Data Analytics

Workbook

Open University of the Netherlands
Faculty of Science

Course team
Dr. D. Iren, *author*
Prof. dr. ir. R.W. Helms, *author*

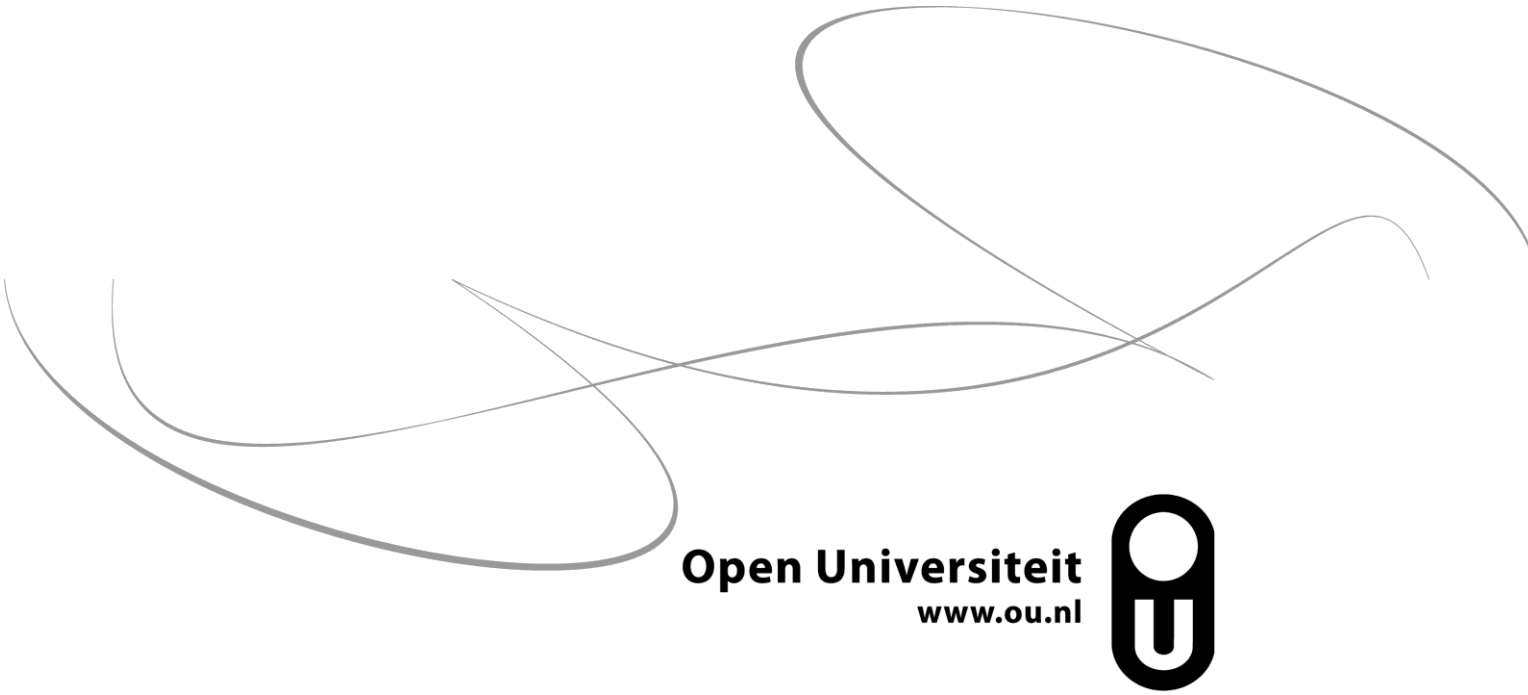
Discipline direction
-

Programme director
Prof. dr. R.J. Kusters

External referent
-

WORKBOOK

Data Analytics



Open Universiteit
www.ou.nl

Production
Open University of the Netherlands

Editor
-

Lay-out
-

Cover
-

Visuele communicatie Open Universiteit

Printing and binding
-

First edition: 2023

-

ISBN XXX XX XXXXXXX X (serie)
ISBN XXX XX XXXXX XX X (workbook)

Course code IM0503

Contents

1 Data Analytic Mindset

Data analytics has received much attention lately due to the growing amount of data that is at our disposal today. Data is considered a valuable asset that needs to be explored and analysed in order to be transformed into information and actionable knowledge. Transforming data into actionable knowledge requires techniques that are generally referred to as data analytics or data mining. Since there are so many different analytical techniques, a good understanding of their strengths and limitations is essential for their application in a business context and is the focus of this course.

Developing a deep understanding of data analytics techniques is imperative for addressing business problems using data. The knowledge gained about data analytics is particularly useful for professionals who want to understand the potential of data analytics for their organizations. Additionally, people who act as an intermediary between the business and data analysts will certainly benefit from this course as it will allow them to communicate effectively with both parties.

Organizations benefit from data analytics in the following ways.

- Get to know their customers, better target new clients,
- Cut operation costs and optimize processes,
- Solve problems,
- Innovate,
- Gain a competitive edge (or just survive!).

Individuals can use data analytics in their careers in the following ways.

- Broaden their professional skills,
- Translate business problems into data problems,
- Acquire and process relevant data to solve problems,
- Unveil insights from the data.

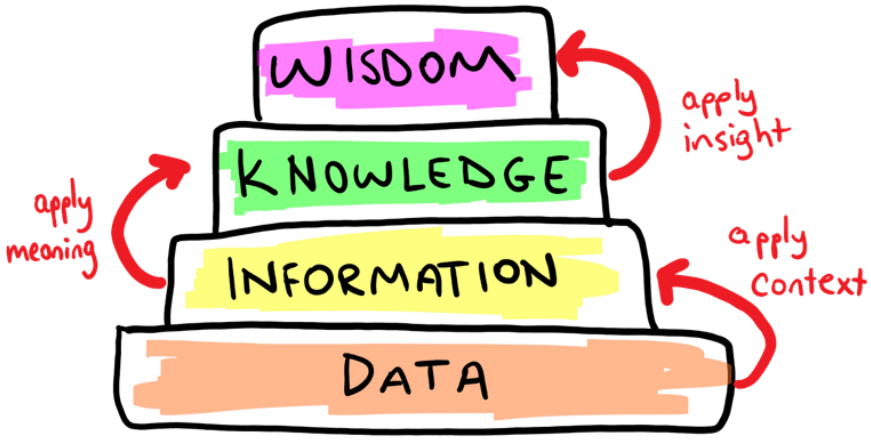


Figure 1 - Data, information, knowledge, and wisdom.

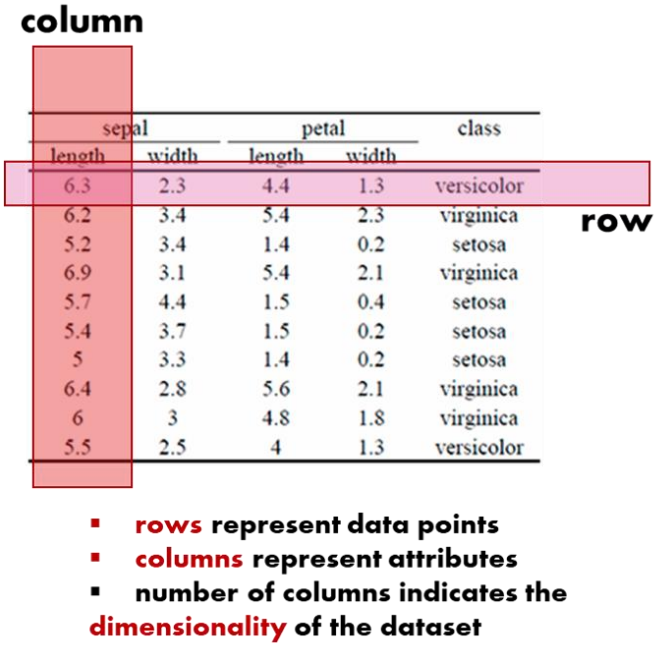
The goal of data analytics can be defined as transforming data into information, information into knowledge, and knowledge into wisdom (see Figure 1).

1.1 Fundamental concepts

This section summarizes the fundamental concepts that will be frequently encountered throughout the course as well as any discussion on data analytics.

Data are informative units that represent characteristics or traits of an entity, that are collected through observation. They consist of qualitative or quantitative values about entities or phenomena. On a **data table**, generally, **rows** represent data elements, and **columns** represent attributes (or features). The number of columns of a data table is the **dimensionality** of the dataset (see Figure 2).

Attributes (or features) are values that represent some aspect of an entity or a phenomenon. Attributes can be categorical or numerical. If an attribute depicts something that we aim at predicting, or if it is describing the essence of the entity in a purposeful manner, then it is a candidate target attribute.



sepal		petal		class
length	width	length	width	
6.3	2.3	4.4	1.3	versicolor
6.2	3.4	5.4	2.3	virginica
5.2	3.4	1.4	0.2	setosa
6.9	3.1	5.4	2.1	virginica
5.7	4.4	1.5	0.4	setosa
5.4	3.7	1.5	0.2	setosa
5	3.3	1.4	0.2	setosa
6.4	2.8	5.6	2.1	virginica
6	3	4.8	1.8	virginica
5.5	2.5	4	1.3	versicolor

Figure 2 - A sample data table

Algorithm is a set of well-defined rules or instructions to perform problem-solving operations.

A **model** is an abstract representation of reality. In data science, a model is created using an algorithm and a set of data. Both the algorithm and the data that are used in the creation of the model have important effects on the model (see Figure 3).

Figure 3 - Conceptual depiction of a model

1.2 Data types

Data can be broadly categorized into two types: *categorical* and *numerical*. **Categorical** data represent qualitative characteristics. **Numerical** data represent mathematical values. Every data type has a different set of statistical and mathematical operations that are meaningful (see Figure 4).

Categorical (or qualitative)

- Nominal Data: Nominal data has no notion of order or ranking. For example, genders; male and female, or spoken languages; English, Dutch, German, are nominal values.
- Ordinal Data: Despite being categorical, ordinal data has the notion of ranking and order. For example, Likert scale answers; strongly agree, agree, neutral, disagree, strongly disagree, have a certain order.

Numerical (or quantitative)

- Interval Data: Interval data consists of ordered numerical units. The difference between each ordered step is equal. For example, 10 degrees Celsius is 5 degrees higher than 5 degrees Celsius. However, interval data has no notion of a true zero. Thus, multiplication and division are not meaningful operations on interval data.
- Ratio Data: Ratio data represent numerical values, and have a true zero. For example, salary and height are ratio data.

Operation	Nominal	Ordinal	Interval	Ratio
Equality	√	√	√	√
Order		√	√	√
Addition and subtraction			√	√
Multiplication and division				√
Mode	√	√	√	√
Median		√	√	√
Arithmetic mean			√	√
Geometric mean				√

Figure 4 - Meaningful operations for different data types

1.3 Data sources

Most organizations use various types of information systems to manage their day-to-day operations as well as the relationship with their clients and other stakeholders. All these enterprise information systems produce data that can be further analysed to create additional business value.

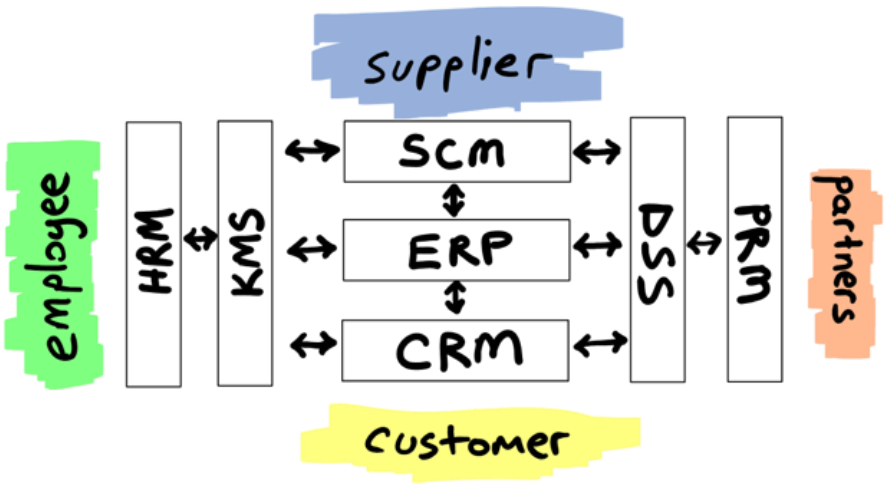


Figure 5 - Enterprise information systems as data sources

Data can also be acquired externally. To train your data analytic skills, you can even find public datasets on many sources. Some of them are listed below.

Data sources for data analytics practice:

Google: https://datasetsearch.research.google.com/
Kaggle: https://www.kaggle.com/
Amazon: https://registry.opendata.aws/
Other: https://github.com/awesomedata/awesome-public-datasets

1.4 Learning algorithms

Learning algorithms are special types of algorithms that utilize data to model an entity, a concept, or a phenomenon. This course covers the two main types of learning algorithms. It is important to understand the principal differences between these learning algorithm types, and grasp the strengths and weaknesses of each, to be able to use them correctly and effectively to solve business problems.

Main types of learning algorithms

- Supervised
- Unsupervised

Other types of learning algorithms

- Semi-supervised
- Self-supervised
- Reinforcement learning

1.5 Supervised learning

Supervised learning algorithms require labelled data to train a model. To train a supervised learning model, we provide a learning algorithm with many data examples that are characterized by the label. For example, to create a model that can predict whether a customer is loyal or will leave the company soon, we must train the algorithm with historical data that is labelled with the classes as churn or loyal. When enough labelled data is given, the model will associate the patterns within the given data with the labels, and thus, be able to predict the outcome when new data about a customer becomes available.

Strengths:

- Easy to train,
- Outcomes are easy to understand.

Weaknesses:

- Requires (a large number of) labelled data.

Common supervised learning tasks:

- Classification: supervised learning tasks in which the target attribute is categorical are called classification problems.
- Regression: supervised learning tasks in which the target attribute is numerical are called regression problems.

Examples:

Example 1: Churn classification (classification)

- Given some data on customer behaviour, we try to predict which customers will churn and which of them will remain loyal.
- Target: customer_loyalty: {churn, loyal}

Example 2: Price estimation (regression)

- Given some data on certain characteristics of a house, we try to predict the price of the house.
- Target: house_price

1.6 Unsupervised learning

Unsupervised learning algorithms do not require labelled data. Instead, they use the naturally occurring patterns within the data to group them together and identify interesting relationships among them.

Strengths:

- Does not require labelled data.

Weaknesses:

- Outcomes of unsupervised models require interpretation.

Common unsupervised tasks:

- Clustering: Clustering partitions the data into clusters based on the similarity of the data instances.
- Association Rule Mining: Uncovers the patterns of co-occurrence, such as items frequently bought together.

Examples:

Example 1: Fraud Detection (clustering)

- Given data on bank transactions, find different patterns in the data, and detect the transactions that are different than normal ones.

Example 2: Market Basket Analysis (association rule mining)

- Given customer shopping transactions, detect shopping behaviour, and give recommendations about items bought together.

1.7 Data analytics process

CRISP-DM stands for Cross-Industry Process for Data Mining, and it has been defined by a consortium of organizations to clarify and standardize how data mining is performed. This chapter summarizes the steps of CRISP-DM.

Business understanding: In this step, the business objectives are identified, clarified, and defined.

Data understanding: In the data understanding step, the business problem is transformed into a data problem. This transformation requires an understanding of the available data. At this stage, business intelligence and statistical methods can be used to explore and present the data. Consequently, data analytic goals and hypotheses can be formulated.

Data preparation: Given the hypotheses formulated, the data is prepared for a modelling task. Data cleaning and transformation also take place in the data preparation stage.

Modelling: In this step, the prepared dataset is used to train a data model. Some of the tasks that can be handled by the models are regression, classification, clustering, and association rule mining.

Evaluation: After the modelling step, the performance of the trained model is evaluated. To perform this evaluation, common performance metrics such as accuracy, precision, recall, F1-Measure, and ROC curves can be used. Additionally, hypothesis testing, and cost-benefit analysis can be applied.

Recommended reading:

CRISP-DM 1.0 Guidelines (2000) https://s2.smu.edu/~mhd/8331f03/crisp.pdf and http://www.statoo.com/CRISP-DM.pdf

A comparative study of CRISP-DM, SEMMA, and KDD (2008): https://recipp.ipp.pt/bitstream/10400.22/136/3/KDD-CRISP-SEMMA.pdf

1.8 Data analytics roles

The execution of data analytics processes requires many roles to be fulfilled. Some of the commonly found roles in the data analytics process are data scientists, data analysts, data architects, business analysts, domain experts, and data analytics managers. All these roles need to have a good understanding of data analytics to be able to conduct a data-dependent project or business.

2 Data Visualization

Data visualization is an effective technique not only for communicating the results of data analytics but also a means for data exploration. This chapter covers the fundamental charts that are frequently used in data analytics such as line charts, bar charts, pie charts, scatter plots, box plots, and histograms.

2.1 Line charts and scatter plots

The examples in this chapter use the Iris dataset which is a public dataset that is commonly used to exemplify classification algorithms. As reported on Wikipedia:[1] 'The data set consists of 50 samples from each of three species of *Iris* (*Iris setosa*, *Iris virginica* and *Iris versicolor*). Four features were measured from each sample: the length and the width of the sepals and petals, in centimetres (…)'.

An extract of such a dataset can be found in Table 1, with the Species column listing the class of each of the records. Historically speaking, the Iris dataset is the very first example of a supervised classification problem that you will encounter.

Table 1: Iris dataset extract

Sepal length	Sepal width	Petal length	Petal width	Species
5.1	3.5	1.4	0.2	*I. setosa*
4.9	3.0	1.4	0.2	*I. setosa*
4.7	3.2	1.3	0.2	*I. setosa*
7.0	3.2	4.7	1.4	*I. versicolor*
6.4	3.2	4.5	1.5	*I. versicolor*
6.9	3.1	4.9	1.5	*I. versicolor*
5.8	2.7	5.1	1.9	*I. virginica*
6.8	3.2	5.9	2.3	*I. virginica*

[1] https://en.wikipedia.org/wiki/Iris_flower_data_set

2D Scatter Plot

Iris dataset contains entities that have four features: **sepal length, sepal width**, **petal length** and **petal width**; and one target variable, in this case the **species** attribute. In a supervised algorithm, given a new item with an unknown target variable value, the question is which class this item belongs to.

Figure 6 shows how to select the Iris dataset. Figure 7 shows how to create a block, and Figure 8 shows how to connect the block to the output. After you connect to the output, execute as shown in Figure 9. Then go to the result interface Figure 10.

3D Scatter Plot

The resulting interface shows you many possibilities in terms of visualisation, in this case we will use the *Visualizations* option of RapidMiner. In the *visualizations* tab, select Scatter / Bubble as the plot type, set the X-axis to a4 of the Iris dataset, then set the value column as a2, and the colour dimension to the *label*. A simple way to get a 3D scatter plot is to add a dimension on the axis in a 2D *scatter plot*. A 3D representation of the data is then produced by the advanced charting services of RapidMiner (see Figure 11). If now you rather want to plot *a line chart*, the procedure is the same, although now you need to set the format as lines (see Figure 12).

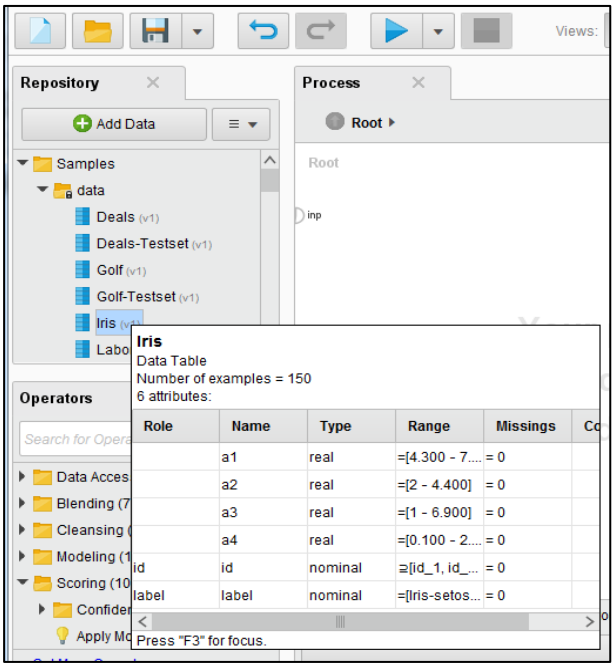


Figure 6 - The Iris dataset.

