este probabilitatea ca un caz (T) să aparțină la una dintre clasele atributului special label ținând cont de clasele la care aparțin cei mai apropiați K vecini ai acelui caz (T);
- norm(Deviation(T)) și norm(KDist(T)) sunt valorile normalizate [0 - 1] ale KDist(T) și Deviation(T);
- Deviation(T) este o măsură a gradului în care cazul T deviază de la cazurile aceleiași clase (suma distanțelor dintre un caz și toate celelalte cazuri din aceeași clasă);
- KDist(T) reprezintă suma distanțelor dintre cazul T și cei K vecini cei mai apropiați.

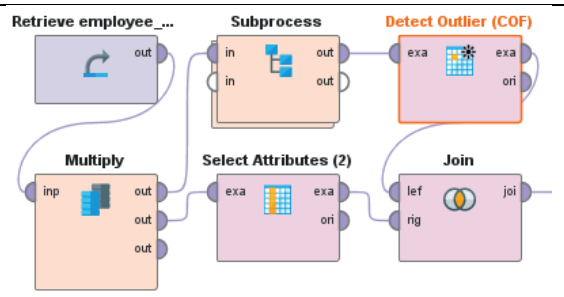
Utilizatorul poate stabili valorile relativ la parametrii numărul de outlieri, numărul de vecini și măsura folosită pentru calcularea distanței dintre cazuri. Distanțele pot fi de mai multe tipuri, funcție de nivelul de măsurare al atributelor incluse în analiză:

- **MixedMeasures**: MixedEuclidianDistances;
- **NominalMeasures**: NominalDistance, DiceSimilarity, JaccardSimilarity, KulczynskiSimilarity, RogersTanimotoSimilarity, RussellRaoSimilarity, SimpleMatchingSimilarity;
- **NumericalMeasures**: EuclidianDistance, CamberraDistance, ChebychevDistance, CorrelationDistance, CosineDistance, CosineSimilarity, DiceSimilarity, DynamicTimeWarpingDistance, InnerProductSimilarity, JaccardSimilarity, KernelEuclidianSimilarity, ManhattanDistance, MaxProductSimilarity, OverlapSimilarity;
- **BregmanDivergences**: GeneralizedDivergence, ItakuraSaitoDistance, LogarithmicLoss, LogisticLoss, MahalanobisDistance, SquareEuclidinaDistance, SquaredLoss.

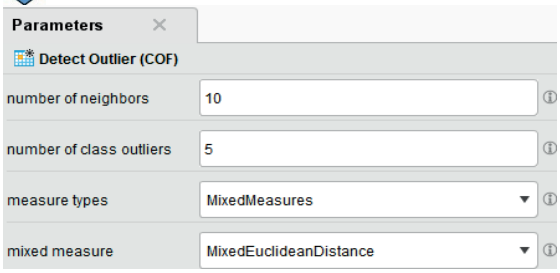
După rularea acestui operator, setul de date va include două atribute noi de tip special. Primul atribut, denumit outlier, ia două valori, true și false, unde true identifică cazurile outlier. Al doilea atribut, „COF Factor”, de tip numeric, indică valoarea COF, adică gradul în care acel caz este un outlier.

Figura 6.5-13. Detectarea cazurilor extreme prin metoda COF (Detect Outlier (COF))

Pasul 1:
Conectăm datele și operatorii conform imaginii alăturate (a se vedea și procesul).
Restul parametrilor ne ajută să păstrăm în setul final ambele versiuni ale atributelor (normalizate și originale).



Pasul 2:
Dorim să identificăm 5 outlieri relativ la fiecare clasă, raportat la cei mai apropiați 10 vecini, folosind o măsură a distanței de tip mixt, și anume MixedEuclidianDistance. O măsură de tip mixt poate calcula distanța atât pentru atribute nominale cât și metrice.



Rezultat:
Atributul special denumit outlier identifică exemplele definite ca outlier (5).
Atributul special „COF Factor” indică gradul în care acel caz este un outlier. Outlierii au venituri medii sau mari și apreciază că echilibrul dintre munca și viața lor privată este foarte bun.

Id	Attrition	outlier	COF ... ↑	MonthlyInco...	WorkLifeBal...	MonthlyInco...
46	Yes	true	0.855	2.770	Very good	19545
137	Yes	true	0.977	0.881	Very good	10650
205	Yes	true	0.995	0.036	Very good	6673
108	Yes	true	0.997	-0.161	Very good	5744
51	Yes	true	0.999	-0.238	Very good	5381
1	Yes	false	∞	-0.108	Bad	5993
2	No	false	∞	-0.292	Very good	5130

Outlier **outlier**	Binominal	0	Negative false	Positive true	Values false (1465), true (5)
Metadata **COF Factor**	Real	0	Min 0.855	Max ∞	Average ∞

6.6. Reducerea numărului de dimensiuni (Dimensionality Reduction)

Adesea, setul de date cu care lucrăm conține multe atribute, unele dintre acestea măsurând același fenomen / concept. Includerea directă într-un model predictiv a mai multor atribute care măsoară același concept crește

complexitatea modelului. Ca urmare vor crește și timpul necesar pentru a ajunge la o soluție, respectiv incertitudinea asociată rezultatelor obținute. Una dintre soluțiile posibile la această problemă constă în reducerea dimensiunilor datelor (a numărului de atribute). Ideea este de a combina atributele care măsoară același concept într-un număr mai mic de atribute, numite dimensiuni sau componente, care să rețină cât mai mult din informația asociată atributelor inițiale.

Combinarea atributelor se poate face simplu, aditiv sau ca medie a valorilor, respectiv folosind un model. În primul caz, fiecare indicator va avea (cel mai adesea) aceeași importanță, indiferent de aspectul măsurat (importanța este stabilită de analist). În al doilea caz, ținem cont de relația dintre fiecare indicator și conceptul măsurat, deci atributele vor avea o importanță diferită la scorul final (importanța este estimată de model). Deși soluțiile produse de cele două metode de agregare a informațiilor corelează puternic cel mai adesea, varianta de agregare în baza unui model este de preferat.

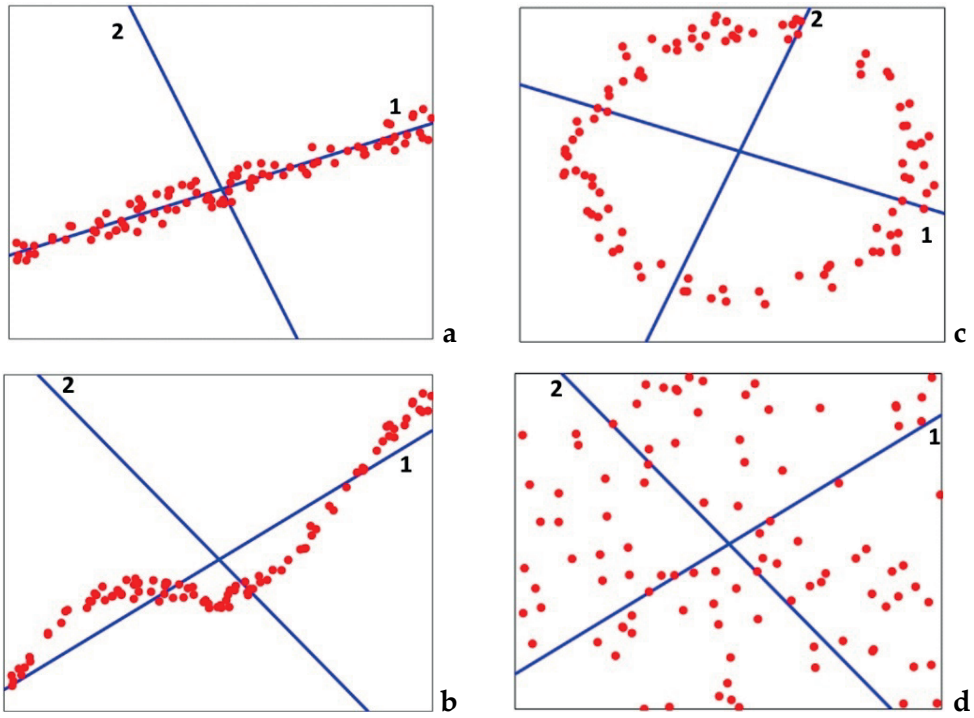
Să presupunem că dorim să măsurăm satisfacția relativ la locul de muncă. Pentru aceasta, am colectat date relativ la mai multe dimensiuni ale satisfacției, una dintre ele fiind satisfacția relativ la recompensele materiale. Această dimensiune include de obicei satisfacția relativ la salariu, prime / bonusuri, participare la profit, creșterea salariului, recompensele în acțiuni, planurile de sănătate și pensie, diferite tipuri de facilități (abonamente, acces). Scala utilizată a fost una ordinală, de 10 puncte (în acest context considerăm că este de tip numeric). Dacă avem patru dimensiuni, fiecare cu opt indicatori, vom avea în total 32 de indicatori ai satisfacției, deci 32 de atribute în setul de date. Și aceasta pentru a măsura un singur concept. Modelul nostru de predicție a părăsirii companiei va include însă o serie de alte concepte, fiecare măsurat prin mai mulți indicatori (deci tot atâtea atribute / variabile în setul de date). Complexitatea modelului va crește exponențial. Putem să o reducem prin combinarea atributelor. Concret, în acest caz, putem însuma valorile indicatorilor asociați dimensiunii satisfacția relativ la recompensele materiale (alternativ, putem calcula

media valorilor) sau putem folosi unul dintre operatorii care modelează relațiile dintre indicatori, extrăgând astfel informația utilă simultan cu reducerea numărului de atribute.

Pentru a ilustra cât mai clar relația dintre numărul de atribute inițiale și numărul de dimensiuni (atributele finale) am ales o situație simplă, ce poate fi vizualizată într-un spațiu bidimensional (Figura 6.6-1). Presupunem că avem două atribute și dorim să analizăm posibilitatea de a exprima informația folosind o singură dimensiune (atribut nou). Putem avea diferite situații, patru dintre acestea fiind următoarele:

- **panel a:** cazurile variază dominant de-a lungul dreptei 1, deci dimensiunea 1 explică cea mai mare parte a variației; prin urmare, la nivel liniar, avem o singură dimensiune;
- **panel b:** cazurile variază dominant de-a lungul dreptei 1, deci dimensiunea 1 explică cea mai mare parte a variației; prin urmare, la nivel liniar, avem o singură dimensiune (chiar dacă dimensiunea 2 are o importanță mai mare comparativ cu dimensiunea 2 din panelul a);
- **panel c:** fiecare dimensiune explică aproximativ jumătate din varianță, deci, la nivel liniar, avem două dimensiuni; soluția non-liniară, cea „corectă”, are o singură dimensiune; pentru astfel de situații folosim algoritmii non-liniari (kernel PCA, ICA, SOM) sau un alt sistem de coordonate (polar în loc de cartezian).
- **panel d:** fiecare dimensiune explică aproximativ jumătate din varianță, deci, la nivel liniar, avem două dimensiuni; soluția obținută este aproape identică cu soluția observată la panelul c; dat fiind faptul că în acest caz atributele sunt generate aleator, rezultă că PCA (Principal Component Analysis) nu distinge între cele două situații complet diferite (structură non-liniară vs. absența unei structuri).

Figura 6.6-1. Două atribute, relații diferite, dimensiuni diferite



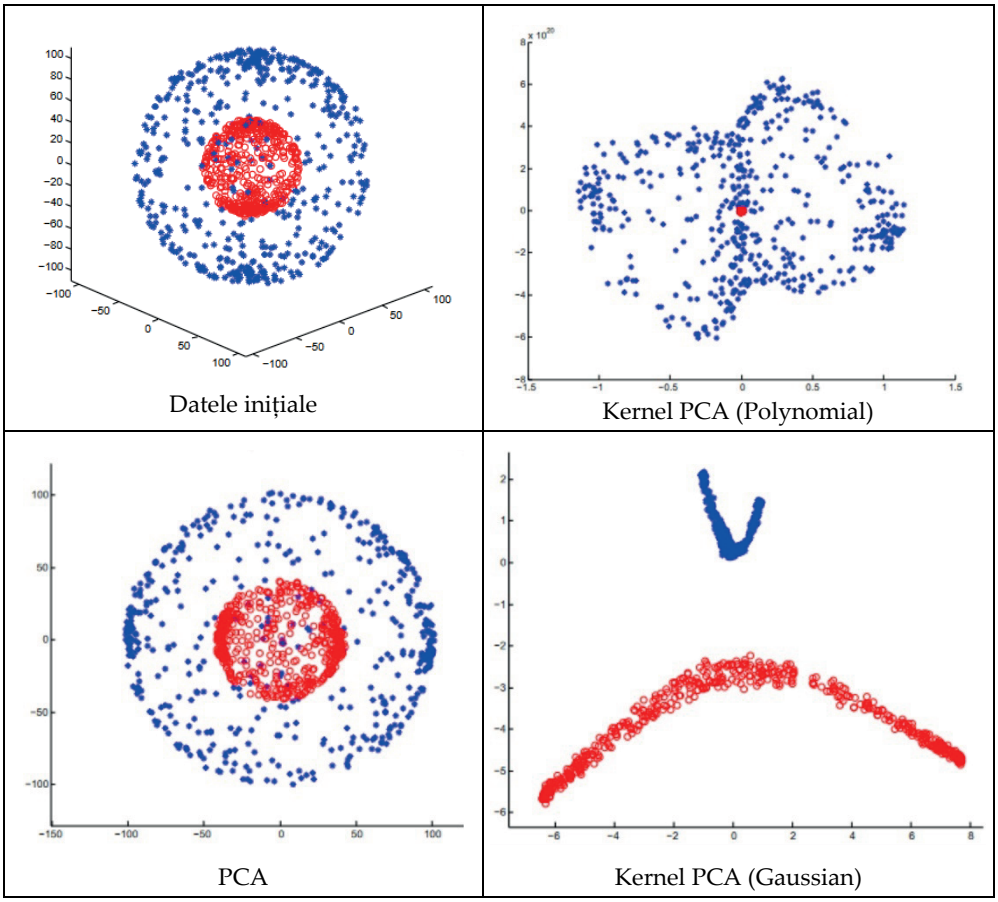
Dacă relațiile sunt de tip non-liniar, PCA nu poate produce o soluție care să separe cazurile. Kernel[55] PCA poate fi utilă în acest context. Însă, nu toate tipurile de kernel sunt la fel de capabile să transforme datele astfel încât să putem separa liniar între ele. În Figura 6.6-2 am ilustrat comparativ soluțiile obținute de diferite tipuri de PCA în cazul unor date de tip non-liniar (două sfere concentrice). Scopul este de a reduce dimensionalitatea datelor (de la trei dimensiuni la două sau chiar una). Observăm următoarele:

- după transpunerea datelor inițiale în două dimensiuni, PCA nu reușește să separe punctele care compuneau cele două sfere;
- în cazul Kernel PCA (Polynomial), după transformare, punctele asociate uneia dintre sfere sunt relativ mai apropiate, iar cele asociate celeilalte sunt relativ mai dispersate; și în acest caz datele proiectate nu pot fi separate liniar;

[55] Funcțiile kernel sunt folosite pentru a proiecta datele de input într-un spațiu vectorial multidimensional superior în vederea identificării unui hiperplan care să le separe.

– în cazul Kernel PCA (Gaussian) punctele asociate celor două sfere sunt bine separate; e important să adăugăm că alegerea valorii parametrului contează (gradul de separare liniară depinde mult de valoarea acestui parametru).

Figura 6.6-2. PCA vs. kernel PCA în cazul unor date non-liniare



Sursa: (Wang, 2014)

Secțiunea „Dimensionality Reduction” din fereastra Operators include cinci operatori care pot fi folosiți pentru a reduce dimensiunile, și anume:

– **Principal Component Analysis** (PCA): analiza componentelor principale; componente rezultate în urma aplicării PCA sunt combinații liniare ale atributelor inițiale;

- **Principal Component Analysis (Kernel)** (Kernel PCA): analiza componentelor principale folosind o tehnică de tip Kernel; componente rezultate în urma aplicării kernel PCA sunt combinații non-liniare ale atributelor inițiale;
- **Independent Component Analysis** (ICA): analiza componentelor independente; componentele rezultate în urma aplicării ICA sunt combinații liniare ale atributelor inițiale;
- **Singular Value Decomposition** (SVD): descompunerea în valori singulare; valorile singulare rezultate în urma aplicării PCA sunt combinații liniare ale atributelor inițiale;
- **Self-Organizing Map** (SOM): hartă auto-organizată.

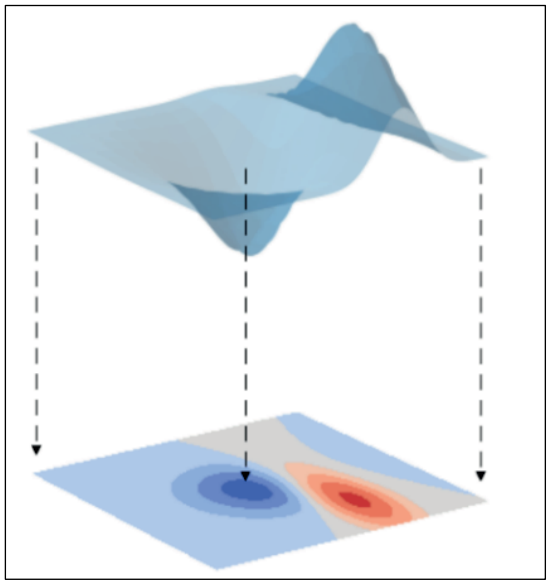
În cazul PCA, ICA și SVD fiecare dintre dimensiunile rezultate (componente) este exprimată ca o funcție liniară a atributelor inițiale. În cazul celorlalți operatori funcțiile pot lua diferite forme non-liniare (utilizatorul specifică funcția; e util să comparăm rezultatele produse de diferite funcții).

Analiza componentelor principale (Principal Component Analysis) (PCA)

Acest tip de analiză își propune să reducă numărul atributelor dintr-un set de date prin identificarea componentelor principale ale acestora folosind matricea de varianță-covarianță a atributelor. Scopul este de a identifica cel mai mic număr de componente / dimensiuni (noile atribute) care să păstreze în același timp cât mai mult din varianța (informația) atributelor inițiale.

În exemplul din Figura 6.6-3 observăm cum un spațiu tridimensional poate fi transpus fără pierdere de informație într-un spațiu bidimensional. Este exact ceea ce PCA urmărește să facă: reducerea numărului de atribute (dimensiuni) folosite pentru reprezentarea unor relații, concomitent cu păstrarea unei părți cât mai mari a informației inițiale. Desigur, numărul inițial de dimensiuni (componente) poate fi mai mare de trei, iar cel final mai mare de doi. Însă, întotdeauna, numărul final de dimensiuni va fi cel mult egal cu numărul inițial de atribute.

