# Machine Learning and Data Mining

Last update: 22 December 2023

Academic Year 2023 – 2024

Alma Mater Studiorum · University of Bologna

# Contents

# Acronyms

**BI** Business Intelligence

**CDC** Change Data Capture

**CRISP-DM** Cross Industry Standard Process for Data Mining

**DFM** Dimensional Fact Model

**DM** Data Mart

**DSS** Decision Support System

**DWH** Data Warehouse

**EIS** Executive Information System

**ERP** Enterprise Resource Planning

**ETL** Extraction, Transformation, Loading

**MIS** Management Information System

**OLAP** Online Analytical Processing

**OLTP** Online Transaction Processing

# 1 Introduction

## 1.1 Data

**Data** Collection of raw values. Data

**Information** Organized data (e.g. relationships, context, ...). Information

**Knowledge** Understanding information. Knowledge

### 1.1.1 Data sources

**Transaction** Business event that generates or modifies data in an information system (e.g. database). Transaction

**Signal** Measure produced by a sensor. Signal

**External subjects**

### 1.1.2 Software

**Online Transaction Processing (OLTP)** Class of programs to support transaction-oriented applications and data storage. Suitable for real-time applications. Online Transaction Processing

**Enterprise Resource Planning (ERP)** Integrated system to manage all the processes of a business. Uses a shared database for all applications. Suitable for real-time applications. Enterprise Resource Planning

### 1.1.3 Insight

Decisions can be classified as:

- **Structured** Established and well-understood situations. What is needed is known. Structured decision
- **Unstructured** Unplanned and unclear situations. What is needed for the decision is unknown. Unstructured decision

Different levels of insight can be extracted by:

- **Management Information System (MIS)** Standardized reporting system built on an existing OLTP. Used for structured decisions. Management Information System
- **Decision Support System (DSS)** Analytical system to provide support for unstructured decisions. Decision Support System
- **Executive Information System (EIS)** Formulate high-level decisions that impact the organization. Executive Information System
- **Online Analytical Processing (OLAP)** Grouped analysis of multidimensional data. Involves a large amount of data. Online Analytical Processing

**Business Intelligence (BI)** Applications, infrastructure, tools and best practices to analyze information.

Business Intelligence

**Big data** Large and/or complex and/or fast-changing collection of data that traditional DBMSs are unable to process.

Big data

- **Structured** e.g. relational tables.
- **Unstructured** e.g. videos.
- **Semi-structured** e.g. JSON.

**Anaylitics** Structured decision driven by data.

Anaylitics

**Data mining** Discovery process for unstructured decisions.

Data mining

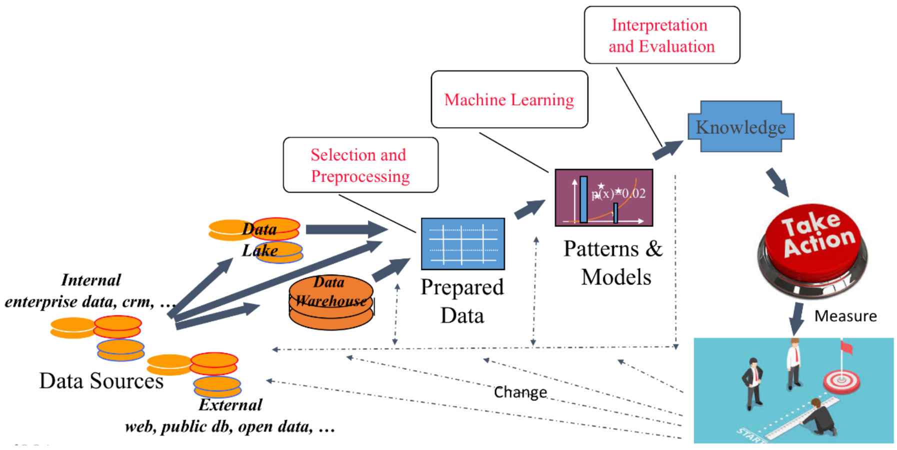



Figure 1.1: Data mining process

**Machine learning** Learning models and algorithms that allow to extract patterns from data.

Machine learning

# 2 Data warehouse

**Business Intelligence** Transform raw data into information. Deliver the right information to the right people at the right time through the right channel.

Business Intelligence

**Data Warehouse (DWH)** Optimized repository that stores information for decision-making processes. DWHs are a specific type of DSS.

Data Warehouse

Features:

- Subject-oriented: focused on enterprise-specific concepts.
- Integrates data from different sources and provides a unified view.
- Non-volatile storage with change tracking.

**Data Mart (DM)** Subset of the primary DWH with information relevant to a specific business area.

Data Mart

## 2.1 Online Analytical Processing (OLAP)

**OLAP analyses** Able to interactively navigate the information in a data warehouse. Allows to visualize different levels of aggregation.

Online Analytical Processing (OLAP)

**OLAP session** Navigation path created by the operations that a user applied.

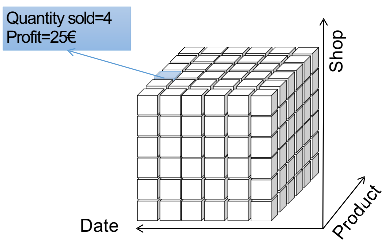
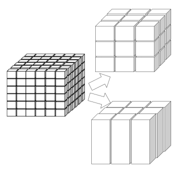



Figure 2.1: OLAP data cube

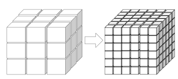
### 2.1.1 Operators

**Roll-up** Increases the level of aggregation (i.e. `GROUP BY` in SQL). Some details are collapsed together.



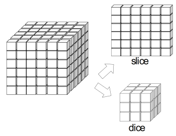
Roll-up

**Drill-down** Reduces the level of aggregation. Some details are reintroduced.



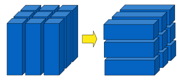
Drill-down

**Slide-and-dice** The slice operator reduces the number of dimensions (i.e. drops columns).
The dice operator reduces the number of data being analyzed (i.e. `LIMIT` in SQL).



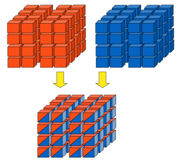

Slide-and-dice

**Pivot** Changes the layout of the data, to analyze it from a different viewpoint.



Pivot

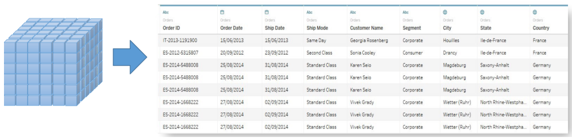
**Drill-across** Links concepts from different data sources (i.e. `JOIN` in SQL).



Drill-across

**Drill-through** Switches from multidimensional aggregated data to operational data (e.g. a spreadsheet).

Drill-through



| Order ID | Order Date | Ship Date | Ship Mode | Customer Name | Segment | City | State | Country |
|---|---|---|---|---|---|---|---|---|
| IT-2013-1191900 | 15/06/2013 | 15/06/2013 | Same Day | Georgia Rosenberg | Corporate | Houilles | Ile-de-France | France |
| ES-2012-5315807 | 20/09/2012 | 23/09/2012 | Second Class | Sonia Cooley | Consumer | Drancy | Ile-de-France | France |
| ES-2014-5488008 | 25/08/2014 | 31/08/2014 | Standard Class | Karen Seio | Corporate | Magdeburg | Saxony-Anhalt | Germany |
| ES-2014-5488008 | 25/08/2014 | 31/08/2014 | Standard Class | Karen Seio | Corporate | Magdeburg | Saxony-Anhalt | Germany |
| ES-2014-5488008 | 25/08/2014 | 31/08/2014 | Standard Class | Karen Seio | Corporate | Magdeburg | Saxony-Anhalt | Germany |
| ES-2014-1668222 | 27/08/2014 | 02/09/2014 | Standard Class | Vivek Grady | Corporate | Wetter (Ruhr) | North Rhine-Westpha... | Germany |
| ES-2014-1668222 | 27/08/2014 | 02/09/2014 | Standard Class | Vivek Grady | Corporate | Wetter (Ruhr) | North Rhine-Westpha... | Germany |
| ES-2014-1668222 | 27/08/2014 | 02/09/2014 | Standard Class | Vivek Grady | Corporate | Wetter (Ruhr) | North Rhine-Westpha... | Germany |

## 2.2 Extraction, Transformation, Loading (ETL)

The ETL process extracts, integrates and cleans operational data that will be loaded into a data warehouse.

Extraction, Transformation, Loading (ETL)

### 2.2.1 Extraction

Extracted operational data can be:

**Structured** with a predefined data model (e.g. relational DB, CSV)

Strucured data

**Untructured** without a predefined data model (e.g. social media content)

Unstrucured data

Extraction can be of two types:

**Static** The entirety of the operational data are extracted to populate the data warehouse for the first time.

Static extraction

**Incremental** Only changes applied since the last extraction are considered. Can be based on a timestamp or a trigger.

Incremental extraction

### 2.2.2 Cleaning

Operational data may contain:

**Duplicate data**

**Missing data**

**Improper use of fields** (e.g. saving the phone number in the `notes` field)

**Wrong values** (e.g. 30th of February)

**Inconsistencies** (e.g. use of different abbreviations)

**Typos**

Methods to clean and increase the quality of the data are:

**Dictionary-based techniques** Lookup tables to substitute abbreviations, synonyms or typos. Applicable if the domain is known and limited. Dictionary-based cleaning

**Approximate merging** Methods to merge data that do not have a common key. Approximate merging

- **Approximate join** Use non-key attributes to join two tables (e.g. using the name and surname instead of a unique identifier).
- **Similarity approach** Use similarity functions (e.g. edit distance) to merge multiple instances of the same information (e.g. typo in customer surname).

**Ad-hoc algorithms** Ad-hoc algorithms

### 2.2.3 Transformation

Data are transformed to respect the format of the data warehouse:

**Conversion** Modifications of types and formats (e.g. date format) Conversion

**Enrichment** Creating new information by using existing attributes (e.g. compute profit from receipts and expenses) Enrichment

**Separation and concatenation** Denormalization of the data: introduces redundancies (i.e. breaks normal form[1]) to speed up operations. Separation and concatenation

### 2.2.4 Loading

Adding data into a data warehouse:

**Refresh** The entire DWH is rewritten. Refresh loading

**Update** Only the changes are added to the DWH. Old data are not modified. Update loading

## 2.3 Data warehouse architectures

The architecture of a data warehouse should meet the following requirements:

**Separation** Separate the analytical and transactional workflows.

**Scalability** Hardware and software should be easily upgradable.

**Extensibility** Capability to host new applications and technologies without the need to redesign the system.

**Security** Access control.

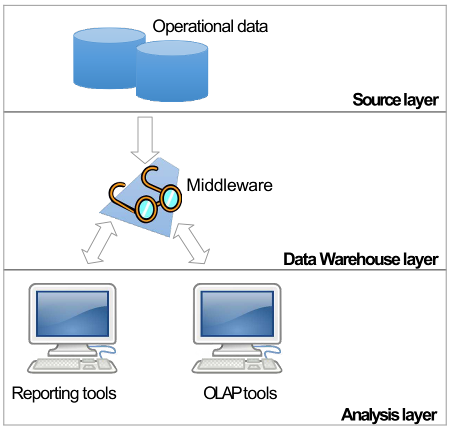
**Administrability** Easily manageable.