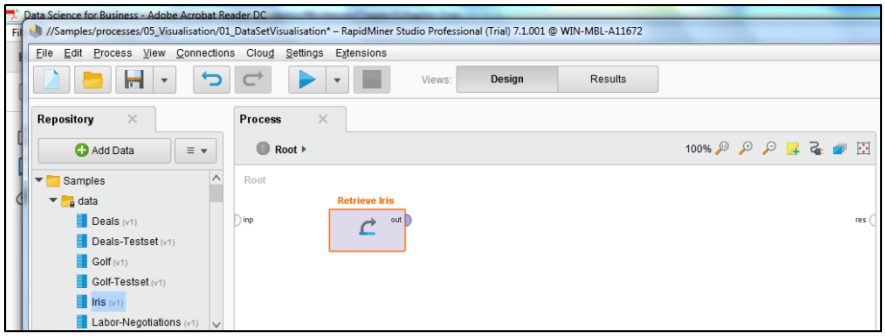


Figure 7 - Adding the Iris dataset to RapidMiner.

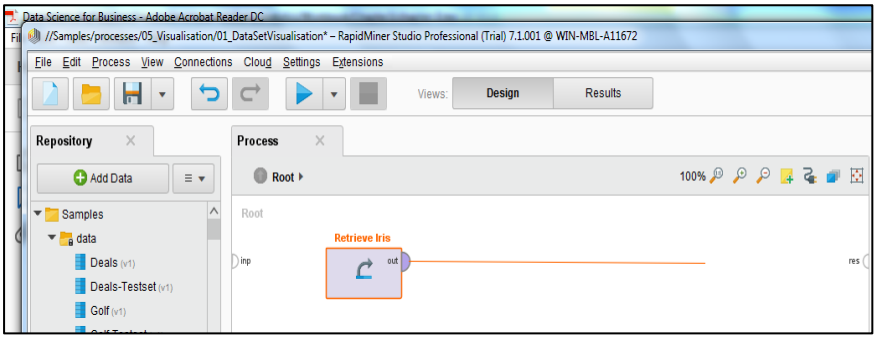


Figure 8 - Connecting the Iris dataset to the output in RapidMiner

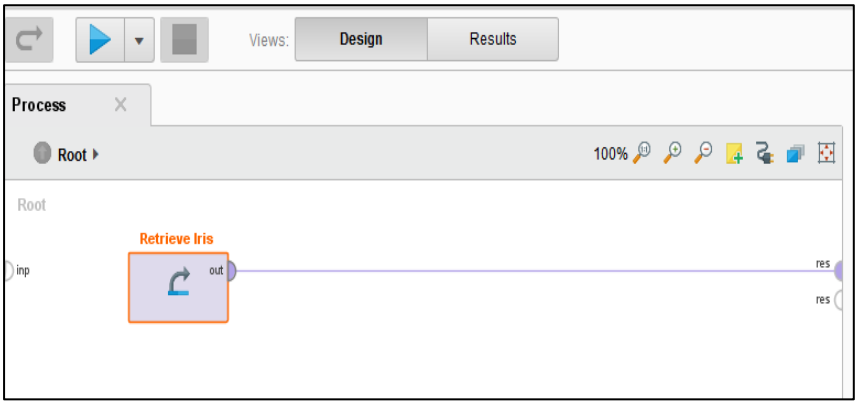
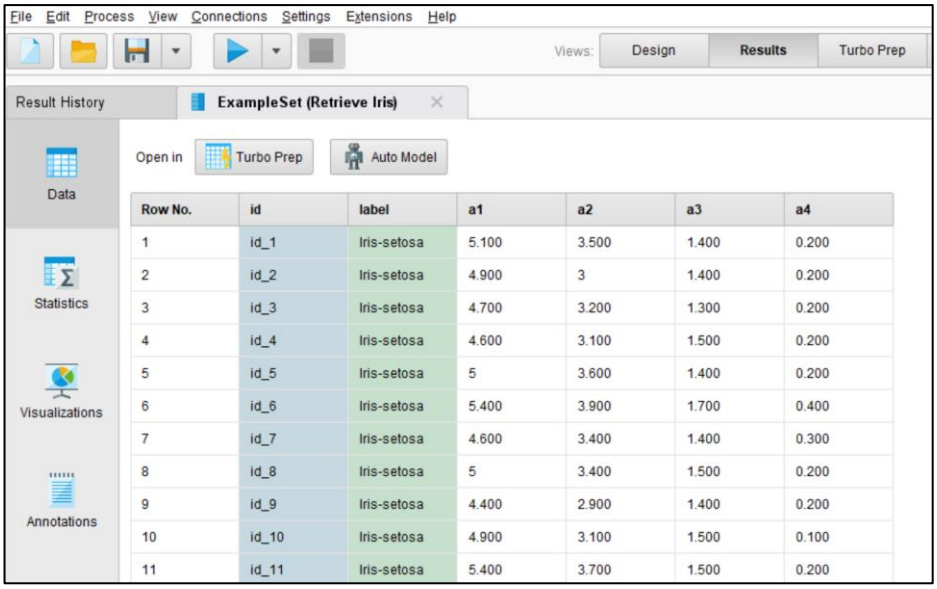


Figure 9 - Execute



Row No.	id	label	a1	a2	a3	a4
1	id_1	Iris-setosa	5.100	3.500	1.400	0.200
2	id_2	Iris-setosa	4.900	3	1.400	0.200
3	id_3	Iris-setosa	4.700	3.200	1.300	0.200
4	id_4	Iris-setosa	4.600	3.100	1.500	0.200
5	id_5	Iris-setosa	5	3.600	1.400	0.200
6	id_6	Iris-setosa	5.400	3.900	1.700	0.400
7	id_7	Iris-setosa	4.600	3.400	1.400	0.300
8	id_8	Iris-setosa	5	3.400	1.500	0.200
9	id_9	Iris-setosa	4.400	2.900	1.400	0.200
10	id_10	Iris-setosa	4.900	3.100	1.500	0.100
11	id_11	Iris-setosa	5.400	3.700	1.500	0.200

Figure 10 - Result interface.

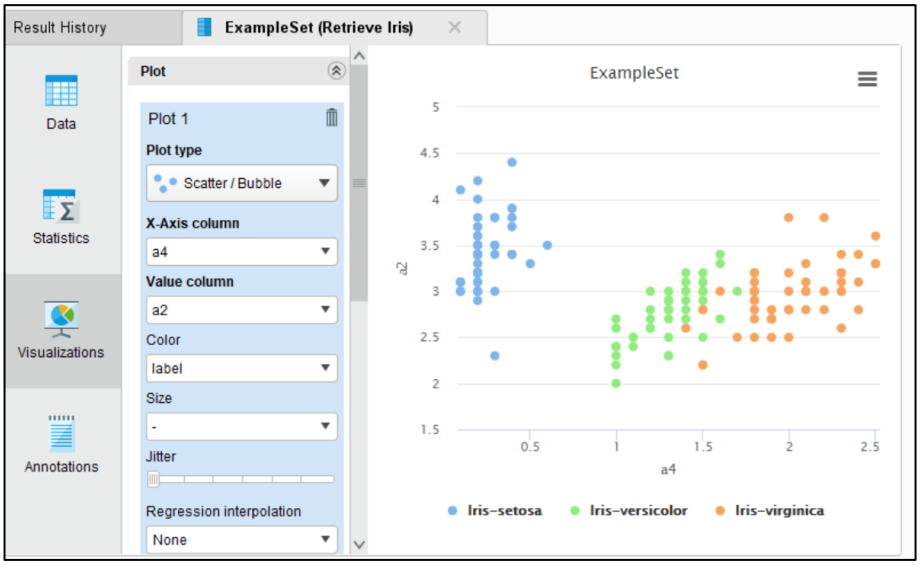


Figure 11 - 2D Scatter plot.

Interactive Line Chart

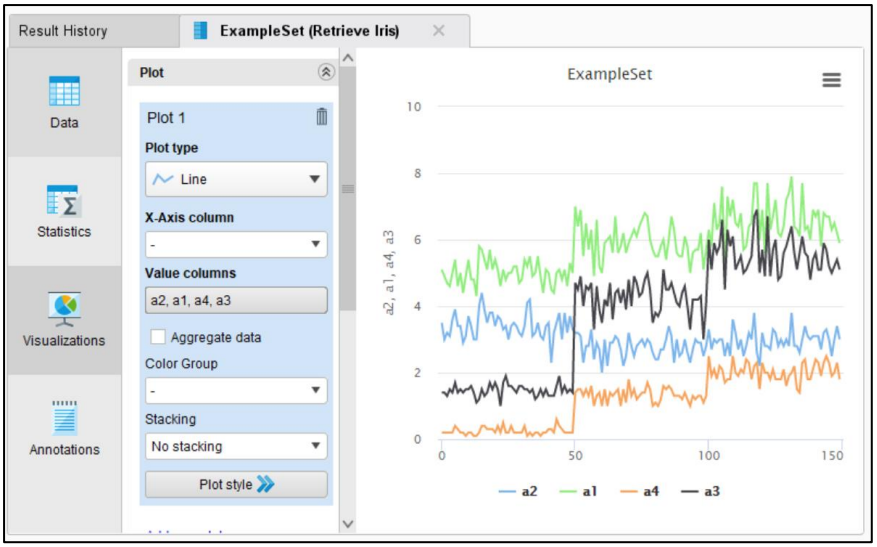


Figure 12 - Line chart

Scatter plots are generally useful with datasets in which the items are static points in time. The line charts are very useful when we plot time series.

2.2 Histogram

Histogram

A histogram is a chart that plots the frequency of the occurrence of a particular value in a data set. More specifically, this means that a set of bins is created for the range values of one of the axes. Once again, consider the iris dataset and the a2, a3, and a4 axis. To plot a histogram in RapidMiner, go to the visualizations tab (as we have seen in the previous section) and select a2, a3, and a4. Now set the number of bins to 40. This should result in the plot depicted in Figure 13.

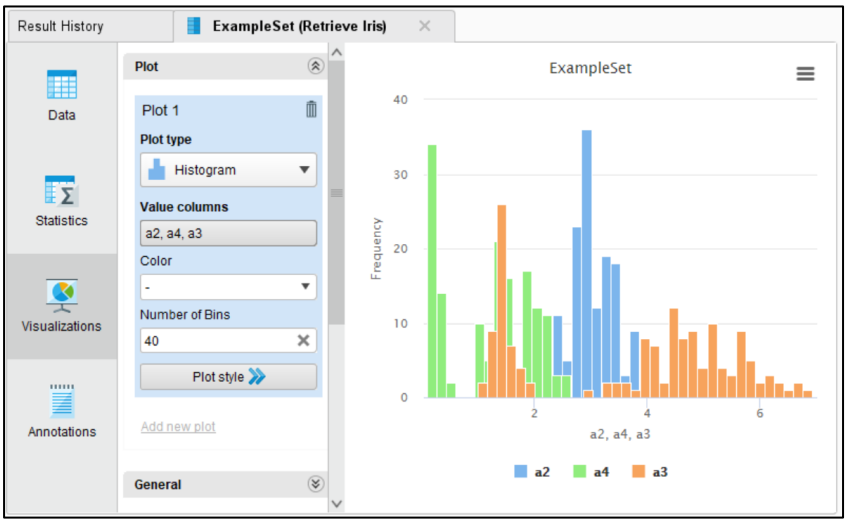


Figure 13 - Histograms in RapidMiner

Histograms are important to understand how an attribute, or a set of attributes are distributed in terms of value.

2.3 Pie charts

Pie Chart

A pie chart is usually represented by a circle divided into sectors, in which each sector represents a proportion of a certain quantity. Very often each of the sectors, or slices, is also annotated with a percentage representing how much of the total falls under a certain category.

In the case of data analytics, pie charts can be useful to observe the proportion of data points belonging to each of the classes in the dataset.

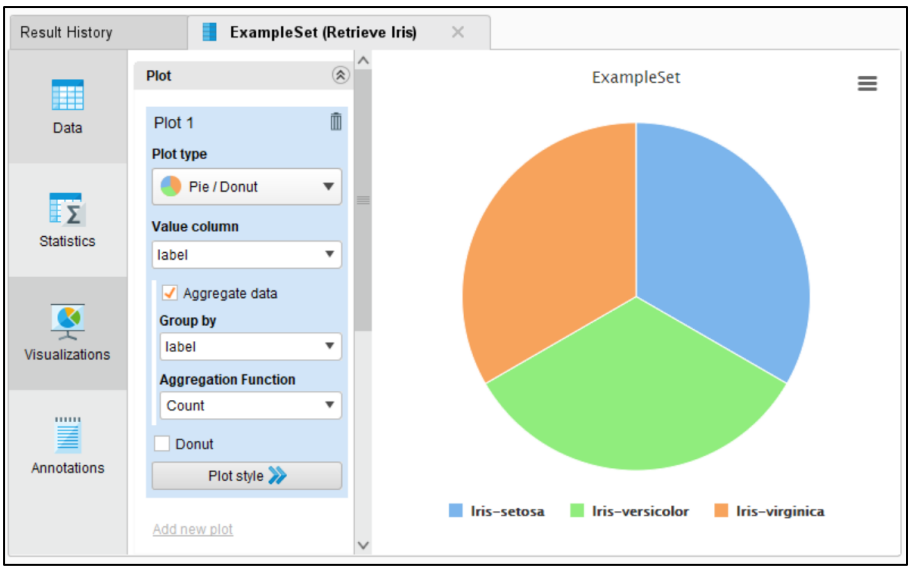


Figure 14 - Pie chart in RapidMiner

To generate this chart with RapidMiner, click on the visualizations tab on the left and then select Pie / Donut chart. Pie charts take only one value in input and in this case the column with the label should be the input. Figure 14 shows an example of such an interface. Specifically, pie charts offer an easy way to show how balanced a dataset is. The Iris dataset is well balanced, as you can see in Figure 14, with the area equally split between the three classes in the dataset.

2.4 Box plots

Box Plot

A *box plot* is a simple way to represent data by its means and quartiles. Figure 15 shows a box plot in RapidMiner. In a box plot five important statistics are presented: minimum, first quartile, median, third quartile and maximum. The first and third quartile delimit the area of the box, the minimum and the maximum are represented as whiskers, the median falls inside the box plot. The outliers are usually presented as white dots. Box plots are an important way of seeing whether the features of the dataset present outliers. It is a first way of learning the quality of a dataset and whether outlier removal methods are necessary to clean the dataset.

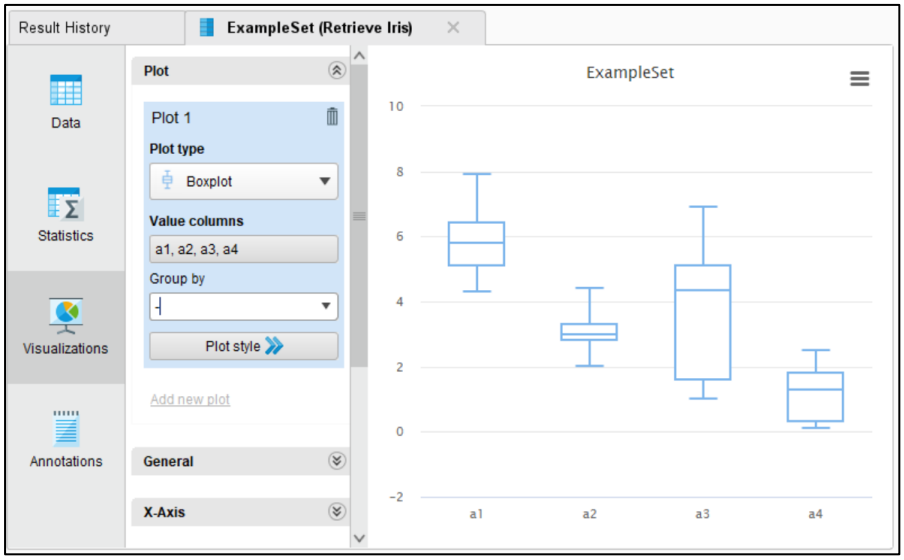


Figure 15 - Box plots in RapidMiner.

2.5 Bubble charts

Bubble Chart

RapidMiner also allows *bubble charts* to be modelled. Bubble charts are a special type of scatter plots. They can be used for representing three-dimensional data in the form of a two-dimensional diagram. The third dimension is presented as the size of a circle. Figure 16 gives an example of this diagram in RapidMiner, using the Iris dataset. Bubble charts are important to give you a 2½-dimensional impression of the dataset, allowing viewers to retain an idea of the third dimension by the size of the circle surrounding the item.

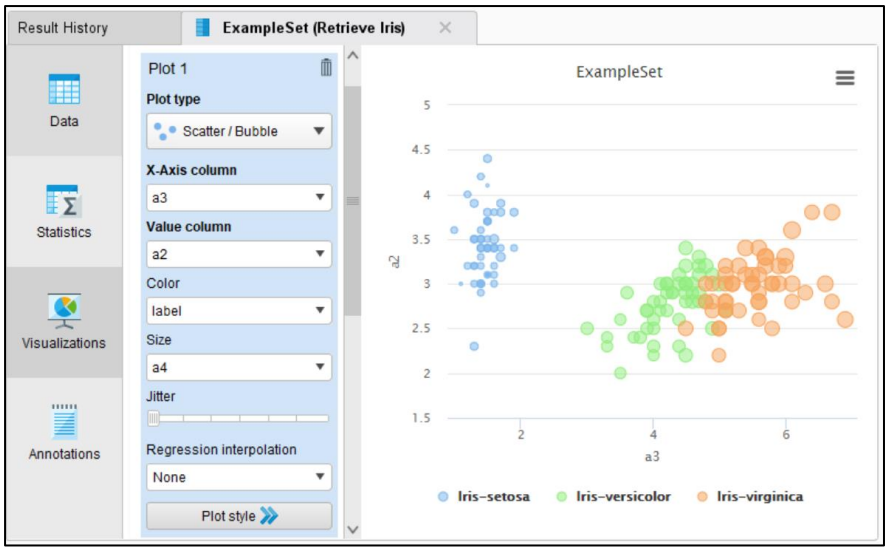


Figure 16 - Bubble charts of the Iris dataset in RapidMiner.

3 Data Quality and Data Preparation

The quality of data is imperative for effective and accurate data analytics. Therefore, data analytics practitioners must ensure the quality of data and prepare it for data analytics tasks such as machine learning modelling. This chapter covers the theory and practice of data preparation tasks.

3.1 Data quality and the need for data preparation

Data rarely comes without quality issues. To get the most accurate outcomes from modelling, first, the quality issues must be addressed. The potential data quality issues are as follows.

- Missing values: some values in the data are missing,
- Outliers: Some data records stand apart from the general distribution; they are too different in some aspects,
- Different scales: Some attributes have different scales. For example, the height of a person ranges between 100 - 190 cm while the salary ranges between 0 - 200.0000. These ranges are quite different from one another.
- Inconsistencies: Some values are recorded in different units (e.g., distance in miles or kilometres)
- Noise: Sometimes, data includes noise; data records that are not representing the phenomenon but are caused by other factors; sensor malfunction, human error, etc.

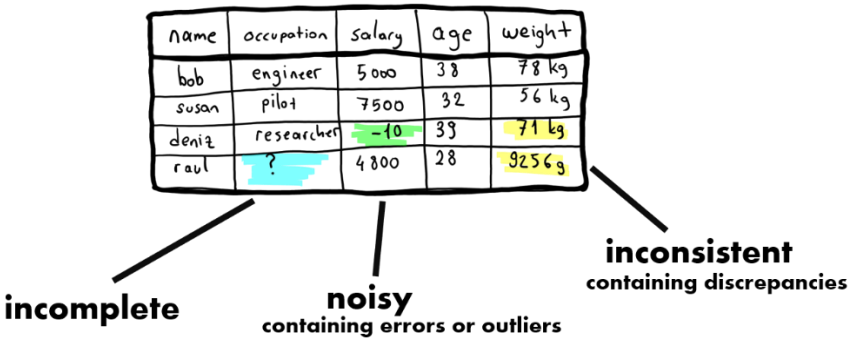


Figure 17 - Examples of data quality issues

Data preparation is important because the performance of data models highly depends on the quality of the training data. Good decisions must be based on

accurate models, thus high-quality data is essential. To diagnose and address the quality issues, certain techniques can be applied to data (see Figure 18).

