DATA PREPARATION:

Business intelligence systems and mathematical models for decision making can achieve accurate and effective results only when the input data are highly reliable. However, the data extracted from the available primary sources may have several anomalies which analysts must identify and correct. Several techniques are employed to reach this goal.

1. **DATA VALIDATION**: the quality of input data may prove unsatisfactory due to:
   - **Incompleteness:** some records may contain missing values corresponding to one or more attributes. Data may be missing because of malfunctioning recording devices. It is also possible that some data were deliberately removed during previous stages of the gathering process (nella fase precedente al processo di raccolta dati). Incompleteness may also derive from a failure to transfer data from the operational databasesto a data mart used for a specific business analysis.
   - **Noise**: data may contain erroneous or anomalous values, which are usually referred to as "outliers". Other possible causes of noise are to be searched in malfunctioning devices for data measurement, recording and transmission.
   - **Inconsistency**: sometimes data contain discrepancies due to changes in the coding system used for their representation, and therefore may appear inconsistent.

   The purpose of data validation techniques is to identify and implement corrective actions in case of incomplete and inconsistence data or data affected by noise.

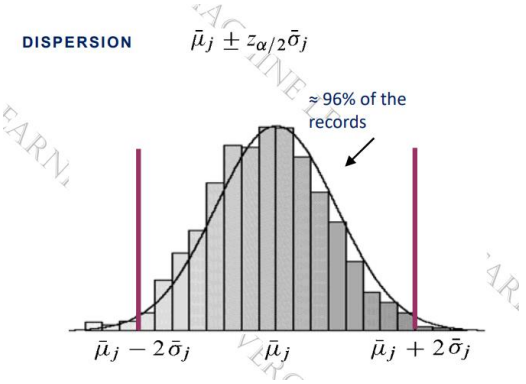
   1.1. Incomplete data: to partially correct incomplete data, there are several techniques
   - Elimination: it is possible to discard all (Scartare) records for which the values of one or more attributes are missing.
   - Inspection: it is possible to opt for an inspection of each missing value. This approach suffers from a high degree of arbitrariness and subjectivity.
   - Identification: a conventional value might be used to encode and identify missing values, making it unnecessary to remove entire records from the given dataset.
   - Substitution: several criteria exist for the automatic replacement of missing data, although most of them appear somehow arbitrary. For instance, missing values of an attribute may be replaced with the mean of the attribute calculated for the remaining observations. This technique can only be applied for numerical attributes. It is also possible to replace (sostituire) missing values by calculating the mean of the attribute only for those records having the same target class. However, this procedure can become rather complex and time-consuming for a large dataset with a high percentage of missing data.

   1.2. Data affected by noise: the term noise refers to a random perturbation within the values of a numerical attribute, usually resulting in noticeable anomalies. First, the outliers in a dataset need to be identified, so that subsequently either they can be corrected and regularized or entire records containing them are eliminated. In this section we will describe a few simple techniques for identifying and regularizing data affected by noise. The easiest way to identify outliers is based on the statistical concept of dispersion. The sample mean $\bar{\mu}j$ and the sample variance $\bar{\sigma}2_j$ of the

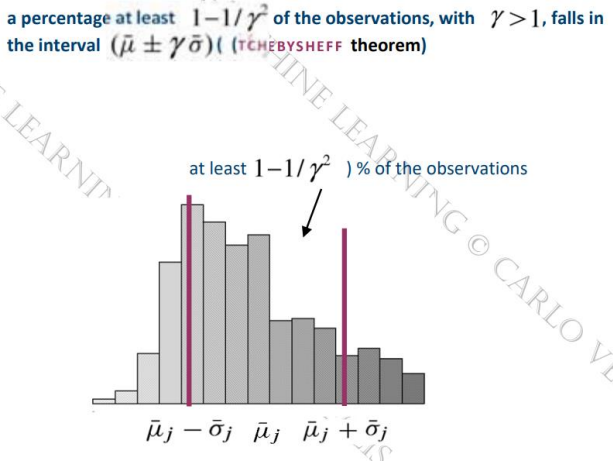
   numerical attribute $\boldsymbol{a}j$ are calculated. If the attribute follows a distribution that is not too far form the normal, the values falling outside an appropriate interval centered around the mean value $\bar{\mu}j$, are identified as outliers. More precisely, with a confidence of $100(1-\alpha)\%$ it is possible to consider as outliers those values that fall outside the interval:

   $$\bar{\mu}j - z(\alpha/2)\,\bar{\sigma}j \quad \text{and} \quad \bar{\mu}j + z(\alpha/2)\,\bar{\sigma}j$$

   Where $z\frac{\alpha}{2}$ is the $\frac{\alpha}{2}$ quantile of the standard normal distribution.



This technique is simple to use, although it has the drawback of relying (benchè abbia l'inconveniente di dover fare affidamento) on the critical assumption that the distribution of the values of the attribute is bell-shaped and roughly normal (forma di campana e normale). However, with **Tchebysheeff's theorem** it is possible to obtain analogous bounds independent from the distribution, with intervals that are only less precise. Once the outliers have been identified, it is possible to correct them with values that are considered more plausible, or to remove an entire record containing them.

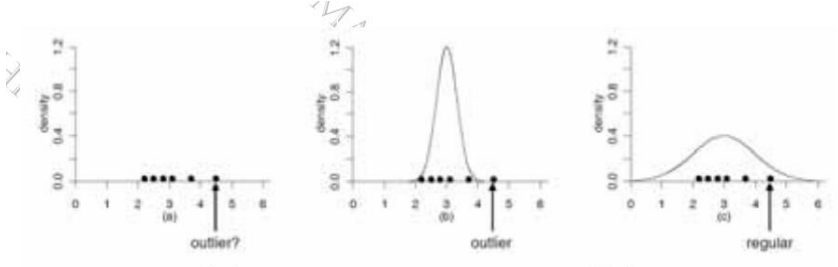


Z.index method for outlier detection

We have to know the density (curve) and the dispersion (number).

>> is very possible that is an outlier

>probably is an oulier

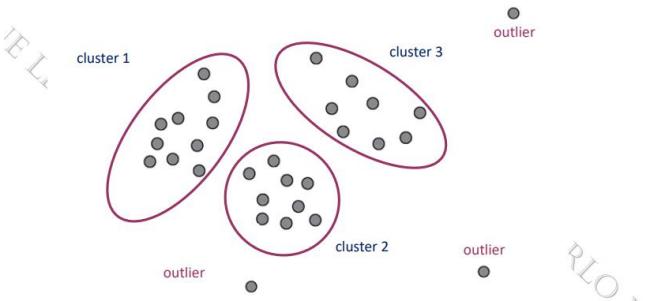


$$z_{ij}^{ind} = \frac{x_{ij} - \bar{\mu}_j}{\bar{\sigma}_j}$$

$$\left|z_i^{ind}\right| > 3$$

$$\left|z_i^{ind}\right| \gg 3$$

An alternative technique, illustrated in the following figure, is **based on the distance between observations and the use of clustering methods**.



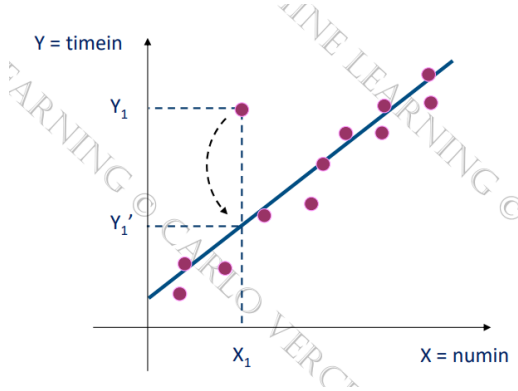
evaluate "density" in a neighborhood of a record: a record is an outlier if a percentage *p* of the records falls at a distance greater than *d*

Once the clusters have been identified, representing sets of records having a mutual distance that is less than the distance from the records included in other groups, **the observations that are not placed in any of the clusters are identified as outliers.** Clustering techniques offer the advantage of simultaneously considering several attributes.

A variant of clustering methods, also based on the distances between the observations, detects the outliers through two parametric values, p and d , to be assigned by the user. **An observation $\boldsymbol{x_i}$ is identified as an outlier if at least a percentage p of the observations in the dataset are found at a distance greater than d from $\boldsymbol{x_i}$.**

Unlike the above methods whose aim is to identify and correct each single anomaly, there also exist regularization techniques, which automatically correct anomalous data(AUTOMATIC CORRECTION OF NOISE OSSERVATION).

For example: simple or multiple regression models predict the value of the attribute $\boldsymbol{a_j}$ that one wishes to regularize based on other variables existing in the dataset. Once the regression model has been developed, and the corresponding confidence interval around the prediction curve has been calculated, it is possible to substitute the value computed along the prediction curve for the values of the attribute $\boldsymbol{a_j}$ that fall outside the interval.



X1,Y1= valore calcolato
X1',Y1'=valore desiderato

2. DATA TRANSFORMATION: In most data mining analysis it is appropriate to apply a few transformations to the dataset in order to improve the accuracy of the learning models subsequently developed.

2.1. Standardization: Most learning models benefits from a preventive standardization of the data, **also called normalization**. The most popular standardization techniques include the decimal scaling method, the min-max method, and the z-index method.

- Decimal scaling: decimal scaling is based on the transformation

$$x'_{ij} = \frac{x_{ij}}{10^h}$$

where h is a given parameter which determines the scaling intensity. In practice, decimal scaling corresponds to shifting the decimal point by h positions toward the left. In general, h is fixed at a value that gives transformed in the range $[-1,1]$.

- Min-max: it is achieved through the transformation

$$x'_{ij} = \frac{x_{ij} - x_{min,j}}{x_{max,j} - x'_{min,j}} (x'_{max,j} - x'_{min,j}) + x'_{min,j}$$

Where:

$$x_{min,j} = \min_i x_{ij}$$
$$x_{max,j} = \max_i x_{ij}$$

are the minimum and the maximum values of the attribute a_j before transformation, while $x'_{min,j}$ and $x'_{max,j}$ are the minimum and the maximum values that we wish to obtain after transformation. In general, the extreme values of the range are defined so that $x'_{min,j} = -1$ or $x'_{min,j} = 0$ and $x'_{max,j} = 1$.

- z-index: it uses the transformation

$$x'_{ij} = \frac{x_{ij} - \bar{\mu}_j}{\bar{\sigma}_j}$$

Where $\bar{\mu}j$ and $\bar{\sigma}j$ are respectively the sample mean and the sample standard deviation of the attribute $\boldsymbol{a}_j$. If the distribution of values of the attribute $\boldsymbol{a}_j$ is roughly normal, the z-index based transformation generated values that are almost certainly within the range (-3, 3).

2.2. Feature extraction: The aim of standardization techniques is to replace the values of an attribute with values obtained through an appropriate transformation. However, there are situations in which more complex transformations are used to generate new attributes that represent a set of additional columns in the matrix $\boldsymbol{X}$ representing the dataset *D*. Transformations of this kind are usually referred to as feature extraction. Attribute extraction may also consist of the **creation of new variables** that summarize with in themselves the relevant information contained in a subset of the original attributes.

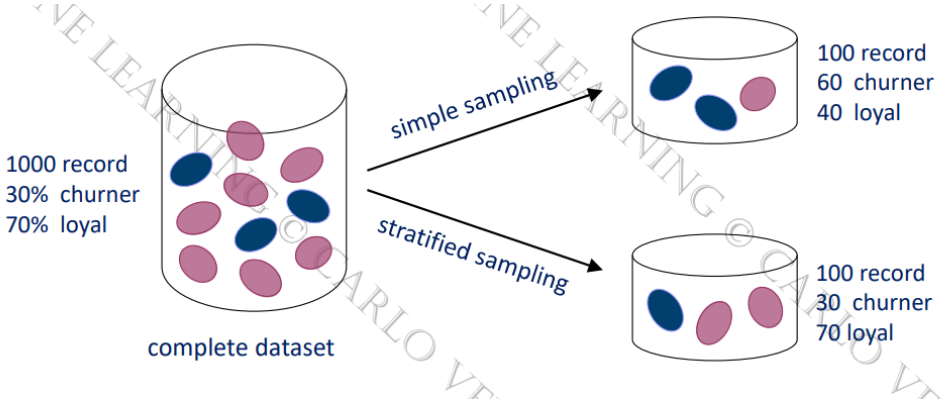
numout	decimal scaling	min-max method	z-index method
45	0.045	0.295	-0.006
20	0.02	0.078	-0.023
69	0.069	0.504	-0.01
66	0.066	0.478	0.008
11	0.011	0	-0.029
42	0.042	0.269	-0.008
126	0.126	1	0.049
original values			

3. **DATA REDUCTION** (because we usually want a faster training of our data) **:** when dealing with a small dataset, the transformations described above are usually adequate to prepare input data for a data mining analysis. However, when facing a large dataset, **it is also appropriate to reduce its size**, in order to make learning algorithms more efficient, without sacrificingthe quality of the result obtained. There are three main criteria to determine whether a data reduction technique should be used: efficiency, accuracy and simplicity of the models generated.
   - Efficiency: the application of learning algorithms to a dataset smaller than the original one usually means a shorter computation time and a reduction in processing time allow the analyses to be carried out more quickly.
   - Accuracy: the accuracy of the models generated represents a critical success factor, and it is therefore the main criteria on followed in order to select one class of learning methods over another. As a consequence, data reduction techniques should not significantly compromise the accuracy of the model generated.
   - Simplicity: in some data mining applications, concerned more with interpretation than with prediction, it is important that the models generated be easily translated into simple rules that can be understood. Data reduction often represents an effective technique for deriving models that are more easily interpretable.

   Data reduction can be pursued in three distinct directions: a reduction in the number of observations through sampling, a reduction in the number of attributes (features) through selection and projection, and a reduction in the number of values through discretization and aggregation.

   3.1. **Sampling (campionamento):** a further reduction in the size of the original dataset can be achieved by extracting a sample of observations that is significant from a statistical standpoint (da un punto di vista statistic). This type of reduction is based on classical inferential reasoning. It is therefore necessary to determine the size of the sample that guarantees the level of accuracy required by the subsequent learning algorithms andto define an adequate sampling procedure. Generally speaking, a sample comprising a few thousand observations is adequate to train most learning models. It is also useful to set up several independent samples, each of a predetermined size, to which learning algorithms should be applied. In this way, computation times increase linearly with the number of samples determined,and it is possible to compare the different models generated, in order to assess the robustness of each model and the quality of the knowledge extracted from data against the random fluctuations existing in the sample. It is obvious that the conclusions obtained can be regarded as robust when the models and the rules generated remain relatively stable as the sample set used for training varies.

   Sampling may be **simple or stratified** depending on whether one wishes to preserve in the sample the percentages of the original dataset with respect to a categorical attribute that is considered critical.

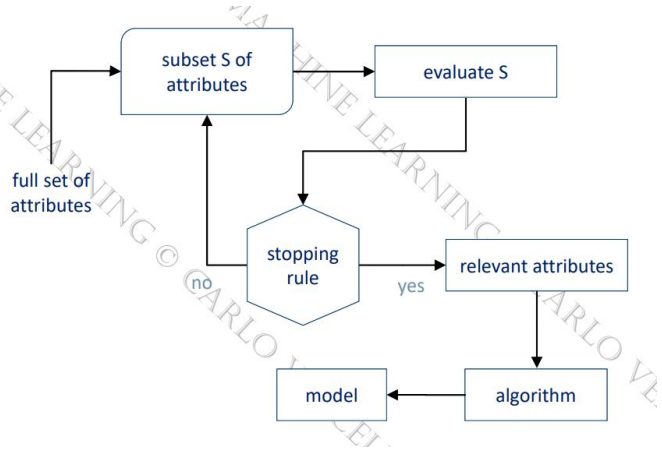


1) Simple sampling→ the risk of this method is perturbating (changing) the percentage of one class respect to the original one. In this example the majority is loyal and after sampling became churner. In this case we take a random sample.
2) Stratified sampling→ is smart way to generate the random sample (as the original one) to keep to the percentages of the class we are interest in. So we have the resulting distribution of the two classes exactly preserved (in term of percentage) like at the beginning of the analysis.

Next toping is elimination columns (variables) when we evaluate that they have no importance.

3.2. Feature selection: the purpose of feature selection, also called "feature reduction", is to **eliminate from the dataset a subset of variables which are not deemed relevant for the purpose of the data mining activities**. Feature reduction has several potential advantages. Due to the presence of fewer columns, learning algorithms can be run more quickly on the reduced dataset than on the original one. Moreover, the **models** generated after the elimination from the dataset of uninfluential attributes are often **more accurate and easier to understand**. Feature selection methods can be classified into three main categories: **filter methods, wrapper methods and embedded methods.**

Filter methods: they select the relevant attribute before moving on to the subsequent learning phase and are therefore independent form the specific algorithm being used. The attributes considered most significant are selected for learning, while the rest are excluded. The simplest filter method to apply for supervised learning involves the assessment of each single attribute based on its level of correlation with the target. Consequently, this conduct to the selection of the attributes that appear mostly correlated with the target.



1)for any independent variables calculate all the linear correlation coefficient with the target (f-statistic for example)

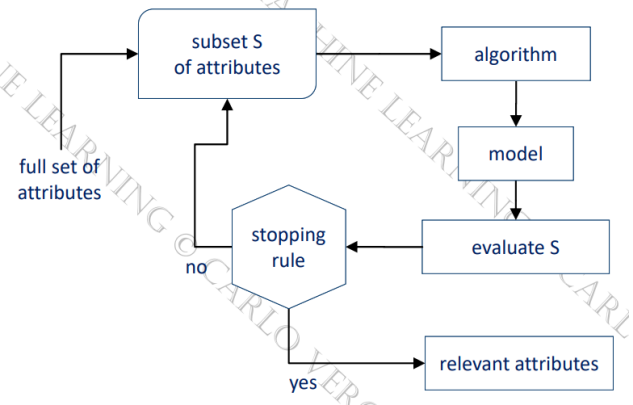
2)if the linear correlation coefficient or f-statistic is minor than the threshold you take it, otherwise you eliminate it.

Advantages: speed (you just run the algorithm once)

Disadvantages: the price to pay for speed is less accuracy

Wrapped methods: If the purpose of the data mining investigation is classification or regression, and consequently performances are assessed mainly in terms of accuracy, the selection of predictive variables should be based not only on the level of relevance of each single attribute but also on the specific learning algorithm being utilized.Wrapper methods are able to meet this need, since they assess a group of variables using the same classification or regression algorithm used to predict the value of the target variable. Each time, the algorithm uses a different subset of attributes forlearning, identified by a search engine that works on the entire set of all possible combinations of variables, and selects the set of attributes that guarantees the best result in terms of accuracy.

Wrapper methods are usually burdensome from a computational standpoint, since the assessment of every possible combination identified by the search engine requires one to deal with the entire training phase of the learning algorithm.



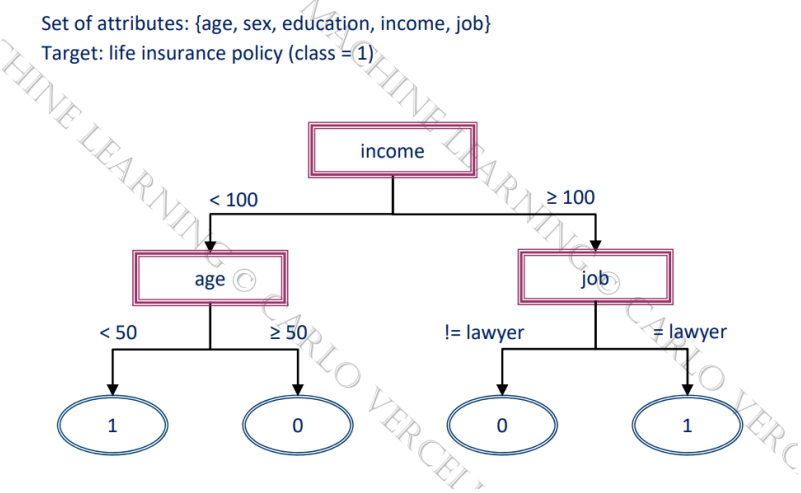
Advantages: most accuracy

Disadvantages: is most cost, we run the algorithm a lot of time

Embedded methods: For these methods, the attribute selection process lies inside (Giace dentro) the learning algorithm, so that the selection of the optimal set of attributes is directly made during the phase of model generation.

Embedded methods means of the selection of the attributes is performed by the method itself.

This is the best model because the method perform automatically selection.



Filter methods are the best choice when dealing with very large datasets, whose observations are described by a large number of attributes. In these cases, the application of wrapped methods is inappropriate due to very long computation times. Moreover, filter methods are flexible and in principle can be associated with any learning algorithm. However, when the size of the problem is moderate, it is preferable to turn to wrapper or embedded methods which afford in most cases accuracy levels that are higher compared to filter methods.

As described above, wrapped methods select the attributes according to a search scheme that inspects in sequence several subsets of attributes and applies the learning algorithm to each subset in order to assess the resulting accuracy of the corresponding model. The procedure for selecting the attributes for wrapped methods is usually of a heuristic nature, based in most caseson a greedy logic which evaluates for each attribute a relevance indicator adequately defined andthen selects the attributes based on their level of relevance. In particular, three distinct myopic search schemes can be followed: forward, backward, and forward–backward search:

Forward: the exploration starts with an empty set of attributes and subsequently introduces the attributes **one** at a time based on the ranking induced by the relevance indicator. The algorithm **stops when the relevance index of all the attributes still excluded is lower than a prefixed threshold.**

Backward: the exploration starts by selecting **all** the attributes and then **eliminates them one at a time** based on the preferred relevance indicator. Thealgorithm **stops when the relevance index of all the attributes still included in the model is higher than a prefixed threshold**.

Forward-Backward: The forward–backward method represents a trade-off between the previous schemes, in the sense that at each step the **best attributeamong those excluded is introduced and the worst attribute among those included is eliminated**. Also, in this case, threshold values for the included and excluded attributes determine the stopping criterion.

The various wrapper methods differ in the choice of the relevance measure as well as well as the threshold preset values for the stopping rule of the algorithm.

3.3. Principal component analysis (PCA): PCA is the most known technique of attribute reduction by means of projection. Generally speaking, the purpose of this method is to obtain a **projective transformation** that replaces a subset of the original numerical attributes with a lower number of new attributes obtained and their **linear combination**, without this change causing a loss of information. Experience shows that a transformation of the attributes may lead in many instances to better accuracy in the learning models subsequently developed. **Before** applying thismethod, it is expedient to **standardize the data**, so as to obtain for all the attributes the same range of values, usually represented by the interval [-1, 1]. Moreover, the mean of each attribute $\boldsymbol{a}_j$ is made equal to 0 by applying the transformation:

$$\tilde{x}_{ij} = x_{ij} - \frac{1}{m}\sum_{i=1}^{m} x_{ij}$$

Let $\boldsymbol{X}$ denote the matrix resulting from applying the transformation above to the original data and let $\boldsymbol{V} = \boldsymbol{X}'\boldsymbol{X}$ be the **covariance matrix** of the attributes. If the correlation matrix is used to develop the PCA method instead of the covariance matrix, the transformation above is not required.

Starting from the n attributes in the original dataset, represented by the matrix $\boldsymbol{X}$, the PCA method derives $\boldsymbol{n}$ **orthogonal vectors**, namely the ***principal components***, which constitute a new basis of the space $\mathbb{R}^n$. Principal components are better suited than the original attributes to explain fluctuations in the data, in the sense that usually a subset consisting of q principal components, with $q < n$, has an information content that is almost equivalent to that of the original dataset. As a consequence, **the original data are projected into a lower-dimensional space of dimension q having the same explanatory capabilities**.

Principal components are generated in sequence by means of an iterative algorithm. The **first component** is determined by **solving an appropriate optimization problem**, in order to explain the highest percentage of variation in the data. At each iteration, the **next principal component is selected, among those vectors that are orthogonal to all components already determined**, as the one which explains the maximum percentage of variance not yet explained by the previously generated components. At the end of the procedure the principal components are ranked in non-increasing order with respect to the amount of variance that they are able to explain.

Let $\boldsymbol{p}_j$ $(j \in N)$, denote the n principal components, each of them being obtained as a linear combination $\boldsymbol{p}_j = \boldsymbol{X}\boldsymbol{w}_j$ of the available attributes, where the weights $\boldsymbol{w}_j$ have to be determined. The projection of a generic example $\boldsymbol{x}_i$ in the direction of the weights vector $\boldsymbol{w}_j$ is given by $\boldsymbol{w'}_j\boldsymbol{x}_i$. It can easily be seen that its variance is given by:

$$\begin{aligned} \mathrm{E}[\mathbf{w}_j'\mathbf{x}_i - \mathrm{E}[\mathbf{w}_j'\mathbf{x}_i]]^2 &= \mathrm{E}[(\mathbf{w}_j'(\mathbf{x}_i - \mathrm{E}[\mathbf{x}_i]))^2] \\ &= \mathbf{w}_j'\,\mathrm{E}[(\mathbf{x}_i - \mathrm{E}[\mathbf{x}_i])'(\mathbf{x}_i - \mathrm{E}[\mathbf{x}_i])]\mathbf{w}_j \\ &= \mathbf{w}_j'\mathbf{V}\mathbf{w}_j. \end{aligned}$$

The first principal component $\boldsymbol{p}_1$ represents a vector in the direction of maximum variance in the space of the original attributes and therefore its weights may be obtained by solving the quadratic constrained maximization problem:

$$\max_{\boldsymbol{w}_1}\{\boldsymbol{w'}_1\boldsymbol{V}\boldsymbol{w}_1 : \ \boldsymbol{w'}_1\boldsymbol{w}_1 = 1\}$$

Where the unit norm constraint for $\boldsymbol{w}_1$ is introduced in order to derive a well-posed problem. By introducing the Lagrangian function:

$$L(\boldsymbol{w}_1, \lambda_1) = \boldsymbol{w'}_1\boldsymbol{V}\boldsymbol{w}_1 - \lambda_1(\boldsymbol{w'}_1\boldsymbol{w}_1 - 1)$$

and applying the Karush-Khun-Tucker conditions, the solution of the maximization problem reduces to the solution of the system:

$$\frac{\partial L(\mathbf{w}_1, \lambda_1)}{\partial \mathbf{w}_1} = 2\mathbf{V}\mathbf{w}_1 - 2\lambda_1\mathbf{w}_1 = \mathbf{0},$$
$$\frac{\partial L(\mathbf{w}_1, \lambda_1)}{\partial \lambda_1} = 1 - \mathbf{w}_1'\mathbf{w}_1 = 0,$$

Which can be rewritten as:

$$(\mathbf{V} - \lambda_1\mathbf{I})\mathbf{w}_1 = \mathbf{0},$$

subject to the unit norm condition $\boldsymbol{w'}_1\boldsymbol{w}_1 = 1$, where $\boldsymbol{I}$ is the identity matrix. The solution of the maximization problem is therefore given by $\boldsymbol{w}_1 = \boldsymbol{u}_1$, where $\boldsymbol{u}_1$ is the eigenvector of unit norm associated with the maximum eigenvalue λ_1 of $\boldsymbol{V}$, through the relation $\boldsymbol{p}_j = \boldsymbol{X}\boldsymbol{u}_j$.

The second principal component may be determined by solving an optimization problem, adding the condition of orthogonality to the previously obtained principal component, expressed by the constraint:

$$\boldsymbol{w'}_2\boldsymbol{u}_1 = 0$$

Proceeding in an iterative way, it is possible to derive the n principal components starting from the eigenvectors $\boldsymbol{u}_j$ $(j \in N)$ of $\boldsymbol{V}$ ordered by non-increasing eigenvalues $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_n$, through equalities $\boldsymbol{p}_j = \boldsymbol{X}\boldsymbol{u}_j$. The variance of the principal component $\boldsymbol{p}_j$ is given by $Var\ (\boldsymbol{p}_j) = \lambda_j$.

The n principal components constitute a new basis in the space $\mathbb{R}^n$, since the vectors are orthogonal to each other. Therefore, they are also uncorrelated and can be ordered according to a relevance indicator expressed by the corresponding eigenvalue.

The interpretation of the principal components may be obtained from the coefficients of the vector $\boldsymbol{w}_j = \boldsymbol{u}_j$ which express their relationship with the original attributes. To this end, notice that the principal component $\boldsymbol{p}_h$ assumes the form:

$$\boldsymbol{p}_h = u_{h1}\boldsymbol{a}_1 + u_{h2}\boldsymbol{a}_2 + \cdots + u_{hn}\boldsymbol{a}_n$$

The coefficient u_{hj} can be therefore interpreted as the weight of the attribute $\boldsymbol{a}_j$ in determining the component $\boldsymbol{p}_h$. The greater the absolute value of u_{hj} is, the more the component $\boldsymbol{p}_h$ is characterized by the attribute $\boldsymbol{a}_j$. At the same time $Var(\boldsymbol{p}_h) = \lambda_h$ represents a measure of the proportion of total variance explained by the principal component $\boldsymbol{p}_h$. For this reason, the index

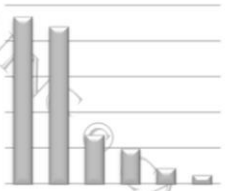
$$\boldsymbol{I}_q = \frac{\lambda_1 + \lambda_2 + \cdots + \lambda_q}{\lambda_1 + \lambda_2 + \cdots + \lambda_n}$$

Expresses the percentage of total variance explained by the first q principal components and provides an indication of the amount of information preserved by the first q components. In order to determine the number of principal components to be appropriately used, it is possible to go on until the level of overall importance $\boldsymbol{I}_q$ of the considered components exceeds a threshold $\boldsymbol{I}_{min}$ deemed reasonable, in relation of the properties of the dataset. The number of principal components is therefore determined as the smallest value q such that $\boldsymbol{I}_q > \boldsymbol{I}_{min}$.

percentage of explained variance

$$I_q = \frac{\lambda_1 + \lambda_2 + \cdots + \lambda_q}{\lambda_1 + \lambda_2 + \cdots + \lambda_n}$$

scree-plot



3.4. Data discretization:

The general purpose of data reduction methods is to obtain a decrease in the number of distinct values assumed by one or more attributes. Data discretization is the primary reduction method. On the one hand, it reduces continuous attributes to categorical attributes characterized by a limited number of distinct values. On the other hand, its aim is to significantly reduce the number of distinct values assumed by the categorical attributes. The model that can be generated on the reduced dataset are likely to be more intuitive and less arbitrary. Among the most popular discretization techniques are subjective subdivision, subdivision into classes and hierarchical discretization.

- Subjective subdivision: is the most popular and intuitive method. Classes are defined based on the experience and judgment of experts in the application domain.
- Subdivision into classes: Subdivision into categorical classes may be achieved in an automated way using the techniques described below. In particular, the subdivision can be based on classes of equal size or equal width.

  The automated procedure of subdivision into classes consists of ordering in a non-decreasing way the values of the attribute $\boldsymbol{a}_j$ and grouping them into a predetermined number K of contiguous classes. It is possible to form the classes of either different size or different width.
- Hierarchical discretization: it is base on hierarchical relationship between concepts and may be applied to categorical attributes, just as for the hierarchical relationships between provinces and regions. In general, given a hierarchical relationship of the one-to-many kind, it is possible to replace each value of an attribute with the corresponding value found at a higher level in the hierarchy of concepts.

DATA EXPLORATION

The primary purpose of exploratory data analysis is to highlight (evidenziare) the relevant features of each attribute contained in a dataset, using graphical methods, and calculating summary statistics, and to identify the intensity of the underlying relationships among the attributes. Exploratory data analysis includes three mainphases:

- Univariate Analysis, in which the properties of each single attribute of a dataset are investigated.
- Bivariate Analysis, in which pairs of attributes are considered, to measure the intensity of the relationship existing between them.
- Multivariate Analysis, in which the relationships holding within (all'interno) a subset of attributes are investigated (more variables will be considered at the same time).

Furthermore, there are two types of variables: **categorical and numerical**.

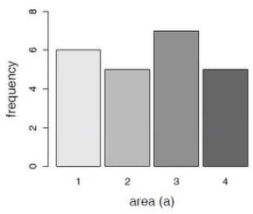
1. **UNIVARIATE ANALYSIS:**

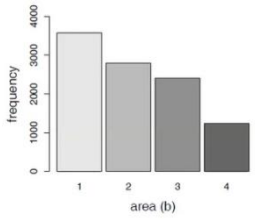
   Univariate analysis is used to **study the behavior of each attribute**, considered as an entity independent of the other variables of the dataset. It is of interest to assess the tendency of the values of a given attribute to arrange (sistemare) themselves around a specific central value (location), to measure the propensity of the variable to assume a more or less wide range of values (dispersion) and to extract information on the underlying probability distribution. On the one hand, some learning models make specific statistical hypotheses regarding the distribution of the variables being examined, and it is therefore necessary to verify the validity of such assumptions before proceeding with the subsequent investigation. On the other hand, univariate analysis intuitively draws conclusions concerning the information content that each attribute may provide. Moreover, univariate analysis plays a key role in pointing out anomalies and non-standard values, that is, in identifying the outliers.

   Suppose that a given dataset D contains m observations and denote by $\boldsymbol{a}_j$ the generic attribute being analyzed. Since for the purpose of univariate analysis a single attribute at a time is being considered, we will denote $\boldsymbol{a}_j$ with $\boldsymbol{a}$ (j is been suppressed). Therefore, the vector of m observations will be denoted by $\boldsymbol{a} = (x_1, x_2, \dots, x_m)$.

   Graphical representations are often a starting point for exploratory data analysis in order to gain insights into the properties of a single attributes. For the purpose of a graphical representation, it is necessary to make a distinction between categorical and numerical attributes.

1.1. Graphical analysis of categorical attributes:





m=total numbers of observation

empirical frequencies **for large samples**

$$f_h = \frac{e_h}{m} \quad h \in \mathcal{H} \qquad f_h \approx \Pr\{x = v_h\} \quad h \in \mathcal{H}$$

A categorical attribute may be graphically analyzed by resorting (ricorrendo) to various representations for the empirical distribution of the observations, that is, the relative frequencies with which the different values occur. Denote by

$$V = \{v_1, v_2, \dots, v_H\}$$

the set of H distinct values that are taken by the categorical attribute $\boldsymbol{a}$, and let $\mathcal{H} = \{1, 2, \dots, H\}$. The most natural representation for the graphical analysis of a categorical attribute is a **vertical bar chart**, which indicates along the vertical axis or ordinate the empirical frequencies, that is the number of observations of the dataset corresponding to each of the values assumed by the attribute a, indicated along the horizontal axis or abscissa. Frequencies may be expressed by the relation:

$$e_h = card\,\{i \in M : x_i = v_h\}, \qquad h \in \mathcal{H}$$

Sometimes, a **horizontal bar chart** is preferable in place of a vertical bar chart. Here the positions of the vertical and horizontal axes are swapped.

It is also possible to calculate the relative empirical frequency, or empirical density:

$$f_h = \frac{e_h}{m} = \frac{card\{i \in \mathcal{M} : x_i = v_h\}}{m}, \qquad h \in \mathcal{H}$$

with which each value v_h is assumed by the attribute $\boldsymbol{a}$. For a sample of adequate size, by virtue of the central limit theorem, the relative empirical frequency represents a good approximation of the probability density of attribute a. More precisely, it can be claimed that for a sample of sufficiently large size the approximation:

$$f_h \approx p_h = \Pr\{x = v_h\}, \qquad h \in \mathcal{H}$$

holds, where p_h represents the probability that the attribute $\boldsymbol{a}$ at a new observation x will assume the value v_h.
The empirical density function may be represented by a chart similar to that for frequencies, the only difference being that the heights of the bars are divided by the number m of observations in the sample. As a consequence, it is necessary to introduce a change in the scale along the vertical axis. Sometimes, it is preferable to use a pie chart (grafico a torta) to represent the empirical density function. In thiscase, the amplitude of each sector in the circle is equal to the density associated with the corresponding value assumed by the categorical attribute. However, notice that the representationby pie charts is not so effective when one wishes to catch the differences between relativefrequencies.

1.2. Graphical analysis of numerical attributes:

For discrete numerical attributes assuming a finite and limited number of values, it is possible to resort to a bar chart representation, just as in the case of categorical attributes. In the presence of continuous or discrete attributes that might assume infinite distinct values, this type of representation cannot be used, as it would require an infinite number of vertical bars. We must therefore subdivide the horizontal axis corresponding to the values assumed by the attribute, into a finite and moderate number of intervals, usually of equal width, which in practice are considered as distinct classes. In essence, this is a discretization procedure which inevitably introduces a degree of approximation, since all the observations that fall within the same interval are considered equivalent and indistinguishable among themselves.
As a consequence, it is appropriate to carry out a partition into R intervals which should be as narrow as possible (il più stretti possible), while at the same time keeping low the number of distinct classes so generated. Once the partition has been performed, the number of observations e_r, $r = 1,2, \dots, R$, falling into each interval is counted, and a chart consisting of contiguous rectangles is generated.
The following procedure describes the generation of the intervals for the calculation of the empirical density:

Histogram for the empirical density:

- ➔ The number R of classes loosely depends (vagamente dipendenti) on the number m of observations in the sample and on the uniformity of the data. Usually, the goal is to obtain between 5 and 20 classes, making sure that in each class the frequency is higher than 5.
- → The total range and the width lr of each class is then defined. Usually the total range, given by the difference between the highest value and the lowest value of the attribute, is divided by the number of classes, so as to obtain intervals of equal width.
- → The boundaries of each class are properly assigned so as to keep the classes disjoint, making sure that no value falls simultaneously into contiguous classes.
- ➔ Finally, the number of observations in each interval is counted and the corresponding rectangle is assigned a height equal to the empirical density pr defined as

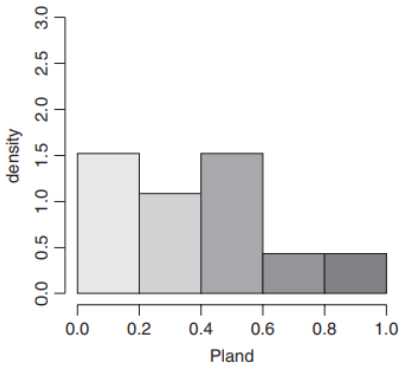
$$p_r = \frac{e_r}{ml_r}$$

Observe that the total area of the rectangles, included under the empirical density curve, is equal to 1:

$$\sum_{r=1}^{R} p_r l_r = \frac{1}{m} \sum_{r=1}^{R} e_r = 1$$

The main difference between a vertical bar chart and a histogram lies in the order of the classes along the horizontal axis. This is exactly determined by the sequence of adjacent intervals in the case of histograms, while it is arbitrary in bar charts.

The properties already highlighted with regard to the relative frequencies for categorical attributes also apply to empirical densities calculated for numerical attributes. These properties provide two interpretations for the area of the rectangles of the histogram: on the one hand, they express for each interval the percentage of observations of the dataset falling inside the interval itself; on the other hand, by virtue of the central limit theorem, the height of a rectangle approximates the probability that a new observation extracted from the population will fall within the associated interval. An example of an empirical density histogram is shown in the following figure



1.3. Measures of central tendency for numerical attributes:

We now describe the main location statistics, also called measures of central tendency.

- Mean: the best-known measure of location used to describe a numerical attribute is certainly the sample arithmetic mean, defined by the expression:

$$\bar{\mu} = \frac{x_1 + x_1 + \cdots + x_m}{m} = \frac{1}{m} \sum_{i=1}^{m} x_i \qquad \bar{\mu}$$

Since all the observations of the attribute are used to calculate the sample mean, its value is strongly affected by the extreme values in the sample. As a consequence, the sample mean value reflects the presence of outliers and therefore is not very robust. It may be that the mean significantly differs from each observation existing in the sample, making it appear a somewhat abstract concept. If the size of the dataset is sufficiently large, the sample arithmetic mean $\bar{\mu}$ approximates the theoretical mean μ of the attribute $\boldsymbol{a}$ for the entire population.

It can easily be verified that the sum of the differences between each value and the sample mean, referred to as its deviation or spread, is equal to zero, as in

$$\sum_{i=1}^{m} (x_i - \bar{\mu}) = 0$$

Moreover, the arithmetic mean is the value that minimizes the sum of squared deviations from a constant reference value:

$$\sum_{i=1}^{m}(x_i-\bar{\mu})^2 = \min_c \sum_{i=1}^{m}(x_i-c)^2$$

At times, each value x_i is found to be associated with a numerical coefficient w_i which can be interpreted as a weight of the corresponding value and used to calculate the weighted sample mean:

- Median: the median of m observations can be defined as **the central value of the observations**, assuming that the observations have been ordered in a non-decreasing way. The following procedure shows how to calculate the median:

$$\bar{\mu} = \frac{w_1x_1 + w_1x_1 + \cdots + w_mx_m}{w_1 + w_2 + \cdots + w_m} = \frac{\sum_{i=1}^{m} w_ix_i}{\sum_{i=1}^{m} w_i}$$

Median calculation:

→ if m is an odd (dispari) number, the median is the observation occupying the position $\frac{(m+1)}{2}$:

$$x^{med} = x_{(m+1)/2}$$

→ if m is an even number, the median is the middle point in the interval between the observations of position $\frac{m}{2}$ and $\frac{(m+1)}{2}$:

$$x^{med} = \frac{x_{m/2} + x_{(m+1)/2}}{2}$$

The median is affected by the number of elements in the series but not by the extreme values, and consequently it is more robust than the sample mean. Therefore, the median is suitable for asymmetric distributions and distributions with open extreme classes. However,if the data are not concentrated in the central part of the distribution, the median value loses any statistical significance.

- Mode: **it is defined as the value that corresponds to the peak of the empirical density curve** for the attribute $\boldsymbol{a}$. If the empirical density curve has been calculated by a partition into intervals, as shown for graphical methods, each value of the interval that corresponds to the maximum empirical frequency can be assumed as the mode.

1.4. Measures of dispersion for numerical attributes:

The location measures gave an indication of the central part of the observed values of a numerical attribute. However, it is necessary to define other indicators that describe the dispersion of the data, representing the level of variability expressed by the observations with respect to central values.

- Range: the simplest measure of dispersion in the range, which is defined as **the difference between the maximum and the minimum of the observations**:

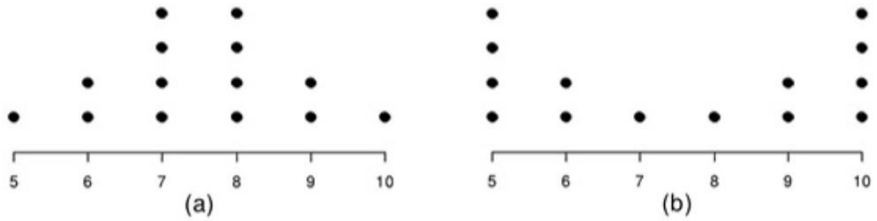
$$x^{range} = x^{max} - x^{min}$$

where:

$$x^{max} = \max_i x_i$$
$$x^{min} = \min_i x_i$$

Although the range is useful in identifying the interval in which the values of the attribute $\boldsymbol{a}$ fall, it is unable to catch the actual dispersion of the data in this interval.

For example, in the follow picture we have the same range but different dispersion. For this reason, we need to introduce another measures for to characterize dispersion.



- Mean absolute deviation: The deviation, or spread, of a value is defined as the signed difference from the sample arithmetic mean:

$$s_i = x_i - \bar{\mu}, \qquad i \in M$$

As observed when we introduced the sample mean, the equality

$$\sum_{i=1}^{m} s_i = 0$$

always holds(sempre verificato).

We can express a measure of dispersion of the observations around their sample mean through the sum of the absolute values of the spreads (differenza), called **mean absolute deviation** (MAD):

MEAN ABSOLUTE DEVIATION:

$$\text{MAD} = \frac{1}{m} \sum_{i=1}^{m} |x_i - \bar{\mu}|$$

The lower the measure of the mean absolute deviation is, the more the values fall in proximity of their sample mean and the lower the dispersion is.

- Variance: More used than the mean absolute deviation is the sample variance. There are both theoretical reasons for this, connected with the role played by the variance σ^2 of a random variable in probability theory, and practical reasons, since the mean absolute deviation is the sum of absolute values and is therefore a non-differentiable function. The sample variance is defined as:

SAMPLE VARIANCE:

$$\bar{\sigma}^2 = \frac{1}{m-1}\sum_{i=1}^{m}(x_i - \bar{\mu})^2$$

A lower sample variance implies a lower dispersion of the values around the sample mean.