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

### 2.3.1 Single-layer architecture

Single-layer architecture

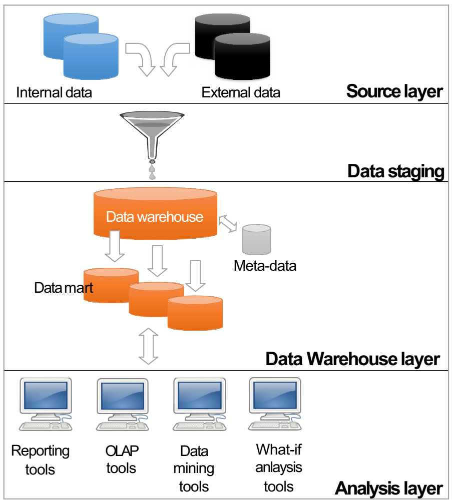
- Minimizes the amount of data stored (i.e. no redundances).
- The source layer is the only physical layer (i.e. no separation).
- A middleware provides the DWH features.




### 2.3.2 Two-layer architecture

Two-layer architecture

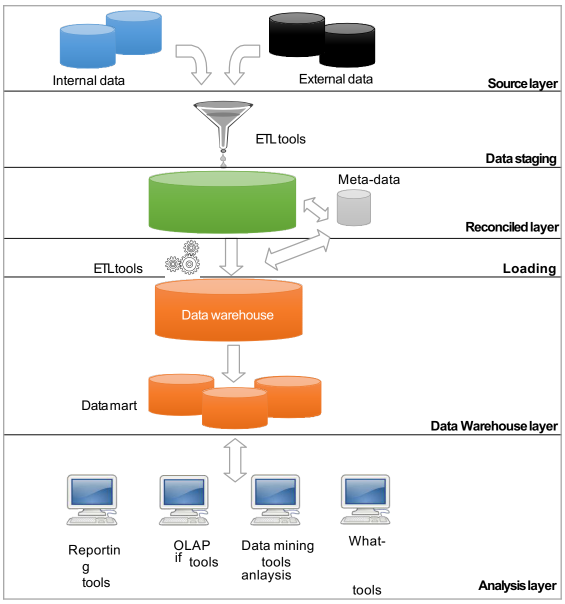
- Source data (source layer) are physically separated from the DWH (data warehouse layer).
- A staging layer applies ETL procedures before populating the DWH.
- The DWH is a centralized repository from which data marts can be created. Metadata repositories store information on sources, staging and data marts schematics.




### 2.3.3 Three-layer architecture

Three-layer architecture

- A reconciled layer enhances the cleaned data coming from the staging step by adding enterprise-level details (i.e. adds more redundancy before populating the DWH).



## 2.4 Conceptual modeling

**Dimensional Fact Model (DFM)** Conceptual model to support the design of data marts. The main concepts are:

Dimensional Fact Model (DFM)

- **Fact** Concept relevant to decision-making processes (e.g. sales).
- **Measure** Numerical property to describe a fact (e.g. profit).
- **Dimension** Property of a fact with a finite domain (e.g. date).
- **Dimensional attribute** Property of a dimension (e.g. month).
- **Hierarchy** A tree where the root is a dimension and nodes are dimensional attributes (e.g. date $\rightarrow$ month).
- **Primary event** Occurrence of a fact. It is described by a tuple with a value for each dimension and each measure.
- **Secondary event** Aggregation of primary events. Measures of primary events are aggregated if they have the same (preselected) dimensional attributes.

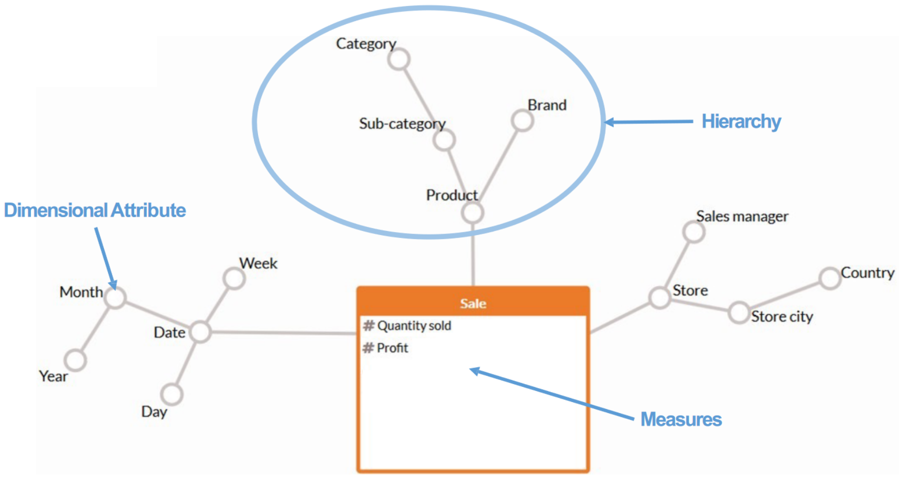



Figure 2.2: Example of DFM

Primary events

| Date | Store | Product | Qty sold | Profit |
|---|---|---|---|---|
| 01/03/15 | Central store | Milk | 20 | 60 |
| 01/03/15 | Central store | Coke | 25 | 50 |
| 02/03/15 | Central store | Bread | 40 | 70 |
| 10/03/15 | Central store | Wine | 15 | 150 |

SUM SUM

Secondary event

| Month | Store | Category | Qty sold | Profit |
|---|---|---|---|---|
| March 2015 | Central store | Food and Beverages | 100 | 330 |

Figure 2.3: Example of primary and secondary events

### 2.4.1 Aggregation operators

Measures can be classified as:

- **Flow measures** Evaluated cumulatively with respect to a time interval (e.g. quantity sold).

  Flow measures

- **Level measures** Evaluated at a particular time (e.g. number of products in inventory).

  Level measures

- **Unit measures** Evaluated at a particular time but expressed in relative terms (e.g. unit price).

  Unit measures

Aggregation operators can be classified as:

- **Distributive** Able to calculate aggregates from partial aggregates (e.g. `SUM`, `MIN`, `MAX`).

  Distributive operators

- **Algebraic** Requires a finite number of support measures to compute the result (e.g. `AVG`).

  Algebraic operators

- **Holistic** Requires an infinite number of support measures to compute the result (e.g. `RANK`).

  Holistic operators

**Additivity** A measure is additive along a dimension if an aggregation operator can be applied.

Additive measure

| | Temporal hierarchies | Non-temporal hierarchies |
|---|---|---|
| **Flow measures** | SUM, AVG, MIN, MAX | SUM, AVG, MIN, MAX |
| **Level measures** | AVG, MIN, MAX | SUM, AVG, MIN, MAX |
| **Unit measures** | AVG, MIN, MAX | AVG, MIN, MAX |

Table 2.1: Allowed operators for each measure type

## 2.5 Logical design

Defining the data structures (e.g. tables and relationships) according to a conceptual model. There are two main strategies:

Logical design

- **Star schema** A fact table that contains all the measures is linked to dimensional tables.

  Star schema

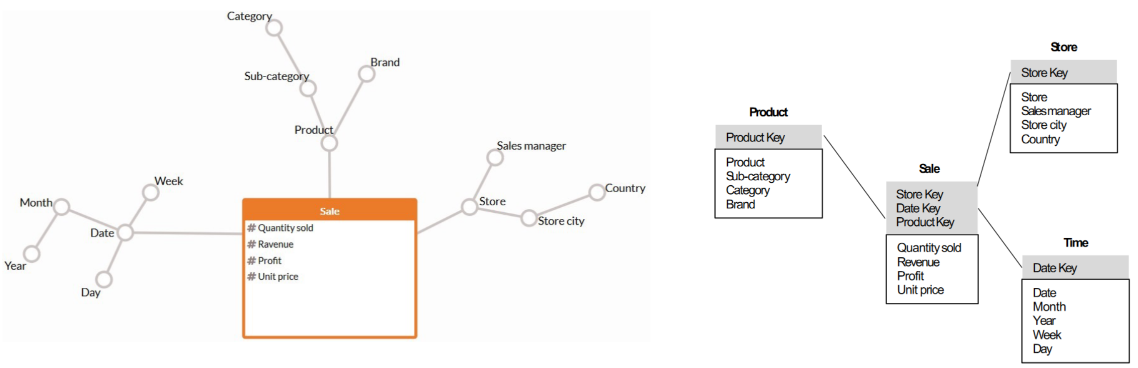



Figure 2.4: Example of star schema

**Snowflake schema** A star schema variant with partially normalized dimensional tables. Snowflake schema

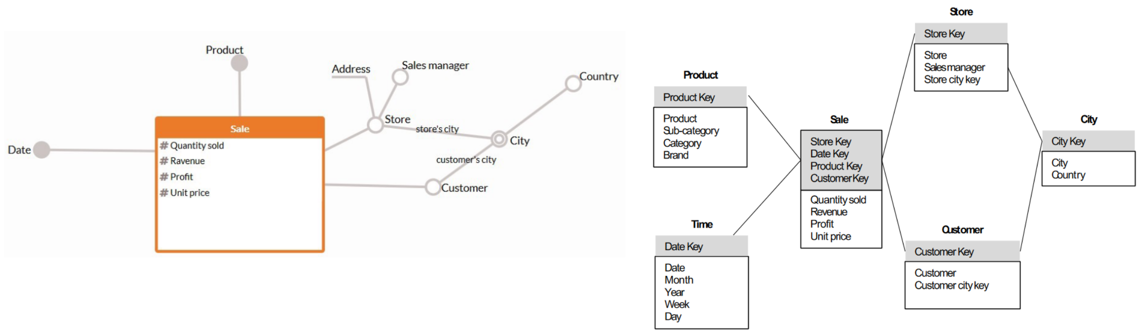



Figure 2.5: Example of snowflake schema

# 3 Data lake

**Dark data** Acquired and stored data that are never used for decision-making processes. Dark data

**Data lake** Repository to store raw (unstructured) data. It has the following features: Data lake

- Does not enforce a schema on write.
- Allows flexible access and applies schemas on read.
- Single source of truth.
- Low cost and scalable.

**Storage** Stored data can be classified as:

**Hot** A low volume of highly requested data that requires low latency. More expensive HW/SW. Hot storage

**Cold** A large amount of data that does not have latency requirements. Less expensive. Cold storage

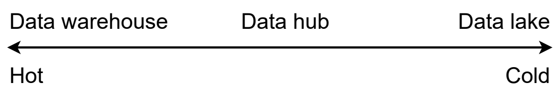



Figure 3.1: Data storage technologies

## 3.1 Traditional vs insight-driven data systems

| | **Traditional (data warehouse)** | **Insight-driven (data lake)** |
|---|---|---|
| **Sources** | Structured data | Structured, semi-structured and unstructured data |
| **Storage** | Limited ingestion and storage capability | Virtually unlimited ingestion and storage capability |
| **Schema** | Schema designed upfront | Schema not fixed |
| **Transformations** | ETL upfront | Transformations on query |
| **Analytics** | SQL, BI tools, full-text search | Traditional methods, self-service BI, big data, machine learning, ... |
| **Price**<br>**Performance** | High storage cost<br>Fast queries | Low storage cost<br>Scalability/speed/cost tradeoffs |
| **Quality** | High data quality | Depends on the use case |

## 3.2 Data architecture evolution

**Traditional data warehouse** (i.e. in-house data warehouse) Traditional data warehouse

- Structured data with predefined schemas.
- High setup and maintenance cost. Not scalable.