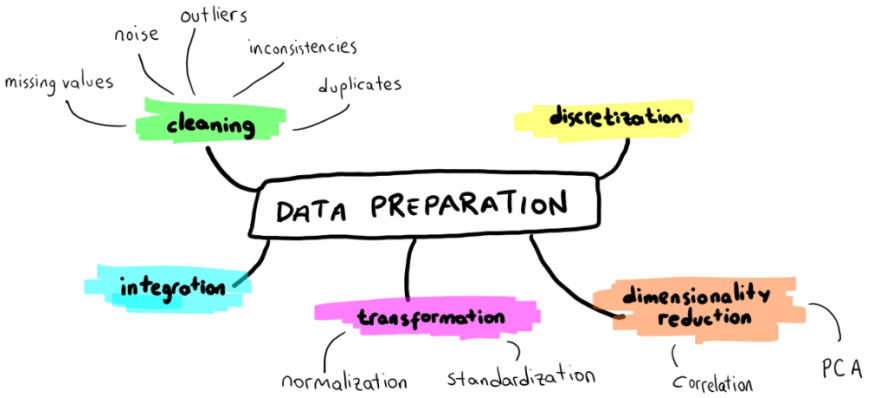


Figure 18 - Major data preparation methods

3.2 Missing values

Occasionally, data have missing values. The frequency and criticality of missing values depend on the domain. There are many possible reasons why missing values come to be. It could be a sensor malfunction, human operator error, or a result of a JOIN operation on multiple database tables. In any case, missing values must be dealt with before proceeding to machine learning modelling. Several strategies for dealing with missing values are as follows.

Video: Remove missing values with RapidMiner

Remove the missing values: Missing values can be removed especially when there are not so many missing values, thus, removing them will not cause a lot of data loss. If the class label is missing, removing the data records that have missing values makes sense due to the importance of labels for supervised learning.

Replace the missing values manually: Manual replacement of data is tedious and infeasible. This approach only works when there is a handful of missing values, and the necessary domain knowledge or intuition in place to know what to replace missing values with.

Video: Replace missing values with RapidMiner

Replace the missing values automatically: To be able to fill in the missing values automatically, there must be a way to impute the value. This can be an approximate or an aggregate value such as the mean, mode, or median of the data column. Alternatively, the statistical tendency of the data records that are representing the same class can be used. A more advanced method requires machine learning inference, such as Bayesian inference models or decision trees.

Practice: Dealing with missing values using RapidMiner

Practice the following two strategies to deal with the missing values using RapidMiner.

- Remove missing values,

- Replace missing values automatically.

For a human analyst, it is generally straightforward to detect missing values. However, RapidMiner requires an explicit definition of what missing values are. To do that, we can use the *Declare Missing Values* operator. The following workflows show how to apply both strategies. The goal is to deal with the missing values in the dataset.

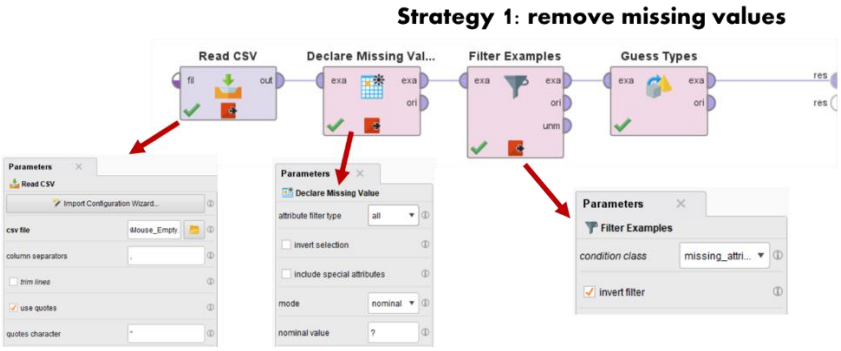


Figure 19 - Removing missing values with RapidMiner

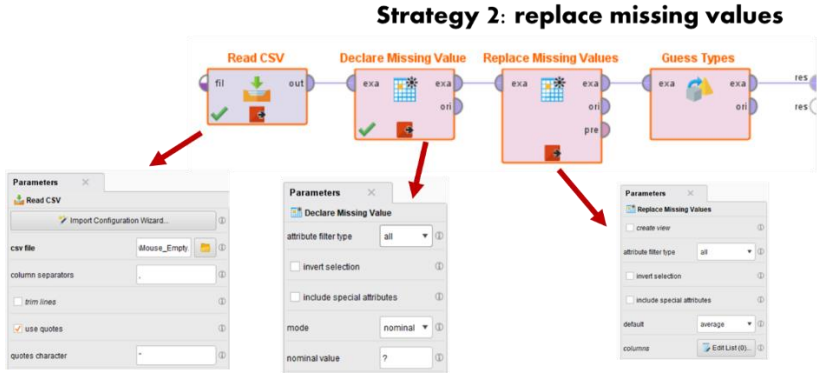


Figure 20 - Replacing missing values with RapidMiner

3.3 Outliers

An outlier is a data record having values that go outside the rest of the data distribution. An outlier stands out from the sample. Outliers can be observed using simple visualization techniques. For instance, the following figure shows a data distribution that has one obvious outlier. Doubtless, visual inspection alone is generally not sufficient for detecting outliers.

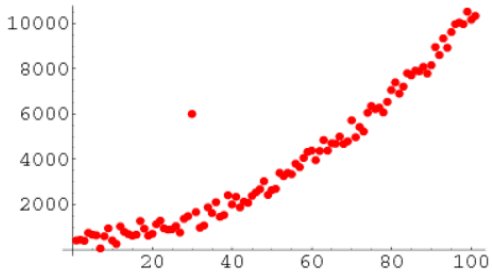


Figure 21-A scatter plot that shows a dataset with an obvious outlier

Anscombe's Quartet

There may be many reasons why outliers exist in a dataset. These could be measurement errors, sensor malfunctions, or genuinely different values that do not represent the general population (e.g., 2.5 meters tall person).

Outliers fall outside the distribution of a dataset. Therefore, they may cause large variations in the statistical properties of the dataset, having potentially adverse effects on the general trend analyses. For this reason, in most cases, outliers are removed before the modelling phase. However, in rare cases, outliers represent interesting observations. For instance, fraudulent financial transactions stand out from the normal ones, thus, outliers are very important for fraud detection.

Visual examination for outlier detection

One of the first steps of exploratory data analysis is to visualize the data. This practice has many uses, including seeing the distribution of the data and detecting outliers. One can use a scatterplot to see the data distribution in two or three dimensions. The points that stand out from the clusters are the potential outliers. Alternatively, histograms and boxplots can be used to get information on the data distribution and highlight the outliers.

A boxplot basically shows the quartiles of the data. The points that fall outside the whiskers are the outliers. The figure below shows boxplots of four data attributes. The first attribute has an obvious outlier which is clearly seen as a point that appears to have the value of 20 where most of the sample have values between 4 and 8.

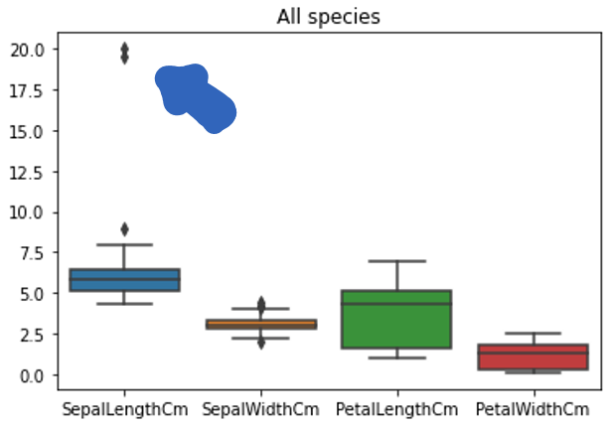


Figure 22 - A boxplot that shows the outliers

A boxplot is also useful to examine the data distribution. The boundaries of the box and the whiskers simply divide the data into four quartiles based on the standard deviation. The figure below depicts the schematics of a boxplot and shows what the boundaries indicate. The difference between Q3 and Q1 is called Inter-Quartile Range (IQR). Simply put, IQR covers 50% of the data around the mean.

Detecting outliers programmatically

To deal with the outliers (e.g., removing outliers) as a part of preprocessing steps in an automated manner, one needs to detect them. There are several metrics that can help detect outliers.

The formula below shows how many IQRs the value is away from the median. After calculating the outlier score for all data elements, one can detect the outliers by comparing the outlier score against a threshold.

$$outlierScore = \frac{(value - median)}{IQR}$$

The formula below uses the z-score that gives us how many standard deviations the value is away from the mean. Z-score is only meaningful when the data follows a normal distribution.

$$outlierScore = \frac{(value - median)}{SD}$$

Practice: **Dealing with outliers using RapidMiner**

With RapidMiner, it is possible to calculate both z-Score and IQR-based outlier scores using the operator Detect Outliers (Univariate). Several parameters need to be set. First, select which attributes you will check for outliers. Then, select a method for calculating the outlier scores. The output of this operator will be the outlier scores, added as a new column on the data table. Then, the threshold that separates the outliers from normal data values can be defined, and the outliers can be detected.

The following workflow shows how to detect outliers using RapidMiner on the Iris dataset. Please try to create the same workflow and compare your

results with the one in this example. You are encouraged to try other metrics and operators and see how the results change.

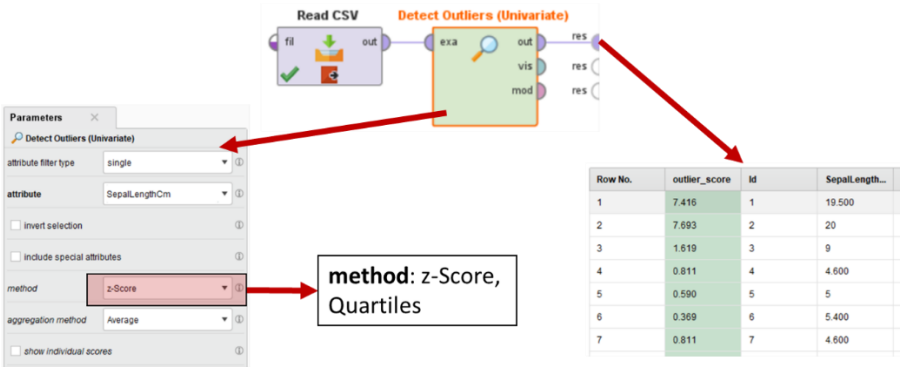


Figure 23 - Detecting outliers using RapidMiner

3.4 Feature scaling

Video: Feature scaling with RapidMiner

Feature scaling is changing the scale of one or more data attributes. Datasets may include features with values across varying degrees of magnitude, different ranges, and units. Such scale differences may hinder certain machine learning algorithms. Thus, as a good practice, feature scaling should be applied prior to the modelling phase.

Machine learning algorithms that require feature scaling:

- Gradient descent-based algorithms (e.g., linear regression, logit, neural networks)
- Distance-based algorithms (e.g., KNN, K-means, SVM)

Feature Scaling

Feature scaling improves the computation performance of these algorithms as well as effectively changing the outcome of the modelling phase.

Algorithms that are not affected by feature scale differences:

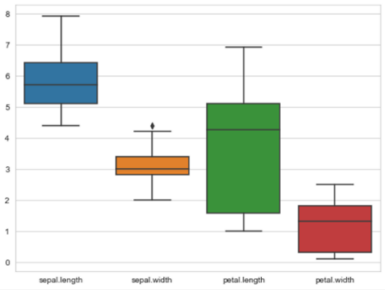
- Tree-based algorithms

For these algorithms, feature scaling does not have a considerable impact on the model.

Two common feature scaling approaches are the following.

- Normalization
- Standardization (z-transformation)

To understand the different effects of normalization and standardization, consider the following example (see Figure 24). The boxplot and the table below represent the original Iris dataset. Notice the values in the table as well as the y-axis of the boxplot and compare them against the values in the later figures.



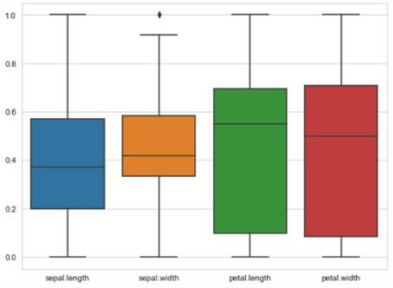
	sepal.length	sepal.width	petal.length	petal.width
count	120.000000	120.000000	120.000000	120.000000
mean	5.828333	3.100000	3.713333	1.198333
std	0.824640	0.447777	1.777730	0.776220
min	4.400000	2.000000	1.000000	0.100000
25%	5.100000	2.800000	1.575000	0.300000
50%	5.700000	3.000000	4.250000	1.300000
75%	6.400000	3.400000	5.100000	1.800000
max	7.900000	4.400000	6.900000	2.500000

Figure 24 - Iris dataset before feature scaling

Normalization

$$X' = \frac{X - X_{min}}{X_{max} - X_{min}}$$

Normalization rescales the values in the range between 0 and 1. To do that, the formula above is used. When the original Iris dataset is normalized using this formula, the results appear as in the following figure and the table. Notice the min and max values on the table, and also on the boxplots. The range of these attributes is set from 0 to 1.



	sepal.length	sepal.width	petal.length	petal.width
count	120.000000	120.000000	120.000000	120.000000
mean	0.408095	0.458333	0.459887	0.457639
std	0.235611	0.186574	0.301310	0.323425
min	0.000000	0.000000	0.000000	0.000000
25%	0.200000	0.333333	0.097458	0.083333
50%	0.371429	0.416667	0.550847	0.500000
75%	0.571429	0.583333	0.694915	0.708333
max	1.000000	1.000000	1.000000	1.000000

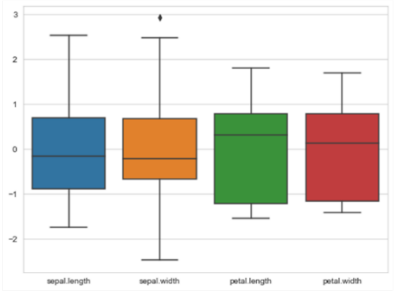
Figure 25-Iris dataset after normalization

Standardization

$$X' = \frac{X - \mu}{\sigma}$$

Standardization is another feature-scaling technique where the values are centred around the mean with one-unit standard deviation. This means that the mean of the attribute becomes zero, and the resulting distribution has one-unit standard deviation. Standardization can be performed using the formula above. μ is the mean of the feature values and ó is the standard deviation of the feature values.

The following table and the figure show the results of standardization on the Iris dataset. Note that in this case, the values are not restricted to a particular range. Also, note that the standard deviation values on the table are all very close to 1.



	sepal.length	sepal.width	petal.length	petal.width
count	1.200000e+02	1.200000e+02	1.200000e+02	1.200000e+02
mean	-7.919591e-16	-2.784809e-16	2.236637e-16	-1.852685e-16
std	1.004193e+00	1.004193e+00	1.004193e+00	1.004193e+00
min	-1.739332e+00	-2.466880e+00	-1.532690e+00	-1.420909e+00
25%	-8.869171e-01	-6.727856e-01	-1.207888e+00	-1.162170e+00
50%	-1.562760e-01	-2.242619e-01	3.031488e-01	1.315257e-01
75%	6.961386e-01	6.727856e-01	7.832914e-01	7.783734e-01
max	2.522741e+00	2.915404e+00	1.800064e+00	1.683960e+00

Figure 26 - Iris dataset after standardization

When to use normalization and standardization?

Normalization is used when the data are not normally distributed. Thus, normalization is useful for the algorithms that do not assume any distribution of the data, such as K-Nearest Neighbours and Neural Networks.

Standardization, on the other hand, may help in cases where the data follows a normal distribution. Unlike normalization, standardization does not have a bounding range. So, even if there are outliers in the data, they will not be affected by standardization.

Eventually, the decision to use normalization or standardization depends on the particular problem and the machine learning algorithm that is used.

Practice: Feature scaling using RapidMiner

Feature scaling with RapidMiner is performed using the Normalize operator. The selection of the operation, i.e., normalization or standardization, can be configured using the method parameter. *range_transformation* method indicates normalization, and this requires min and max values to be set to define the range. The z-transform method is used for standardization.

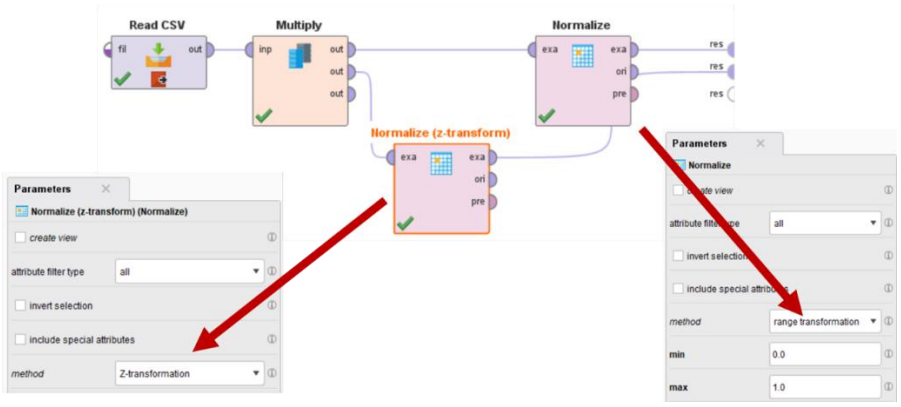


Figure 27 - Feature scaling using RapidMiner

3.5 Discretization

Discretization is the mathematical operation of transforming continuous data into discrete. The input of discretization is numerical, and the output is

Discretization

nominal/categorical. Generally, discretization is performed to divide the data into k equal lengths or percentages.

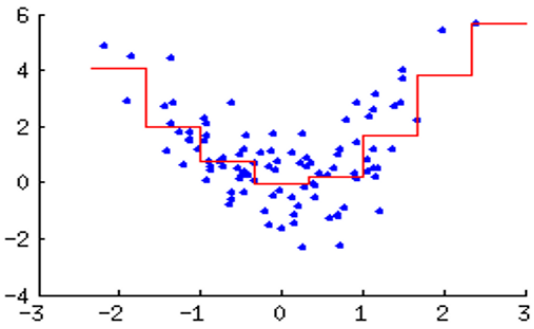


Figure 28 - A dataset that is discretized into bins of equal length

Practice: Discretization using RapidMiner

In RapidMiner, discretization is performed by using the Discretize operator. This operator has one essential parameter, which is the number of bins. This number specifies, how many discrete values will the continuous data be separated into.

Video: Discretization with RapidMiner

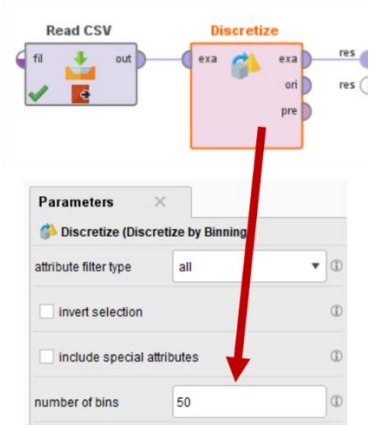


Figure 29 - Discretization using RapidMiner

In the scatter plot below, the mouse-shaped data distribution appears to be pixelated. This is because, some of the numerical values that are close to one another but not the same, are now having the same values. So, they appear to be aligned on a grid structure.

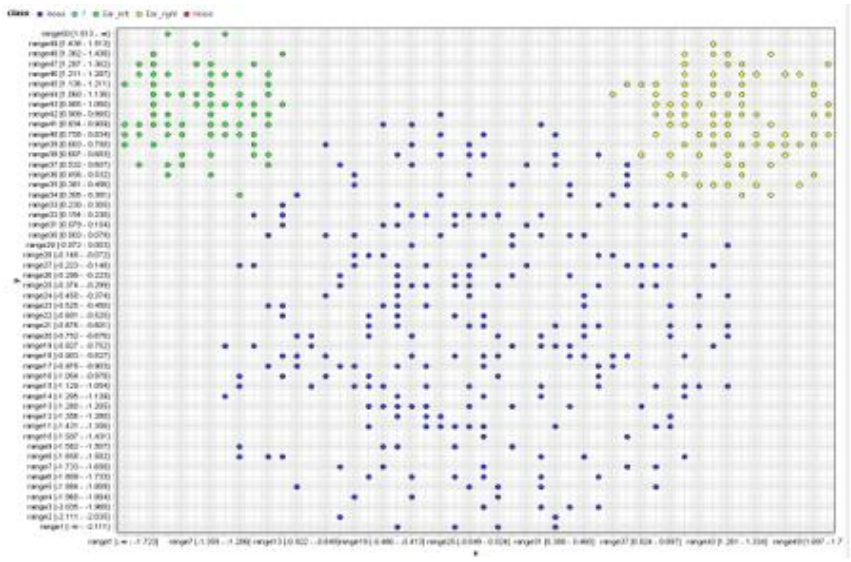


Figure 30 - A scatterplot showing the effect of discretization

3.6 Dimensionality reduction

The number of features/attributes in a dataset is called dimensionality. When the dimensionality of a dataset is too high, the modeling performance might suffer, and the resulting model might become difficult to interpret. An obvious approach to handling high dimensionality is removing features. However, eliminating features arbitrarily might run the risk of losing important information captured by those features. Dimensionality reduction aims at decreasing the dimensionality of a dataset while preserving most of the information captured by the features. Ideally, dimensionality reduction does not hinder the model performance, and generally, the computation efficiency of the model increases. In addition to these benefits, another reason to apply dimensionality reduction is to visualize a dataset, as meaningful spatial visualization can only display up to three dimensions.