As previously observed regarding the relationship between the sample mean $\bar{\mu}$ and the population mean μ, as the size of the sample increases the sample variance $\bar{\sigma}^2$ approximates the variance σ^2 of the distribution from which the values of the attribute $\boldsymbol{a}$ are drawn.

The sample variance is obtained as the mean of the squares of the deviations and therefore tends to dilate the most conspicuous errors. To bring the measure of dispersion back to the original scale in which the observations are expressed, the *sample standard deviation* is then introduced:

SAMPLE STANDARD DEVIATION:

$$\bar{\sigma} = \sqrt{\bar{\sigma}^2}$$

Basically, the variance can be used to delimit the interval around the sample mean where it is reasonable to expect that the sample values will fall. In this way, the values falling outside such interval are identified as anomalous values or outliers. To obtain the appropriate neighborhood of the sample mean, we should draw a distinction between two cases:

- Normal distribution: If the distribution of attribute $\boldsymbol{a}$ is normal, or at least bell-shaped and approximately normal, we can say that:
  - The interval $(\bar{\mu} \pm \bar{\sigma})$ contains approximately 68% of the observed values.
  - The interval $(\bar{\mu} \pm 2\bar{\sigma})$ contains approximately 95% of the observed values.
  - The interval $(\bar{\mu} \pm 3\bar{\sigma})$ contains approximately 100% of the observed values.

  The values of the sample that fall outside the interval $(\bar{\mu} \pm 3\bar{\sigma})$ can therefore be considered suspicious outliers and candidates for removal from the sample.

- Arbitrary distribution: If the distribution of the attribute $\boldsymbol{a}$ differs significantly from the normal, it is still possible to obtain intervals within which one may reasonably expect the values of the sample to fall. Such intervals apply to any distribution and are inevitably more conservative than the corresponding intervals for (approximately) normal distributions. The theoretical result that is invoked to obtain distribution-free intervals is as follows:

Theorem (Chebyshev).

Given a number $\gamma \geq 1$ and a group of m values $\mathbf{a} = (x_1, x_2, \ldots, x_m)$, a proportion at least equal to $(1 - 1/\gamma^2)$ of the values will fall within the interval $(\bar{\mu} \pm \gamma\bar{\sigma})$, namely at no more than γ standard deviations from the sample mean.

- Coefficient of variation: A further dispersion index is represented by the coefficient of variation, which is defined as the ratio between the sample standard deviation and the sample mean, expressed in percentage terms as:

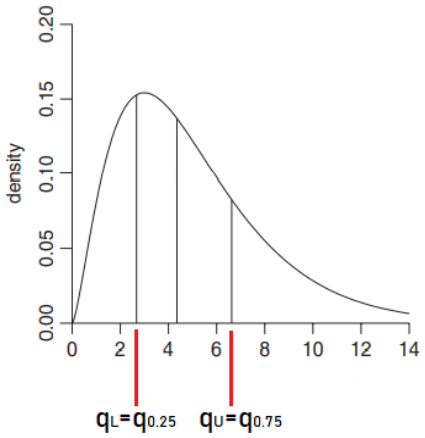
$$CV = 100\frac{\bar{\sigma}}{\bar{\mu}}$$

The coefficient of variation is used to compare two or more groups of data, usually obtained from different distributions.

1.5. Measures of relative location for numerical attributes:

Measures of relative location for a numerical attribute are used to examine the localization of a value with respect to other values in the sample.

Quantiles: suppose we arranged the m values $\{x_1, x_2, \dots, x_m\}$ of an attribute $\boldsymbol{a}$ in non-decreasing order. Given any value p, with $0 \leq p \leq 1$, the *p-order quantile* is the value q_p such that pm observations will fall on the left of q_p and the remaining $(1-p)m$ on its right. Sometimes, p quantiles are called 100 pth *percentiles*. It should be clear that the 0.5-order quantile coincides withthe median. Other noteworthy quantiles include the 0.25- and 0.75-order quantiles, respectively called the lower and upper quartiles and denoted by q_L and q_U. The two quartiles and the median divide the observations into four portions of equal size, except for the small rounding required when m is not divisible by 4. Therefore, in a density histogram the quartiles and the median split the axis of the observations into four parts, such that the area under the density curve for each portion is equal to 0.25, as shown in the following figure:

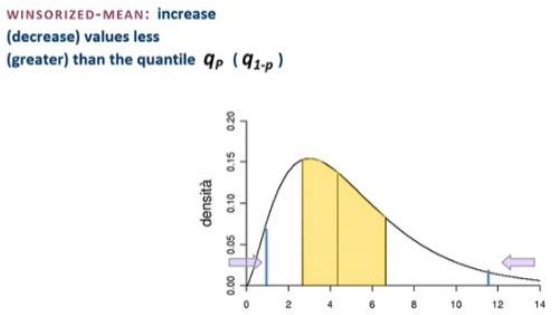


Sometimes it is useful to refer to the **interquartile range,** which is defined as the difference between the upper and the lower quartiles:

$$D_q = q_U - q_L = q_{0.75} - q_{0.25}$$

➔ Measure of central tendency based on quantiles: In order to reduce the dependence of the mean on the extreme values of the sample, one may resort to measures of central tendency based on quantiles, which in general are more robust because in this way we can cut off the outliers.

- *Mid-mean:* The mid-mean is obtained by calculating the **mean of the values** of the attribute $\boldsymbol{a}$ falling **between the lower and the upper quartiles** (media dei valori tra ql e qu).
- *Trimmed mean*: The trimmed mean is a generalization of the mid-mean, since only the values falling between quantiles of order p and $(1-p)$ are used to calculate the mean. Usually, $p = 0.05$ is the preferred choice, in order toexclude from the calculation, the lowest 5% and highest 5% of the values of $\boldsymbol{a}$. **Is the mean of the values between q_p e q_{1-p}**
- *Winsorized mean:* **increase (Decrease) values less (greater) than the quantile q_p (q_{1-p})** The winsorized mean modifies the value falling in the tails (code) according to an intuitive rule: the values lower than the p-order quantile are increased to q_p, while the values higher than the $(1-p)$ order quantile are decreased to q_{1-p}.



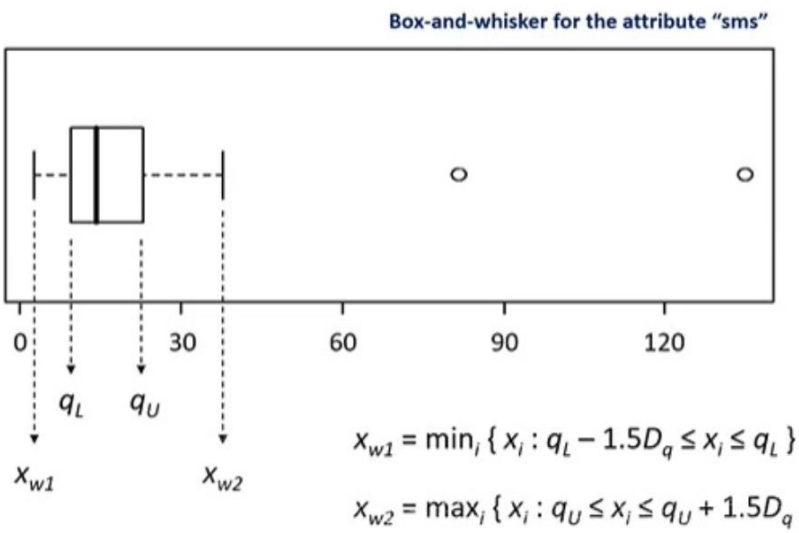
1.6. Identification of outliers for numerical attributes:

➔ BOX AND WHISKER→A way to identify outliers is based on the use of box plots, sometimes called box-and-whisker plots,in which the median and the lower and upper quartiles are represented on the axis where the observations are placed. An observation is identified as an outlier if it falls outside four threshold values, called edges (bordi), defined as:

$$\begin{aligned}
&\text{external lower edge} = q_L - 3D_q \\
&\text{internal lower edge} = q_L - 1.5D_q \\
&\text{internal upper edge} = q_U + 1.5D_q \\
&\text{external upper edge} = q_U + 3D_q
\end{aligned}$$

Using a box plot to identify outliers has the advantage of being basically independent of the extreme values of the observations, since both the median and the quartiles enjoy such independence.

We can see that the box lies between the lower and the upper quartiles. The horizontal lines departing from the box end in thick marks, called whiskers, which correspond respectively to the minimum and the maximum values of the attribute falling inside the internal edges. In general, the values lying outside the external edges are considered outliers.



$$x_{w1} = \min_i \{ x_i : q_L - 1.5D_q \le x_i \le q_L \}$$

$$x_{w2} = \max_i \{ x_i : q_U \le x_i \le q_U + 1.5D_q \}$$

1.7. Measures of heterogeneity for categorical attributes:

For categorical attributes, the foregoing measures of central tendency, dispersion and relative location **cannot be used.** For a categorical attribute $\boldsymbol{a}$ it is preferable to define some measures that express the regularity of the arrangement (disposizione) of the data $\{x_1, x_2, \ldots, x_m\}$ within the set of $\mathcal{H}$ distinct values taken by the attribute.

We can have two particular situations:

1) In particular, the **highest heterogeneity** is achieved when the relative empirical frequencies f_h are equal for all classes $h \in \mathcal{H}$. So, if I have the same probability of achieving a values for each class, this is the maximum degree of disorder because all the values have the same probability.
2) In contrast, **the lowest heterogeneity** occurs when $f_g = 1$ for a class $g \in \mathcal{H}$ and $f_g = 0$ for the remaining classes $h \in \mathcal{H}$, $h \neq g$. This is the most order situation.

We will now describe two measures of heterogeneity for categorical attributes: the Gini index and the entropy index.

- Gini index: it is defined as:

$$G = 1 - \sum_{h=1}^{H} f_h^2$$

Its value is equal to 0 in the case of lowest heterogeneity, that is, when a class is assumed at a frequency equal to 1 and all the other classes are never assumed. In contrast, when all classes have the same relative empirical frequency and the highest heterogeneity is recorded, the Gini index achieves its maximum value $(H - 1)/H$.

It is possible to normalize the Gini index so that it assumes values in the interval $[0,1]$, by using the following transformation which leads to the relative Gini index:

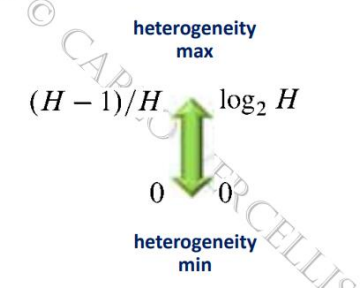
$$G_{rel} = \frac{G}{(H-1)/H}$$

- Entropy index: it is defined as:

$$E = -\sum_{h=1}^{H} f_h \log_2 f_h$$

Its value is equal to 0 in the case of lowest heterogeneity, that is, when a class is assumed at a frequency equal to 1 and all the other classes are never assumed. In contrast, when all classes have the same relative empirical frequency and the highest heterogeneity is recorded, the entropy index achieves its maximum value $log_2 H$. It is also possible to normalize the entropy index so that it assumes values in the interval $[0,1]$, by using the following transformation which leads to the relative entropy index:

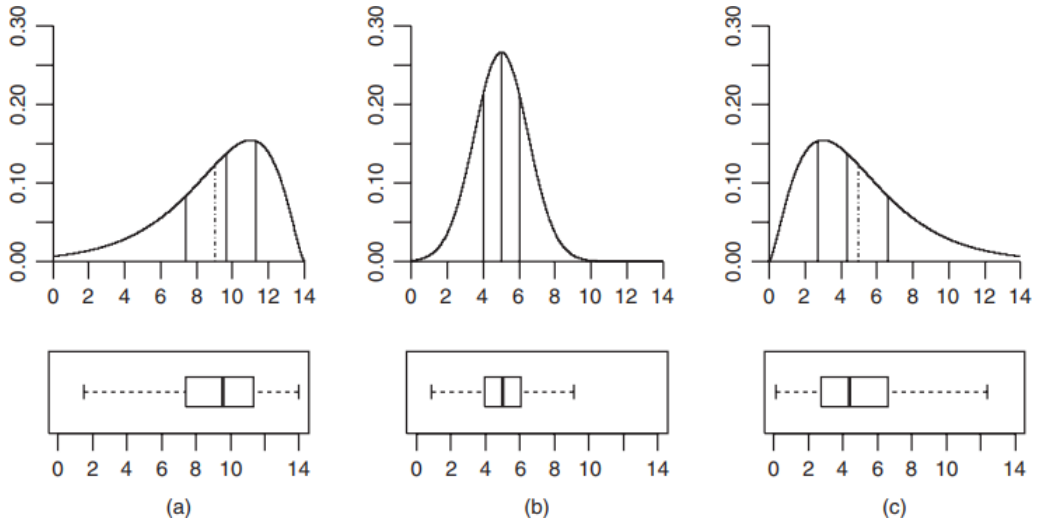
$$E_{rel} = \frac{E}{log_2 H}$$



1.8. Analysis of empirical density:

As shown at the beginning of the section, the relative empirical frequency histogram is a valuable tool for the graphical analysis of both categorical and numerical attributes. It is therefore convenient to make use of summary indicators in order to investigate the properties of the empirical density curve.

Asymmetry of the density curve: A density curve is called *symmetric* if the mean coincides with the median. From a graphical standpoint, in a symmetric density the two halves of the curve on either side of the mean, and therefore of the median, are mirror images (speculari). A density curve is said to be *asymmetric* when the mean and the median do not coincide. When the mean is greater than the median, we say that the curve is *skewed to the right*, whereas when the median is greater than the mean the curve is said *skewed to the left*. The symmetry or asymmetry of an empirical density will be apparent from graphical analysis of its histogram. The figure below shows three examples of empirical density curves, namely a curve skewed to the left (a-left asymmetry), a symmetric curve (b-symmetry) and a curve skewed to the right (c-right symmetry):



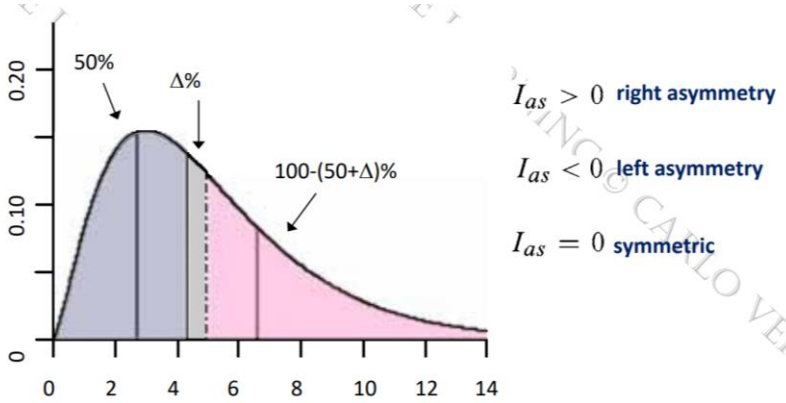
The vertical lines in the charts show the quartiles (solid line) and the mean (dashed). Box plots reflect the type and level of asymmetry in an empirical density in a particularly intuitive way. If the density is symmetric (b), the median is equally distant from the lower and upper quartiles. When it is skewed to the left (a), the median is closer to the upper quartile, while when it is skewed to the right (c) the median is closer to the lower quartile. In addition to graphical analysis, we may also make use of an **index of asymmetry**, the sample skewness, based on the third sample moment. The third sample moment is given by:

$$\bar{\mu}_3 = \frac{1}{m}\sum_{i=1}^{m}(x_i - \bar{\mu})^3$$

The sample skewness is therefore defined as:

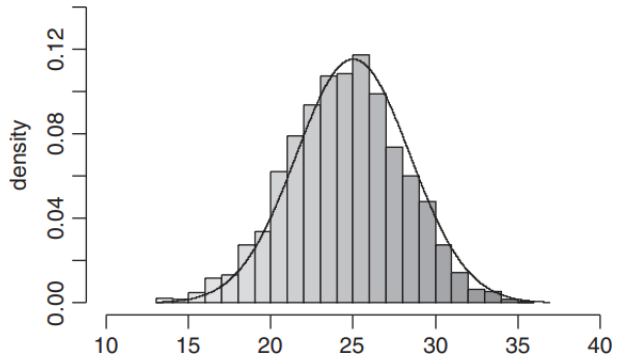
$$I_{as} = \frac{\bar{\mu}_3}{\bar{\sigma}^3}$$

If the density curve is symmetric, then $I_{skew} = 0$. If instead $I_{skew} > 0$ the density is skewed to the right. Finally, if $I_{skew} < 0$ the density is skewed to the left.



Kurtosis of the density curve: A further significant problem regarding the density histogram is the identification of the type of theoretical probability distribution, usually unknown beforehand, from which the observations are drawn. We observe that the problem of estimating an unknown distribution based on sampled data is rather complex in its general form. However, in the case of the normal distribution, a continuous distribution which occurs again and again in applications, easy graphical and summary criteria can be used to assess the level of approximation for a given empirical density. The first criterion is graphical and is based on a visual comparison between the empirical density histogram and a normal curve having mean $\bar{\mu}$ and standard deviation $\bar{\sigma}$ coinciding with those of the given density.

The following figure shows a histogram with a good approximation to the normal density:



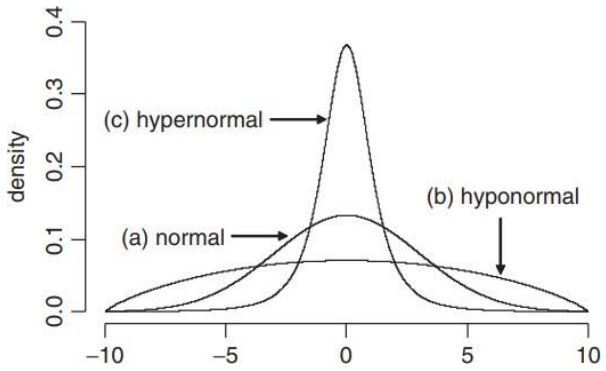
The index of kurtosis expresses in a concise way the level of approximation of an empirical density to the normal curve, and it uses the fourth sample moment:

$$\bar{\mu}_4 = \frac{1}{m}\sum_{i=1}^{m}(x_i - \bar{\mu})^4$$

The kurtosis is therefore defined as:

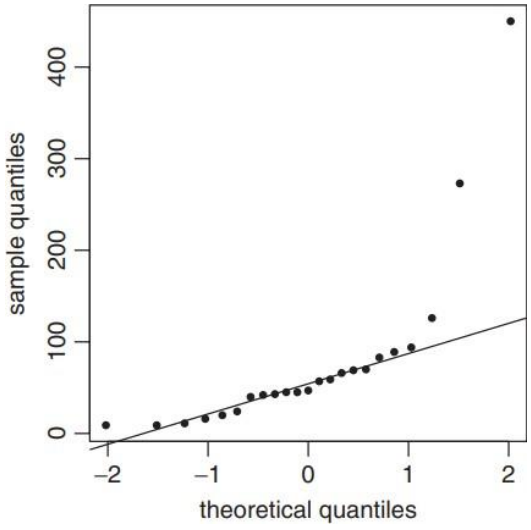
$$I_{kurt} = \frac{\bar{\mu}_4}{\bar{\sigma}^4} - 3$$

If the empirical frequency perfectly fits a normal density, as in curve (a) in the figure below, we have $I_{kurt} = 0$. If $I_{kurt} < 0$, **the empirical density is said to be *hyponormal***, since it shows a greater dispersion than the normal density and therefore assigns lower frequencies to values close to the mean, as in curve (b). Finally, **if $I_{kurt} > 0$ the e empirical density is said to be *hypernormal (is less dispersion)***, since it shows a dispersion lower than the normal density and therefore assigns higher frequencies to valuesclose to the mean, as in curve (c).

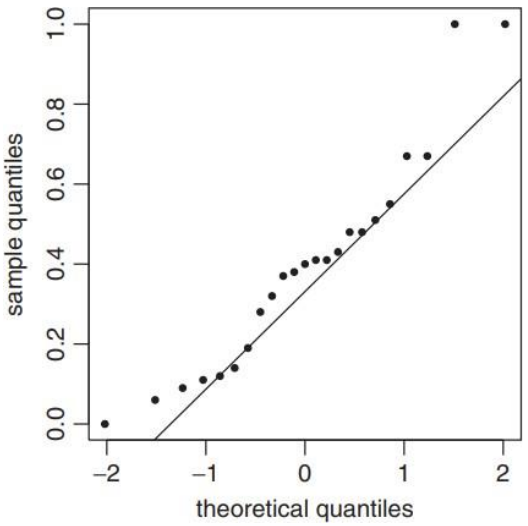


Normal probability plot: It is possible to use a different type of chart, called a normal probability plot, which allows the level of approximation of an empirical density to a normal density to be analyzed. This is a special type of quantile– quantile plot.

In a normal probability plot, the values $\{x_1, x_2, \ldots, x_m\}$ of the attribute of interest are placed in non-decreasing order on the vertical axis. Let q_i be the order of the quantile corresponding to the value x_i. The quantiles N_i of order q_i, i $\in M$ of the normal distribution are then placed on the horizontal axis and the points with coordinates $(N_i, x_i), i \in M$, are plotted on the Cartesian plane. It is possibleto prove that if the values of the attribute of interest are sampled from a normal distribution, and therefore the empirical density is asymptotically normal as the size of the dataset increases, then the points on the normal probability plot will fall on a straight line. Conversely, the more the empirical density diverges from normal, the more the points of the normal probability plot diverge from a linear trend. The figure below shows the normal probability plot of an attribute:



The almost concave curve defined by the points in the plot indicates that the empirical density is skewed to the right and is a hyponormal curve. Contrast this with the normal probability plot for another attribute in the figure below:



The S-shaped curve indicates the occurrence of a hypernormal distribution, and also that the empirical density is significantly far from a normal curve.

2. BIVARIATE ANALYSIS (we will consider two variables):

Once the characteristics of each attribute contained in a dataset D have been investigated using the methods for univariate exploratory analysis described in the previous section, it is appropriate to exploit the relationships existing between pairs of attributes through bivariate analysis. We will denote by $\boldsymbol{a}_j$ and $\boldsymbol{a}_k$ a generic pair of attributes to be analyzed.

It is useful to distinguish three cases that may occur within bivariate analysis:

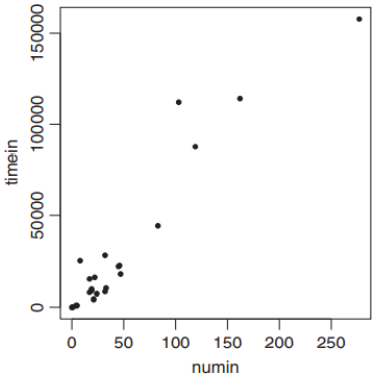
- both attributes are numerical.
- one attribute is numerical, and the other is categorical.
- both attributes are categorical.

In supervised learning problems it is usually important to investigate the relationship between the target attribute and each explanatory attribute. Hence, it frequently happens that one of the attributes of the pair $\{\boldsymbol{a}_j, \boldsymbol{a}_k\}$ represents the target of a supervised learning problem, which is categorical for classification and numerical for regression. In particular, we will denote by $\boldsymbol{a}_j = (x_{1j}, x_{2j}, \ldots, x_{mj})$ and $\boldsymbol{a}_k = (x_{1k}, x_{2k}, \ldots, x_{mk})$ the vectors composed of m observations corresponding to the two attributes considered.

2.1. Graphical analysis:

There are several types of graphical representations that allow the relationship between two attributes to be visualized. For each type of chart, we will indicate the categories of attributes to which it can be applied.

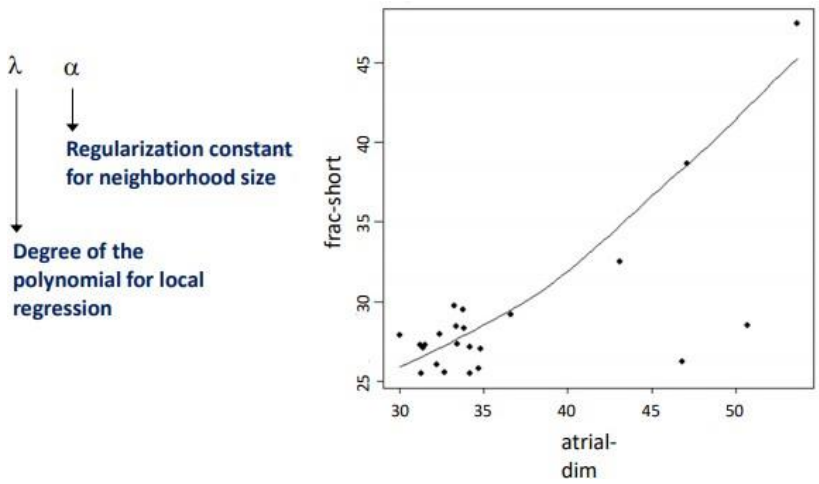
- Scatter plots: A scatter plot is definitely the most intuitive graphical representation of the relationship between two numerical attributes. This is a two-dimensional Cartesian chart obtained by placing the first attribute $\boldsymbol{a_j}$ on the horizontal axis and the second attribute $\boldsymbol{a_k}$ on the vertical axis. The points plotted thus correspond to the pairs of values (x_{ij}, x_{ik}), $i \in M$, of the dataset D.



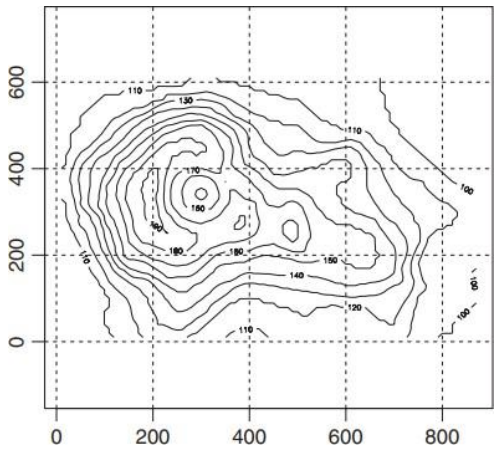
- Loess plots: Loess plots are based on scatter plots and can therefore be applied in turn to pairs of numerical attributes. Starting from a scatter plot, it is possible to add a trend curve to express the functional relationship between the attribute $\boldsymbol{a_k}$ and the attribute $\boldsymbol{a_j}$. The trend curve can be obtained using local regression techniques. This explains the term loess, which stands for local regression. To calculate the value of the trend curve at any point, a regression model is applied.

  The loess curve is regulated by two parameters, described below:

  - The degree $\lambda > 0$ of the polynomial that represents the local regression curve; usually the value 1 or 2 is assigned to λ in order to obtain **a linear or quadratic local approximation.**
  - The regularization constant $\alpha > 0$, which regulates the size of the neighborhood. When the value of α is increased, the regularization of the trend line is strengthened, since more observations are used to calculate the regression. Conversely, when α decreases the trend line sticks more closely to the observations, although the generalization capability of the model decreases. In general, values of α ranging between 1/4 and 1 are chosen.

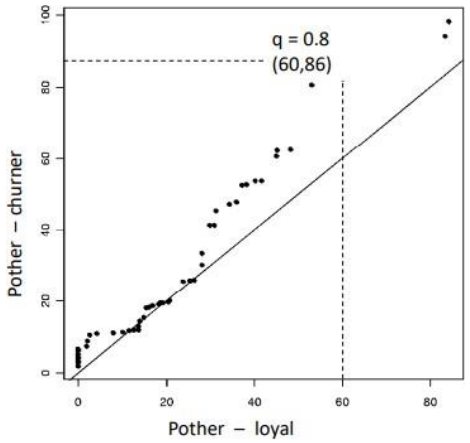


- Level curves (countour lines) : Level curves are a further development of scatter plots and can only be used for numerical attributes. They highlight (evidenziano) the value of a third numerical attribute $\boldsymbol{a}_z$ as the attributes $\boldsymbol{a}_j$ and $\boldsymbol{a}_k$ placed on the axes of the plot vary. Connecting to each other the pointsin the plot that share the value of the third attribute, possibly by using some form of numerical interpolation, curved lines are obtained representing the geometric locus of the points for which the attribute $\boldsymbol{a}_z$ assumes a given value.

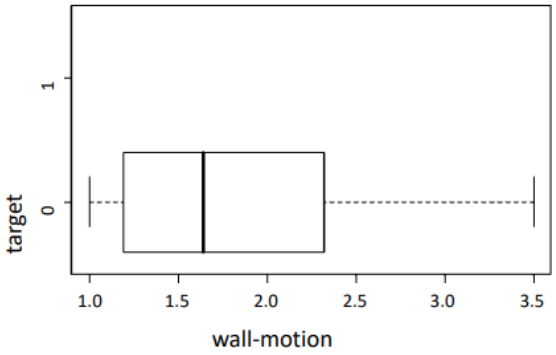


An example of level curves known to everyone are contour lines, which can be obtained by highlighting in a geographical map the points that share the same altitude, which in this case corresponds to the attribute $\boldsymbol{a}_z$

- Quantile-quantile plots (QQ plots): QQ plots are used to compare the distributions of the same attribute for two different characteristics of the population or for samples extracted from two different populations.
  The analysis is applicable to numerical attributes and is carried out by comparing (svolta comparando) thequantiles of the two series of observations. It is also possible to obtain a QQ plot even when one of the series is more populated than the other.
  Suppose that we have calculated and arranged in non-decreasing order the values of the quantiles $\{q_{lj}\}$ and $\{q_{lk}\}$, $l \in f$, for each of the two attributes at a predetermined series f of quantile orders, such as the consequence of deciles. The coordinates of the point in the plot are (x_{lj}, x_{lk}), , $l \in f$. If the two series of data are extracted from the same probability distribution, the points in the chart will fall on a straight line with slope 45 degrees, usually depicted in the QQ plot.



- Box plots: box plots can be used for the identification of outliers. They are also useful for comparing the distributions of the same variable for observations belonging to distinct groups. This analysis technique is applicable to a pair consisting of a numerical attribute and a categorical attribute. Often, though not necessarily, the categorical attribute represents the target in a supervised learning process.



2.2. Measures of correlation for numerical attributes:

As in the case of univariate analysis, besides graphical methods it is useful to introduce summary indicators that express the nature and intensity of the relationship between numerical attributes.

- Covariance: given the pair of attributes $\boldsymbol{a}_j$ and $\boldsymbol{a}_k$, let $\bar{\mu}_j$ and $\bar{\mu}_k$ be the corresponding sample means. The ***sample covariance*** is defined as:

SAMPLE COVARIANCE:

$$v_{jk} = \text{cov}(\mathbf{a}_j, \mathbf{a}_k) = \frac{1}{m-2} \sum_{i=1}^{m} (x_{ij} - \bar{\mu}_j)(x_{ik} - \bar{\mu}_k)$$

The sample covariance can be easily interpreted. Indeed, if values the attribute $\boldsymbol{a}_j$ greater than the mean $\bar{\mu}_j$ are associated with values of the attribute $\boldsymbol{a}_k$ also greater than the mean

$\bar{\mu}_k$, the two elements of the product in the summation above will agree in sign and therefore they will provide a positive contribution to the sum. By the same token, if values the attribute $\boldsymbol{a}_j$ lower than the mean $\bar{\mu}_j$ are associated with values of the attribute $\boldsymbol{a}_k$ lower than the mean $\bar{\mu}_k$, the two elements of the product in the summation above will still agree in sign andagain will provide a positive contribution to the sum. In these cases, the attributes $\boldsymbol{a}_j$ and $\boldsymbol{a}_k$ are said to be in *concordance* with one another. Conversely, if values the attribute $\boldsymbol{a}_j$ greater

than the mean $\bar{\mu}_j$ are associated with values of the attribute $\boldsymbol{a}_k$ lower than the mean $\bar{\mu}_k$, the two elements of the product in the summation above will not agree in sign and therefore they will provide a negative contribution to the sum. The same occurs if values of the attribute $\boldsymbol{a}_j$ lower than the mean $\bar{\mu}_j$ are associated with values of the attribute $\boldsymbol{a}_k$ greater than the mean $\bar{\mu}_k$. In these cases, the attributes $\boldsymbol{a}_j$ and $\boldsymbol{a}_k$ are said to be ***discordant*** with one another.

We can therefore conclude that positive covariance values indicate that the attributes $\boldsymbol{a}_j$ and $\boldsymbol{a}_k$ are concordant, while negative covariance values indicate that the attributes are discordant.

- Correlation: the covariance is usually a number ranging on a variable scale and is therefore inadequate to assess the intensity of the relationship between the two attributes. For this reason, the linear correlation coefficient between two attributes, also termed the *Pearson coefficient*, is more useful. It is defined as:

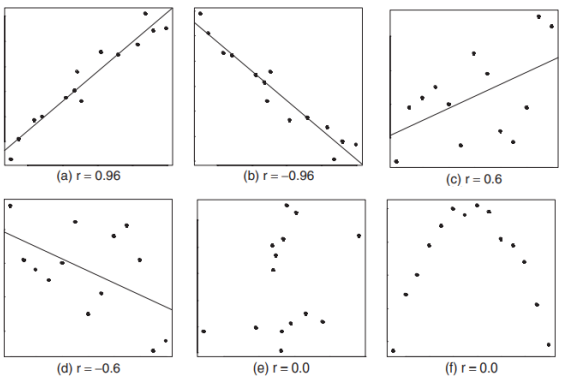
where $\bar{\sigma}j$ and $\bar{\sigma}$k are the sample standard deviations of $\boldsymbol{a}_j$ and $\boldsymbol{a}_k$, respectively.

It can be proven that the maximum value achievable by cov $(\boldsymbol{a}_j, \boldsymbol{a}_k)$ is equal to the product $\bar{\sigma}_j \bar{\sigma}_k$ of the sample standard deviations, while the minimum value is equal to $-\bar{\sigma}_j \bar{\sigma}_k$. As a result, the linear correlation coefficient r_{jk} always lies in the interval $[-1, 1]$, and represents a relative index expressing the intensity of a possible linear relationship between the attributes $\boldsymbol{a}_j$ and $\boldsymbol{a}_k$. The main properties of the linear correlation coefficient r_{jk} can be summarized as follows:

SAMPLE LINEAR CORRELATION :

$$r_{jk} = \text{corr}(\mathbf{a}_j, \mathbf{a}_k) = \frac{v_{jk}}{\bar{\sigma}_j \bar{\sigma}_k} \qquad r_{jk} \in [-1, 1]$$

- If $r_{jk} > 0$ the attributes are concordant. This means that if the pairs of observations are represented on a scatter plot, they will show a trend consisting of a straight line with a positive slope. The approximation to the line increases as r_{jk} gets closer to 1. If $r_{jk} = 1$ the points will lie exactly on a straight line.
- $r_{jk} < 0$ the attributes are discordant. In this case the pairs of observations represented on a scatter plot will tend to lie on a line with a negative slope. The approximation to the line increases as r_{jk} gets closer to −1. If $r_{jk} = -1$ the points will lie exactly on a straight line.
- If $r_{jk} = 0$, or at least $r_{jk} \approx 0$, no linear relationship exists between the two attributes. In this case, the pairs of values on a scatter plot either are placed in a random way or tend to lie on a nonlinear curve.



The figure above shows the linear correlation coefficients for different sets of data. In particular, notice that the linear regression coefficient can be close to zero when a strong nonlinear relationship exists between the two attributes.

2.3. Contingency tables for categorical attributes:

When dealing with a pair of categorical attributes $\boldsymbol{a}_j$ and $\boldsymbol{a}_k$, let

$$V = \{v_1, v_2, \dots, v_J\}, \qquad U\{u_1, u_2, \dots, u_K\}$$

Denote the sets of distinct values respectively assumed by each of them. A contingency table is defined as a matrix $\boldsymbol{T}$ whose generic element t_{rs} indicates the frequency with which the pair of values $\{x_{ij} = v_r\}$ and $\{x_{ik} = u_s\}$ appears in the records of the dataset D.

	family		
area	0	1	totale
1	2	4	6 (f_1)
2	4	2	6
3	2	5	7
4	3	3	6
totale	11 (g_1)	14 (g_2)	25

It is also possible to compute the sum of the values for each row and for each column of the contingency table, obtaining the marginal frequencies:

$$f_r = \sum_{s=1}^{K} t_{rs}, \qquad g_r = \sum_{r=1}^{J} t_{rs}$$

Two attributes $\boldsymbol{a}_j$ and $\boldsymbol{a}_k$ are said to be statistically independent if the following conditions occur:

$$\frac{t_{r1}}{g_1} = \frac{t_{r2}}{g_2} = \cdots = \frac{t_{rK}}{g_K}, \qquad r = 1, 2, \dots, J$$

It can be shown that the equalities above are equivalent to the conditions:

$$\frac{t_{1s}}{f_1} = \frac{t_{2s}}{f_2} = \cdots = \frac{t_{Js}}{f_J}, \qquad s = 1, 2, \dots, K$$

which in turn can be used to define the concept of independence between categorical attributes. Intuitively, the two attributes are independent if the analysis of $\boldsymbol{a}_j$ in relation to the second attribute is $\boldsymbol{a}_k$ equivalent to the univariate analysis of $\boldsymbol{a}_j$.

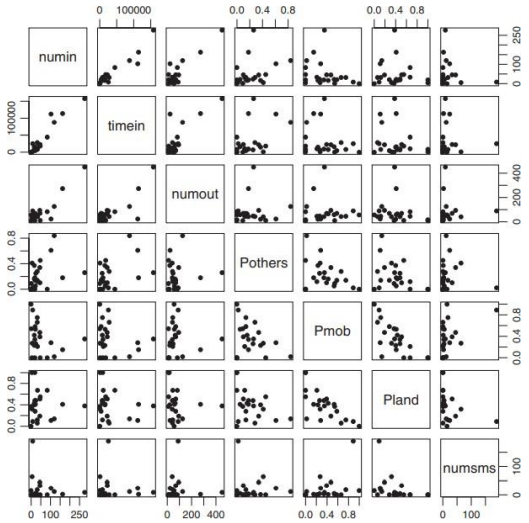
3. MULTIVARIATE ANALYSIS:

The purpose of multivariate analysis is to extend the concepts introduced for the bivariate case in order to assess the relationships existing among multiple attributes in a dataset.

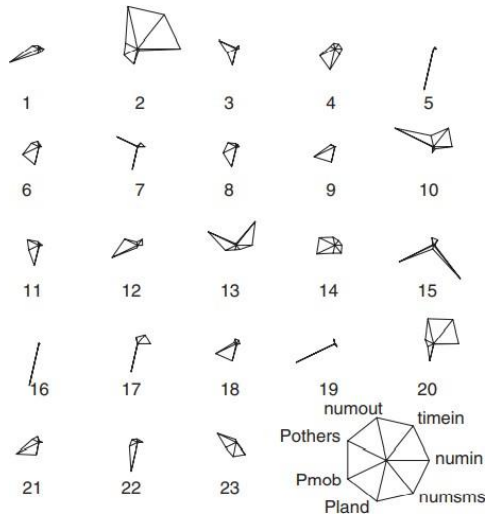
3.1. Graphical analysis:

Notice that all methods for graphical analysis described in this section exclusively apply to numerical attributes.

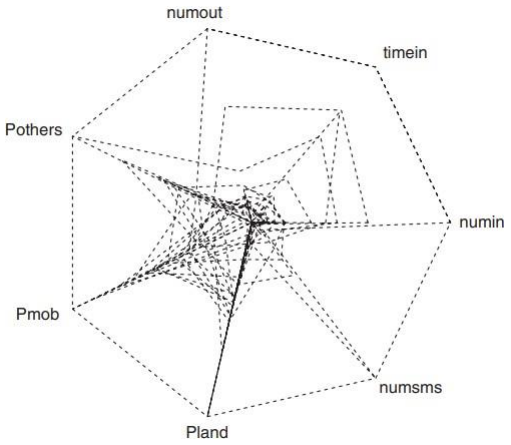
- Scatter plot matrix: Since scatter plots show in an intuitive way the relationships between pairs of numerical attributes, in the case of multivariate analysis it is natural to consider matrices of plots evaluated for every pair of numerical variables. In this way, it is possible to visualize the nature and intensity of the pairwise relationships in a single chart.



- Star plots: Star plots belong to the broader class of icon-based charts. They show in an intuitive way the differences among values of the attributes for the records of a dataset. To be effective, they should be applied to a limited number of observations, say no more than a few dozen, and the comparison should be based on a small number of attributes.
  The basic concept involves matching each record with a star-shaped icon, from the center of which depart as many rays as the number of attributes. The length of each ray is equal to the value of the corresponding attribute, normalized to fall in the interval $[0,1]$ so as to give a consistent representation of the various attributes.



- Spider web chart**:** Spider web charts are grids where the main rays correspond to the attributes analyzed. For every record, the position is calculated on each ray, based on the value of the corresponding attribute. Finally, the points so obtained on the rays for each record are sequentially connected to each other, thus creating a circuit for every record in the dataset.



3.2. Measures of correlation for numerical attributes:

For multivariate analysis of numerical attributes, covariance and correlation matrices are calculated among all pairs of attributes. For notational convenience, we will suppose that all the n attributes of the dataset D are numerical, having removed for the time being any categorical attribute. Let $\boldsymbol{V}$ and $\boldsymbol{R}$ be the two $n \times n$ square matrices whose elements are represented by the covariance values and by the correlations. Both matrices $\boldsymbol{V}$ and $\boldsymbol{R}$ are symmetric and positive definite. Notice that the covariance matrix $\boldsymbol{V}$ contains on its main diagonal the sample covariance values of each single attribute, and for this reason it is also called the *variance–covariance matrix*.

In order to devise a summary indicator that expresses the total variability of the dataset, and compares different datasets, the trace of the matrix $\boldsymbol{V}$ can be used, defined as the sum of the elements along its main diagonal:

$$tr(\boldsymbol{V}) = \sum_{j=1}^{n} v_{jj} = \sum_{j=1}^{n} \bar{\sigma}_{jj}^2$$

It can be shown that the trace equals the sum of the eigenvalues λ_j, $j \in \mathrm{N}$, of the covariance matrix $\boldsymbol{V}$**:**

$$tr(\boldsymbol{V}) = \sum_{j=1}^{n} \lambda_j$$

CLASSIFICATION

If the target is categorical we call this task classification
If the target is numerical we call this task regression

Let's start with classification…

Classification models are supervised learning methods for predicting the value of a categorical target attribute, unlike regression models which deal with numerical attributes. Starting from a set of past observations whose target class is known, classification models are used to generate a set of rules that allow the target class of future examples to be predicted. From a theoretical viewpoint, the development of algorithms capable of learning from past experience represents a fundamental step in emulating theinductive capabilities of the human brain. On the other hand, the opportunities afforded by classification extend into several different application domains.

1. **CLASSIFICATION PROBLEMS:**

   In a classification problem, we have a dataset D containing m observations described in terms of n explanatory attributes (esplicativi) and a categorical target attribute.

   The explanatory attributes, also called *predictive variables*, may be partly categorical and partly numerical. The target attribute is also called a *class* or a *label*, while the observations are also termed *examples* or *instances*. Unlike regression (a differenza), for classification models the target variable takes a finite number of values. In particular, we have a **binary classification** problem if the instances belong to two classes only, and a **multiclass or multicategory classification** if there are more than two classes.

   The purpose of a classification model is to identify recurring relationships among the explanatory variables which describe the examples belonging to the same class. Such relationships are then translated into classification rules which are used to predict the class of examples for which only the values of the explanatory attributes are known. The rules may take different forms depending on the type of model used.

   From a mathematical viewpoint, in a classification problem m known examples are given, consisting of pairs $(\boldsymbol{x}_i, y_i),\ i \in M$, where $\boldsymbol{x}_i \in \mathbb{R}^n$ is the vector of the values taken by the n predictive attributes for the ith example and $y_i \in \mathcal{H} = \{v_1, v_2, \dots, v_H\}$ denotes the corresponding target class. Each component x_{ij} of the vector $\boldsymbol{x}_i$ is regarded as a realization of the random variable X_j, $j \in N$, which represents the attribute $\boldsymbol{a}_i$ in the dataset D.

   In a binary classification problem, one has $\mathcal{H} = 2$, and the two classes may be denoted as $\mathcal{H} = \{0, 1\}$ or as $\mathcal{H} = \{-1, 1\}$, without loss of generality.