- Relational high-quality data.
- Slow data ingestion.

**Modern cloud data warehouse**

Modern cloud data warehouse

- Structured and semi-structured data.
- Low setup and maintenance cost. Scalable and easier disaster recovery.
- Relational high-quality data and mixed data.
- Fast data ingestion if supported.

**On-premise big data** (i.e. in-house data lake)

On-premise big data

- Any type of data with schemas on read.
- High setup and maintenance cost.
- Fast data ingestion.

**Cloud data lake**

Cloud data lake

- Any type of data with schemas on read.
- Low setup and maintenance cost. Scalable and easier disaster recovery.
- Fast data ingestion.

## 3.3 Components

### 3.3.1 Data ingestion

Data ingestion

**Workload migration** Inserting all the data from an existing source.

**Incremental ingestion** Inserting changes since the last ingestion.

**Streaming ingestion** Continuously inserting data.

**Change Data Capture (CDC)** Mechanism to detect changes and insert the new data into the data lake (possibly in real-time).

Change Data Capture (CDC)

### 3.3.2 Storage

**Raw** Immutable data useful for disaster recovery.

Raw storage

**Optimized** Optimized raw data for faster query.

Optimized storage

**Analytics** Ready to use data.

Analytics storage

**Columnar storage**

- Homogenous data are stored contiguously.
- Speeds up methods that process entire columns (i.e. all the values of a feature).
- Insertion becomes slower.

**Data catalog** Methods to add descriptive metadata to a data lake. This is useful to prevent an unorganized data lake (data swamp).

### 3.3.3 Processing and analytics

Processing and analytics

**Interactive analytics** Interactive queries to large volumes of data. The results are stored back in the data lake.

**Big data analytics** Data aggregations and transformations.

**Real-time analytics** Streaming analysis.

## 3.4 Architectures

### 3.4.1 Lambda lake

Lambda lake

**Batch layer** Receives and stores the data. Prepares the batch views for the serving layer.

**Serving layer** Indexes batch views for faster queries.

**Speed layer** Receives the data and prepares real-time views. The views are also stored in the serving layer.

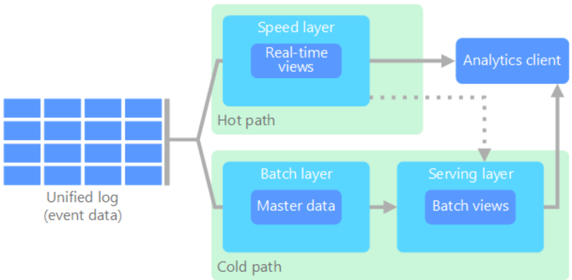



Figure 3.2: Lambda lake architecture

### 3.4.2 Kappa lake

Kappa lake

The data are stored in a long-term store. Computations only happen in the speed layer (avoids lambda lake redundancy between batch layer and speed layer).

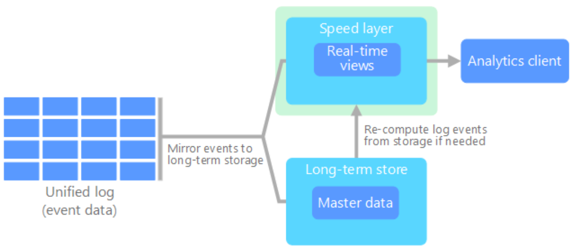



Figure 3.3: Kappa lake architecture

### 3.4.3 Delta lake

Delta lake

Framework that adds features on top of an existing data lake.

- ACID transactions

- Scalable metadata handling
- Data versioning
- Unified batch and streaming
- Schema enforcement

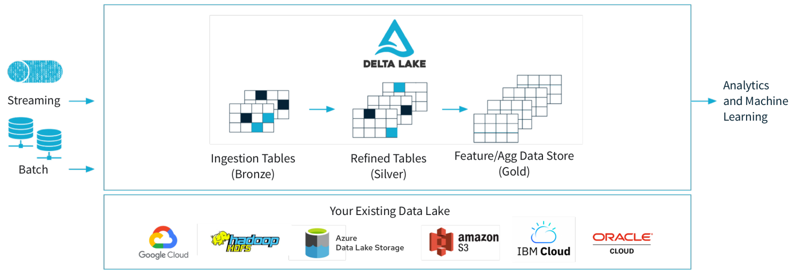



Figure 3.4: Delta lake architecture

## 3.5 Metadata

Metadata is used to organize a data lake. Useful metadata are: Metadata

**Source** Origin of the data.

**Schema** Structure of the data.

**Format** File format or encoding.

**Quality metrics** (e.g. percentage of missing values).

**Lifecycle** Retention policies and archiving rules.

**Ownership**

**Lineage** History of applied transformations or dependencies.

**Access control**

**Classification** Sensitivity level of the data.

**Usage information** Record of who accessed the data and how it is used.

# 4 CRISP-DM

**Cross Industry Standard Process for Data Mining** Standardized process for data mining. CRISP-DM

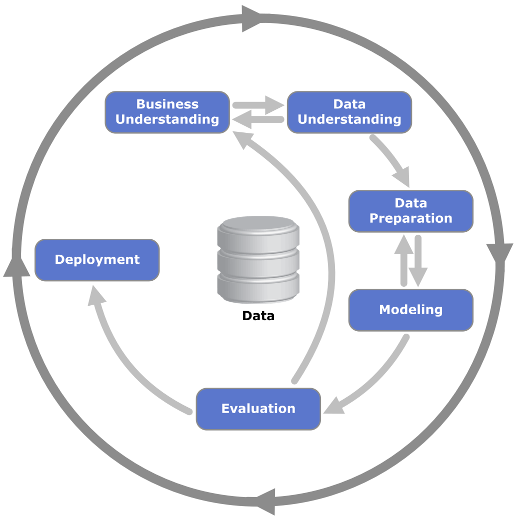



Figure 4.1: CRISP-DM workflow

## 4.1 Business understanding

Business understanding

- Determine the objective and the success criteria.
- Feasibility study.
- Produce a plan.

## 4.2 Data understanding

Data understanding

- Determine the available (raw) data.
- Determine the cost of the data.
- Collect, describe, explore and verify data.

## 4.3 Data preparation

Data preparation

- Data cleaning.
- Data transformations.

## 4.4 Modeling

- Select modeling technique. Modeling
- Build/train the model.

## 4.5 Evaluation

- Evaluate results. Evaluation
- Review process.

## 4.6 Deployment

- Plan deployment. Deployment
- Plan monitoring and maintenance.
- Final report and review.

# 5 Machine learning

**Machine learning** Application of methods and algorithms to extract patterns from data. Machine learning

## 5.1 Tasks

**Classification** Estimation of a finite number of classes.

**Regression** Estimation of a numeric value.

**Similarity matching** Identify similar individuals.

**Clustering** Grouping individuals based on their similarities.

**Co-occurrence grouping** Identify associations between entities based on the transactions in which they appear together.

**Profiling** Behavior description.

**Link analysis** Analysis of connections (e.g. in a graph).

**Data reduction** Reduce the dimensionality of data with minimal information loss.

**Casual modeling** Understand the connections between events and actions.

## 5.2 Categories

**Supervised learning** Problem where the target(s) is defined. Supervised learning

**Unsupervised learning** Problem where no specific target is known. Unsupervised learning

**Reinforcement learning** Learn a policy to generate a sequence of actions. Reinforcement learning

## 5.3 Data

**Dataset** Set of $N$ individuals, each described by $D$ features. Dataset

### 5.3.1 Data types

**Categorical** Values with a discrete domain.

**Nominal** The values are a set of non-ordered labels. Categorical nominal data

**Operators.** $=, \neq$

**Example.** Name, surname, zip code.

**Ordinal** The values are a set of totally ordered labels. Categorical ordinal data

**Operators.** $=, \neq, <, >, \leq, \geq$

**Example.** Non-numerical quality evaluations (excellent, good, fair, poor, bad).

**Numerical** Values with a continuous domain.

**Interval** Numerical values without an univocal definition of 0 (i.e. 0 is not used as reference). It is not reasonable to compare the magnitude of this type of data. Numerical interval data

**Operators.** $=, \neq, <, >, \leq, \geq, +, -$

**Example.** Celsius and Fahrenheit temperature scales, CGPA, time, . . . .

For instance, there is a 6.25% increase from 16°C to 17°C, but converted in Fahrenheit, the increase is of 2.96% (from 60.8°F to 62.6°F).

**Ratio** Values with an absolute 0 point. Numerical ratio data

**Operators.** $=, \neq, <, >, \leq, \geq, +, -$

**Example.** Kelvin temperature scale, age, income, length.

For instance, there is a 10% increase from 100\$ to 110\$. Converted in euro (1€= 1.06\$), the increase is still of 10% (from 94.34€ to 103.77€).

### 5.3.2 Transformations

| Data type | | Transformation |
|---|---|---|
| Categorical | Nominal | One-to-one transformations |
| | Ordinal | Order preserving transformations (i.e. monotonic functions) |
| Numerical | Interval | Linear transformations |
| | Ratio | Any mathematical function, standardization, variation in percentage |

### 5.3.3 Dataset format

**Relational table** The attributes of each record are the same. Relational table

**Data matrix** Matrix with $N$ rows (entries) and $D$ columns (attributes). Data matrix

**Sparse matrix** Data matrix with lots of zeros. Sparse matrix

**Example** (Bag-of-words)**.** Each row represents a document, each column represents a term. The $i, j$-th cell contains the frequency of the $j$-th term in the $i$-th document.

**Transactional data** Each record contains a set of objects (not necessarily a relational table). Transactional data

**Graph data** Set of nodes and edges. Graph data

**Ordered data** e.g. temporal data. Ordered data

### 5.3.4 Data quality

**Noise** Alteration of the original values. Noise

**Outliers** Data that considerably differ from the majority of the dataset. May be caused by noise or rare events. Outliers

Box plots can be used to visually detect outliers.

**Missing values** Data that have not been collected. Sometimes they are not easily recognizable (e.g. when special values are used to mark missing data instead of `null`). Missing values

Can be handled in different ways:

- Ignore the records with missing values.
- Estimate or default missing values.
- Ignore the fact that some values are missing (not always applicable).
- Insert all the possible values and weigh them by their probability.

**Duplicated data** Data that may be merged. Duplicated data

# 6 Data preprocessing

## 6.1 Aggregation

Combining multiple attributes into a single one. Useful for: Aggregation

**Data reduction** Reduce the number of attributes.

**Change of scale** View the data in a more general level of detail (e.g. from cities and regions to countries).

**Data stability** Aggregated data tend to have less variability.

## 6.2 Sampling

Sampling can be used when the full dataset is too expensive to obtain or too expensive to process. Obviously, a sample has to be representative. Sampling
The types of sampling techniques are:

**Simple random** Extraction of a single element following a given probability distribution. Simple random

**With replacement** Multiple extractions with repetitions following a given probability distribution (i.e. multiple simple random extractions). With replacement

If the population is small, the sample may underestimate the actual population.