Common dimensionality reduction techniques:

- Feature selection (e.g., removing highly correlated features)
- Principal Component Analysis (PCA)

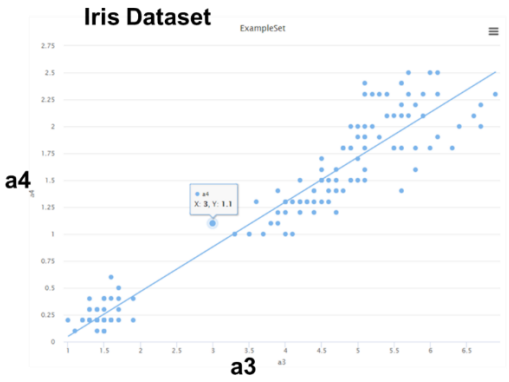
Removing highly correlated features

Video: Removing correlating features with RapidMiner

Correlation is the linear relationship between variables. For example, the time spent running and distance covered correlate well with each other. Therefore, these two features capture some overlapping information, possibly causing redundancy.

Here is a visual example. The scatter plot and the correlation matrix below belong to the Iris Dataset. Notice on the scatter plot that there is a good-fitting trend line that shows almost a perfect linear relationship between the attributes A3 and A4. The correlation matrix shows the pairwise correlation values between all attribute pairs. Pay attention that a variable is in perfect

correlation with itself, therefore the diagonal values are all 1. A correlation value that is close to 1 means a very high correlation.



Correlation Matrix

Attrib...	a1	a2	a3	a4
a1	1	-0.109	0.872	0.818
a2	-0.109	1	-0.421	-0.357
a3	0.872	-0.421	1	0.963
a4	0.818	-0.357	0.963	1

Figure 31 - A scatterplot and a correlation matrix of the Iris dataset

Highly correlated attributes capture redundant information. Therefore, removing some of them would potentially not hinder the model performance, while decreasing the number of attributes. The decision to select which features to remove can be automated by setting a maximum correlation threshold and removing one from each pair of features that correlate with each other higher than the threshold.

Video: Principal Component Analysis in RapidMiner

PCA

Principal Component Analysis (PCA)

PCA is a mathematical operation that transforms a set of data attributes that potentially correlate with each other into a set of uncorrelated values that are called principal components. The number of principal components is less than or equal to the number of original attributes. Unlike the original attributes, principal components do not directly represent a concept or a characteristic of the data. Instead, each principal component is a linear combination of the original attributes. Principal components are orthogonal, thus, they do not correlate with each other. PCA is a very effective way of reducing the dimensionality of the dataset. However, because the resulting principal components are not directly representing data characteristics, the interpretation of the model outcomes becomes more difficult.

Which dimensionality reduction technique to use?

The difference between removing highly correlated features and PCA is the following.

- PCA combines the original attributes into principal components, effectively removing most of the correlation.
- Feature removal does not necessarily remove most of the correlation in the data.
- The outcome of PCA, i.e., principal components are not directly meaningful as they are a combination of the original features.

- The outcome of feature removal is a subset of the original features, therefore, they preserve their original meaning.

The selection of the dimensionality reduction technique should be done on a case-by-case basis, considering the points above.

Practice: Dimensionality reduction using RapidMiner

In RapidMiner, it is possible to define a maximum correlation threshold and remove the variables that correlate with each other two well. To do that, we can use the Remove Correlated Attributes operator. The important parameter to set for this operator is the correlation threshold. The output of this operation is the dataset with reduced dimensionality.

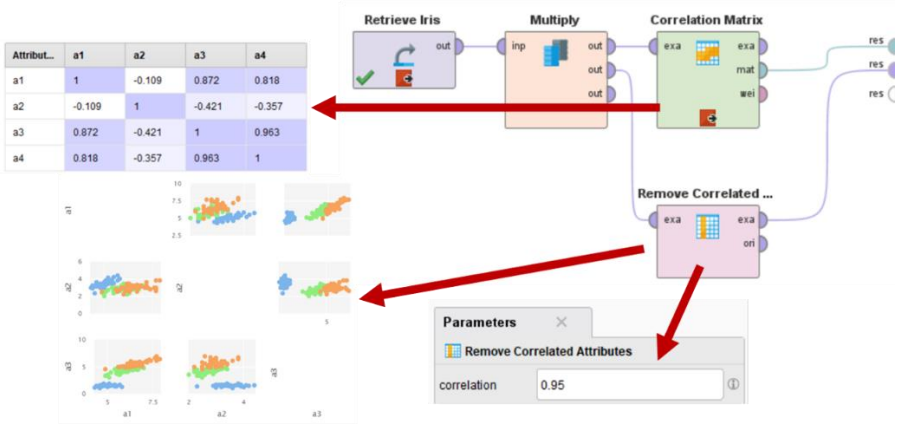


Figure 32 - Removing correlated attributes using RapidMiner

RapidMiner also supports the use of PCA. The PCA operator requires the user to set the desired number of principal components. Alternatively, the user can set a variance threshold to let RapidMiner know how much variance to preserve in the data.

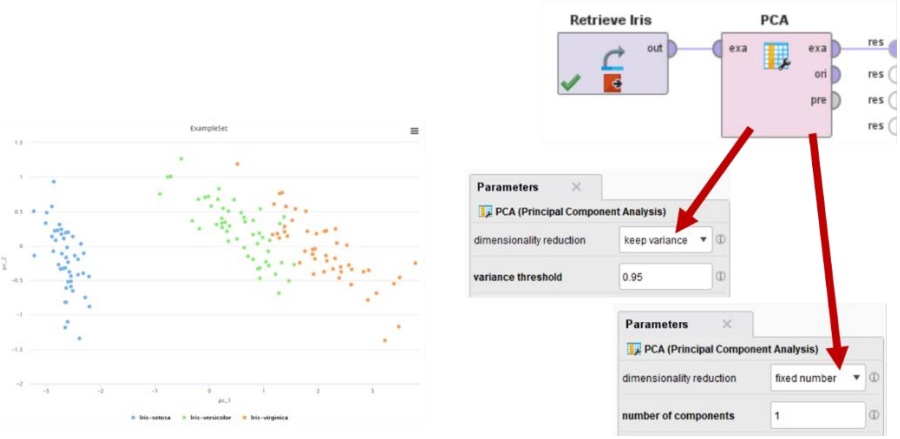


Figure 33 - PCA using RapidMiner

4 Fundamentals of Machine Learning

This chapter provides the theoretical and practical details on various types of machine learning algorithms that are commonly used for data analytics in business.

4.1 Machine learning and data mining

Machine learning is the use and development of systems that can learn and adapt without explicit instructions, by using algorithms and statistical techniques to analyze and draw inferences from patterns in data. The "learning" aspect of machine learning comes from the ability to take example data and use the characteristics and patterns of those data to dynamically create decisions. The process of discovering and evaluating useful patterns in data is called Data Mining. Therefore, there is a semantic overlap between the concepts of data mining and machine learning, as they both rely on the use of data to assist manual or automated decision making.

Major types of learning

- Supervised learning
- Unsupervised learning

Other types

- Semi-supervised learning
- Self-supervised learning
- Reinforcement learning

In the last decade, machine learning has become a very dynamic field in which significant changes and advances take place often. For this reason, many different categorizations of machine learning exist in practice and literature. However, the fundamental categories remain the same both in terms of context and rationale of operation.

Two major types of machine learning are *supervised* and *unsupervised* learning. In supervised learning, there is a target, and the training process requires labelled data. On the other hand, in unsupervised learning, there is no target. Thus, there is no need for labelled data in the training process. Instead, unsupervised learning algorithms find the naturally occurring patterns in the data and put data records in different categories based on their characteristics.

4.2 Descriptive, predictive, prescriptive analytics

Figure 34 shows a broad categorization of the purposes of machine learning. These differ in terms of the value they provide and the difficulty of implementation.

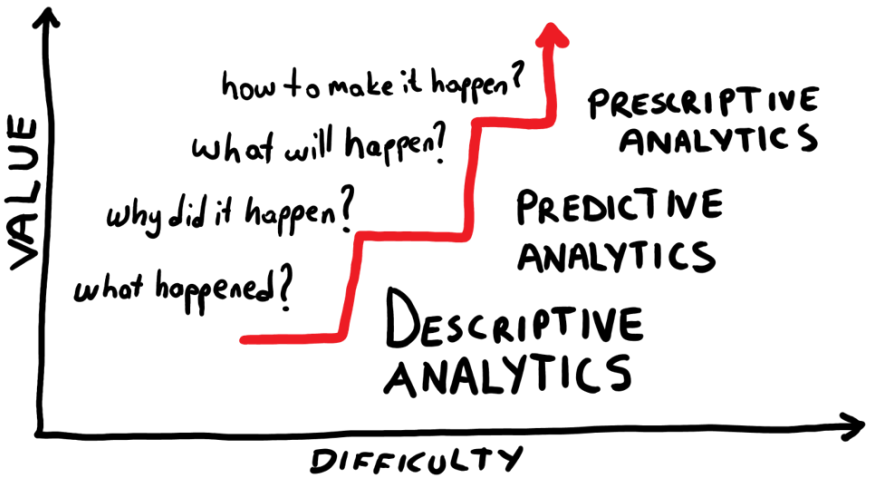


Figure 34 - Types of analytics based on different purposes

Descriptive analytics focuses on historical data to help understand what happened in the past. Companies use descriptive analytics to understand a situation or a process that took place. Descriptive analytics techniques generally do not require complex models and techniques. Frequently encountered examples of descriptive analytics include business intelligence dashboards and data visualization.

Predictive analytics utilizes algorithms that learn. Predictive analytics uses historical data to train machine learning models, and in turn, yields predictions to support decision-making. Churn models are excellent examples of predictive models. They predict which customer will leave the company as a client and who will stay as loyal clients. Another example is the recommender systems that predict which action will an individual take next.

Prescriptive analytics builds upon the foundations of predictive analytics. In other words, predictive analytics tell what is going to happen next, and prescriptive analytics tell which actions are necessary to change the course of events and reach a particular goal.

4.3 Prediction models

Prediction models are used to predict something that is not yet known. Examples could vary greatly from predicting the weather or the next actions

of a customer that visits the company website. It is important to understand that prediction is done on data that is outside the training dataset. The data records that are included in the training set are already known by the model, thus, they do not require to be predicted.

The training of prediction models is essentially the same as the training process of any other supervised machine learning model. Figure 35 provides an example of the training and utilization of a prediction model. The dataset consists of green and red objects. A portion of the data is collected and used as the training set. Ideally, the training set contains a balanced distribution of red and green objects. When the model is trained, it can be used to predict whether a previously unknown object is red or green.

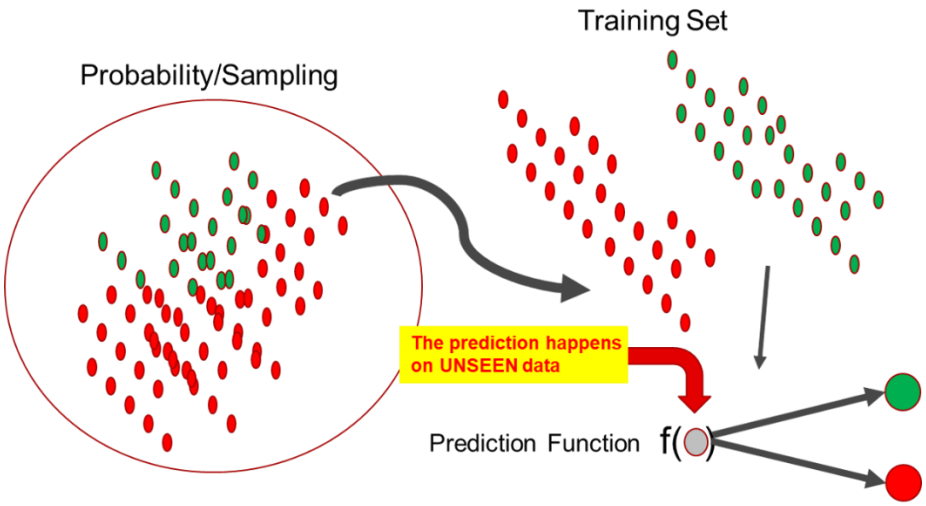


Figure 35 - Depiction of the training and utilization of a model

Figure 36 shows a generic schema of a model, which also applies to prediction models. Prediction models embody a learning algorithm that is trained on the provided dataset. Learning algorithms have parameters that configure their operations; thus, they play a role in determining the behaviour of the prediction model. Trained prediction models take unseen data as input, and they provide predictions as output. As no model is perfect, prediction models also make mistakes. It is important to know what kinds of mistakes a prediction model tends to make because each mistake may have different consequences.

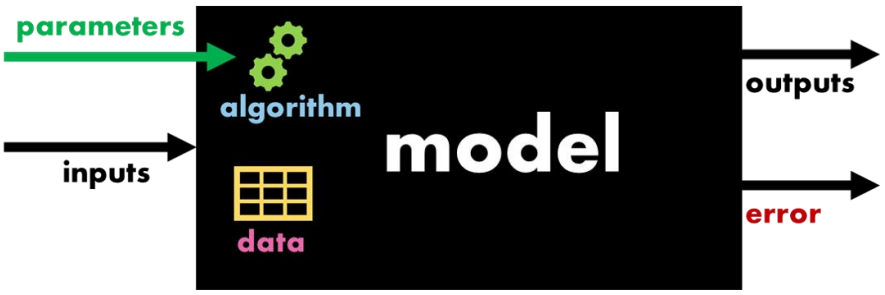


Figure 36 - Generic schema of a model

4.4 Performance evaluation: confusion matrix

Confusion Matrix

There are many ways to evaluate the performance of a prediction model. Using a confusion matrix is one such way. Confusion matrices show how many correct and incorrect predictions a prediction model is making, as well as the types of correct and incorrect predictions. Figure 37 depicts a confusion matrix and shows what each part of the confusion matrix indicates.

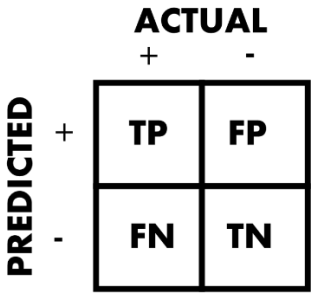


Figure 37 - Depiction of a confusion matrix

The y-axis shows the predicted classes, and the x-axis shows the actual classes. This example has two possible classes; positives (+) and negatives (-). However, a confusion matrix can also be applied in scenarios that have more than two classes. If an unknown data record is predicted to be positive, and it is actually positive, it means that the model has predicted it correctly. This is considered a True Positive (TP) case. Similarly, if the unknown data record is predicted to be negative, and it is actually negative, it means the model correctly predicted a negative case. Thus, it is considered a True Negative (TN) case. A model is expected to make prediction mistakes. If the model predicts an unknown data record as positive but it is actually negative, then the prediction is considered a False Positive (FP) case. Similarly, if the model predicts a data record as negative but it is actually positive, then the prediction is considered a False Negative (FN) case.

A confusion matrix enables the calculation of several performance metrics. **Precision** is the number of TP divided by the number of all positives. **Recall**

is the number of TP divided by the sum of TP and FN. Finally, the **F-1 measure** can be calculated as the harmonic mean of Precision and Recall.

TP: True Positive
FP: False Positive
TN: True Negative
FN: False Negative

- Precision = TP / (TP+FP)
- Recall = TP / (TP+FN)
- F-1 Measure = 2 x Precision x Recall / (Precision+Recall)

4.5 Training machine learning models

Training and testing data separation: Splitting the data into separate portions for training and testing a machine learning model is important to be able to evaluate the performance of a model in a fair way. Separating the dataset into training, validation, and holdout portions is considered a good practice (see Figure 38).

The *training* set is the part of the data from which the model learns the patterns. In other words, the model learns the relationships between the features and the target variable based on the training data. The *validation* set is used to evaluate the model performance evaluation while hyperparameter tuning. The *holdout* set is the portion that is separated from the dataset at the beginning and is never used for training or parameter tuning purposes. The holdout set is only used at the end to get a fair measurement of the model performance. Once the evaluation of mode on the holdout data is good enough, the model is deployed.

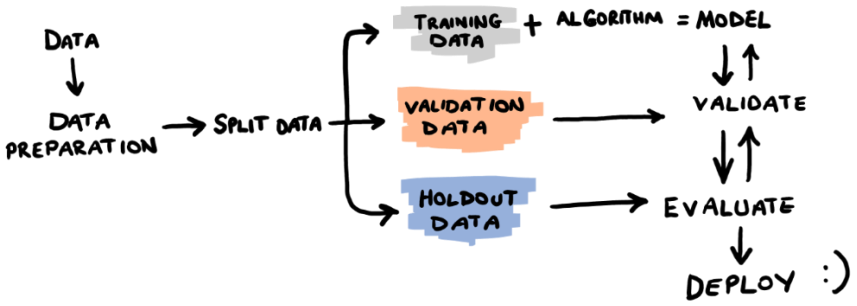


Figure 38 - Data splitting for training, validation, and holdout

There are no hard rules that dictate how to arrange the portions of the data. In most cases, the larger portion is used as training data, and the smaller portions are used as validation and holdout sets (see Figure 39).

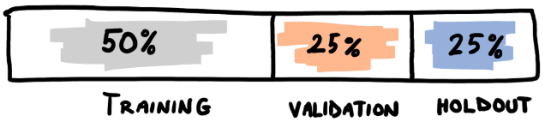


Figure 39 - Example portions of data splits

Cross validation: There is another approach for performance evaluation that is built on the same logic of separating the training and testing sets: Cross validation. It is also called k-fold Cross validation due to the iterative nature of the method. k-fold cross validation divides the data into k portions of equal size. In every iteration, it uses one portion for testing and the rest as training data. Cross validation performs this k times, shifting the testing portion so that every portion becomes the testing data once. At the end of each iteration, there is one performance measurement. After the completion of k iterations, the mean of all performance scores constitutes the overall performance score (see Figure 40).

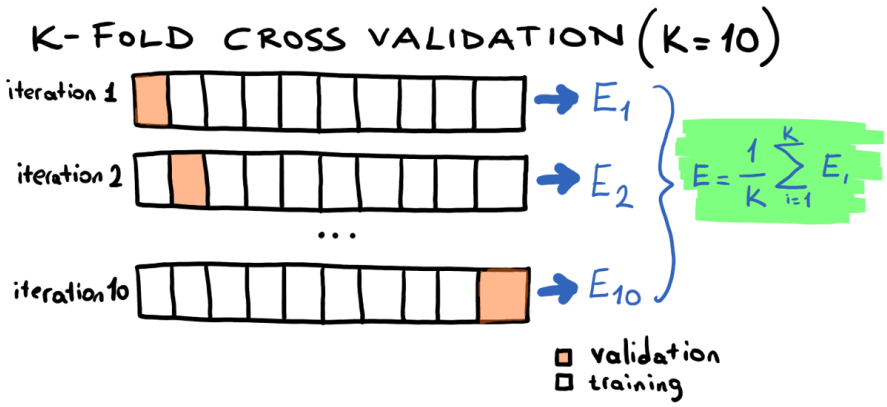


Figure 40 - K-fold cross validation

When to use cross validation and the holdout approach?

In the holdout method, the data is split only once. The portion that is reserved for testing is only used once, at the end. Therefore, the opportunity of training the model on the entire dataset is lost. However, in cross validation, the entire dataset is used for training while still evaluating the model in a fair way.

Thus, if the dataset is large enough that losing some training data is acceptable, then the holdout approach works well. In other cases, cross validation can be used. However, due to the iterative nature of the method, cross validation is computationally heavier.

4.6 Generalizability of models

Model Generalizability

Sometimes models that perform very well in training suffer when applied to unseen data. This indicates a serious problem that hinders the generalizability of the models and decreases their value. A common reason for this issue is that the model became too complex, so much fine-tuned on the training data distribution that it does not do well in general use.