   Let $\mathcal{F}$ be a class of functions $f(\boldsymbol{x})$: $\mathbb{R}^n \to \mathcal{H}$ called ***hypothesis*** that represent hypothetical relationships of dependence between y_i and $\boldsymbol{x}_i$. A classification problem consists of defininf an appropriate hypothesis space $\mathcal{F}$ and an algorithm $A_{\mathcal{F}}$ that identifies a function $f^* \in \mathcal{F}$ that can optimally describe the relationship between the predictive attributes and the target class.

attributes

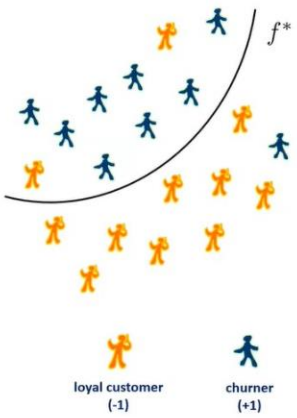
customers

Area	Pothers	Pmob	…	NumSMS	Class
2	0.14	0.59	…	18	1
3	0.26	0.35	…	9	-1
1	0.37	0.23	…	1	1
⋮	⋮	⋮	⋮	⋮	⋮
4	0.41	0.27	…	64	-1

past data

Area	Pothers	Pmob	…	NumSMS	Class
1	0.27	0.67	…	36	?
4	0.44	0.22	…	50	?
4	0.31	0.47	…	14	?
⋮	⋮	⋮	⋮	⋮	⋮
2	0.31	0.14	…	49	?

future data



Classification: prediction of HIV protease-cleavable peptides

attributes

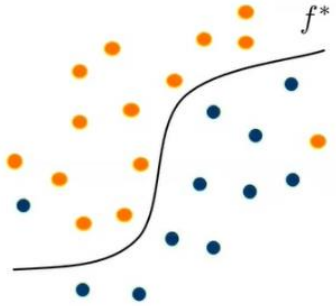
peptides

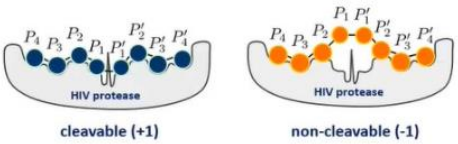
P_4	P_3	P_2	P_1	P_1'	P_2'	P_3'	P_4'	Class
T	Q	I	M	F	E	T	F	1
G	Q	V	N	Y	E	E	F	1
Y	T	F	L	F	M	N	S	-1
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
D	T	V	L	E	E	M	S	1

past data

P_4	P_3	P_2	P_1	P_1'	P_2'	P_3'	P_4'	Class
S	Q	N	Y	P	I	V	Q	?
T	Q	I	M	F	E	T	F	?
G	L	A	A	P	Q	F	S	?
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
R	Q	A	G	F	L	G	L	?

future data



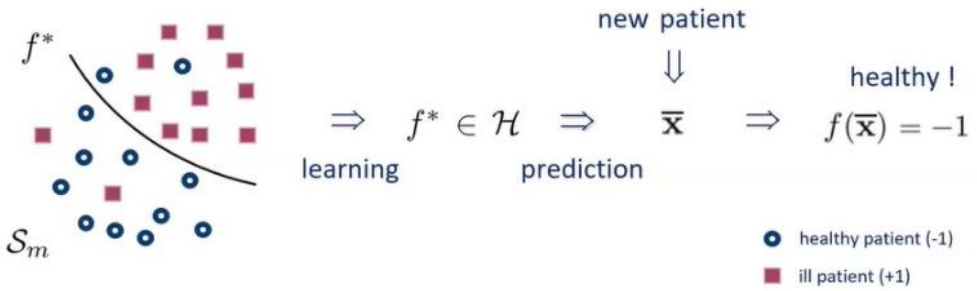


cleavable (+1)

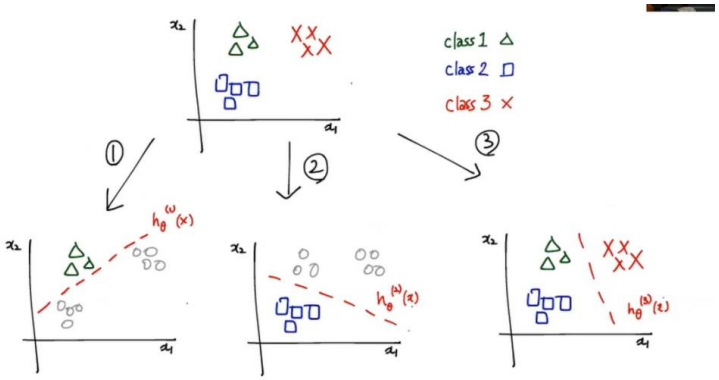
non-cleavable (-1)

Classification as a learning task

- $\mathcal{S}_m = \{(\mathbf{x}_i, y_i),\ i \in \mathcal{M}\}$: **training set, where** $\mathbf{x}_i \in \Re^n$ **and** $y_i \in \mathcal{D}$
- $\mathcal{H}$ **denotes a set of functions** $f(\mathbf{x}) : \Re^n \mapsto \mathcal{D}$
- **Classification problem: define a hypotheses space** $\mathcal{H}$ **and a function** $f^* \in \mathcal{H}$ **which optimally describes the relationship between** $\mathbf{x}_i$ **and** y_i

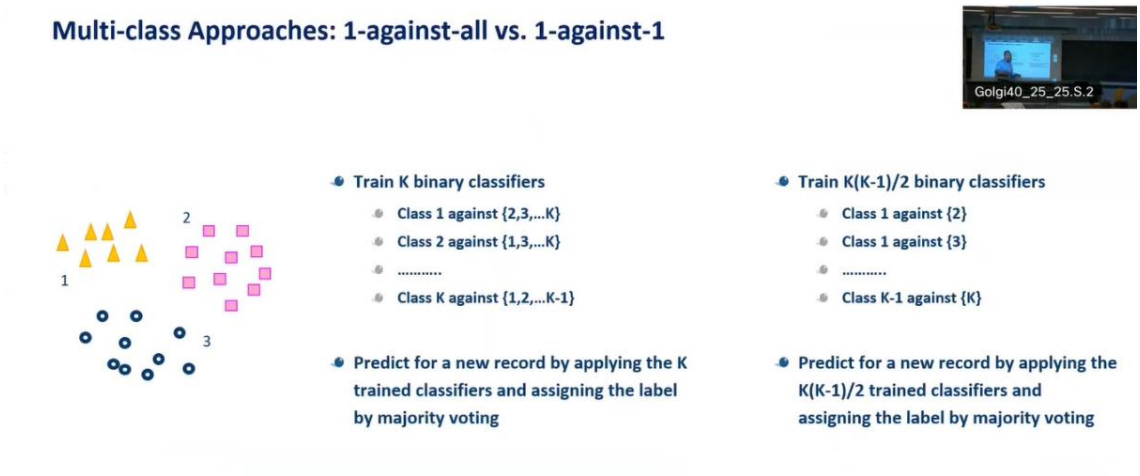


Se però il problema non è di classificazione binaria dobbiamo cercare di ridurlo a tale classificazione→ you need a way of being able to precict more than two classes→ tecnica uno contro tutti (ONE-VERSUS-ALL)



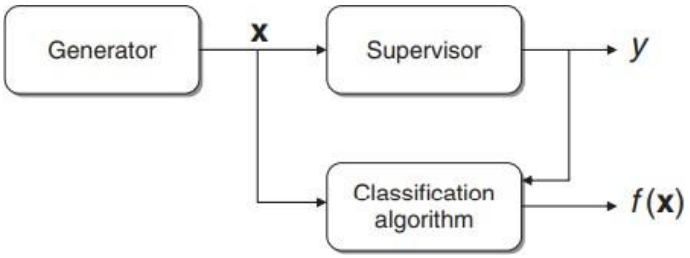
But there is a problem with the one-versus-all approach to multi-class classification.
You often have imbalance in the number of examples of subjects in one class verus all (es: 3 triangoli + 4

quadrati = **7 vs 4** x). When there are less examples for a model to train on, the model usually doesn't predict as well on those examples. If you give your model a lot of examples of one classes and a very small number of examples of the other class, it's not to going to predict well on the other class→ so we will use one versus one.



The joint probability distribution $P_{\boldsymbol{x},y}(\boldsymbol{x}, y)$ of the examples in the dataset D, defined over the space $\mathbb{R}^n \times \mathcal{H}$, is generally unknown and most classification models are non-parametric, in the sense that they do not make any prior assumption on the form of the distribution $P_{\boldsymbol{x},y}(\boldsymbol{x}, y)$.

The flow diagram shown below may clarify the probability assumptions concerning the three components of a classification problem: a generator of observations, a supervisor of the target class and a classification algorithm:



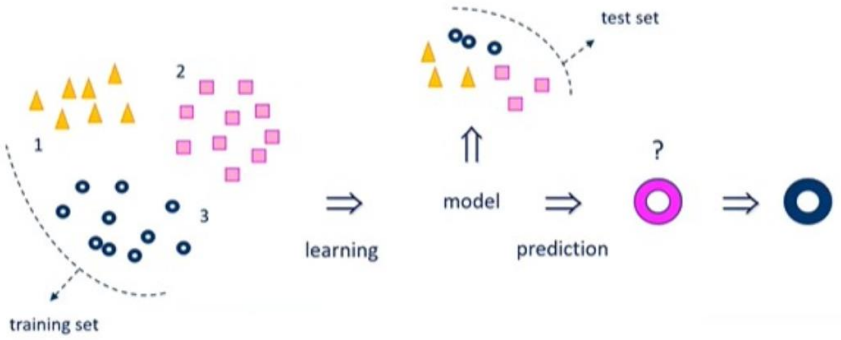
- Generator: The task of the generator is to extract random vectors $\boldsymbol{x}$ of examples according to an unknown probability distribution $P_{\boldsymbol{x}}(\boldsymbol{x})$.
- Supervisor: The supervisor returns for each vector $\boldsymbol{x}$ of examples the value of the target class according to a conditional distribution $P_{y|\boldsymbol{x}}(y|\boldsymbol{x})$ which is also unknown.
- Algorithm: A classification algorithm $A_{\mathcal{F}}$ also called a *classifier*, chooses a function $f^* \in \mathcal{F}$ in the hypothesis space so as to minimize a suitably defined loss function.

Classification models, just like regression models, are suitable for both interpretation and prediction, with general reference to supervised learning methods. Simpler models usually yield intuitive classification rules that can be easily interpreted, while more advanced models derive less intelligible rules although they usually achieve better prediction accuracy.
A portion of the examples in the dataset D is used for training a classification model, that is, for deriving the functional relationship between the target variable and the explicative variables expressed by means of the hypothesis $f^* \in \mathcal{F}$. What remains of the available data is used later to evaluate the accuracy of the

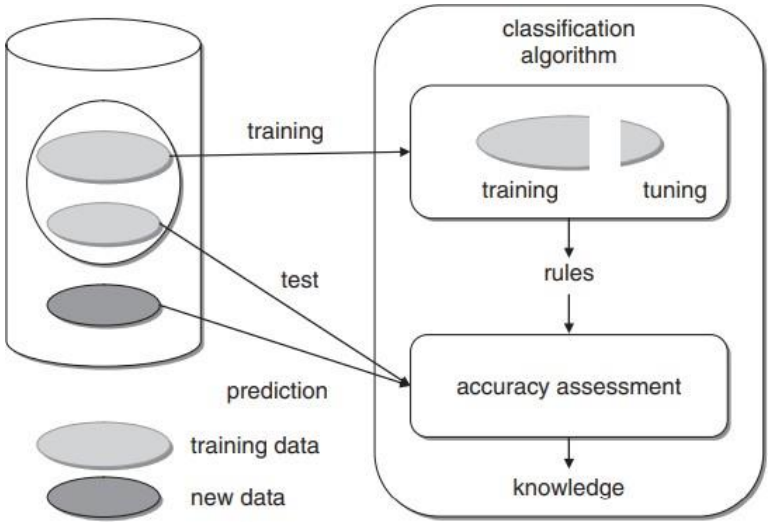
generated model and to select the best model out of those developed using alternative classification methods.

The development of a classification model consists therefore of three main phases:

- Training phase: During the training phase, the classification algorithm is applied to the examples belonging to a subset T of the dataset D, called the training set, in order to derive classification rules that allow the corresponding target class y to be attached to each observation $\boldsymbol{x}$.
- Test phase: In the test phase, the rules generated during the training phase are used to classify the observations of D not included in the training set, for which the target class value is already known. To assess the accuracy of the classification model, the actual target class of each instance in the test set $\mathcal{V} = D - \mathrm{T}$ is then compared with the class predicted by the classifier. To avoid an overestimate of the model accuracy, the training set and the test set must be disjoint.
- Prediction phase: The prediction phase represents the actual use of the classification model to assign the target class to new observations that will be recorded in the future. A prediction is obtained by applying the rules generated during the training phase to the explanatory variables that describe the new instance.



The figure below shows the logical flow of the learning process for a classification algorithm. In addition to the phases described above, the figure also shows a portion of the training data, called the ***tuning set*****,** which is used by some classification algorithms to identify the optimal value of some parameters appearing in the functions $f \in \mathcal{F}$.



1.1. Taxonomy of classification models:

We can distinguish four main categories of classification models:

- Heuristic models: Heuristic methods make use of classification procedures based on simple and intuitive algorithms. This category includes ***nearest neighbor methods*****,** based on the concept of distance between observations, and ***classification trees***, which use divide-and-conquer schemes to derive groups of observations that are as homogeneous as possible with respect to the target class.
- Separation models: Separation models divide the attribute space $\mathbb{R}^n$ into H disjoint regions $\{S_1, S_2, \dots, S_H\}$, separating the observations based on the target class. The observations $\boldsymbol{x}_i$ in region S_h are assigned to class $y_i = v_h$. Each region S_h may comprise a composite set obtained from set-theoretic operations of union and intersection applied to regions of an elementary form, such as halfspaces or hyperspheres. However, the resulting regions should be kept structurally simple, to avoid compromising the generalization capability of the model. In general, it is hard to determine a collection of simple regions that exactly subdivide the observations of the dataset based on the value of the target class. Therefore, a loss function is defined to take the misclassified points into account, and an optimization problem is solved in order to derive a subdivision into regions that minimizes the total loss. The various classification models that belong to this category differ from one another in terms of the type of separation regions, the loss function, the algorithm used to solve the optimization problem. The most popular separation techniques include ***discriminant analysis*****,** ***perceptron methods*****,** ***neural networks,*** **and** ***support vector machines*****.** Some variants of classification trees can also be placed in this category.
- Regression models: Regression models, described in Chapter 8 for the prediction of continuous target variables, make an explicit assumption concerning the functional form of the conditional probabilities $P_{y|\boldsymbol{x}}(y|\boldsymbol{x})$, which correspond to the assignment of the target class by the supervisor. For linear regression models, it is assumed that a linear relationship exists between the dependent variable and the predictors, and this leads to the derivation of the value of the regression coefficients. Extension of linear regression suited to handling (adatto a gestire) binary classification problems. ***Ex: logist regression***
- Probabilistic models: In probabilistic models, a hypothesis is formulated regarding the functional form of the conditional probabilities $P_{\boldsymbol{x}|y}(\boldsymbol{x}|y)$ of the observations given the target class, known as class-conditional probabilities. Subsequently, based on an estimate of the prior probabilities $P_y(y)$ and using Bayes' theorem, the posterior probabilities $P_{y|\boldsymbol{x}}(y|\boldsymbol{x})$ of the target class assigned by the supervisor can be calculated. The functional form of $P_{\boldsymbol{x}|y}(\boldsymbol{x}|y)$ may be parametric or nonparametric. Naive Bayes classifiers and Bayesian networks are well-known families of probabilistic methods. ***Bayesian methods***

Besides the taxonomy described above, it should be mentioned that most classifiers also generate a *score function* $g(\boldsymbol{x}) : \mathbb{R}^n \to \mathbb{R}$ which associates each observation $\boldsymbol{x}$ with a real number. Possibly after standardization, the score function may be interpreted as an estimate of the probability that the class predicted by the classifier for the observation $\boldsymbol{x}$ is correct. This property is evident in probabilistic models. In separation models, the score of an observation may be set equal to its distance from the boundary of the region corresponding to its target class value. Finally, in classification trees the score function is associated with the density of observations having the same target class which have been included in the leaf node to which the observation itself has been assigned. Starting from the score function g, it is possible to derive a classification rule to predict the target class of the observation $\boldsymbol{x}$.

2. EVALUATION OF CLASSIFICATION MODELS:

Within a classification analysis it is usually advisable to develop alternative models and then select the method affording the best prediction accuracy. To obtain alternative models one may vary the method and also modify the values of the parameters involved. Classification methods can be evaluated based on several criteria, as follows:

- **Accuracy:** Evaluating the accuracy of a classification model is crucial for two main reasons. First, the accuracy of a model is an indicator of its ability to predict the target class for future observations. Based on their accuracy values, it is also possible to compare different models in order to select the classifier associated with the best performance.

  Let $\mathcal{T}$ be the training set and $\mathcal{V}$ the test set, and t and v be the numbers of observations in each subset, respectively. The relation $D = \mathcal{T} \cup \mathcal{V}$ and $m = t + v$ obviously hold. The most natural indicator of the accuracy of a classification model is the proportion of observations of the test set $\mathcal{V}$ correctly classified by the model. If y_i denotes the class of the generic observation $\boldsymbol{x}_i \in \mathcal{V}$ and $f(\boldsymbol{x}_i)$ the class predicted through the function $f \in \mathcal{F}$ identified by the learning algorithm $A = A_{\mathcal{F}}$, the following loss function can be defined:

$$L(y_i, f(\boldsymbol{x}_i)) = \begin{cases} 0, & if\ y_i = f(\boldsymbol{x}_i) \\ 1, & if\ y_i \neq f(\boldsymbol{x}_i) \end{cases}$$

  The accuracy of model A can be evaluated as:

$$acc_A(\mathcal{V}) = acc_{A_{\mathcal{F}}}(\mathcal{V}) = 1 - \frac{1}{v}\sum_{i=1}^{v} L(y_i, f(\boldsymbol{x}_i))$$

  In some cases, it is preferable to use an alternative performance indicator given by the proportion of errors made by the classification algorithm:

$$err_A(\mathcal{V}) = err_{A_{\mathcal{F}}}(\mathcal{V}) = 1 - acc_{A_{\mathcal{F}}}(\mathcal{V}) = \frac{1}{v}\sum_{i=1}^{v} L(y_i, f(\boldsymbol{x}_i))$$

Prediction accuracy

- Loss function counting the number of misclassifications

$$L(y, f(\mathbf{x})) = \begin{cases} 0 & \text{if } y = f(\mathbf{x}) \\ 1 & \text{if } y \neq f(\mathbf{x}) \end{cases}$$

- Empirical error

$$R_{\text{emp}} = \frac{1}{m}\sum_{i=1}^{m} L(f(\mathbf{x}_i))$$

- **Speed**: how fast is the training of your model by your algorithm. Is an indicator of the computation time. Some methods require shorter computation times than others and can handle larger problems. However, classification methods characterized by longer computation times may be applied to a small-size training set obtained from a large number of observations by means of random sampling schemes. It is not uncommon to obtain more accurate classification rules in this way.

- **Robustness:** A classification method is robust if the classification rules generated, as well as the corresponding accuracy, do not vary significantly as the choice of the training set and the test set varies, and if it is able to handle missing data and outliers. Something is robust when the small changes in data, do not change the solution (Small change in input→ small change in the solution). If the rules and the accuracy don't vary, changing train/test
- **Scalability:** The scalability of a classifier refers to its ability to learn from large datasets (related to speed), and it is inevitably related to its computation speed. Therefore, the remarks made in connection withsampling techniques for data reduction, which often result in rules having better generalization capability, also apply in this case.
- **Interpretability:** If the aim of a classification analysis is to interpret as well as predict, then the rules generated should be simple and easily understood by knowledge workers and experts in the application domain.

How do we choose the train and the test set? We have 2 alternatives, one is good for large dataset and one is good for small dataset.

Prediction accuracy can be estimated through four main approaches:

2.1. Holdout method:

The holdout estimation method involves subdividing the m observations available into two disjoint subsets T and $\mathcal{V}$, for training and testing purposes respectively, and then evaluating the accuracy of the model through the accuracy $acc_A(\mathcal{V})$ on the test set. In general, T is obtained through a simple sampling procedure, which randomly extracts t observations from D, leaving the remaining examples for the test set $\mathcal{V}$. The portion of data used for training may vary based on the size of the dataset D. Usually, the value t falls somewhere between one half and two thirds of the total number m of observations. The accuracy of a classification algorithm evaluated via the holdout method depends on the test set $\mathcal{V}$ selected, and therefore it may over- or under estimate the actual accuracy of the classifier as $\mathcal{V}$ varies.

If we have **large dataset** we have a good approximation of the true accuracy of your model.
For small dataset is better to use alternative methods because in this case we have large fluctuations of the accuracy when we repeat this process.

2.2. Repeated random sampling method:

The repeated random sampling method involves replicating the holdout method a number r of times until I reach a good approximation. But this method is not convenient, so we use other methods. For each repetition, a random independent sample T_k is extracted, which includes t observations, and the corresponding accuracy $acc_A(\mathcal{V}_k)$ is evaluated, where $\mathcal{V}_k = D - \mathrm{T}_k$. At the end of the procedure, the accuracy of the classifier $A_{\mathcal{F}}$ is estimated using the sample mean:

$$acc_A = acc_{A_{\mathcal{F}}} = \frac{1}{r}\sum_{k=1}^{r} acc_{A_{\mathcal{F}}}(\mathcal{V}_k)$$

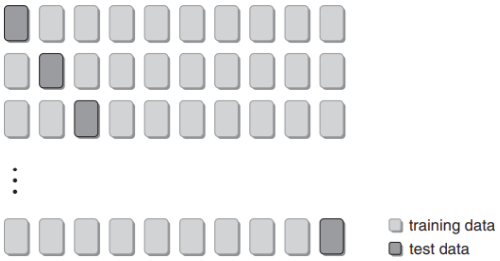
The total number r of repetitions may be evaluated a priori using techniques from statistical inference for determining the appropriate sample size. The method of repeated random sampling is preferred over the holdout method since it yields a more reliable estimate. However, there is no control over the number of times that each observation may appear in the training or in the test set. In this way, a dominant observation containing outliers for one or more attributes may cause undesired effects on both the classification rules generated and the resulting accuracy estimate.

2.3. Cross-validation:

The method of cross-validation offers an alternative to repeated random sampling techniques and guarantees that each observation of the dataset D appears the same number of times in the training sets and exactly once in the test sets. The cross-validation scheme is based on a partition of the dataset D into r disjoint subsets $L_1, L_2, \ldots, L_r$, and requires r iterations. At the kth iteration of the procedure, subset L_k is selected as the test set and the union of all the other subsets in the partition as the training set, that is:

$$\mathcal{V}_k = \mathcal{L}_k, \qquad \mathcal{T}_k = \bigcup_{l \neq k} \mathcal{L}_l$$

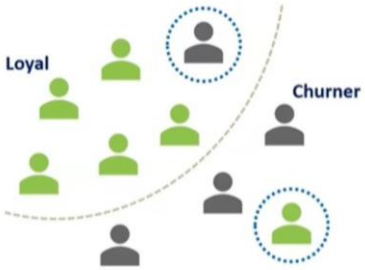
Once the partition has been generated, by randomly extracting the observations for each set f_k, the classification algorithm $A_{\mathcal{F}}$ is applied r times, using each of the r training sets in turn and evaluating the accuracy each time on the corresponding test set. At the end of the procedure, the overall accuracy is computed as the arithmetic mean of the r individual accuracies. If $r = 2$, each observation is used precisely once for training and once for evaluating the accuracy on the test set. However, higher values of r are usually preferred in order to obtain a more robust estimate of the accuracy. A popular choice in practice is tenfold cross-validation, where by the dataset D is partitioned into 10 subsets. The figure below illustrates the tenfold cross-validation scheme.



2.4. Leave-one-out:

It is a different evaluation method, obtained by choosing $r = m$. In this case each of the m test sets include only one observation, and each example is used in turn to measure the accuracy. By virtue of a greater computational effort, due to m executions of the classifier, the leave-one-out method affords the advantage of carrying out the training operation with a larger number of observations, equal to $m - 1$ for each iteration. It is usually required that each subset f_k in the partition contains the same proportion of observations belonging to each target class. I*f for examplethe binary target class indicates by the value {1} that a customer has churned, it is appropriate thateach subset* f_k *contain the same proportion of observations having the target class value {1} and that this proportion be the same as for the full dataset* D. In order to preserve in the partition, the proportions originally contained in D, stratified random sampling is used to generate the subsets f_k.

REMARK: As we see in this slide, accuracy cannot be the only parameter to use to decide which model is best. In fact, on the left we see that even if the accuracy is better, I have a classification between greens and grays; on the right I have a better accuracy but the classifier will always give me 1 as a result, there is never a classification (separation between greens and grays).



Accuracy = % accurate predictions= 80%
Error = 100 – Accuracy = 20%

Accuracy = 90%
Error = 10%

2.5 Confusion matrix:

The accuracy measurement methods described above are not always adequate for discriminating among models, and in some instances, they may even yield paradoxical results. It is not enough to consider the number of accurate predictions, but also the type of error committed should be accounted for. For this purpose, it is useful to resort to decision tables, usually called *confusion matrices*, which for the sake of simplicity we will only describe in connection with binary classification, though they can be easily extended to multicategory classification.

Let us assume that we wish to analyze a binary classification problem where the values taken by the target class are {−1, 1}. We can then consider a 2 × 2 matrix whose rows correspond to the observed values and whose columns are associated with the values predicted using a classification model, as shown in the table below:

		predictions		
		−1 (negative)	+1 (positive)	total
examples	−1 (negative)	p	q	$p+q$
	+1 (positive)	u	v	$u+v$
	total	$p+u$	$q+v$	m

The elements of the confusion matrix have the following meanings: p is the number of correct predictions for the negative examples, called true negatives; u is the number of incorrect predictions for the positive examples, called false negatives; q is the number of incorrect predictions for the negative examples, called false positives; and v is the number of correct predictions for the positive examples, called true positives. Using these elements, further indicators useful for validating a classification algorithm can be defined.

Accuracy = % records correctly classified= (p + v)/m

Recall (true positive rate tp) = % true positives correctly classified = v /(u + v)

Precision (prc) = % true positives among those predicted positives = v /(q + v)

False positive rate = q/(p+q)

True negative rate = p/(p+q)

False negative rate = u/(u+v)

- Geometric mean: it is defined as:

$$gm = \sqrt{tp \times pcr}$$

and sometimes also as:

$$gm = \sqrt{tp \times tn}$$

In both cases, the geometric mean is equal to 0 if all the predictions are incorrect.

- F-measure:

$$F = \frac{(\beta^2 - 1)tp \times prc}{\beta^2 prc + tp}$$

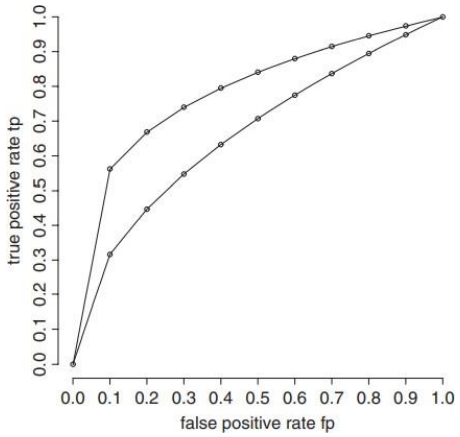
Where $\beta \in [0, \infty]$ regulates the relative importance of the precision with respect to thetrue positive rate. The F-measure is also equal to 0 if all the predictions are incorrect.

With some classifiers it is possible to assign a matrix of costs for incorrectly classified examples, whichwe will refer to as *misclassification costs*. Obviously, the cost associated with the $p + v$ correctly classified observations is zero. By adequately regulating the relative weight of the two costs of misclassification corresponding to the u false negatives and to the q false positives, it is possible to direct a classifier toward the actual objectives of the decision makers.

2.6. ROC curve charts:

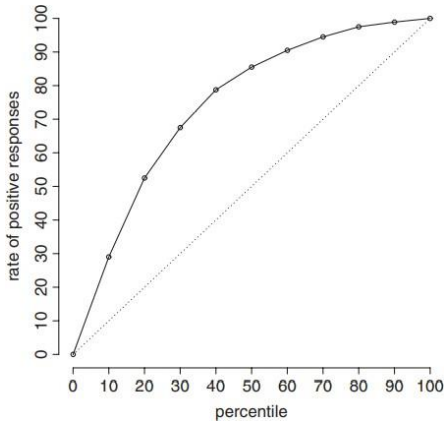
Receiver operating characteristic **(ROC)** curve charts allow the user to visually evaluate the accuracy of a classifier and to compare different classification models. They visually express the information content of a sequence of confusion matrices and allow the ideal trade-off between the number of correctly classified positive observations and the number of incorrectly classified negative observations to be assessed. In this respect, they are an alternative to the assignment of misclassification costs.

A ROC chart is a two-dimensional plot with the proportion of false positives fp on the horizontal axis and the proportion of true positives tp on the vertical axis. The point (0,1) represents the ideal classifier, which makes no prediction error since its proportion of false positives is null $(fp = 0)$ and its proportion of true positives is maximum $(tp = 1)$. The point (0,0) corresponds to a classifierthat predicts the class {−1} for all the observations, while the point (1,1) corresponds to a classifier predicting the class {1} for all the observations. Most classifiers allow a few parameters in the optimal hypothesis $f * \in \mathcal{F}$ to be adjusted so as to increase the number of true positives tp, albeit at the expense of a corresponding increase in the number of false positives fp. To obtain the trajectory of the ROC curve for a specific classifier, it is therefore necessary to use pairs of values (fp, tp) which have been empirically obtained for different values of the parameters in $f * \in \mathcal{F}$. A classifier with no parameters to be tuned yields only one point in the diagram. The figure below shows an example of ROC curves chart. The area beneath the ROC curve gives a concise measurement comparing the accuracy of various classifiers: the classifier associated with the ROC curve with the greatest area (AUC= area under the curve) is then preferable.



2.7. Cumulative gain charts and lift charts:

The cumulative gains chart shows the percentage of the overall number of cases in a given category "gained" by targeting a percentage of the total number of cases. The cumulative gains curve is an evaluation curve that assesses the performance of the model and compares the results with the random pick. It shows the percentage of targets reached when considering a certain percentage of the population with the highest probability to be target according to the model. The diagonal line is the "baseline" curve. On the left-hand side of the horizontal axis, we place the observations with the highest probability to be target according to the model and vice versa for the right-hand side. On the vertical axis, the curve indicates which **percentage of all targets is included in this curve.**



Gain or lift is a measure of the effectiveness of a classification model calculated as the ratio between the results obtained with and without the model.

The aim of gain and lift chart is to evaluate performance of classification models; both charts consist of a lift curve and a baseline. However, in contrast to the confusion matrix that evaluates models on the whole population gain or lift chart evaluates model performance ina portion of the population. The greater the area (Maggiore è l'area) between the lift curve anche the baseline, the better (migliore è) the model.

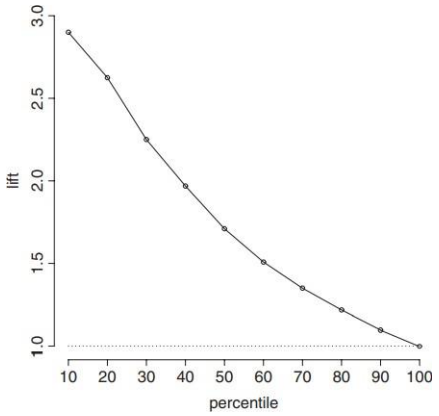
Da questo grafico ottengo quello della lift. In particolare, per ogni decile vado a dividere il valore del nostro modello (curva nera) per il decile stesso. Otterrò una curva descrescente che e il lift ????

The *lift* measure corresponds to the intuitive idea of evaluating the accuracy of a classifier based on the density of positive observations inside the set that has been identified based on model

predictions. To further clarify this point, let S be a subset of observations of cardinality s selected according to the classifier. The lift is defined as the ratio between the proportion b/s of positive observations existing in S and the proportion a/m of positive observations in the whole dataset D, that is:

$$lift = \frac{b/s}{a/m}$$

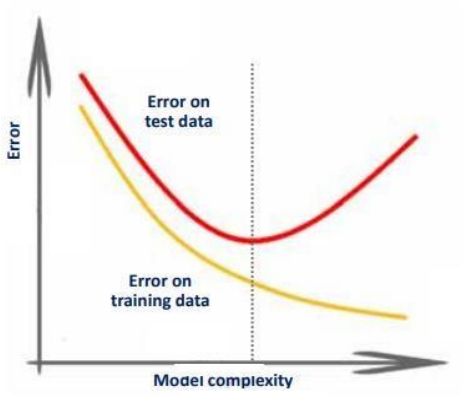
where b and a are the number of positive observations in the sets S and D, respectively.
Cumulative gain and lift diagrams allow the user to visually evaluate the effectiveness of a classifier.
The lift chart is derived from the cumulative gains chart; the values on the y axis correspond to the ratio of the cumulative gain for each curve to the baseline.
Constructing a lift curve follows a similar process as forming the cumulative gain curve. Indeed, it is derived from the gain chart. First, we order observations on the horizontal axis with the highest probability of being a target on the left and vice versa for the right-hand side. On the vertical axis, the lift curve indicates how many times more than average targets are included in this group. Lift is calculated as the ratio of Cumulative Gains from classification and random models.



Two criteria based on cumulative gain and lift curves allow different classifiers to be compared.
Onthe one hand, the area between the cumulative gain curve and the line corresponding to random sampling is evaluated: a greater area corresponds to a classification method that is more effective overall. However, it may be the case that a method producing a greater area is associated with a lower lift value than a different method at a specific value s on the horizontal axis that indicates the actual number of recipients of the marketing campaign, determined according to the available budget.
The second criterion is therefore based on the maximum lift value at a specific value s on the horizontal axis.

2.8. Overfitting:

It happens while trying to reduce too much the error on the training set by using a hypothesis space which is very large.
We can use the following representation to explain it better:



On the x axis there is the model complexity, which could be measure for instance as the degree of the polynomial in the hypothesis space, or for example in SVM could be represented by the VC complexity, or in neural network it could be represented by the number of nodes and layers. On the y axis there is Error. The behavior of the error on the training data is the yellow line: the more complex is the model, the better will be the capability to minimize the error in the training data. The red line represents the error on the test data: we can see that the error decreases up to a given point and then it starts to go up, because from that point ahead there is overfitting.

When we train a model and we test it, the different way we have introduced to measure the accuracy, should be more or less the same between training and test set. if the accuracy is higher on the training set, then on the test set, we will run into overfitting.

3.9. Transformation of explanatory categorical variables into numerical representation:

Suppose we have a categorical variable X_j, which assumes values in the set $V = \{v_1, v_2, \dots, v_H\}$. To be more specific we take for example the months associated to an observation (dummies $D_{j,1}, D_{j,2}, \dots, D_{j,H-1}$), we introduce an indicator for each different months, in general for each value assumed to our variable:

$$\{D_{gen}, D_{feb}, D_{mar}, D_{apr}, D_{mag}, D_{giu}, D_{lug}, D_{ago}, D_{sett}, D_{ott}, D_{nov}\}$$

I want to provide the numerical representation:

categorical	Gen	Feb	Mar	Apr	Mag	Giu	Lug	Ago	Sett	Ott	Nov
January	1	0	0	0	0	0	0	0	0	0	0
September	0	0	0	0	0	0	0	0	1	0	0
March	0	0	1	0	0	0	0	0	0	0	0
…etc											

We can simply replace each categorical variable with the corresponding binary numerical variables. We did not include the dummy $D_{j,H}$ (in our example D_{dic}) because it would have been redundant. The last one is clearly dependent from the previous ones. If I know that there is a 1 before the last column, the last column will surely be 0. If there are not 1 before the last column, it will surely contain 1. The last replacement is automatic.

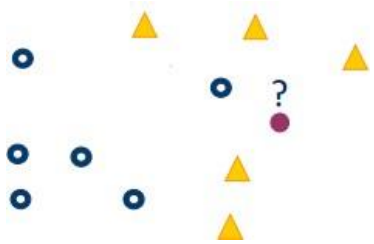
EVALUATION METHODS:

Before introducing the models, we recap the most significant evaluation methods of the performances:

- AUC = Area Under the Curve (ROC Curve).
- CA = classification accuracy.
- F1 = F-measure for $\beta = 1$. It corresponds to the harmonic mean between Precision and Recall
- Precision
- Recall

3. K-NEAREST NEIGHBOR (KNN):

It is the simplest family of classification algorithms. The entire training set is used to performing already prediction.



To a new record is attributed the class to which belong the majority of records out of the K closest ones (majority voting). Odd (dispari) values of K are suggested, because we want to apply *majority voting* and majority voting is good when there is **no balance between the two classes**. If there is a new observation, we calculate the distance between this and the others, and based on those distances we choose the nearest neighbor and we could associated the new observation to the correct class. But to improve robustness of this, I establish a score, which is the number of the majority class nearest neighbor out of (tra) the numbers of the possible nearest neighbors (for example:the purple dot is the new observation. Its nearest neighbors are the yellow triangles because they are shown to be the majority class, but there is also a blue dot near it, so, if K=5, the score will be equal to 5yellow triangles out of (5 yellow triangles + one blue dot), and so 5/6). (SE LA NUOVA PALLINA FOSSE IN ALTO VICINO A QUELLA BLU SINGOLARMENTE, QUELLA BLU è UN OUTLIER, VADO A CONSIDERARE LA MAGGIORANZA DI PALLINE A CUI è VICINA DI CHE COLORE SONO)

The choice of K has consequences on behavior of the algorithm, in terms of robustness and accuracy. Precisely, if we increase K we obtain less sensitivity to outliers and on less distinction among classes and so we risk to lose accuracy.

K is the hyperparameter of this class of algorithms.

We define hyperparameters, those parameters that characterize the behavior of the algorithm and are available to the user choice. How do we select K? there is not a theorem or a criterium that drives us in the choice of key. We are dealing with a really empirical discipline. The best way to choose hyperparameters is by experiments. we have to perform some experiments, collect the metrics, and then evaluate which one is the best.

In KNN there is another hyperparameter which is hidden: distance. The type of distance is another parameter that user is available to choose.

4. CLASSIFICATION TREES:

Is a base **recursive** greedy algorithm:

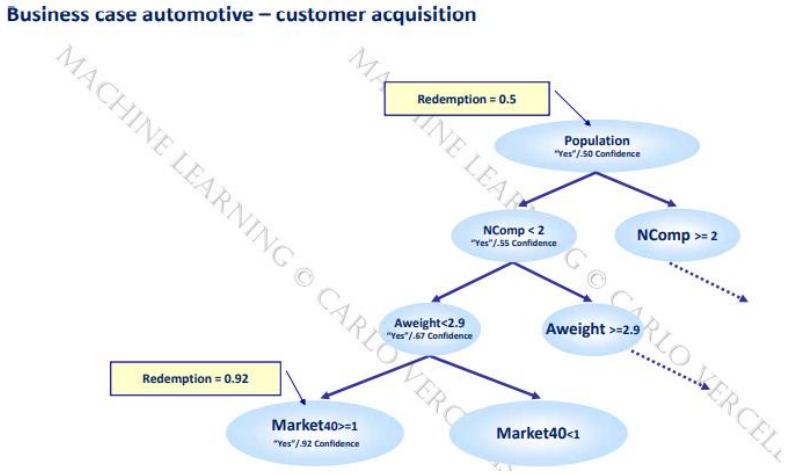
- All records are placed in the root node
- Records in each node are split according to a heuristic test based on the value of one or more attributes

Ex. Marketing problem, an automotive industry want to evaluate the best target for their marketing campaigns based on the results of past different campaigns; I start with a **balance situation** of dataset (suppose for simplicity 50% positive, purchase (acquistare) the product , 50% negative didn't make a purchase). We start with "population", a **root node** in which we have all the observation of the balance dataset (training set), but we have a confusion situation (no distinction between positive and negative class). We have to introduce new indexes in order to reduce entropy and other indexes, in particular we could make different distinction based on a thresholds values.

This approach is **not slow** because I have linearity in the number of attribute (I have to try just *n* attribute) and I have only some values of threshold (every attribute could assume some specific values). We split the dataset and every iteration I use a threshold value for different attributes: some values go to the left, other to the right for each iteration. We stop to split values when I have a good level of confidence (in this example 0.92).

This classification algorithms is a **recursive algorithm.** Algoritmo ricorsivo: la funzione lancia sé stessa fino a quando la condizione non è soddisfatta a differenza di quelli iterativi (struttura ciclica) dove una funzione si ripete fino a quando la condizione diventa falsa.

The goal is to contain the number of ramification because if I have a lot of ramification, I could have **overfitting (this is the risk).**



Classification trees are perhaps the best-known and most widely used learning methods in data mining applications. The reasons for their popularity lie in their conceptual simplicity, ease of usage, computational speed, robustness with respect to missing data and outliers and, most of all, the interpretability of the rules they generate. To separate the observations belonging to different classes, methods based on trees obtain simple and explanatory rules for the relationship existing between the target variable and predictive variables. The development of a classification tree corresponds to the training phase of the model and is regulated by a recursive procedure of heuristic nature, based on a divide-and-conquer partitioning scheme referred to as top-down induction of decision trees, described by the Procedure below. Some of the mechanisms governing the development of a tree may be implemented by following different approaches, so that classification trees actually represent a broad (ampia) class of methods that we will first describe in general terms and then illustrate in detail in some specific cases.

Procedure 10.1 – Top-down induction of decision trees

1. In the initialization phase, each observation is placed in the root node of the tree. The root is included in the list L of active nodes.
2. If the list L is empty the procedure is stopped, otherwise a node J belonging to the list L is selected, is removed from the list and is used as the node for analysis.
3. The optimal rule to split the observations contained in J is then determined, based on an appropriate preset criterion. The splitting rule generated in this way is then applied, and descendant nodes are constructed by subdividing the observations contained in J. For each descendant node the conditions for stopping the subdivision are verified. If these are met, node J becomes a leaf, to which the target class is assigned according to the majority of the observations contained in J. Otherwise, the descendant nodes are added to the list L. Finally, step 2 is repeated.

The observations of the training set initially contained in the ***root node*** of the tree are divided into disjoint subsets that are tentatively placed in two or more descendant nodes (*branching*). At each of the nodes determined in this way, a check is applied to verify if the conditions for stopping the development of the node are satisfied. If at least one of these conditions is met, no furthersubdivision is performed, and the node becomes a *leaf* of the tree. Otherwise, the actual subdivisionof the observations contained in the node is carried out. At the end of the procedure, when no tree node can be further subdivided, each **leaf node** is labeled (etichettato) with the value of the class to which the majority of the observations in the node belong, according to a criterion called *majority voting*. The subdivision of the examples in each node is carried out by means of a ***splitting rule***, also termed a *separating rule*, to be selected based upon a specific evaluation function. By varying the metrics used to identify the splitting rule, different versions of classification trees can be obtained. Most of the proposed evaluation criteria share the objective of maximizing the uniformity of the target class for the observations that are placed in each node generated through the separation. The different splitting rules are usually based on the value of some explanatory variables that describe the observations.

At the end of the procedure, the set of splitting rules that can be found along the path connecting the tree root to a leaf node constitutes a *classification rule*. During the prediction phase, in order to assign the target class to a new observation, a path is followed from the root node to a leaf node by obeying the sequence of rules applied to the values of the attributes of the new observation. The predicted target class then coincides with the class by which the leaf node so reached has been labeled during the development phase, that is, with the class of the majority of the observations in the training set falling in that leaf node.

Most classifiers associate with each observation a score function, which is then converted into a prediction of the target class. This is also true for classification trees, which associate each observation contained in a leaf node with the highest proportion of the target class for the observations contained in the leaf itself, which also determines its labeling by majority voting.

Starting from a training dataset it is possible to construct an exponential number of distinct classification trees. It can be shown that the problem of determining the optimal tree is NP-hard, that is, computationally difficult. As a consequence, the methods for developing classification trees are heuristic in nature. As observed, the scheme for generating classification trees described in Procedure 10.1 is a general framework and requires that some steps be specified before deriving an

implementable classification algorithm.