Figura 6.6-3. O ilustrare vizuală a scopului PCA (de la 3 la 2 dimensiuni)



Sursa: https://nirpyresearch.com/pca-kernel-pca-explained/

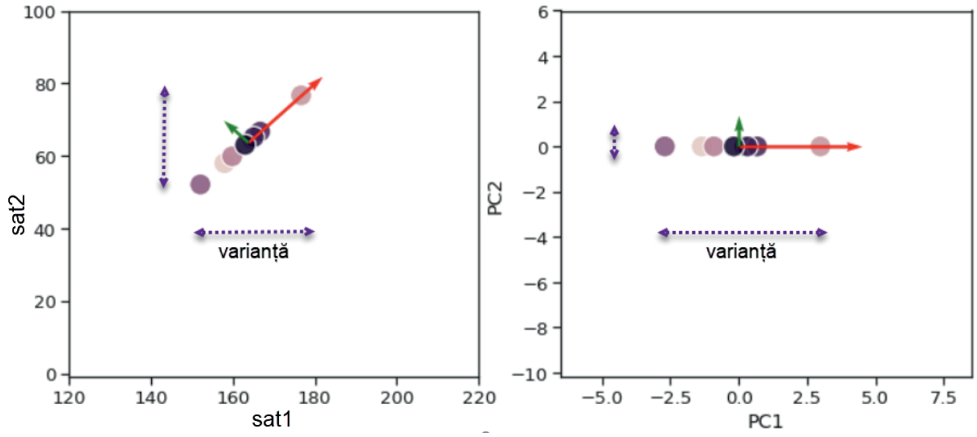
Pentru a transforma datele, PCA operează în doi pași: (1) încearcă „să înțeleagă” datele, să-și de-a seama care atribute sunt relativ mai importante, au mai multă informație, și care sunt mai puțin importante (conțin mai puțină informație) și (2) sumarizează datele (reduce numărul de dimensiuni). Pentru a realiza aceste scopuri, PCA definește importanța unui atribut folosind conceptul de varianță. Un atribut este cu atât mai important, conține cu atât mai multă informație, cu cât varianța asociată lui este mai mare. Simplu spus, varianța reprezintă o modalitate obiectivă (formulată matematic) de cuantificare a cantității de informație asociată unui atribut (unor atribute).

Să presupunem că am măsurat satisfacția angajaților prin doi indicatori, sat1 și sat2 (Figura 6.6-4). Firesc, cei doi indicatori corelează perfect (panel stânga). Indicatorii au aproximativ aceeași varianță.[56] În panelul din dreapta am rotit axele inițiale (roșu și verde) spre dreapta. Observăm că prima componentă

[56] În imagine am sugerat doar mărimea variației. Aceasta poate fi calculată folosind formula $S^2 = \frac{\sum(x_i - \bar{x})^2}{n-1}$, unde n = numărul de cazuri, x_i = valoarea atributului x pentru cazul i, $\bar{x}$ = media atributului x.

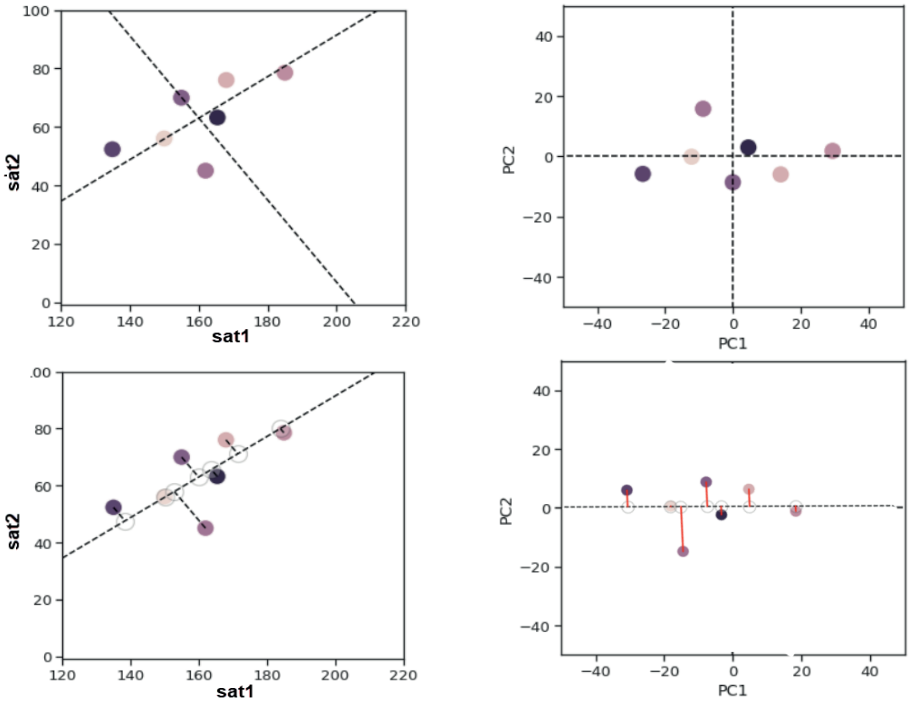
(axa x = PC1) captează toată varianța celor două atribute inițiale. A doua componentă are o varianță nulă, deci putem renunța la ea fără să pierdem informație.

Figura 6.6-4. Aceleași date înainte și după PCA (o componentă)



Situațiile de tipul celei prezentate în exemplul din Figura 6.6-4 (una dintre componente reține 100% din varianța inițială a datelor) sunt extrem de rare atunci când lucrăm cu date reale. În practică, lucrurile arată mai degrabă precum cele din Figura 6.6-5. De această dată, cei doi indicatori ai satisfacției au o corelație pozitivă, dar nu perfectă. În urma aplicării PCA (axele inițiale sunt rotite spre dreapta) rezultă două componente. PC1 reține o pondere semnificativ mai mare a variației inițiale (împrăștierea punctelor relativ la axa orizontală – PC1 – este clar mai mare). În acest caz, dacă alegem să reținem doar prima componentă, vom pierde informație. Cantitatea de informație pierdută este ilustrată cu ajutorul liniilor întrerupte / roșii. Aceste linii indică distanța dintre poziția inițială a punctelor (spațiu bidimensional) versus poziția finală (spațiu unidimensional). Renunțarea la una sau mai multe dintre componente distorsionează distanțele dintre cazuri (distanțele inițiale pot deveni mai mici sau mari mari). Mai mult, distorsiunea corelează negativ cu distanța inițială dintre cazuri (distorsiunea este mai mare atunci când distanța inițială este mai mică).

Figura 6.6-5. Aceleași date înainte și după PCA (două componente)



Sursa: Exemplul și imaginile sunt construite de autor pornind de la articolul lui Cheng (Cheng, Casey. 2022. Principal Component Analysis (PCA) Explained Visually with Zero Math. https://towardsdatascience.com/principal-component-analysis-pca-explained-visually-with-zero-math-1cbf392b9e7d

Pentru a înțelege și mai bine modul în care PCA transformă datele dintr-un sistem de coordonate în altul, să considerăm un alt exemplu (Figura 6.6-6). În această figură, punctele albastre reprezintă cazurile, iar cele roșii reprezintă proiecția punctelor albastre pe dreapta care se rotește. Din mulțimea dreptelor posibile, PCA caută dreapta care îndeplinește simultan următoarele două condiții:

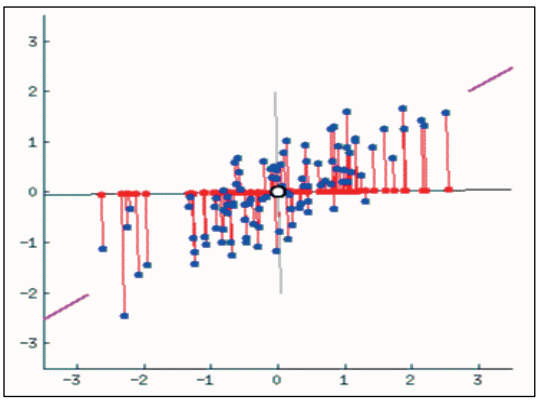
- variația valorilor luate de punctele roșii pe această dreaptă să fie maximă; în imagine se poate observa cum împrăștierea (variația) punctelor roșii se schimbă atunci când dreapta se rotește;
- eroarea de reconstrucție a coordonatelor inițiale (în două dimensiuni) pe baza noilor coordonate (o dimensiune) este minimă; eroarea este egală cu suma liniilor roșii, adică suma distanțele dintre fiecare punct albastru și proiecția lui pe dreaptă (punctul roșu corespunzător);

Cele două criterii (varianță maximă și eroare minimă) sunt îndeplinite simultan atunci când dreapta cu punctele proiectate (roșii) se suprapune peste cele două linii magenta. Această dreaptă va fi prima componentă principală (dimensiune). A doua componentă va fi aleasă similar, cu condiția suplimentară de a fi ortogonală (perpendiculară) pe prima componentă. Altfel spus, prima componentă va extrage cea mai mare varianță posibilă (va explica cât mai mult posibil din varianța atributelor inițiale), următoarea cea mai mare varianță posibilă din cea rămasă și tot așa până la ultima componentă; toate covariațele (deci și corelațiile) dintre noile atribute (componente) vor fi zero (acestea sunt ortogonale / independente).

Componentele rezultate vor defini noul sistem de axe în care sunt reprezentate punctele. Aceste componente (noile dimensiuni / atribute) sunt combinații liniare ale vechilor axe (dimensiuni / atribute). Atenție, PCA produce soluții diferite dacă scala (unitatea de măsură a) atributelor inițiale este schimbată.

În general, atributele care măsoară același concept au aceeași scală. Însă, uneori, varianța acestor atribute diferă semnificativ. Dacă observăm că se întâmplă acest lucru, avem la dispoziție două soluții: (1) folosim matricea de corelații în locul celei de varianță-covarianță sau (2) normalizăm atributele înainte de a le introduce în analiza PCA.

Figura 6.6-6. Identificarea primei componente în analiza PCA



Sursa: https://stats.stackexchange.com/questions/2691/making-sense-of-principal-component-analysis-eigenvectors-eigenvalues

Pentru a extrage componentele principale avem la dispoziție trei opțiuni (vezi parametrul „dimensionality reduction”):

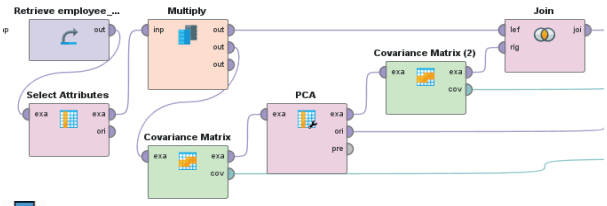
- **none** (niciuna): softul va păstra toate componentele rezultate (numărul lor va fi egal cu numărul de atribute inițiale);
- **fixed number** (număr fix): indicăm numărul de componente pe care dorim să le păstrăm; acest număr nu poate fi mai mare decât numărul de atribute inițiale;
- **keep variance** (păstrează varianța): indicăm pragul variației explicate; vor fi păstrate toate componentele care au o valoare a variației explicate cel puțin egală cu acest prag.

În cele ce urmează, prezentăm două exemple de utilizare a PCA, unul în care reducem dimensiunile alegând la parametrul „dimensionality reduction” opțiunea „fixed number”, iar la celălalt „keep variance”. Profităm de ocazie pentru a ilustra utilizarea matricei de varianță-covarianță, respectiv extragerea rezultatelor obținute. În ambele exemple am folosit un set de date cu puține cazuri și șase atribute de interes, trei care măsoară satisfacția angajaților și trei nivelul de frustrare. Scorurile iau valori în intervalul 1-6, unde 6 indică un nivel ridicat de satisfacție, respectiv frustrare. În acest context am tratat toate atributele ca numerice.

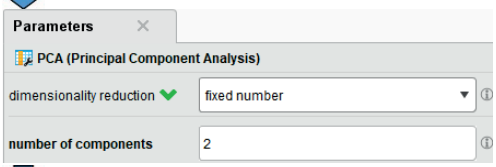
În exemplul din Figura 6.6-7, prima dată am calculat matricea de varianță-covarianță (aceasta e folosită ca imput pentru PCA), apoi am ales să extragem un număr fix de componente (două). Conform așteptărilor teoretice, indicatorii aferenți unui concept corelează pozitiv între ei, respectiv negativ cu indicatorii celuilalt concept. Dimensiunile (componentele) rezultate sunt ortogonale (corelație zero). Intuitiv, prima componentă măsoară starea pozitivă, adică satisfacție mare și frustrare redusă. Cu cât valoarea acestei componente este mai mare, cu atât angajatul declară o stare relativ mai pozitivă. A doua componentă pare să măsoare situația în care angajatul este satisfăcut și frustrat în același timp. O astfel de situație poate fi una reală, respectiv poate fi rezultatul erorilor de măsurare (respondenți care au ales scoruri mari fără a citi suficient de atent formulările itemilor).

Figura 6.6-7. Analiza componentelor principale (Principal Component Analysis) (1)

Pasul 1:
Conectăm datele și operatorii conform imaginii alăturate (a se vedea și procesul).



Pasul 2:
Dorim să păstrăm două componente principale.



Rezultat:
Matricea de varianță-covarianță inițială. Covariațele sunt diferite de zero. Variațele sunt în intervalul 1.1-2.5.

Attribut...	work fr...	work fr...	work fr...	job sati...	job sati...	job sati...
work frus...	2.510	1.613	1.810	-0.707	-0.885	-0.694
work frus...	1.613	2.608	1.564	-0.701	-0.829	-0.578
work frus...	1.810	1.564	2.690	-0.828	-1.214	-0.732
job satisf...	-0.707	-0.701	-0.828	1.356	1.213	0.891
job satisf...	-0.885	-0.829	-1.214	1.213	1.717	1.043
job satisf...	-0.694	-0.578	-0.732	0.891	1.043	1.116

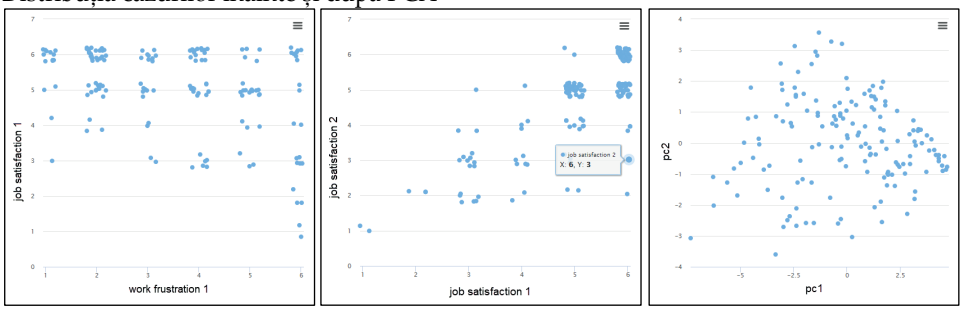
Matricea de varianță-covarianță a componentelor. Covariațele sunt zero (componentele nu corelează). Prima componentă are o varianță semnificativ mai mare comparativ cu a doua.

Attribut...	pc_1	pc_2
pc_1	7.442	-0.000
pc_2	-0.000	2.073

Fragment din setul de date produs: valorile atributelor inițiale vs. valorile componentelor.

work fr...	work fr...	work fr...	job s...	job s...	job s...	pc_1	pc_2
5	5	6	5	4	4	-3.262	0.776
2	4	2	6	6	6	2.257	1.072
5	3	3	5	5	4	-0.415	-0.001
6	6	6	6	3	3	-4.538	1.067

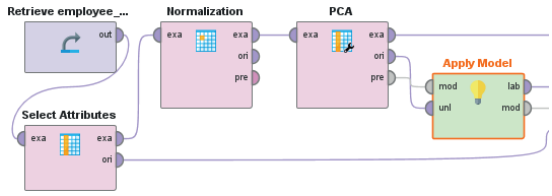
Distribuția cazurilor înainte și după PCA



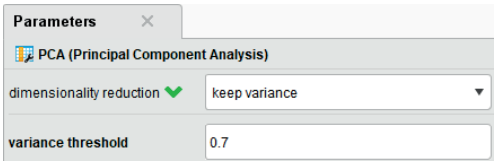
În exemplul secund (Figura 6.6-8), prima dată am normalizat atributele (scoruri z), apoi am folosit scorurile normalizate ca input pentru PCA. Am ales să păstrăm un număr de dimensiuni (componente) care împreună să însumeze 70% din varianța inițială a datelor. La final am extras rezultatele folosind operatorul „Apply Model”. Prima componentă indică situația pozitivă (frustrare redusă și satisfacție mare) și este responsabilă pentru 61% din varianța datelor. A doua componentă corelează pozitiv cu toți indicatorii (angajați care sunt simultan frustrați și satisfăcuți) și reține 19% din varianța datelor.

Figura 6.6-8. Analiza componentelor principale (Principal Component Analysis) (2)

Pasul 1:
Conectăm datele și operatorii conform imaginii alăturate (a se vedea și procesul).



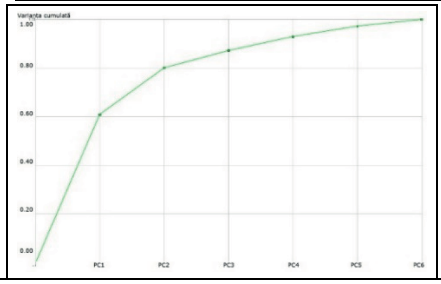
Pasul 2:
Dorim să păstrăm toate componentele principale până la pragul de 70% varianță explicată cumulativă.



Rezultat:
Relația dintre componente și atribute. PC1 combină satisfacția și lipsa frustrării.

Comp	PC1	PC2	PC3	PC4	PC5	PC6
work frustration 1	-0.39	0.46	0.25	0.61	-0.40	0.21
work frustration 2	-0.36	0.47	-0.77	-0.17	0.18	0.04
work frustration 3	-0.41	0.35	0.55	-0.46	0.24	-0.37
job satisfaction 1	0.42	0.41	0.19	0.15	0.64	0.44
job satisfaction 2	0.45	0.35	-0.11	0.31	-0.04	-0.76
job satisfaction 3	0.41	0.40	0.04	-0.52	-0.59	0.24

Varianța explicată și varianța explicată cumulată. Primele două componente rețin 61%, respectiv 19% din varianța totală.



Comp	SD	Var	VarC
PC 1	1.91	0.61	0.61
PC 2	1.07	0.19	0.80
PC 3	0.65	0.07	0.87
PC 4	0.58	0.06	0.93
PC 5	0.51	0.04	0.97
PC 6	0.41	0.03	1.00

Atributele inițiale vs. componentele.

id	quit	age	te...	wo...	wo...	wo...	job ...	job ...	job ...
1	no	36	6	5	5	6	5	4	4
2	no	63	31	2	4	2	6	6	6
3	yes	46	42	5	3	3	5	5	4
4	no	44	74	6	6	6	6	3	3
5	no	41	104	5	5	6	5	2	5

id	quit	pc_1	pc_2
1	no	-1.982	0.855
2	no	1.812	0.482
3	yes	-0.439	-0.104
4	no	-2.823	1.144
5	no	-2.271	0.709

Principal Component Analysis (Kernel) (Kernel PCA)

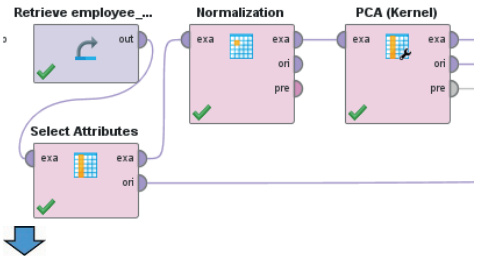
În cazul PCA, componentele sunt combinații liniare ale atributelor inițiale. Dacă datele (relațiile dintre atribute) sunt complexe, non-liniare, PCA nu le poate separa. Pentru astfel de date avem nevoie de o extensie a PCA, și anume kernel PCA. Kernel este un nume generic dat metodelor care fac posibilă folosirea unui clasificator de tip liniar la o problemă non-liniară prin translatarea datelor non-liniare într-un spațiu cu mai multe dimensiuni (egal cu numărul de cazuri). Translatarea se face fără a fi necesară calcularea efectivă a coordonatelor cazurilor în noul spațiu. Astfel, pentru a putea separa liniar datele, kernel PCA (1) calculează distanțele dintre fiecare pereche de cazuri, (2) aplică funcția kernel acestei matrice, apoi (3) supune datele unei analize PCA obișnuite. Altfel spus, kernel PCA proiectează datele într-un spațiu cu mult mai multe dimensiuni decât cele inițiale (exact opusul a ceea ce face PCA), identifică în acest nou spațiu o funcție liniară care separă datele cât mai bine, apoi transformă funcția găsită în spațiul original.