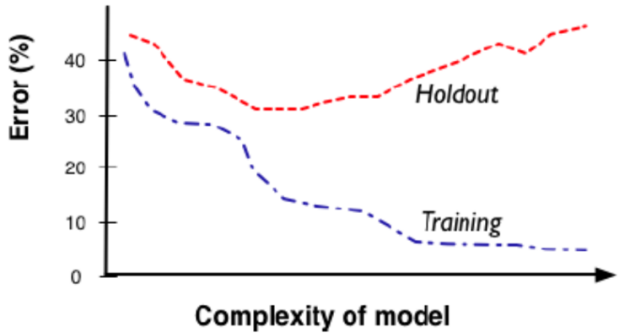


Figure 41 - Error rates for different data portions due to increasing model complexity

Figure 42 depicts three model-fitting scenarios demonstrating the increased complexity and performance. The model on the left-hand side is underfitting. Note that the model consists of four linear separators that can be easily expressed as a few formulas. However, due to the simple nature of this model, it does not separate the data very well, resulting in many misclassifications. The model on the right-hand side is overfitting. It consists of very complex curves that perfectly separate the dataset. However, when the model is applied to another dataset, the overcomplexity will potentially cause the performance to drop. Finally, the figure in the middle shows an appropriate fitting. Expectedly, there are some misclassifications but the model seems to be separating the data in a way that it captures the essential differences in the data distribution.

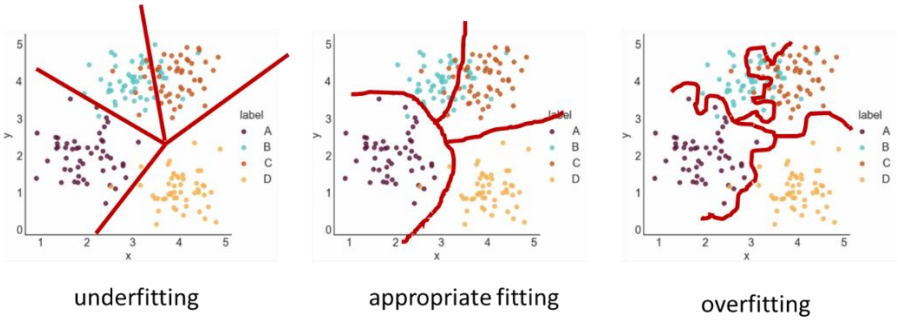


Figure 42 - Different model-fit scenarios

4.7 Entropy and information gain

In information science, *Entropy* is defined as the measure of disorder. Entropy in a dataset can be calculated using the formula below.

$$Entropy(D) = \sum_{i=1}^{C} -p_i log_2(p_i)$$

Consider Figure 43. If a population contains all (-), then the probability of drawing a (+) among that population is zero. Such a scenario has no surprises, no disorder. Therefore, the entropy is zero. The more balanced the attributes are distributed in the population, the higher the entropy. The entropy is the highest when all attributes are distributed uniformly. For instance, when 50% of the population is (+) and the rest is (-), the entropy of the data is 1.

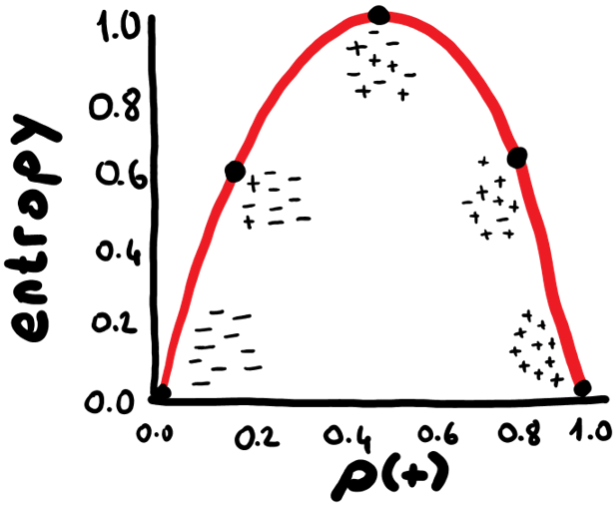


Figure 43 - Entropy in different data distributions

A related concept, *Information Gain* is the difference between two entropy levels. Information Gain is used to understand how much an attribute in a dataset provides information with respect to the target.

4.8 Algorithm families

It is important to understand the goals and characteristics of certain types of algorithms to effectively transform the business problem into a data analytics problem. This course covers three main machine learning algorithm families.

- Classification
- Regression
- Clustering

Classification and Regression are considered supervised methods while Clustering is unsupervised.

Classification: If the business problem requires the separation of individuals into groups, and those groups can be represented by meaningful labels, then it is highly likely that this problem can be tackled using classification algorithms.

Examples:

- Predicting which clients will stop working with the company, and who will stay loyal. In other words, classifying customers into two categories: churn / loyal
- Identifying the risk category of credit applicants by separating them into three groups: high-risk, medium-risk, and low-risk.
- Detecting fraud attempts in financial transactions by separating transactions into two groups: normal transactions and red flags.

What these examples have in common is that, the business problem requires grouping, and that each group can be associated with a label. The labels that represent the groups are called **classes**. Therefore, the task of predicting which class an individual data record belongs to is called classification. Note that in **classification**, the target classes are categorical.

Regression: While classification aims at predicting a discrete class, the goal of Regression is to predict a continuous value.

Examples:

- Predicting the price of a house given its characteristics.
- Calculating the (numerical) risk score of a credit applicant.
- Predicting the price of a financial asset in the future.

Both classification and regression are cases of supervised learning. They both aim at predicting a target variable. In classification, the target variable is a class (categorical), in regression the target to be predicted is continuous.

Clustering: Clustering is the task of separating the data into groups that do not have predefined labels. Instead, clustering algorithms identify similarities between the data records and separate them into clusters according to their similarities and differences. Clustering problems are generally handled using unsupervised learning methods and they do not require a labeled dataset to be trained.

Examples:

- Customer segmentation: Exploring different segments in the customer base due to the characteristics and behavior of individuals.
- Recommending a product or a service: Providing a product or a service recommendation based on the activities of similar individuals in the data.

Notice that in these examples, the problem requires evaluating the similarities and differences among individuals in the population. The individuals that are assigned to a cluster are similar to one another in terms of data attributes, and they are somewhat different than the individuals in the other clusters.

The outcome of clustering is a set of clusters. It is important to realize that these clusters have no predefined meanings. In order to have a meaningful analysis, clustering outcomes need to be interpreted. For example, assume a customer segmentation analysis yields three clusters; cluster 1, cluster 2, and cluster 3. Inside each cluster are the individuals that are similar to each other in terms of their characteristics. At this point, the analyst should interpret the clusters, and perhaps, assign them names that represent their essential behavior differences. Such as, cluster 1: early adopters, cluster 2: majority, and cluster 3: laggards.

5 Classification

This chapter covers supervised machine learning algorithms for classification and the techniques to evaluate the performance of classification.

5.1 Decision trees

Decision Tree

Information Gain is essential for certain algorithms such as Decision Trees. The decision Tree algorithm calculates Information Gain for each attribute and creates the branches of the tree accordingly. A simple definition of the algorithm steps is as the following.

1. Calculate the entropy of the dataset
2. Select the attribute that yields the maximum Information Gain and create the next branch using that attribute.
3. Go one level down
4. Repeat until the entropy of the dataset is zero or the maximum tree depth threshold is exceeded.

Video: Decision trees in RapidMiner

Practice: Decision trees with RapidMiner

Here is a simple implementation of Decision Trees in RapidMiner which also demonstrates how decision trees use Information Gain. Note that for the sake of simplicity no train-test split is applied in the example. Thus, this is not an ideal way of using data to train machine learning algorithms.

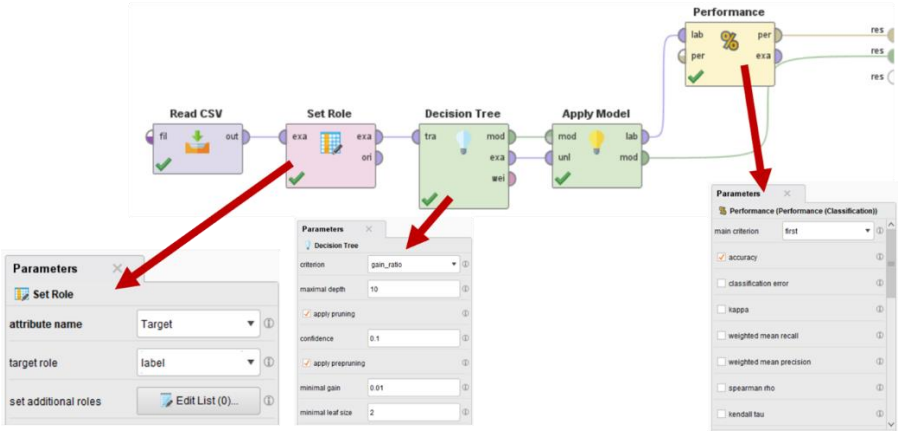
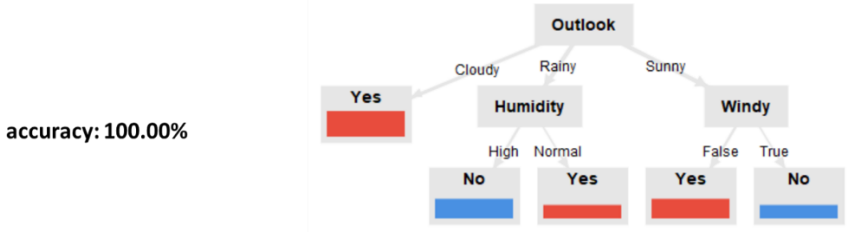


Figure 44 - Decision trees with RapidMiner

The first operator is Read CSV, that unsurprisingly reads a CSV document and loads the data as an Example Set in RapidMiner. Then Set Role operator is used to indicate which attribute is the target. Consecutively, the Decision Tree operator is used in combination with the Apply Model operator to train the decision tree model. Finally, the Performance operator is used to evaluate the model performance. Once again, note that the training and testing split of the data is not done in this example. Therefore, the model performance will not be realistic.

The figure below visualizes the trained Decision Tree. The performance of the model seems to be 100%, indicating a perfect model. As stated before, this is an unrealistic measure of performance because the same data is used to train and test the mode in the example.



	true No	true Yes	class precision
pred. No	5	0	100.00%
pred. Yes	0	9	100.00%
class recall	100.00%	100.00%	

Figure 45 - Decision tree and the performance report

5.2 K-nearest neighbours (KNN)

KNN

KNN stands for K-Nearest Neighbours, and it is a classification algorithm that predicts the class of a data record based on the most similar data records in the training set. As data records can be expressed as vectors, the similarity between them can be calculated using vector operations. Data records that are similar to one another appear closer in Euclidean space, and the ones that are not so similar appear apart from each other. Thus, the similarity can simply be expressed as the distance between data records in Euclidean space.

KNN is a classification algorithm that is trained using supervised learning principles. First, labelled training data is provided to the algorithm. KNN creates a vector space that contains all data records in the training set. When an unseen data record is given as input, KNN places that data record in the vector space based on its attributes. Subsequently, KNN calculates the pairwise similarity between the new data record and all other data records inside the vector space and identifies K data records that are the most similar to the new data record. These K data records are called the nearest neighbours. Because the training set is labelled, each of these K nearest neighbours has a class associated with it. Finally, KNN examines the classes

of the K nearest neighbours and applies majority decision to decide which class to assign to the new data record.

To understand how KNN works, consider Figure 46. The (+) and (-) depict the classes of each data record in the training set. The red dot is the unseen data record, and KNN aims at predicting which class it belongs to. The figure on the left-hand side shows the vector space and the classes of the K nearest neighbours when K is set to 1. The closest neighbour is a (+), thus, the new data record is also assigned to the (+) class. The plot in the middle shows the same vector space when K is set to 3. This time, KNN considers the closest 3 neighbours while deciding which class the new data record belongs to. The nearest 3 neighbours belong to (+), (+), and (-). Again, the majority decision yields (+) as the class of the new data record.

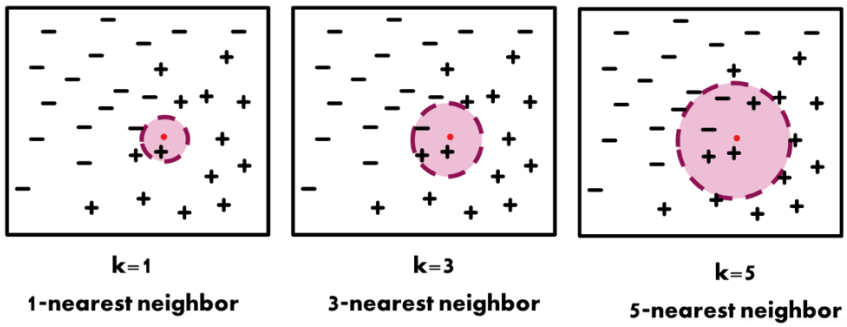


Figure 46 - Visual representation of KNN for different values of K

The pairwise distance between two vectors can be calculated using the following formula. Similar data records have a smaller distance from each other.

$$d(p,q) = \sqrt{\sum_i (p_i - q_i)^2}$$

KNN requires the parameter K to be set so that the algorithm knows how many of the nearest neighbours to consider when deciding which class the new data record belongs to. The selection of this parameter needs caution. If K is a too small number, then the classification might be sensitive to noise in the training data. If K is too large, then there is a risk of including too many data records which belong to the nearby groups of data records that are the members of a different class.

Characteristics of KNN:

- KNN uses Euclidean distance, thus it is sensitive to data attributes that are on different scales. As with all algorithms that use the notion of distance for classification, feature scaling, i.e., normalization,

should be applied. Otherwise, the distance measurement might be dominated by attributes that have bigger values.

- KNN is a lazy learner. It means, the training process merely consists of storing the data. Each time a new data record is classified, KNN calculates the pairwise distances between the training data records and the new data item. Thus, KNN takes less time learning, and more time classifying. This characteristic of KNN makes it a computationally heavy algorithm.

Video: KNN in RapidMiner

Practice: KNN using RapidMiner

The figure below shows an example of a classification workflow using KNN. In RapidMiner, KNN is implemented using the k-NN operator. This operator requires several parameters to be configured. K denotes the number of nearest neighbours to consider in the classification task. Measure types and measure parameters determine which distance calculation method to be used by the algorithm.

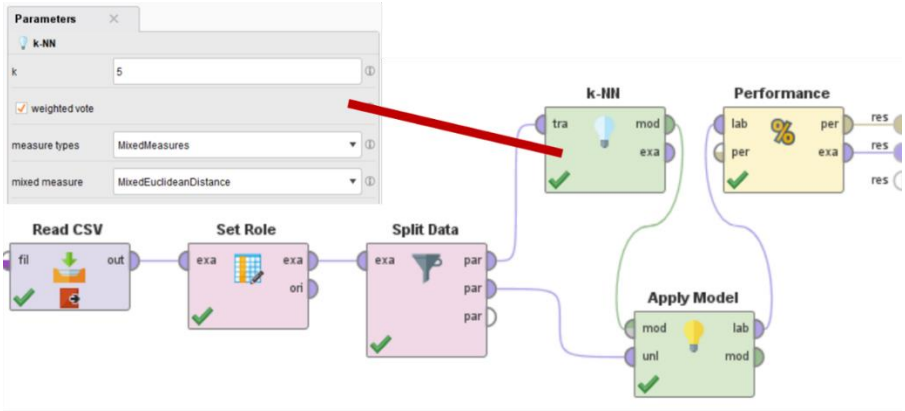


Figure 47 - KNN using RapidMiner

5.3 Classification performance

Classification performance of machine learning models can be evaluated using a multitude of metrics. The most common metrics are the following.

- Confusion matrix
- Precision, recall, F-1 score
- AUC-ROC curve
- Cost/benefit reasoning

Confusion matrix:

The figure below exemplifies a binary confusion matrix.

The y-axis shows the actual classes, and the x-axis shows the predicted. The classifier in this example is a spam email detection model. The model is trained to classify incoming emails as spam or not spam. Expectedly,

sometimes, the classifier makes mistakes and misclassifies the emails. A confusion matrix shows how often the model makes a certain kind of mistake, and how often it classifies correctly.

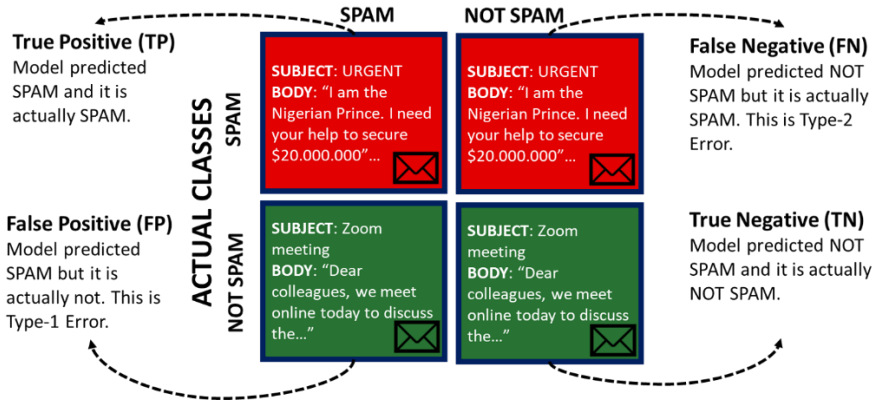


Figure 48 - Confusion matrix example

In the layout of this confusion matrix, the upper-left cell shows the instances that are actually spam correctly predicted as spam. These cases are called true positives. The lower-right cell contains the second type of correct prediction. These are true negatives where the emails are actually not spam and detected correctly as not spam.

The bottom-left cell shows the emails that are actually not spam but misclassified as spam. These cases are false positives, and this type of error is also called Type-1 Error. In the top right cell, there are the emails that are actually spam but are misclassified by the model as not spam. These are false negatives, or, Type-2 errors.

How to use confusion matrices for model selection?

A confusion matrix shows how often a model makes certain types of errors. Depending on the business problem, each type of error has a different consequence and impact. Thus, confusion matrices provide essential information that can be used for model selection.

The figure below shows the confusion matrix of a COVID-19 prediction model. Note that every outcome of the model has a different scenario. Specifically, the mistakes have consequences with severely different costs. This confusion matrix shows that the model yields 35 true positives and 40 true negatives out of 100. Thus, the accuracy of the model is 0.75. The confusion matrix also shows that the model makes false positive classifications 15 times, and false negative classifications 10 times.

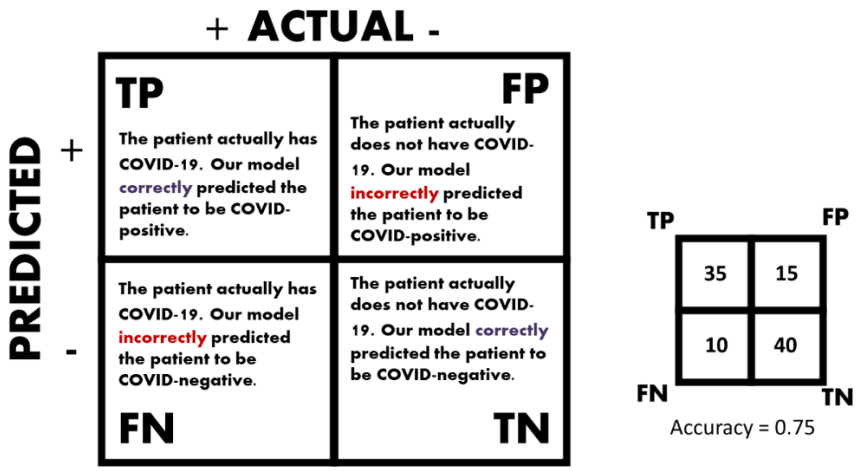


Figure 49 - Confusion matrix and accuracy calculation example

Consider the confusion matrices which belong to two different COVID-19 prediction models. Model A has an accuracy of 0.75 while the Model B has 0.80. It seems logical to select the model which yields the better accuracy. However, model selection based on accuracy alone could be misleading. Take a look at the frequency of different types of errors while also considering the impact of different errors. Try to interpret the confusion matrices and select the model that provides the best utility.

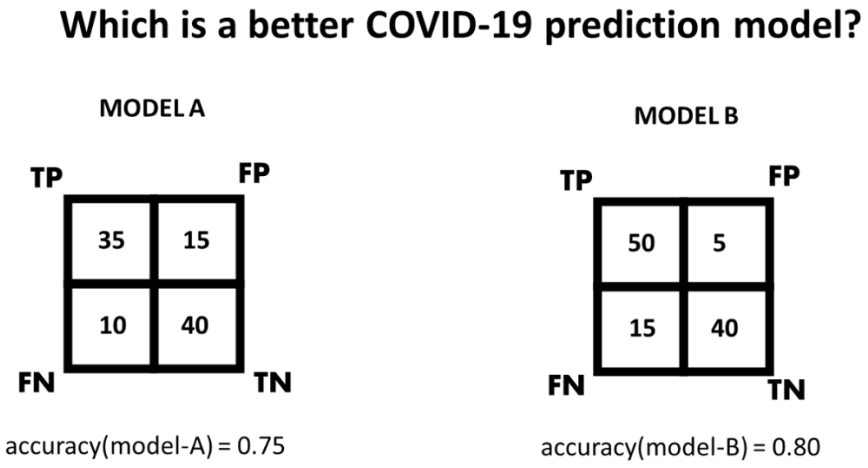


Figure 50 - Model selection based on confusion matrices

A comparison between the two confusion matrices shows an important point. Model B is making many more false negatives than Model A. In the context of COVID-19 prediction, a false negative means that the model is failing to detect a sick person and incorrectly classifies the individual as healthy. As a consequence, the sick individual walks around and infects other people. The effective selection of the model that yields the highest utility requires the

following question to be answered. Are the costs incurred by extra false negatives worth the 0.5 improvement in the overall accuracy?