- Splitting rules: For each node of the tree, it is necessary to specify the criteria used to identify the optimal rule for splitting the observations and for creating the descendant nodes. There are several alternative criteria, which differ in the number of descendants, the number of attributes and the evaluation metrics.
- Stopping criteria: At each node of the tree different stopping criteria are applied to establish whether the development should be continued recursively, or the node should be considered as a leaf. In this case too, various criteria have been proposed, which result in quite different topologies of the generated trees, all other elements being equal.
- Pruning criteria: Finally, it is appropriate to apply a few pruning criteria, first to avoid excessive growth of the tree during the development phase (pre-pruning), and then to reduce the number of nodes after the tree has been generated (post-pruning).

To prevent overfitting I could make:
- pre pruning by stopping criteria
  purity threshold
  minimum cardinality
  minimum information gain
- post pruning: identifies and removes ramifications affected by noise

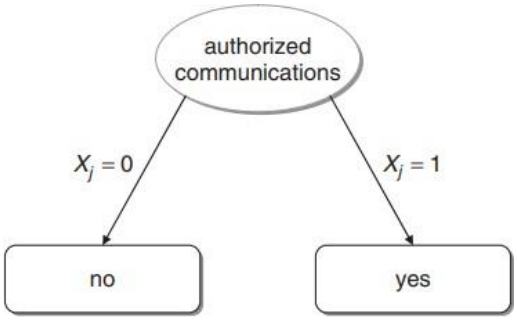
4.1. Splitting rules:

Classification trees can be divided into binary and general trees based on the maximum number of descendants that each node is allowed to generate.

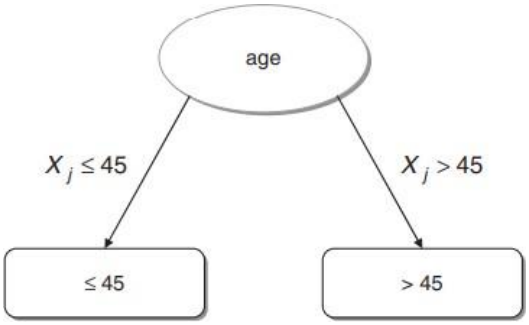
- Binary trees: A tree is said to be binary if each node has at most two branches. Binary trees represent in a natural way the subdivision of the observations contained at a node based on the value of a binary explanatory attribute. When dealing with categorical attributes with more than two classes, binary trees should necessarily form two groups of categories in order to perform a split. Numerical attributes can be separated based on a threshold value. Finally, binary trees may also be used to develop multicategory classification.
- Multi-split classification trees (general trees): A tree is said to be multi-split if each node has an arbitrary number of branches. This allows multi-valued categorical explanatory attributes to be handled more easily. On the other hand, with numerical attributes, it is again necessary to group together adjacent values. The latter operation is basically equivalent to discretization, obtained in a dynamic way by the algorithm itself during the tree development phase. Based on the empirical evidence, no significant differences seem to emerge in the predictive accuracy of classification trees in connection with the maximum number of descendant nodes.

Another relevant distinction within classification tree methods concerns the way the explanatory attributes contribute to the definition of the splitting rule at each node. In particular, we may distinguish between univariate and multivariate trees.

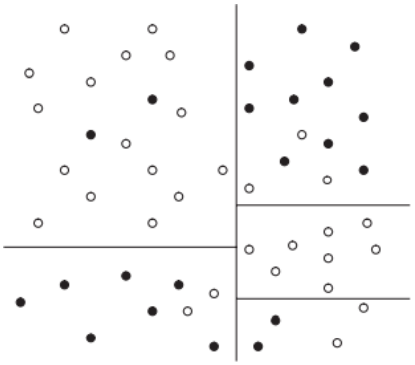
- Univariate trees: for univariate trees, the splitting rule is based on the value assumed by a single explanatory attribute X_j. If the selected attribute is categorical, the observations at a given node are divided by means of conditions of the form $X_j \in B_k$, where the collection $\{B_k\}$ is composed of disjoint and exhaustive subsets of the set of values assumed by the attribute X_j. For example, for a binary attribute taking the values {0, 1}, the two subsets B_0 and B_1 correspond to the values {0} and {1}, as shown in figure:



If X_j is a numerical attribute, the univariate partition consists of a rule in the form $X_j \leq b$ or $X_j \geq b$, as shown in figure:



Univariate trees are also referred to as axis-parallel trees, since the splitting rules induce a partition of the space of the observations into hyper-rectangles, determined by the intersection of halfspaces whose support hyperplanes are parallel to the components of the vector of instances, as shown in figure:

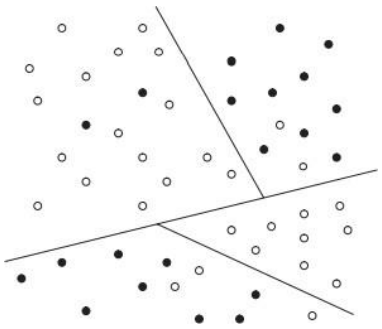


Classification by means of univariate splitting rules (axis-parallel). Each line corresponds to a splitting rule generated at a node in the development of the tree

- Multivariate trees: For multivariate trees, the partition of the observations at a given node is based on the value assumed by a function $\varphi(x_1, x_2, \dots, x_n)$ of the attributes and leads to a rule of the form $\varphi(\boldsymbol{x}) \leq b$ or $\varphi(\boldsymbol{x}) > b$. Different methods have been proposed whereby the function φ represents a linear combination of the explanatory variables. In this case, the expression that leads to the separation of the observations takes the form:

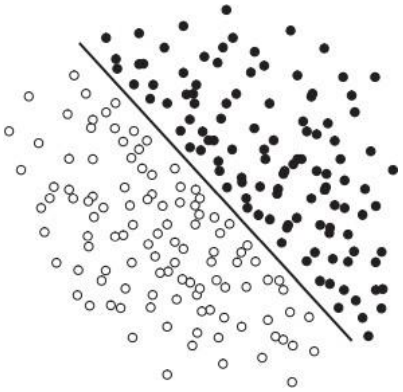
$$\sum_{j=1}^{n} w_j x_j \leq b$$

where the threshold value b and the coefficients $w_1, w_2, \dots, w_n$ of the linear combination have to be determined, for example by solving an optimization problem for each node, just like for classification trees generated by means of discrete variants of support vector machines. Multivariate trees are also referred to as oblique decision trees, as they generate polygonal partitions of the space of the observations by means of separating hyperplanes, as shown in figure:



Classification by means of multivariate splitting rules (oblique). Each straight line corresponds to a splitting rule generated at a node in the development of the tree

Oblique trees are usually characterized by greater predictive accuracy than univariate trees, against a lower interpretability of the classification rules generated. In many cases, a limited number of separating hyperplanes may be enough to classify with high accuracy the instances, whereas to achieve the same result an axis-parallel tree would require the partition of the space of the observations in several hyper-rectangles. The figure below shows an example of a two-dimensional dataset for which the two target classes may be easily separated by a single oblique line, while they would require a high number of univariate rules:



Classification by means of a single multivariate (oblique) splitting rule for a dataset that cannot be accurately split by univariate (axis-parallel) rules

Notice also that the number of regions generated is a function of the number of leaves in the tree, and therefore of its depth. Accuracy being equal, oblique trees usually generate a lower number of classification rules with respect to univariate trees.

4.1.1 Univariate splitting criteria:

Although they are usually characterized by a lower accuracy, the algorithms that develop classification trees based on univariate rules are more popular than their multivariate counterpart, partly because of the simplicity and interpretability of the rules generated and partly because they were proposed first.

The main component that differentiates the variants of univariate classification trees proposed so far is the splitting rule, used to identify the best explanatory attribute out of those available and to select the most effective partitioning criterion among the ones it induces. Usually, both choices are performed by calculating an evaluation function, for each attribute and for each possible partition, which provides a heterogeneity measure in the values of the target class between the examples belonging to the parent node and those belonging to the descendants. The maximization of the evaluation function therefore identifies the partition that generates descendant nodes that are more homogeneous within themselves than the parent node is.

Let p_h be the proportion of examples of target class v_h, $h \in \mathcal{H}$, at a given node q and let Q be the total number of instances at q. We have:

$$\sum_{h=1}^{H} p_h = 1$$

The heterogeneity index $I(q)$ of a node is usually a function of the relative frequencies p_h, $h \in \mathcal{H}$, of the target class values for the examples at the node, and it has to satisfy three requirements: it must take its maximum value when the examples at the node are distributed homogeneously among all the classes; it must take its minimum value when all the instances at the node belong to the same class; and it must be a symmetric function with respect to the relative frequencies p_h, $h \in \mathcal{H}$.

Among the heterogeneity indices of a node q that satisfy these properties, also referred to as impurity or inhomogeneity measures, the most popular are the *misclassification index*, the *entropy index,* and the *Gini index*.

The main criteria for splitting are:

- Misclassification index: The misclassification index is defined as

$$Miscl(q) = 1 - \max_h p_h$$

  and measures the proportion of misclassified examples when all the instances at node q are assigned to the class to which the majority of them belong, according to the majority voting principle.

- Entropy index: the entropy index is defined as

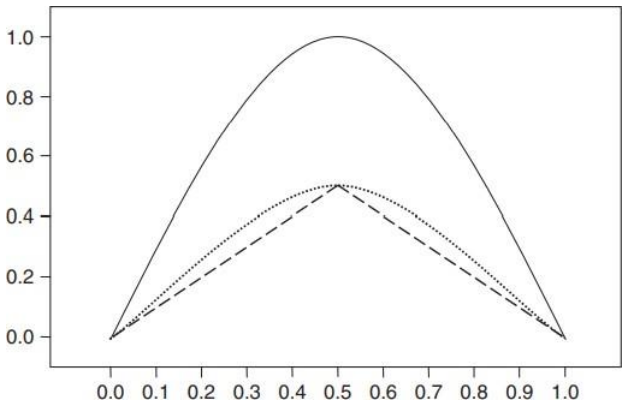
$$Entropy(q) = -\sum_{h=1}^{H} p_h \log_2 p_h$$

  Note that, by convention $0 \log_2 0 = 0$.

- Gini index: The Gini index is defined as

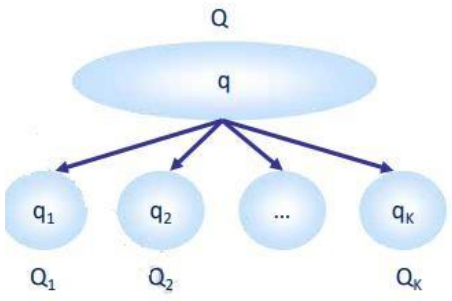
$$Gini(q) = 1 - \sum_{h=1}^{H} p_h^2$$

In a binary classification problem, the impurity measures defined above reach their maximum value when $p_1 = p_2 = 1 - p_1 = 0.5$, and are 0 when $p_1 = 0$ or $p_1 = 1$, as shown in figure:



Graph of the misclassification index (dashed line), the Gini index (dotted line) and the entropy (full line) for a binary target attribute as the frequency of the examples in one class varies

The univariate splitting criteria based on the information gain compare one of the impurity indices evaluated for the parent node with the same index computed for the set of descendant nodes, and then choose the attribute and the corresponding partition that maximize the difference. Since the value of the impurity index for the parent node is independent of the partition, the choice may be equivalently done by minimizing the overall impurity index for the descendant nodes.
Let $I(\cdot)$ be one of the impurity indices previously defined and suppose that a splitting rule separates the examples contained at node q into K descendant nodes $\{q_1, q_2, \ldots, q_K\}$, each containing Q_k instances. If we analyze the partition originated by a categorical attribute X_j taking H_j distinct values, the set of examples at q can be separated into H_j disjoint subsets, as shown in the figure below:



In this way we have $K = H_j$, and the descendant node q_j contains the examples for which the explanatory variable X_j takes the value v_j. If $H_j > 2$, this partition is possible only if the tree is of general type. If a binary tree is developed, it is necessary to divide the H_j values into two non-empty sets and then compute the heterogeneity indices for all $2H_j - 1$ possible partitions. Finally, if the attribute X_j is numerical, the examples can be subdivided by intervals of values. To avoid the analysis of all possible threshold values for the separation, the algorithm operates a binary search among the values actually taken by the attribute X_j in the dataset D. The impurity of the descendant nodes, and therefore **the impurity of the splitting** rule, is defined as

$$I(q_1, q_2, \ldots, q_K) = \sum_{k=1}^{K} \frac{Q_k}{Q} I(q_k)$$

Hence, the impurity of the partition is expressed by a weighted sum of the impurities of each descendant node, where each weight equals the percentage of examples from the parent node that are placed in the corresponding descendant.

The algorithms for the development of univariate trees select for each node the rule and the corresponding attribute that determine the minimum value of the expression above. This choice is equivalent **to maximizing the information gain Δ(·),** defined as:

$$\Delta(q, q_1, q_2, \ldots, q_K) = I(q) - I(q_1, q_2, \ldots, q_K)$$
$$= I(q) - \sum_{k=1}^{K} \frac{Q_k}{Q} I(q_k).$$

EXAMPLE:
We have 23 observation (10 positive and 13 negative). Compute I(q) using **entropy:**

area	numin	timein	numout	Pothers	Pmob	Pland	numsms	numserv	numcall	dropt	churner
2	1	1	2	1	4	1	3	2	2	0	1
1	1	3	3	2	4	1	4	2	3	0	0
3	2	1	2	2	4	1	3	2	1	0	0
1	2	3	2	3	4	1	1	2	1	0	0
2	3	4	4	4	1	1	3	2	1	0	0
3	3	4	1	4	2	1	4	3	1	0	0
3	3	3	4	4	3	1	4	3	1	1	0
1	1	1	1	1	3	2	1	1	1	0	1
2	2	2	2	1	3	2	2	3	1	1	1
4	2	1	3	2	3	2	1	2	1	1	1
3	1	1	2	2	2	2	2	2	1	0	1
4	3	4	4	2	1	2	2	4	1	1	1
2	1	1	3	2	3	2	2	4	1	1	0
4	2	1	2	3	2	2	2	2	1	0	0
3	3	4	4	3	2	2	2	4	1	1	0
1	1	2	1	4	2	2	2	2	1	0	0
4	1	1	2	4	2	2	4	2	1	0	1
1	1	1	1	1	1	3	1	1	1	0	0
3	1	1	1	1	1	3	1	1	1	0	1
2	3	4	3	1	1	3	1	1	1	0	1
1	3	3	3	1	2	3	4	2	1	0	0
4	2	2	2	2	2	3	1	1	1	0	1
3	3	2	1	4	1	3	1	1	1	0	0

$$I_E(q) = \text{Entropy}(q) = -\frac{13}{23}\log_2\frac{13}{23} - \frac{10}{23}\log_2\frac{10}{23} = 0.988$$

$$p_0(q_1) = 5/6, \quad p_0(q_2) = 2/5, \quad p_0(q_3) = 5/7, \quad p_0(q_4) = 1/5,$$
$$p_1(q_1) = 1/6, \quad p_1(q_2) = 3/5, \quad p_1(q_3) = 2/7, \quad p_1(q_4) = 4/5.$$

Entropy is close to 1 because

$$I_E(q_1, q_2, q_3, q_4) = \frac{6}{23} I_E(q_1) + \frac{5}{23} I_E(q_2) + \frac{7}{23} I_E(q_3) + \frac{5}{23} I_E(q_4)$$
$$= \frac{6}{23}0.650 + \frac{5}{23}0.971 + \frac{7}{23}0.863 + \frac{5}{23}0.722 = 0.8.$$

the two probability are similar (10/23 e 13/23).

We have q1, q2,q3,q4 that correspond at the number 1,2,3,4 in the column area. The denominator of p0(q1) is the number of q1 in the column area, the numerator is the number of time that we have q1 equal to zero in the last column, and so on.

Ie(q1) = -5/6 log2(5/6) -1/6 log2(1/6)

$$\Delta_E(area) = \Delta_E(q, q_1, q_2, q_3, q_4) = I_E(q) - I_E(q_1, q_2, q_3, q_4)$$
$$= 0.988 - 0.8 = 0.188.$$

$$\Delta_E(numin) = 0.057, \quad \Delta_E(Pland) = 0.125,$$
$$\Delta_E(timein) = 0.181, \quad \Delta_E(numsms) = 0.080,$$
$$\Delta_E(numout) = 0.065, \quad \Delta_E(numserv) = 0.057,$$
$$\Delta_E(Pothers) = 0.256, \quad \Delta_E(numcall) = 0.089,$$
$$\Delta_E(Pmob) = 0.043, \quad \Delta_E(dropt) = 0.005.$$

Alternative: compute I(q) using gini

$$I_G(q) = \text{Gini}(q) = 1 - \left(\frac{13}{23}\right)^2 - \left(\frac{10}{23}\right)^2 = 0.491$$

$$I_G(q_1, q_2, q_3, q_4) - \frac{6}{23} I_G(q_1) + \frac{5}{23} I_G(q_2) + \frac{7}{23} I_G(q_3) + \frac{5}{23} I_G(q_4)$$
$$= \frac{6}{23} 0.278 + \frac{5}{23} 0.638 + \frac{7}{23} 0.194 + \frac{5}{23} 0.528 = 0.370.$$

$$\Delta_G(area) = \Delta_G(q, q_1, q_2, q_3, q_4) = I_G(q) - I_G(q_1, q_2, q_3, q_4)$$
$$= 0.491 - 0.370 = 0.121.$$

$$\Delta_G(numin) = 0.037, \quad \Delta_G(Pland) = 0.078,$$
$$\Delta_G(timein) = 0.091, \quad \Delta_G(numsms) = 0.052,$$
$$\Delta_G(numout) = 0.043, \quad \Delta_G(numserv) = 0.038,$$
$$\Delta_G(Pothers) = 0.146, \quad \Delta_G(numcall) = 0.044,$$
$$\Delta_G(Pmob) = 0.028, \quad \Delta_G(diropt) = 0.003.$$

4.2. Stopping criteria and pruning rules:

Is necessary to establish whether the development should be continued or if the node should be considered as a leaf.

Stopping criteria are a set of rules used at each node during the development of a tree in order to determine whether it is appropriate to create more branches and generate descendant nodes or whether the current node should become a leaf. There are two main reasons to limit the growth of a classification tree. First, a tree with too many ramifications usually achieves better accuracy with the training set but causes larger errors when used to make predictions on the test set or on future data. From an intuitive point of view, one may think that a tree with many branches excessively reflects the peculiarity of the examples in the training set and is therefore less capable of generalization. This phenomenon, usually referred to as overfitting, can be well explained from a theoretical perspective within the framework of statistical learning theory: a tree with too many ramifications actually represents a broader space of hypotheses, and therefore reduces the empirical error for the training set, but at the same time increases the generalization error.

Furthermore, a more ramified tree implies a proliferation of leaves and hence generates deep classification rules, obtained by combining a large number of splitting rules along the path leading from the root node to a leaf. This reduces the overall interpretability of the resulting classification model.

In principle, node splitting might be stopped when there is only one observation in the node. The stopping criteria followed in practice prevent this situation from happening, by introducing some restrictions on the minimum number of observations that a node must contain to be partitioned. Additionally, it can be assigned a minimum threshold on the uniformity, sometimes called *purity*, of the target class proportion within a node to be split.

More precisely, a node becomes a leaf of the tree when at least one of the following conditions occurs:

- Node size: The node contains a number of observations that is below a preset minimum threshold value.
- Purity: The proportion of observations at the node and belonging to the same class is above a preset maximum threshold value that corresponds to the accuracy that one wishes to achieve.
- Improvement: The possible subdivision of the node would generate a gain(·) that is below a preset minimum threshold value.

The stopping criteria described above are also referred to as *pre-pruning* rules because their aim is

to prune a priori the tree by limiting its growth. However, there are other *post-pruning*, or simply *pruning*, techniques which are applied upon completion of tree development to reduce the number of ramifications without worsening, and hopefully even improving, the predictive accuracy of the resulting model. At each iteration during the post-pruning phase, the possible advantage deriving from the removal of a given branch is evaluated, by comparing the predictive accuracy of the original tree with that of the reduced tree in classifying the observations of the tuning set. At the end of the evaluation, the reduced tree associated with the minimum prediction error is selected.

5. BAYESAN METHODS:

Bayesian methods belong to the family of probabilistic classification models. They explicitly calculate the posterior probability $P(y|\boldsymbol{x})$ that a given observation belongs to a specific target class by means of Bayes' theorem, once the prior probability $P(y)$ and the class conditional probabilities $P(\boldsymbol{x}|y)$ are known.

Unlike other methods described in this chapter, which are not based on probabilistic assumptions, Bayesian classifiers require the user to estimate the probability $P(\boldsymbol{x}|y)$ that a given observation may occur, provided it belongs to a specific class. The learning phase of a Bayesian classifier may therefore be identified with a preliminary analysis of the observations in the training set, to derive an estimate of the probability values required to perform the classification task.

Let us consider a generic observation $\boldsymbol{x}$ of the training set, whose target variable y may take H distinct values denoted as $\mathcal{H} = \{v_1, v_2, \dots, v_H\}$.

Bayes' theorem is used to calculate the posterior probability $P(y|\boldsymbol{x})$, that is, the probability of observing the target class y given the example (specific vector) $\boldsymbol{x}$:

$$P(y|\boldsymbol{x}) = \frac{P(\boldsymbol{x}|y)P(y)}{\sum_{l=1}^{H} P(\boldsymbol{x}|y)P(y)} = \frac{P(\boldsymbol{x}|y)P(y)}{P(\boldsymbol{x})}$$

In order to classify a new instance $\boldsymbol{x}$, the Bayes classifier applies a principle known as the *maximum a posteriori hypothesis* (MAP), which involves calculating the posterior probability $P(y|\boldsymbol{x})$ using the expression above and assigning the example $\boldsymbol{x}$ to the class that yields the maximum value $P(y|\boldsymbol{x})$, that is:

$$y_{MAP} = arg\max_{y \in \mathcal{H}} P(y|\boldsymbol{x}) = arg\max_{y \in \mathcal{H}} \frac{P(\boldsymbol{x}|y)P(y)}{P(\boldsymbol{x})}$$

Since the denominator $P(\boldsymbol{x})$ is independent of y, in order to maximize the posterior probability, it is enough to maximize the numerator of the expression above. Therefore, the observation $\boldsymbol{x}$ is assigned to the class v_h if and only if:

$$P(\boldsymbol{x}|y = v_h)P(y = v_h) \geq P(\boldsymbol{x}|y = v_l)P(y = v_l), \qquad l = 1, 2, \dots, H, l \neq h$$

The prior probability $P(y)$ can be estimated using the frequencies mh with which each value of the target class v_h appears in the dataset D, that is:

$$P(y = v_h) = \frac{m_h}{m}$$

Given a sufficiently large sample the estimates of the prior probabilities obtained through the expression above will be quite accurate. The difficulty of this method is that x is not a single value, but is a vector with many components.

Unfortunately, an analogous sample estimate of the class conditional probabilities $P(\boldsymbol{x}|y)$ can be hardly done in practice due to the computational complexity and the huge number of sample observationsthat it would require.

To overcome the computational difficulty described, it is possible to introduce two simplifying hypotheses that lead to *naive Bayesian classifiers* and to *Bayesian networks* respectively, described next.

5.1. Nayve Bayesian classifier:

Naive Bayesian classifiers are based on the assumption that the explanatory variables are conditionally independent given the target class. This hypothesis allows us to express the probability

$P(\boldsymbol{x}|y)$ as:

$$P(\mathbf{x}|y) = P(x_1|y) \times P(x_2|y) \times \cdots \times P(x_n|y) = \prod_{j=1}^{n} P(x_j|y)$$

The probabilities $P(x_j|y)$, $j \in$ N, can be estimated using the examples from the training set, depending on the nature of the attribute considered.

Two possible cases:

Categorical or discrete numerical attributes: for a categorical or discrete numerical attribute $\boldsymbol{a}_j$ which may take the values $\{r_{j1}, r_{j2,\ldots,}r_{jK_j}\}$, the probability $P(x_j|y) = P(x_j = r_{jk}|y = v_h)$ is evaluated as the ration between the number s_{jhk} of instances of class v_h or which the attribute $\boldsymbol{a}_j$ takes the value r_{jk}, and the total number m_h of instances of class v_h in the dataset D, that is:

$$P(x_j|y) = P(x_j = r_{jk}|y = v_h) = \frac{s_{jhk}}{m_h}$$

Numerical attributes: For a numerical attribute $\boldsymbol{a}_j$, the probability $P(x_j|y)$ is estimated assuming that the examples follow a given distribution. For example, one may consider a Gaussian density function, for which:

$$P(x_j|y = v_h) = \frac{1}{\sqrt{2\pi\sigma_{jh}}} e^{-\frac{(x_j-\mu_{jh})^2}{2\sigma_{jh}^2}}$$

Where μ_{jh} and σ_{jh} respectively denote the mean and standard deviation of the variable X_j for the examples of class v_h, and may be estimated on the basis of the examples contained in D.

In spite of the simplifying assumption of conditional independence of the attributes, which makes it easy to compute the conditional probabilities, the empirical evidence shows that Bayesian classifiers are often able to achieve accuracy levels which are not lower than those provided by classification trees or even by more complex classification methods.

area

$$P(area = 1|churner = 0) = \frac{5}{13}, \quad P(area = 1|churner = 1) = \frac{1}{10},$$
$$P(area = 2|churner = 0) = \frac{2}{13}, \quad P(area = 2|churner = 1) = \frac{3}{10},$$
$$P(area = 3|churner = 0) = \frac{5}{13}, \quad P(area = 3|churner = 1) = \frac{2}{10},$$
$$P(area = 4|churner = 0) = \frac{1}{13}, \quad P(area = 4|churner = 1) = \frac{4}{10}.$$

Pothers

$$P(Pothers = 1|churner = 0) = \frac{2}{13}, \quad P(Pothers = 1|churner = 1) = \frac{5}{10},$$
$$P(Pothers = 2|churner = 0) = \frac{3}{13}, \quad P(Pothers = 2|churner = 1) = \frac{4}{10},$$
$$P(Pothers = 3|churner = 0) = \frac{3}{13}, \quad P(Pothers = 3|churner = 1) = 0,$$
$$P(Pothers = 4|churner = 0) = \frac{5}{13}, \quad P(Pothers = 4|churner = 1) = \frac{1}{10}.$$

...and relative frequencies of the two classes

$$P(churner = 0) = \frac{13}{23} = 0.56, \quad P(churner = 1) = \frac{10}{23} = 0.44,$$

Predict the target class for a new observation

$$\mathbf{x} = (1, 1, 1, 2, 1, 4, 2, 1, 2, 1, 0)$$

Compute probabilities

$$P(\mathbf{x}|0) = \frac{5}{13}\cdot\frac{4}{13}\cdot\frac{4}{13}\cdot\frac{3}{13}\cdot\frac{2}{13}\cdot\frac{3}{13}\cdot\frac{4}{13}\cdot\frac{3}{13}\cdot\frac{7}{13}\cdot\frac{12}{13}\cdot\frac{10}{13} = 0.81\cdot 10^{-5}$$

$$P(\mathbf{x}|1) = \frac{1}{10}\cdot\frac{5}{10}\cdot\frac{6}{10}\cdot\frac{5}{10}\cdot\frac{5}{10}\cdot\frac{1}{10}\cdot\frac{6}{10}\cdot\frac{5}{10}\cdot\frac{4}{10}\cdot\frac{9}{10}\cdot\frac{7}{10} = 5.67\cdot 10^{-5}$$

obtaining the posterior probability

$$\begin{aligned} P(churner = 0|\mathbf{x}) &= P(\mathbf{x}|0)\,P(churner = 0) \\ &= 0.81\cdot 10^{-5}\cdot 0.56 = 0.46\cdot 10^{-5}, \\ P(churner = 1|\mathbf{x}) &= P(\mathbf{x}|1)\,P(churner = 1) \\ &= 5.67\cdot 10^{-5}\cdot 0.44 = 2.495\cdot 10^{-5}. \end{aligned}$$

The sum of two probabilities isn't 1 because this is not the posterior probability, just the numerator is the posterior probability.

5.2. Bayesian networks:

Bayesian networks, also called *belief networks*, allow the hypothesis of conditional independence of the attributes to be relaxed, by introducing some reticular hierarchical links through which it is possible to assign selected stochastic dependencies that experts of the application domain deem relevant.

A Bayesian network comprises two main components. The first is an *acyclic oriented graph* in which the nodes correspond to the predictive variables and the arcs indicate relationships of stochastic dependence. In particular, it is assumed that the variable X_j associated with node $\boldsymbol{a}_j$ in the network is dependent on the variables associated with the predecessor nodes of $\boldsymbol{a}_j$, and conditionally independent of the variables associated with the nodes that are not directly reachable from $\boldsymbol{a}_j$.

The second component consists of a table of conditional probabilities assigned for each variable. In particular, the table associated with the variable X_j indicates the conditional distribution of $P(X_j|\checkmark_j)$, where $\checkmark_j$ represents the set of explanatory variables associated with the predecessor nodes of node $\boldsymbol{a}_j$ in the network and is estimated based on the relative frequencies in the dataset.

The complexity required to compute all possible combinations of predictor values for estimating the conditional probabilities, already pointed out above, is limited in Bayesian networks, since the calculation simply reduces to those conditioning relationships determined by the precedence links included in the network. Consequently, the number of precedence links should be restricted to avoid the overwhelming computational effort mentioned above.

6. LOGISTIC REGRESSION:

Logistic regression is a technique for converting binary classification problems into linear regression ones, by means of a proper transformation.

Suppose that the response variable y takes the values $\{0,1\}$, as in a binary classification problem. The logistic regression model postulates that the posterior probability $P(y|\boldsymbol{x})$ of the response variable conditioned on the vector $\boldsymbol{x}$ follows a *logistic function*, given by (*):

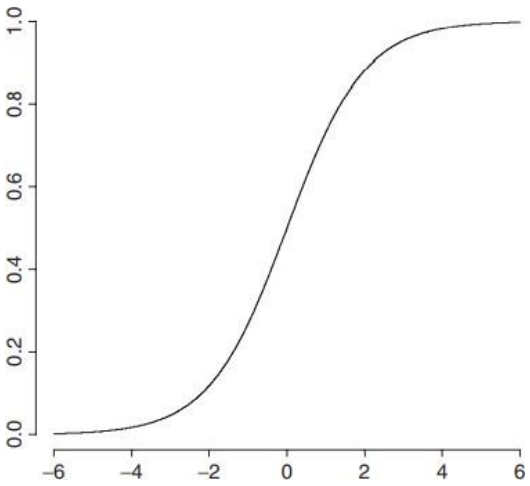
$$P(y=0|\boldsymbol{x}) = \frac{1}{1+e^{\boldsymbol{w}'\boldsymbol{x}}}$$

$$P(y=1|\boldsymbol{x}) = \frac{e^{\boldsymbol{w}'\boldsymbol{x}}}{1+e^{\boldsymbol{w}'\boldsymbol{x}}}$$

Here we suppose that the matrix $\boldsymbol{X}$ and the vector $\boldsymbol{w}$ have been extended to include the intercept. The standard logistic function $S(t)$, also known as the *sigmoid function*, can be found in many applications of statistics in the economic and biological fields and is defined as:

$$S(t) = \frac{1}{1+e^{-t}}$$

The function S(t) has the graphical shape shown in figure:



The ratio between these two probabilities is called odds ratio, is the ratio between the probability of being positive given x over the probability of being negative given x.

By inverting the two expressions above (*), we observe that the logarithm of the ratio between the conditional probabilities of the two classes depends linearly on the predictive variables, that is:

$$\log\frac{P(y=1|\boldsymbol{x})}{P(y=0|\boldsymbol{x})} = \boldsymbol{w}'\boldsymbol{x}$$

By setting:

$$z = \log\frac{P(y=1|\boldsymbol{x})}{P(y=0|\boldsymbol{x})}$$

We obtain the solution of linear regression model:

$$z = \boldsymbol{w}'\boldsymbol{x}$$

Consequently, the binary classification problem is traced back to the identification of a linear regression model between the dependent variable z and the original explanatory attributes. Once the linear regression coefficients have been calculated and the significance of the model verified, one may use the model to predict the target class of a new observation $\boldsymbol{x}$. The coefficients $\boldsymbol{w}$ are computed using an iterative method, usually aimed at maximizing the likelihood, by minimizing the sum of logarithms of predicted probabilities.

In general, logistic regression models present the same difficulties described in connection with regression models, from which they derive. To avoid multicollinearity phenomena that jeopardize the significance of the regression coefficients it is necessary to proceed with attribute selection. Moreover, the accuracy of logistic regression models is in most cases lower than that obtained using other classifiers and usually requires a greater effort for the development of the model. Finally, it appears computationally cumbersome to treat large datasets, both in terms of number of observations and number of attributes.

The limitations I should have a small number of variables, when the number increase is very hard to get a solution, also the accuracy is not good. It maybe worth to be used if we want an interpretation and I have a small number of variables.

Logistic regression: an example

January 1986: Explosion of the Space Shuttle Challenger

Launch temperature and *O Rings* failures

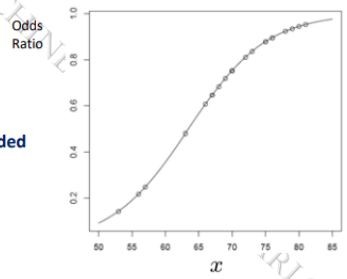
launch	temp.	failure	launch	temp.	failure	launch	temp.	failure
1	53	Y	9	68	N	17	75	N
2	56	Y	10	69	N	18	75	Y
3	57	Y	11	70	N	19	76	N
4	63	N	12	70	Y	20	76	N
5	66	N	13	70	Y	21	78	N
6	67	N	14	70	Y	22	79	N
7	67	N	15	72	N	23	80	N
8	67	N	16	73	N	24	81	N

Odds Ratio = $\ln \frac{p(x)}{1-p(x)} = \beta_0 + \beta_1 x$ $p(x)$ **is the success probability (failure = N) when temp. =** x

Parameters estimate:

$\beta_1 = 0.17$

$\beta_0 = -10.8$ (intercept)



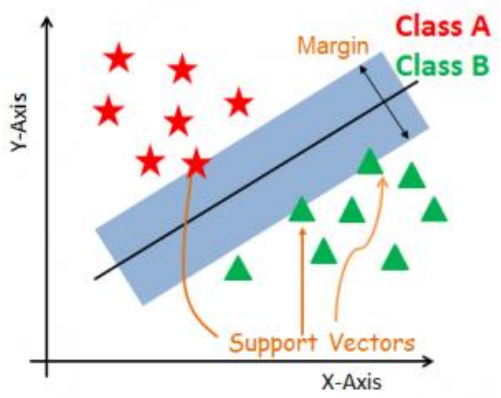
Temperature when Challenger exploded was 31°F...

... the Odds Ratio was ?

$x = 31$ Odds Ratio = 0.0038 !!!

7. SUPPORT VECTOR MACHINES:

Le **Support-Vector Machine** o **SVM** (in italiano **Macchine a vettori di supporto**) sono modelli di classificazione il cui obiettivo è quello di trovare la retta di separazione delle classi che massimizza il *margine* tra le classi stesse, dove con *margine* si intende la distanza minima dalla retta ai punti delle due classi.



Questo obiettivo viene raggiunto utilizzando una parte minimale del dataset di allenamento, i cosiddetti **vettori di supporto** (da cui il nome della famiglia di modelli).

Nella figura a fianco si nota quale sia la retta di separazione che massimizza il margine tra le due classi di dati. La stella e i due triangoli indicati sono i vettori di supporto, che come si può notare sono gli unici esempi del dataset che risiedono sul margine. Una volta trovati questi, tutti gli altri esempi del dataset saranno ininfluenti ai fini della classificazione, perchè sono loro a definire la retta di separazione ed il margine.

Ma cosa rappresentano i vettori di supporto? Sono i valori di una classe più vicini alla retta di separazione, quelli che più si avvicinano all'altra classe. In sostanza sono i valori classificabili con maggiore difficoltà.

Ed è proprio questo che differisce la SVM da altri algoritmi di classificazione: mentre una regressione logistica "impara" a classificare prendendo come riferimento gli esempi più rappresentativi di una classe, la SVM cerca gli esempi più difficili, quelli che tendono ad essere più vicini all'altra classe (i vettori di supporto, appunto) e considera solo quelli per eseguire la classificazione.

Questo approccio lo possiamo riassumere con questa frase: "*se la classificazione vale per gli esempi più difficili, vale sicuramente anche per tutti gli altri*".

Un'altro aspetto interessante è che maggiore è il margine, migliore sarà la generalizzazione. Il motivo è abbastanza semplice da capire: maggiore è il margine, maggiore è la distanza tra le classi e quindi minore la possibilità di fare confusione

Support vector machines are a family of separation methods for classification and regression developed in the context of statistical learning theory. They have been shown to achieve better performance in terms of accuracy with respect to other classifiers in several application domains, and to be efficiently scalable for large problems. A further important feature is concerned with the interpretation of the classification rules generated. Support vector machines identify a set of examples, called support vectors, which appear to be the most representative observations for each target class. In a way, they play a more critical role than the other examples, since they define the position of the separating surface generated by the classifier in the attribute space.

7.1. Structural risk minimization:

We want to minimize the empirical error on the training set.

As already observed, a classification algorithm $A_{\mathcal{F}}$ defines an appropriate hypothesis space $\mathcal{F}$ and a function $f^* \in \mathcal{F}$ which optimally describes the relationship between the class value y and the vector of explanatory variables $\boldsymbol{x}$. In order to describe the criteria for selecting the function f^*, let $V(y, f(\boldsymbol{x}))$ denote a loss function which measures the discrepancy between the values returned by the predictive function $f(\boldsymbol{x})$ and the actual values of the class y.

To select an optimal hypothesis $f^* \in \mathcal{F}$, decision theory suggests minimizing the *expected risk* functional, defined as:

$$R(f) = \frac{1}{2}\int V(y, f(\boldsymbol{x}))dP(\boldsymbol{x}, y)$$

Where $P(\boldsymbol{x}, y) = P_{\boldsymbol{x},y}(\boldsymbol{x}, y)$ denotes the joint probability distribution over $\mathbb{R}^n \times \mathcal{H}$ of the examples $(\boldsymbol{x}, y)$ from which the instances in the dataset D are assumed to be independently drawn.

Since the distribution $P(\boldsymbol{x}, y)$ is generally unknown, in place of the expected risk one is naturally led to minimize the *empirical risk* over the training set T, defined as:

$$R_{emp}(f) = \frac{1}{t}\sum_{i=1}^{t} V(y_i, f(\boldsymbol{x}_i))$$

However, this induction principle for selecting the function f^*, called *empirical risk minimization* (ERM), suffers from two critical issues:

- ***Overfitting:*** more I reduce the error on the training set, more increase the error on the test set; the first limitation is to increase the error on the test set
- ***ill-posedness:*** it's a problem in which small changes in the input data result in large changes in the solution (solution is not robust)

If the space $\mathcal{F}$ is chosen too broad, the empirical error can be significantly reduced, eventually to zero, but the optimal hypothesis f^* has a low generalization capability in predicting the output value of the instances in the test set $\mathcal{V}$ or of future unseen examples. In these cases, the expected risk may still be large even if the empirical risk approaches 0, since the minimization of $R_{emp}(f\ )$ does not necessarily imply the minimization of $R(f)$.

This phenomenon, known as *overfitting*, has been investigated in detail in the framework of statistical learning theory, leading to probability bounds on the difference between expected risk and empirical risk. In general, these bounds are inversely dependent on the size t of the training set T and directly dependent on the capacity, known as the *Vapnik–Chervonenkis dimension* (*VC*), which **measures the complexity of the hypothesis space $\mathcal{F}$.**

The VC dimension of the hypothesis space $\mathcal{F}$ is defined as the maximum number of training points that can be correctly classified by a function from $\mathcal{F}$, wherever the points are placed in the space and whatever their binary class values $\{-1, 1\}$ are. If the VC dimension is h, then there exists at least one set of $h + 1$ points that cannot be correctly classified by the hypotheses belonging to the space $\mathcal{F}$.

For instance, if the training examples are represented by points in two-dimensional space, the class of functions represented by oriented straight lines has dimension *VC = 3*, since there exist setsof four points which cannot be shattered by means of a single separating line, as depicted in the figure below.

More generally, it has been shown that the class of separating hyperplanes in the space $\mathbb{R}^n$ has dimension $VC = n + 1$. According to the most common bound on the expected risk, for a binary classification problem the inequality (**):

$$R(f) \leq \sqrt{\frac{\gamma \log(2t/\gamma) + 1 - \log(\eta/4)}{t}} + R_{emp}(f)$$
$$\leq \Psi(t, \gamma, \eta) + R_{emp}(f),$$

holds (contenuta) with probability $1 - \eta$, where $0 \leq \eta \leq 1$ is a predefined confidence level, γ is the VC dimension of the chosen hypothesis space $\mathcal{F}$, and $\Psi(t, \gamma, \eta)$ is a function called the *VC confidence* which represents the generalization error. $\Psi(t, \gamma, \eta)$ depends of three parameters:

- t depends on the size of training set (larger is the training, more the bound will assume the value of zero)
- gamma is the VC dimension
- eta represent the level of confidence (more confidence (confidence=1-eta)→ eta smaller, I expect this bound go to infinity).

As a consequence, in order to control the classification error against broader hypothesis space, a larger training set is required, according to a proportion which is regulated by theoretical bounds on the error, cast in the form of the expression (**). Hence, in order to identify a hypothesis $f^* \in \mathcal{F}$ capable of achieving a high level of accuracy on the training set and a good generalization capability on the test set and on future sets of examples, one may resort to the *structural risk minimization* (SRM) principle, which formally establishes the concept of choosing the function $f^* \in \mathcal{F}$ which minimizes the right hand-side in the expression (**). Furthermore, the minimization of the empirical risk is an ill-conditioned problem, in the sense that the parameters of the optimal hypothesis f^* may vary significantly for small changes in the input data.

The solution is regularization theory, with this we introduce a regularize term.

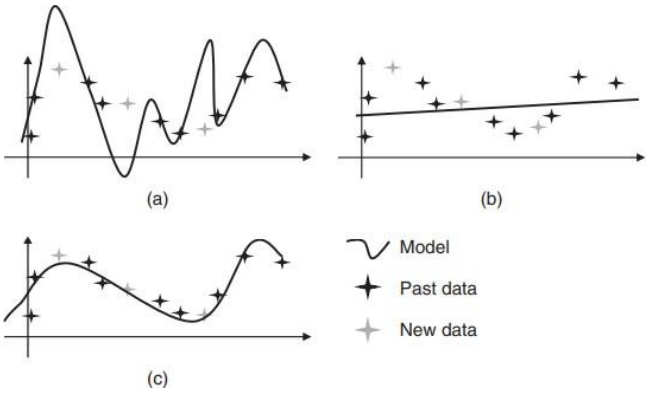
A way to circumvent the second weakness has been devised in the context of regularization theory and also applied in statistical learning theory to prevent overfitting. It relies on the solution of a well-posed problem which corresponds to the structural risk minimization principle, and chooses as the optimal hypothesis $f^* \in \mathcal{F}$ the one that minimizes a modified risk functional:

$$\hat{R}(f) = \frac{1}{t}\sum_{i=1}^{t} V(y_i, f(\boldsymbol{x}_i)) + \lambda ||f||_K^2$$

where K is a given symmetric positive definite function called *kernel*, $||f||_K^2$ denotes the norm of f in the reproducing kernel Hilbert space induced by K, and λ is a parameter that controls the trade-off between the empirical error and the generalization capability. The parameter λ can be interpreted as a penalty against a hypothesis f with high complexity VC and low generalization capability, and also as a smoothing regularizer for transforming the variational problem into a well-posed one. If I take lambda smaller, close to zero, we are focusing on the past data, the train data;

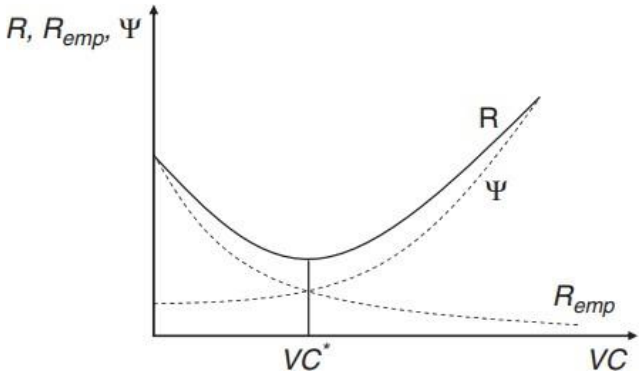
if I take lambda larger we increase the attention on regularize term that is the generalization capability of my model. Lambda is chosen by performing experiments, different values are used to performing test and checking what happen.