Atributele pot fi combinate non-liniar într-un număr foarte mare de moduri (putem folosi multe funcții, fiecare cu diferite valori ale parametrilor specifici). Acest lucru crește foarte mult complexitatea calculelor. Prin urmare, spre deosebire de PCA, atunci când numărul de cazuri este foarte mare, kernel PCA va rula mult mai lent. Similar cu PCA, kernel PCA poate gestiona ușor un număr mare de atribute.

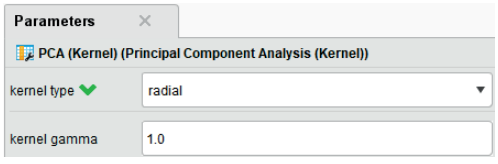
Putem alege între diferite tipuri de kernel: dot, radial, polynomial, sigmoid, anova, epanechnikov, gausian_combination, multiquadratic. Cu excepția tipului dot, pentru toate celelalte avem posibilitatea să setăm valorile parametrilor specifici.

Figura 6.6-9. Analiza componentelor principale (Kernel) (Principal Component Analysis (Kernel)) (Kernel PCA)

Pasul 1:
Conectăm datele și operatorii conform imaginii alăturate (a se vedea și procesul).



Pasul 2:
Am lăsat valorile implicite ale parametrilor.



Rezultat:
Fragment din setul de date produs: valorile normalizate ale atributelor originale vs. valorile componentelor.
Numărul componentelor este 156 (același cu numărul de cazuri).

id	quit	work f...	work f...	work f...	job s...	job s...	job s...
1	no	0.943	0.973	1.532	-0.061	-0.484	-0.940
2	no	-0.951	0.353	-0.907	0.798	1.042	0.953
3	yes	0.943	-0.266	-0.297	-0.061	0.279	-0.940
4	no	1.574	1.592	1.532	0.798	-1.247	-1.887
5	no	0.943	0.973	1.532	-0.061	-2.010	0.006

id	quit	kpc_1	kpc_2	kpc_3	kpc_4	kpc_5	kpc_6	kpc_154	kpc_155	kpc_156
1	no	0.021	-0.018	-0.044	-0.024	-0.023	0.065	0.021	0.001	0.040
2	no	0.069	0.104	0.040	-0.130	0.216	-0.107	0.164	0.235	-0.053
3	yes	0.037	0.005	0.011	-0.070	-0.045	0.023	-0.050	0.752	0.102
4	no	0.001	-0.087	-0.003	-0.007	-0.009	-0.091	-0.000	-0.012	-0.029
5	no	-0.014	-0.158	-0.058	0.019	0.001	0.003	0.000	0.026	0.005

Analiza componentelor independente (Independent Component Analysis) (ICA)

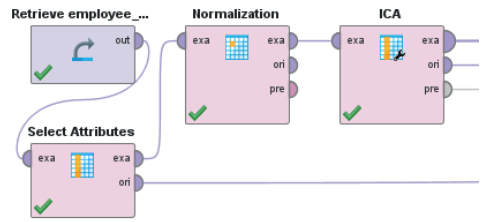
ICA este utilizată pentru a identifica factorii aflați în spatele unor atribute. ICA reușește adesea să găsească structura ascunsă a datelor chiar și în situațiile în care metodele clasice (PCA sau analiza factorială) eșuează (Hyvärinen et al., 2001). ICA are la bază două asumpții, ambele relativ la componente: (1) sunt statistic independente și (2) au o distribuție non-gausiană (nu urmează o curbă normală). Principalele aplicații ale ICA au în comun separarea unui semnal audio astfel încât să reproducă separat sursele

originale ale acestuia. ICA este mai puțin utilizată în științele sociale deoarece, cel mai adesea, nu știm dacă prima asumpție este îndeplinită sau nu (componentele sunt statistic independente).

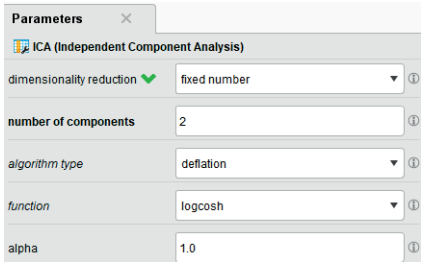
În Figura 6.6-10 am reluat analiza prezentată în cazul PCA, de această dată folosind ICA. Cele două componente rezultate par să fie similare cu cele obținute în cazul PCA. De exemplu, prima componentă corelează negativ cu atributele care măsoară frustrarea (-0.478, -0.423, -0.563), respectiv pozitiv cu atributele ce măsoară satisfacția (0.911, 0.927, 0.895). Fiecare dintre cele trei atribute asociate unui concept (frustrarea, satisfacția) are o contribuție similară la scorul primei dimensiuni. Atributele asociate satisfacției au o contribuție relativ mai mare comparativ cu atributele asociate frustrării.

Figura 6.6-10. Analiza componentelor independente (Independent Component Analysis) (ICA)

Pasul 1:
Conectăm datele și operatorii conform imaginii alăturate (a se vedea și procesul).



Pasul 2:
Dorim să păstrăm două componente principale.



Rezultat:
Fragment din setul de date produs: valorile normalizate ale atributelor originale vs. valorile componentelor.

id	quit	work f...	work f...	work f...	job s...	job s...	job s...
1	no	0.943	0.973	1.532	-0.061	-0.484	-0.940
2	no	-0.951	0.353	-0.907	0.798	1.042	0.953
3	yes	0.943	-0.266	-0.297	-0.061	0.279	-0.940
4	no	1.574	1.592	1.532	0.798	-1.247	-1.887
5	no	0.943	0.973	1.532	-0.061	-2.010	0.006

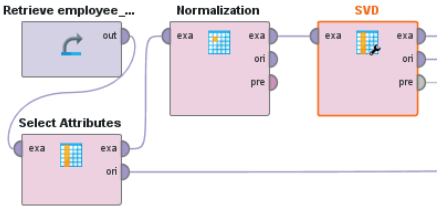
id	quit	ic_1	ic_2
1	no	-0.620	1.155
2	no	1.052	0.022
3	yes	-0.250	0.006
4	no	-0.912	1.583
5	no	-0.815	1.093

Descompunerea în valori singulare (Singular Value Decomposition) (SVD)

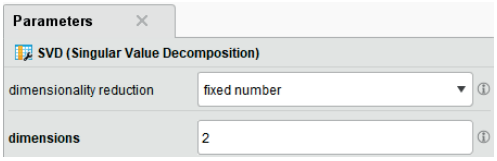
SVD reduce numărul de atribute dintr-un set de date. Reducerea este cu atât mai mare cu cât atributele respective sunt dependente liniar unul de altul. SVD este o generalizare a PCA (sau PCA este o particularizare a SVD). Și în acest caz putem alege un număr fix de atribute sau un prag pentru varianța cumulată reținută. Reluarea analizelor anterioare folosind SVD produce o soluție similară (Figura 6.6-11). De această dată, primele două componente rețin o pondere relativ mai mică din informația inițială (58%).

Figura 6.6-11. Descompunerea în valori singulare (Singular Value Decomposition) (SVD)

Pasul 1:
Conectăm datele și operatorii conform imaginii alăturate (a se vedea și procesul).



Pasul 2:
Dorim să păstrăm două componente principale. Alternativ, putem indica varianța cumulată minimă dorită.



Rezultat:
Descrierea componentelor rezultate: valorile, proporțiile, valorile cumulate, proporțiile cumulate.
Fragment din setul de date produs: valorile normalizate ale atributelor originale vs. valorile componentelor.

Comp	SV	%SV	CumSV	Cum%SV
SVD 1	23.81	0.37	23.81	0.37
SVD 2	13.36	0.21	37.17	0.58
SVD 3	8.12	0.13	45.29	0.71
SVD 4	7.27	0.11	52.56	0.82
SVD 5	6.32	0.10	58.88	0.92
SVD 6	5.09	0.08	63.97	1.00

id	quit	work f...	work f...	work f...	job s...	job s...	job s...
1	no	0.943	0.973	1.532	-0.061	-0.484	-0.940
2	no	-0.951	0.353	-0.907	0.798	1.042	0.953
3	yes	0.943	-0.266	-0.297	-0.061	0.279	-0.940
4	no	1.574	1.592	1.532	0.798	-1.247	-1.887
5	no	0.943	0.973	1.532	-0.061	-2.010	0.006

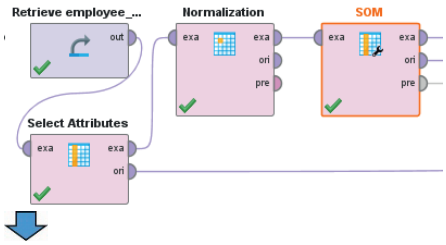
id	quit	svd_1	svd_2
1	no	0.083	-0.064
2	no	-0.076	-0.036
3	yes	0.018	0.008
4	no	0.119	-0.086
5	no	0.095	-0.053

Hartă auto-organizată (Self-Organizing Map) (SOM)

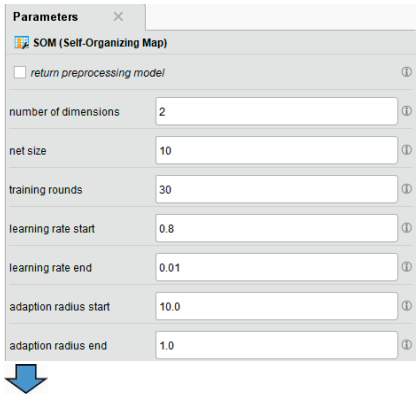
SOM (numită și Kohonen map) este un tip de rețea neuronală care este antrenată folosind o tehnică nesupervizată (nu urmărește predicția unei variabile dependente, doar identificarea relațiilor dintre mai multe variabile, toate cu același statut). SOM este utilă pentru a vizualiza date multidimensionale într-un spațiu bidimensional (de obicei). Utilizatorul poate specifica numărul de dimensiuni și o serie de alți parametri care definesc rețeaua neuronală (mărimea rețelei, numărul de runde de învățare, rata de învățare la start și la final, raza sferei la start și la final) (Figura 6.6-12).

Figura 6.6-12. Hartă auto-organizată (Self-Organizing Map) (SOM)

Pasul 1:
Conectăm datele și operatorii conform imaginii alăturate (a se vedea și procesul).



Pasul 2:
Dorim să păstrăm două dimensiuni.
Valorile parametrilor sunt cele implicite. Net size (mărimea rețelei e 10), deci valoarea maximă luată de fiecare dintre dimensiunile rezultate va fi 9.



Rezultat:
Fragment din setul de date produs: valorile normalizate ale atributelor originale vs. valorile componentelor. Valoarea maximă este 9 (10-1).

id	quit	work f...	work f...	work f...	job s...	job s...	job s...
1	no	0.943	0.973	1.532	-0.061	-0.484	-0.940
2	no	-0.951	0.353	-0.907	0.798	1.042	0.953
3	yes	0.943	-0.266	-0.297	-0.061	0.279	-0.940
4	no	1.574	1.592	1.532	0.798	-1.247	-1.887
5	no	0.943	0.973	1.532	-0.061	-2.010	0.006

id	quit	SOM_0	SOM_1
1	no	7	9
2	no	9	4
3	yes	2	8
4	no	6	9
5	no	8	7

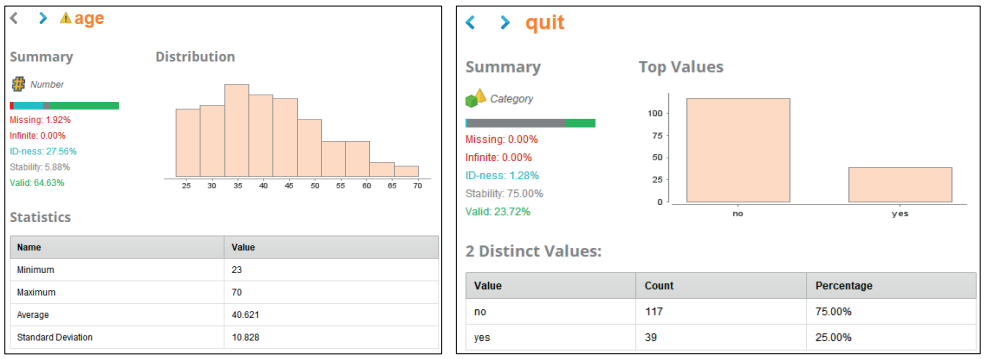
			Min	Max	Average
SOM_0	Numeric	0	0	9	4.763
SOM_1	Numeric	0	0	9	4.699

Descrierea statistică a unui atribut (Statistics)

Pentru fiecare atribut din setul de date, operatorul Statistics prezintă diferite tipuri de informație cu privire la distribuția valorilor (Figura 6.6-13). Pentru toate atributele este inclus un grafic de tip histogramă. Pentru atributele numerice sunt prezentate patru măsuri statistice: valoarea minimă, maximă, media și abaterea standard. Pentru atributele non-metrice este inclus un tabel cu frecvențele absolute și relative (procente). Pentru toate atributele sunt calculați câțiva indicatori utili pentru evaluarea calității datelor. Situațiile problematice sunt semnalate prin colorarea în roșu a indicatorilor și valorilor acestora (culoarea verde indică situația de preferat, date de calitate, iar gri o situație intermediară). Indicatorii sunt:

- **missing**: ponderea cazurilor cu valori lipsă; ne dorim să fie o pondere cât mai mică, nulă de preferat;
- **infinite**: ponderea cazurilor care au valori infinite; ne dorim să fie o pondere cât mai mică, nulă de preferat;
- **ID-ness**: ponderea cazurilor care i-au valori diferite (similar cu un atribut de tip id); ne dorim să fie o pondere cât mai mică, nulă de preferat; se calculează simplu, împărțind numărul de valori diferite la numărul de cazuri;
- **stability**: ponderea cazurilor care i-au aceleași valori; ne dorim să fie o pondere cât mai mică, nulă de preferat; în cazul atributelor non-metrice, cu puține categorii, ponderea poate fi mare; pentru a calcula acest indicator împărțim numărul de cazuri care i-au valoarea cea mai frecventă (cu excepția categoriei valoare lipsă) la numărul total de cazuri valide;
- **valid**: ponderea cazurilor valide; se calculează scăzând din total (100%) scorul la indicatorii missing, infinite, id-ness, stability;
- **text-ness**: atribute care par să fie de tip text.

Figura 6.6-13. Outputul produs de operatorul Statistics



Atributele care au o calitate redusă conform acestor indicatori merită o atenție specială. Funcție de situația concretă, astfel de atribute pot fi eliminate din setul de date, corectate sau transformate. Să presupunem că dorim să importăm în RapidMiner un set de date Excel (sau csv). Acesta include un atribut care într-un singur loc are și alte caractere decât cifre. La importare atributul va fi identificat ca atribut de tip categorial. Dacă vom corecta eroarea înainte de import, atributul va fi identificat corect ca metric. În cazul atributelor cu valori lipsă putem decide să eliminăm atributele care au o pondere foarte mare a cazurilor fără valori, respectiv să înlocuim valorile lipsă atunci când ponderea nu este mare.

Măsuri ale calității datelor (Quality Measures)

Acest operator prezintă într-un format tabelar diferiți indicatori care măsoară calitatea atributelor dintr-un set de date (Tabelul 6.6-1. Outputul operatorului Quality Measures). Cu excepția indicatorului Correlation, toți ceilalți au fost discutați anterior. Correlation se referă la coeficientul de corelație Pearson (ia valori în intervalul [-1-1], unde -1 indică o corelație negativă maximă, 1 corelație pozitivă maximă, iar 0 absența corelației). În acest caz e vorba despre corelația dintre acel atribut și atributul label. De exemplu, corelația dintre părăsirea companiei și vârstă este 0.043.

Tabelul 6.6-1. Outputul operatorului Quality Measures

Attribute	Correlation	ID-ness	Stabillity	Missing	Text-ness
age	0.043	0.276	0.059	0.019	0
tenure (months)	0.008	0.526	0.058	0.006	0
work frustration 1	0.002	0.038	0.256	0	0
work frustration 2	0.002	0.038	0.263	0	0
work frustration 3	0.005	0.038	0.282	0	0
job satisfaction 1	0.046	0.038	0.468	0	0
job satisfaction 2	0.010	0.038	0.397	0	0
job satisfaction 3	0.043	0.038	0.449	0	0
intent to quit	0.040	0.038	0.342	0.006	0

7. Utilitare (Utility)

Fereastra Operators din RapidMiner Studio conține o secțiune numită Utility unde sunt grupați tematic o serie de operatori care au în comun faptul că ușurează lucrul cu seturile de date, respectiv ajută la construirea și rularea proceselor. Acești operatori sunt un fel de unelte care ne fac munca de pregătire și analiză a datelor mai ușoară. Grupările tematice sunt următoarele:

- **Scripting**: scripturi sau sintaxe; ne ajută să rulăm în RapidMiner comenzi din alte programe; desigur, aceste programe trebuie să fie instalate, iar calea spre fișierele executabile ale acestora să fie definită; putem rula comenzi în SQL, Python și R; putem rula propriul cod Java (în cazul în care RapidMiner nu conține comenzile dorite) sau comenzi din alte programe (Windows);
- **Process Control**: aici apar o serie de operatori necesari pentru a gestiona rularea proceselor, grupați în câteva sub-categorii:
    - **Loops**: conține o serie de operatori care permit rularea automată a aceleiași comenzi în cazul unei mulțimi definite de elemente (seturi de date, cazuri, atribute, valori etc.);
    - **Branches**: acești operatori pot rula la un moment dat un singur sub-proces din două sau mai multe posibile, funcție de condițiile definite de utilizator;
    - **Collections**: permit lucrul cu obiecte și colecții de obiecte; de exemplu, putem combina mai multe obiecte într-o colecție de obiecte (Collect) sau putem selecta un obiect dintr-o colecție (Select);
- **Exceptions**: ne ajută să gestionăm situațiile pe care le considerăm a fi o excepție;

- Altele: operatorul **Remember** salvează un anumit obiect (de exemplu, setările parametrilor unui operator) pentru a putea fi folosit ulterior cu ajutorul operatorului **Recall**;
- **Macros**: permite lucrul cu macrocomenzi (macros); folosind acești operatori putem defini macrocomenzi, genera sau extrage macrocomenzi; o macrocomandă reprezintă o modalitate generică de a ne referi la o serie de elemente, o comandă care aplică automat un set de comenzi cu scopul de a realiza o anumită sarcină / activitate;
- **Files**: aceste comenzi ne ajută să lucrăm cu fișiere; putem deschide, scrie, redenumi, copia, muta, șterge un fișier; suplimentar, putem crea un folder, o arhivă, adăuga fișiere la o arhivă;
- **Annotations**: ne ajută să lucrăm cu comentarii relativ la obiecte (să adăugăm comentarii, să le extragem);
- **Logging**: acești operatori aplică diferite comenzi în relație cu fișiere de tip log; fișierele de tip log stochează comenzile realizate și rezultatele obținute în urma rulării unui proces; folosind acești operatori putem scrie într-un fișier rezultatele obținute în urma rulării unei comenzi repetitive sau putem transforma un fișier log într-un alt tip de fișier (ponderări, date);
- **Data Anonymization**: ne ajută să anonimizăm atributele de tip nominal (de exemplu, dacă setul de date conține un atribut cu numele angajaților, putem folosi operatorul Obfuscate pentru a înlocui numele cu alte caractere alese aleator);
- **Random Data Generation**: operatorii incluși au rolul de a genera seturi de date potrivite pentru diferite tipuri de probleme practice precum analiza unei campanii de promovare directă a unui produs (Generate Direct Mailing Data), părăsirea companiei (Generate Churn Data), analiza vânzărilor (Generate Sales Data), analiza tranzacțiilor (Generate Transaction Data); putem genera un set de date care să conțină atribute definite de noi folosind diferite funcții;
- **Misc**: putem trimite un email (Send Mail), amâna rularea unui proces (Delay) etc.;
- Altele: multiplicarea unui set de date (**Multiply**); realizarea unui sub-proces (**Subprocess**), adică a unui proces care conține alte procese;

programarea execuției unui proces la o anumită dată și oră (**Schedule Process**); rularea unui proces (**Execute Process**).