**Without replacement** Multiple extractions without repetitions following a given probability distribution. Without replacement

**Stratified** Split the data and sample from each partition. Useful when the partitions are homogenous. Stratified

**Sample size** The sampling size represents a tradeoff between data reduction and precision. In a labeled dataset, it is important to consider the probability of sampling data from all the possible classes.

## 6.3 Dimensionality reduction

**Curse of dimensionality** Data with a high number of dimensions result in a sparse feature space where distance metrics are ineffective. Curse of dimensionality

**Dimensionality reduction** Useful to: Dimensionality reduction

- Avoid the curse of dimensionality.
- Reduce noise.
- Reduce the time and space complexity of mining and learning algorithms.
- Visualize multi-dimensional data.

### 6.3.1 Principal component analysis

Projection of the data into a lower-dimensional space that maximizes the variance of the data. It can be proven that this problem can be solved by finding the eigenvectors of the covariance matrix of the data. PCA

### 6.3.2 Feature subset selection

Local technique to reduce dimensionality by: Feature subset selection

- Removing redundant attributes.
- Removing irrelevant attributes.

This can be achieved by:

**Brute force** Try all the possible subsets of the dataset.

**Embedded approach** Feature selection is naturally done by the learning algorithm (e.g. decision trees).

**Filter approach** Features are filtered using domain-specific knowledge.

**Wrapper approaches** A mining algorithm is used to select the best features.

## 6.4 Feature creation

Useful to help a learning algorithm capture data characteristics. Possible approaches are: Feature creation

**Feature extraction** Features extracted from the existing ones (e.g. from a picture of a face, the eye distance can be a new feature).

**Mapping** Projecting the data into a new feature space.

**New features** Add new, possibly redundant, features.

## 6.5 Data type conversion

### 6.5.1 One-hot encoding

A discrete feature $E \in \{e_1, \dots, e_n\}$ with $n$ unique values is replaced with $n$ new binary features $H_{e_1}, \dots, H_{e_n}$ each corresponding to a value of $E$. For each entry, if its feature $E$ has value $e_i$, then $H_{e_i} =$ `true` and the rests are `false`. One-hot encoding

### 6.5.2 Ordinal encoding

A feature whose values have an ordering can be converted into a consecutive sequence of integers (e.g. ["good", "neutral", "bad"] $\mapsto$ [1, 0, -1]). Ordinal encoding

### 6.5.3 Discretization

Convert a continuous feature to a discrete one. Discretization

**Binarization** Given a continuous feature and a threshold, it can be replaced with a new binary feature that is `true` if the value is above the threshold and `false` otherwise. Binarization

**Thresholding** Same as binarization but using multiple thresholds. Thresholding

**K-bins** A continuous feature is discretized using $k$ bins each representing an integer from 0 to $k-1$. K-bins

## 6.6 Attribute transformation

Useful for normalizing features with different scales and outliers.

**Mapping** Map the domain of a feature into a new set of values (i.e. apply a function). Mapping

**Standardization** Transform a feature with Gaussian distribution into a standard distribution. Standardization

$$x = \frac{x - \mu}{\sigma}$$

**Rescaling** Map a feature into a fixed range (e.g. scale to $[0, 1]$ or $[-1, 1]$). Rescaling

**Affine transformation** Apply a linear transformation on a feature before rescaling it. This method is more robust to outliers. Affine transformation

**Normalization** Normalize each data row to unit norm. Normalization

# 7 Classification

**(Supervised) classification** Given a finite set of classes $C$ and a dataset $\boldsymbol{X}$ of $N$ individuals, each associated to a class $y(\mathbf{x}) \in C$, we want to learn a model $\mathcal{M}$ able to guess the value of $y(\bar{\mathbf{x}})$ for unseen individuals.

Classification

Classification can be:

**Crisp** Each individual has one and only one label.

Crisp classification

**Probabilistic** Each individual is assigned to a label with a certain probability.

Probabilistic classification

**Classification model** A classification model (classifier) makes a prediction by taking as input a data element $\mathbf{x}$ and a decision function $y_{\boldsymbol{\theta}}$ parametrized on $\boldsymbol{\theta}$:

Classification model

$$\mathcal{M}(\mathbf{x}, \boldsymbol{\theta}) = y_{\boldsymbol{\theta}}(\mathbf{x})$$

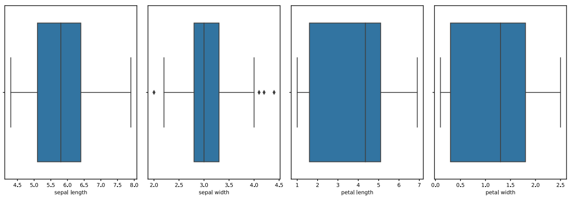
**Vapnik-Chervonenkis dimension** A dataset with $N$ elements defines $2^N$ learning problems. A model $\mathcal{M}$ has Vapnik-Chervonenkis (VC) dimension $N$ if it is able to solve all the possible learning problems with $N$ elements.

Vapnik-Chervonenkis dimension

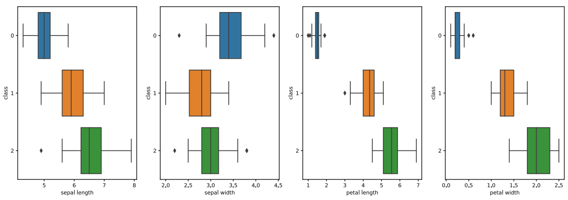
**Example.** A straight line has VC dimension 3.

### Data exploration

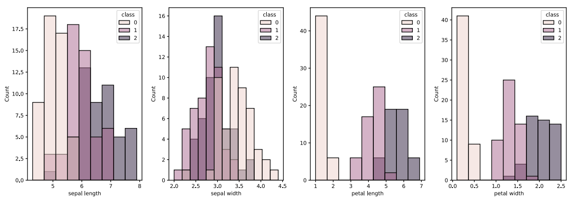
Data exploration



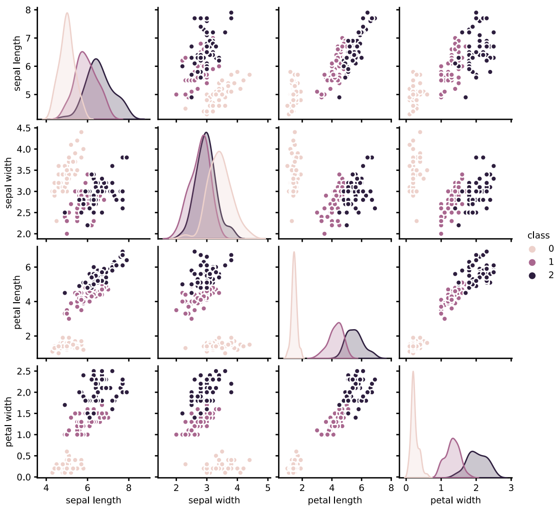
(a) Iris dataset general boxplot



(b) Iris dataset class boxplot



(c) Iris dataset histograms




(d) Iris dataset pairplots

**Hyperparameters** Parameters of the model that have to be manually chosen.

## 7.1 Evaluation

**Dataset split** A supervised dataset can be randomly split into:

**Train set** Used to learn the model. Usually the largest split. Can be seen as an upper bound of the model performance. Train set

**Test set** Used to evaluate the trained model. Can be seen as a lower bound of the model performance. Test set

**Validation set** Used to evaluate the model during training and/or for tuning parameters. Validation set

It is assumed that the splits have similar characteristics.

**Overfitting** Given a dataset $\boldsymbol{X}$, a model $\mathcal{M}$ is overfitting if there exists another model $\mathcal{M}'$ such that: Overfitting

$$\begin{gathered}\texttt{error}_{\text{train}}(\mathcal{M}) < \texttt{error}_{\text{train}}(\mathcal{M}') \\ \texttt{error}_{\boldsymbol{X}}(\mathcal{M}) > \texttt{error}_{\boldsymbol{X}}(\mathcal{M}')\end{gathered}$$

Possible causes of overfitting are:

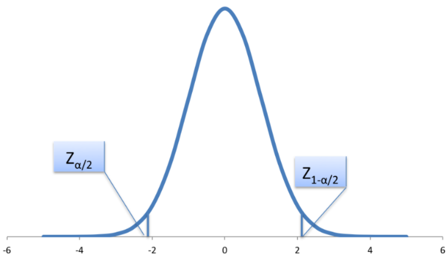
- Noisy data.
- Lack of representative instances.

### 7.1.1 Test set error

**Disclaimer: I'm very unsure about this part**
The error on the test set can be seen as a lower bound error of the model. If the test set error ratio is $x$, we can expect an error of $(x \pm \text{confidence interval})$.
Predicting the elements of the test set can be seen as a binomial process (i.e. a series of $N$ Bernoulli processes). We can therefore compute the empirical frequency of success as $f = (\text{correct predictions}/N)$. We want to estimate the probability of success $p$.
We assume that the deviation between the empirical frequency and the true frequency is due to a normal noise around the true probability (i.e. the true probability $p$ is the mean).
Fixed a confidence level $\alpha$ (i.e. the probability of a wrong estimate), we want that:

$$\mathcal{P}\left( z_{\frac{\alpha}{2}} \leq \frac{f-p}{\sqrt{\frac{1}{N}p(1-p)}} \leq z_{(1-\frac{\alpha}{2})} \right) = 1 - \alpha$$

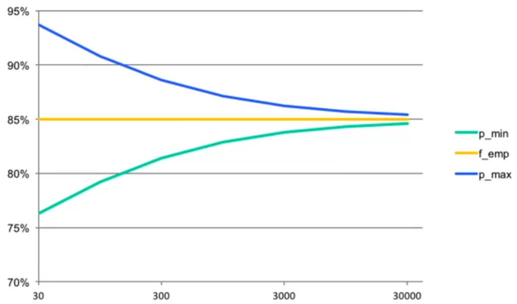
In other words, we want the middle term to have a high probability to be between the $\frac{\alpha}{2}$ and $(1-\frac{\alpha}{2})$ quantiles of the gaussian.



We can estimate $p$ using the Wilson score interval[1]:

$$p = \frac{1}{1 + \frac{1}{N}z^2} \left( f + \frac{1}{2N}z^2 \pm z\sqrt{\frac{1}{N}f(1-f) + \frac{z^2}{4N^2}} \right)$$

where $z$ depends on the value of $\alpha$. For a pessimistic estimate, $\pm$ becomes a $+$. Vice versa, for an optimistic estimate, $\pm$ becomes a $-$.
As $N$ is at the denominator, this means that for large values of $N$, the uncertainty becomes smaller.




### 7.1.2 Dataset splits

**Holdout** The dataset is split into train, test and, if needed, validation. Holdout

**Cross-validation** The training data is partitioned into $k$ chunks. For $k$ iterations, one of the chunks is used to test and the others to train a new model. The overall error is obtained as the average of the errors of the $k$ iterations. Cross-validation

In the end, the final model is still trained on the entire training data, while cross-validation results are used as an evaluation and comparison metric. Note that cross-validation is done on the training set, so a final test set can still be used to evaluate the resulting model.