The figure below provides an intuitive explanation of how it is possible to find an ideal trade-off between the accuracy on the training set and the generalization capability to future unseen examples.



The dark gray points represent the training examples, while the light gray ones correspond to new instances. The hypothesis shown Figure (a) belongs to a too broad space $\mathcal{F}$, which may be composed, for example, of high order polynomials; it is associated with a negligible classification error on the training set, but it is not able to generalize well to future instances. On the other hand,the hypothesis in Figure (b) originates from a too narrow space, whereas in Figure (c) an ideal trade-off is achieved between the ability to provide an accurate classification of the training examples andthe future generalization capability.

The following figure shows the conflict effects produced on the two right-hand-side terms in the expression (**) by an increase in the complexity of the hypothesis space $\mathcal{F}$, measured by the VC dimension along the horizontal axis



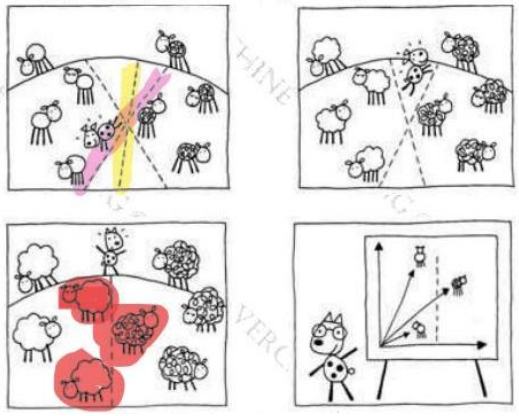
An increase in the complexity corresponds to a decrease in the empirical error but, at the same time, to an increase in the VC confidence which represents the generalization error.

ACCURACY VS GENERALIZATION

In terms of generalization aren't the same, the best trajectory is that one who minimize the distance between trajectory and sheep.

Black and white can be separated in many different ways, there are infinitive different trajectory. But what is the best trajectory? All trajectories are equal in terms of empirical error, but in terms of generalization capability aren't equal, some are near the sheep other are distant from sheep.

The intuitive idea is that **the best trajectory is the one with the maximum minimum** distance from the sheep.



In the first imagine the best trajectory between yellow and pink one is the yellow and not pink, because if the two nearest sheep move themselves we couldn't classify them (the model is not robust).

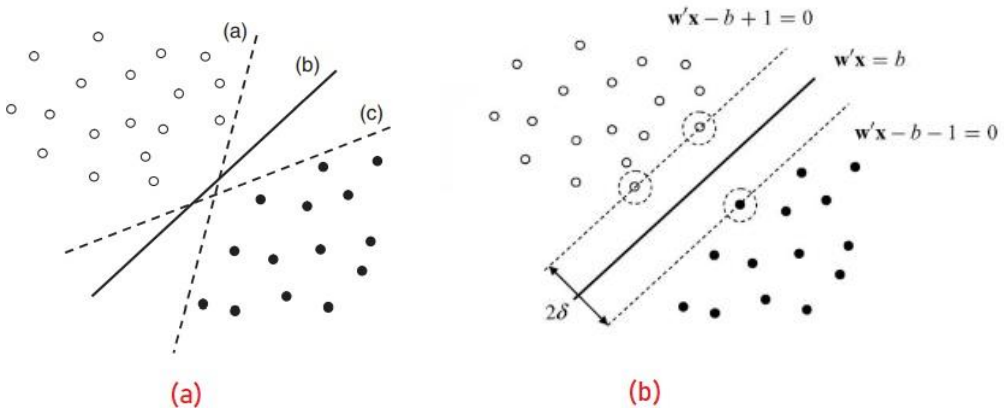
In order to solve optimization problem, we have **to maximize the margin separation.**

In the last figure the dog understand that some sheep are more important than other because their position is determining the position of the best trajectory, this sheep are the support vector.

7.2. Maximal margin hyperplane for linear separation:

Once the theoretical framework for the supervised learning process has been established, we are able to provide a geometrical interpretation of the structural risk minimization principle, and to formulate an optimization problem which aims to determine a linear separating surface for solving binary classification problems.

For linear separating functions represented by the hyperplanes in the space $\mathbb{R}^n$, the minimization of the right-hand side in (**) can be traced back to the maximization of the margin of separation, which will be described with reference to the figures below.



(a) (b)

Two sets of points belonging to binary target classes, like the ones depicted in Figure (a), are said to be linearly separable if there exists a hyperplane capable of separating them in the space $\mathbb{R}^n$, reducing to a line in the two-dimensional case. As shown in Figure (a), there are an infinite number of linear separating functions that appear equivalent to each other in terms of empirical error on

the training set, which is equal to zero for all the lines drawn in the figure. However, lines (a) and (c), which lie near the training examples, have lower generalization capability with respect to a line, such as (b), that is equally distant from the nearest points of the two sets.
The optimal separating line, endowed with the maximum generalization capability and the minimum empirical error, is drawn in Figure (b). The margin of separation is defined as the distance between the pair of parallel *canonical supporting hyperplanes*, as shown in the same figure. Hence, the margin is equal to twice the minimum distance between the training points and the separating hyperplane. Those training points which are at the minimum distance from the separating hyperplane, and thus lie on the canonical hyperplanes, are called *support vectors*, and play a more prominent role with respect to the other examples, since it is precisely these points that determinethe classification rule. Letting $\boldsymbol{w}$ denote the vector of the hyperplane coefficients and b the intercept, the separating hyperplane is given by:

$$\boldsymbol{w}'\boldsymbol{x} = b$$

while the two canonical supporting hyperplanes are:

$$\boldsymbol{w}'\boldsymbol{x} - b - 1 = 0, \qquad \boldsymbol{w}'\boldsymbol{x} - b + 1 = 0$$

The margin of separation δ is defined as:

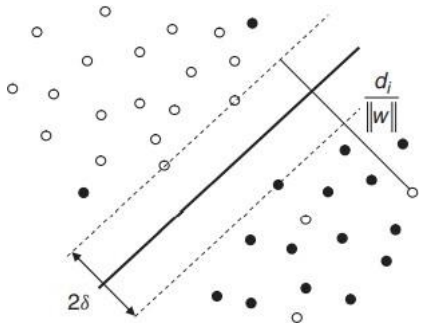
$$\delta = \frac{2}{||w||}, where\ ||w|| = \sqrt{\sum_{j \in \mathcal{N}} w_j^2}$$

I want to maximize delta, so I want to minimize w. I have to impose the separating conditions, that a point lie on one support vectors and the other point on the other support vectors.
In order to determine the coefficients $\boldsymbol{w}$ and b of the optimal separating hyperplane, a quadratic optimization problem with linear constrains can be solved:

$$\min_{\boldsymbol{w},b} \frac{1}{2}||\boldsymbol{w}||^2$$

$$\text{s.to} \quad y_i(\boldsymbol{w}'\boldsymbol{x}_i - b) \geq 1, \qquad i \in \mathcal{M} \quad (*)$$

The aim of the objective function is to maximize the margin of separation through the minimization of its reciprocal, while the constraints $(*)$ force each point $\boldsymbol{x}_i$ to lie in the half space corresponding to the class value y_i.
It is most likely that the m points of a dataset cannot be linearly separable, as for the set of examples depicted in the figure below.



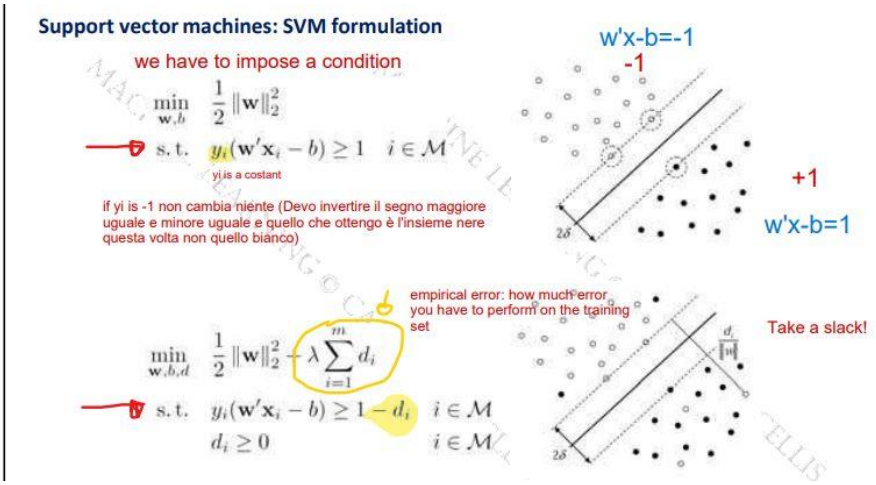
In these cases, it is necessary to relax the constraints $(*)$ by replacing them with weaker conditions which allow for the presence of possible misclassification errors, and to modify the objective function of the optimization problem. This can be done by introducing a new set of slack variables $d_i, i \in M$, which measure the **positive** differences between the values of the misclassified examples on the vertical axis and the ordinate values along the canonical hyperplane that defines the region associated with the class value y_i, as geometrically illustrated in the figure above.
In the non-separable case, we can therefore formulate the following optimization problem:

$$\min_{w,b,d} \frac{1}{2}||w||^2 + \lambda \sum_{i=1}^{m} d_i \quad (\bullet)$$

$$\text{s.to} \quad y_i(w'x_i - b) \geq 1 - d_i, \qquad i \in \mathcal{M} \quad (\bullet\bullet)$$

$$d_i \geq 0, \quad i \in \mathcal{M} \quad (\bullet\bullet\bullet)$$

The objective function is composed of the weighted sum of two terms representing respectively the reciprocal of the margin of separation and the empirical error. The parameter λ is introduced in order to regulate the trade-off between the generalization capability, represented by the reciprocal of the margin, and the accuracy on the training set, evaluated as the sum of the slack variables. The quadratic problem $(\bullet)$ can be solved via Lagrangian duality. Among other advantages, this allows us to identify the support vectors, which are associated with positive Lagrange multipliers in the optimal solution of the dual problem.



Denote by $\alpha_i \geq 0$ the Lagrangian multipliers of the constraints $(\bullet\bullet)$, and by $\mu_i \geq 0$ the multipliers of the constraints $(\bullet\bullet\bullet)$. The Lagrangian function of problem $(\bullet)$ is given by:

$$L(\mathbf{w}, b, \mathbf{d}, \boldsymbol{\alpha}\, \boldsymbol{\mu}) = \frac{1}{2}\|\mathbf{w}\|^2 + \lambda \sum_{i=1}^{m} d_i - \sum_{i=1}^{m} \alpha_i [y_i(\mathbf{w}'\mathbf{x}_i - b) - 1 - d_i] - \sum_{i=1}^{m} \mu_i d_i$$

In order to find the optimal solution, the derivatives with respect to the variables $\boldsymbol{w}, \boldsymbol{d}, b$ of the primal problem (•) must be set to 0:

$$\frac{\partial L(\mathbf{w}, b, \mathbf{d}, \boldsymbol{\alpha}\, \boldsymbol{\mu})}{\partial \mathbf{w}} = \mathbf{w} - \sum_{i=1}^{m} \alpha_i y_i \mathbf{x}_i = \mathbf{0},$$
$$\frac{\partial L(\mathbf{w}, b, \mathbf{d}, \boldsymbol{\alpha}\, \boldsymbol{\mu})}{\partial b} = \lambda - \alpha_i - \mu_i = 0,$$
$$\frac{\partial L(\mathbf{w}, b, \mathbf{d}, \boldsymbol{\alpha}\, \boldsymbol{\mu})}{\partial \mathbf{d}} = \sum_{i=1}^{m} \alpha_i y_i = 0,$$

Leading to the conclusions:

$$\boldsymbol{w} = \sum_{i=1}^{m} \alpha_i y_i \boldsymbol{x}_i \,,$$

$$\lambda = \alpha_i + \mu_i \,,$$

$$\sum_{i=1}^{m} \alpha_i y_i = 0.$$

By substituting these equality constraints into the Lagrangian function (^), we obtain the objective function of the dual problem:

$$L(\mathbf{w}, b, \mathbf{d}, \boldsymbol{\alpha}\, \boldsymbol{\mu}) = \sum_{i=1}^{m} \alpha_i - \frac{1}{2} \sum_{i=1}^{m} \sum_{k=1}^{m} y_i y_k \alpha_i \alpha_k \mathbf{x}_i' \mathbf{x}_k$$

with the additional constraints $\alpha_i \leq \lambda,\ i \in M$.

The Karush–Kuhn–Tucker complementarity conditions applied to the pair of primal–dual problems lead to the equalities:

$$\alpha_i[y_i(\boldsymbol{w}'\boldsymbol{x}_i - b) - 1 + d_i] = 0, \quad i \in M$$

$$\mu_i(\alpha_i - \lambda) = 0, \quad i \in M \ (\blacksquare)$$

In particular, conditions (■) allow us to identify the support vectors, which are the most representative points in determining the classification rules learned from the training set. Indeed, the examples $\boldsymbol{x}_i$ whose Lagrangian multipliers satisfy the condition $0 < \alpha < \lambda$ are at distance $1/||\boldsymbol{w}||$ from the separating hyperplane, and thus are located on one of the two canonical hyperplanes (support vectors).

Finally, observe that for support vector machines the decision function which classifies a new observation $\boldsymbol{x}$ is given by:

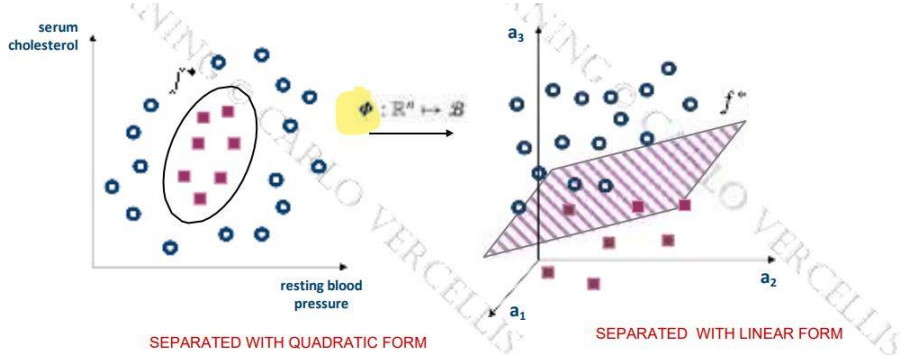
$$f(\boldsymbol{x}) = sgn(\boldsymbol{w}'\boldsymbol{x} - b)$$

By substituting $\boldsymbol{w} = \sum_{i=1}^{m} \alpha_i y_i \boldsymbol{x}_i$ into the expression above, we can reformulate the decision function as (^^):

$$f(\boldsymbol{x}) = sgn\left(\sum_{i=1}^{m} \alpha_i y_i \boldsymbol{x}'\boldsymbol{x}_i - b\right)$$

7.3. Nonlinear separation with SVM:

Linear separating functions are not able to perform accurate classifications when the set of examples is intrinsically characterized by a nonlinear pattern, like that shown in Figure (a)



We want to transform the original space in another space using "phi", I want to transform my space in another with more dimension one.

In such cases, one may resort to mappings of the attributes which allow one to obtain linearly separable datasets in the transformed space, which is called the *feature space*. The figure above shows a possible transformation enjoying this property. In Figure(a) the examples of opposite classes can be separated by an elliptical function represented by a second-order polynomial in the explanatory variables. If the original two-dimensional space is transformed into a five-dimensional space which contains the second-order monomials x_1, x_2, ${x_1}^2$, ${x_2}^2$, in addition to the original attributes x_1 and x_2, it is possible to linearly separate the transformed points by means of a hyperplane defined in the space $\mathbb{R}^n$. Notice that the linear separation achieved in this way is made possible by an increase in the number of attributes in the feature space, which may be quite relevant for some applications.

Besides the intuitive justification based on the figure above, the mapping from the original attribute space to a feature space of high – sometimes even infinite – dimensionality can also be justified from a theoretical point of view by Cover's theorem. We already know that m points in a general position in the n-dimensional space can be linearly separated if $m \leq n + 1$, since the VC dimension of the class of hyperplanes is $n + 1$, so that the number of possible linear separations is $2m$. If, however, $m > n + 1$, Cover's theorem states that the number of linear separations is givenby:

$$2\sum_{i=0}^{n}\binom{m-1}{i}$$

As n increases, the number of terms in the sum increases in turn, so that there are more linear separations.

This line of reasoning may, however, entail some practical difficulties.

Indeed, the transformation must be carried out very efficiently even if the feature space is of high or infinite dimension. Fortunately, there exists a wide class of mappings which can be evaluated in a very efficient way, consisting of *kernel functions*, for which the mapping of the original observations into the feature space is not explicitly computed. The use of kernels is based on the dual formulation of problem (•), and implicitly involves a linear separation of the examples in high-dimensional functional spaces – or even infinite-dimensional for some families of kernels.
Specifically, consider a map $\boldsymbol{\Phi}$: $\mathbb{R}^n \mapsto B$ that transforms the examples given in the space $\mathbb{R}^n$ of the original attributes into a Hilbert space B, called *feature space*.
Observe now that the arguments set out in the previous section to derive the maximal margin hyperplane can easily be adapted to achieve an optimal linear separation in the feature space. To do this, we simply have to substitute $\boldsymbol{\Phi}(\mathbf{x})$ wherever we wrote $\boldsymbol{x}$ in expressions $\boldsymbol{w}'\boldsymbol{x} = b$ through (^^). In particular, the Lagrangian dual problem takes the form (Δ):

$$L(\boldsymbol{w}, b, \boldsymbol{d}, \boldsymbol{\alpha}, \boldsymbol{\mu}) = \sum_{i=1}^{m} \alpha_i - \frac{1}{2} \sum_{i=1}^{m} \sum_{h=1}^{m} y_i y_h \alpha_i \alpha_h \boldsymbol{\Phi}(\mathbf{x}_i)' \boldsymbol{\Phi}(\mathbf{x}_h)$$

And we obtain the following decision function in the feature space:

$$f(\boldsymbol{x}) = sgn\left(\sum_{i=1}^{m} \alpha_i y_i \boldsymbol{\Phi}(\mathbf{x})' \boldsymbol{\Phi}(\mathbf{x}_i) - b\right)$$

Now, the key point is that the data appear in the Lagrangian dual (Δ) and in the corresponding decision function $f(\boldsymbol{x})$ only in the form of inner products $\boldsymbol{\Phi}(\mathbf{x}_i)'\boldsymbol{\Phi}(\mathbf{x}_h)$ and $\boldsymbol{\Phi}(\mathbf{x})'\boldsymbol{\Phi}(\mathbf{x}_i)$. Hence, the potentially expensive computation of the images $\boldsymbol{\Phi}(\mathbf{x}_i)$, $i \in M$, of the observations under the mapping $\boldsymbol{\Phi}$ can be dramatically simplified if there exists a positive definite *kernel function k* such that:

$$\boldsymbol{\Phi}(\mathbf{x}_i)'\boldsymbol{\Phi}(\mathbf{x}_h) = k(\mathbf{x}_i, \mathbf{x}_h)$$

If such a kernel function to represent the mapping $\boldsymbol{\Phi}$ exists, then the dual problem takes the form (ΔΔ):

$$L(\boldsymbol{w}, b, \boldsymbol{d}, \boldsymbol{\alpha}, \boldsymbol{\mu}) = \sum_{i=1}^{m} \alpha_i - \frac{1}{2} \sum_{i=1}^{m} \sum_{h=1}^{m} y_i y_h \alpha_i \alpha_h \, k(\mathbf{x}_i, \mathbf{x}_h)$$

With the additional constraints $\alpha_i \leq \lambda$, $i \in M$. Since k is positive definite, the matrix $\boldsymbol{Q} = [Q_{ij}] = [y_i y_h k(\mathbf{x}_i, \mathbf{x}_h)]$ is positive definite in turn, so that the problem (ΔΔ) is convex and can be solved efficiently to derive the dual multipliers α_i, $i \in M$. Therefore, this allows us to identify the support vectors, and to express the decision function in the form:

$$f(\boldsymbol{x}) = sgn\left(\sum_{i=1}^{m} \alpha_i y_i k(\mathbf{x}_i, \mathbf{x}_h) - b\right)$$

The linear separation in the feature space hence corresponds to a nonlinear separation in the space of the original attributes, which by means of kernel functions can be computed with basically the same effort as a linear separation in the original space.

There remains the question of the existence of meaningful kernel functions that enable accurate nonlinear separations to be computed. Sufficient conditions for the existence of pairs $(k, \mathbf{\Phi})$ consisting of a kernel function and a map $\mathbf{\Phi}$ represented by k, are provided by Mercer's theorem, whose formulation goes beyond the scope of our discussion and for which the reader is directed to the references suggested at the end of this chapter. Mercer's theorem also implies that the linear combination of kernel functions is in turn a kernel function, so that it provides a rule for deriving more complex kernels starting from simple ones.

Many kernels have been proposed in the literature, among which the most popular parametric families are:

- *polynomial kernels* of degree d:

$$k(\mathbf{x}_i, \mathbf{x}_h) = (\mathbf{x}'_i \mathbf{x}_h + 1)^d$$

- *radial basis function kernels*, also called *Gaussian kernels*, of width $\sigma > 0$:

$$k(\mathbf{x}_i, \mathbf{x}_h) = \exp\left(\frac{-||\mathbf{x}_i - \mathbf{x}_h||^2}{2\sigma^2}\right)$$

- *neural network kernels*, with a hyperbolic tangent activation function $\kappa > 0$:

$$k(\mathbf{x}_i, \mathbf{x}_h) = \tanh(\kappa \mathbf{x}'_i \mathbf{x}_h - \delta)$$

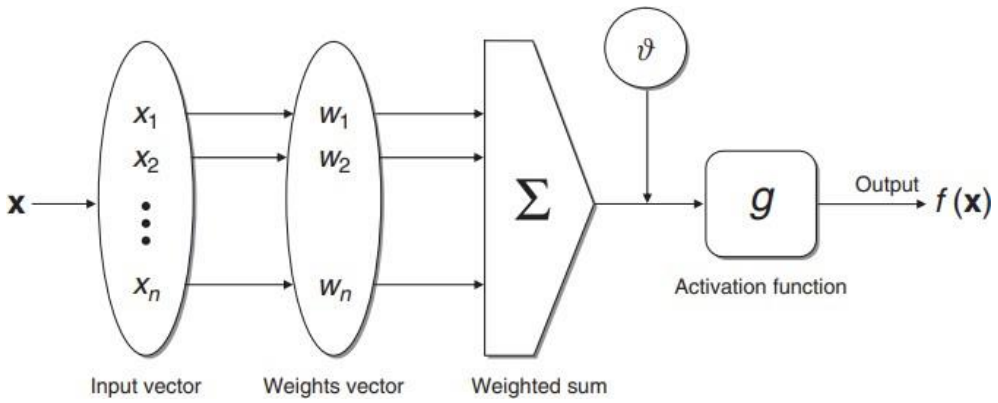
Notice that nonlinear separations can be also attained in the original space of attributes where the examples are defined, without resorting to mappings carried out by means of kernel functions.

8. NEURAL NETWORKS:

Neural networks are intended to simulate the behavior of biological systems composed of neurons.
A neural network is an oriented graph consisting of nodes, which in the biological analogy represent neurons, connected by arcs, which correspond to dendrites and synapses. Each arc is associated with a *weight*, while at each node an *activation function* is defined which is applied to the values received as input by the node along the incoming arcs, adjusted by the weights of the arcs. The training stage is performed by analyzing in sequence the observations contained in the training set one after the other and by modifying at each iteration the weights associated with the arcs.

8.1. The Rosenblatt perceptron:

The perceptron, shown in the figure below, is the simplest form of neural network and corresponds to a single neuron that receives as input the values $(x_1, x_2, \dots, x_n)$ along the incoming connections, and returns an output value $f(\boldsymbol{x})$.



Theinput values coincide with the values of the explanatory attributes, while the output value determines the prediction of the response variable y. Each of the n input connections is associated with a weight w_j . An activation function g and a constant teta, called the *distortion*, are also assigned.Suppose that the values of the weights and the distortion have already been determined during thetraining phase. The prediction for a new observation $\boldsymbol{x}$ is then derived by performing the following steps.
First, the weighted linear combination of the values of the explanatory variables for the new observation is calculated and the distortion is subtracted from it (*):

$$w_1x_1 + w_2x_2 + \cdots + w_nx_n - \vartheta = \boldsymbol{w}'\boldsymbol{x} - \vartheta$$

The prediction $f(\boldsymbol{x})$ is then obtained by applying the activation function g to the linear combination of the predictors:

$$f(\boldsymbol{x}) = g(w_1x_1 + w_2x_2 + \cdots + w_nx_n - \vartheta) = g(\boldsymbol{w}'\boldsymbol{x} - \vartheta)$$

The purpose of the function g is to map the linear combination into the set $\mathcal{H} = \{v_1, v_2, \dots, v_H\}$ of the values assumed by the target variable, usually by means of a sigmoid profile. For binary classification problems we have $\mathcal{H} = \{-1, 1\}$, so that one may select $g(\cdot) = sgn(\cdot)$, making the prediction coincide with the sign of the weighted sum in (*):

$$f(\boldsymbol{x}) = sgn(w_1x_1 + w_2x_2 + \cdots + w_nx_n - \vartheta) = g(\boldsymbol{w}'\boldsymbol{x} - \vartheta)$$

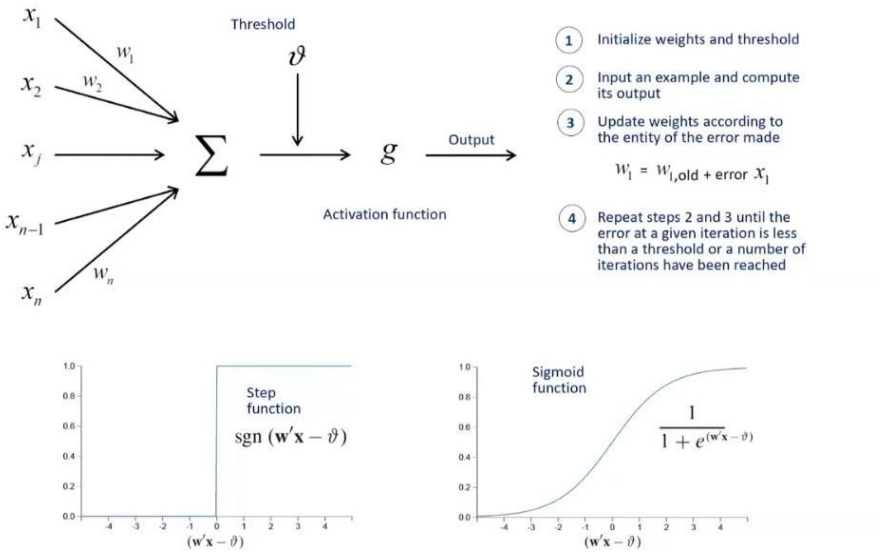
An iterative algorithm is then used to determine the values of the weights w_j and the distortion teta, examining the examples in sequence, one after the other. For each example $\boldsymbol{x}_i$ the prediction $f(\boldsymbol{x}_i)$ is calculated, and the value of the parameters is then updated using recursive formulas that consider

the error $y_i - f(\boldsymbol{x}_i)$.
For binary classification problems it is possible to give a geometrical interpretation of the prediction obtained using a Rosenblatt perceptron. Indeed, if we place the m observations of the training dataset in the space $\mathbb{R}^n$, the weighted linear combination in (*) calculated for $\boldsymbol{x}_i$ expresses the slack between the observation and the hyperplane:

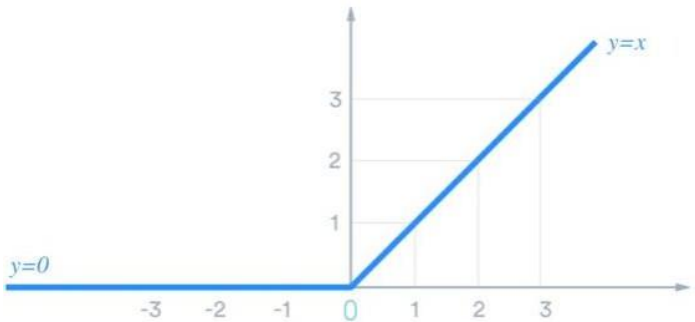
$$z = w_1 x_1 + w_2 x_2 + \cdots + w_n x_n - \vartheta = \boldsymbol{w}'\boldsymbol{x} - \vartheta$$

The purpose of the activation function $g(\cdot) = sgn(\cdot)$ is therefore to establish if the point associated with the example $\boldsymbol{x}_i$ is placed in the lower or upper halfspace with respect to the separating hyperplane. Hence, the Rosenblatt perceptron corresponds to a linear separation of the observations based on the target class. The aim of the iterative procedure is therefore to determine the coefficients of the separating hyperplane.

We summarize the structure of Rosenblatt's Perceptron:



The step function and sigmoid function are used as activation functions in the output node. Most of the times, for internal nodes, it is more convenient to use a different activation function, which is called **ReLu** activation function (Rectified Linear unit activation function), for negative value the error is zero and becomes positive for positive values:

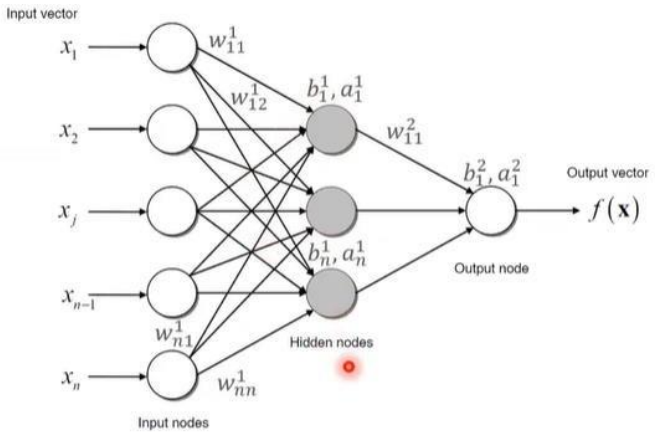


Let's clarify what internal nodes (hidden nodes) are, by introducing the Multi-layer feed-forward neural networks.

8.2. Multi-layer feed-forward networks:

A multi-level feed-forward neural network, shown in the figure below, is a more complex structure than the perceptron, since it includes the following components:

- Input nodes: the purpose of the input nodes is to receive as input the values of the explanatory attributes for each observation. Usually, the number of input nodes equals the number of explanatory variables.
- Hidden nodes: hidden nodes apply given transformations to the input values inside the network. Each node is connected to incoming arcs that go from other hidden nodes or from input nodes, and it is connected with outgoing arcs to output nodes or to other hidden nodes.
- Output nodes: Output nodes receive connections from hidden nodes or from input nodes and return an output value that corresponds to the prediction of the response variable. In classification problems, there is usually only one output node.



1. Initialize weights and threshold.
2. Input an example, compute its output and the corresponding error.
3. Starting with the output layer propagate the error back to the previous layer and update the weights according to the "strength" of the connection.
4. Repeat steps 2 and 3 for all the training examples.

$$w_j^{(k+1)} = w_j^{(k)} + \lambda(y_i - \hat{y}_i^{(k)})x_{ij}$$

Each hidden unit can be considered as a single perceptron.

Each node of the network basically operates as a perceptron, in the sense that given weights are associated with the input arcs, while each node is associated with a distortion coefficient and an activation function. In general, the activation function may assume forms that are more complex than the sign function $sgn(\cdot)$, such as a linear function, a sigmoid or a hyperbolic tangent, such as:

$$a_i^l = \Phi\left(\sum_n w_{ni}^l a_n^{l-1} + b_i^l\right)$$

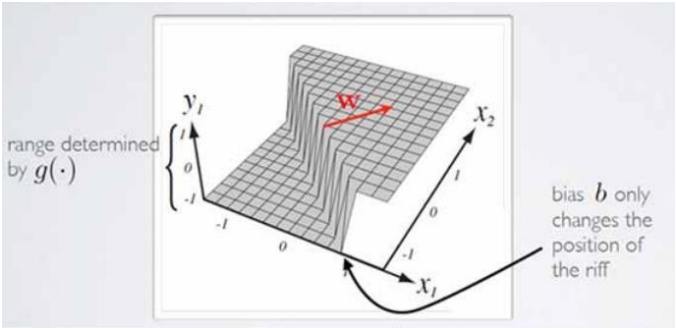
where Φ is the element-wise activation function (which is usually the ReLu activation function).

The method that determines the weights of all the arcs and the distortions at the nodes, called the *backpropagation algorithm*, follows a logic that is not dissimilar from that described for the single perceptron. The weights are initialized in an arbitrary way, for instance by setting their value equal to randomly generated numbers. The examples of the training set are therefore examined in sequence, using at each iteration the current values of the weights, in order to calculate the prediction and the corresponding misclassification error. This latter is used to recursively correct the values of the weights, used at a later time to analyze the subsequent example within the procedure. The weights are updated using a descent algorithm, which is a variant of the gradient method. The formula above highlighted in yellow represents the general relationship for the updating.

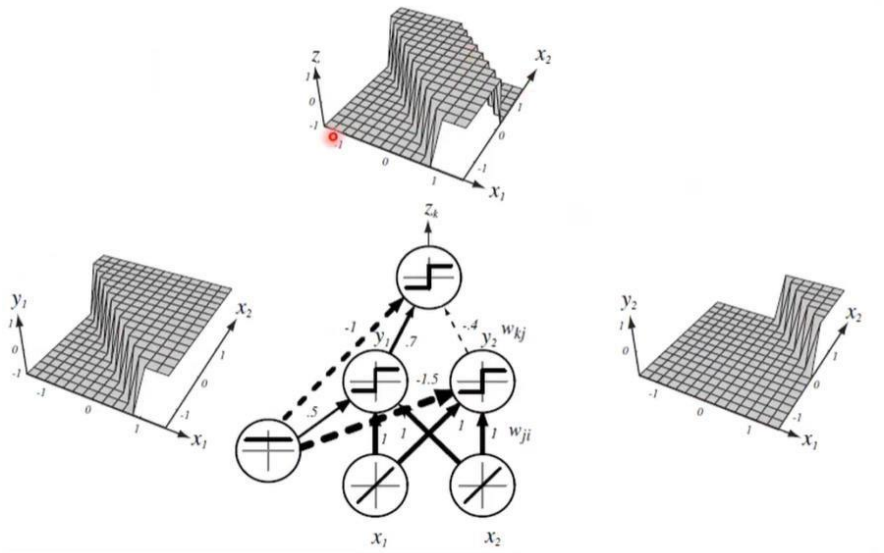
One of the strengths of neural networks is that they are a learning mechanism applicable to both classification and regression problems. Furthermore, they perform attribute selection automatically, since irrelevant or redundant variables can be excluded from the analysis by looking at the coefficients assuming negligible values. However, neural networks require very long times for model training, provide results with modest interpretability that are dependent upon the order in which the examples are analyzed, and also present a lower robustness with respect to data affected by noise.

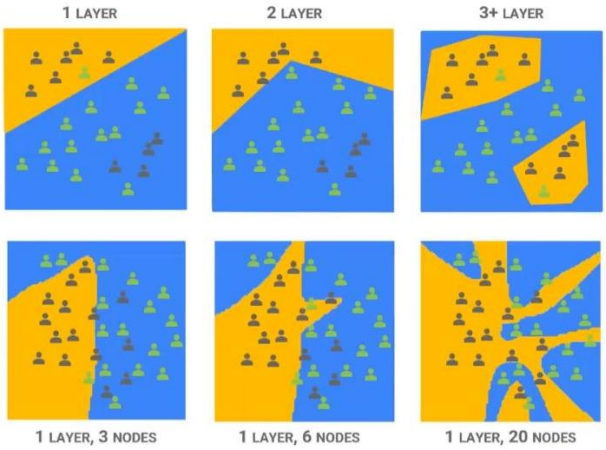
8.2.1. **Nonlinear learning**:

Multi-layer feed-forward neural network with a back-propagation method is useful for nonlinear separation, that can be improved by increasing the number of layers and increasing the number of hidden nodes in each layer.
In the graph above we can see the case of linear learning:



We have a single node and the step function which moves from -1 to 1 (range determined by the activation function $g(\cdot)$). The slope of this direction is determined by w. The position of the riff is determined by the bias b.
If we introduce a single layer with two nodes, we can see different behaviors depending on the position of b and the direction of w. If we combine them, we obtain what we see in the figure in the center at the top:





We obtain a nonlinear separation. It can be improved by increasing the number of nodes in a single layer. If we act on the number of layers, nonlinear separation is improved more and more. The only way to choose the stop point, so the maximum number of layers and nodes that produce the best model is to perform experiments.

8.3. Deep learning:

It is considered deep learning (deep neural network) when the number of layers is more than 4 or 5 layers. In practical terms deep learning is just a subset of machine learning. In fact, deep learning technically is machine learning and functions in a similar way (hence why the terms are sometimes loosely interchanged). However, its capabilities are different.

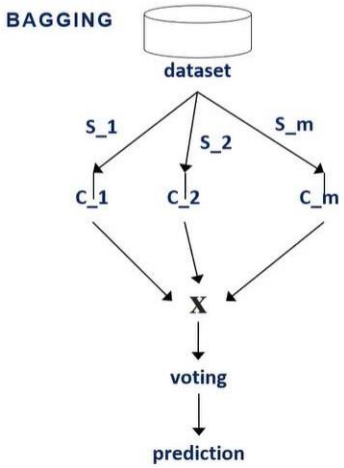
While basic machine learning models do become progressively better at whatever their function is, they still need some guidance. If an AI algorithm returns an inaccurate prediction, then an engineer has to step in and male adjustments. With a deep learning model, an algorithm can determine on its own if a prediction is accurate or not, through its own neural network.

9. ENSEMBLE CLASSIFIERS:

They are aimed at combining predictions obtained from different classifiers. Usually, the classifiers that are combined are not strong, but it appears empirically more convenient to combine many weak (deboli) classifiers. We get many classifiers by introducing some randomization step.

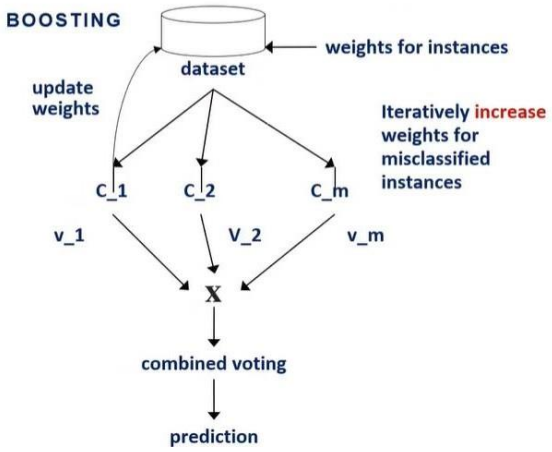
We describe two different approaches: bagging and boosting. We will then describe two models: Adaboost which falls into boosting category and Random Forest, which is neither bagging nor boosting.

Bagging: we start with the entire dataset. The idea is to randomize the selection of subsets of the dataset.



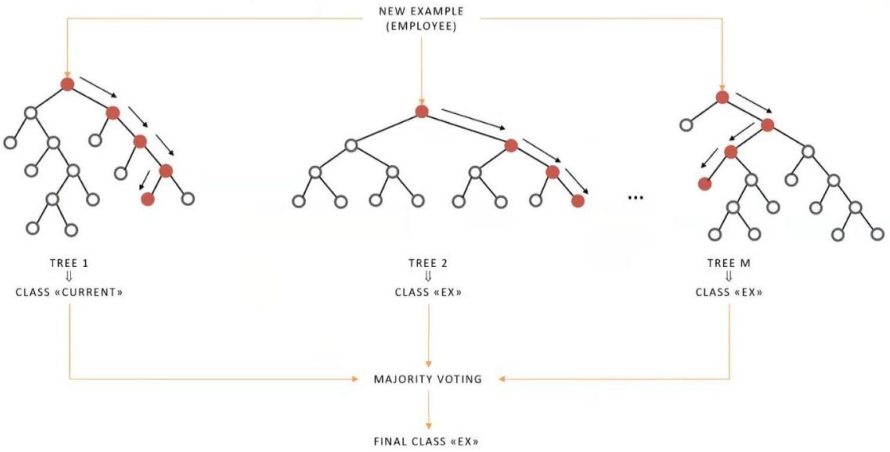
We randomly select the first subset S_1, we randomly select the second substet S_2 and so on up to S_m. In this way, each time we train a classifier: $c_1, c_2, \dots, c_m$. The classifiers can be different from each other, or they can be all the same with the same hyperparameters, but each of them fed with a different dataset (subset of the original dataset). At this point there is a new observation $\boldsymbol{X}$: we inquire all the different classifiers; we have the prediction associated to each of them and we apply the usual majority voting principle.

Boosting: here the logic is to associate to each observation in the dataset a relative weight. Usually, at the beginning of the algorithm all weights are taken equal, so at the beginning all the observations have the same importance but, as soon as the procedure we are about to describe continues, the idea is that some weights are increased and other are decreased. The observations for which are obtaining from the classifier wrong predictions are attributedthe higher way, instead, the observations that were correctly classified by the previous classifiers are given less importance, because they are considered to be easier to be classified. The procedure consists of training the classifiers (c) and deriving an error/performance (v) for each classifier. $v_1, v_2, \dots, v_m$ are so the weights that we attribute to the classifiers and that are proportional to the accuracy. According to the errors we updatethe weights, by iteratively increasing the weights for the misclassified observations and by decreasing it for the well-classified ones. Then we redo the same procedure with the newupdated weights and so on. We then formulate the prediction by performing a combined voting where the combination considers the relative accuracies shown by the different classifiers.



9.1. RANDOM FOREST:

It is based on the idea of training classification trees randomly and then using majority voting. Random forest is more similar to bagging, but the difference is that this method does not take random subset of the observation, but it takes random subsets of the features/variables.



So, each classifier in the ensemble is a classification tree and is generated using a random selection of attributes at each node to determine the split.

There are two possible representation of this idea:

- Forest-RI (Random Input selection): it randomly selects at each node, F attributes (F is fixed) as candidates for the split at the node. There is still correlation between the trees because they are based on the same observations. A more radical approach is represented by the Forest RC.
- Forest-RC (Random linear Combination selection): instead of using the original attributes, this method performs a perturbation which is even greater than the original dataset. It creates new attributes that are random linear combination of the existing attributes. In this way it is possible to reduce the correlation between the individual classifiers/trees.

9.2. ADABOOST:

Initially, all the weights of observations are set the same $(1/d)$. Then, the model generates K classifiers in K steps. At round i, the observations from the original dataset D are sampled (with replacement, so it is possible to have more than one observation in the sampled dataset) to form a training set D_i, of the same size (the cardinality of the sampled dataset is the same of the original dataset). Each observation probability of being selected is based on its weight, so the higher the weight, the more probable is that the observation is sampled. An observation can also be resampled, because of the replacement. At this point, by applying the base classifier (which can be any classifier), we derive a classification model M by using the dataset D_i, and its error rate is calculated. If an observation is miss classified, its weight is increased, otherwise it is decreased.

The error rate $err(X_j)$ is the misclassification error of the observation X_j. It can be either 0 or 1 for a single observation. We can perform it for the entire classifier.

- **Error rate: err(X_j) is the misclassification error of tuple X_j. Classifier M_i error rate is the sum of the weights of the misclassified tuples:**

$$error\ (M_i) = \sum_{j}^{d} w_j \times err\ (\mathbf{X}_j)$$

- **The weight of classifier M_i's vote is** $\log \frac{1-error(M_i)}{error(M_i)}$

REGRESSION

Regression models deal with a dataset consisting of past observations, for which both the value of the explanatory attributes and the value of the continuous numerical target variable are known.