În cele ce urmează vom ilustra utilizarea doar a unora dintre aceste categorii de operatori și doar a unora dintre operatori. Alți operatori din această categorie majoră au fost utilizați în cadrul unora dintre procesele prezentate pe parcursul acestui manual, iar alții vor apărea în procesele prezentate în volumele următoare ale acestui manual.

7.1. Macro-comenzi (Macros)

Un macro (macro-comandă) este o instrucțiune singulară care se extinde automat la un set de instrucțiuni cu scopul de a realiza o acțiune (task) specifică. O macrocomandă reprezintă o modalitate generică de a ne referi la o serie de elemente, o comandă care aplică automat un set de comenzi cu scopul de a realiza o anumită sarcină / activitate. Conceptul de macro este similar cu conceptul de variabilă globală dintr-un limbaj general de programare.

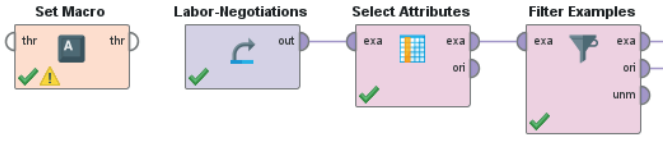
Valorile asociate unui macro sunt inerent de tip string / text dar acestea pot fi evaluate în diferite moduri: ca text, formulă sau nume de atribut. Pentru a defini un macro trebuie să-i dăm un nume și să-i atribuim o valoare (text, număr, formulă, nume atribut), direct sau indirect. Pentru a ne referi la un macro folosim expresia „**%{denumire_macro}**". Atunci când va întâlni un operator în cadrul căruia a fost folosită această expresie, RapidMiner va înlocui expresia cu valoarea asociată acelui macro și doar apoi va rula operatorul respectiv. Dacă folosim expresia „**eval(%{denumire_macro})**", RapidMiner va înțelege că valoarea asociată acelui macro reprezintă o formulă (expresie matematică). Dacă folosim expresia „**#{denumire_macro}**", RapidMiner va înțelege că valoarea asociată acelui macro reprezintă numele unui atribut.

Set Macro

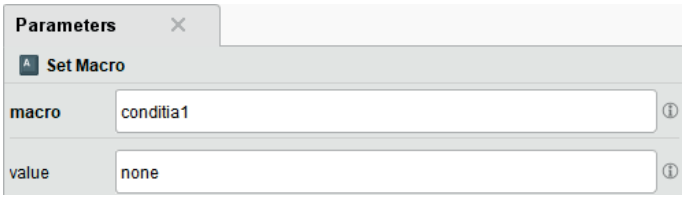
În Figura 7.1-1 am prezentat un exemplu de utilizare a operatorului „Set Macro". Prima dată am definit macro-comanda atribuindu-i un nume (conditia1) și o valoare (none), apoi am făcut referire la acel macro în interiorul operatorului „Filter Examples". Pentru a indica faptul că e vorba de un macro am folosit succesiunea de caractere „%{denumire_macro}", adică „%{condiția1}" în acest caz. Aplicând acest macro am indicat operatorului „Filter Examples" că dorim să reținem doar cazurile care la atributul pension au ales varianta de răspuns none.

Figura 7.1-1. Exemplu de utilizare a operatorului „Set Macro"

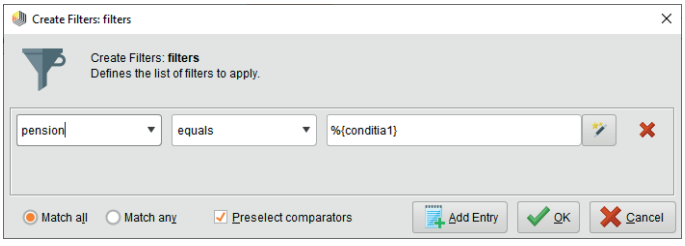
Pasul 1:
Încărcăm setul de date „labor_negociations" și conectăm operatorii precum în imagine.



Pasul 2:
La operatorul „Set Macro" definim un macro: îi atribuim un nume (conditia1) și o valoare (none).



Pasul 3:
La operatorul „Filter Examples" definim un filtru precum în imagine. Atenție la format: %{condiția1}.



Rezultat:
Setul de date rezultat include doar cazurile care la atributul pension au ales varianta none.

Row No.	pension
1	?
2	ret_allw
3	empl_contr
4	?
5	?

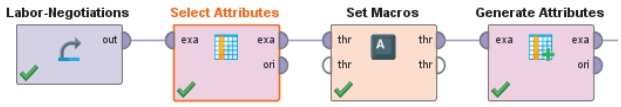
Row No.	pension
1	none
2	none
3	none
4	none
5	none

Set Macros

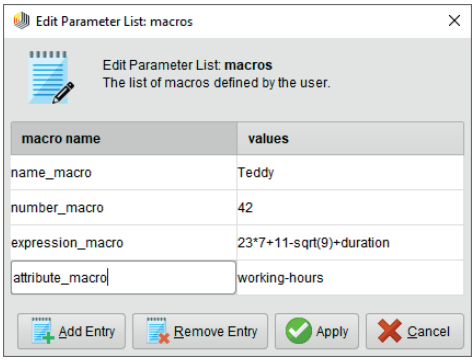
În Figura 7.1-2 am prezentat câteva exemple de generare a unor atribute folosind mai multe macrocomenzi definite simultan cu operatorul „Set Macros”. Prima dată am definit macrocomenzile, apoi le-am folosit pentru a defini expresiile folosite pentru generarea atributelor. De exemplu, am definit un macro pe care l-am denumit „name_macro” și care ia valoarea Teddy. În continuare, atunci când vom folosi denumirea acestui macro, softul va înțelege că e vorba de valoarea Teddy și va realiza acțiunea indicată (de exemplu, va căuta această valoare, o va completa, va reține doar cazurile care au această valoare etc.). În cazul de față, am folosit acest macro pentru a genera un atribut care va lua valoarea Teddy pentru toate cazurile (lower e pentru a scrie textul cu litere mici). Similar, am definit un alt macro, denumit „attribute_macro”, care ia valoarea atributului „working-hours”. Apoi am generat un atribut denumit „macro with attribute”. Valorile acestui atribut vor fi egale cu suma dintre valoarea înregistrată la atributul „working-hours” și 5.

Figura 7.1-2. Exemplu de utilizare a operatorului „Set Macros”

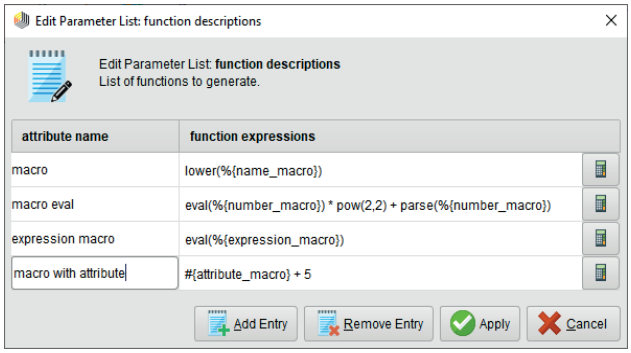
Pasul 1:
Încărcăm setul de date „labor_negociations” și conectăm operatorii precum în imagine.



Pasul 2:
La operatorul „Set Macros” definim lista de macrocomenzi, adică atribuim un nume, respectiv indicăm formula / valoarea pentru fiecare macro.



Pasul 3:
La operatorul „Generate Attributes” denumim atributul nou apoi indicăm funcția. Includem în definirea funcțiilor numele uneia dintre macrocomenzile definite anterior. Atenție la format: # sau %{nume_macro}, eval(%{nume_macro}).



Rezultat:
Setul de date rezultat include cele patru atribute noi.

class	duration	working-...	macro	macro eval	expression macro	macro with attribute
good	1	40	teddy	210	170	45
good	2	35	teddy	210	171	40
good	?	38	teddy	210	?	43
good	3	?	teddy	210	172	?

Extract Macro

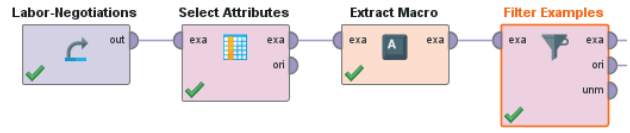
Pentru a extrage un macro folosim operatorul „Extract Macro”. Putem extrage patru tipuri de macro:

- **number_of_examples**: valoarea asociată macrocomenzii va fi egală cu numărul de cazuri din setul de date;
- **number_of_attributes**: valoarea asociată macrocomenzii va fi egală cu numărul de atribute din setul de date;
- **data_value**: valoarea asociată macrocomenzii va fi egală cu valoarea luată de atributul și cazul indicate;
- **statistics**: valoarea asociată macrocomenzii va fi egală cu valoarea indicată de măsura statistică selectată relativ la un anumit atribut.

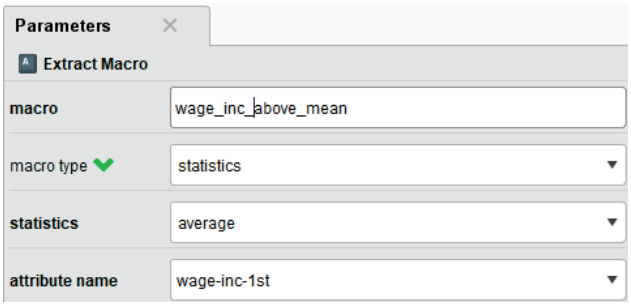
În exemplul din Figura 7.1-3 am selectat doar cazurile care au o valoare mai mică decât media în cazul atributului wage-inc-1st. Inițial, setul de date conținea 40 de cazuri, media atributului wage-inc-1st era 3.6 iar valoarea maximă 6.9. După aplicarea macrocomenzii setul de date conține 21 de cazuri, media atributului wage-inc-1st este 2.6 iar valoarea maximă 3.5.

Figura 7.1-3. Exemplu de utilizare a operatorului „Extract Macro"

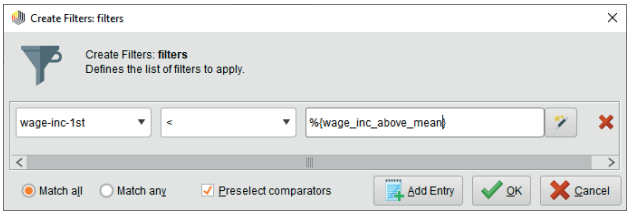
Pasul 1:
Încărcăm setul de date „labor_negociations" și conectăm operatorii precum în imagine.



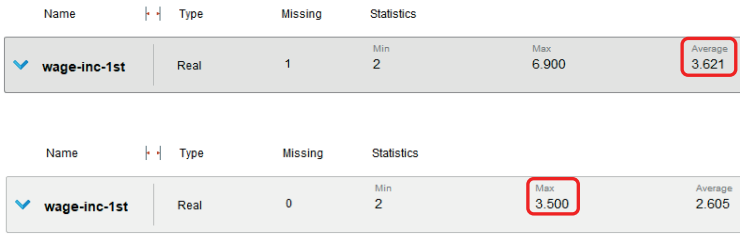
Pasul 2:
La operatorul „Extract Macro" alegem tipul de macro numit statistics. La parametrul macro dăm un nume, la statistics alegem average, iar la „attribute name" numele atributului. Valoarea macro-ului va fi egală cu media atributului wage-inc-1st.



Pasul 3:
La operatorul „Filter Examples" definim relația. Vor fi selectate cazurile care la atributul wage-inc-1st au o valoarea mai mică decât valoarea setată prin macro.



Rezultat:
Setul de date rezultat vla include doar cazurile cu valori mai mici decât media.



7.2. Comenzi repetitive (Loops)

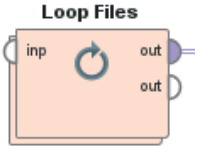
Categoria Loops conține o serie de operatori care permit rularea automată a aceleiași comenzi în cazul unei mulțimi definite de elemente. Elementele pot fi oricare dintre următoarele: fișiere, cazuri, atribute, valori, parametri, etichete etc. De fiecare dată când dorim să aplicăm aceleași comenzi de mai multe ori în același timp, comenzile Loops sunt extrem de utile.

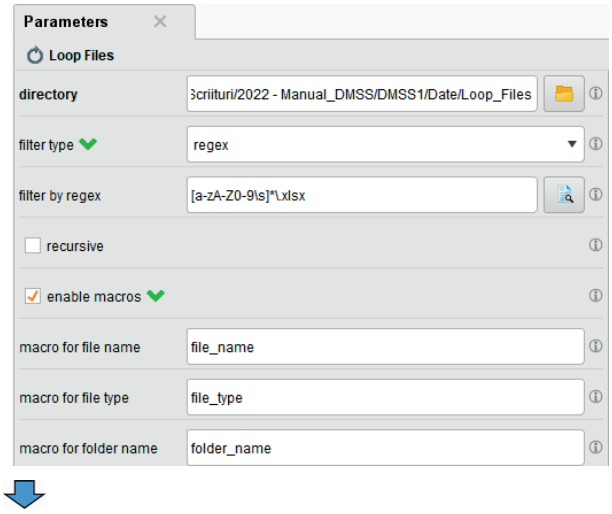
Comenzi repetitive cu fișiere (Loop Files)

Să presupunem că dorim să importăm și salvăm în format RapidMiner mai multe fișiere Excel (sau de alt tip). Putem face acest lucru simplu, cu o singură comandă ceva mai complexă, „Loop Files" (Figura 7.2-1).[57] Pentru a putea identifica toate fișierele de tip Excel dintr-un folder trebuie să generăm un macro, respectiv să definim tipul de fișiere cu ajutorul regex.

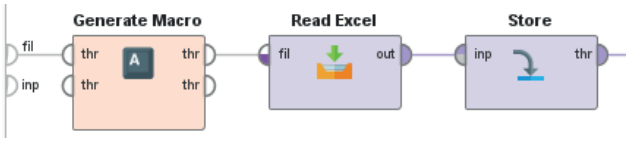
Figura 7.2-1. Importarea și salvarea mai multor fișiere Excel (Loop Files)

Pasul 1:
Încărcăm operatorul „Loop Files". Observăm că este un operator „nested" = putem include alți operatori în el. La Parameters indicăm folderul, expresia regex (semnifică orice fișier cu extensia xlsx), informațiile relativ la macrocomandă.

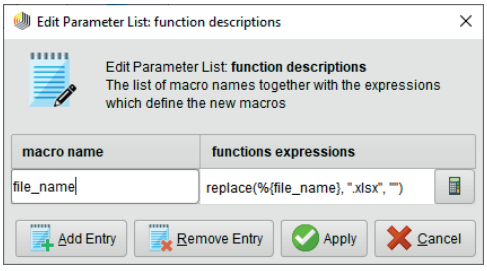




Pasul 2:
În interiorul operatorului „Loop Files" includem operatorii din imaginea alăturată.

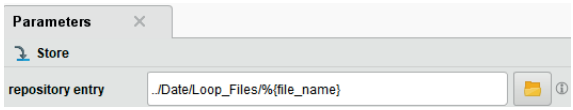


Pasul 3:
La operatorul „Generate Macro" definim macro-comanda din imagine (elimină extensia xlsx din denumirea fișierelor înainte de a le salva ca tabele RapidMiner).
La operatorul „Read Excel" lăsăm valorile implicite ale parametrilor.

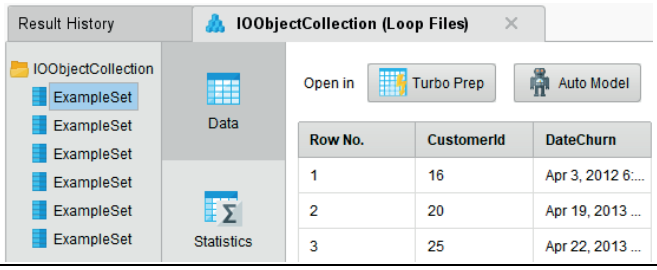


[57] Prezentare video: https://academy.rapidminer.com/learn/video/extracting-data-demo.

La operatorul „Store” indicăm folderul în care dorim să salvăm fișierele. În loc de numele fișierelor vom trece denumirea macrocomenzii.



Rezultat:
Observăm că toate cele șase fișiere Excel au fost importate și salvate ca fișiere RapidMiner de tip tabel (set de date).

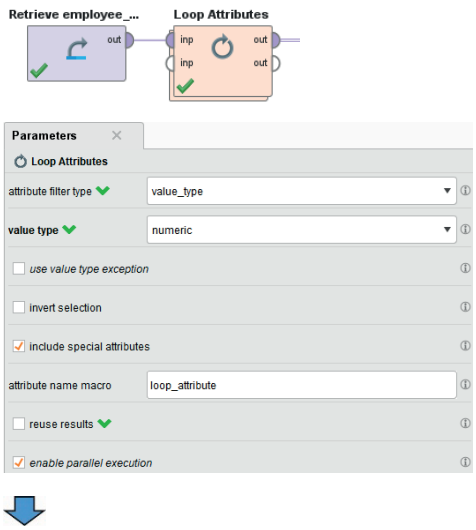


Comenzi repetitive cu atribute (Loop Attributes)

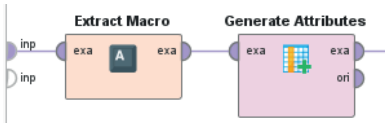
Să presupunem că dorim să calculăm diferența dintre valorile unui atribut și media acelui atribut pentru toate atributele numerice dintr-un set de date. Putem face acest lucru folosind o singură comandă, „Loop Attributes” (Figura 7.2-2).

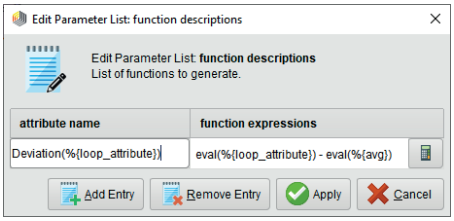
Figura 7.2-2. Generarea mai multor atribute simultan (Loop Attributes) (1)

Pasul 1:
Încărcăm setul de date și îl conectăm la operatorul „Loop Attributes”. Valorile parametrilor sunt cele din imagine. Indicăm faptul că dorim să selectăm toate atributele de tip numeric și că numele macrocomenzii este loop_attribute.

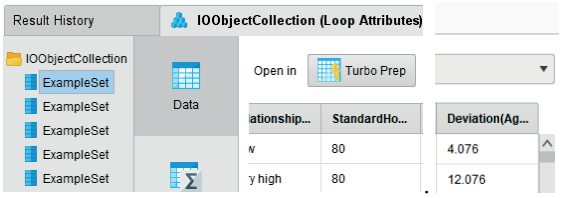


Pasul 2:
În interiorul operatorului „Loop Attributes” includem operatorii din imaginea alăturată.
Setările operatorului „Extract Macro” indică faptul că dorim să extragem media atributelor.
Setările operatorului „Generate Attributes” indică faptul că dorim să generăm o serie de atribute, fiecare nou atribut fiind calculat ca diferență între vechiul atribut și media vechiului atribut.





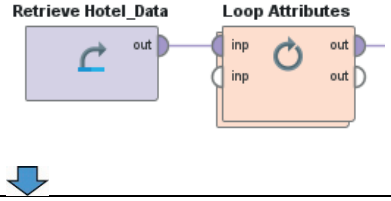
Rezultat:
Observăm că a fost generată o colecție de tabele, fiecare tabel având la final unul dintre noile atribute generate.