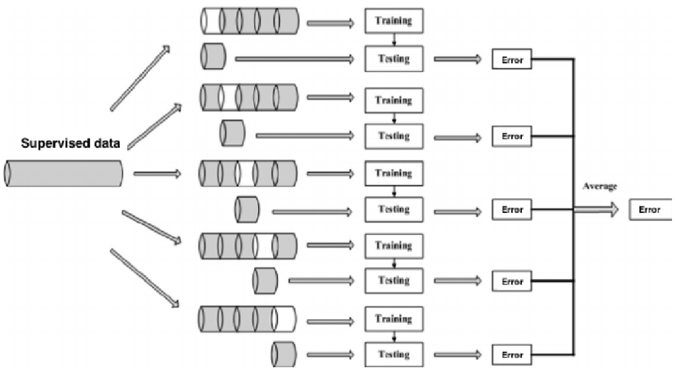



Figure 7.2: Cross-validation example

**Leave-one-out** Extreme case of cross-validation with $k = N$, the size of the training set. In this case, the whole dataset but one element is used for training and the remaining entry for testing. Leave-one-out

**Bootstrap** Statistical sampling of the dataset with replacement (i.e. an entry can be selected multiple times). The selected entries form the training set while the elements that have never been selected are used for testing. Bootstrap

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

### 7.1.3 Binary classification performance measures

In binary classification, the two classes can be distinguished as the positive and negative labels. The prediction of a classifier can be a:

True positive ($TP$) · False positive ($FP$) · True negative ($TN$) · False negative ($FN$)

| | | Predicted | |
|---|---|---|---|
| | | Pos | Neg |
| True | Pos | $TP$ | $FN$ |
| | Neg | $FP$ | $TN$ |

Given a test set of $N$ element, possible metrics are:

**Accuracy** Number of correct predictions. Accuracy

$$\text{accuracy} = \frac{TP + TN}{N}$$

**Error rate** Number of incorrect predictions. Error rate

$$\text{error rate} = 1 - \text{accuracy}$$

**Precision** Number of true positives among what the model classified as positive (i.e. how many samples the model classified as positive are real positives). Precision

$$\text{precision} = \frac{TP}{TP + FP}$$

**Recall/Sensitivity** Number of true positives among the real positives (i.e. how many real positives the model predicted). Recall

$$\text{recall} = \frac{TP}{TP + FN}$$

**Specificity** Number of true negatives among the real negatives (i.e. recall for negative labels). Specificity

$$\text{specificity} = \frac{TN}{TN + FP}$$

**F1 score** Harmonic mean of precision and recall (i.e. measure of balance between precision and recall). F1 score

$$\text{F1} = 2\frac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}}$$

### 7.1.4 Multi-class classification performance measures

**Confusion matrix** Matrix to correlate the predictions of $n$ classes: Confusion matrix

| | | Predicted | | | |
|---|---|---|---|---|---|
| | | a | b | c | Total |
| True | a | $TP_a$ | $FP_{a-b}$ | $FP_{a-c}$ | $T_a$ |
| | b | $FP_{b-a}$ | $TP_b$ | $FP_{b-c}$ | $T_b$ |
| | c | $FP_{c-a}$ | $FP_{c-b}$ | $TP_c$ | $T_c$ |
| | Total | $P_a$ | $P_b$ | $P_c$ | $N$ |

where:

- $a$, $b$ and $c$ are the classes.
- $T_x$ is the true number of labels of class $x$ in the dataset.
- $P_x$ is the predicted number of labels of class $x$ in the dataset.
- $TP_x$ is the number of times a class $x$ was correctly predicted (true predictions).
- $FP_{i-j}$ is the number of times a class $i$ was predicted as $j$ (false predictions).

**Accuracy** Accuracy is extended from the binary case as: Accuracy

$$\text{accuracy} = \frac{\sum_i TP_i}{N}$$

**Precision** Precision is defined w.r.t. a single class: Precision

$$\text{precision}_i = \frac{TP_i}{P_i}$$

**Recall** Recall is defined w.r.t. a single class: Recall

$$\text{recall}_i = \frac{TP_i}{T_i}$$

If a single value of precision or recall is needed, the mean can be used by computing a macro (unweighted) average or a class-weighted average.

**$\kappa$-statistic** Evaluates the concordance between two classifiers (in our case, the predictor and the ground truth). It is based on two probabilities: $\kappa$-statistic

**Probability of concordance** $\mathcal{P}(c) = \frac{\sum_i^{\texttt{classes}} TP_i}{N}$

**Probability of random concordance** $\mathcal{P}(r) = \frac{\sum_i^{\texttt{classes}} T_i P_i}{N^2}$

$\kappa$-statistic is given by:

$$\kappa = \frac{\mathcal{P}(c) - \mathcal{P}(r)}{1 - \mathcal{P}(r)} \in [-1, 1]$$

When $\kappa = 1$, there is perfect agreement ($\sum_i^{\texttt{classes}} TP_i = 1$), when $\kappa = -1$, there is total disagreement ($\sum_i^{\texttt{classes}} TP_i = 0$) and when $\kappa = 0$, there is random agreement.

### 7.1.5 Probabilistic classifier performance measures

**Lift chart** Used in binary classification. Given the resulting probabilities of the positive class of a classifier, sort them in decreasing order and plot a 2d-chart with increasing sample size on the x-axis and the number of positive samples on the y-axis. Lift chart

Then, plot a straight line to represent a baseline classifier that makes random choices. As the probabilities are sorted in decreasing order, it is expected a high concentration of positive labels on the right side. When the area between the two curves is large and the curve is above the random classifier, the model can be considered a good classifier.

**ROC curve** The ROC curve can be seen as a way to represent multiple confusion matrices of a classifier that uses different thresholds. The x-axis of a ROC curve represents the false positive rate while the y-axis represents the true positive rate. ROC curve

A straight line is used to represent a random classifier. A threshold can be considered good if it is high on the y-axis and low on the x-axis.

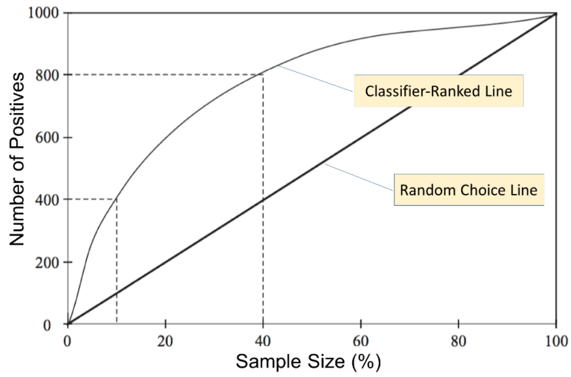



Figure 7.3: Example of lift chart

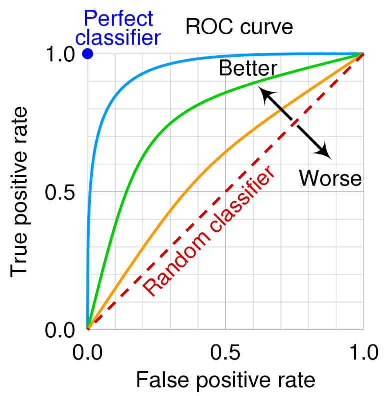



Figure 7.4: Example of ROC curves

### 7.1.6 Data imbalance

A classifier may not perform well when predicting a minority class of the training data. Possible solutions are:

**Undersampling** Randomly reduce the number of examples of the majority classes. Undersampling

**Oversampling** Increase the examples of the minority classes. Oversampling

**Synthetic minority oversampling technique (SMOTE)** SMOTE

1. Randomly select an example $x$ belonging to the minority class.
2. Select a random neighbor $z_i$ among its $k$-nearest neighbors $z_1, \dots, z_k$.
3. Synthesize a new example by selecting a random point of the feature space between $x$ and $z_i$.

**Cost sensitive learning** Assign a cost to the errors. Higher weights are assigned to minority classes. This can be done by: Cost sensitive learning

- Altering the proportions of the dataset by duplicating samples to reduce its misclassification.
- Weighting the classes (possible in some algorithms).

## 7.2 Decision trees

### 7.2.1 Information theory

**Shannon theorem** Let $\boldsymbol{X} = \{\mathbf{v}_1, \dots, \mathbf{v}_V\}$ be a data source where each of the possible values has probability $p_i = \mathcal{P}(\mathbf{v}_i)$. The best encoding allows to transmit $\boldsymbol{X}$ with an average number of bits given by the **entropy** of $X$:

Shannon theorem

Entropy

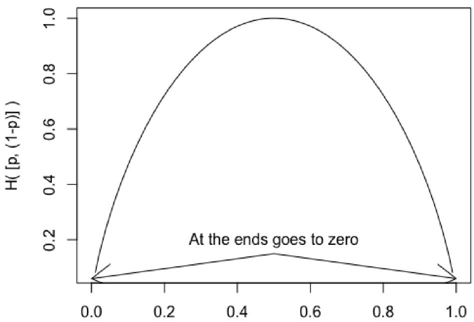
$$H(\boldsymbol{X}) = -\sum_j p_j \log_2(p_j)$$

$H(\boldsymbol{X})$ can be seen as a weighted sum of the surprise factor $-\log_2(p_j)$. If $p_j \sim 1$, then the surprise of observing $\mathbf{v}_j$ is low, vice versa, if $p_j \sim 0$, the surprise of observing $\mathbf{v}_j$ is high.

Therefore, when $H(\boldsymbol{X})$ is high, $\boldsymbol{X}$ is close to a uniform distribution. When $H(\boldsymbol{X})$ is low, $\boldsymbol{X}$ is close to a constant.

**Example** (Binary source)**.**

The two values of a binary source $\boldsymbol{X}$ have respectively probability $p$ and $(1 - p)$. When $p \sim 0$ or $p \sim 1$, $H(\boldsymbol{X}) \sim 0$.
When $p \sim 0.5$, $H(\boldsymbol{X}) \sim \log_2(2) = 1$




**Entropy threshold split** Given a dataset $\boldsymbol{D}$, a real-valued attribute $d \in \boldsymbol{D}$, a threshold $t$ in the domain of $d$ and the class attribute $c$ of $\boldsymbol{D}$. The entropy of the class $c$ of the dataset $\boldsymbol{D}$ split with threshold $t$ on $d$ is a weighted sum:

Entropy threshold split

$$H(c \mid d : t) = \mathcal{P}(d < t) H(c \mid d < t) + \mathcal{P}(d \geq t) H(c \mid d \geq t)$$

**Information gain** Information gain measures the reduction in entropy after applying a split. It is computed as:

Information gain

$$IG(c \mid d : t) = H(c) - H(c \mid d : t)$$

When $H(c \mid d : t)$ is low, $IG(c \mid d : t)$ is high as splitting with threshold $t$ results in purer groups. Vice versa, when $H(c \mid d : t)$ is high, $IG(c \mid d : t)$ is low as splitting with threshold $t$ is not very useful.

The information gain of a class $c$ split on a feature $d$ is given by:

$$IG(c \mid d) = \max_t IG(c \mid d : t)$$

### 7.2.2 Tree construction

**Decision tree (C4.5)** Tree-shaped classifier where leaves are class predictions and inner nodes represent conditions that guide to a leaf. This type of classifier is non-linear (i.e. does not represent a linear separation).