Multiclass confusion matrix

Confusion matrices can also represent models that have more than two classes. The usage of multiclass confusion matrices is essentially the same. The figure below shows a multiclass confusion matrix.

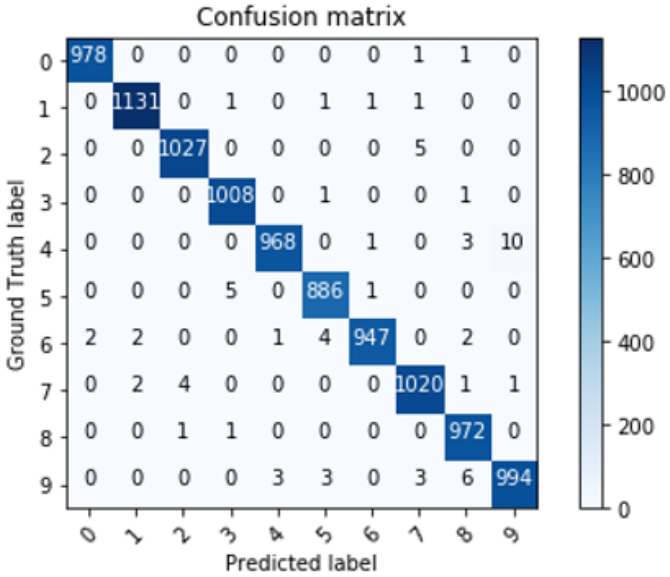


Figure 51 - Multiclass confusion matrix example

Precision, Recall, and F1 Score:

Precision, Recall, and F1 Score are commonly used performance metrics. They can be calculated using the values in a confusion matrix.

The figure below demonstrates these metrics visually. The rectangle contains all the data. The circle in the middle covers the data that are classified as positives by the model, some are true positives, others false positives.

The precision of the model is the number of true positives divided by the number of all positives. Thus, precision indicates how many selected items are relevant. The recall is the number of true positives divided by the sum of true positives and false negatives. In other words, recall shows how many relevant items are selected. Finally, F1-score is calculated as the harmonic mean of precision and recall.

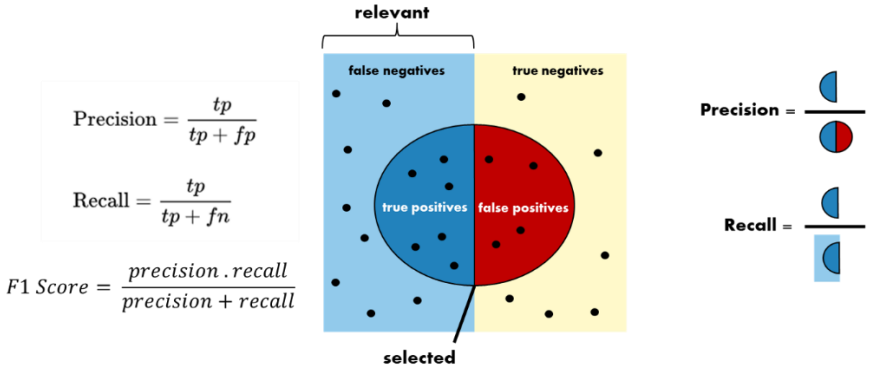


Figure 52 - Precision, recall, and F-1 score

Expected Value (Cost/Benefit Reasoning):

Video: Cost performance analysis in RapidMiner

Every misclassification outcome in the confusion matrix has a different cost, and every correct classification has a different benefit. A cost/benefit matrix helps calculate the expected value of an overall model. Expected Value is among the most important performance metrics for model selection. It is possible, and in fact not uncommon, that a model which performs better than others in terms of F1 score, offers a lower expected value due to the cost of mistakes it makes.

Practice: Expected value calculation with RapidMiner

In RapidMiner, Expected Value calculation is implemented using the Performance (Cost) operator. This operator requires a cost matrix to be provided. The figure below shows the workflow and the parameter configuration for Expected Value analysis.

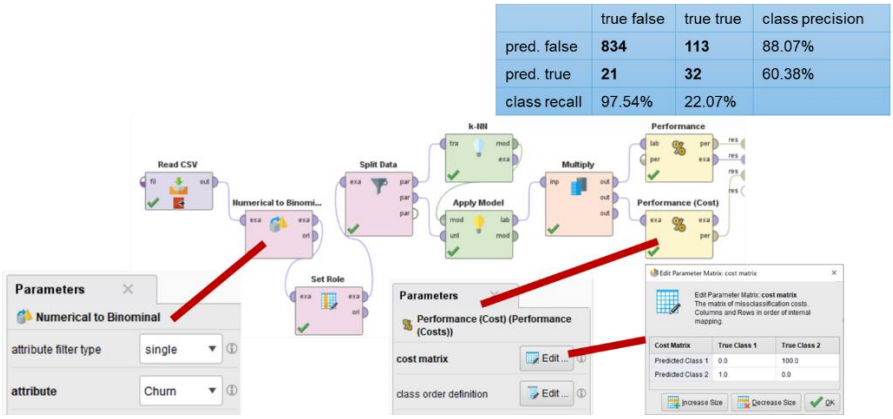


Figure 53 - Expected value calculation with RapidMiner

6 Regression

This chapter addresses regression; a supervised learning algorithm family in which the target is numerical and continuous. Additionally, this chapter introduces statistical tendency concepts such as mean, variance, and standard deviation as they are used to define the objective functions of regression models.

6.1 Linear Regression

$$y = mX + B$$

Linear Regression

The formula above represents a linear relationship between two variables. It can be visualized as a line in space as shown in the figure below. X and Y are the variables. m is the constant weight that is the slope of the line that shows the relationship. B is the offset constant.

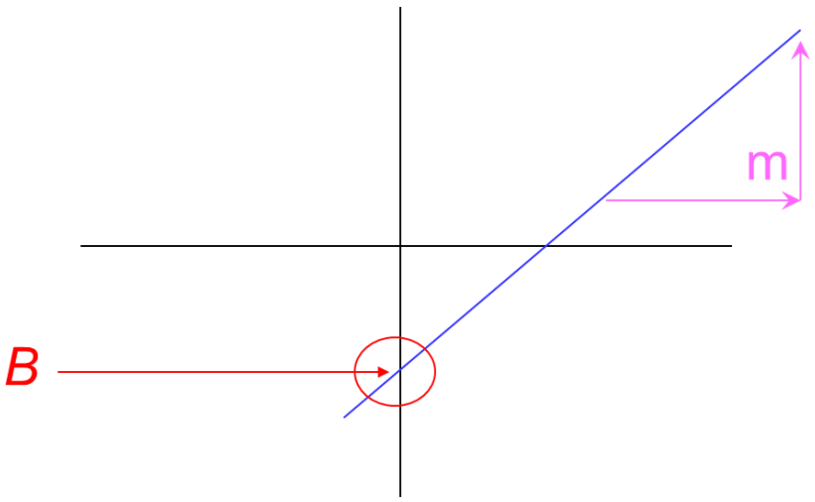


Figure 54 - Linear regression formula

The figure below shows two variables. The scatterplot helps visually examine the relationship between the two attributes of the dataset, namely, horsepower and miles per gallon (mpg). The trend line represents the best line that can describe the relationship.

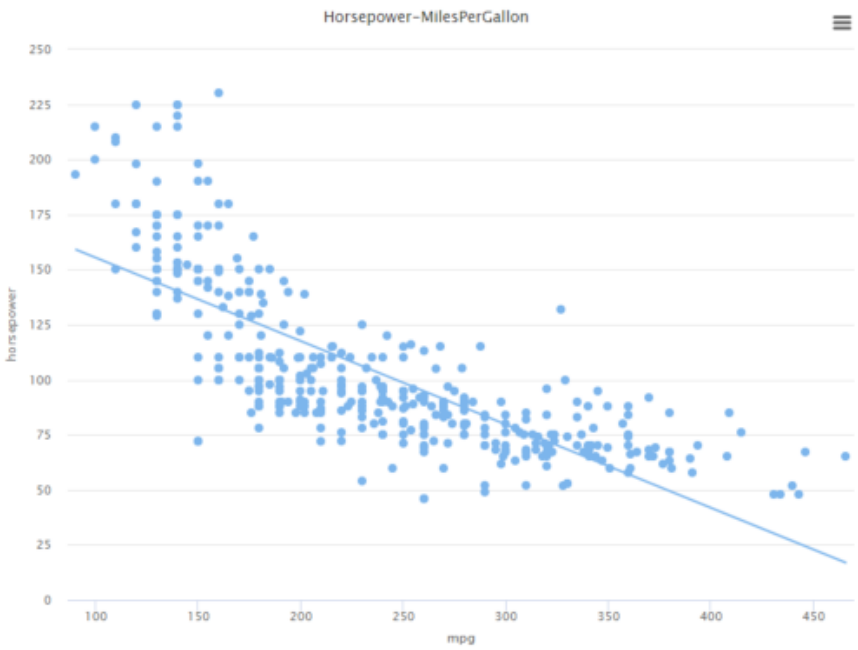


Figure 55 - Visual example of linear regression

The task of finding the line that best represents the relationship between attributes is called linear model fitting. To perform linear model fitting, an objective function must be defined. An example of an objective function could be minimizing the sum of errors. Some of the commonly used objective functions are covered in this learning unit. The objective functions use statistical tendency measures; hence, a list of central tendency measures is also provided.

Statistical central tendency measures

Mean: The mean average of a set of values x =$(\mathbf{x_1}, \mathbf{x_2}, \ldots, \mathbf{x_n})$ can be expressed as:

$$mean = \left(\frac{1}{N}\right) * \sum_{i=1}^{n} x_i$$

The mean of a data set is important for understanding what the central tendency is of the population concerned. It is a very general concept with applicability in many fields. It provides a very first characterisation of a process and in the case of regression it will usually give the value which is equidistant to any other value.

Median: Median is the value that divides the dataset in half: half of the values are bigger than the median, the other half are smaller. The median value is often useful for understanding whether a dataset is balanced or if it is skewed towards small or bigger values.

Variance: The variance of a data set of values $\mathbf{x} = (x_1, x_2, \ldots, x_n)$ can be understood to be the distance that the points in the dataset are from the mean, or the deviation that the dataset has from the mean. This can be expressed as

$$variance = \left(\frac{1}{N}\right) * \sum_{i=1}^{n} (x_i - mean)^2$$

The variance gives a measure of the spread of the dataset with respect to the mean value.

Standard Deviation: The standard deviation (STD) is defined as the square root of the variance. Specifically, one can express it as:

$$STD = \sqrt{variance}$$

The immediate interpretation of a STD is that a small STD means that your data are close to the *mean,* while larger STDs imply that your data is sparser. It has the same interpretation as the variance, but with the advantage that its unit is the same as that of the original data.

6.2 Common objective functions for regression models

Mean Absolute Error (MAE): The mean absolute error is a typical metric to calculate the error in a time series analysis. If $x = (\mathbf{x_1}, \mathbf{x_2}, \ldots, \mathbf{x_n})$ is the original vector of values and $x' = (\mathbf{x'_1}, \mathbf{x'_2}, \ldots, \mathbf{x'_n})$ is the predicted vector, the MAE can be expressed as:

$$MAE = \left(\frac{1}{N}\right) * \sum_{i=1}^{n} |x_i - x'_i|$$

This can be basically understood as an average of the error produced by the prediction in the time series or sequence that we are trying to regress. MAE provides a first understanding of how well the regression is performing, but in itself this is not enough as it does not take into account the effect of variance and outliers.

Mean Squared Error (MSE): Similar to the MAE, the MSE measures how well the regression algorithm is approximating the sequence of points in the trend we are trying to analyse. If $x = (\mathbf{x_1}, \mathbf{x_2}, \ldots, \mathbf{x_n})$ is the original vector of values and $x' = (\mathbf{x'_1}, \mathbf{x'_2}, \ldots, \mathbf{x'_n})$ is the predicted vector, the MSE can be expressed as:

$$MSE = \left(\frac{1}{N}\right) * \sum_{i=1}^{n} (x_i - x_i')^2$$

Like the MAE, the MSE is a measure of how near the calculated result is to the real values. Other than the MAE, the MSE considers the average of the squares of the errors. As such, it models the variance of the errors of the estimator with respect to the real values.

Root Mean Squared Error (RMSE): The RMSE is a measure that is closely related to the MSE, since it is its square root. If $\mathbf{x} = (x_1, x_2, \ldots, x_n)$ is the original vector of values and $\mathbf{x'} = (x_1', x_2', \ldots, x_n')$ is the predicted vector, the RMSE can be expressed as:

$$RMSE = \sqrt{\left(\frac{1}{N}\right) * \sum_{i=1}^{n} (x_i - x_i')^2}$$

As the square root of the MSE, which is related to the variance of the estimator/predictor, the RMSE is strongly related to the standard deviation of the estimator/predictor with respect to the data being analysed. The smaller the RMSE, the closer a regression model is to the real trend being studied.

Median Absolute Deviation (MAD): The median absolute deviation, other than the previous two metrics, focuses on calculating the median error in the time series. It is less used than MAE and MSE, but is still interesting because it is robust with regard to outliers. Since it reports the median error, the outliers will not influence this particular metric, so a combination of RMSE and MAD can tell us whether the dataset has many outliers, since a big RMSE with a small MAD would mean that the regression algorithm is yielding many outliers. The MAD can be expressed as:

$$MAD = median(|x_1 - x_1'|, |x_2 - x_2'|, \ldots, |x_n - x_n'|)$$

where median(…) is an operator that selects the median value in a vector of values.

Multiple Linear Regression

The following figure visually exemplifies the notion of errors in linear model fitting. Error, in this context, can be defined as the distance between the actual data points and the line. If MSE is used as an objective function, then the effect of larger errors would be higher. In other words, if the error gets bigger, the effect becomes exponentially larger. However, MAE considers the effect of the error exactly as the absolute magnitude of the error, thus, not penalizing the large errors.

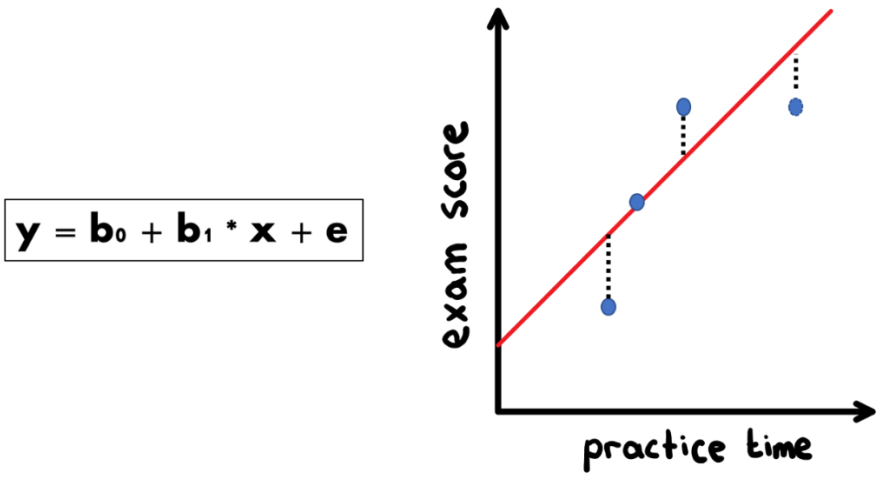


Figure 56 - Visual depiction of errors in a linear regression model

Video: Linear regression in RapidMiner

Practice: Linear regression in RapidMiner

In RapidMiner, Linear Regression is implemented using the Linear Regression operator. This operator requires a target/label role to be set in the dataset. The performance of Linear Regression models can be measured using the Performance (Regression) operator. This operator allows the use of various objective functions as performance evaluation criteria.

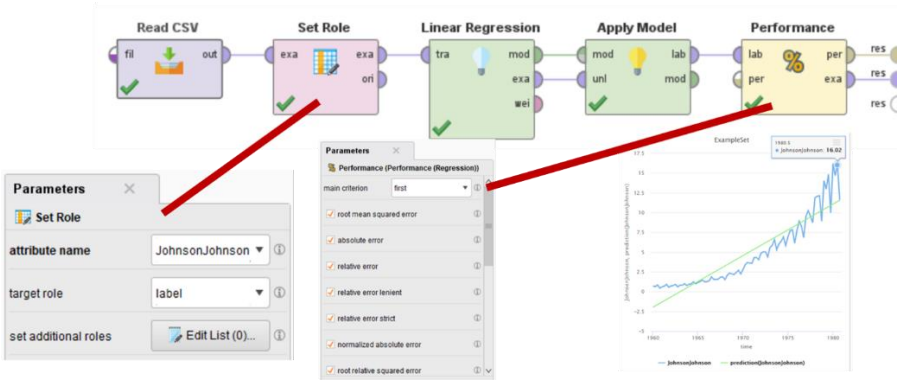


Figure 57 - Linear regression in RapidMiner

7 Clustering

This chapter introduces the unsupervised machine learning techniques to perform clustering and find groupings in the data without the need for an explicit label.

7.1 Clustering algorithms

Hierarchical Clustering

Clustering refers to a set of algorithms that find groupings of data, based on the notion of similarity and distance. The operation principle of clustering algorithms is that the data records that are similar to one another appear in the same or closer clusters, while the others appear in far apart clusters.

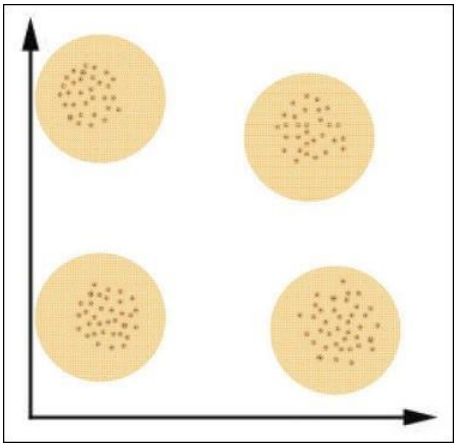


Figure 58 - Visual exemplification of clustering

The main types of clustering algorithms are visually exemplified in the figure below. These are hierarchical clustering, objective function-based clustering, density-based clustering, and grid-based clustering.

Objective Function-based Clustering

- Hierarchical Clustering: This type of clustering algorithms group the data in a hierarchical manner. An example of this type of clustering is Dendrograms.
- Objective Function-based Clustering: This algorithm type forms the clusters based on the objective functions. Thus, the algorithm

Grid-based Clustering

attempts to minimize or maximize a function with the way how the data is clustered. An example of this type of clustering is K-means.

- Density-based Clustering: This type of clustering algorithm use the density of the data distribution to create the clusters. An example is DBScan.

- Grid-based Clustering: This type of algorithm, divide the Euclidean space into a grid structure, and merge the cells to form clusters.

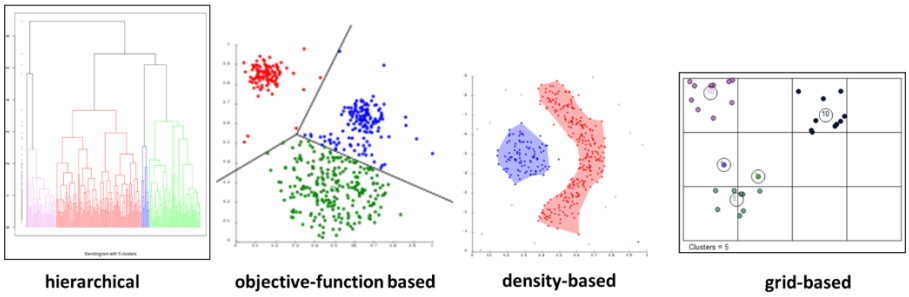


Figure 59 - Main types of clustering

7.2 Clustering performance evaluation

The notion of performance in the context of clustering models relates to how well the clusters are formed. Ideally, clustering models put similar entities in the same cluster and not-so-similar ones in distant clusters. There are many ways to measure clustering performance. The figure below visualizes relevant concepts that are required to formulate clustering performance metrics.