Pentru următorul exemplu (Figura 7.2-3), folosim tot operatorul „Loop Attributes”, de această dată pentru a genera trei atribute care măsoară ponderea tranzacțiilor realizate prin intermediul fiecărui canal de plată.[58]

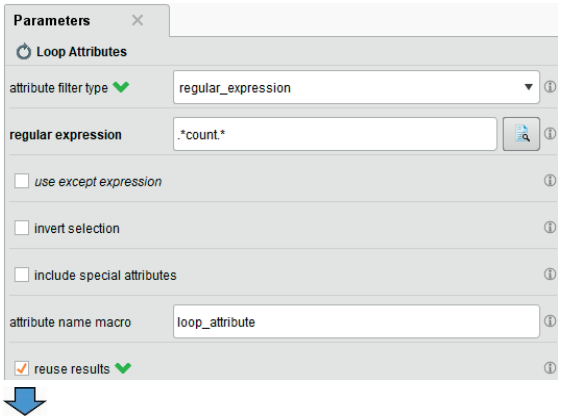
Figura 7.2-3. Generarea mai multor atribute simultan (Loop Attributes) (2)

Pasul 1:
Încărcăm setul de date „Hotel_Data” și conectăm operatorul „Loop Attributes”.

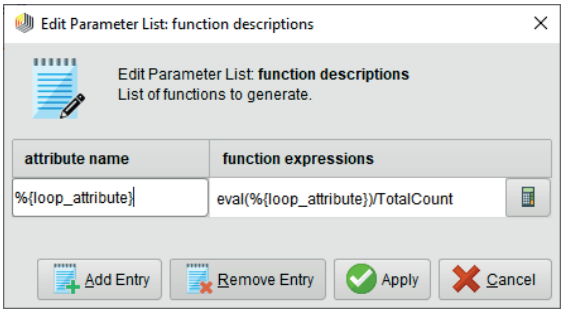


[58] Prezentare video: https://academy.rapidminer.com/learn/video/loop-attributes.

Pasul 2:
La operatorul „Loop Attributes" setările sunt cele din imaginea alăturată.
Selectăm doar atributele care au în denumire cuvântul count.
Numele macro este „loop_attribute".
Selectăm parametrul „reuse_attribute" pentru a include toate atributele noi în același set de date.



Pasul 3:
În interiorul operatorului „Loop Attributes" includem operatorul „Generate Attributes". Indicăm numele și funcția asociate atributelor care vor fi generate. Pentru a indica toate numele, folosim macrocomanda definită anterior (loop_attribute). La „function expressions" denumirea macro apare ca eval(%{denumire}) pentru a indica faptul că e vorba de o expresie matematică.



Rezultat:
Observăm că a fost generat un tabel care conține la final noile atribute generate.

TotalCount	countPaymentMethod_creditcard	countPaymentMethod_cheque	countPaymentMethod_cash
1	0	0	1
3	0	0.333	0.667
7	0.857	0	0.143
2	0.500	0	0.500

7.3. Alte comenzi utilitare

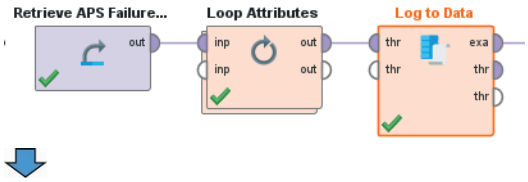
Fișiere de tip log (Log)

Execuția unui proces, a unei componente a acestuia sau orice altă informația relativ la unul sau mai mulți operatori (cum ar fi setările parametrilor) poate fi înregistrată într-un log. Informația respectivă poate fi afișată în output (perspectiva Results) sau poate fi salvată ca fișier de tip log, date sau

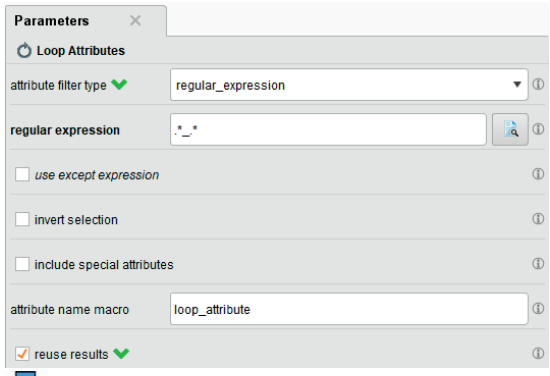
ponderări (weights). Pentru a înregistra astfel de date folosim operatorul Log. Pentru a transforma informațiile colectate în log într-un set de date folosim operatorul „Log to Data”. Exemplul din Figura 7.3-1 prezintă un astfel de exemplu.

Figura 7.3-1. Înregistrarea unor informații relativ la un proces (Log)

Pasul 1:
Încărcăm setul de date „Hotel_Data” și conectăm operatorul „Loop Attributes”.



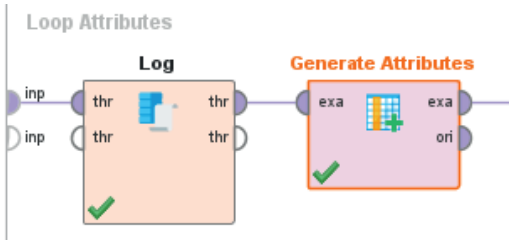
Pasul 2:
La operatorul „Loop Attributes” setările sunt cele din imaginea alăturată.
Selectăm doar atributele care au în denumire caracterul „_”.
Numele macro este „loop_attribute”.
Selectăm parametrul „reuse_attribute” pentru a include toate atributele noi în același set de date.

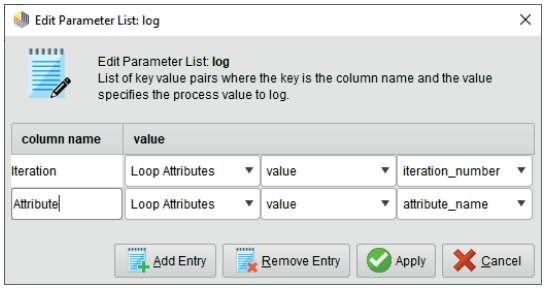


Pasul 3:
În interiorul operatorului „Loop Attributes” includem operatorii Log și „Generate Attributes”.

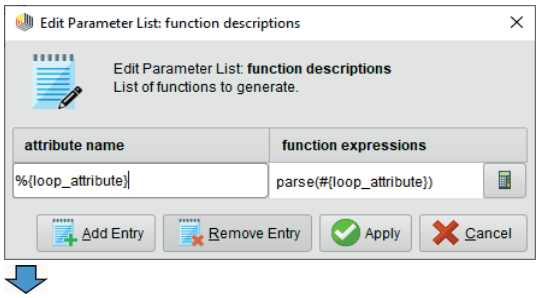
La log definim cele două tipuri de informații pe care dorim să le salvăm.

La „Generate Attributes” indicăm numele și funcția asociate atributelor care vor fi generate. Pentru a indica toate numele, folosim macrocomanda definită anterior (loop_attribute). La „function expressions” denumirea macro apare ca #{denumire} pentru a indica faptul că e vorba de numele unui atribut.

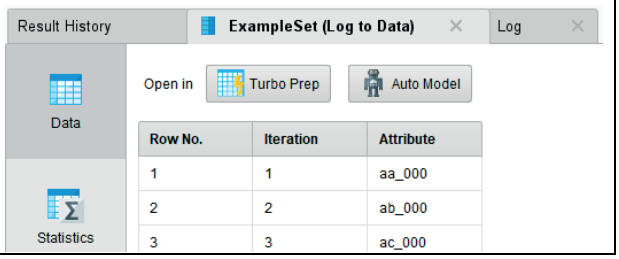




Comanda parse e necesară pentru a schimba tipul atributelor din nominal în numeric.



Rezultat:
Observăm că aceleași informațiile cerute (iterația și atributul) au fost salvate într-un tabel de date, respectiv apar în tabul log (salvat și el ca fișier log).



Generarea unor tabele de date

Folosind operatorii din categoria „Utility/Random Data Generation" putem genera diferite tipuri de date predefinite, respectiv definite de noi prin indicarea numărului de atribute dorit și a funcțiilor asociate acestora. În Figura 7.3-2 am prezentat câteva exemple de tabele generate cu astfel de operatori.

Figura 7.3-2. Generarea unor seturi de date

Generate Data:
Indicăm numărul dorit de cazuri, atribute, intervalul de variație al atributelor, funcția de generare (random, sum, polynomial etc.).

label	att1	att2	att3	att4	att5
0.961	2.426	5.258	3.629	-8.776	4.707
0.721	-0.234	8.681	-7.473	6.816	7.454
0.304	2.640	0.314	-4.380	1.370	8.331

Generate Direct Mailing Data:
Indicăm doar numărul dorit de cazuri.

label	name	age	lifestyle	zip code	family status	car	sports	earnings
response	d9vUhHlf	63	cozily	35659	single	practical	badminton	88907
no response	SnRGk9Ot	24	cozily	55761	married	expensive	badminton	78024
no response	14jVO8M7	21	cozily	66267	single	practical	soccer	96359
no response	9um5M1HG	33	healthy	89727	married	expensive	soccer	36671

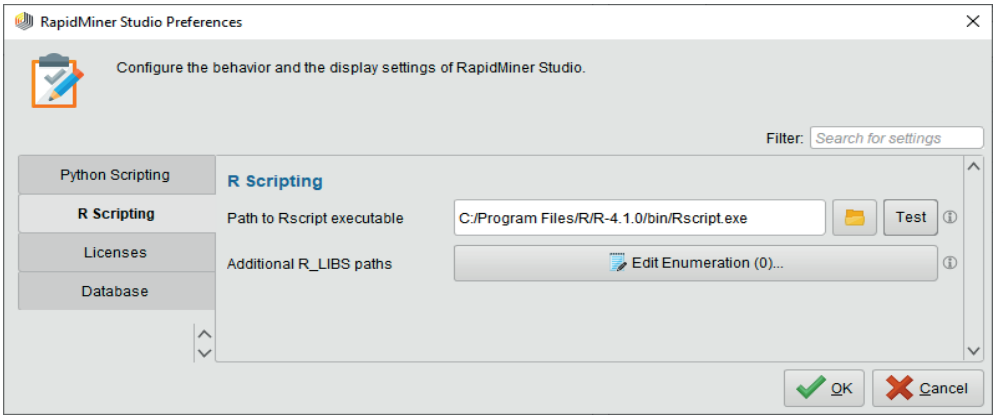
Generate Churn Data:
Indicăm doar numărul dorit de cazuri.

label	Year 1	Year 2	Year 3	Year 4	Year 5
terminate	Collect Infor...	New Credit	Additional ...	Collect Info...	Nothing
terminate	Additional C...	New Credit	Additional ...	Collect Info...	Nothing
ok	Additional C...	Collect Infor...	End Credit	Collect Info...	End Credit
ok	New Credit	End Credit	New Credit	Collect Info...	Nothing

Rularea unui script R în RapidMiner Studio (Execute R)

În cazul în care dorim să rulăm anumite comenzi în R din RapidMiner Studio putem folosi operatorul „Execute R". Firesc, pentru a putea utiliza acest operator trebuie să avem instalat softul R și să setăm în RapidMiner calea spre fișierul executabil Rscript. În meniul RapidMiner, la Settings/ Preferences, alegem R Scripting, indicăm calea, apoi o testăm și acceptăm (Figura 7.3-3).

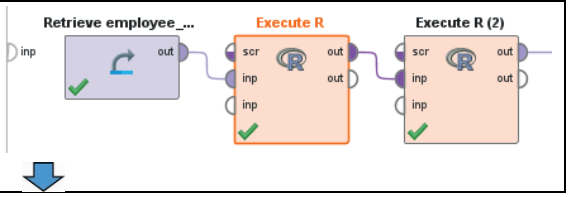
Figura 7.3-3. Setarea locației fișierului executabil Rscript



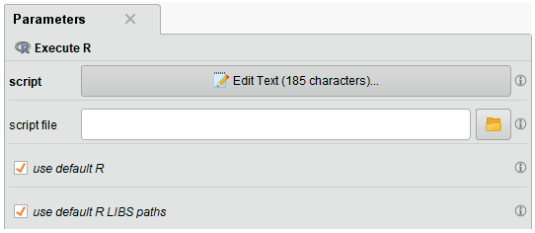
În exemplul din Figura 7.3-4 am ilustrat rularea a două scripturi R din RapidMiner în R (nu e nevoie să deschidem R, scriptul va rula în background). Prima dată am exportat un set de date din RapidMiner în R și apoi l-am citit din nou în RapidMiner (câteva atribute doar).

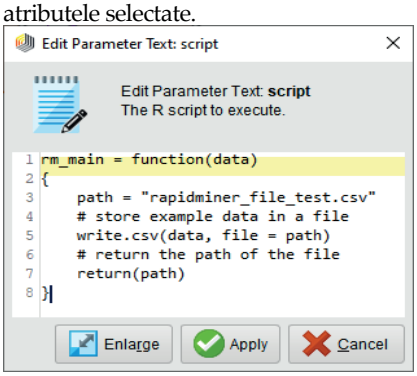
Figura 7.3-4. Rularea unui script R în RapidMiner Studio (Execute R)

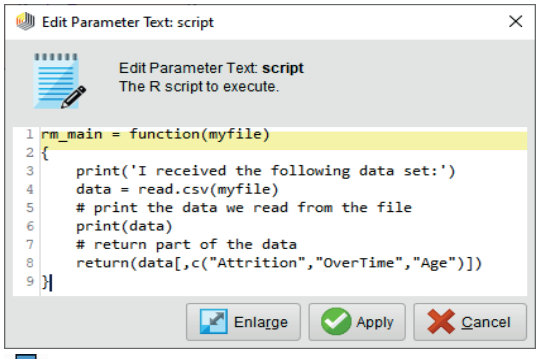
Pasul 1:
Încărcăm setul de date și conectăm operatorul „Execute R".



Pasul 2:
La operatorul „Execute R" alegem parametrul script și introducem liniile de comandă R. Comenzile sunt puse între acolade. Scriptul 1 exportă datele din RapidMiner în R, le scrie în format csv, iar scriptul 2 citește fișierul csv și returnează în RapidMiner tabelul cu atributele selectate.







Rezultat:
Setul de date conține doar atributele selectate.

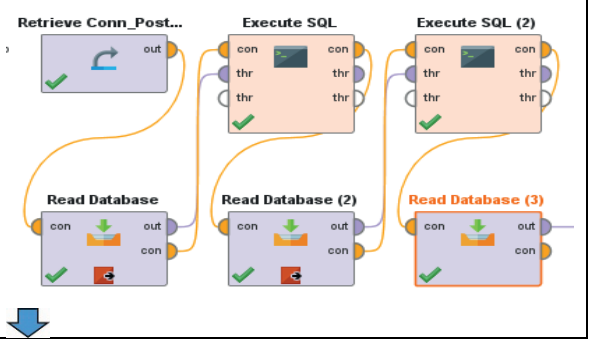
Row No.	Attrition	OverTime	Age
1	Yes	Yes	41
2	No	No	49
3	Yes	Yes	37

Rularea unui script SQL în RapidMiner Studio (Execute SQL)

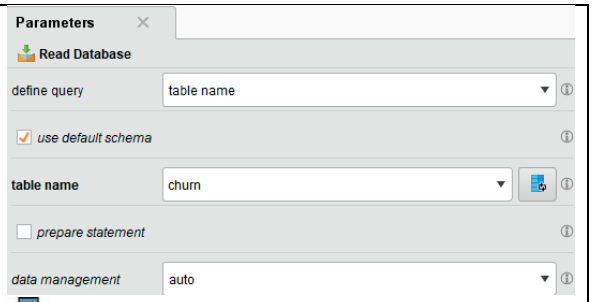
În Figura 7.3-5 am ilustrat rularea a două scripturi SQL în RapidMiner. Prima dată ne-am conectat la o bază de date online (conexiunea trebuie definită anterior), am șters toate cazurile din tabelul churn și apoi am introdus trei cazuri noi. La final observăm că tabelul churn conține doar cele trei cazuri.

Figura 7.3-5. Rularea unui script SQL în RapidMiner Studio (Execute SQL)

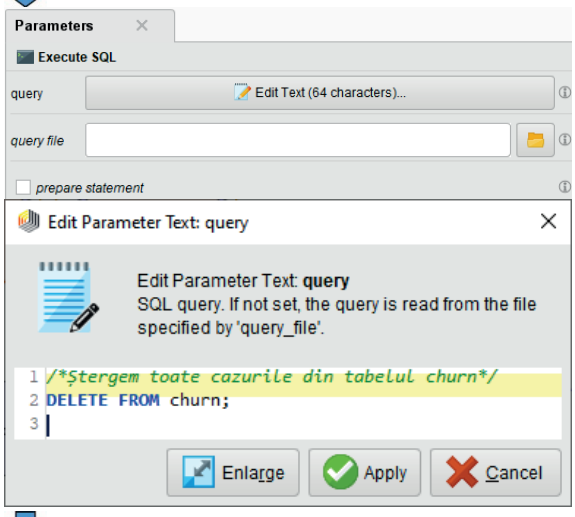
Pasul 1:
Realizăm procesul din imaginea alăturată. Folosim conexiunea PostgreSQL inclusă în folderul cu procese aferent acestui volum.



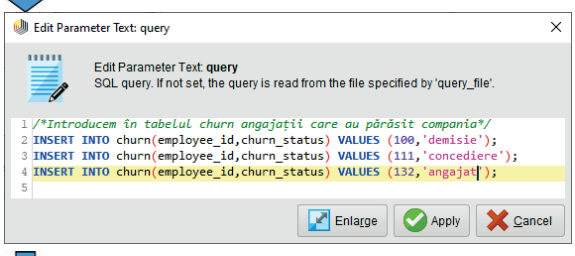
Pasul 2:
Setările operatorului „Read Database" indică faptul că dorim să citim tabelul churn.



Pasul 3:
La operatorul „Execute SQL" scriem comanda SQL „DELETE FROM churn;". Această comandă șterge toate cazurile din tabelul churn.
Observăm că setul de date afișat nu conține niciun caz.



Pasul 4:
La operatorul „Execute SQL (2)" scriem comenzile SQL din imagine. Introducem astfel trei cazuri în tabelul churn.



Rezultat:
Setul de date afișat (tabelul churn) conține cele trei cazuri introduse cu ajutorul scriptului SQL.

Row No.	employee_id	churn_status
1	100	da
2	111	da
3	132	da

8. Pregătirea asistată a datelor (Turbo Prep)

Perspectiva Turbo Prep oferă o modalitate simplă, rapidă și intuitivă pentru a pregăti un set de date pentru analiză.[59] Toți pașii necesari și posibili în cadrul unui proces de pregătire a unui set de date sunt sugerați, utilizatorul având posibilitatea de alege operatorii potriviți și seta rapid parametrii acestora. Pașii de urmat pot fi structurați astfel:

- încărcarea unui set de date (**Load Data**): putem accesa ultimele seturi de date folosite, încărca un set din Repository, importa un set de date;
- alegerea uneia dintre categoriile de comenzi **Transform**, **Cleanse**, **Generate**, **Pivot** sau **Merge**; pot fi alese mai multe categorii, pe rând; categoriile sunt active doar după ce am încărcat un set de date;
- alegerea comenzilor (una sau mai multe) din categoria selectată; aplicarea comenzii; comenzile sunt active doar după ce am ales anterior o categorie de comenzi;
- comanda finală (**Commit**); alegând această comandă indicăm faptul că dorim să aplicăm definitiv toate comenzile performate;
- salvarea tuturor comenzilor performate într-un proces.