Decision tree

Each node of the tree contains:

- The applied splitting criteria (i.e. feature and threshold). Leaves do not have this value.
- The purity (e.g. entropy) of the current split.
- Dataset coverage of the current split.
- Classes distribution.

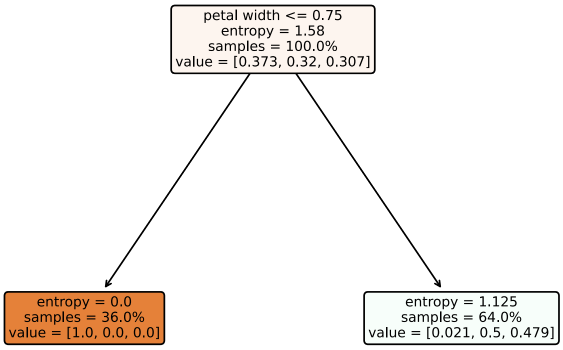



Figure 7.5: Example of decision tree

Note: the weighted sum of the entropies of the children is always smaller than the entropy of the parent.

Possible stopping conditions are:

- When most of the leaves are pure (i.e. nothing useful to split).
- When some leaves are impure but none of the possible splits have positive $IG$. Impure leaves are labeled with the majority class.

**Purity** Value to maximize when splitting a node of a decision tree. Purity

Nodes with uniformly distributed classes have a low purity. Nodes with a single class have the highest purity.

Possible impurity measures are:

**Entropy/Information gain** See Section 7.2.1.

**Gini index** Let $\boldsymbol{X}$ be a dataset with classes $C$. The Gini index measures how often an element of $\boldsymbol{X}$ would be misclassified if the labels were randomly assigned based on the frequencies of the classes in $\boldsymbol{X}$. Gini index

Given a class $i \in C$, $p_i$ is the probability (i.e. frequency) of classifying an element with $i$ and $(1 - p_i)$ is the probability of classifying it with a different label. The Gini index is given by:

$$\begin{aligned} GINI(\boldsymbol{X}) &= \sum_{i}^{C} p_i(1 - p_i) = \sum_{i}^{C} p_i - \sum_{i}^{C} p_i^2 \\ &= 1 - \sum_{i}^{C} p_i^2 \end{aligned}$$

When $\boldsymbol{X}$ is uniformly distributed, $GINI(\boldsymbol{X}) \sim (1 - \frac{1}{|C|})$. When $\boldsymbol{X}$ is constant, $GINI(\boldsymbol{X}) \sim 0$.

Given a node $x$ split in $n$ children $x_1, \dots, x_n$, the Gini gain of the split is given by:

$$GINI_{\text{gain}} = GINI(x) - \sum_{i=1}^{n} \frac{|x_i|}{|x|} GINI(x_i)$$

**Misclassification error** Skipped. Misclassification error

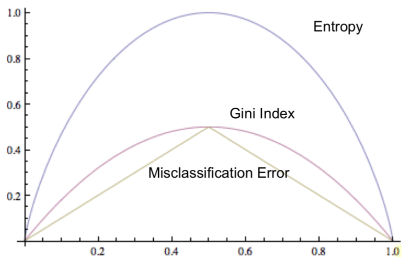



Figure 7.6: Comparison of impurity measures

Compared to Gini index, entropy is more robust to noise.

Misclassification error has a bias toward the major class.

**Algorithm 1** Decision tree construction using information gain as impurity measure

```
def buildTree(split):
    node = Node()
    if len(split.classes) == 1: # Pure split
        node.label = split.classes[0]
        node.isLeaf = True
    else:
        ig, attribute, threshold = getMaxInformationGain(split)
        if ig < 0:
            node.label = split.majorityClass()
            node.isLeaf = True
        else:
            node.left = buildTree(split[attribute < threshold])
            node.right = buildTree(split[attribute >= threshold])
    return node
```

**Pruning** Remove branches to reduce overfitting. Different pruning techniques can be employed: Pruning

**Maximum depth** Maximum depth allowed for the tree.

**Minimum samples for split** Minimum number of samples a node is required to have to apply a split.

**Minimum samples for a leaf** Minimum number of samples a node is required to have to become a leaf.

**Minimum impurity decrease** Minimum decrease in impurity for a split to be made.

**Statistical pruning** Prune the children of a node if the weighted sum of the maximum errors of the children is greater than the maximum error of the node if it was a leaf.

### 7.2.3 Complexity

Given a dataset $\boldsymbol{X}$ of $N$ instances and $D$ attributes, each level of the tree requires to evaluate the whole dataset and each node requires to process all the attributes. Assuming an average height of $O(\log N)$, the overall complexity for induction (parameters search) is $O(DN \log N)$.
Moreover, the other operations of a binary tree have complexity:

- Threshold search and binary split: $O(N \log N)$ (scan the dataset for the threshold).
- Pruning: $O(N \log N)$ (requires to scan the dataset).

For inference, to classify a new instance it is sufficient to traverse the tree from the root to a leaf. This has complexity $O(h)$, with $h$ the height of the tree.

### 7.2.4 Characteristics

- Decision trees are non-parametric in the sense that they do not require any assumption on the distribution of the data.
- Finding the best tree is an NP-complete problem.
- Decision trees are robust to noise if appropriate overfitting methods are applied.
- Decision trees are robust to redundant attributes (correlated attributes are very unlikely to be chosen for multiple splits).
- In practice, the impurity measure has a low impact on the final result, while the pruning strategy is more relevant.

## 7.3 Naive Bayes

**Bayes' theorem** Given a class $c$ and the evidence $\mathbf{e}$, we have that:

$$\mathcal{P}(c \mid \mathbf{e}) = \frac{\mathcal{P}(\mathbf{e} \mid c)\mathcal{P}(c)}{\mathcal{P}(\mathbf{e})}$$

**Naive Bayes classifier** Classifier that uses the Bayes' theorem assuming that the attributes are independent given the class. Given a class $c$ and the evidence $\mathbf{e} = \langle e_1, e_2, \dots, e_n \rangle$, the probability that the observation $\mathbf{e}$ is of class $c$ is given by: Naive Bayes classifier

$$\mathcal{P}(c \mid \mathbf{e}) = \frac{\prod_{i=1}^{n} \mathcal{P}(e_i \mid c) \cdot \mathcal{P}(c)}{\mathcal{P}(\mathbf{e})}$$

As the denominator is the same for all classes, it can be omitted.

### 7.3.1 Training and inference

**Training** Given the classes $C$ and the features $E$, to train the classifier the following priors need to be estimated: Training

- $\forall c \in C : \mathcal{P}(c)$
- $\forall e_{ij} \in E, \forall c \in C : \mathcal{P}(e_{ij} \mid c)$, where $e_{ij}$ is the $j$-th value of the domain of the $i$-th feature $E_i$.

**Inference** Given a new observation $\mathbf{x}_{\text{new}} = \langle x_1, x_2, \dots, x_n \rangle$, its class is determined by computing the likelihood:

Inference

$$c_{\text{new}} = \arg\max_{c \in C} \mathcal{P}(c) \prod_{i=1}^{n} \mathcal{P}(x_i \mid c)$$

### 7.3.2 Problems

**Smooting** If the value $e_{ij}$ of the domain of a feature $E_i$ never appears in the dataset, its probability $\mathcal{P}(e_{ij} \mid c)$ will be 0 for all classes. This nullifies all the probabilities that use this feature when computing the chain of products during inference. Smoothing methods can be used to avoid this problem.

**Laplace smoothing** Given:

Laplace smoothing

$\alpha$ The smoothing factor.

$\text{af}_{e_{ij},c}$ The absolute frequency of the value $e_{ij}$ of the feature $E_i$ over the class $c$.

$|\mathbb{D}_{E_i}|$ The number of distinct values in the domain of $E_i$.

$\text{af}_c$ The absolute frequency of the class $c$.

The smoothed frequency is computed as:

$$\mathcal{P}(e_{ij} \mid c) = \frac{\text{af}_{e_{ij},c} + \alpha}{\text{af}_c + \alpha|\mathbb{D}_{E_i}|}$$

A common value of $\alpha$ is 1. When $\alpha = 0$, there is no smoothing. For higher values of $\alpha$, the smoothed feature gains more importance when computing the priors.

**Missing values** Naive Bayes is robust to missing values.

Missing values

During training, the record is ignored in the frequency count of the missing feature.

During inference, the missing feature can be simply excluded in the computation of the likelihood as this equally affects all classes.

**Numeric values** For continuous numeric values, the frequency count method cannot be used. Therefore, an additional assumption is made: numeric values follow a Gaussian distribution.

Gaussian assumption

During training, the mean $\mu_{i,c}$ and variance $\sigma_{i,c}$ for a numeric feature $E_i$ is computed with respect to a class $c$. Its probability is then obtained as:

$$\mathcal{P}(E_i = x \mid c) = \mathcal{N}(\mu_{i,c}, \sigma_{i,c})(x)$$

## 7.4 Perceptron

**Perceptron** A single artificial neuron that takes $n$ inputs $x_1, \dots, x_n$ and a bias $b$, and computes a linear combination of them with weights $w_1, \dots, w_n, w_b$.

Perceptron

The learnt weights $w_b, w_1, \dots, w_n$ define a hyperplane for binary classification such that:

$$w_1 x_1 + \dots + w_n x_n + w_b b = \begin{cases} \texttt{positive} & \text{if } > 0 \\ \texttt{negative} & \text{if } < 0 \end{cases}$$

It can be shown that there are either none or infinite hyperplanes with this property.

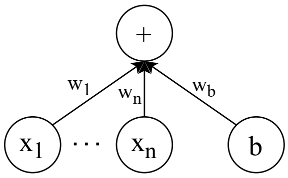



Figure 7.7: Example of perceptron

### 7.4.1 Training

**Algorithm 2** Perceptron training

```
def trainPerceptron(dataset):
    perceptron = Perceptron(weights=[0 ... 0])

    while accuracy(perceptron, dataset) != 1.0:
        for x, y in dataset:
            if perceptron.predict(x) != y:
                if y is positive_class:
                    perceptron.weights += x
                else:
                    perceptron.weights -= x
```

Note that the algorithm converges only if the dataset is linearly separable. In practice, a maximum number of iterations is set.

## 7.5 Support vector machine

**Convex hull** The convex hull of a set of points is the tightest enclosing convex polygon that contains those points.

Note: the convex hulls of a linearly separable dataset do not intersect.

**Maximum margin hyperplane** Hyperplane with the maximum margin between two convex hulls.

Maximum margin hyperplane

In general, a subset of points (support vectors) in the training set is sufficient to define the hulls.