1. **Structure of regression models:**

   The purpose of regression models, also known as explanatory models, is to identify a functional relationship between the target variable and a subset of the remaining attributes contained in the dataset. Thus, their goal is twofold. On one hand, regression models serve to highlight and interpret the dependency of the target variable on the other variables. On the other hand, they are used to predict the future value of the target attribute, based upon the functional relationship identified and the futurevalue of the explanatory attributes. Therefore, the development of a regression model allows knowledge workers to acquire a deeper understanding of the phenomenon analyzed and to evaluate the effects determined on the target variable by different combinations of values assigned to the remaining attributes. This opportunity is of great interest, particularly for analyzing those attributes that are control levers available to decision makers.

   Suppose we are given a dataset D composed of m observations and $n + 1$ attributes, among which we distinguish a *target variable* and n other variables that may play an explanatory role with respect to the target. The target attribute is also called the dependent variable, response or output, while the explanatory variables are also termed independent variables or predictors. The independent variables of each observation may be represented by a vector $\boldsymbol{x}_i$, $i \in M$, in the n-dimensional space $\mathbb{R}^n$, while the target attribute is denoted by y_i. For the sake of conciseness, we will write the m vectors of observations as a matrix $\boldsymbol{X}$ having dimension $m\ x\ n$, and the corresponding m-dimensional vector associated with the target variable as $\boldsymbol{y} = (y_1, y_2, \dots, y_m)$. Finally, let Y be the random variable that represents the target attribute and $X_j, j \in N$, the random variables associated with the explanatory attributes.

   Regression models conjecture the existence of a function $f : \mathbb{R}^n \to \mathbb{R}$ that expresses the relationship between the dependent variable Y and the n explanatory variables X_j:

$$Y = f(X_1, X_2, \dots, X_n)$$

   In general, the process of identifying the function f, called *hypothesis*, can be divided into two sequential phases. First, a choice is made of an adequate class $\mathcal{F}$ of hypotheses which must fulfill two conflicting requirements clearly and rigorously defined within statistical learning theory. On the one hand, the class must be broad enough to allow the identification of an accurate relationship between the target and the independent attributes, in order to guarantee small errors in explaining past data. On the other hand, it must be narrow enough to guarantee good generalization capability in predicting the target variable of new future observations. Considering these conflicting needs, the most popular classes of hypotheses consist of simple and parametric functional relationships of linear, quadratic, logarithmic and exponential nature. In the second phase, once the class of hypotheses $\mathcal{F}$ has been established, the value of the parameters defining the specific function f within the class $\mathcal{F}$ is determined through the solution of an optimization problem appropriately formulated.

   In order to achieve greater robustness in a regression model, it is preferable that the functional relationship between the dependent variable and the independent ones be of a *causal* nature, that is, that it express a cause–effect nexus, where the independent variables clearly play the causal role and the dependent variable the effect role. Indeed, if it is possible to find a logical and causal explanation of the relationship represented in mathematical form by the function f, the interpretation of the model is more convincing and its use for predictive purposes more sound, as long as the causal relationship continues to hold.

A model that is based on a functional relationship between variables devoid of plausible causal nexus is said to be spurious (impuro). However, the development and use of a spurious model should not be ruled out, provided that the model satisfies the significance requirements.
In the regression models described in this chapter the class of hypotheses $\mathcal{F}$ will consist of linear functions, which lead to the study of linear regression models. In other words, we assume that the functional relationship f between the dependent variable and the independent variables is linear.
The purpose of an explanatory model is to express a simple and basic relationship between the dependent variable and the independent variables that is able to approximate past data but also to produce a good generalization on future observations. As stated earlier, both theoretical results and empirical evidence suggest that a good trade-off between these opposite requirements is afforded by moderately broad classes of hypotheses. The hypothesis space should be simple:

2. Simple linear regression:

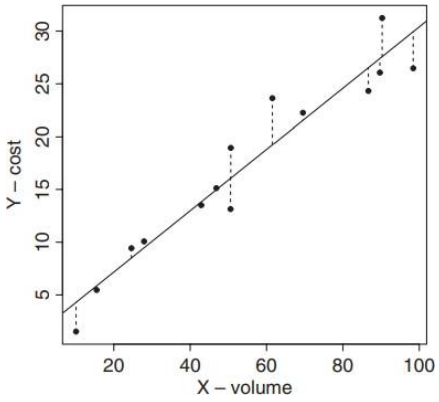
Linear regression models represent the most widely known family of regression models and are based on a class of hypotheses consisting of linear functions. As a consequence, the functional relation $Y = f(X_1, X_2, \dots, X_n)$ reduces to:

$$Y = w_1X_1 + w_2X_2 + \cdots + w_nX_n + b = \sum_{j=1}^{n} w_jX_j + b$$

If there is only a single independent variable $X = X_1$ $(n = 1)$, the linear regression model is called *simple*, and we have:

$$Y = wX + b$$

For simple regression models the dataset actually reduces to m pairs of values (x_i, y_i), $i \in M$, which are realizations of the random variables X and Y. In a first stage of analysis, the pairs of values can be graphically represented by a scatter plot to develop a visual perception of any possible relationship existing between the random variables X and Y.



In general, it is unlikely that m pairs of observations (x_i, y_i), $i \in M$, will fall exactly on a straight line on the plane. As a consequence, instead of the expression $Y = w\,X + b$, it is more realistic to suppose that an approximate relationship between Y and X exists, as expressed by the *probabilistic model*:

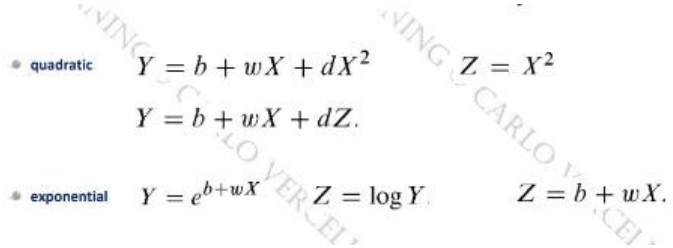
$$Y = wX + b + \varepsilon$$

In which ε is a random variable, referred to as *error*, which indicates the discrepancy between the response Y and the prediction $f(X) = wX + b$. The variable ε must satisfy some hypothesis of stochastic nature.

Having established that the relationship of interest is an approximate one, it is necessary to specify a criterion by which it will be possible to identify in the space $\mathcal{F}$ of hypotheses – that is, among all straight lines in the plane, the line that best represents the relationship between Y and X.

Not all the relationship are linear, but we are able to transform nonlinear relationship in to linear models.

Per esempio se ho una Y quadratica o esponenziale vedo come trasformarla con il metodo di sostituzione della variabile Z:



2.1. Calculating the regression line:

Let us suppose that we draw in the figure above one of the infinite number of lines that can describe the relationship between Y and X. The identification of the regression line then reduces to the specification of its parameters, represented by the slope coefficient w and the intercept b.

In the figure the dotted vertical segments highlight the residuals e_i, which are the signed lengths of the segments that connect the ordinates y_i of the observed points to the ordinates $f(x_i)$ of the points on the line at abscissas x_i:

$$e_i = y_i - f(x_i) = y_i - w x_i - b, \qquad i \in M$$

If we assume that the coefficients w and b of the straight line in the figure correspond to the ideal values expressed by the probabilistic model $Y = wX + b + \varepsilon$, the residuals can be thought of as m realizations of the random variable ε. For this reason, in a slight abuse of language, the random variable ε is also referred to as the *residual variable*.

This intuitive perception, based on the analysis of the scatter plot, is insufficient for assessing the plausibility of the model in a rigorous and systematic way. So, we will describe a number of diagnostics aimed at assessing the significance of a linear regression model.

Among the possible criteria for identifying the regression coefficients w and b, the most widely known and intuitive one consists of minimizing the *sum of squared errors*, expressed by the function:

$$SSE = \sum_{i=1}^{m} e_i^2 = \sum_{i=1}^{m} [y_i - f(x_i)]^2 = \sum_{i=1}^{m} [y_i - wx_i - b]^2$$

Notice that SSE is a convex quadratic function of the variables w and b, possessing a single minimum point which can be determined by requiring that the partial derivatives with respect to the regression coefficients be equal to zero:

$$\frac{\partial \text{SSE}}{\partial b} = -2\sum_{i=1}^{m}[y_i - wx_i - b] = 0$$

$$\frac{\partial \text{SSE}}{\partial w} = -2\sum_{i=1}^{m} x_i[y_i - wx_i - b] = 0$$

In order to derive the optimal values of w and b, we have to solve the previous system of two equations in two variables, which can be formulated in the following normal equation form:

$$\begin{pmatrix} m & \sum_{i=1}^{m} x_i \\ \sum_{i=1}^{m} x_i & \sum_{i=1}^{m} x_i^2 \end{pmatrix} \begin{pmatrix} b \\ w \end{pmatrix} = \begin{pmatrix} \sum_{i=1}^{m} y_i \\ \sum_{i=1}^{m} x_i y_i \end{pmatrix}$$

It is possible to obtain the solution of the normal equation in analytic form, by expressing the regression coefficients through the equalities:

$$\hat{w} = \frac{\sigma_{xy}}{\sigma_{xx}}$$

$$\hat{b} = \bar{\mu}_y - \hat{w}\bar{\mu}_x$$

Where:

$$\bar{\mu}_x = \frac{\sum_{i=1}^{m} x_i}{m}, \qquad \bar{\mu}_y = \frac{\sum_{i=1}^{m} y_i}{m}$$

represent the sample means of the predictor X and the target Y, respectively, while

$$\sigma_{xy} = \frac{1}{m-2}\sum_{i=1}^{m}(x_i - \bar{\mu}_x)(y_i - \bar{\mu}_y)$$

$$\sigma_{xx} = \frac{1}{m-1}\sum_{i=1}^{m}(x_i - \bar{\mu}_x)^2$$

$$\sigma_{yy} = \frac{1}{m-1}\sum_{i=1}^{m}(y_i - \bar{\mu}_y)^2$$

express the sample covariance of x_i and y_i, the sample variance of x_i, and the sample variance of y_i, respectively, except for the multiplicative factors $1/(m-2)$ and $1/(m-1)$.

Notice that the values $\hat{w}$ and $\hat{b}$ of the coefficients, obtained by minimizing the function SSE, actually represent pointwise estimates of the true regression coefficients w and b that appear in the probabilistic model. Such estimators have indeed been derived by using the sample consisting of the m available observations.

The simple linear regression model, derived by minimizing the sum of squared errors, turns out to be:

$$\hat{Y} = \hat{f}(X) = \hat{b} + \hat{w}X = \bar{\mu}_y + \frac{\sigma_{xy}}{\sigma_{xx}}(X - \bar{\mu}_x)$$

In those regression problems where the observations x_i, $i \in M$, assume both positive and negative values, falling therefore on both sides of the value 0 along the x-axis, it may be appropriate to impose the condition $b = 0$, which geometrically corresponds to forcing the straight line to cross the origin. Hence, the expressions for the regression coefficients become:

$$\hat{w} = \frac{\sum_{i=1}^{m} x_i y_i}{\sum_{i=1}^{m} x_i^2}$$

$$\hat{b} = b = 0$$

again by minimizing the function SSE, this time only with respect to the coefficient w. However, the condition $b = 0$ compromises one of the two degrees of freedom available in choosing the linear

approximation for the data, hence generally worsening the accuracy of the model. In light of this remark, one should resort to this expedient only in those cases in which the condition $b = 0$ is considered mandatory.

3. Multiple linear regression:

Having considered the description of simple linear regression models, we will now consider the general case, referred to as *multiple linear regression*, whereby the number n of independent variables is greater than one.

Suppose that we are given m observations $\boldsymbol{x}_i$, comprising n-dimensional vectors which represent realizations of the independent variables X_j, and m values y_i of the dependent variable Y. It is further assumed that a linear probabilistic relationship exists between the response Y and the explanatory variables X_j, $j \in N$, expressed as:

$$Y = w_1X_1 + w_2X_2 + \cdots + w_nX_n + b + \varepsilon$$

It is possible to derive a simple interpretation of the slope coefficients w_j appearing in the model expression above. Indeed, if a single explanatory variable X_j is increased by one, while all the other explanatory attributes remain unchanged, the response variable Y is affected by a variation in value equal to w_j. Therefore, it can be claimed that the regression coefficient w_j expresses the marginal effectof the variable X_j on the target, conditioned on the current value of the remaining predictive variables. Obviously, this information may prove very useful for the interpretation of the regression model and can provide a measure of the relative importance to the response variable of the various predictors. It is, however, necessary to keep in mind a few conditions that limit the validity of an interpretation. First, the value of each coefficient depends on the whole set of explanatory variables. As a consequence, the removal of some variables or the introduction of new predictors implies a change in all the regression coefficients and therefore can also alter their relative ranking. Furthermore, the scale of the values assumed by a predictor, and therefore also the measurement unit in which the values are expressed, influence the value of the corresponding regression coefficient. For this reason, it might be useful to perform a preliminary standardization of all the independent variables before proceeding with the development of a regression model.

3.1. Calculating the regression coefficients:

For multiple linear regression models, the $n + 1$ parameters w_j and b that identify the hyperplane expressing the linear relationship in the space $\mathbb{R}^n$ can be derived by the least squares principle, through the minimization of the sum of squared errors. If $\boldsymbol{e} = (e_1, e_2, \dots, e_m)$ denotes the vector of the residuals associated with the m pairs of observations $(\boldsymbol{x}_i, y_i)$, the n equalities:

$$y_i = w_1x_{i1} + w_2x_{i2} + \cdots + w_nx_{in} + b + e_i, \qquad i \in M$$

must hold. Assume that the matrix $\boldsymbol{X}$ associated with the original dataset has been modified by placing to the left an m-dimensional column vector with all components equal to 1 and denote by $\boldsymbol{w} = (b, w_1, w_2, \dots, w_n)$ the vector of the slope regression coefficients, extended in turn to the left with the intercept b. We can therefore reformulate the equalities above in matrix notation as:

$$\boldsymbol{y} = \boldsymbol{X}\boldsymbol{w} + \boldsymbol{e}$$

Hence, the sum of squared errors can be expressed as:

$$SSE = \sum_{i=1}^{m} e^2_i = ||\boldsymbol{e}||^2 = \sum_{i=1}^{m} (y_i - \boldsymbol{w}'^{x_i})^2 = (\boldsymbol{y} - \boldsymbol{X}\boldsymbol{w})'(\boldsymbol{y} - \boldsymbol{X}\boldsymbol{w})$$

As in the case of simple regression, SSE is a convex quadratic function possessing a unique minimum point that can be calculated analytically by setting equal to zero the partial derivatives evaluated with respect to the components of the vector $\boldsymbol{w}$:

$$\frac{\partial \text{SSE}}{\partial \boldsymbol{w}} = -2\boldsymbol{X}'\boldsymbol{y} + 2\boldsymbol{X}'\boldsymbol{X}\boldsymbol{w} = \boldsymbol{0}$$

Again, we obtain the normal equation, expressed in this case in matrix form:

$$\boldsymbol{X}'\boldsymbol{X}\boldsymbol{w} = \boldsymbol{X}'\boldsymbol{y}$$

The solution of this equation uniquely determines the value of the $n + 1$ regression coefficients:

$$\hat{\boldsymbol{w}} = (\boldsymbol{X}'\boldsymbol{X})^{-1}\boldsymbol{X}'\boldsymbol{y}$$

provided that the matrix $\boldsymbol{X}'\boldsymbol{X}$ is invertible.
Notice that for multiple regression, as for simple regression, the solution $\hat{\boldsymbol{w}}$, obtained through the minimization of the function SSE, represents a pointwise estimate of the regression coefficient $\boldsymbol{w}$. It can be shown that the vector $\hat{\boldsymbol{w}}$ coincides with the maximum likelihood estimate of the vector $\boldsymbol{w}$.
We can easily derive the values of the response variable Y predicted by the model, also termed *fitted values*, as:

$$\hat{\boldsymbol{y}} = \boldsymbol{X}\,\hat{\boldsymbol{w}} = (\boldsymbol{X}(\boldsymbol{X}'\boldsymbol{X})^{-1}\boldsymbol{X}')\boldsymbol{y}$$

where $\boldsymbol{H} = \boldsymbol{X}(\boldsymbol{X}'\boldsymbol{X})^{-1}\boldsymbol{X}'$ is called the *hat matrix*, which has the properties of being symmetric and idempotent. The matrix $\boldsymbol{H}$ allows the residual to be expressed as:

$$\boldsymbol{e} = \boldsymbol{y} - \hat{y} = (\boldsymbol{I} - \boldsymbol{H})\boldsymbol{y}$$

3.2. Assumptions on the residuals:

The random variable ε, which represents the errors and appears in the probabilistic models $Y = wX + b + \varepsilon$ and $Y = w_1X_1 + w_2X_2 + \cdots + w_nX_n + b + \varepsilon$ must satisfy a number of stochastic assumptions, partly dependent on the criterion used for calculating the coefficients of the regression model.
In general, the variable ε should behave as an intrinsically random noise, devoid of effects that can be observed in the response variable Y. Alternatively, ε can also be seen as the result of the action of countless independent variables excluded from the model, each of them having a negligible effecton the target attribute Y.
In particular, if the regression coefficients are determined by minimizing the sum of squared errors SSE, the random variable ε must follow a normal distribution with 0 mean and standard deviation σ. This assumption translates into the following expressions, relative to the expected value and the variance of the residuals conditioned on the value of the observations:

$$E(\varepsilon_i|\boldsymbol{x}_i) = 0$$

$$var(\varepsilon_i|\boldsymbol{x}_i) = \sigma^2$$

Moreover, the residuals ε_i and ε_k, corresponding to two distinct observations $\boldsymbol{x}_i$ and $\boldsymbol{x}_k$, should be independent for any choice of i and k.

Notice that the standard deviation σ should ideally remain constant ad the values of each single component of the observations vary. This is referred to a *homoscedasticity* in the residuals. Conversely, if the value of the standard deviation σ is not constant – that is, the model is not steady as the variables X_j vary along their axes – we have heteroscedasticity in the residuals. This situation compromises the significance of the regression model and must be corrected, usually by resorting to a scaling of the variables.

A regression model is more accurate, and therefore more useful from an application standpoint, as the deviation σ is close to 0. It is therefore interesting to calculate an unbiased pointwise estimator of the deviation σ^2, given by the expression:

$$\bar{\sigma}^2 = \frac{SSE}{m-n-1} = \frac{\sum_{i=1}^{m}(y_i - \boldsymbol{w}'\boldsymbol{x}_i)^2}{m-n-1} = \frac{\boldsymbol{y}'(\boldsymbol{I}-\boldsymbol{H})\boldsymbol{y}}{m-n-1}$$

in which the denominator takes into account the degrees of freedom of the model, which are equal to the number m of observations less the total number of variables, one of which is dependent, and the remaining n are independent. We have also made use of the symmetry and the idempotency of the matrix $\boldsymbol{H}$, and therefore of the matrix $(\boldsymbol{I} - \boldsymbol{H})$. The estimator $\bar{\sigma}$ of the standard deviation σ is referred to as the standard error and plays a critical role in the determination of the accuracy of a regression model, since it determines the dispersion of the data around the prediction line.

3.3. Ridge regression:

The computation of the inverse matrix $(\boldsymbol{X}'\boldsymbol{X})^{-1}$, usually based on efficient decomposition algorithms, become impossible if $\boldsymbol{X}'\boldsymbol{X}$ is a singular matrix and turns out to be critical if the number m of observations is insufficient or if some explanatory variables are dependent on each other, thus causing a phenomenon called *multicollinearity*.

In such cases, although the matrix $\boldsymbol{X}'\boldsymbol{X}$ is in principle invertible, the actual identification of the regression coefficients represents an *ill-conditioned problem*, in the sense that small changes in the input data may cause significant changes in the estimate of the regression parameters. To put it another way, the amount of information contained in the data proves insufficient to create a robust and accurate regression model.

A possible remedy has been proposed within the theory of regularization and involves limiting the space $\mathcal{F}$ of hypotheses, that is, those functions f intended to represent the relationship among the response variable and the explanatory variables. In general terms, the restriction of the class $\mathcal{F}$ is obtained by limiting the norm of the candidate functions f, or looking for an ideal trade-off between the norm of f and its accuracy on the training data. In particular, instead of obtaining the regression coefficients through the minimization of the sum of squared errors SSE, the method of *ridge regression* relies on the optimization model:

$$\min_{\boldsymbol{w}} RR(\boldsymbol{w}, D) = \min_{\boldsymbol{w}} \lambda ||\boldsymbol{w}||^2 + \sum_{i=1}^{m}(y_i - \boldsymbol{w}'\boldsymbol{x}_i)^2 =$$

$$= \min_{\boldsymbol{w}} \lambda ||\boldsymbol{w}||^2 + (\boldsymbol{y} - \boldsymbol{X}\boldsymbol{w})'(\boldsymbol{y} - \boldsymbol{X}\boldsymbol{w})$$

in which λ is a positive parameter controlling the relative importance of the regularization term, expressed by the norm of $\boldsymbol{w}$, with respect to the sum of squared errors.

Also in this case, the learning problem is brought back to the solution of an appropriate minimization problem, a fact that confirms the key role played by optimization theory within pattern recognition and machine learning models.

3.4. Lasso regression:

It uses the first order norm in place of the second order norm of the ridge regression. The first order norm is just the sum of the absolute values of the coefficients and not the sum of the squares of the absolute values. It has some disadvantages: the optimization problem becomes more difficult.

$$\min_{\boldsymbol{w}} \boldsymbol{L}R(\boldsymbol{w}, D) = \min_{\boldsymbol{w}} \lambda|\boldsymbol{w}| + \sum_{i=1}^{m}(y_i - \boldsymbol{w}'^{x_i})^2 =$$

$$= \min_{\boldsymbol{w}} \lambda|\boldsymbol{w}| + (\boldsymbol{y} - \boldsymbol{X}\boldsymbol{w})'(\boldsymbol{y} - \boldsymbol{X}\boldsymbol{w})$$

3.5. Generalized linear regression:

It is possible to extend the multiple linear regression model in order to adapt it to a rather broad class of transformations that can be applied to the independent variables. To this end, consider a generalized probabilistic model:

$$Y = \sum_{h} w_h g_h(X_1, X_2, \dots, X_n) + b + \varepsilon$$

where the functions g_h represent any set of bases, such as polynomials, kernels and other groups of nonlinear functions.

In this case, too, the coefficients w_h and b can be determined through the minimization of the sum of squared errors. However, the corresponding function SSE in this formulation is more complex than in the case of linear regression, and the solution of the minimization problem is therefore more difficult. Usually, the solution of the equation above is embodied in the framework of duality theory, through a representation of the regression coefficients $\hat{w}$as a linear combination of the observations.

4. KNN for regression:

In the classification the new target is given depending the majority class of the K neighbors. Here we compute the distances following the metrics, we then group the K closest neighbors around the new value and finally we assign it the value of the average of its K neighbors (in the regression case we use the principle of average and not the majority voting).

5. REGRESSION TREES:

In the classification case we simply have labels/classes and so, the trees method is based on the majority voting principle. In the regression case we have to deal with numbers. So, the regression trees method uses the average principle, so it computes the average of the values of the leaves. To be reasonable, the average approach requires that each leaf presents the smallest variance as possible. If there is a big variance, it can bring us to mistakes and the average approach would not be reliable. So, in this case, we have to choose a splitting rule that minimizes the variance.

$$S = \sum_{c \in leaves(T)} \sum_{i \in C} (y_i - m_c)$$

where m_c is the average of the target, and where the prediction is:

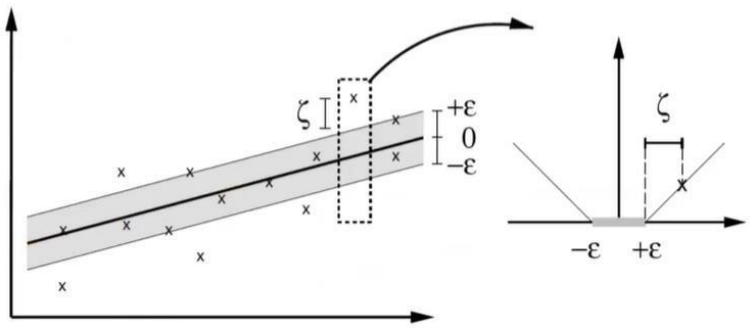
$$m_c = \frac{1}{n_c} \sum_{i \in C} y_i$$

So, to resume, the regression tree method has two main characteristics:
- new splitting rule that tends to minimize the variance.
- New prediction method based on average principle.

It is also possible to perform Random Forest for regression: it consists of the combination of regression trees. At each node we randomly select the candidate attributes in order to generate a randomregression tree.

6. SUPPORT VECTOR REGRESSION:

The idea is the following:



We remember that in SVM, we were considering separated hyperplanes, so the aim of a hyperplane was to separate the positives from the negatives. In the regression case, the idea is to consider a fitting hyperplane, so the aim is to fit properly the points (separating in classification / fitting in regression). Since it's very unlikely that all the points lie on the same hyperplane, the idea is to don't consider an error the difference between a point and a fitting hyperplane if the provided point lies within a band of variation delimited by two parallel canonical hyperplanes, whose distance from the fitting hyperplane is $\pm\varepsilon$. Only the points falling outside the admissible band of variation are considered to be errors.
The small graph on the right shows the structure of the error, which is made of the ramp functions, one to the right and one symmetrical to the left. The error inside the band is zero, while if we move from the vertical axis, to the right or to the left, the error is proportional to ξ, which is the distance, along the vertical axis, between the point and the corresponding closer canonical hyperplane.
The implementation of this model is expressed by:

$$\frac{1}{2}||w||^2 + C\sum_{i=1}^{t}(\xi_i + \xi_i^*) \qquad (*)$$

$$\text{s. to} \begin{cases} y_i - \langle w, x_i \rangle - b \leq \varepsilon + \xi_i \\ \langle w, x_i \rangle + b - y_i \leq \varepsilon + \xi_i^* \\ \xi_i, \xi_i^* \geq 0 \end{cases}$$

$$|\xi|_\varepsilon := \begin{cases} 0 & \text{if } |\xi| \leq \varepsilon \\ |\xi| - \varepsilon & \text{otherwise} \end{cases}$$

Where $\langle w, x_i \rangle$ means transposed. and ξ_i and ξ_i^* are the corrections.

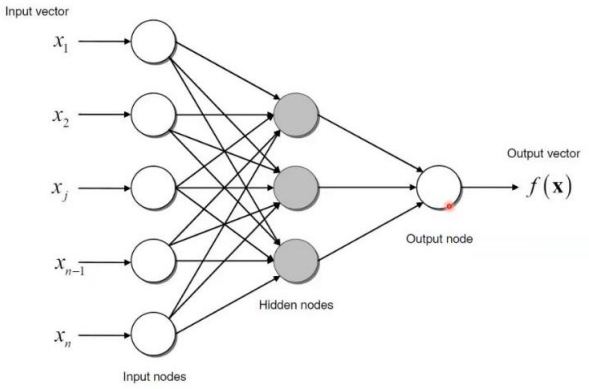
The term $\frac{1}{2}||w||^2$ of $(*)$ expresses generalization and the term $C\sum_{i=1}^{t}(\xi_i + \xi_i^*)$ of $(*)$ expresses accuracy.

The hyperparameters of this model are ξ and ε, which are available to user's choice.
This method can be extended to non-linear fitting, by using kernels.

7. NEURAL NETWORKS FOR REGRESSION:

The modification required is extremely simple in this case. It remains all the same as for the classification case, the only difference lies in the activation function at the output node. When we are dealing with positive-negative binary classification, we took as the activation function the following expression preceded by $sgn(\cdot)$:

$$g(\boldsymbol{w}'\boldsymbol{x} - \vartheta) = w_1x_1 + w_2x_2 + \cdots + w_nx_n - \vartheta = \boldsymbol{w}'\boldsymbol{x} - \vartheta$$



The $sgn(\cdot)$ has the effect of telling us if the point is falling above or below the separating hyperplane. In this case we do not introduce the $sgn(\cdot)$ because we are dealing with a fitting hyperplane.
The expression above represents the linear activation function just for the output node. The linear expression is only for the output node. Before the output node, we achieve non-linearity approximation because we are using many layers and many nodes per layer.

8. Validation of regression models:

There are different diagnostic criteria that enable the quality and the predictive accuracy of a linear regression model to be evaluated:

- normality and independence of the residuals.
- significance of the coefficients.
- analysis of variance.
- coefficient of determination.
- linear correlation coefficient.
- multicollinearity of the independent variables.
- confidence and prediction limits.

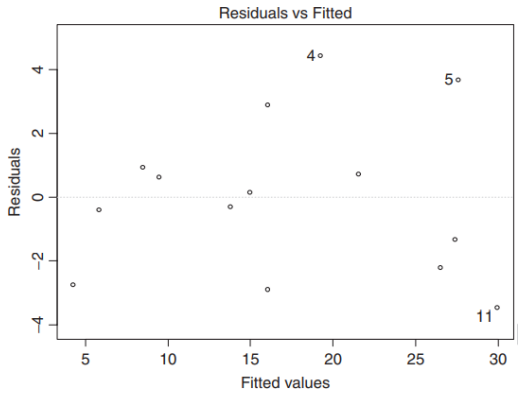
8.1. Normality and independence of the residuals:

The aim of the first diagnostic test is to verify the hypotheses of normality and independence of the residuals. This statistical check is usually performed a posteriori, after the estimate of the regression coefficients $\hat{\boldsymbol{w}}$ and the vector $\boldsymbol{e}$ of the residuals have been computed.

In order to verify whether the assumption that the residuals follow a normal distribution is correct, one of the alternative goodness-of-fit hypothesis tests can be applied, such as the *chi-square* test (is aimed to check if a given set of numbers may reasonably come from a given distribution, in particular the normal one) or the *Kolmogorov–Smirnov* test (it is aimed just to verify normality).

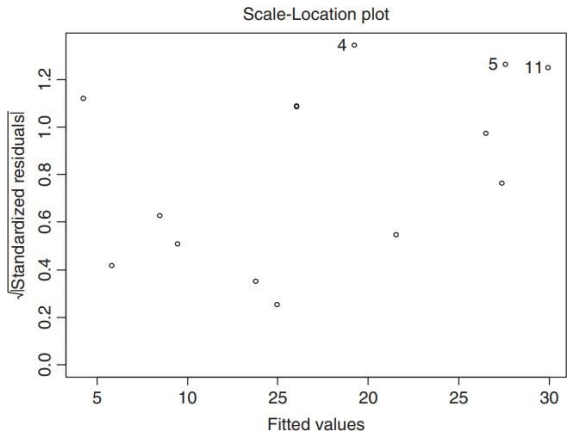
The statistics on the residuals provide further relevant information from which the magnitude of the errors can be evaluated. Furthermore, it is useful to analyze the distribution of the residuals through a series of **graphical representations.**

The first graph, shown in the figure below is a **scatter plot** of the residuals against the fitted values $\hat{\boldsymbol{y}}$. Such a graph can be used to highlight any abnormality or regularity in the profile of residuals.

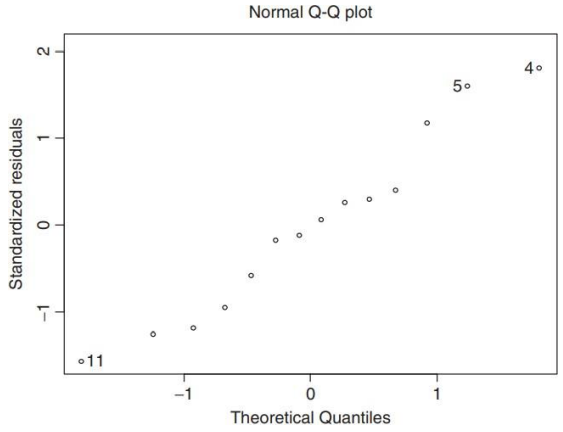


Notice that the plot highlights abnormal residual values by showing the number of the corresponding observation in the dataset. A regular trend in the points on the scatter plot indicates the existence of some explanatory factors not included in the model that might help explain the response variable. Alternatively, a regular pattern in the graph may suggest that the hypothesis f does not adequately express the relationship between the response and the predictors.

The second type of scatter plot, shown in the following figure, shows the square root of the standardized **residuals on the vertical axis.** These are positive and the differences between them are smaller, as a result of using the square root.

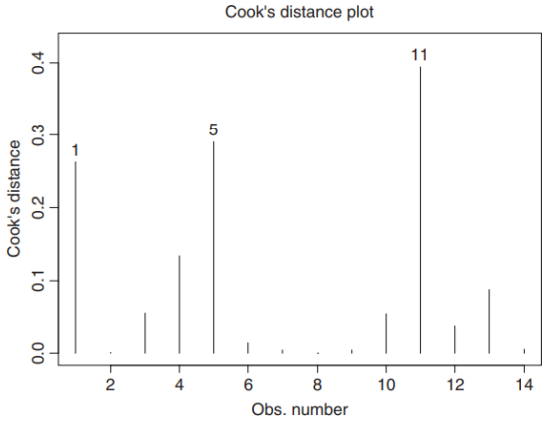


The third graph, shown in the figure below, is a QQ plot of the residuals which provides a way to evaluate visually whether they follow a normal distribution. For an (approximately) normal distribution the points must fall (approximately) on a **straight line.** An S-shaped curve indicates that the tails of the distribution of the residuals are shorter compared to the tails of a normal density. On the other hand, for an upside-down 'S' the tails are longer.



If normality is not verified we should reject the model, revise its.

The following shows for each observation, placed on the horizontal axis, the value of an indicator, called the *Cook's distance*, which highlights the presence of abnormal and large values of the residuals. The Cook's distance of an observation is a measure of its global influence on all the predicted values, obtained by evaluating the effect of removing the observation from the dataset. In particular, an observation with a Cook's distance greater than 1 is considered abnormal with respect to the regression model, and such as to significantly influence the values of the regression coefficients. (NON LO HA FATTO)



8.2. Significance of the coefficients:

As already observed, the coefficients $\hat{\boldsymbol{w}}$ obtained through the minimization of the sum of squared errors are pointwise estimates of the regression coefficients $\boldsymbol{w}$ that appear in the probabilistic model.

The estimation of $\boldsymbol{w}$ can be regarded as a classical problem of inferential statistics, where the regression coefficients $\boldsymbol{w}$ are the unknown parameters of the population, the m observations are a sample extracted from the population and the coefficients $\hat{\boldsymbol{w}}$ are the pointwise estimators with which a confidence interval can be associated.

Observe that the covariance matrix of the estimator $\hat{\boldsymbol{w}}$ is expressed by:

$$\boldsymbol{V} = \mathrm{cov}(\hat{\boldsymbol{w}}) = \bar{\sigma}^2(\boldsymbol{X}'\boldsymbol{X})^{-1}$$

The elements v_{jj} on the main diagonal of the matrix $\boldsymbol{V}$ thus provide the variance for each estimated coefficient, leading to a $100(1-\alpha)\%$ confidence interval of the form:

$$\hat{w}_j \pm t_{\alpha/2}\sqrt{v_{jj}}$$

where $t_{\alpha/2}$ is the $\alpha/2$-order quantile of the Student t-distribution with $m - n - 1$ degrees of freedom (the difference between t-student distribution and normal distribution: al posto di t alfa/2 avevamo z alfa/2. We use t for small dataset, the t have a shape similar to normal distribution, but taller tails and lower in the pick, have a greater dispersion). For a simple linear regression model, the intervals for the coefficients can be expressed as:

$$\hat{w} \pm t_{\alpha/2}\frac{\bar{\sigma}}{\sqrt{\sum_{i=1}^{m}(x_i - \bar{\mu}_x)^2}}$$

$$\hat{b} \pm t_{\alpha/2}\frac{\bar{\sigma}}{\sqrt{n}}\sqrt{1 + \frac{n\bar{\mu}_x}{\sum_{i=1}^{m}(x_i - \bar{\mu}_x)^2}}$$

The confidence intervals for the regression coefficients lead to a third criterion for the validation of a regression model. We can actually conclude that a slope coefficient $\hat{w}_j$ has no significance if its confidence interval contains the value 0. Indeed, if 0 belongs to the confidence interval, the coefficient of the independent variable X_j may be either positive or negative with a non-negligible probability. In other words, the model is unable to establish if the dependent variable Y increases or decreases as the independent variable X_j increases. It is worth noticing that even when the estimator $\hat{w}_j$ appears sufficiently far from 0, it may be the case that the value 0 is included in the confidence interval, provided the corresponding variance v_{jj} of the coefficient $\boldsymbol{w}_j$ is large enough.

predictor	value	standard error	t-value	$\Pr > \lvert t \rvert$
(intercept)	1.36411	1.48944	0.916	0.3780
volume	0.29033	0.02423	11.980	< 0.0001

The table above presents the diagnostics on the significance of the coefficients for a simple regression model (numbers are just an example), obtained with respect to the hypothesis test:

$$H_0: w = 0 \text{ (null hypothesis)}, \quad H_a: w \neq 0 \text{ (alternative hypothesis)}$$

The first two columns show the name of the predictive variable and the estimate of the corresponding regression coefficient. The third column provides the value of the standard deviation v_{jj} for each coefficient. The fourth column indicates the t-value, which expresses the z-index of the estimated coefficient value $\hat{w}$. If the null hypothesis H_0 that the actual value w equals to 0 is true. Finally, the last column indicates the p-value corresponding to the test, which is the probability that the actual value w is greater in absolute value than the estimated value $\hat{w}$, obtained as the area under the tails for a Student t-density with $m - 2$ degrees of freedom.

A t-value greater than 2 is taken to mean that the interval around $\hat{w}$ does not include the value 0 and that therefore w differs from 0, with 95% confidence. Analogously, the same conclusion is reached if the p-value is lower than 0.05.

8.3. Analysis of variance:

The analysis of variance is an important diagnostic for the validation of a regression model. The analysis of variance can be reproduced in a table (numbers are just an example)

predictor	df	sum of sq.	mean sq.	F-value	Pr $> F$
volume	1	933.18	933.18	143.53	< 0.0001
residuals	12	78.02	6.50		
total	13	1011.20	77.79		

The first row in the table, headed *volume*, provides the following information:

- **df**: the number n of degrees of freedom.
- **sum of sq**: The sum of squared differences between the predictions and the sample mean of the response variable, defined as:

$$RSS_{reg} = \sum_{i=1}^{m} (\hat{y}_i - \bar{\mu}_y)^2$$

- **mean sq**: The ratio between the sum of squared prediction differences and the degrees of freedom, given in the two previous columns.
- **F-value**: The value of the F-statistic.
- **Pr>F**: The p-value for the F-statistic.

The second row, headed *residuals*, contains the following information:

- **df:** The number $m - n - 1$ of degrees of freedom.
- **sum of sq:** The sum SSE of squared residuals.
- **mean sq:** The ratio between the sum of squared residuals and the degrees of freedom, given in the two previous columns; this ratio corresponds to the sample variance $\bar{\sigma}^2$ of the residuals.

Finally, the last row in the table, headed *total*,

- **df:** The number $m - 1$ of degrees of freedom.
- **sum of sq:** The sum of squared differences between the actual values and the sample mean for the response variable, given by:

$$RSS_{tot} = \sum_{i=1}^{m} (y_i - \bar{\mu}_y)^2$$

- **mean sq:** The ratio between the sum of squared differences and the degrees of freedom, given in the two previous columns, expressing the sample variance of the response variable Y.

The analysis of variance can be interpreted in intuitive terms. To see this, recall that the aim of a regression model is to explain as far as possible through the predictive variables the variance RSS_{tot} inherent in the dependent variable, leaving aside the explanation of purely random fluctuation represented by the residuals. If this goal is achieved, one may expect the sample variance $\bar{\sigma}^2$ of the residuals, to be significantly smaller than the sample variance of the response variable Y.

If the residuals have a normal distribution, the ratio:

$$F = \frac{RSS_{reg}/n}{SSE/(m-n-1)}$$

Follow an F distribution with n and $m-n-1$ degrees of freedom.

8.4. Coefficient of determination:

The coefficient of determination R^2, also known as *multiple R-squared*, expresses **the proportion of total variance explained by the predictive variables**, and therefore by the regression model. It is defined as:

$$R^2 = \frac{RSS_{reg}}{RSS_{tot}} = \frac{\sum_{i=1}^{m}(y-\bar{\mu}_y)^2}{\sum_{i=1}^{m}(y_i-\bar{\mu}_y)^2}$$

and it therefore lies in the interval [0,1]. If R^2 assumes values close to 1, it may be concluded that the model explains a large portion of the variation inherent in the response variable.

The coefficient of determination tends to increase as the number of observations increases and therefore to overestimate the proportion of variance explained by the model. To avoid this inaccuracy, a corrected coefficient of determination, called *adjusted R-squared*, is defined as:

$$R^2_{adj} = 1-(1-R^2)\frac{m-1}{m-n-1}$$

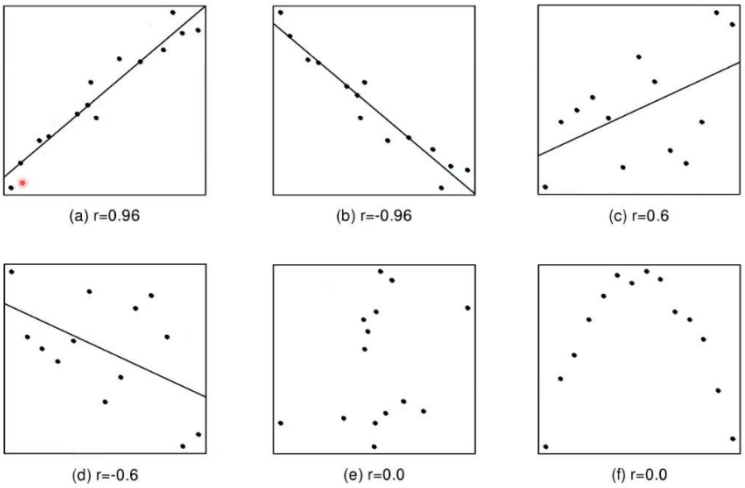
8.5. Coefficient of linear correlation:

In simple regression models it can be observed that the coefficient of determination coincides with the squared coefficient of linear correlation, $R^2 = r^2$, where:

$$r = \mathrm{corr}(\boldsymbol{y}, \boldsymbol{x_i}) = \frac{\sigma_{xy}}{\sqrt{\sigma_{xx}\sigma_{yy}}} = \frac{\sum_{i=1}^{m}(x_i-\bar{\mu}_x)(y_i-\bar{\mu}_y)}{\sqrt{\sum_{i=1}^{m}(x_i-\bar{\mu}_x)^2\sum_{i=1}^{m}(y_i-\bar{\mu}_y)^2}}$$

The value of the linear correlation coefficient r lies in the interval [−1, 1]. In particular, the linear correlation coefficient can be explained in the following way:

- If $r > 0$, then X and Y are concordant. This means that the pairs of observations will tend to lie on a straight line with positive slope, the approximation to a straight line increasing as r gets closer to 1; if $r = 1$ the points will lie exactly on a straight line.
- If $r < 0$, then X and Y are discordant. In this case, the pairs of observations will tend to lie on a straight line with negative slope, the approximation to a straight line increasing as r gets closer to −1; if $r = -1$ the points will lie exactly on a straight line.
- If $r = 0$, or if $r \approx 0$, there is no linear relationship between X and Y; in this case, the pairs of observations may form a totally random pattern, or alternatively may tend to lie on a nonlinear curve.



A- Positive relationship, b- negative relationship, e-cloud of point, f- non linear relationship.

If a model has r-square, determination coefficient larger than another model, should be prefer. The determination coefficient represent from an intuitive point of view the percentage of total variance of your data explained by the model.

8.6. Multicollinearity of the independent variables:

In a multiple linear regression model, predictive variables should not be linearly correlated. If a significant linear correlation exists between two or more independent variables, the model is said to be affected by *multicollinearity*. When there is multicollinearity, the estimate of the regression coefficients is inaccurate and compromises the overall significance of the model. In a situation of multicollinearity, it may even occur that the coefficient of determination is close to 1, while the regression coefficients of the predictors are not significantly different from 0. This represents a flaw in the model, which, however, can be overcome either by selecting a subset of non-collinear variables or by eliminating some variables that are collinear with others.

To assess the extent of multicollinearity, the linear correlation coefficients among all pairs of independent variables should first be calculated. For those correlation coefficients that exceed a given threshold, it may be concluded that the corresponding pairs of variables are linearly correlated, while for the remaining pairs one may deduce that no evidence of linear correlation exists.