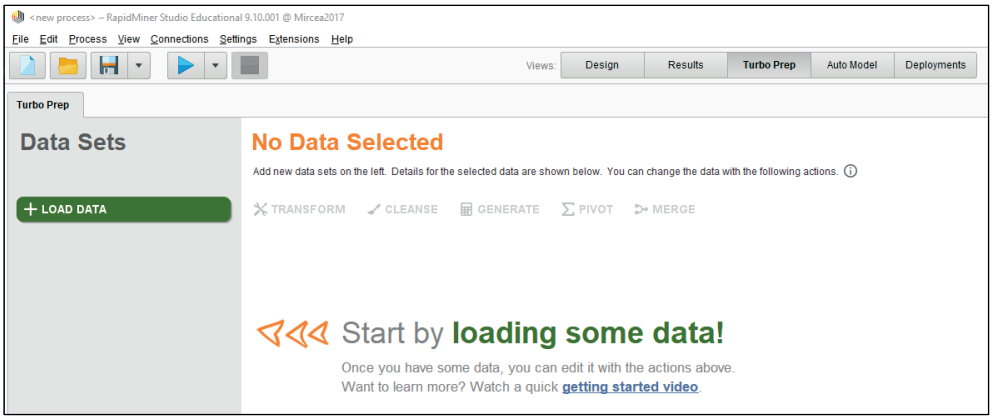
8.1. Turbo Prep: Încărcarea și inspectarea unui set de date (Load Data)

Prima opțiune oferită de perspectiva Turbo Prep este încărcarea unui set de date Figura 8.1-1. Suntem avertizați că nu am selectat niciun set de date și că

[59] Pentru un exemplu extins pot fi consultate textul lui Ingo Mierswa, „Data prep and machine learning made fun, fast and simple" (https://rapidminer.com/blog/data-prep-machine-learning/) și prezentarea video pe aceeași temă „Intuitive Data Prep for Machine Learning" (https://rapidminer.com/resource/data-prep-machine-learning/).

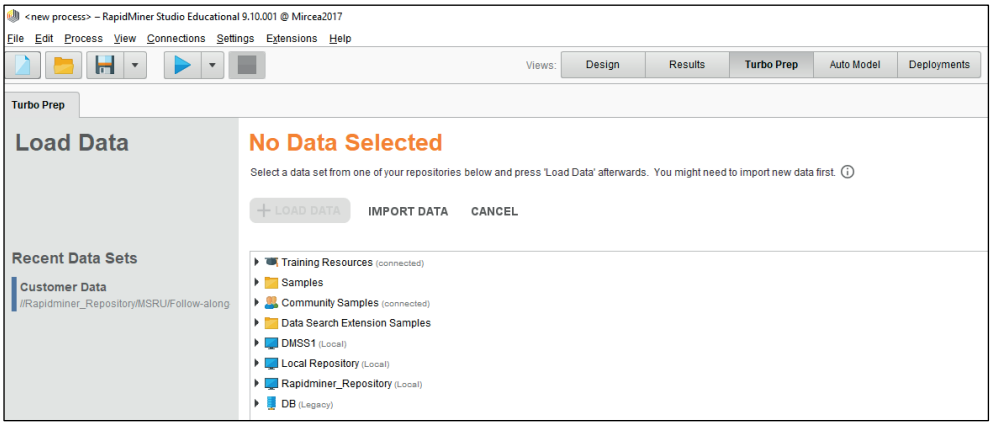
trebuie să încărcăm unul. Dacă nu știm cum să facem acest lucru putem urmări un scurt video în care ne sunt prezentați pașii de urmat (linkul apare pe pagina respectivă, direct în soft). Categoriile de comenzi disponibile nu sunt deocamdată active. Ele vor deveni active doar după încărcarea unui set de date.

Figura 8.1-1. Perspectiva Turbo Prep la start



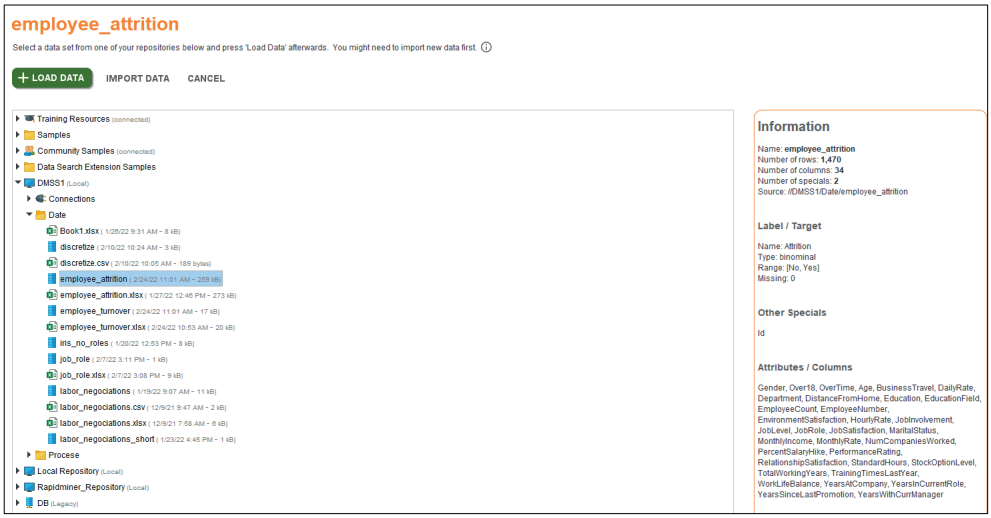
Putem încărca un set de date folosind una dintre opțiunile disponibile (Figura 8.1-2). Astfel, putem să alegem un set de date care a fost utilizat recent alegând setul dorit dintre cele care apar în fereastra din stânga (Recent Data Sets). Alternativ, putem importa un set de date (Import Data) sau alege unul inclus în folderele și depozitele de date disponibile.

Figura 8.1-2. Încărcarea unui set de date în perspectiva Turbo Prep (1)



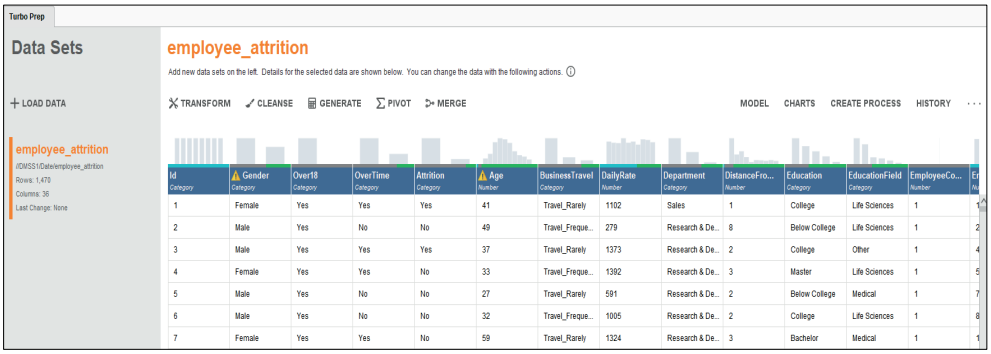
În cazul de față am ales un set de date (employee_attrition) din depozitul de date aferent acestui manual. După selectarea acestuia, comanda „Load Data" devine activă, iar în dreapta apare o fereastră în care sunt prezentate câteva informații relativ la acest set de date (Figura 8.1-3). Informațiile sunt următoarele: numele setului de date, numărul de cazuri, numărul de atribute, numărul de atribute speciale, descrierea variabilei label / target (nume, nivel de măsurare, interval de variație, valori lipsă), denumirea celorlalte variabile speciale, respectiv a restului atributelor. Pentru a încărca setul de date ales, apăsăm butonul „Load Data".

Figura 8.1-3. Încărcarea unui set de date în perspectiva Turbo Prep (2)



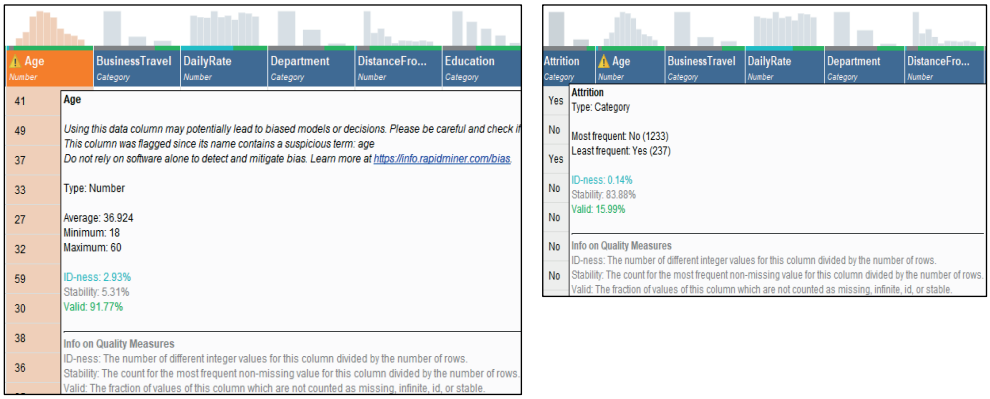
După apăsarea butonului „Load Data", setul de date este afișat, sunt oferite câteva informații despre acesta (locație, număr cazuri și atribute, ultima modificare), devin active categoriile de opțiuni disponibile pentru pregătirea setului de date și o serie de alte opțiuni (Model, Charts, Create Process, History) (Figura 8.1-4). Pentru fiecare atribut este specificat numele, nivelul de măsurare (category / number), respectiv este afișat un mic grafic de tip histogramă. Culorile care apar deasupra denumirii fiecărui atribut indică prezența (roșu) sau absența (verde) unor probleme relativ la calitatea datelor aferente acelui atribut (gri reprezintă o situație intermediară). Pentru fiecare atribut pot să apară una mai multe culori, una pentru fiecare dintre cele cinci tipuri de probleme.

Figura 8.1-4. Afișarea unui set de date în perspectiva Turbo Prep



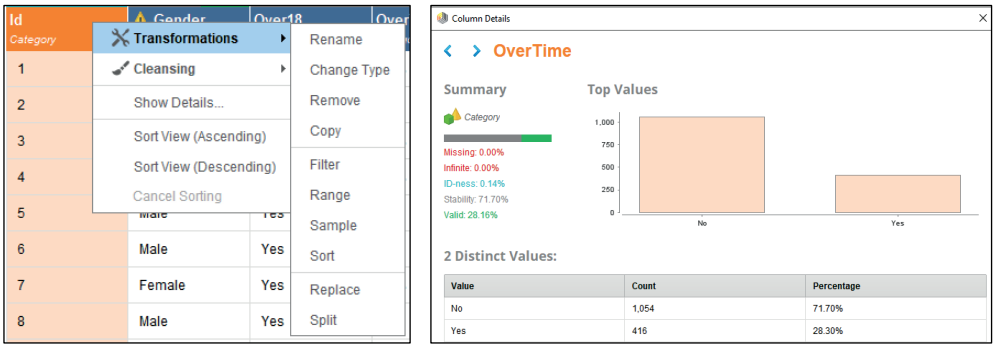
La trecerea cu cursorul peste numele unui atribut sunt afișate o serie de informații utile relativ la acesta: nivelul de măsurare, diferite statistici (funcție de nivelul de măsurare) și indicatorii de calitate (ID-ness, Stability, Valid) (Figura 8.1-5).

Figura 8.1-5. Informații relativ la atribute în perspectiva Turbo Prep



Dacă dăm click dreapta pe un atribut, ne sunt afișate o serie de comenzi, grupate pe două categorii: Transformations și Cleansing (Figura 8.1-6). La opțiunea „Show Details" apar o serie de informații despre acel atribut: indicatorii de calitate, informații statistice (tabel de frecvențe pentru atributele categoriale, respectiv media, abaterea standard, valoarea minimă, valoarea maximă) și distribuția valorilor sub formă de histogramă.

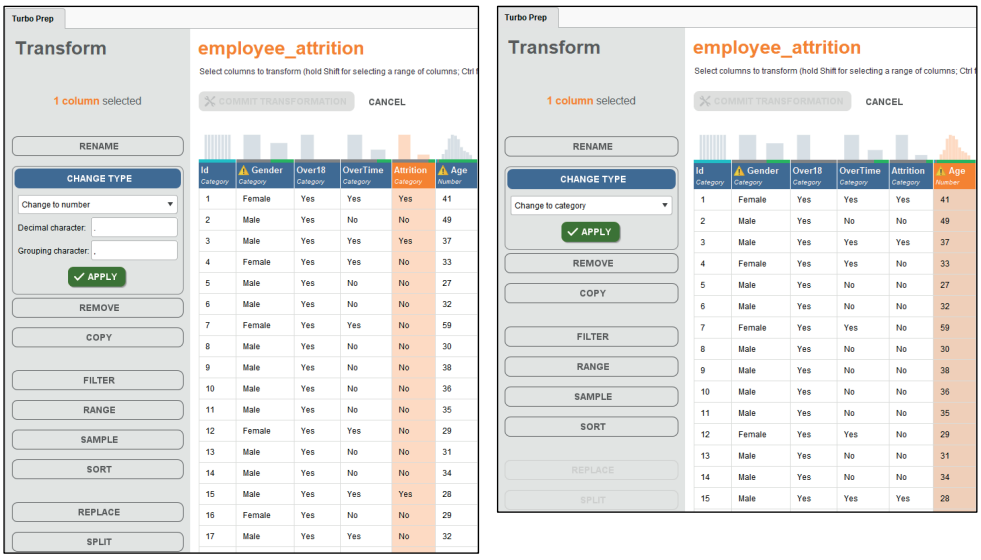
Figura 8.1-6. Acțiuni și informații relativ la atribute în perspectiva Turbo Prep



8.2. Turbo Prep: Transformarea datelor (Transform)

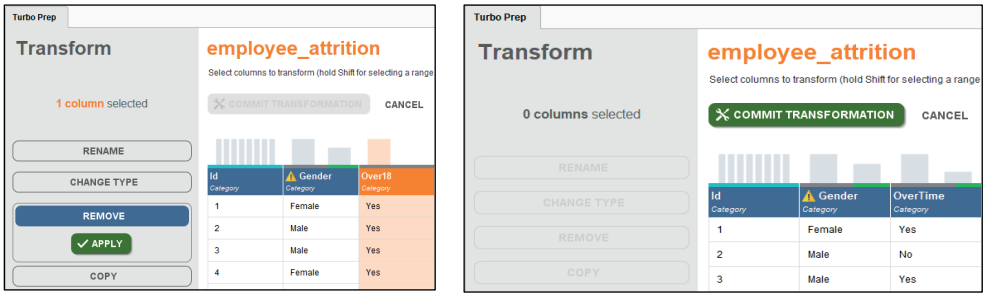
Dacă selectăm categoria de comenzi Transform, RapidMiner afișează în stânga comenzile din acea categorie. Dacă selectăm un atribut, vor fi active doar comenzile care pot fi aplicate în cazul acelui atribut. De exemplu, în Figura 8.2-1, în cazul atributului numeric Age comanda Split nu este activă (această comandă poate fi aplicată doar în cazul atributelor de tip categorial).

Figura 8.2-1. Turbo Prep: Transform



Pentru a ilustra pașii de aplicare a unei comenzi am selectat atributul Over18 (Figura 8.2-2). Acest atribut este constant (toți angajații au cel puțin 18 ani). Deoarece atributul nu conține informație utilă pentru analiză dorim să-l eliminăm din setul de date. Pentru aceasta, selectăm comanda Remove, apoi aplicăm comanda selectată cu Apply (observăm că atributul nu mai este afișat), iar la final alegem „Commit Transformation" (sau Cancel dacă dorim să renunțăm) pentru a elimina efectiv atributul. Desigur, putem alege oricâte comenzi, relativ la oricâte atribute și doar apoi să implementăm efectiv toate comenzile definite („Commit Transformation"). După fiecare comandă Commit, informațiile relativ la setul de date sunt actualizate. De exemplu, în acest caz, setul de date are mai puțin cu o variabilă, iar la ultima transformare e menționat faptul că am eliminat atributul Over18.

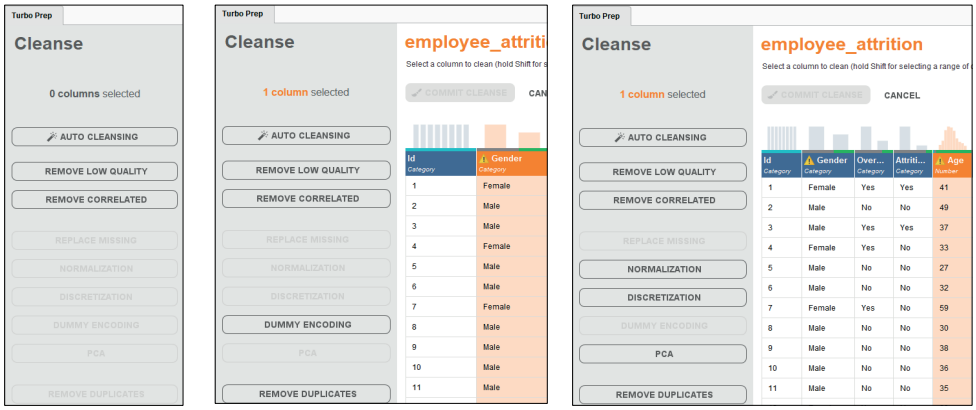
Figura 8.2-2. Turbo Prep: Transform (aplicarea unei comenzi)



8.3. Turbo Prep: „Curățarea" datelor (Cleanse)

Categoria Cleanse conține o serie de comenzi utile pentru „curățarea" setului de date. „Curățarea" poate lua forme precum eliminarea unor atribute, transformarea unor atribute și eliminarea cazurilor duplicate (Figura 8.3-1). Concret, putem elimina atributele cu indicatori reduși ai calității sau pe cele care au corelații mari, înlocui valorile lipsă, normaliza și discretiza atributele numerice, recodifica un atribut cu multe categorii în mai multe atribute binare, condensa informația din mai multe atribute în mai puține (PCA), elimina cazurile identice. După aplicare unor comenzi e nevoie să le și implementăm efectiv (butonul „Commit Cleanse").

Figura 8.3-1. Turbo Prep: Cleanse



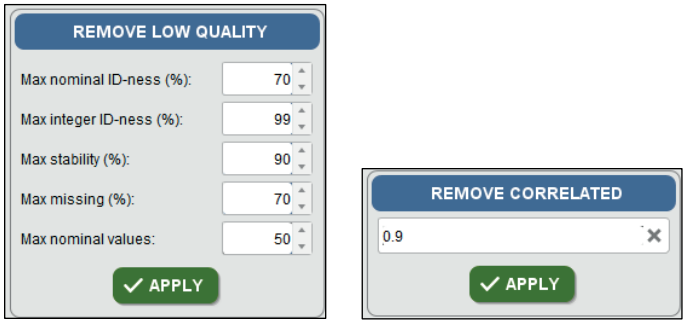
Pentru a elimina automat toate atributele care corelează puternic între ele folosim opțiunea „Remove Correlated" (Figura 8.3-2). Putem seta pragul dorit al coeficientul de corelație (valoarea e de tip absolut). Comanda calculează coeficientul de corelație pentru toate perechile de atribute. Dacă două atribute au o corelație mai mare decât pragul ales, unul dintre atribute va fi eliminat (care dintre acestea depinde de mărimea corelațiilor cu restul atributelor).