Support vectors

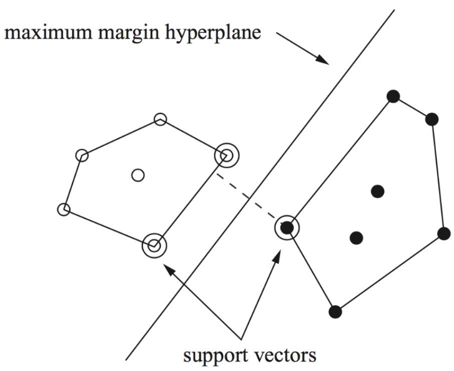



Figure 7.8: Maximum margin hyperplane of linearly separable data

**Support vector machine** SVM[2] finds the maximum margin hyperplane and the support vectors as a constrained quadratic optimization problem. Given a dataset of $D$ elements and $n$ features, the problem is defined as:

Support vector machine

$$\max_{w_0, w_1, \dots, w_n} M$$

$$\text{subject to } \sum_{i=1}^{n} w_i^2 = 1$$

$$c_i(w_0 + w_1 x_{i1} + \cdots + w_n x_{in}) \geq M \;\; \forall i = 1, \dots, D$$

where $M$ is the margin, $w_i$ are the weights of the hyperplane and $c_i = \{-1, 1\}$ is the class. The second constraint imposes the hyperplane to have a large margin. For positive labels ($c_i = 1$), this is true when the hyperplane is positive. For negative labels ($c_i = -1$), this is true when the hyperplane is negative.

**Soft margin** As real-world data is not always linearly separable, soft margin relaxes the margin constraint by adding a penalty $C$. The margin constraint becomes:

Soft margin

$$c_i(w_0 + w_1 x_{i1} + \cdots + w_n x_{in}) \geq M - \xi_i \;\; \forall i = 1, \dots, D$$

$$\text{where } \xi_i \geq 0 \text{ and } \sum_{i=0}^{D} \xi_i = C$$

### 7.5.1 Kernel trick

For non-linearly separable data, the boundary can be found using a non-linear mapping to map the data into a new space (feature space) where a linear separation is possible. Then, the data and the boundary is mapped back into the original space.

Kernel trick

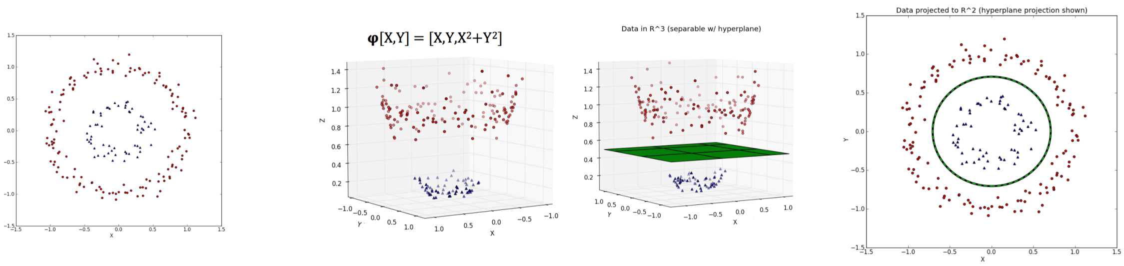



Figure 7.9: Example of mapping from $\mathbb{R}^2$ to $\mathbb{R}^3$

The kernel trick allows to avoid explicitly mapping the dataset into the new space by using kernel functions. Known kernel functions are:

**Linear** $K(x, y) = \langle x, y \rangle$.

**Polynomial** $K(x, y) = (\gamma \langle x, y \rangle + r)^d$, where $\gamma$, $r$ and $d$ are parameters.

**Radial based function** $K(x, y) = \exp(-\gamma \|x - y\|^2)$, where $\gamma$ is a parameter.

**Sigmoid** $K(x, y) = \tanh(\langle x, y \rangle + r)$, where $r$ is a parameter.

[2] `https://www.cs.princeton.edu/courses/archive/spring16/cos495/slides/AndrewNg_SVM_note.pdf`

### 7.5.2 Complexity

Given a dataset with $D$ entries of $n$ features, the complexity of SVM scales from $O(nD^2)$ to $O(nD^3)$ depending on the effectiveness of data caching.

### 7.5.3 Characteristics

- Training an SVM model is generally slower.
- SVM is not affected by local minimums.
- SVM does not suffer the curse of dimensionality.
- SVM does not directly provide probability estimates. If needed, these can be computed using a computationally expensive method.

## 7.6 Neural networks

**Multilayer perceptron** Hierarchical structure of perceptrons, each with an activation function. Multilayer perceptron

**Activation function** Activation functions are useful to add non-linearity. Activation function

**Remark.** In a linear system, if there is noise in the input, it is transferred to the output (i.e. linearity implies that $f(x+\text{noise}) = f(x)+f(\text{noise})$). On the other hand, a non-linear system is generally more robust (i.e. non-linearity generally implies that $f(x + \text{noise}) \neq f(x) + f(\text{noise})$)

**Feedforward neural network** Network with the following flow: Feedforward neural network

$$\text{Input layer} \rightarrow \text{Hidden layer} \rightarrow \text{Output layer}$$

Neurons at each layer are connected to all neurons of the next layer.

### 7.6.1 Training

Inputs are fed to the network and backpropagation is used to update the weights.

**Learning rate** Size of the step for gradient descent. Learning rate

**Epoch** A round of training where the entire dataset is processed. Epoch

**Stopping criteria** Possible conditions to stop the training are: Stopping criteria

- Small weights update.
- The classification error goes below a predefined target.
- Timeout or maximum number of epochs.

**Regularization** Smoothing of the loss function. Regularization

## 7.7 K-nearest neighbors

**K-nearest neighbors** Given a similarity metric and a training set, to predict a new observation, the $k$ most similar entries in the training set are selected and the class of the new data is determined as the most frequent class among the $k$ entries. K-nearest neighbors

## 7.8 Binary to multi-class classification

One-vs-one strategy (OVO)

**One-vs-one strategy (OVO)** Train a classifier for all the possible pairs of classes (this will result in $\frac{C \cdot (C-1)}{2}$ pairs). The class assigned to a new observation is determined through a majority vote.

One-vs-rest strategy (OVR)

**One-vs-rest strategy (OVR)** Train $C$ classifiers where each is specialized to classify a specific class as positive and the others as negative. The class assigned to a new observation is determined by the confidence score of each classifier.

## 7.9 Ensemble methods

Ensemble methods

Train a set of base classifiers and make predictions by majority vote. If all the classifiers have the same but independent error rate, the overall error of the ensemble model is lower (derived from a binomial distribution).

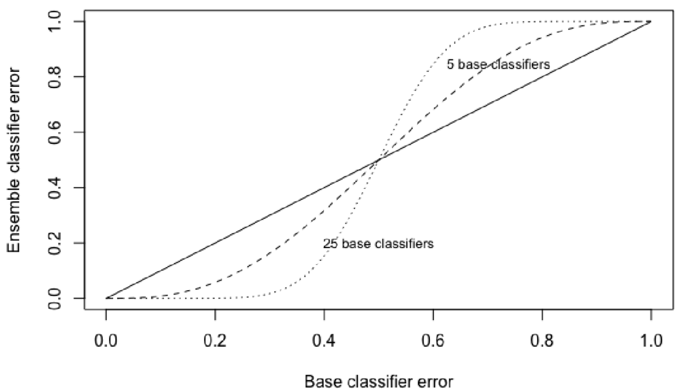



Figure 7.10: Relationship between the error of base classifiers and ensemble models

Different strategies to train an ensemble classifier can be used:

**Dataset manipulation** Resampling the dataset for each base classifier:

- **Bagging** Sample with replacement with a uniform distribution.
- **Boosting** Iteratively change the distribution of the training data prioritizing examples difficult to classify.
  - Adaboost
  - **Adaboost** Iteratively train base classifiers on a dataset where samples misclassified at the previous iteration have a higher weight.

**Feature manipulation** Train a base classifier using only a subset of the features.

**Class labels manipulation** Train a base classifier to classify a partition of the class labels. For instance, class labels can be partitioned into two groups $A_1$ and $A_2$, and the base classifier is trained to assign as label one of the two groups. During inference, when a group is predicted, all labels within that group receive a vote.

### 7.9.1 Random forests

Random forests

Multiple decision trees trained on a different random sampling of the training set and different subsets of features. A prediction is made by averaging the output of each tree.

**Bias** Simplicity of the target function of a model. Bias

**Variance** Amount of change of the target function when using different training data (i.e. how much the model overfits). Variance

Random forests aim to reduce the high variance of decision trees.

# 8 Regression

**Linear regression** Given: Linear regression

- A dataset $\boldsymbol{X}$ of $N$ rows and $D$ features.
- A response vector $\mathbf{y}$ of $N$ continuous values.

We want to learn the parameters $\mathbf{w} \in \mathbb{R}^D$ such that:

$$\mathbf{y} \approx \boldsymbol{X}\mathbf{w}^T$$

**Mean squared error** To find the parameters for linear regression, we minimize as loss function the mean squared error: Mean squared error

$$\mathcal{L}(\mathbf{w}) = \|\boldsymbol{X}\mathbf{w}^T - \mathbf{y}\|^2$$

Its gradient is:

$$\nabla\mathcal{L}(\mathbf{w}) = 2\boldsymbol{X}^T(\boldsymbol{X}\mathbf{w}^T - \mathbf{y})$$

Constraining it to 0, we obtain the problem:

$$\boldsymbol{X}^T\boldsymbol{X}\mathbf{w}^T = \boldsymbol{X}^T\mathbf{y}$$

If $\boldsymbol{X}^T\boldsymbol{X}$ is invertible, this can be solved analytically but could lead to overfitting. Numerical methods are therefore more suited.

Note that:

- MSE is influenced by the magnitude of the data.
- It measures the fitness of a model in absolute terms.

**Coefficient of determination** Given: Coefficient of determination

- The mean of the observed data: $y_{\text{avg}} = \frac{1}{N}\sum_i \mathbf{y}_i$.
- The sum of the squared residuals: $SS_{\text{res}} = \sum_i(\mathbf{y}_i - \mathbf{w}^T\mathbf{x}_i)^2$.
- The total sum of squares: $SS_{\text{tot}} = \sum_i(\mathbf{y}_i - y_{\text{avg}})^2$.

The coefficient of determination is given by:

$$\mathrm{R}^2 = 1 - \frac{SS_{\text{res}}}{SS_{\text{tot}}}$$

Intuitively, $\mathrm{R}^2$ compares the model with a horizontal straight line ($y_{\text{avg}}$). When $\mathrm{R}^2 = 1$, the model has a perfect fit. When $\mathrm{R}^2$ is outside the range $[0, 1]$, then the model is worse than a straight line.

Note that:

- $\mathrm{R}^2$ is a standardized index.
- $\mathrm{R}^2$ tells how well the variables of the predictor can explain the variation in the target.
- $\mathrm{R}^2$ is not suited for non-linear models.

**Polynomial regression** Find a polynomial instead of a hyperplane. Polynomial regression

# 9 Clustering

## 9.1 Similarity and dissimilarity

**Similarity** Measures how alike two objects are. Often defined in the range $[0, 1]$. Similarity

**Dissimilarity** Measures how two objects differ. 0 indicates no difference while the upper bound varies. Dissimilarity

| Attribute type | Dissimilarity | Similarity |
|---|---|---|
| Nominal | $d(p,q) = \begin{cases} 0 & \text{if } p = q \\ 1 & \text{if } p \neq q \end{cases}$ | $s(p,q) = 1 - d(p,q)$ |
| Ordinal | $d(p,q) = \frac{\lvert p-q \rvert}{V}$ with $p, q \in \{0, \dots, V\}$ | $s(p,q) = 1 - d(p,q)$ |
| Interval or ratio | $d(p,q) = \lvert p - q \rvert$ | $s(p,q) = \frac{1}{1+d(p,q)}$ |

Table 9.1: Similarity and dissimilarity by attribute type

**Similarity properties**

1. $\texttt{sim}(p,q) = 1$ iff $p = q$.
2. $\texttt{sim}(p,q) = \texttt{sim}(q,p)$.