However, in this way only pairwise correlations can be detected. In order to highlight the presence of multiple linear relationships among independent variables, it is possible to calculate the variance inflation factor, which is defined for each predictor X_j as:

$$VIF_j = \frac{1}{1 - R_j^2}$$

where R_j^2 is the coefficient of determination for the linear regression model that explains the variable X_j, treated as a response, through the remaining independent variables. Values of VIF_j greater than 5 point to the existence of multicollinearity.

8.7. Confidence and prediction limits:

Prediction and confidence intervals are two other interesting diagnostics for the validation of a regression model. In particular, the purpose of the prediction interval is to contain the expected value $E[Y]$ of the response variable, whereas the confidence interval includes the value Y of the response variable corresponding to a new vector $\boldsymbol{x}$ of values assigned to the independent variables. Let $\hat{y} = \hat{\boldsymbol{w}}\boldsymbol{x}$ be the prediction associated with the new observation $\boldsymbol{x}$. The variance of $\hat{y}$ is given by:

$$\text{var}(\hat{y}) = \boldsymbol{x}'(\boldsymbol{X}'\boldsymbol{X})^{-1}\boldsymbol{x}\bar{\sigma}^2$$

and allows one to derive a confidence interval around $\hat{y}$ within which one can expect to find the future observed value $E[Y]$. In case of a simple linear regression model, an interval at $100(1-\alpha)\%$ confidence is given by:

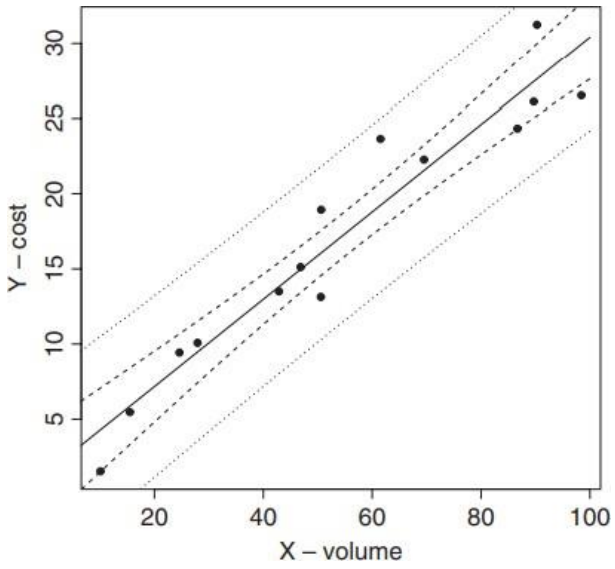
$$\hat{y} \pm t_{\alpha/2}\bar{\sigma}\sqrt{\frac{1}{m} + \frac{x - \bar{\mu}_x}{\sum_{i=1}^{m}(x_i - \bar{\mu}_x)^2}}$$

where x is the value of the new observation of X and $t_{\alpha/2}$ is the percentile of order $100\alpha/2$ of a Student t-distribution with $m - n - 1 = m - 1 - 1 = m - 2$ degrees of freedom. As we can see, the magnitude varies as x varies, and tends to increase in a nonlinear way when x goes away from its average $\bar{\mu}_x$. Intuitively, this means that the uncertainty relative to future predictions increases near the boundaries of the domain of variation of the independent variable.

For simple regression, a similar prediction interval within which we can expect to find the single value Y associated with the new observation x, with a confidence level $(1 - \alpha)\%$, is given by:

$$\hat{y} \pm t_{\alpha/2}\bar{\sigma}\sqrt{1 + \frac{1}{m} + \frac{x - \bar{\mu}_x}{\sum_{i=1}^{m}(x_i - \bar{\mu}_x)^2}}$$

Also in this case the uncertainty increases when x goes away from its average $\bar{\mu}_x$. Obviously, the prediction interval appears wider than the confidence interval, since besides the uncertainty relativeto the prediction it also includes the uncertainty caused by the fluctuation of Y around its average. The following figure shown an example of the confidence interval, which corresponds to the innermost (dashed) lines, and the prediction interval, represented by the outermost (dotted) lines.



(esempio)

9. **Selection of predictive variables:**

The development of a multiple linear regression model requires the selection of a subset of variables that, among all the predictive variables available in the dataset D, are most effective in explaining the response variable. The model obtained using all the predictive variables is rarely significant, since it is often affected by multicollinearity among the independent attributes.

The selection of the predictive variables is a complex activity, particularly when the dataset includes hundreds of explanatory attributes. It is necessary to adopt a wrapper method, whereby a regression model is developed for each subset of predictive variables, then evaluate the significance diagnostics and finally select the best model.

As the number n of attributes increases it clearly becomes impossible to carry out the analysis for all 2^n combinations of the attributes. Therefore, it is usually preferable to automate the procedure for attribute selection by calculating a summary indicator that may direct the algorithms in search for the best subset of predictors. The resulting sequential greedy algorithms are based on myopic criteria of local improvement and do not guarantee the identification of the optimal set of predictors. However, they may prove useful in the preliminary stages of development of a model. Yet, even starting from the suggestions obtained through an automatic search algorithm, a subsequent careful consideration on the part of the analyst is highly recommended in order to identify the most effective predictive variables.

The selection methods are based on three distinct patterns: *forward inclusion*, *backward exclusion* and *exhaustive search.*

- Forward inclusion. Forward inclusion involves adding one predictive variable at a time, starting with an empty set of predictors in the initial stage. At each iteration of the algorithm, the variable that affords the greatest increase in predictive power should be chosen out of all the variables still excluded from the model, provided that the increase exceeds a minimum preset threshold. There are several alternative indicators to express the potential for improvement. One of the most widely applied criteria consists of assessing the reduction in the sum of squared residuals brought about by each variable if introduced into the model. A further indicator is based on the increase in the adjusted coefficient of determination R^2_{adj}A third possible criterion is based on the increase in the value of the F-statistic. It is obviously possible to devise other criteria, perhaps consisting of an appropriate combination of the previous ones or based on totally different indicators.
- Backward exclusion. Backward exclusion is based on an opposite perspective to that of forward inclusion and involves starting with all the predictive variables in the model, and then removing one variable at a time until a stopping condition is met. At each iteration, the variable whose removal causes the largest increase in the explanatory power of the model is selected for exclusion. In this case too, the applied criteria are analogous to those previously described for forward schemes.
- Exhaustive search. Exhaustive search analyzes all subsets of predictors, thus potentially identifying the optimal model, so that it is applicable only when the number of explanatory variables is low, due to the exponential growth in the number of combinations.

ASSOCIATION RULES:

In this chapter we will describe a class of unsupervised learning models that can be used when the dataset of interest does not include a target attribute. These are methods that derive association rules whose aim is to identify regular patterns and recurrences within a large set of transactions. They are fairly simple and intuitive.

1. MOTIVATION AND STRUCTURE OF ASSOCIATION RULES:

In several areas of application, the systematic collection of data gives rise to massive lists of transactions that lend themselves to analysis through association rules in order to identify possible recurrences in the data.

- Market basket analysis: When a customer makes a purchase at a point of sale and receives a receipt for her payment, this transaction is recorded by the information system of the retailer that manages the point of sale. For each transaction recorded, a list of purchased items is stored along with the price, time, and place of the transaction. These transactions are gathered into a massive dataset, at least for large retailers, which can be exploited to perform a data mining analysis aimed at identifying recurrent rules that relate the purchase of a product, or group of products, to the purchase of another product, or group of products. The association rules for market basket analysis can be quite useful for marketing managers in planning promotional initiatives or defining the assortment and location of products on the shelves.
- Web mining: Within web mining analyses it is particularly useful to understand the pattern of navigation paths and the frequency with which combinations of web pages are visited by a given individual during a single session or consecutive sessions. In this case too the list of pages visited during a session is recorded as a transaction, possibly matched with a sequence number and the time of visit. It is therefore interesting to identify regular patterns possibly hidden in the data that allow the association of one or more pages being viewed with visits to other pages. The rules may assume a form such as 'if an individual visits the site timesonline.co.uk then within a week she will also visit the site economist.com with a probability of 0.87'. Association rules of this kind may influence the structure of the links between pages, in order to ease the navigation and to recommend specific navigation paths, or to place advertisement banners and other promotional messages.
- Purchase with a credit card: Association rules are also used to analyze the purchases made by credit card holders in order to direct future promotions. In this case each transaction consists of the purchases and the payments made by a credit card holder. Notice that products and services that can potentially be accessed by the credit card owner are virtually infinite in this situation.
- Fraud detection: In fraud identification, transactions consist of incident reports and applications for compensation for the damage suffered. The existence of special combinations may reveal potentially fraudulent behaviors and therefore justify an in-depth investigation on the part of the insurance company.

Given two propositions Y and Z, which may be true or false, we can state in general terms that a rule is an implication of the type $Y \Rightarrow Z$ with the following meaning: if Y is true, then Z is also true. A rule is called *probabilistic* if the validity of Z is associated with a probability p: if Y is true then Z is also true with probability p.

Rules represent a classical paradigm for knowledge representation, popular because of their simple and intuitive structure, which makes them easily understandable.

To formally represent association rules, it is convenient to introduce some notation. Let:

$$\Theta = \{o_1, o_2, \dots, o_n\}$$

be a set of n objects. **A generic subset $L \subseteq \Theta$ is called *itemset***. An itemset that contains k objects is called *k-itemset*. A *transaction* represents a generic itemset that has been recorded in a database in conjunction

with an activity or cycle of activities. The dataset D is therefore composed of a list of m transactions T_i, each associated with a unique identifier, denoted by t_i.
M is usually very large and k is small in the order of units.

With reference to market basket analysis, the objects represent the items available from a retailer, and each transaction corresponds to the items listed in a sales receipt. Similarly, in web mining the objects represent the pages in the website of interest and each transaction corresponds to the list of pages visited by a navigator during one session.

Notice also that a dataset composed of transactions can be represented by a two-dimensional matrix $\boldsymbol{X}$, where the n objects of the set Θ correspond to the columns of the matrix, the m transactions T_i to the rows, and the generic element of $\boldsymbol{X}$ is defined as:

$$x_{ij} = \begin{cases} 1, & \text{if object } o_j \text{ belongs to transaction } T_i \\ 0, & \text{otherwise} \end{cases}$$

The representation described above could be generalized, assuming that each object o_j appearing in a transaction T_i is associated with a number f_{ij} representing the frequency with which the object appears in the same transaction. In this way it is possible to fully describe multiple sales of a given item during a single transaction, or repeated visits to the same web page during a single navigation session.

Let $L \subseteq \Theta$ be a given set of objects. A transaction T is said to *contain* the set L if the relationship $L \subseteq T$ holds.

The *empirical frequency* $f(L)$ of an itemset L is defined as the number of transactions T_i existing in the dataset D that contain the set L, that is:

$$f(L) = card\{T_i : L \subseteq T_i, i = 1,2, \dots, m\}$$

When dealing with a large sample (i.e. as m increases), the ratio $f(L)/m$ between the empirical frequency and the total number of transactions approximates the probability $Pr(L)$ of occurrence of the itemset L, interpreted as the probability that L is contained in a new transaction T recorded in the database.

2. SINGLE-DIMENSION ASSOCIATION RULES:

Given two itemsets $L \subset O$ and $H \subset O$ such that $L \cap H = \emptyset$ and a transaction T, an association rule is a probabilistic implication denoted by $L \Rightarrow H$ with the following meaning: if L is contained in T, then H is also contained in T with a given probability p, termed the *confidence* of the rule in D and defined as:

$$p = conf\{L \Rightarrow H\} = \frac{f(L \cup H)}{f(L)}$$

The set L is called the *antecedent* or *body* of the rule, and H is the *consequent* or *head*. The confidence of the rule indicates the proportion of transactions containing the set H among those that include the set L, and therefore expresses the inferential reliability of the rule. As the number m of transactions increases, the confidence approximates the conditional probability that H belongs to a transaction T given that L does belong to T. Consequently, a higher confidence corresponds to a greater probability that the itemset H exists in a transaction that also contains the itemset L.
The rule $L \Rightarrow H$ is said to have a *support* s in D, if the proportion of transactions containing both L and H is equal to s, that is, if:

$$s = supp\{L \Rightarrow H\} = \frac{f(L \cup H)}{m}$$

The support of a rule expresses the proportion of transactions that contain both the body and the head of the rule, and therefore measures the frequency with which an antecedent– consequent pair appear together in the transactions of a dataset. A low support suggests that a rule may have occurred occasionally. Low-support rules are usually of little interest to decision makers, and therefore the support is used to discard rules of scant significance. As the size of the dataset increases, the support approximates the probability that both H and L are contained in some future transaction.
The notions of support and confidence lead to a formal treatment of the analysis of association rules. Once a dataset D of m transactions has been assigned and minimum threshold values s_{min} for the support and p_{min} for the confidence have been fixed, all strong association rules should be determined, characterized by a support $s \geq s_{min}$ and by a confidence $p \geq p_{min}$.
However, when dealing with a large dataset, extracting all association rules through a complete enumeration procedure is prohibitive in terms of computation time. The number N_T of possible association rules increases exponentially as the number n of objects increases, according to the formula

$$N_T = 3^n - 2^{n+1} + 1$$

On the other hand, most of these rules are not strong, in the sense that they do not fulfill the requirements of exceeding the predetermined minimum thresholds for support and confidence. An enumerative algorithm that generates all the rules, calculates the support and confidence for each of them, and finally discards unfeasible rules, carries out a lot of useless operations. It is therefore appropriate to devise a method capable of deriving strong rules only, implicitly filtering out the rules that do not fulfill the minimum threshold requirements. The problem of generating strong association rules may be divided into two successive phases: first, the generation of *frequent itemsets*; and second, the generation of *strong rules*.

- Generation of frequent itemsets: As we can infer from definition $s = supp\{L \Rightarrow H\} = \frac{f(L \cup H)}{m}$, the support of a rule only depends on the union of the itemsets L and H, and not on the actual distribution of the objects between the body and the head of the rule.
  Based on this observation, the aim of generating frequent itemsets is to extract all sets of objects whose relative frequency is greater than the assigned minimum support smin. This phase is more burdensome from a computational viewpoint than the subsequent phase of rule generation. Therefore, several algorithms have been proposed for obtaining the frequent itemsets in an efficient manner.
- Generation of strong rules: After all frequent itemsets have been generated, we can proceed to the next phase of identifying strong rules. It is necessary to separate the objects contained in each frequent itemset according to all possible combinations of head and body of the rule and verify if theconfidence of the rule in turn exceeds the minimum threshold p_{min}.

A third measure of the significance of an association rule is the *lift*, defined as:

$$l = lift\{L \Rightarrow H\} = \frac{conf\{L \Rightarrow H\}}{f(H)} = \frac{f(L \cup H)}{f(L)f(H)}$$

Lift values greater than 1 indicate that the rule being considered is more effective than the relative frequency of the head in predicting the probability that the head is contained in some transaction of the dataset. In this case body and head of the rule are positively associated. Conversely, if the lift is less than 1 the rule is less effective than the estimate obtained through the relative frequency of the head. In this case body and head are negatively associated. If the lift is less than 1, the rule that negates the head, expressed as $\{L \Rightarrow (G - H)\}$, is more effective than the original rule, since it has confidence $1 - conf\{L \Rightarrow H\}$ and therefore a lift greater than 1.

3. APRIORI ALGORITHM:

A dataset D of m transactions defined over a set G of n objects may contain up to $2^n - 1$ frequent itemsets, excluding the empty set from the collection of all subsets of G. Consequently, since in real-world applications the cardinality n is at least of the order of a few dozen objects, the number of itemsets increases exponentially and makes it impractical to use an exhaustive generation method based on a complete enumeration of the itemsets to extract the frequent sets.

The *Apriori algorithm* is a more efficient method of extracting strong rules contained in a set of transactions. During the first phase the algorithm generates the frequent itemsets in a systematic way, without exploring the space of all candidates, while in the second phase it extracts the strong rules.

The theoretical assumption on which the Apriori algorithm is based consists of a property called the Apriori principle:

If an itemset is frequent, then all its subsets are also frequent.

If an itemset is not frequent, then each of the itemset containing it must turn out to be not frequent. Once a non-frequent itemset is identified in the course of the algorithm, all the other itemsets (with grater cardinality) containing it are implicitly eliminated and excluded from the analysis.

ESEMPIO?

3.1. Generation of frequent itemsets:

The Apriori algorithm generates frequent itemsets iteratively, starting with the frequent 1-itemsets and then determining the frequent k-itemsets based upon the frequent $(k-1)$-itemsets generated during the previous step. The total number of iterations is equal to $k_{max} + 1$, where k_{max} denotes the maximum cardinality of a frequent itemset in the dataset D.

Procedure 11.1 describes the phase of frequent itemset generation for the Apriori algorithm.

Procedure 11.1 – Apriori algorithm: generation of frequent itemsets

1. The transactions in the dataset are scanned to compute the relative frequency of each object. Objects with a frequency lower than the support threshold $s_{\min}$ are discarded. At the end of this step, the collection of all frequent 1-itemsets has been generated, and the iteration counter is set to $k = 2$.
2. The candidate k-itemsets are iteratively generated starting from the $(k-1)$-itemsets, determined during the previous iteration.
3. The support of each candidate k-itemset is computed by scanning through all transactions included in the dataset.
4. Candidates with a support lower than the threshold $s_{\min}$ are discarded.
5. The algorithm stops if no k-itemset has been generated. Otherwise, set $k = k + 1$ and return to step 2.

3.2. Generation of strong rules:

The second phase, generation of the strong rules, receives as input a list of all frequent itemsets, generated during the first phase, each associated with a relative frequency greater than the minimum threshold s_{min}.

Given a frequent k-itemset L, it is possible to generate $2^k - 2$ candidate association rules to be extracted as strong rules, leaving out the rules that have an empty set as the body or the head. For

each of these the confidence is computed, and this is then compared with the minimum threshold p_{min}. Suppose that the rule $L \Rightarrow H$ has been generated. Observe that the computation of the confidence does not require further scanning of the dataset, since

$$p = conf\{L \Rightarrow H\} = \frac{f(L \cup H)}{f(L)}$$

and the frequencies $f(L)$ and $f(L \cup H)$ are known at the end of the first phase of the algorithm since both L and $L \cup H$ are frequent itemsets.
Procedure 11.2 describes the generation of strong rules for the Apriori algorithm.

Procedure 11.2 – Apriori algorithm: generation of strong rules

1. The list of frequent itemsets generated during the first phase is scanned. If the list is empty, the procedure stops. Otherwise, let B be the next itemset to be considered, which is then removed from the list.
2. The set B of objects is subdivided into two non-empty disjoint subsets L and $H = B - L$, according to all possible combinations.
3. For each candidate rule $L \Rightarrow H$, the confidence is computed as
$$p = \text{conf}\{L \Rightarrow H\} = \frac{f(B)}{f(L)}.$$
4. If $p \geq p_{\min}$ the rule is included into the list of strong rules, otherwise it is discarded.

The number of operations required by the Apriori algorithm grows exponentially as the number n of objects increases.

Several tactics have been suggested, which are briefly described here, to increase the efficiency of the Apriori algorithm; for further information see the references at the end of the chapter. First, it is useful to adopt more advanced data structures for representing the transactions, such as dictionaries and binary trees. Furthermore, it is possible to partition the transactions into disjoint subsets and then separately apply the Apriori algorithm to each partition to identify local frequent itemsets. At a later stage, starting from the latter, the entire set of transactions is considered to obtain global frequent itemsets and the corresponding strong rules. Finally, it is possible to randomly extract a significant sample of transactions and to obtain strong association rules for the extracted sample, with a greater efficiency, even at the expense of possible inaccuracies with respect to the full set of transactions

4. GENERAL ASSOCIATION RULES:

The association rules described in the previous sections are *binary*, *asymmetric*, and *one-dimensional*. They are binary since they refer to the presence or absence of each object in the transactions that make up the dataset. Furthermore, the rules are asymmetric since we implicitly assumed, as is customary in market basket analysis, that the presence of an object in a single transaction, corresponding to an actual purchase, is far more relevant than its absence, corresponding to a non-purchase. Finally, the rules are

one-dimensional because they involve only one logical dimension of data from multi-dimensional data warehouses and data marts.

However, it is possible to identify more general association rules that may be useful across a range of applications.

- Association rules for symmetric binary attributes. There are situations where we might be interested in binary attributes for which the 0 and 1 values are equally relevant. For instance, situations of answers given to questions on the gender of an individual, classified as {male, female}, or on consent to receive advertising material, expressed as {yes, no}, are examples of symmetric binary variables. It is possible to extend the analysis with symmetric attributes to the case of asymmetric binary attributes, in order to apply the algorithms for the generation of association rules described in previous sections. In fact, a simple transformation can be used: for each symmetric binary variable, two asymmetric binary variables are introduced which correspond respectively to the presence of a given value and to the presence of its opposite.
- Association rules for categorical attributes. In other situations, the attributes may be categorical but not binary. There may be questions about the province of residence or the level of education. In this case too it is possible to introduce, for each categorical attribute, a set of asymmetric binary variables, equal in number to the levels of the categorical attribute. Each binary variable takes the value 1 if in the corresponding record the categorical attribute assumes the level associated with the binary variable, otherwise it takes the value 0.
- Association rules for continuous attributes. When dealing with continuous attributes, such as the age or income of those who have filled out a registration form, it is appropriate to proceed in two sequential phases. First, the continuous attribute is transformed into a categorical variable through one of the discretization techniques. Then the discretized attribute is also transformed into asymmetric binary variables, as previously described, in order to proceed with the extraction of strong rules.
- Multidimensional association rules. Data are gathered and stored in data warehouses or data marts, and organized according to several logical dimensions, which then direct the analyses and the visualizations. As a consequence, it is reasonable to require the association rules to involve multiple dimensions. A rule in the context of market basket analysis may appear in the form ‘if a customer buys a digital camera, is 30–40 years old, and has an annual average expenditure of ¤300–500 on electronic equipment, then she will also buy a color printer with probability 0.78’. This is a three-dimensional rule, since the body consists of three dimensions (purchased items, age and expenditure amount), while the head refers to a purchase. Observe that the numerical variables that represent the age and the expenditure amount have previously been discretized, as described in the previous paragraph.
- Multi-level association rules. In some applications, association rules do not allow strong associations to be extracted due to data rarefaction. It may be, for example, that each of the items for sale can be found in too small a proportion of transactions to be included in the frequent itemsets, thus thwarting the search for association rules. However, the objects making up the transactions usually belong to hierarchies of concepts. Items are usually grouped by type and in turn by sales department. A possible way to remedy the rarefaction of objects in the transactions is to transfer the analysis to a higher level in the hierarchy of concepts. In this way the number of objects decreases and consequently the number of transactions containing the same object increases. Also the computational complexity significantly decreases, due to the reduced number of objects, so that the algorithms for rule extraction become much more efficient.
- Sequential association rules. Often the transactions are recorded according to a specific temporal sequence. For example, the transactions for a loyalty card holder correspond to the sequence of sale receipts. By the same token, the transactions that gather the navigation paths followed by a given web user are associated with the temporal sequence of the sessions. In situations like these, analysts

are often interested in extracting association rules that take into account temporal dependencies. The description of the algorithms used to extract association rules in the different situations reviewed in this section goes beyond the scope of this book, and therefore readers are referred to the references given below. For now readers should bear in mind that the proposed techniques usually consist of extensions of the Apriori algorithm and its variants.

CLUSTERING:

The second class of models for unsupervised learning is represented by clustering methods. By defining appropriate metrics and the induced notions of distance and similarity between pairs of observations, the purpose of clustering methods is the identification of homogeneous groups of records called clusters. With respect to the specific distance selected, the observations belonging to each cluster must be close to one another and far from those included in other clusters.

1. CLUSTERING METHODS:

The aim of clustering models is to subdivide the records of a dataset into homogeneous groups of observations, called clusters, so that observations belonging to one group are similar to one another and dissimilar from observations included in other groups.

Grouping objects by affinity is a typical reasoning pattern applied by the human brain. Also for this reason clustering models have long been used in various disciplines, such as social sciences, biology, astronomy, statistics, image recognition, processing of digital information, marketing and data mining. Clustering models are useful for various purposes. In some applications, the clusters generated may provide a meaningful interpretation of the phenomenon of interest. Furthermore, subdivision into clusters may be the preliminary phase of a data mining project that will be followed by the application of other methodologies within each cluster. Finally, grouping into clusters may prove useful in the course of exploratory data analysis to highlight outliers and to identify an observation (selecting landmarks) that might represent on its own an entire cluster, in order to reduce the size of the dataset. Clustering methods must fulfill a few general requirements, as indicated below.

- Flexibility: for numerical and categorical attributes. Some clustering methods can be applied to numerical attributes only, for which it is possible to use the Euclidean metrics to calculate the distances between observations. However, a flexible clustering algorithm should also be able to analyze datasets containing categorical attributes. Algorithms based on the Euclidean metrics tend to generate spherical clusters and have difficulty inidentifying more complex geometrical forms.
- Robustness: The robustness of an algorithm manifests itself through the stability of the clusters (noise) generated with respect to small changes in the values of the attributes of each observation. This property ensures that the given clustering method is basically unaffected by the noise possibly existing in the data. Moreover, the clusters generated must be stable with respect to the order of appearance of the observations in the dataset
- Efficiency: small computing time. In some applications the number of observations is quite large and therefore clustering algorithms must generate clusters efficiently in order to guarantee reasonable computing times for large problems. In the case of massive datasets, one may also resort to the extraction of samples of reduced size in order to generate clusters more efficiently. However, this approach inevitably impliesa lower robustness for the clusters so generated. Clustering algorithms must also prove efficient with respect to the number of attributes existing in the dataset.

1.1. Taxonomy of clustering methods:

Clustering methods can be classified into a few main types based on the logic used for deriving the clusters: *partition methods*, *hierarchical methods*, *density-based methods,* and *grid methods*.

➔ Partition methods: Partition methods develop a subdivision of the given dataset into a predetermined number K of non-empty subsets. They are suited to obtaining groupings of a spherical or at most convex shape and can be applied to datasets of small or medium size.

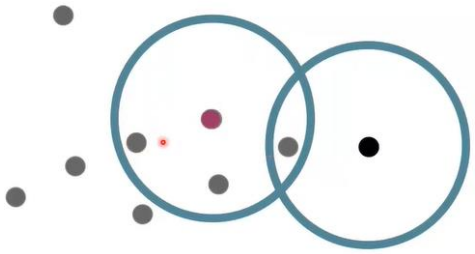
➔ Hierarchical methods: Hierarchical methods carry out multiple subdivisions into subsets, based on a tree structure and characterized by different homogeneity thresholds within each cluster and inhomogeneity thresholds between distinct clusters. Unlike partition methods, hierarchical algorithms do not require the number of clusters to be predetermined.

➔ Density-based methods: look at the number of points lying within the neigborhood of each point. Whereas the two previous classes of algorithms are founded on the notion of distance between observations and between clusters, density-based methods derive clusters from the number of observations locally falling in a neighborhood of each observation. More precisely, for each record belonging to a specific cluster, a neighborhood with a specifieddiameter must contain a number of observations which should not be lower than a minimum threshold value. Density-based methods can identify clusters of non-convex shape and effectively isolate any possible outliers.

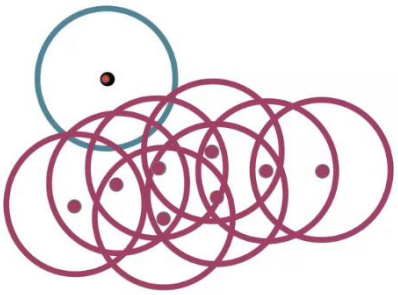
Density based methods

Core points, Reachable points, Noise

It groups together points with many nearby neighbours.
Points that lie alone in low-density regions are marked as outliers.
The final model depends on two parameters: the radius of the neighborhood and the minimum number of points falling in it.
DBSCAN algorithm is deterministic (i.e. generates the same clusters when the same data in the same order are given).



The figure shows that the points that remains in blue circle , after some iterations, is considered an outliers. Ogni punto è il centro di un cerchio, in tutti i cerchi ce ne sono almeno due quindi è facile arrivare a trovare gli outliers.

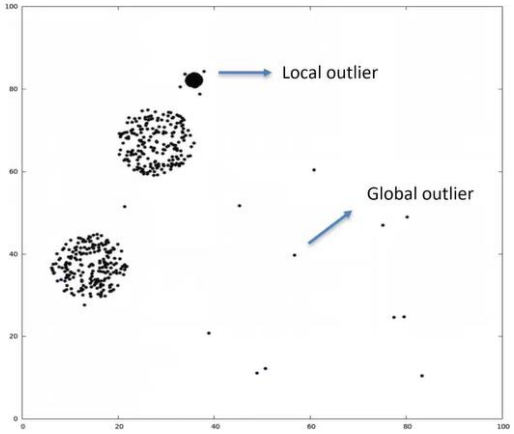


HDBSCAN algorithm (self-adjusting)
Outlier detection resorting to density-based clustering.

It identifies "local outliers", i.e. points that might be different from other parts in their local neighborhood but which are not necessarily global outliers.

Uses a range of distances to separate clusters of varying densities from sparser noise.

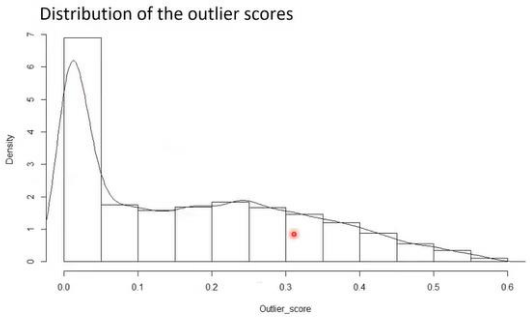
Example of a 2D dataset with global and local outliers



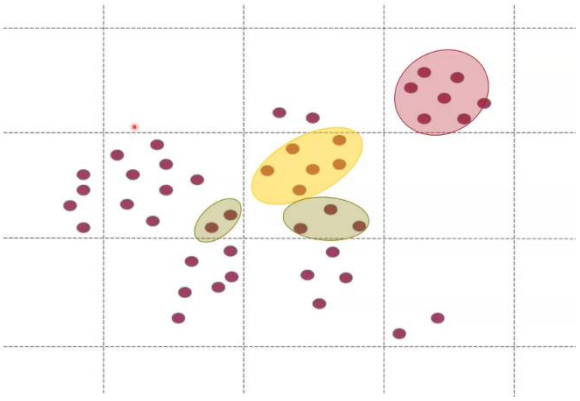
HDBSCAN algorithm
For each point a neighborhood density (ND) measure is computed.

The outlier score measures at what extent the ND of a point is far from the NDs of its neighbors.

The higher the outlier score, the more likely the point is an outlier.



➔ Grid methods: perform a preliminary partition based on a grid structure. Grid methods first derive a discretization of the space of the observations, **obtaining a grid structure consisting of cells.** Subsequent clustering operations are developed with respect to the grid structure and generally achieve reduced computing times, despite a lower accuracy in the clusters generated.



A second distinction is concerned with the methods for assigning the observations to each single cluster. It is possible to include each observation *exclusively* in a single cluster or to place it by *superposition* into multiple clusters. Furthermore, *fuzzy* methods have been developed which assign the observations to the clusters with a weight between 0 (the observation is totally extraneous to the cluster) and 1 (the observation exclusively belongs to the cluster), with the additional condition that the sum of the weights over all clusters be equal to 1.

Finally, a distinction should be made between *complete* clustering methods, which assign each observation to at least one cluster, and partial methods, which may leave some observations outside the clusters. The latter methods are useful for identifying outliers.

Most clustering methods are heuristic in nature, in the sense that they generate a subdivision into clusters of good quality although not necessarily optimal. Actually, an exhaustive method based on a complete enumeration of the possible subdivisions of m observations into K clusters would require

$$\frac{1}{K!}\sum_{h=1}^{K}\binom{K}{h}h^m$$

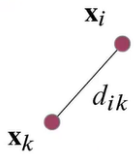
combinations to be examined. Therefore, it would be inapplicable even to small datasets, due to the exponential growth in computing time. In terms of computational complexity, clustering problems actually belong to the class of difficult (NP-hard) problems whenever $K \geq 3$.

1.2. Affinity measures:

Clustering models are usually based on a measure of similarity between observations. In many instances this can be obtained by defining an appropriate notion of distance between each pair of observations.

Given a dataset D, we can represent the m observations by means of n-dimensional vectors of attributes, so as to formally represent the data using a matrix $\boldsymbol{X}$ with m rows and n columns. Let

$$\mathbf{D} = [d_{ik}] = \begin{bmatrix} 0 & d_{12} & \cdots & d_{1,m-1} & d_{1m} \\ & 0 & \cdots & d_{2,m-1} & d_{2m} \\ & & \cdots & \vdots & \vdots \\ & & & 0 & d_{m-1,m} \\ & & & & 0 \end{bmatrix}$$



(DISTANCE MATRIX)

be the symmetric $m \times m$ matrix f distances between pairs of observations, obtained by setting:

$$d_{ik} = \text{dist}(\boldsymbol{x}_i, \boldsymbol{x}_k) = \text{dist}(\boldsymbol{x}_k, \boldsymbol{x}_i), \qquad i, k \in M$$

where $\text{dist}(\boldsymbol{x}_i, \boldsymbol{x}_k)$ denotes the distance between observations $\boldsymbol{x}_i$ and $\boldsymbol{x}_k$.

Notice that it is possible to transform the distance d_{ik} between two observations into a similarity measure s_{ik}, by alternatively using:

$$s_{ik} = \frac{1}{1 + d_{ik}}, \qquad or \qquad s_{ik} = \frac{d_{max} - d_{ik}}{d_{max}}$$

In which $d_{max} = \max_{i,k} d_{ik}$ denotes the maximum distance between the observations in the dataset D.

The definition of an appropriate notion of distance depends on the nature of the attributes that make up the dataset D, which may be numerical, binary, nominal categorical, ordinal categorical or of mixed composition

Numerical attributes:

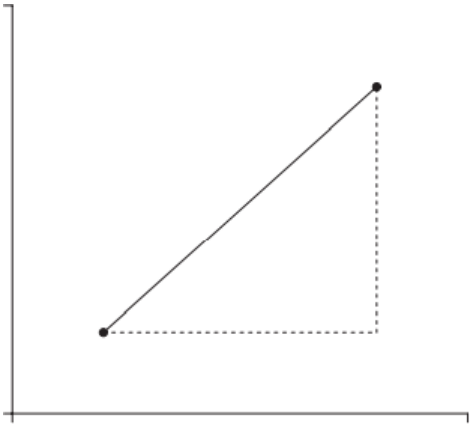
If all n attributes in a dataset are numerical, we may turn to the *Euclidean distance* between the vectors associated with the pair of observations $\boldsymbol{x}_i = (x_{i1}, x_{i2}, \dots, x_{in})$ and $\boldsymbol{x}_k = (x_{k1}, x_{k2}, \dots, x_{kn})$ in n-dimensional space, which is defined as:

$$\text{dist}(\boldsymbol{x}_i, \boldsymbol{x}_k) = \sqrt{\sum_{j=1}^{n} (x_{ij} - x_{kj})^2} = \sqrt{(x_{i1} - x_{k2})^2 + (x_{i2} - x_{k2})^2 + \cdots + (x_{in} - x_{kn})^2}$$

As an alternative we may consider the *Manhattan distance*:

$$\text{dist}(\boldsymbol{x}_i, \boldsymbol{x}_k) = \sum_{j=1}^{n} |x_{ij} - x_{kj}| = |x_{i1} - x_{k1}| + |x_{i2} - x_{k2}| + \cdots + |x_{in} - x_{kn}|$$

which is so called because in order to reach one point from another we have to travel along two sides of a rectangle having the two points as its opposite vertices. The figure below shows the difference between Euclidean and Manhattan metrics, using a two-dimensional example.



Eclidian distance (full line) and Manhattan distance (dottet line)

A third option, which generalizes both the Euclidean and Manhattan metrics, is the *Minkowski distance*, defined as:

$$\text{dist}(\boldsymbol{x}_i, \boldsymbol{x}_k) = \sqrt[q]{\sum_{j=1}^{n} |x_{ij} - x_{kj}|^q} = \sqrt[q]{|x_{i1} - x_{k1}|^q + |x_{i2} - x_{k2}|^q + \cdots + |x_{in} - x_{kn}|^q}$$

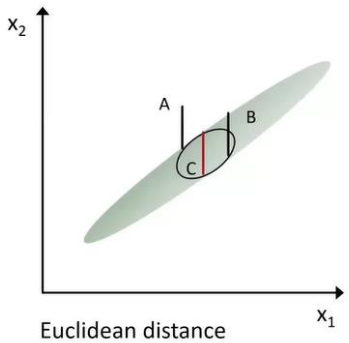
for some positive integer q. The Minkowski distance reduces to the Manhattan distance when $q = 1$, and to the Euclidean distance when $q = 2$.

A further generalization of the Euclidean distance can be obtained through the *Mahalanobis distance*, defined as:

$$\text{dist}(\boldsymbol{x}_i, \boldsymbol{x}_k) = \sqrt{(x_i - x_k)\boldsymbol{V}^{-1}(x_i - x_k)'}$$

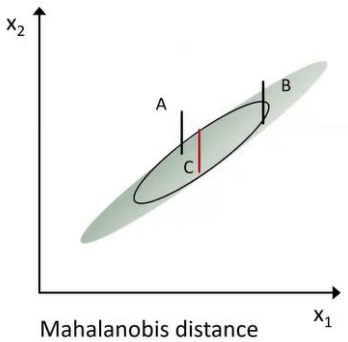
where $\boldsymbol{V}_{ik}^{-1}$ is the inverse of the covariance matrix of the pair of observations $\boldsymbol{x}_i$ and $\boldsymbol{x}_k$. If the observations $\boldsymbol{x}_i$ and $\boldsymbol{x}_k$ are independent, so that the covariance matrix reduces to the identity matrix, the Mahalanobis distance coincides with the Euclidean distance. The Mahalanobis distanceis used when the observations are highly correlated, with different variances and a different range.The expressions described above for computing the distance between pairs of observations are affected by the possible existence of attributes taking on absolute values significantly greater than the others. In extreme situations, a single prevailing attribute could affect the groupings generatedby clustering algorithms.

A few tactics can be adopted in order to avoid such imbalance in the evaluation of distances between observations. On the one hand, one may standardize the values of the numerical attributes so as to obtain new values on a uniform scale, for example in the interval [0, 1] or [−1, 1]. Alternatively, one may assign to each attribute $\boldsymbol{a}_j$ a weight w_j that is inversely proportional to the greatest value assumed by that same attribute, modifying the previous definitions to obtain weighted distances and thus balancing the importance of each attribute.



Euclidean distance

→ the set of points equidistant from a given location is a sphere



Mahalanobis distance

→ stretches the sphere to account for correlation among variables

Binary attributes:

Suppose that a given attribute $\boldsymbol{a}_i = (x_{1j}, x_{2j}, \ldots, x_{mj})$ is binary, so that it assumes only one of the two values 0 or 1. Even if it is possible to formally calculate the difference $x_{ij} - x_{kj}$ for every two observations of the dataset, it is clear that this quantity does not represent a distance that can be meaningfully associated with the metrics defined for numerical attributes, since the values 0 and 1 are purely conventional, and their meanings could be interchanged.

The need thus arises to define an appropriate notion of proximity that can be applied to binary attributes.

Assume for the moment that all n attributes in the dataset D are binary. In order to define a metric, reference should be made to the contingency table associated with the n attributes at two distinct observations $\boldsymbol{x}_i$ and $\boldsymbol{x}_k$, as indicated in the table below:

		observation $\mathbf{x}_k$		
		0	1	total
observation $\mathbf{x}_i$	0	p	q	$p+q$
	1	u	v	$u+v$
	total	$p+u$	$q+v$	n

The value p is the number of binary attributes for which both observations $\boldsymbol{x}_i$ and $\boldsymbol{x}_k$ assume the value 0. Similarly, q is the number of attributes for which observation $\boldsymbol{x}_i$ takes the value 0 and observation $\boldsymbol{x}_k$ takes the value 1. The value u is the number of attributes for which $\boldsymbol{x}_i$ assumes the value 1 and $\boldsymbol{x}_k$ assumes the value 0. Finally, there are v attributes for which the value 1 is assumed by both observations.

Binary attributes can be symmetric or asymmetric. In the first case the presence of the value 0 is as interesting as the presence of the value 1. In the second case we are mostly interested in the presence of the value 1, which can usually be found in a small proportion of the observations.

If all the n binary attributes are symmetric, we can define the degree of similarity between pairs of observations through the *coefficient of similarity*, given by:

$$\text{dist}(\boldsymbol{x}_i, \boldsymbol{x}_k) = \frac{q+u}{p+q+u+v} = \frac{q+u}{n}$$

Assume instead that all n attributes are binary and asymmetric. As stated before, for a pair of asymmetric attributes it is much more interesting to match positives, i.e. records possessing the property relative to each attribute and therefore being coded by the value 1, with respect to matching negatives, representing observations coded by the value 0. For binary variables, the *Jaccard coefficient* is therefore used, defined as (*):

$$\text{dist}(\boldsymbol{x}_i, \boldsymbol{x}_k) = \frac{q+u}{p+u+v}$$

Finally, if the dataset includes both symmetric and asymmetric binary attributes, as well as for simultaneous numerical and binary attributes, the methodologies described in the section on mixed composition attributes should be applied.

Categorical attributes:

- **Nominal categorical attributes:**
  A nominal categorical attribute can be interpreted as a generalization of a symmetric binary attribute, for which the number of distinct values is greater than two. As a consequence, for nominal attributes we can also use an extension of the similarity coefficient, previously defined for symmetric binary variables, using the new expression:

$$\text{dist}(\boldsymbol{x}_i, \boldsymbol{x}_k) = \frac{n - f}{n}$$

  in which f is the number of attributes for which the observations $\boldsymbol{x}_i$ and $\boldsymbol{x}_k$ take the same nominal value. Is 1 when they are completely different.
  Alternatively, it is possible to represent each nominal categorical variable by means of a number of asymmetric binary attributes equal to the number of distinct values assumed by the nominal variable. Once the nominal variables in the dataset have been replaced by the categorical attributes, we can therefore use the Jaccard coefficient (*) to express the degree of affinity.
- **Ordinal categorical attributes:**
  Ordinal categorical attributes can be placed on a natural ordering scale, whose numerical values are, however, arbitrary. Therefore, they require a preliminary standardization so that they can be adapted to the metrics defined for numerical attributes.

To clarify this concept, assume that the values of each ordinal categorical attribute are represented through the corresponding position in the natural order. If, for example, the ordinal variable corresponds to the level of education, and can be expressed by the values {elementary school, high school graduate, bachelor's degree, postgraduate}, the elementary level is matched with numerical value 1, the high school graduate level with 2, and so on.
Let $\mathcal{H}_j = \{1, 2, \dots, H\}$ be the ordered values associated with the ordinal attribute $\boldsymbol{a}_j$.
To standardize the values assumed by the attribute $\boldsymbol{a}_j$ to the interval [0,1], the following transformation is used:

$$x'_{ij} = \frac{x_{ij} - 1}{H_j - 1}$$

After we have carried out the transformation indicated for all the ordinal variables, we can use the measures of distance previously introduced for numerical attributes.

Mixed composition attributes:
Suppose that the n attributes in the dataset D have a mixed composition, in the sense that some of them are numerical, while others are binary symmetric or binary asymmetric or nominal categorical or ordinal categorical. We wish to define an overall affinity measure that can be used to evaluate the degree of similarity between two observations $\boldsymbol{x}_i$ and $\boldsymbol{x}_k$ even if mixed composition attributes are present.
Let us define a binary indicator δ_{ikj} that takes the values 0 if and only if one of the following cases occurs:

- at least one of the two values x_{ij} or $x_{kj}$is missing in the corresponding records of the dataset;
- the attribute $\boldsymbol{a}_j$ is binary and asymmetric and, moreover, $x_{ij} = x_{kj} = 0$;

For all the remaining cases the indicator δ_{ikj} takes the value 1. Also define the contribution Δ_{ikj} of the variable $\boldsymbol{a}_j$ to the similarity between $\boldsymbol{x}_i$ and $\boldsymbol{x}_k$, based on the nature of the attribute $\boldsymbol{a}_j$, as follows:

- if the attribute $\boldsymbol{a}_j$ is binary or nominal, if $x_{ij} = x_{kj}$ we set $\Delta_{ikj} = 0$, otherwise $\Delta_{ikj} = 1$;
- if the attribute $\boldsymbol{a}_j$ is numerical, we set:

$$\Delta_{ikj} = \frac{|x_{ij} - x_{kj}|}{\max_l x_{lj}}$$

- if the attribute $\boldsymbol{a}_j$ is ordinal, its standardized value is computed as previously described and we set Δ_{ikj} as for numerical attributes.