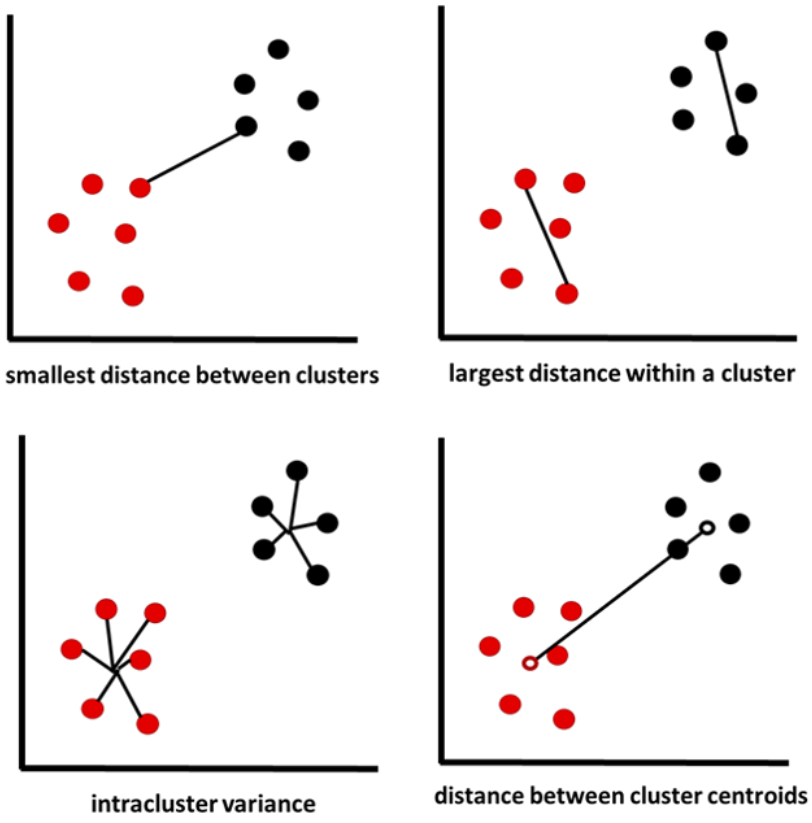


Figure 60 - Concepts related to measuring clustering performance

Silhouette Analysis, Dunn Index, Davies-Bouldin Index

A silhouette analysis (first introduced in [1]) is used to interpret the quality of the obtained clusters. Simply speaking, it measures the degree of similarity of an element to its own cluster with respect to other clusters. The mathematical formula is:

$$s(i) = \frac{b(i) - a(i)}{\max\{a(i), b(i)\}}$$

Wherein $a(i)$ is the average Euclidean distance between a point and the cluster in which it is positioned by a clustering algorithm, $b(i)$ is the average Euclidean distance between a point and all the other clusters to which it does not belong. The formula in the denominator simply selects the maximum between $a(i)$ and $b(i)$. As a consequence, the silhouette coefficient is a value that ranges between -1 and 1. The idea behind it is that points that are highly centred within a cluster and far away from other clusters would have values close to 1. The sum of all the silhouette coefficients would then give an idea on the cohesion of the cluster and how well it is separated from other clusters.

Other indexes that work similarly to the Silhouette analysis are the *Dunn Index* [2] and the *Davies-Bouldin Index* [3]. The Dunn Index can be understood to be the ratio of the smallest distance between elements of different clusters with the biggest distance between elements of the same cluster.

The interpretation of the Dunn Index is that the larger this index is, the further away the clusters are. Dunn Index takes a value between 0 and positive infinity; the higher the value, the better.

Video: DBSCAN in RapidMiner

The Davies-Bouldin Index evaluates the ratio between average intra-cluster similarity and average inter-cluster similarity. The idea is that it is used to measure how much clusters are compact, similarly to Silhouette analysis and it is a value in the range between 0 and 1, a number close to 1 indicates a good separation. RapidMiner cannot currently implement these metrics, but students should be aware of their existence and interpretation.

Practice: Clustering in RapidMiner

In RapidMiner, several clustering algorithms are implemented, and they can generally be utilized using the operator that has the name of the algorithm. For instance, to execute the k-Means clustering algorithm, one should use the Clustering (k-Means) operator. The following figure shows the workflow that exemplifies the use of clustering. Note that most of the clustering algorithms use Euclidean distance as a measure of similarity, thus, they require feature scaling, i.e., normalization.

Video: K-means clustering in RapidMiner

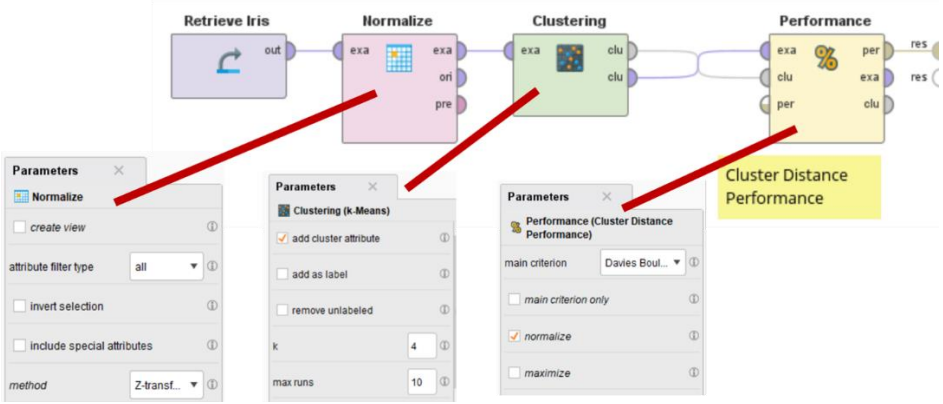


Figure 61 - Clustering in RapidMiner

References:

[1] P.J. Rousseeuw (1987): **Silhouettes: A graphical aid to the interpretation and validation of cluster analysis**, Journal of Computational and Applied Mathematics, Vol. 20, pp. 53–65.

[2] J.C. Dunn (1974): **Well separated clusters and optimal fuzzy partitions**, Journal of Cybernetica, Vol. 4, pp. 95–104.

[3] D.L. Davies and D.W. Bouldin (1979): **Cluster separation measure**, IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 1, No. 2, pp. 95–104.

8 Association Rule Mining

This chapter covers association rule mining; an unsupervised algorithm family that is used to analyze transactional data to reveal the patterns of co-occurring.

Association Rule Mining: motivation and main concepts

Association rule mining (ARM) is used to find the patterns of co-occurring, for example, events that happen together or items that are sold together. Companies very often use ARM for the following tasks:

- Market basket analysis
- Recommender systems
- Personalisation and customization of a service

First of all, an *association rule* takes the following form:

$$LHS \rightarrow RHS$$

LHS means 'left hand side' and RHS means 'right hand side'. The LHS is usually an item, set of items or set of attributes, while the RHS is also an item, set of items or set of attributes. Stated otherwise, the LHS implies the RHS, or rather 'if LHS, then RHS'.

A concrete example of an association rule could be:

$$\{milk, bread, tomatoes\} \rightarrow \{eggs\}$$

This can be read as 'if the customer buys milk, bread and tomatoes, then it is highly likely that eggs will be bought too'.

In order to select interesting rules from the dataset, the algorithms use two important metrics: *support* and *confidence*. Support is a number between 0 and

1 and indicates how frequently that particular rule is true in the dataset. Confidence is also a number between 0 and 1 and represents how many times the rule has been found to be true.

Another important concept is the concept of *lift*, that is a number ≥ 1. Unfortunately, there is a complex probabilistic explanation behind the lift concept. In our case though it is simply important to know that the lift represents the degree to which the attributes occur independently from each other, and that a value > 1 implies that there is some sort of dependency. A value of 1 implies that these attributes are independent and no rule can be created between the two attributes or items.

8.1 Apriori Algorithm

This section introduces an informal description of the Apriori algorithm presented in [1]. The *Apriori* algorithm was designed to work on transactions to identify which items occur simultaneously most often. Here, each of the transactions considered is expected to be a set of *items (itemset).* To determine a relationship to be interesting, the algorithm defines a threshold **T** as the number of transactions in a database in which an itemset has to appear in order to be considered interesting.

Apriori uses an approach that begins by checking items of one element against the transactions in the database and then including further elements in the basket. If an itemset is found to be irrelevant with respect to the threshold **T**, the itemset is discarded to pass on to the next itemset in a breadth first search. At the end of the algorithm, the set of itemsets that pass the threshold **T** check are returned to the user.

In order to further prune the returned itemset (since this may contain a large number of items if only **T** is defined) the user can also specify a support ε (support as defined in Section 1 above in this chapter) stating the percentage of transactions in which the itemset has to present itself in order to consider it a relevant association rule.

Due to its simplicity, despite being historically relevant the Apriori algorithm has a number of *limitations*. The two most relevant limitations are that it generates a large number of subsets and that its breadth first traversing strategy takes a very long time to traverse the entire database. These limitations led researchers to look into more efficient algorithms for association rule mining, one of which is the FP-Growth algorithm introduced in the next section (also available in RapidMiner).

8.2 FP-Growth Algorithm

This section introduces, again in informal fashion, the FP-growth algorithm [2]. FP-growth stands for '*frequent pattern growth*'. FP-Growth improves upon the Apriori algorithm quite significantly. The major improvement to Apriori

is particularly related to the fact that the FP-growth algorithm only needs two passes on a dataset.

- In the first step, the algorithm builds a compact data structure called the *FP-tree.*
- In the second step, the algorithm builds frequent itemsets directly from the FP-tree.

In addition, the algorithm has the parameter φ, which is user defined, to define a threshold for which items are considered as frequent.

As an illustrative example, consider the following dataset of transactions:

Table 2: Sample transaction data

Transaction ID	Items
1	{apricots, bread}
2	{bread, carrots, dumplings}
3	{apricots, carrots, dumplings, eggs}
4	{ apricots, dumplings, eggs}
5	{ apricots, bread, carrots }
6	{ apricots, bread, carrots, dumplings}
7	{ apricots }
8	{ apricots, bread, carrots}
9	{ apricots, bread, dumplings }
10	{ bread, carrots, eggs}

Considering the first transaction, for example, it means that apricots and bread were bought together.

For this dataset, with regard to step 1, the FP-tree is constructed as follows:

1. Read the first transaction {apricots, bread}.
   a. Create a root node.
   b. Create a path root → apricots → bread and set their counts to 1.
2. Read the second transaction {bread, carrots, dumplings}.
   a. Create a path root → bread → carrot → dumplings.
   b. Set their count to 1.
   c. Transactions 1 and 2 share bread, but their path is distinct because they do not have the same prefix.
   d. Link bread in the first path with bread in the second path.
3. Read the third transaction {apricots, carrots, dumplings, eggs}.
   a. Since this path shares apricots with the first path, set the count of apricots to 2, then add nodes carrots, dumplings and eggs after the apricots node.
   b. Add links between carrots and dumplings.

This is continued in the same fashion until the entire dataset has been mapped. Figure 62 shows a depiction of the algorithm used to create the FP-tree.

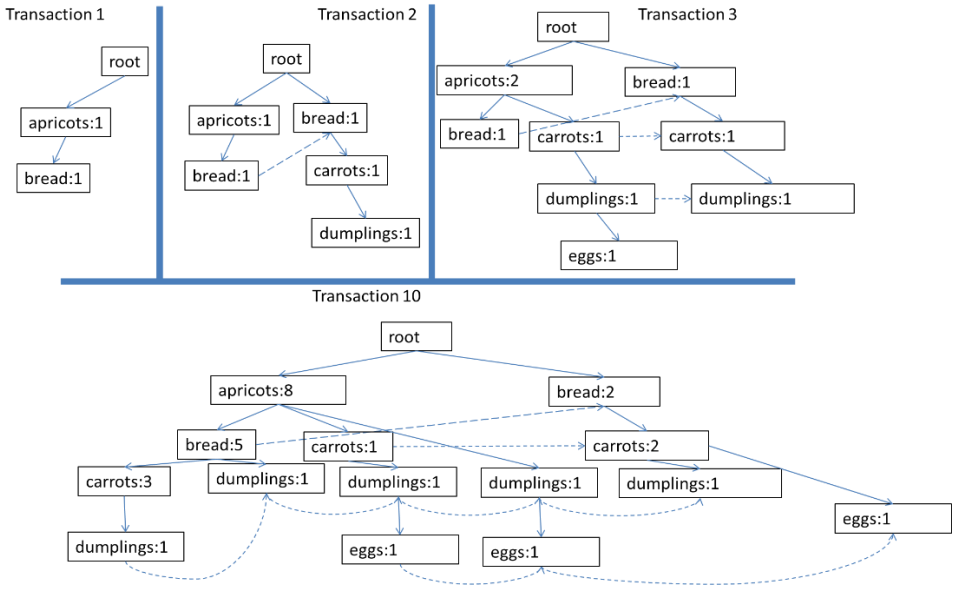


Figure 62 - FP-tree building.

After creating the FP-tree, the algorithm proceeds to identify the frequent itemsets by using the FP-tree. In order to do this, the algorithm takes each of the items and checks the entire prefix path subtree preceding the items. For the eggs item, the prefix path subtree is as shown in Figure 63. Informally speaking, the prefix path subtree is nothing more than the part of the tree above the eggs items.

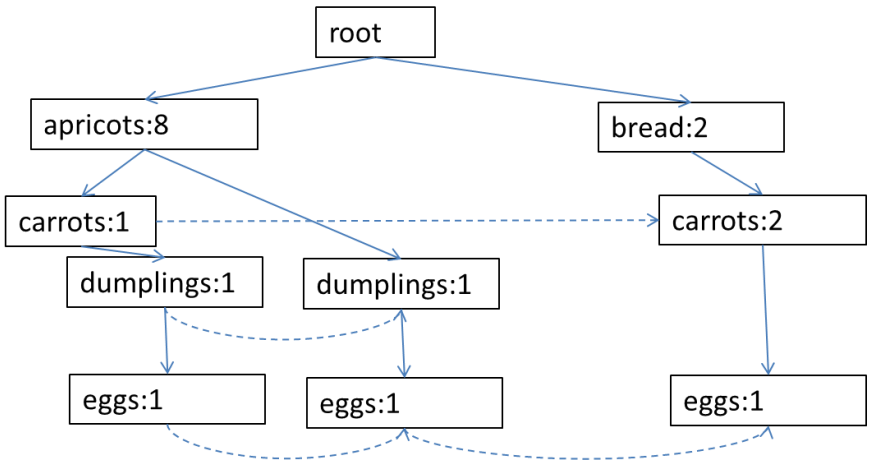


Figure 63 - Prefix path subtree for the eggs item.

Now it must be checked whether eggs are a frequent item. To do so count the occurrences of the eggs item in the subtree (along the dotted lines) and sum the frequencies. In this case the count = 3. If the user defined parameter $\varphi \geq 2$ then {eggs} are considered a frequent item. Now that eggs have been extracted as a frequent item, use the prefix path subtree to find all the itemsets ending in {eggs}. See Figure 64 for an example of how FP-Growth creates these itemsets.

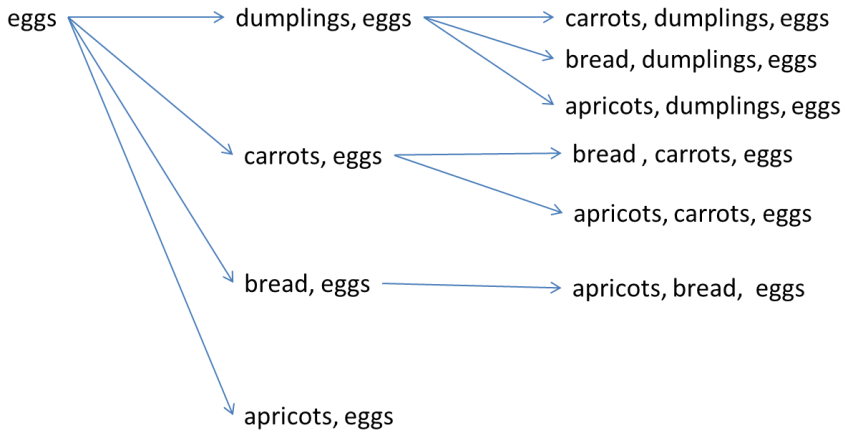


Figure 64 - Frequent itemsets evaluation ending in eggs.

The main issue now is to identify which of these itemsets are more relevant. In order to do this FP-growth uses the prefix path subtree to create an FP-tree around the eggs item. This is called a *conditional FP-tree* (for the eggs item). To do this, the eggs item is removed from the prefix path sub tree. This is done because eggs are no longer needed. The goal is to see what happens in the prefix of the transactions, or to see what is frequent simultaneously with the eggs item. Figure 65 illustrates such a tree for the eggs item.

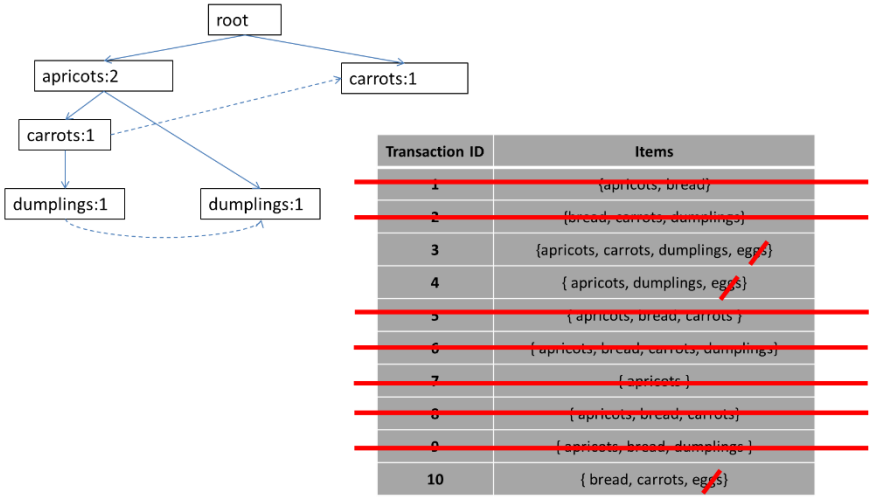


Figure 65 - Conditional FP-Tree for the eggs item.

The interesting thing is that at this point it is only necessary to count. So now the count = 2 for dumplings, so {dumplings, eggs} may be called a frequent itemset. Similarly, {apricots, dumplings, eggs} is also a frequent itemset. To find out whether the {carrots, eggs} pattern is a frequent itemset, the prefix path ending in {carrots, eggs} must be found. The same procedure as before is applied recursively and the prefix path is then found in the following prefix path (Figure 5).

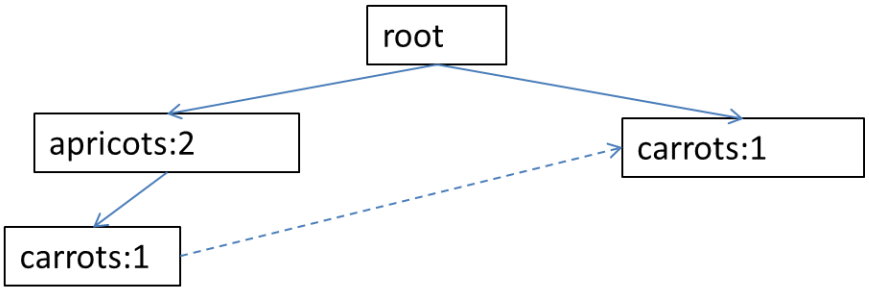


Figure 66 - {carrots, eggs} prefix path.

Thus, the count ≥ 2 with respect to carrots. After applying this procedure, a number of times recursively, for this dataset the following itemsets are found to be the frequent ones:

Table 3: Frequent itemsets

Suffix	Frequent Itemset
{eggs}	{eggs}, {eggs, dumplings}, {apricots, dumplings, eggs}, {carrots, eggs}, {apricots, eggs}

{dumplings}	{dumplings}, {carrots, dumplings}, {bread, carrots, dumplings}, {apricots, carrots, dumplings}, {bread, dumplings}, {apricots, bread, dumplings}, {apricots, dumplings}
{carrots}	{carrots}, {bread, carrots}, {apricots, bread, carrots}, {apricots, carrots}
{bread}	{bread}, {apricots, bread}
{apricots}	{apricots}

As previously stated, FP-growth has a number of advantages with respect to Apriori, in particular in that it only requires two steps to define the general FP-tree to start the rule mining procedure, as has been illustrated. It is also much faster than Apriori in the rule mining task.