Pentru a elimina automat toate atributele care nu îndeplinesc anumite criterii de calitate putem alege pragurile aferente acestor criterii folosind opțiunea „Remove Low Quality" (Figura 8.3-2). Criteriile posibile sunt:

- **Max nominal ID-ness (%)**: elimină atributele nominale care arată precum un atribut de tip Id, adică au foarte multe categorii diferite de răspuns; indicatorul de calitate se calculează prin împărțirea numărului de valori diferite la numărul de cazuri; sunt eliminate atributele care depășesc pragul ales;
- **Max integer ID-ness (%)**: elimină atributele numerice care arată precum un atribut de tip Id, adică au foarte multe valori diferite de răspuns; indicatorul de calitate se calculează prin împărțirea numărului de valori diferite la numărul de cazuri; sunt eliminate atributele care depășesc pragul ales;
- **Max stability (%)**: elimină atributele care au la o singură categorie / variantă de răspuns o proporție a cazurilor mai mare decât pragul ales;
- **Max missing (%)**: elimină atributele care au o pondere a valorilor lipsă mai mare decât pragul ales;

– **Max nominal values**: elimină atributele nominale care au un număr de categorii mai mare decât pragul ales.

Figura 8.3-2. Turbo Prep: Cleanse – Remove Low Quality & Remove Correlated



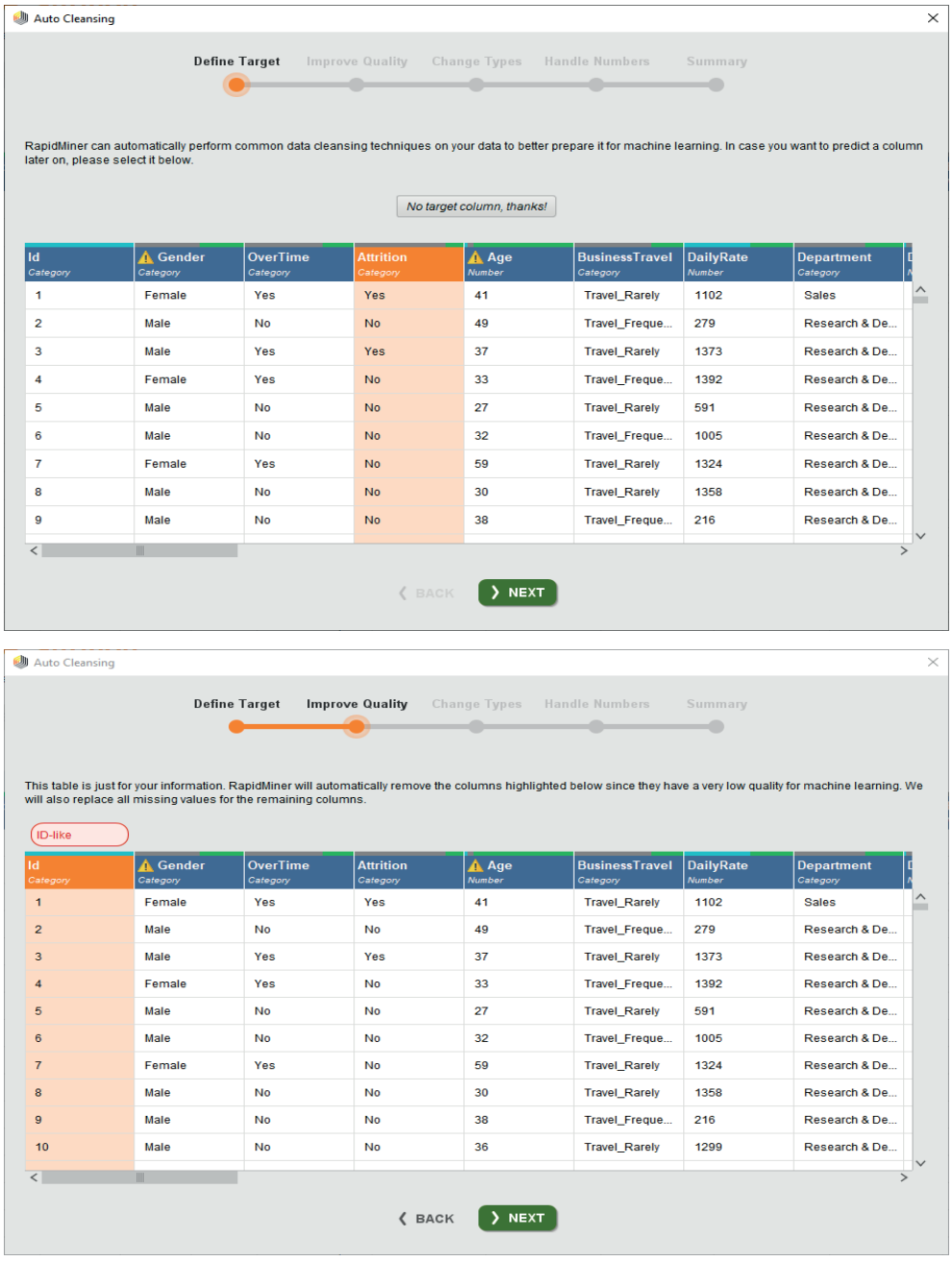
RapidMiner oferă și posibilitatea „curățării" automate a datelor. Comanda „Auto Cleansing" permite realizarea acestei acțiuni ținând sau nu cont și de atributul label (column target) (Figura 8.3-3). Dacă alegem să definim un atribut ca label (target), procesul de curățare va lua în calcul suplimentar și relația dintre acest atribut și celelalte, deci va elimina atributele care nu sunt utile pentru predicția atributului label. În cazul de față am indicat faptul că Attrition este atributul pe care dorim să-l prezicem (pasul „Define Target"). La pasul următor, „Improve Quality", RapidMiner înlocuiește automat valorile lipsă, apoi identifică și elimină atributele prea puțin utile pentru predicția atributului Attrition. În cazul de față e vorba de atributele:

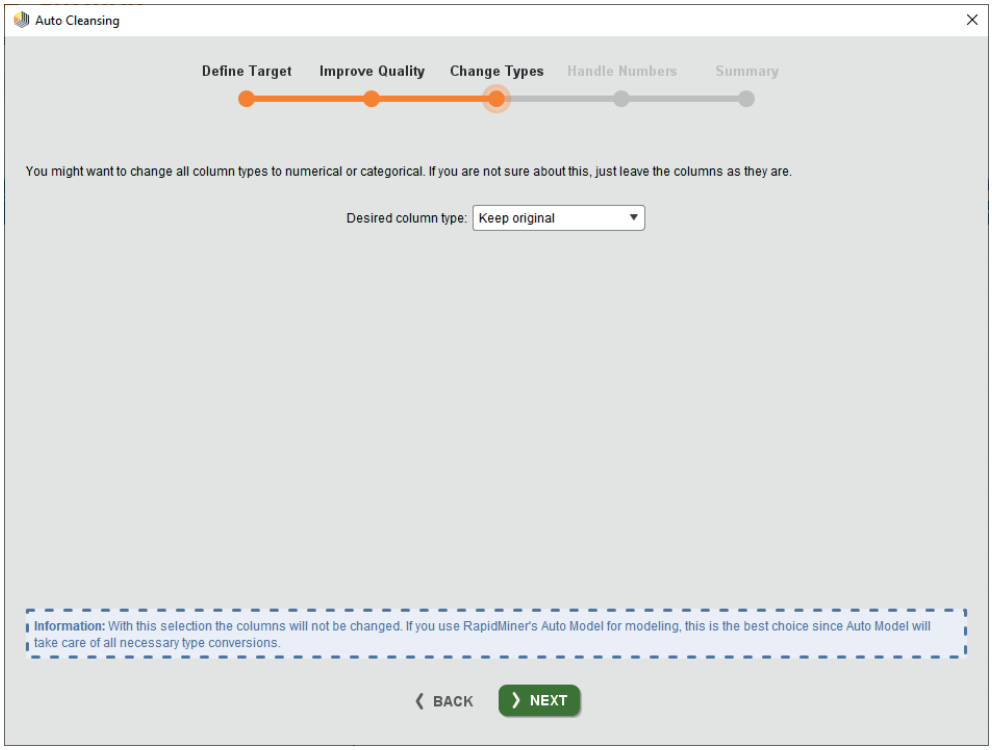
– Id: numărul de categorii este egal cu numărul de cazuri; atributul arată ca un atribut de tip Id;
– EmployeeCount: este (aproape) constant;
– StandardHours: este (aproape) constant

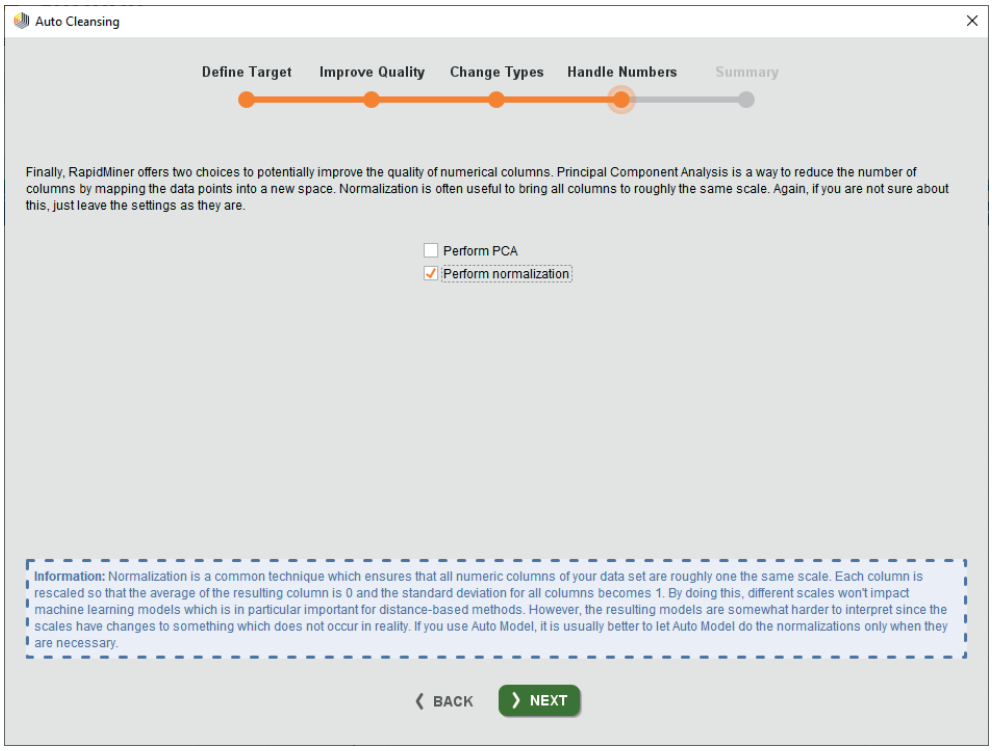
În continuare (pasul „Change Types"), putem schimba sau nu nivelul de măsurare al atributelor. Dacă alegem să-l schimbăm, avem două opțiuni: toate atributele să fie numerice sau toate categoriale. În cazul de față am păstrat nivelele de măsurare originale. La pasul patru (Handle Numbers) avem posibilitatea să normalizăm atributele numerice (Perform Normalization) sau să le reducem numărul păstrând informația comună (Perform PCA). Pot fi alese ambele opțiuni, una sau niciuna. În cazul de față am ales doar normalizarea (nu avem atribute care măsoară același concept). La ultimul pas ne sunt reamintite comenzilor alese și avem posibilitatea de a

le aplica (butonul „Apply Auto Cleansing”). „Curățarea” efectivă va fi realizată doar după ce am apăsat butonul „Commit Cleanse”. Acum numărul de atribute este 32 (înainte era 35), iar ultima modificare este „Normalize”.

Figura 8.3-3. Turbo Prep: Cleanse – Auto Cleansing



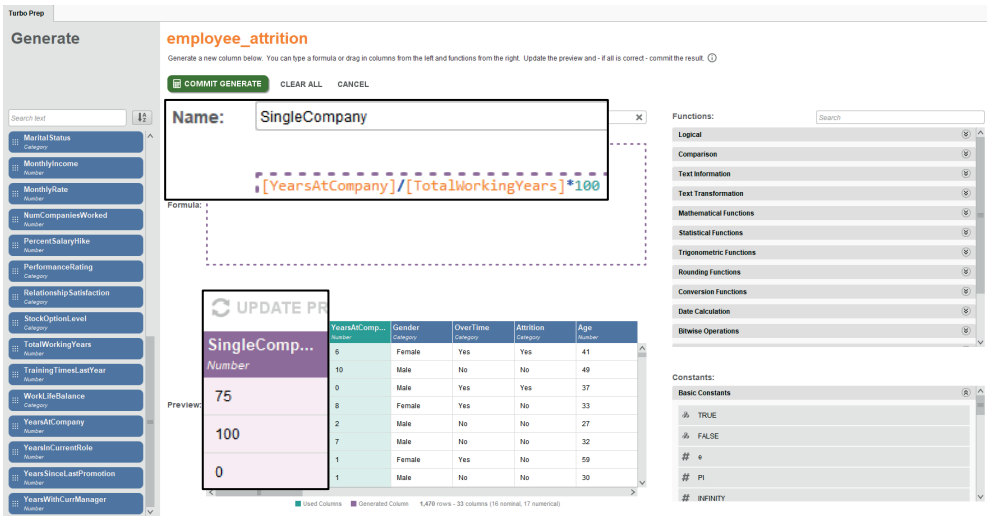

Auto Cleansing
Define Target
Improve Quality
Change Types
Handle Numbers
Summary
You might want to change all column types to numerical or categorical. If you are not sure about this, just leave the columns as they are.
Desired column type:
Keep original
Information: With this selection the columns will not be changed. If you use RapidMiner's Auto Model for modeling, this is the best choice since Auto Model will take care of all necessary type conversions.
BACK
NEXT


Auto Cleansing
Define Target
Improve Quality
Change Types
Handle Numbers
Summary
Finally, RapidMiner offers two choices to potentially improve the quality of numerical columns. Principal Component Analysis is a way to reduce the number of columns by mapping the data points into a new space. Normalization is often useful to bring all columns to roughly the same scale. Again, if you are not sure about this, just leave the settings as they are.
Perform PCA
Perform normalization
Information: Normalization is a common technique which ensures that all numeric columns of your data set are roughly one the same scale. Each column is rescaled so that the average of the resulting column is 0 and the standard deviation for all columns becomes 1. By doing this, different scales won't impact machine learning models which is in particular important for distance-based methods. However, the resulting models are somewhat harder to interpret since the scales have changes to something which does not occur in reality. If you use Auto Model, it is usually better to let Auto Model do the normalizations only when they are necessary.
BACK
NEXT

8.4. Turbo Prep: Generarea unor atribute (Generate)

Categoria Generate include o serie de funcții care pot fi folosite pentru a genera noi atribute. În acest caz vom genera un nou atribut numit SingleCompany. Atributul este calculat după formula „YearsInCompany / TotalWorkingYears * 100”, deci măsoară ponderea anilor petrecuți de un angajat în această companie relativ la numărul total de ani de muncă ai acelui angajat. Pentru a calcula acest atribut îi scriem denumirea în căsuța Name, în căsuța Formula trecem formula de calcul (numele atributelor poate fi preluat din lista afișată în stânga), apoi apăsăm butonul „Commit Generate”. Înainte de a implementa efectiv generarea putem observa ce modificare va fi produsă apăsând butonul update (noua variabilă va fi afișată în tabel).

Figura 8.4-1. Turbo Prep: Generate

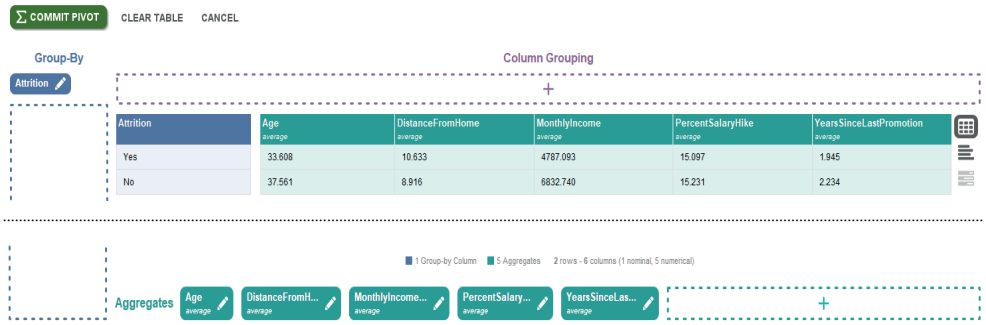


8.5. Turbo Prep: Pivotarea unui tabel (Pivot)

Categoria Pivot e utilă pentru a produce tabele cu diferite tipuri de date agregate. De exemplu, în Figura 8.5-1 am ilustrat construirea unui tabel

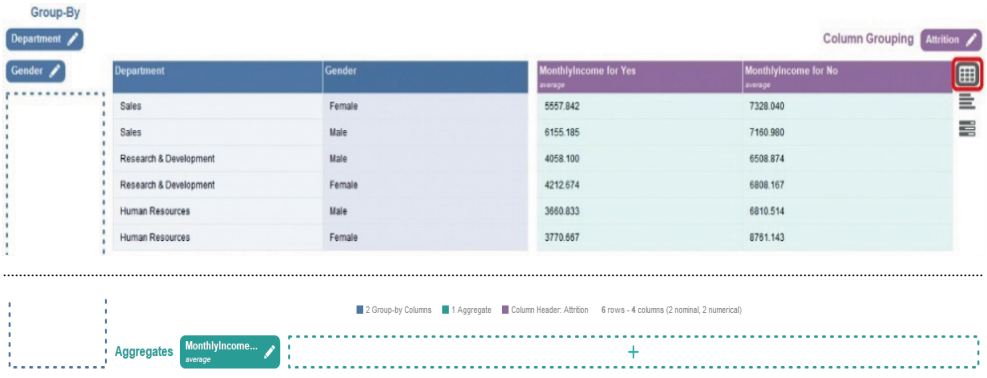
sintetic care prezintă mediile câtorva atribute pentru fiecare dintre categoriile atributului Attrition. Astfel, putem compara caracteristicile angajaților care părăsesc compania versus cei care rămân. Datele prezentate arată că, în medie, angajații care pleacă sunt mai tineri, locuiesc la o distanță mai mare față de locul de muncă și au un venit mai mic. Pentru a realiza un astfel de tabel am selectat atributul Attrition din lista de atribute și l-am tras în coloana Group-By, apoi am procedat la fel cu celelalte atribute, pe rând, de această dată adăugându-le la linia Aggregates.

Figura 8.5-1. Turbo Prep: Pivot



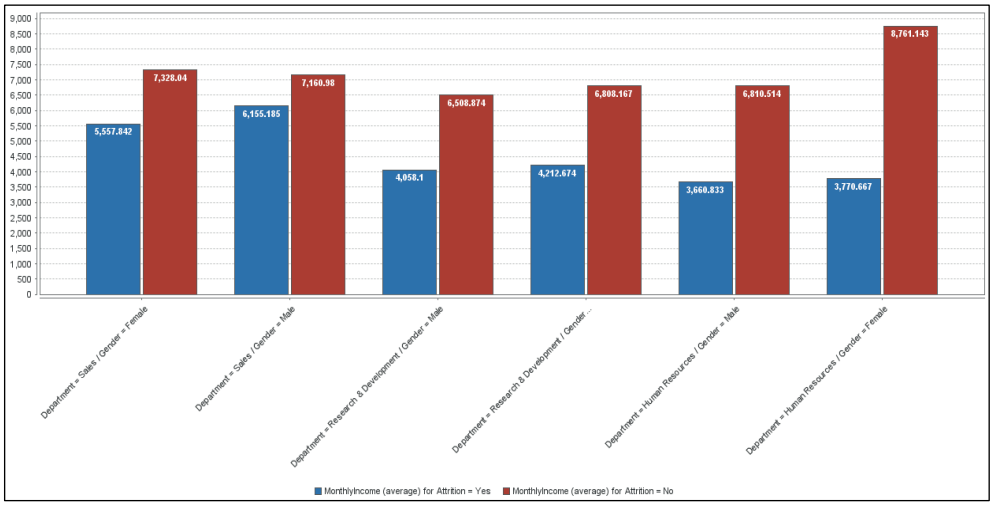
Putem produce tabele cu și mai multe atribute, funcție de tipul de statistici dorit. Astfel, în Figura 8.5-2 am calculat salariul mediu pentru fiecare combinație a atributelor Department, Gender și Attrition.