Therefore, we can define the similarity coefficient between the observations $\boldsymbol{x}_i$ and $\boldsymbol{x}_k$ as:

$$\text{dist}(\boldsymbol{x}_i, \boldsymbol{x}_k) = \frac{\sum_{j=1}^{n} \delta_{ikj} \Delta_{ikj}}{\sum_{j=1}^{n} \delta_{ikj}}$$

2. PARTITION METHODS:

Given a dataset D of m observations, each represented by a vector in n-dimensional space, partition methods construct a **subdivision** of D into a collection of non-empty subsets $C = \{C_1, C_2, \dots, C_K\}$, where $K \leq m$. In general, the number K of clusters is predetermined and assigned as an input to partition algorithms. The clusters generated by partition methods are usually exhaustive and mutually exclusive, in the sense that each observation belongs to one and only one cluster. There are, however, fuzzy partition methods that assign each observation to different clusters according to a specific proportion.
Partition methods start with an initial assignment of the m available observations to the K clusters. Then, they **iteratively** apply a reallocation technique whose purpose is to place some observations in a different

cluster, so that the overall quality of the subdivision is improved. Although alternative measures of the clustering quality can be used, all the various criteria tend to express the degree of homogeneity of the observations belonging to the same cluster and their heterogeneity with respect to the records included in other clusters. Partition algorithms usually **stop** when during the same iteration no reallocation occurs, and therefore the subdivision appears stable with respect to the evaluation criterion chosen.
Partition methods are therefore of a heuristic nature, in the sense that they are based on a myopic logic typical of the class of so-called greedy methods, and at each step they make the choice that locally appears the most advantageous. Operating in this way, there is no guarantee that a globally optimal clustering will be reached, but only that a good subdivision will be obtained, at least for the majority of the datasets.
The *K-means* method and the *K-medoids* method, which will be described next, are two of the best-known partition algorithms. They are rather efficient clustering methods that are effective in determining clusters of spherical shape

2.1. K-means algorithm:

The K-means algorithm receives as input a dataset D, a number K of clusters to be generated and a function dist($\boldsymbol{x}_i$, $\boldsymbol{x}_k$) that expresses the inhomogeneity between each pair of observations, or equivalently the matrix $\mathbf{D}$ of distances between observations.
Given a cluster C_h, $h = 1,2, \dots, K$, the *centroid* of the cluster is defined as the point $\boldsymbol{z}_h$ having coordinates equal to the mean value of each attribute for the observations belonging to that cluster, that is:

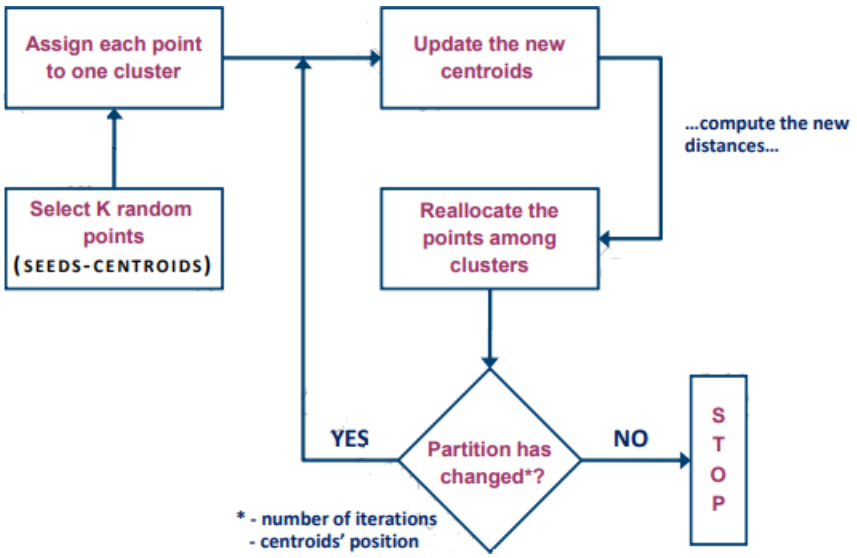
$$z_{hj} = \frac{\sum_{\boldsymbol{x}_i \in C_h} x_{ij}}{\text{card}\{C_h\}}$$

The algorithm, described in Procedure 12.1, starts by arbitrarily selecting K observations that constitute the initial centroids. For example, the K points can be randomly selected among the m observations in D. At each subsequent iteration, each record is assigned to the cluster whose centroid is the closest, that is, which minimizes the distance from the observation among all centroids. If no observation is reallocated to a cluster different from the one to which it belongs, determined during the previous iteration, the procedure stops, since any subsequent iteration cannot alter the current subdivision in clusters. Otherwise, the new centroid for each cluster is computed and a new assignment made.

Procedure 12.1 – K-means algorithm

1. During the initialization phase, K observations are arbitrarily chosen in $\mathcal{D}$ as the centroids of the clusters.
2. Each observation is iteratively assigned to the cluster whose centroid is the most similar to the observation, in the sense that it minimizes the distance from the record.
3. If no observation is assigned to a different cluster with respect to the previous iteration, the algorithm stops.
4. For each cluster, the new centroid is computed as the mean of the values of the observations belonging to the cluster, and then the algorithm returns to step 2.

It can also be summarized as:



The figure below illustrates the application of the K-means algorithm to a set of observations in the Euclidean plane. Suppose that we have decided on $K = 3$ as the desired number of clusters.

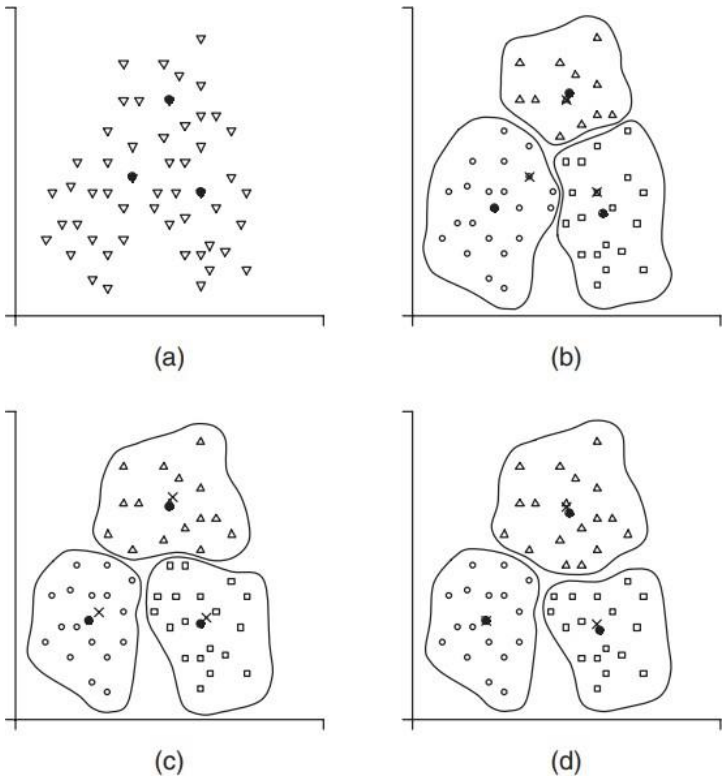
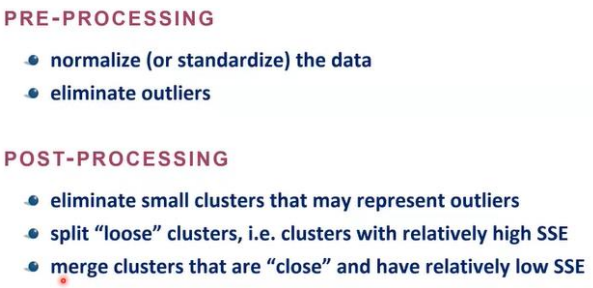


Figure (a) shows the points corresponding to the two-dimensional observations that are to be subdivided into three clusters. During the initialization phase, three observations are arbitrarily selected, which are indicated in the figure as large dots and constitute the three initial centroids. At the first iteration, each observation is assigned to the closest centroid, generating the clusters highlighted in Figure (b), where the observations belonging to distinct clusters are indicated by three different graphical symbols. Figure (b) also shows the new centroids, computed as the mean of the attributes for the observations belonging to each cluster and represented as large dots. The three centroids of the previous iteration are indicated in diagram (b) by the symbol ×. Figures (c) and (d)

respectively show the second and the third iterations of the algorithm. Since at the end of the third iteration no observation is assigned to a different cluster, the algorithm stops.

The K-means algorithm has strengths and weaknesses:

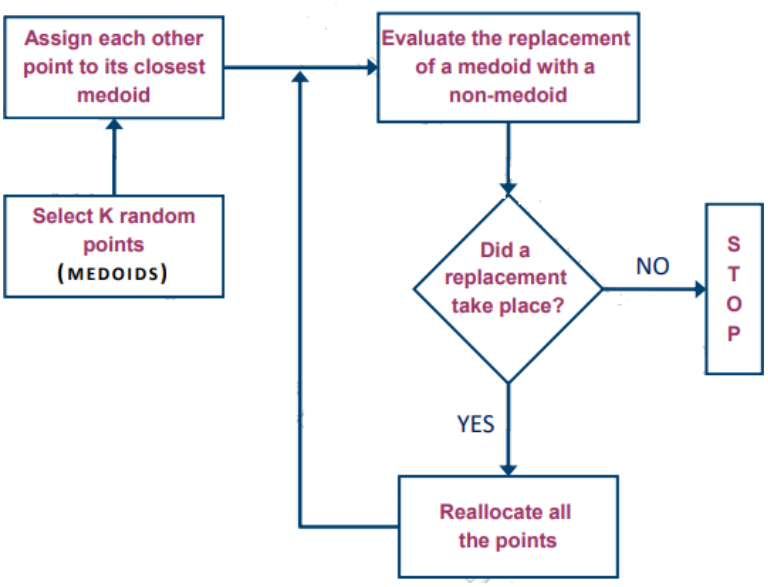
- It is relatively efficient: $O(m * k * t * d)$ [m point, K clusters, t iterations, d variables].
- It converges for common similarity measures mentioned above (most of the convergence happens in the first few iterations).
- Need to specify the number of clusters in advance.
- Often terminates at a local optimum (greedy algorithm).
- Results may vary based on random seed selection, i.e. clusters may be different from one run to another;
- Applicable only when mean is defined
- Unable to handle noisy data outliers.
- It may have problems when (natural) clusters are of different sizes, have different densities, or have non-convex shapes.



2.2. k-medoids algorithm:

The K-medoids algorithm, also known as partitioning around medoids, is a variant of the K-means method. It is based on the use of medoids instead of the means of the observations belonging to each cluster, with the purpose of mitigating the sensitivity of the partitions generated with respect to the extreme values in the dataset. Given a cluster C_h, a medoid $\boldsymbol{u}_h$ is the most central observation, in a sense that will be formally defined below, among those that are assigned to C_h. Once the medoid representing each cluster has been identified, the K-medoids algorithm proceedslike the K-means method, using medoids instead of centroids. The initialization phase involves the arbitrary selection of K observations representing the medoids at the first iteration. Each of the remaining observations is assigned to the medoid to which it is most similar, so as to minimize the distance measure $\mathrm{dist}(\boldsymbol{x}_i, \boldsymbol{u}_h)$. During each subsequent iteration, the algorithm tries to substitute each medoid with an observation that does not represent a medoid, provided that the overall quality of the subdivision in clusters is improved. The quality of the subdivision is evaluated through a measure of average inhomogeneity between each observation and the medoid associated with the cluster to which it belongs.

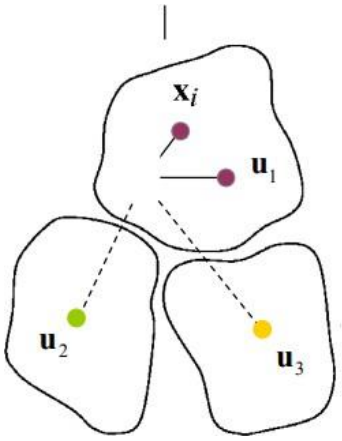
The algorithm can be summarized as:


Assign each other point to its closest medoid
Evaluate the replacement of a medoid with a non-medoid
Select K random points (MEDOIDS)
Did a replacement take place?
NO
STOP
YES
Reallocate all the points

Let U be the set of K medoids at a given iteration. To assess the advantage afforded by an exchange of medoids, the algorithm considers all the $(m-K)K$ pairs $(\boldsymbol{x}_i, \boldsymbol{u}_h)$. Composed of an observation $x_i \notin U$ and a medoid $\boldsymbol{u}_h \in U$. For each pair, every observation $\boldsymbol{x}_k \notin U$ different from $\boldsymbol{x}_i$ is considered, and the contribution R_{ijk} made by $\boldsymbol{x}_k$ to the exchange between $\boldsymbol{x}_i$ and $\boldsymbol{u}_h$ is evaluated. The following four cases are then considered:

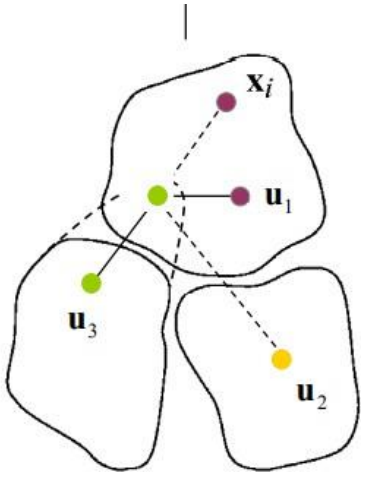
- $\boldsymbol{x}_k$ is currently assigned to the cluster corresponding to medoid $\boldsymbol{u}_h$, and also $\text{dist}(\boldsymbol{x}_i, \boldsymbol{x}_k) \leq \min_{\boldsymbol{u}_e \in U, e \neq h} \text{dist}(\boldsymbol{u}_i, \boldsymbol{x}_k)$. In this first case the observation $\boldsymbol{x}_k$ is assigned to the new medoid $\boldsymbol{x}_i$ intended to represent cluster C_h and the contribution to the exchange, which can be positive, zero, or negative, is given by:

$$R_{ihk} = \text{dist}(\boldsymbol{x}_i, \boldsymbol{x}_k) - \text{dist}(\boldsymbol{u}_h, \boldsymbol{x}_k)$$



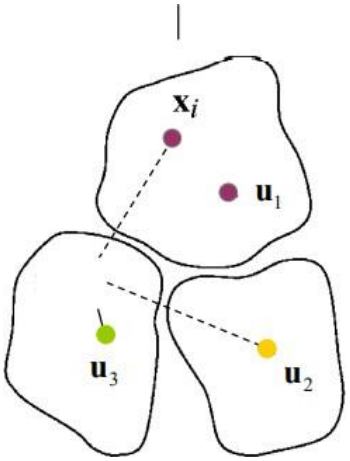
- $\boldsymbol{x}_k$ is currently assigned to the cluster corresponding to medoid $\boldsymbol{u}_h$, and also $\text{dist}(\boldsymbol{x}_i, \boldsymbol{x}_k) \leq \min_{\boldsymbol{u}_e \in U, e \neq h} \text{dist}(\boldsymbol{u}_i, \boldsymbol{x}_k)$. In this second case the observation $\boldsymbol{x}_k$ remains assigned to a different cluster, and the contribution to the exchange is given by:

$$R_{ihk} = \min_{\boldsymbol{u}_e \in U, e \neq h} \text{dist}(\boldsymbol{u}_e, \boldsymbol{x}_k) - \text{dist}(\boldsymbol{u}_h, \boldsymbol{x}_k)$$



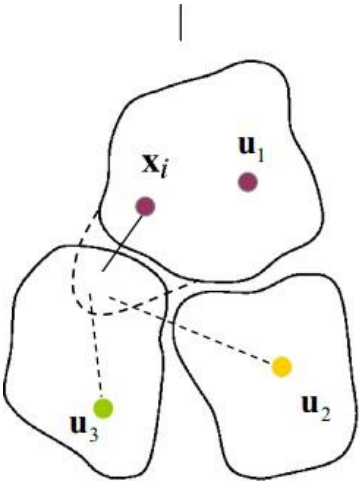
- $\boldsymbol{x}_k$ is not currently assigned to the cluster corresponding to the medoid $\boldsymbol{u}_h$ and also $\text{dist}(\boldsymbol{x}_i, \boldsymbol{x}_k) \leq \min_{\boldsymbol{u}_e \in U, e \neq h} \text{dist}(\boldsymbol{u}_i, \boldsymbol{x}_k)$. In this third case the observation $\boldsymbol{x}_k$ remains assigned to a different cluster, and the contribution to the exchange is given by:

$$R_{ihk} = 0$$



- $\boldsymbol{x}_k$ is not currently assigned to the cluster corresponding to the medoid $\boldsymbol{u}_h$ and also $\text{dist}(\boldsymbol{x}_i, \boldsymbol{x}_k) \leq \min_{\boldsymbol{u}_e \in U, e \neq h} \text{dist}(\boldsymbol{u}_i, \boldsymbol{x}_k)$. In this last case the observation $\boldsymbol{x}_k$ is assigned to cluster C_h, and the contribution to the exchange is given by:

$$R_{ihk} = \text{dist}(\boldsymbol{x}_i, \boldsymbol{x}_k) - \min_{\boldsymbol{u}_e \in U, e \neq h} \text{dist}(\boldsymbol{u}_e, \boldsymbol{x}_k)$$



Once the contribution R_{ihk} made by each observation $\boldsymbol{x}_k \notin U$ to the exchange between $\boldsymbol{x}_i$ and $\boldsymbol{u}_i$ has been calculated, the overall contribution (for each pair xi, uk) can be evaluated as:

$$T_{ih} = \sum_{\boldsymbol{x}_k \notin U} R_{ihk}$$

If the minimum value among the overall contributions T_{ih} is negative, the pair $(\boldsymbol{x}_i, \boldsymbol{u}_h)$ that corresponds to such minimum value is actually exchanged and the algorithm proceeds with a new

iteration, stopping when no exchange occurs, that is, when the minimum value of the overall contribution is non-negative. The K-medoids algorithm requires a large number of iterations and is not suited to deriving clusters for large datasets. A few variants have therefore been proposed, based on the extraction of samples of observations to which the K-medoids method can be applied.

3. HIERARCHICAL METHODS:

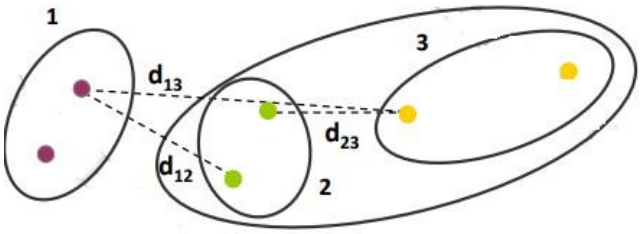
Hierarchical clustering methods are based on a tree structure. Unlike partition methods, they do not require the number of clusters to be determined in advance. Hence, they receive as input a dataset D containing m observations and a matrix of distances $\text{dist}(\boldsymbol{x}_i, \boldsymbol{x}_k)$ between all pairs of observations. Use the distances among points to derive cluster merging or splitting.

In order to evaluate the distance between two clusters, most hierarchical algorithms resort to one of five alternative measures: minimum distance, maximum distance, mean distance, distance between centroids, and Ward distance.

Suppose that we wish to calculate the distance between two clusters C_h and Cf and let $\boldsymbol{z}_h$ and $\boldsymbol{z}_f$ be the corresponding centroids.

Minimum distance: According to the criterion of *minimum distance*, also called the *single linkage criterion*, the dissimilarity between two clusters is given by the minimum distance among all pairs of observations such that one belongs to the first cluster and the other to the second cluster, that is:

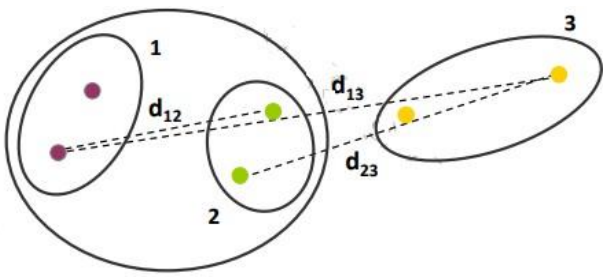
$$\text{dist}(C_h, C_f) = \min_{\substack{\boldsymbol{x}_i \in C_h \\ \boldsymbol{x}_k \in C_f}} \text{dist}(\boldsymbol{x}_i, \boldsymbol{x}_k)$$



Is sensitive to noise and outliers, biased toward elliptical cluster.

Maximum distance: According to the criterion of *maximum distance*, also called the *complete linkage criterion*, the dissimilarity between two clusters is given by the maximum distance among all pairs of observations such that one belongs to the first cluster and the other to the second cluster, that is:

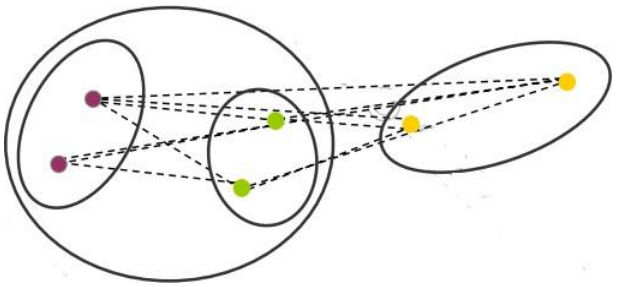
$$\text{dist}(C_h, C_f) = \max_{\substack{\boldsymbol{x}_i \in C_h \\ \boldsymbol{x}_k \in C_f}} \text{dist}(\boldsymbol{x}_i, \boldsymbol{x}_k)$$



Less sensitive to noise and outliers, tend to break large cluster, biased towards globular cluster.

Mean distance or average distance: The *mean distance criterion* expresses the dissimilarity between two clusters via themean of the distances between all pairs of observations belonging to the two clusters, that is:

$$\text{dist}(C_h, C_f) = \frac{\sum_{\boldsymbol{x}_i \in C_h} \sum_{\boldsymbol{x}_k C_f} \text{dist}(\boldsymbol{x}_i, \boldsymbol{x}_k)}{\text{card}\{C_h\}\text{card}\{C_f\}}$$



Less sensitive to noise and outliers, biased towards globular cluster.

Distance between centroids: The criterion based on the *distance between centroids* determines the dissimilarity between two clusters through the distance between the centroids representing the two clusters, that is:

$$\text{dist}(C_h, C_f) = \text{dist}(\boldsymbol{z}_h, \boldsymbol{z}_f)$$

Ward distance: minimizes the total within-cluster variance. The *criterion of Ward distance*, based on the analysis of the variance of the Euclidean distances between the observations, is slightly more complex than the criteria described above. Indeed,it requires the algorithm to first calculate the sum of squared distances between all pairs of observations belonging to a cluster. Afterwards, all pairs of clusters that could be merged at the current iteration are considered, and for each pair the total variance is computed as the sum of the two variances between the distances in each cluster, evaluated in the first step. Finally, the pair of clusters associated with the minimum total variance are merged. Methods based on the Ward distance tend to generate a large number of clusters, each containing a few observations.
let d_{ik} and d_{jk} be the pairwise distances between the clusters C_i, C_j and C_k. Let $d_{(ij)k}$ eb the distance (within cluster-variance) between the new cluster C_i U C_j and C_k, where:

$$d_{(ij)k} = \alpha_i \cdot d_{ik} + \alpha_j \cdot d_{jk} + \beta \cdot d_{ij} + \gamma |d_{ik} - d_{jk}|$$

WARD'S MINIMUM VARIANCE METHOD

It minimizes the total within-cluster variance

- at the initial step all clusters are singletons
- compute the squared Euclidean distances among all the points
- compute the increase in total within-cluster variance for all possible merging

  Let d_{ik} d_{ij} and d_{jk} be the pairwise distances between the clusters C_i, C_j and C_k
  Let $d_{(ij)k}$ the distance (within-cluster variance) between the new cluster $C_i \cup C_j$ and C_k, where $d_{(ij)k} = \alpha_i \cdot d_{ik} + \alpha_j \cdot d_{jk} + \beta \cdot d_{ij} + \gamma |d_{ik} - d_{jk}|$
  Parameters depend on clusters size.

- merge the two clusters leading to the minimum increase of within-cluster variance

  - less sensitive to noise and outliers
  - biased towards globular clusters

Hierarchical methods can be subdivided into two main groups: *agglomerative* and *divisive* methods.

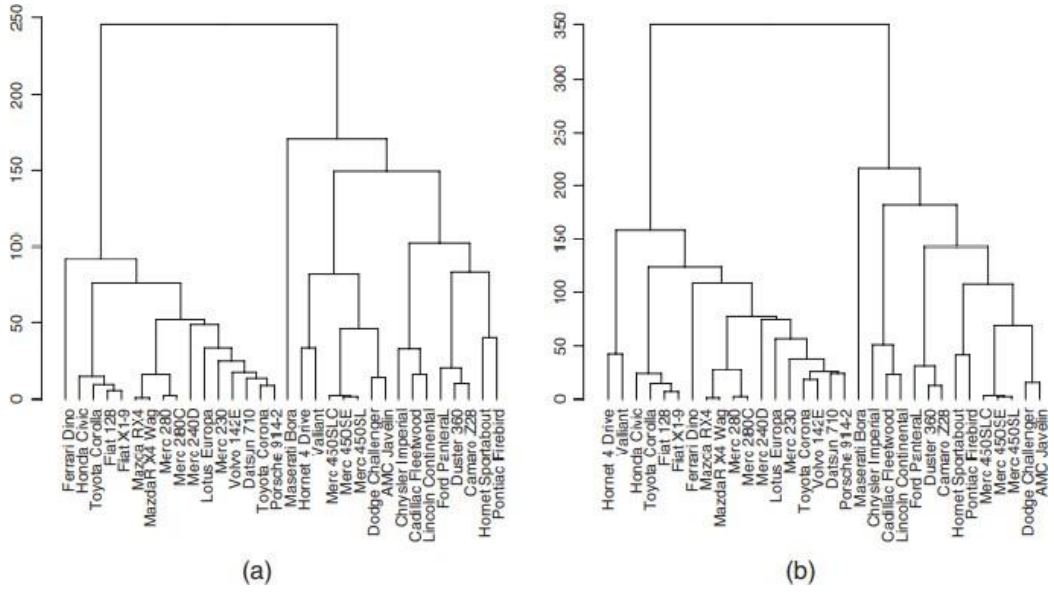
3.1. Agglomerative hierarchical methods:

Agglomerative methods are *bottom-up* techniques in which each single observation initially represents a distinct cluster. These clusters are then aggregated during subsequent iterations, deriving clusters of increasingly larger cardinalities (iteratively the 2 clusters with the minimum distance are merged together and the proximity matrix (distance among clusters) is updated.). The algorithm is stopped when a single cluster including all the observations has been reached. It is then necessary for the user to decide on a cut point and thus determine the number of clusters.
Procedure 12.2 describes the general structure of an agglomerative method.

Procedure 12.2 – Agglomerative algorithm

1. In the initialization phase, each observation constitutes a cluster. The distance between clusters therefore corresponds to the matrix **D** of the distances between all pairs of observations.
2. The minimum distance between the clusters is then computed, and the two clusters C_h and C_f with the minimum distance are merged, thus deriving a new cluster C_e. The corresponding minimum distance $\text{dist}(C_h, C_f)$ originating the merger is recorded.
3. The distance between the new cluster C_e, resulting from the merger between C_h and C_f, and the preexisting clusters is computed.
4. If all the observations are included into a single cluster, the procedure stops. Otherwise it is repeated from step 2.

At the end of the algorithm, it is possible to graphically represent the process of subsequent mergers using a *dendrogram*, indicating on one axis the value of the minimum distance corresponding to each merger and the observations on the other axis.
The figure below shows the dendrograms for the clustering obtained by applying an agglomerative hierarchical algorithm to a dataset, using (a) the mean Euclidean distance and (b) the mean Manhattan distance.



As we can see, the observations are placed on the horizontal axis and the distance value on the vertical axis. Actually, each dendrogram provides a whole hierarchy of clusters, corresponding to different threshold values for the minimum distance between clusters, indicated on the vertical axis. In the dendrogram shown in Figure (a), four clusters can be obtained if the tree is truncated at the distance 125, while nine clusters are generated by cutting the tree at 70. It can be observed that the dendrograms and the corresponding hierarchies of clusters obtained using the Euclidean and the Manhattan distances differ significantly.

The following figure in which the labels of each single observation have been removed, illustrates the effect of tree pruning.

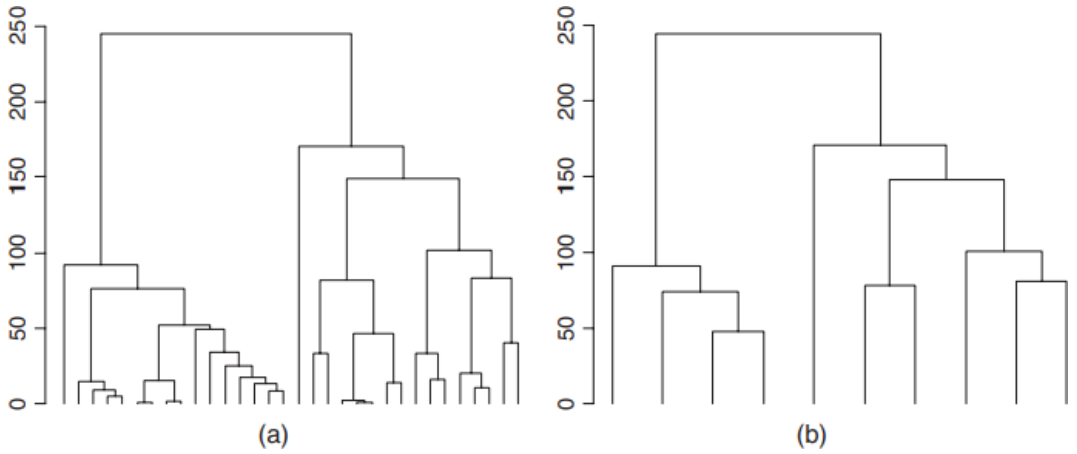
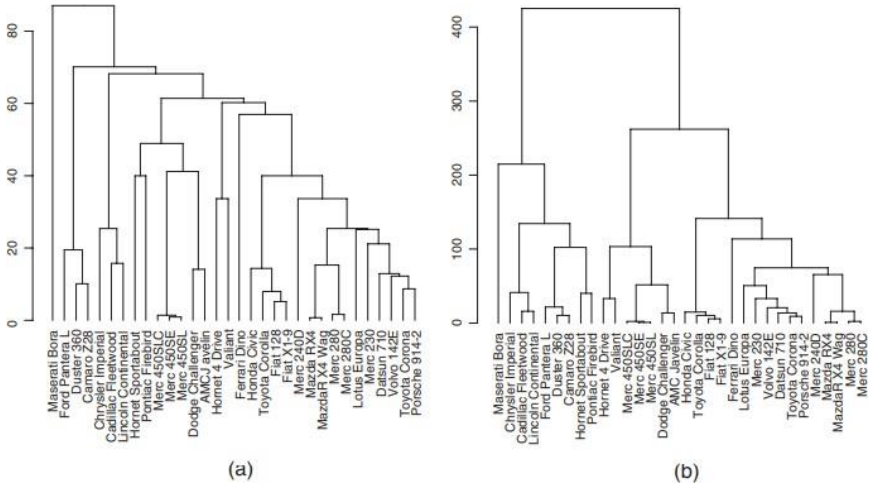


Figure (a) shows the same tree as in Figure (a) of the previous Figure, while Figure (b) has been obtained by means of a cut that results in ten clusters. In this way, a more aggregated view of the phenomenon can be achieved that leads to an easier interpretation.

The two figures below, Figure1 and Figure2, show the effect of changes in the metrics used by the agglomerative hierarchical algorithm.



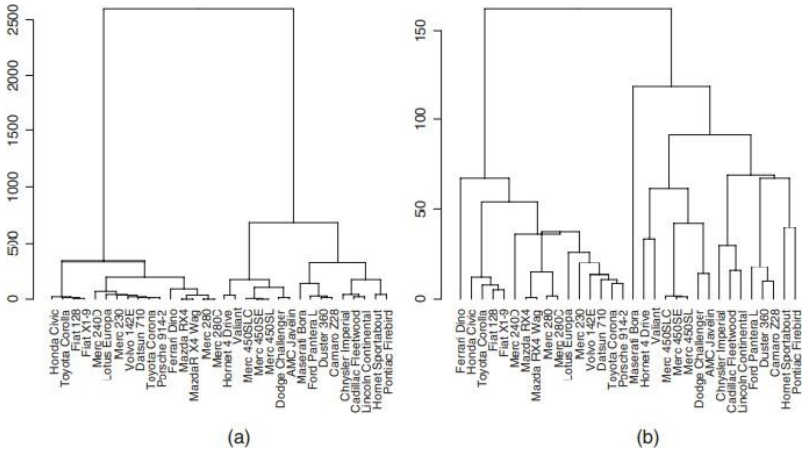


Figure1(a) shows the dendrogram obtained with the minimum distance and Figure1(b) the dendrogram generated with the maximum distance, respectively. Figure2(a) shows the dendrogram obtained with the Ward distance and Figure2(b) the dendrogram generated with the distance between the centroids, respectively. Confirming previous remarks, the Ward method produces a tree with several ramifications toward the leaves and therefore a structure made up of many small clusters.

More generally, these figures suggest that the type of metric selected has a relevant influence on the clustering generated.

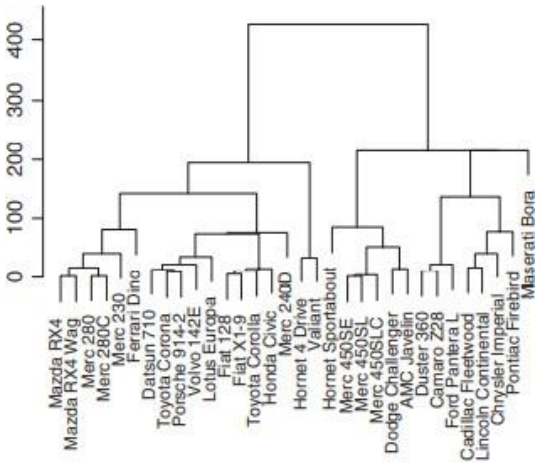
3.2. Divisive hierarchical methods:

Divisive algorithms are the opposite of agglomerative methods, in that they are based on a *top-down technique*, which initially places all the observations in a single cluster. This is then subdivided into clusters of smaller size, so that the distances between the generated subgroups are minimized. The procedure is repeated until clusters containing a single observation are obtained, or until an analogous stopping condition is met.

Unlike agglomerative methods, in order to keep computing times within reasonable limits, divisive algorithms require a strict limitation on the number of combinations that can be analyzed. In order to understand the motivations for this choice, consider the first step of the algorithm. The only cluster containing all the observations must be subdivided into two subsets, so that the distance between the two resulting clusters is maximized. Since there are $2m - 2$ possible partitions of the whole set into two non-empty disjoint subsets, this results in an exponential number of operations already at the first iteration.

To circumvent this difficulty, at any given iteration divisive hierarchical algorithms usually determine for each cluster the two observations that are furthest from each other and subdivide the cluster by assigning the remaining records to the one or the other, based on their proximity.

The figure below shows the dendrogram obtained by applying a divisive hierarchical algorithm to the same dataset of the examples above, using the mean Euclidean distance. This can be compared with the corresponding dendrogram, obtained using the same metrics but with an agglomerative hierarchical algorithm. As we can see, for this dataset the type of algorithm seems to exert less influence than the metric adopted.



(a)

4. EVALUATION OF CLUSTERING MODELS:

We have seen that for supervised learning methods, such as classification, regression and time series analysis, the evaluation of the predictive accuracy is part of the development process of a model and is based on specific numerical indicators. The same does not apply to clustering methods and, more generally, to unsupervised learning models. Even though the absence of a target attribute makes the evaluation of an unsupervised model less direct and intuitive, it is possible to define reasonable measures of quality and significance for clustering methods.

To evaluate a clustering method, it is first necessary to verify that the clusters generated correspond to an actual regular pattern in the data. It is therefore appropriate to apply other clustering algorithms and to compare the results obtained by different methods. In this way it is also possible to evaluate if the number of identified clusters is robust with respect to the different techniques applied.

At a subsequent phase it is recommended to calculate some performance indicators. Let $\checkmark = \{C_1, C_2, \dots, C_K\}$ be the K clusters generated.

An indicator of homogeneity of the observations within each cluster C_h (measures how closely related are objects in a cluster) is given by the *cohesion*, defined as:

$$\text{coh}(C_h) = \sum_{\substack{\boldsymbol{x}_i \in C_h \\ \boldsymbol{x}_k \in C_h}} dist(\boldsymbol{x}_i, \boldsymbol{x}_k)$$

The overall cohesion of the partition $\mathcal{C}$ can therefore be defined as:

$$\text{coh}(\mathcal{C}) = \sum_{C_h \in \mathcal{C}} \text{coh}(C_h)$$

One clustering is preferable over another, in terms of homogeneity within each cluster, if it has a smaller overall cohesion.

An indicator of inhomogeneity between a pair of clusters is given by the *separation (how distinct or well-separated a cluster is from other cluster)*, defined as:

$$\text{sep}(C_h, C_f) = \sum_{\substack{\boldsymbol{x}_i \in C_h \\ \boldsymbol{x}_k \in C_f}} dist(\boldsymbol{x}_i, \boldsymbol{x}_k)$$

Again, the overall separation of the partition $\mathcal{C}$ can be defined as:

$$\text{sep}(\mathcal{C}) = \sum_{\substack{C_h \in \mathcal{C} \\ C_f \in \mathcal{C}}} sep(C_h, C_f)$$

One clustering is preferable over another, in terms of inhomogeneity among all clusters, if it has a greateroverall separation.

A further indicator of the clustering quality is given by the *silhouette coefficient*, which involves a combination of cohesion and separation. To calculate the silhouette coefficient for a single observation $\boldsymbol{x}_i$, three steps should be followed, as detailed in Procedure 12.3

Procedure 12.3 – Calculation of the silhouette coefficient

1. The mean distance u_i of $\mathbf{x}_i$ from all the remaining observations belonging to the same cluster is computed.
2. For each cluster C_f other than the cluster to which $\mathbf{x}_i$ belongs, the mean distance w_{if} between $\mathbf{x}_i$ and all the observations in C_f is calculated. The minimum v_i among the distances w_{if} is determined by varying the cluster C_f.
3. The silhouette coefficient of $\mathbf{x}_i$ is defined as

$$\text{silh}(\mathbf{x}_i) = \frac{v_i - u_i}{\max(u_i, v_i)}.$$

The silhouette coefficient varies between −1 and 1. A negative value indicates that the mean distance u_i of the observation $\boldsymbol{x}_i$ from the points of its cluster is greater than the minimum value v_i of the mean distances from the observations of the other clusters, and it is therefore undesirable since the membership of $\boldsymbol{x}_i$ in its cluster is not well characterized. Ideally the silhouette coefficient should be positive and u_i should be as close as possible to 0. Finally, it should be noticed that the overall silhouette coefficient of a clustering may be computed as the mean of the silhouette coefficients for all the observations in the dataset D.
Silhouette coefficients can be illustrated by silhouette diagrams, in which the observations are placed on the vertical axis, subdivided by clusters, and the values of the silhouette coefficient for each observation are shown on the horizontal axis. Usually, the mean value of the silhouette coefficient for each cluster is also given in a silhouette diagram, as well as the overall mean for the whole dataset.

SILHOUETTE COEFFICIENT:

- compute the average distance u_i of $\mathbf{x}_i$ from all the other points in the same cluster
- compute the average distance of $\mathbf{x}_i$ from all the other points comprised in a different cluster
  (let v_i be the minimum of these distances)
- the silhouette coefficient is given by

$$\text{silh}(\mathbf{x}_i) = \frac{v_i - u_i}{\max(u_i, v_i)}$$

- within [-1,1]
- the closer to 1 the better
- average silhouette

The figures below show the silhouette diagrams corresponding to different clusterings.

Figure (*)

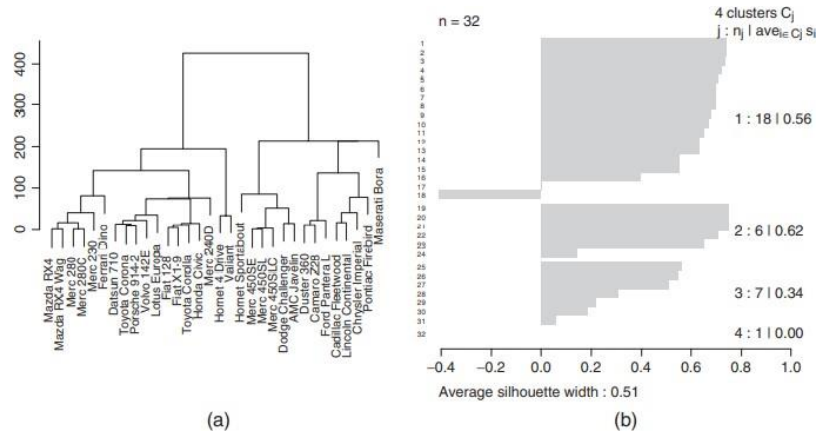
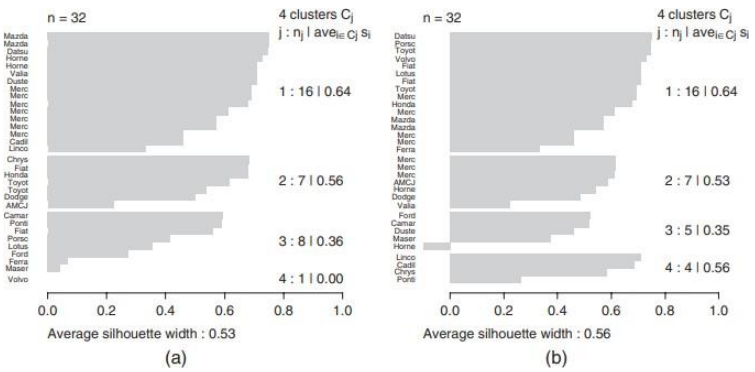


Figure (**)


n = 32
4 clusters Cj
j : nj | avei∈Cj si
1 : 16 | 0.64
2 : 7 | 0.56
3 : 8 | 0.36
4 : 1 | 0.00
0.0
0.2
0.4
0.6
0.8
1.0
Average silhouette width : 0.53
(a)
n = 32
4 clusters Cj
j : nj | avei∈Cj si
1 : 16 | 0.64
2 : 7 | 0.53
3 : 5 | 0.35
4 : 4 | 0.56
0.0
0.2
0.4
0.6
0.8
1.0
Average silhouette width : 0.56
(b)

In particular, Figure (*) (b) shows the silhouette diagram corresponding to a cut of four clusters in the dendrogram shown in Figure (*) (a), obtained by applying a divisive hierarchical algorithm to the mtcars dataset using the mean Euclidean distance. Figure (**) (a) shows the silhouette diagram corresponding to a cut of four clusters in the dendrogram obtained by applying an agglomerative hierarchical algorithm to the dataset using the mean Euclidean distance. Finally, Figure (**) (b) shows the silhouette diagram corresponding to the clustering with K = 4 obtained by applying a medoids partitioning algorithm.

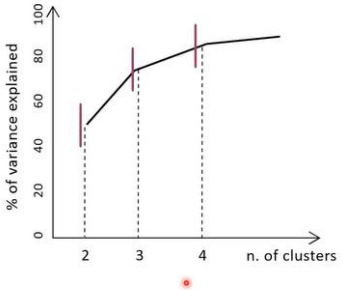
How many clusters?

Alternative methods are available...

- **RULE OF THUMB** $K \approx \left(\frac{m}{2}\right)^{1/2}$
- **ELBOW METHOD**

  The percentage of variance explained* is set as a function of the number of clusters

  *Is the ratio of the within-cluster variance to the total variance



- **SILHOUETTE-BASED METHOD**

  Choose the number of clusters giving rise to the largest average silhouette