FP-growth also has some *disadvantages.* First of all, the FP-tree may be expensive to build, since if the dataset is big, it may not fit in memory. Secondly the *support* metric can only be calculated once the tree has been created. This effectively means that in datasets in which the tree cannot be built, the support also cannot be calculated, meaning that the data analyst should resort to something simpler, like, for example, the Apriori algorithm previously explained.

References

1. R. Agrawal and R. Srikant (1994): **Fast algorithms for mining association rules in large databases,** *Proceedings of the 20th International Conference on Very Large Data Bases, VLDB, Santiago, Chile, September 1994*, pp. 487-499.
2. J. Han, H. Pei, and Y. Yin**: Mining Frequent Patterns without Candidate Generation**, *Proc. Conf. on the Management of Data (SIGMOD'00, Dallas, TX)*, New York, NY: 1-12 ACM Press.

9 Text Analytics

This chapter covers text analytics; a special case of data analytics techniques applied on text data. Most data analytics tools and techniques are applicable to text; however, text is an unstructured form of data, and there are certain things to consider when working with text.

9.1 Text as data

Text data is largely available on the Internet. News articles, web pages, books, online product reviews, and emails are among common sources of text data.

There are many applications of text analytics that serve to unravel the intricate tapestry of written language. One such application involves the comparison of text documents to identify overlapping sections, a function vital for plagiarism checkers. Using text analytics techniques, one can employ queries to locate specific terms, much like the search engines we rely on daily. Furthermore, text classification allows for the categorization of textual documents into groups, while sentiment analysis helps discern the emotional tone within the text. Topic modelling identifies prevalent subjects across a corpus of text. Summarization condenses extensive text while retaining its essence. Lastly, automated spelling and grammar checks ensure the precision and clarity of written content. The applications are diverse, and their implications profound, within text analysis.

Text data are abundant, and text analytics tools and techniques are mature. However, text analytics has specific challenges of its own. The greatest challenge is the inherent unstructured nature of text. Text is not readily understood by computers, thus necessitating pre-processing and transformation. While text adheres to linguistic structures understandable to humans, it can have noise, due to various reasons: human errors, intricacies of language such as synonyms and abbreviations, and the critical role of context in determining meaning.

In text analytics, certain fundamental concepts are frequently used, and it is important to understand their essence. A “**corpus”** represents a collection of text documents, forming a body of textual data for analysis. The core unit of analysis in text analytics is a "**document**". A document contains text regardless of its scale. Files, emails, paragraphs, or even single sentences can be considered documents. Within each document, there are "**tokens**". Tokens are generally words, phrases, or sentences. Thet serve as the foundational elements for analysis.

Text analytics involves the extraction of relevant information from unstructured text, transforming unstructured text into structured data forms.

This transformation serves as a fundamental precursor to the integration of text into the process of machine learning. The subsequent phases of text analytics follow the conventional machine learning process. In a typical text analysis problem, the initial step is **data acquisition**. Following the acquisition step, text undergoes cleansing and **preprocessing** to prepare it for **modelling**. In the third step, models are trained and deployed to address the specific problem at hand. As with all machine learning applications, the final stage entails the **evaluation** of the model's performance and the presentation of results (or **deployment**), thereby completing the cycle of text analytics process.

9.2 Text processing

To transform the unstructured text into structured data forms, text needs to be processed. There are many techniques available to process text. These should be selected and applied depending on the necessities of the problem at hand. This subsection introduces several fundamental techniques that can be used in text processing.

Tokenization

Tokens serve as the building blocks of text, representing independent and atomic components. A text document has a hierarchical structure: sentences can be deconstructed into clauses, phrases, and individual words. Tokenization is defined as the process of breaking down textual data into these smaller units, referred to as tokens.

Sentence tokenization involves the segmentation of a text corpus into individual sentences, often achieved by employing delimiters like periods, newline characters, or semicolons. On the other hand, word tokenization breaks the sentences down into their constituent words.

Example:

> *"Two vast and trunkless legs of stone stand in the desert. Near them, on the sand, half sunk a shattered visage lies. And on the pedestal, these words appear: My name is Ozymandias, King of Kings; Look on my Works, ye Mighty, and despair! Nothing beside remains. The lone and level sands stretch far away."*

Sentence tokens:

> ["Two vast and trunkless legs of stone stand in the desert.",
> "Near them, on the sand, half sunk a shattered visage lies.",
> "And on the pedestal, these words appear:",
> "My name is Ozymandias, King of Kings; Look on my Works, ye Mighty, and despair!",
> "Nothing beside remains.",
> "The lone and level sands stretch far away."]

Word tokens of the first sentence:

[“Two”, “vast”, “and”, “trunkless”, “legs”, “of”, “stone”, “stand”, “in”, “the”, “desert.”]

Text normalization

Text normalization encompasses a series of steps aimed at tidying, refining, and standardizing textual data for analysis. This process uses a variety of techniques to prepare the text data for effective text analytics. The choice of techniques applied depends on the specific task at hand, ensuring a tailored approach to data preparation.

Here are several examples. Converting all text to lowercase ensures consistency and removes case-related ambiguities. Punctuation and special characters (e.g., newline characters and specific symbols) can be removed. Markup, such as HTML tags, can be stripped away, enhancing the purity of the textual content. The handling of numbers is another crucial consideration, as they can be either removed or substituted if it serves the purpose of the analysis. Addressing encoding issues is necessary to ensure data integrity. In addition, excessively short sentences can be eliminated to enhance the overall quality of the text. Finally, detecting and replacing abbreviations or specialized terms contributes to better comprehensibility and more accurate analyses.

Stemming

The smallest, independent units of natural languages are morphemes. Morphemes comprise both word stems and affixes, having a unique role in shaping the structure and meaning of words. Word stems are base form of a word. Affixes are units that are attached to a word stem to alter its meaning or creating new words. Affixes encompass both prefixes and suffixes.

The process of attaching affixes to a word stem is referred to as inflection, allows for the expansion of vocabulary by attaching these affixes to word stems. Conversely, stemming is the reverse process, entailing the derivation of the base form of a word from its modified form. For instance, starting with the word "jump," the application of various affixes such as "s," "ed," and "ing" results in distinct forms like "jumps," "jumped," and "jumping." Stemming strips the word from affixes, yielding the base word.

There are many libraries and tools for stemming, referred to as stemmers. Each stemmer has a distinct manner of work, often yielding different outcomes. Thus, the selection of a stemmer depends on the problem at hand.

Part-of-Speech (POS) Tagging

The multifaceted nature of language is evident in how words can take on varying meanings depending on their roles within a sentence. These roles are formally referred to as "**Part-of-Speech**," abbreviated as POS, and they fall into eight standard categories:

Adjectives serve to modify nouns, as exemplified by "*brown*" in the phrase "*brown* jacket." **Adverbs**, on the other hand, modify verbs and enhance descriptions, as in "she walked *quickly*." **Conjunctions** and **determiners** are the connective elements of language. **Nouns** are fundamental to language as they represent objects, entities, or concepts. **Prepositions** establish spatial and relational connections within sentences. **Pronouns** replace specific nouns, streamlining language and avoiding repetition. **Verbs** are the action words. These eight standard POS categories provide a framework for understanding the roles words play in a sentence.

Lemmatization

Lemmatization and stemming share a common goal of simplifying words by removing their affixes. However, they differ in a crucial aspect: lemmatization aims to find the **root word** (or lemma), not just the base word. This root word is the dictionary form, ensuring that the result is a lexicographically correct word found in the dictionary, which is not always the case with stemming.

Lemmatization is a slower process compared to stemming. The lemma of a word might differ based on the function of the word that is indicated by its POS tag. For instance, the lemma of the word “meeting” is “meeting” when it is used as a noun. However, when it is used as a verb, the lemma of the word “meeting” is “meet”.

Stopword removal

In languages, certain words hold negligible meaning, such as "the," "a," or "me" in English. These words are referred to as "stopwords", and they contribute little to the text.

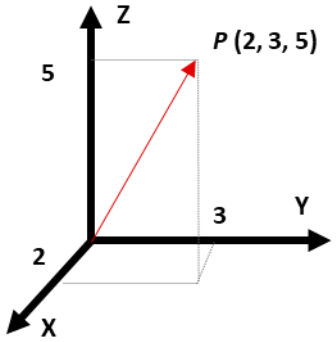
Often, the removal of stopwords is a standard practice in text analytics, as they tend to clutter text without substantially contributing to its meaning. However, the decision to remove stopwords can be context-specific and contingent on the particular problem at hand.

To facilitate this removal process, readily available lists of stopwords exist. One can use such a list to filter out stopwords.

9.3 Text Representation: Vectorization

Raw text is unstructured and needs to be transformed in a structured data format. This transformation can be done by means of vectorization. Vectorization is the represent an entity or a phenomenon in the form of a vector. Thus, text can also be represented as a vector.

A vector is simply an arrow with a certain length and direction. Vectors can have many dimensions. For instance, the vector *P* on Figure 67 is a three-dimensional vector.



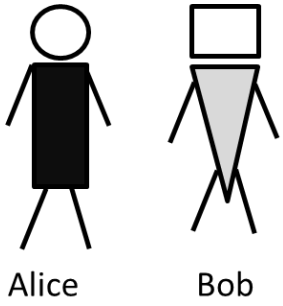
Vector	X	Y	Z
P	2	3	5

P = (x, y, z)

P = (2, 3, 5)

Figure 67 – A 3D Vector

It is possible to represent any entity as vectors. It simply requires a way to encode their characteristics meaningfully. Consider the stick figures in Figure 68. We can represent each individual as a vector that contains certain characteristics about them which we want to keep. In this example, the vectors include observations about head shape, body shape, and body colour. Therefore, each vector consists of these three attributes.



Vector	Head shape	Body shape	Body color
V	v1	v2	v3

Figure 68 - Representing entities as vectors.

Based on these we represent Alice and Bob as the following two vectors.

Alice = (round, rectangle, black)

Bob = (rectangle, triangle, grey)

However, to leverage the computational potential of these representations, we must convert these attributes into numerical values. This is where a technique such as one-hot encoding becomes invaluable.

9.4 One-hot encoding

In one-hot encoding, we expand the vector's dimensions to cover all potential values for each categorical attribute. For instance, within the head-shape dimension, we create sub-dimensions for every conceivable value, like

"round-head" and "rectangle-head." In this extended vector space, a Boolean value of 1 is assigned to a dimension if the entity being represented possesses that specific attribute, while other dimensions remain at 0, as depicted on Figure 69.

Vector	Round Head	Rectangle Head	Rectangle Body	Triangle Body	Black Body	Gray Body
Alice	1	0	1	0	1	0
Bob	0	1	0	1	0	1

Figure 69 - One-hot encoding of the vectors.

One-hot encoding enables us to transform categorical data into a numerical format suitable for efficient processing and analysis, making it an essential tool in data representation and the field of machine learning.

9.5 Bag-of-Words

A similar concept can be applied to the vectorization of text documents, and one of the most straightforward techniques for this purpose is known as the "Bag-of-Words" model.

Creating a bag-of-words model commences with the establishment of the vocabulary for the entire corpus of text. Each word within this vocabulary subsequently becomes a vector dimension in the bag-of-words model. Once the model is trained, it can be utilized to transform other text documents into vectors. This transformation involves assigning a numeric value to each word found in a given text document.

One of the most basic features in this approach is binary encoding, whereby vector values are set to 1 if a particular word appears in the text document and 0 if it does not. In this manner, the bag-of-words model converts the complexity of text into a simplified numerical representation, making it a fundamental tool in text analysis and natural language processing.

Example:

Consider the following two documents. The bag-of-words model is created by establishing the vocabulary. Then, the documents are represented as vectors of which the attributes correspond to the words in the vocabulary. If a word exists in a document, the value of the attribute that correspond to that word becomes a binary 1, or else a binary 0, as depicted in Figure 70.

Document 1 (D1) = "The sky is blue."

Document 2 (D2) = "The sun is bright."

Vocabulary = ['blue', 'bright', 'is', 'sky', 'sun', 'the’]

		blue	bright	is	sky	sun	the
D1:	The sky is blue	1	0	1	1	0	1
D2:	The sun is bright	0	1	1	0	1	1

Figure 70 - Bag-of-words model representation.

In the bag-of-words model for text vectorization, each document is regarded as a mere collection of individual words. This approach disregards the rules of grammar, word order, and sentence structure, focusing only on the presence of words within the text.

Bag-of-words modelling is both straightforward and cost-effective to produce. The vector elements can encapsulate various features of the text. These features often take on different forms:

Binary: In this binary representation, vector elements indicate whether a term exists in the document (1) or not (0).

Term Count: This aspect quantifies how many times a specific term appears within the document, resulting in a count-based vector representation.

Term Frequency: It indicates the relative frequency of a term within the document, offering a measure of how prominently the term features in the text.

Example:

Consider the following two documents: D1 and D2. The bag-of-words representation of them that uses different types of features are exemplified in Figure 71.

D1 = "John likes to watch movies. Mary likes movies too."

D2 = "John also likes to watch football games."

		John	likes	to	watch	movies	Mary	too	also	football	games
Binary	D1	1	1	1	1	1	1	1	0	0	0
	D2	1	1	1	1	0	0	0	1	1	1
Term Count	D1	1	2	1	1	2	1	1	0	0	0
	D2	1	1	1	1	0	0	0	1	1	1
Term Frequency	D1	1/9	2/9	1/9	1/9	2/9	1/9	1/9	0	0	0
	D2	1/7	1/7	1/7	1/7	0	0	0	1/7	1/7	1/7

Figure 71 - Bag-of-words representation exemplified using various types of features.

Even though the bag-of-words model is simple, it provides a valuable tool for text analysis by converting the complex language into a numerical format that can be readily processed and analysed.

9.6 Frequency metrics

Term Frequency (TF) provides insights into how frequently a specific term appears within a single document. It is a measure of the prominence of a term in a particular context. However, to assess the significance of a term in a broader corpus of documents, we introduce the concept of Inverse Document Frequency (IDF).

Inverse Document Frequency (IDF) is a metric that quantifies the sparsity of a term within a collection of documents. It operates on the premise that the fewer documents in which a term occurs, the more valuable or distinctive it is to the documents where it is present. In essence, IDF aims to highlight the uniqueness and discriminative power of a term by considering its scarcity across the entire corpus.

$$IDF(t) = 1 + \log\left(\frac{\#\ documents}{\#\ documents\ containing\ t}\right)$$

TF-IDF: When we merge the concepts of Term Frequency and Inverse Document Frequency, we arrive at a crucial metric known as **TF-IDF**. TF-IDF, or Term Frequency-Inverse Document Frequency, serves as a measure of a word's specificity within a document, relative to the entire corpus.

The fundamental idea underlying TF-IDF is highlighted in the following two arguments.

- The more frequently a term appears within a document, the more significant and relevant it is to that document.
- The rarer the term is across other documents in the corpus, the more unique and specific it becomes for the particular document in which it occurs.

By combining these principles, TF-IDF assigns weights to terms in a way that highlights their importance and distinctiveness. Thus, it is a useful metric for information retrieval, document classification, and text mining tasks.

$$TFIDF(t,d) = TF(t,d)\ .IDF(t)$$

N-grams

While the bag-of-words representation treats each individual word as a separate term, there are instances in which word order plays an important role in understanding the context. Consider the phrase "t*he quick brown fox jumps*." In this case, the constituent words include {*quick, brown, fox, jumps*}, but the adjacent pairs like "*quick-brown*," "*brown-fox*," and "*fox-jumps*" are also essential for capturing the context and meaning of the text.

This broader approach to representation, which considers adjacent word pairs, is known as **n-grams**. Specifically, adjacent pairs are often referred to

as *bi-grams*. By incorporating n-grams, the number of features in the vector representation significantly increases.

Example:

Sentence: "*The quick brown fox jumps*"

BOW: ["The", "quick", "brown", "fox", "jumps"]

Bigrams (N=2): ["The_quick", "quick_brown", "brown_fox", "fox_jumps"]

Trigrams (N=3): ["The_quick_brown", "quick_brown_fox", "brown_fox_jumps"]

9.7 Text Similarity

One of the objectives of text vectorization is to facilitate the assessment of similarity between documents. Once text is transformed into numerical vectors, mathematical operations become meaningful. This allows for the comparison of documents through quantitative methods.

The concepts of similarity and distance are essentially opposites. When two documents are identical, their distance is zero, and their similarity is at its maximum, represented as 1.

In the vector space, we can define distances and metrics to quantify the relationships between documents. For instance, the Euclidean distance, which measures the straight-line distance between two vectors, serves as one such metric. This concept forms the mathematical basis for evaluating the dissimilarities and commonalities between textual data.

$$d(p,q) = \sqrt{\sum_{i=1}^{n}(q_i - p_i)^2}$$

Common distance measures:

Manhattan Distance: This metric calculates the distance based on gridlines. It measures the distance one would travel in a grid layout that run parallel and perpendicular. It was named after the grid-like street layout in Manhattan.

Euclidean Distance: This metric represents the physical distance between two points in a 2D space, calculated along the shortest path, often referred to as a "bird's flight" or "beeline."

Minkowski Distance: This metric is a more generalized form of the Euclidean distance.

Cosine Similarity: This metric is calculated by taking the dot product of two vectors and normalizing it by their magnitudes. The result represents the cosine of the angle between the vectors. If the vectors are identical, the angle between them is 0 degrees, yielding a cosine value of 1, indicating maximum similarity. Conversely, if the vectors are dissimilar and form a 90-degree angle, the cosine value is 0, denoting no similarity.

9.8 Text classification

Text classification, also referred to as text categorization, is the task of assigning predefined categories to text documents. This process essentially applies classification techniques to textual data, allowing for the systematic organization and categorization of a wide range of text.

A text classifier takes text as input, analyses its content, and then automatically assigns relevant tags or categories to the documents. Text classifiers can be used for structuring and categorizing text in various contexts. For instance, they can be employed to categorize news articles by topics, prioritize support tickets based on urgency, group chat conversations by language, or assess brand mentions by sentiment, among numerous other applications.

Figure 72 - Conceptual depiction of a text classifier

Training a text classifier begins with the feature extraction step. This process entails transforming each text into a numerical representation in the form of a vector. Various techniques for feature extraction, which we've explored earlier, enable the conversion of text into a format that machine learning algorithms can process. Subsequently, a machine learning algorithm is provided with training data, which consists of pairs of feature sets (vectors representing each document) and corresponding tags or labels (e.g., "sports," "politics"). The algorithm employs this training data to build a classification model. Once the model is sufficiently trained on training samples, it gains the capability to make predictions: they are used to categorize new or unseen text documents.

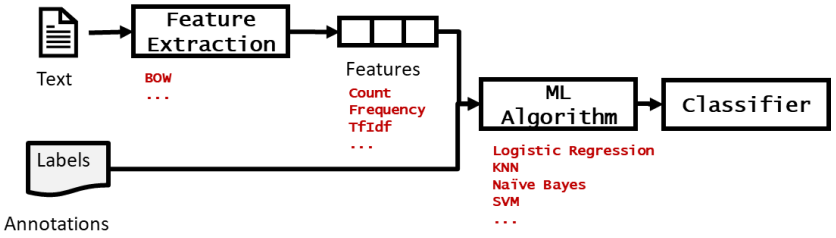


Figure 73 - Conceptual schematics of the training process of a text classifier

A trained classifier operates on a valid input, which is a vector of features derived from the text. While classifying the unseen text, the same feature extractor that was used during the training phase must be employed to transform the new text into feature sets and converting it into a numerical representation.

These feature sets are then fed into the previously trained classification model, which has learned patterns from the training data. The model can then generate predictions regarding the appropriate tags or categories for the new text documents.

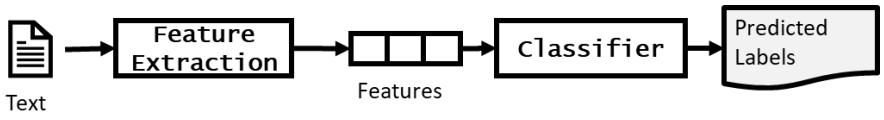


Figure 74 - Conceptual depiction of the utilization of a text classifier

Some of the algorithms that are frequently used in text classification are as follows.

- K-Nearest Neighbours (KNN)
- Logistic Regression
- Support Vector Machines
- Naïve Bayes

9.9 Text clustering

Text clustering is essentially clustering applied to textual data. The process involves vectorizing the text, as discussed earlier, and subsequently applying clustering algorithms to these vectors. The same clustering techniques employed in other domains can be used in text clustering to group similar text documents together.

Topic Modelling is a special type of unsupervised machine learning used to extract and identify underlying themes (or topics) within a body of text. These topics are essentially sets of words that are related in context While it provides valuable insights, the interpretation of these topics and their practical application often demands human judgment and contextual understanding.