Figura 8.5-2. Turbo Prep: Pivot (format tabelar)



Aceleași date pot fi ilustrate și grafic. Pentru aceasta trebuie doar să alegem în loc de formatul tabelar, formatul grafic. Acest lucru se face simplu, selectând semnul grafic potrivit din partea dreapta sus.

Figura 8.5-3. Turbo Prep: Pivot (format grafic)

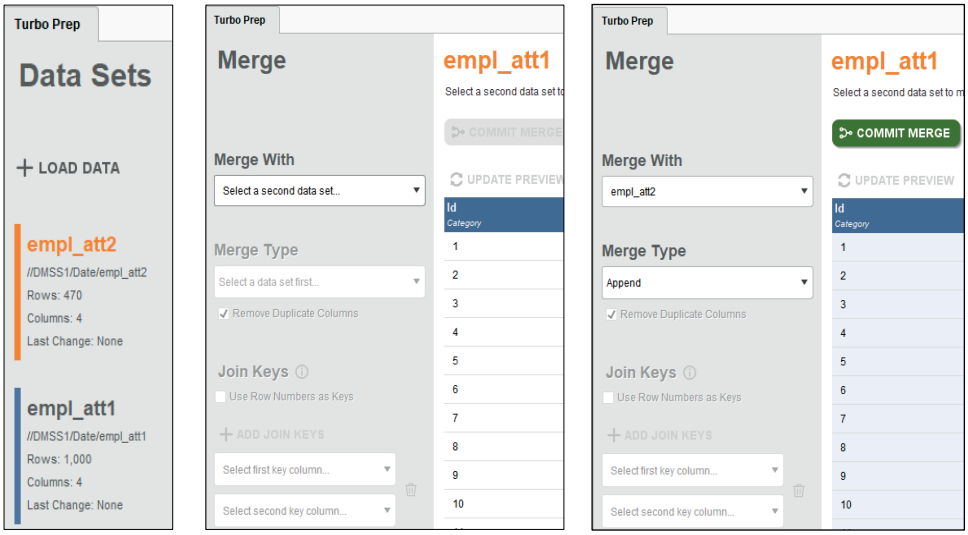


8.6. Turbo Prep: Unirea a două seturi de date (Merge)

Comanda Merge ne asistă în procesul de unire a două seturi de date. Pentru a putea folosi această comandă trebuie în prealabil să încărcăm în Turbo Prep cel puțin două seturi de date. Putem executa toate comenzile descrise la categoria Merge din fereastra Operators. De exemplu, pentru a pune împreună cazurile din două seturi de date parcurgem următorii pași (Figura 8.6-1):

- încărcăm seturile de date emp_att1 și emp_att2 (Load Data);
- ne poziționăm pe primul set de date și alegem la „Merge With" al doilea set de date;
- alegem Append la „Merge Type";
- apăsăm butonul „Commit Merge";
- dacă dorim, putem salva toate aceste comenzi sub forma unui proces „Create Process".

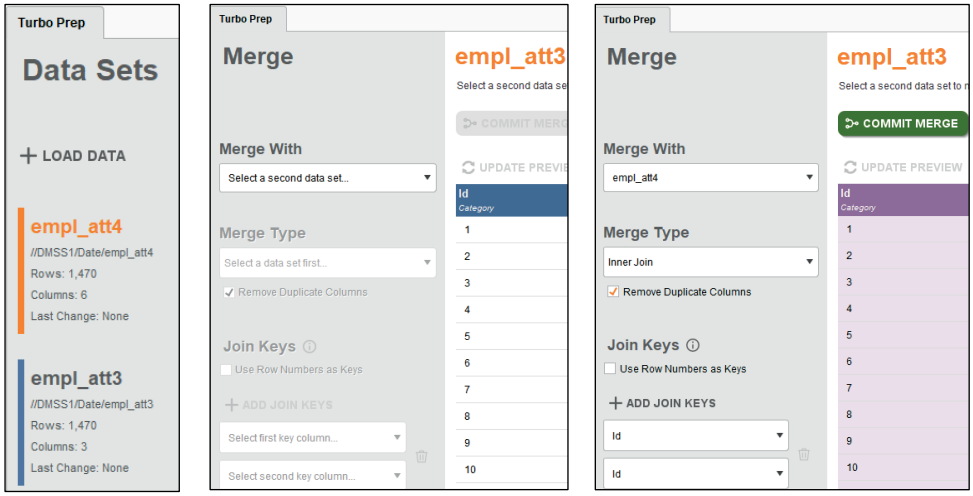
Figura 8.6-1. Turbo Prep: Merge - Append



În cazul în care dorim să unim mai multe variabile din două seturi de date, cazurile fiind aceleași, folosim opțiunea „Inner Join” (Figura 8.6-2). Pașii sunt relativ similari:

- încărcăm seturile de date emp_att3 și emp_att4 (Load Data);
- ne poziționăm pe primul set de date și alegem la „Merge With” al doilea set de date;
- alegem „Inner Join” la „Merge Type”;
- la „Join Keys” indicăm cele două atribute de tip Id în funcție de care dorim să facem unirea (nu e obligatoriu ca acestea să aibă același nume);
- apăsăm butonul „Commit Merge”;
- dacă dorim, putem salva toate aceste comenzi sub forma unui proces „Create Process”.

Figura 8.6-2. Turbo Prep: Merge - Inner Join

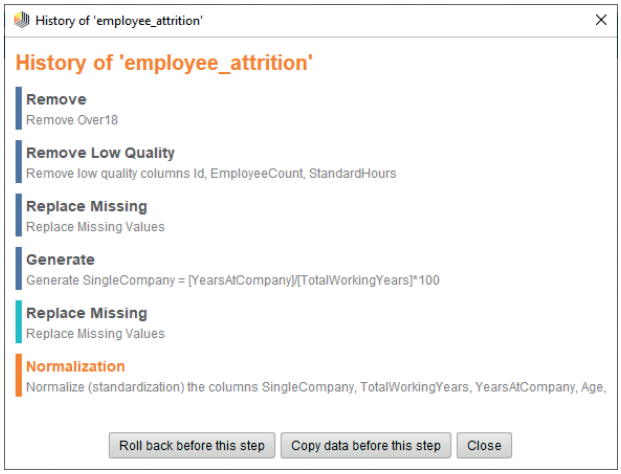


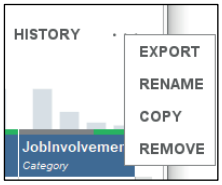
8.7. Turbo Prep: Salvarea procesului și istoricul modificărilor

În acest moment setul de date este pregătit pentru analiză. Dacă dorim, putem salva toate comenzile implementate într-un fișier de tip proces folosind butonul „Create Process” (dreapta sus). Procesul rezultat conține toți operatorii folosiți. Fiecare operator are asociat un scurt text care indică clar comanda efectuată și variabilele implicate. În cazul de față am salvat doar comenzile din categoriile Transform, Cleanse și Generate (exemplele relativ la categoriile Pivot și Merge au fost ilustrate separat, folosind setul original de date).

Butonul History (dreapta sus) ne arată toate comenzile efectuate, în ordine și ne oferă posibilitatea de a aduce setul de date la starea anterioară aplicării unei comenzi, respectiv de a face o copie a setului de date așa cum arăta acesta înainte de rularea comenzii respective (Figura 8.7-1). Dacă apăsăm cele trei puncte de lângă butonul History, putem aplica diferite comenzi setului de date încărcat (export, redenumire, copiere, eliminare).

Figura 8.7-1. Turbo Prep: butoanele History și „…”





Bibliografie

Aguinis, H., Gottfredson, R. K., & Joo, H. (2013). Best-Practice Recommendations for Defining, Identifying, and Handling Outliers. *Organizational Research Methods*, *16*(2), 270–301. https://doi.org/10.1177/1094428112470848

Attewell, P., & Monaghan, D. (2015). *Data Mining for the Social Sciences: An Introduction*. University of California Press.

Balusamy, B., Abirami, N. R., Kadry, S., & Gandomi, A. H. (2021). *Big Data: Concepts, Technology, and Architecture*. John Wiley & Sons, Inc.

Bunge, J. A., & Judson, D. H. (2005). Data Mining. In K. Kempf-Leonard (Ed.), *Encyclopedia of Social Measurement* (pp. 617–624). Elsevier.

Buuren, S. van. (2018). *Flexible imputation of missing data*. CRC Press, Taylor & Francis Group.

Chisholm, A. (2013). *Exploring Data with RapidMiner. Explore, understand, and prepare real data using RapidMiner's practical tips and tricks*. Packt Publishing.

De Vaus, D. (2002). *Analyzing social science data : 50 key problems in data analysis*. Sage.

Dușa, A., Oancea, B., Caragea, N., Alexandru, C., Jula, N. M., & Dobre, A.-M. (2015). *R cu aplicații în statistică*. Editura Universității din București.

Enders, C. K. (2010). *Applied missing data analysis*. Guilford Press.

Finch, W. H. (2012). Distribution of variables by method of outlier detection. *Frontiers in Psychology*, *3*, 1–12. https://doi.org/10.3389/fpsyg.2012.00211

Foster, I., & Heus, P. (2021). Databases. In I. Foster, G. Rayid, S. R. Jarmin, F. Kreuter, & J. Lane (Eds.), *Big Data and Social Science. Data Science Methods*

and Tools for Research and Practice (2nd ed., pp. 67–99). Chapman and Hall/CRC.

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.

Gassen, J., & Veenman, D. (2022). Outliers and Robust Inference in Archival Accounting Research. *SSRN Electronic Journal*. https://doi.org/10.2139/ssrn.3880942

Grolemund, G., & Wickham, H. (2017). *R for Data Science*. O'Reilly Media.

Hofmann, M., & Klinkenberg, R. (Eds.). (2016). *RapidMiner. Data Mining Use Cases and Business Analytics Applications*. Chapman and Hall/CRC.

Howitt, D., & Cramer, D. (2010). *Introducere în SPSS pentru psihologie*. Polirom.

Hyvärinen, A., Karhunen, J., & Oja, E. (2001). *Independent component analysis*. John Wiley & Sons.

Japec, L., Kreuter, F., Berg, M., Biemer, P., Decker, P., Lampe, C., Lane, J., O'Neil, C., & Usher, A. (2015). Big data in survey research: AAPOR Task Force Report. *Public Opinion Quarterly*, *79*(4), 839–880. https://doi.org/10.1093/poq/nfv039

Kotu, V., & Deshpande, B. (2015). *Predictive analytics and data mining: concepts and practice with RapidMiner*. Morgan Kaufmann.

Kotu, V., & Deshpande, B. (2019). *Data science: concepts and practice* (2nd ed.). Morgan Kaufmann.

Leys, C., Delacre, M., Mora, Y. L., Lakens, D., & Ley, C. (2019). How to Classify, Detect, and Manage Univariate and Multivariate Outliers, With Emphasis on Pre-Registration. *International Review of Social Psychology*, *32*(1), 1–10. https://doi.org/10.5334/irsp.289

Leys, C., Klein, O., Dominicy, Y., & Ley, C. (2018). Detecting multivariate outliers: Use a robust variant of the Mahalanobis distance. *Journal of Experimental Social Psychology*, *74*, 150–156. https://doi.org/10.1016/j.jesp.2017.09.011

Leys, C., Ley, C., Klein, O., Bernard, P., & Licata, L. (2013). Detecting outliers: Do not use standard deviation around the mean, use absolute deviation around the median. *Journal of Experimental Social Psychology*, *49*(4), 764–766. https://doi.org/10.1016/j.jesp.2013.03.013

Little, R. J. A., & Rubin, D. B. (2019). *Statistical Analysis with Missing Data*. Wiley.

McNeely, C. L., & Schintler, L. A. (2022). Big Data Concept. In L. A. Schintler & C. L. McNeely (Eds.), *Encyclopedia of Big Data* (pp. 79–82). Springer, Cham. https://doi.org/10.1007/978-3-319-32010-6

Mierswa, I. (2016a). Getting Used to RapidMiner. In M. Hofmann & R. Klinkenberg (Eds.), *RapidMiner. Data Mining Use Cases and Business Analytics Applications* (pp. 19–30). Chapman and Hall/CRC.

Mierswa, I. (2016b). What This Book is About and What It is Not. In M. Hofmann & R. Klinkenberg (Eds.), *RapidMiner. Data Mining Use Cases and Business Analytics Applications* (pp. 3–18). Chapman and Hall/CRC.

Millea, V.-Z. (2018). *Gestionarea și analiza datelor cu SPSS și PSPP. Partea 1: Gestionarea și analiza datelor*. Presa Universitară Clujeană.

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

Newman, D. A. (2014). Missing Data: Five Practical Guidelines. *Organizational Research Methods*, *17*(4), 372–411. https://doi.org/10.1177/1094428114548590

Nisbet, R., Miner, G. D., & Yale, K. (2018). *Handbook of Statistical Analysis and Data Mining Applications*. Academic Press.

North, M. (2018). *Data mining for the masses*. CreateSpace Independent Publishing Platform.

Olkin, I., & Sampson, A. R. (2001). Multivariate Analysis: Overview. In N. J. Smelser & P. Baltes (Eds.), *International Encyclopedia of the Social & Behavioral Sciences* (pp. 10240–10247). Pergamon.

Opariuc-Dan, C. (2009). *Statistică aplicată în ştiinţele socio-umane: Noţiuni de bază - Statistici univariate*. ASCR.

Pagans, F. G. (2015). *Predictive Analytics Using Rattle and Qlik Sense*. Packt Publishing.

Popa, M. (2008). *Statistică pentru psihologie. Teorie și aplicații SPSS*. Polirom.

Ranga Suri, N. N. R., Narasimha, M. M., & Athithan, G. (2019). *Outlier Detection: Techniques and Applications : A Data Mining Perspective*. Springer. https://doi.org/10.1007/978-3-030-05127-3

RapidMiner. (2014). *RapidMiner Studio Manual*. RapidMiner GmbH.

RapidMiner. (2022). *RapidMiner 9. Operator Reference Manual*. RapidMiner GmbH.

Roiger, R. J. (2017). *Data mining: a tutorial-based primer* (2nd ed.). Chapman and Hall/CRC.

Rotariu, T. (1991). *Curs de metode și tehnici de cercetare sociologică*. Presa Universitară Clujeană.

Sava, F. A. (2011). *Analiza datelor în cercetarea psihologică*. ASCR.

Smiti, A. (2020). A critical overview of outlier detection methods. *Computer Science Review*, *38*, 100306. https://doi.org/10.1016/j.cosrev.2020.100306

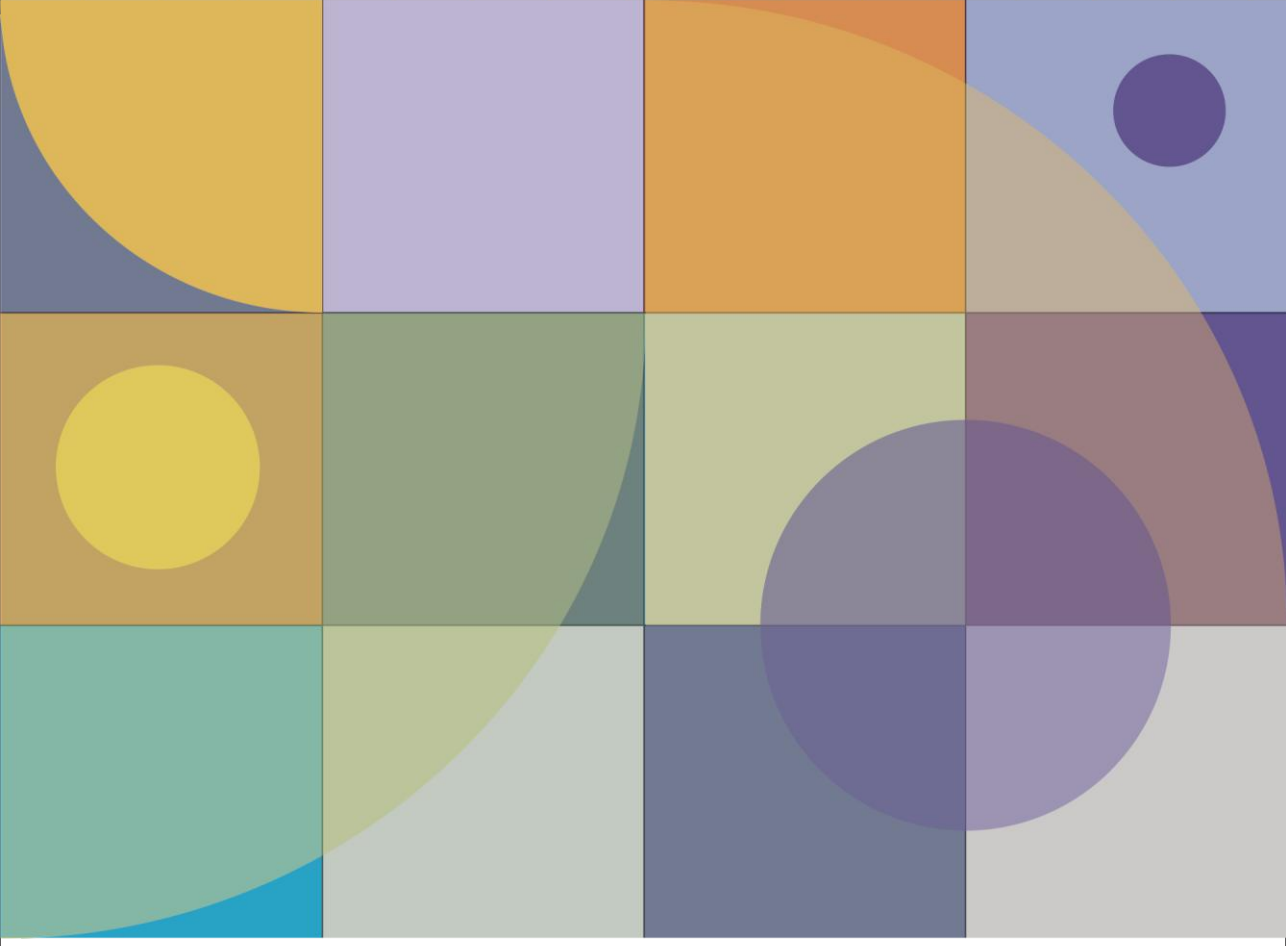
Spiess, M., & Augustin, T. (2021). Handling missing data in large databases. In U. Engel, A. Quan-Haase, S. X. Liu, & L. Lyberg (Eds.), *Handbook of Computational Social Science, Volume 2: Data Science, Statistical Modelling, and Machine Learning Methods* (pp. 82–94). Routledge.

Tabachnick, B. G., & Fidell, L. S. (2019). *Using Multivariate Statistics*. Pearson.

Tretter, M. J. (2003). Data Mining. In H. Bidgoli (Ed.), *Encyclopedia of Information Systems* (pp. 477–488). Elsevier. https://doi.org/10.1016/B0-12-227240-4/00033-2

Vasile, M. (2014). *Introducere în SPSS pentru cercetarea socială și de piață*. Polirom.

Wang, Q. (2014). *Kernel Principal Component Analysis and its Applications in Face Recognition and Active Shape Models*. https://doi.org/10.48550/ARXIV.1207.3538



ISBN: 978-606-37-1496-2
ISBN: 978-606-37-1497-9