### 9.1.1 Distance

Given two $D$-dimensional data entries $p$ and $q$, possible distance metrics are:

**Minkowski distance ($L_r$)** Minkowski distance

$$\texttt{dist}(p,q) = \left( \sum_{d=1}^{D} \lvert p_d - q_d \rvert^r \right)^{\frac{1}{r}}$$

where $r$ is a parameter.

Common values for $r$ are:

$r = 1$ Corresponds to the $L_1$ norm. It is useful for discriminating 0 distance and near-0 distance as an $\varepsilon$ change in the data corresponds to an $\varepsilon$ change in the distance.

$r = 2$ Corresponds to the Euclidean distance or $L_2$ norm.

$r = \infty$ Corresponds to the $L_\infty$ norm. Considers only the dimensions with the maximum difference.

**Mahalanobis distance**

Mahalanobis distance

$$\texttt{dist}(p, q) = \sqrt{(p - q)\boldsymbol{\Sigma}^{-1}(p - q)^T}$$

where $\boldsymbol{\Sigma}$ is the covariance matrix of the dataset. The Mahalanobis distance of $p$ and $q$ increases when the segment connecting them points towards a direction of greater variation of the data.

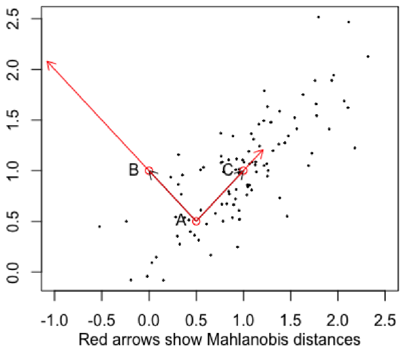



Figure 9.1: The Mahalanobis distance between $(A, B)$ is greater than $(A, C)$, while the Euclidean distance is the same.

**Distance properties**

**Positive definiteness** $\texttt{dist}(p, q) \geq 0$ and $\texttt{dist}(p, q) = 0$ iff $p = q$.

**Symmetry** $\texttt{dist}(p, q) = \texttt{dist}(q, p)$

**Triangle inequality** $\texttt{dist}(p, q) \leq \texttt{dist}(p, r) + \texttt{dist}(r, q)$

### 9.1.2 Vector similarity

**Binary vectors** Given two examples $p$ and $q$ with binary features, we can compute the following values:

$$\begin{aligned}
M_{00} &= \text{ number of features that equals to 0 for both } p \text{ and } q \\
M_{01} &= \text{ number of features that equals to 0 for } p \text{ and 1 for } q \\
M_{10} &= \text{ number of features that equals to 1 for } p \text{ and 0 for } q \\
M_{11} &= \text{ number of features that equals to 1 for both } p \text{ and } q
\end{aligned}$$

Possible distance metrics are:

**Simple matching coefficient** $\texttt{SMC}(p, q) = \frac{M_{00}+M_{11}}{M_{00}+M_{01}+M_{10}+M_{11}}$

Simple matching coefficient

**Jaccard coefficient** $\texttt{JC}(p, q) = \frac{M_{11}}{M_{01}+M_{10}+M_{11}}$

Jaccard coefficient

**Cosine similarity** Cosine of the angle between two vectors:

Cosine similarity

$$\cos(p, q) = \frac{p \cdot q}{\|p\| \cdot \|q\|}$$

**Extended Jaccard coefficient (Tanimoto)** Variation of the Jaccard coefficient for continuous values:

Extended Jaccard coefficient (Tanimoto)

$$\texttt{T}(p, q) = \frac{p \cdot q}{\|p\|^2 + \|q\|^2 - p \cdot q}$$

### 9.1.3 Correlation

**Pearson's correlation** Measure of linear relationship between a pair of quantitative attributes $e_1$ and $e_2$. To compute Pearson's correlation, the values of $e_1$ and $e_2$ are first standardized and then ordered to obtain the vectors $\mathbf{e}_1$ and $\mathbf{e}_2$. The correlation is then computed as the dot product between $\mathbf{e}_1$ and $\mathbf{e}_2$: Pearson's correlation

$$\texttt{corr}(e_1, e_2) = \langle \mathbf{e}_1, \mathbf{e}_2 \rangle$$

Pearson's correlation has the following properties:

- If the variables are independent, then the correlation is 0 (but not vice versa).
- If the correlation is 0, then there is no linear relationship between the variables.
- $+1$ implies positive linear relationship, $-1$ implies negative linear relationship.

**Symmetric uncertainty** Measure of correlation for nominal attributes: Symmetric uncertainty

$$U(e_1, e_2) = 2\frac{H(e_1) + H(e_2) - H(e_1, e_2)}{H(e_1) + H(e_2)} \in [0, 1]$$

where $H$ is the entropy.

## 9.2 Clustering definitions

**Clustering** Given a set of $D$-dimensional objects $\mathbf{x}_i$, we want to partition them into $K$ clusters (and potentially recognize outliers). In other words, we are looking for a mapping: Clustering

$$\texttt{cluster}(\mathbf{x}_i) \in \{1, \dots, K\}$$

such that objects in the same cluster are similar.

**Centroid** Average of the coordinates of the points in a cluster. For a cluster $K_i$, the $d$-th coordinate of its centroid is given by: Centroid

$$\texttt{centroid}(K_i)\texttt{[}d\texttt{]} = \frac{1}{|K_i|} \sum_{\mathbf{x} \in K_i} \mathbf{x}\texttt{[}d\texttt{]}$$

**Medoid** Element of the cluster with minimum average dissimilarity to all other points. Differently from the centroid, the medoid must be an existing point of the dataset. Medoid

**Proximity functions** Measures to determine the similarity of two data points: Proximity function

**Euclidean distance**

## 9.3 Metrics

**Cohesion** Measures the similarity (proximity) of the objects in the same cluster. Given a cluster $K_i$, cohesion is computed as: Cohesion

$$\texttt{cohesion}(K_i) = \sum_{\mathbf{x} \in K_i} \texttt{dist}(\mathbf{x}, \mathbf{c}_i)$$

where $\mathbf{c}_i$ can be the centroid or medoid and `dist` is a proximity function.

**Separation** Measures the distance of two clusters. Given two clusters $K_i$ and $K_j$, their separation is:

Separation

$$\texttt{separation}(K_i, K_j) = \texttt{dist}(\mathbf{c}_i, \mathbf{c}_j)$$

where $\mathbf{c}_i$ and $\mathbf{c}_j$ are respectively the centroids of $K_i$ and $K_j$, and `dist` is a proximity function.

**Sum of squared errors** Measures for each cluster the distance between its points to its centroid. Can be seen as the application of distortion (Section 9.4) to clustering:

Sum of squared errors

$$\texttt{SSE}_j = \sum_{\mathbf{x}_i \in K_j} \texttt{dist}(\mathbf{x}_i, \mathbf{c}_j)^2$$

where $K_j$ is the $j$-th cluster and $\mathbf{c}_j$ is its centroid.

If $\texttt{SSE}_j$ is high, the cluster has low quality. If $\texttt{SSE}_j = 0$, all points in the cluster correspond to the centroid.

The sum of squared errors of $K$ clusters is:

$$\texttt{SSE} = \sum_{j=1}^{K} \texttt{SSE}_j$$

**Sum of squares between clusters** Given the global centroid of the dataset $\mathbf{c}$ and $K$ clusters each with $N_i$ objects, the sum of squares between clusters is given by:

Sum of squares between clusters

$$\texttt{SSB} = \sum_{i=1}^{K} N_i \cdot \texttt{dist}(\mathbf{c}_i, \mathbf{c})^2$$

**Total sum of squares** Sum of the squared distances between the points of the dataset and the global centroid. It can be shown that the total sum of squares can be computed as:

Total sum of squares

$$\texttt{TSS} = \texttt{SSE} + \texttt{SSB}$$

**Theorem 9.3.1.** Minimize `SSE` $\iff$ maximize `SSB`.

**Silhouette score** The Silhouette score of a data point $\mathbf{x}_i$ belonging to a cluster $K_i$ is given by two components:

Silhouette score

**Sparsity contribution** The average distance of $\mathbf{x}_i$ to the other points in $K_i$:

$$a(\mathbf{x}_i) = \frac{1}{|K_i| - 1} \sum_{\mathbf{x}_j \in K_i, \mathbf{x}_j \neq \mathbf{x}_i} \texttt{dist}(\mathbf{x}_i, \mathbf{x}_j)$$

**Separation contribution** The average distance of $\mathbf{x}_i$ to the points in the nearest cluster:

$$b(\mathbf{x}_i) = \min_{K_j, K_j \neq K_i} \left( \frac{1}{|K_j|} \sum_{\mathbf{w} \in K_j} \texttt{dist}(\mathbf{x}_i, \mathbf{w}) \right)$$

The Silhouette score of $\mathbf{x}_i$ is then computed as:

$$s(\mathbf{x}_i) = \frac{b(\mathbf{x}_i) - a(\mathbf{x}_i)}{\max\{a(\mathbf{x}_i), b(\mathbf{x}_i)\}} \in [-1, 1]$$

The Silhouette score $\mathcal{S}$ of $K$ clusters is given by the average Silhouette scores of each data point. $\mathcal{S} \to 1$ indicates correct clusters, $\mathcal{S} \to -1$ indicates incorrect clusters.

**Golden standard** Evaluation using a labeled dataset. Consider the elements of the same cluster as labeled with the same class. Golden standard

**Classification-oriented** Traditional classification metrics such as accuracy, recall, precision, ...

**Similarity-oriented** Given a learnt clustering scheme $y_K(\cdot)$ and the golden standard scheme $y_G(\cdot)$ where $y_i(\mathbf{x})$ indicates the label/cluster of $\mathbf{x}$, each pair of data $(\mathbf{x}_1, \mathbf{x}_2)$ can be labeled with:

`SGSK` if $y_G(\mathbf{x}_1) = y_G(\mathbf{x}_2)$ and $y_K(\mathbf{x}_1) = y_K(\mathbf{x}_2)$.

`SGDK` if $y_G(\mathbf{x}_1) = y_G(\mathbf{x}_2)$ and $y_K(\mathbf{x}_1) \neq y_K(\mathbf{x}_2)$.

`DGSK` if $y_G(\mathbf{x}_1) \neq y_G(\mathbf{x}_2)$ and $y_K(\mathbf{x}_1) = y_K(\mathbf{x}_2)$.

`DGDK` if $y_G(\mathbf{x}_1) \neq y_G(\mathbf{x}_2)$ and $y_K(\mathbf{x}_1) \neq y_K(\mathbf{x}_2)$.

Then, the following metrics can be computed:

**Rand score** $\frac{\texttt{SGSK}+\texttt{DGDK}}{\texttt{SGSK}+\texttt{SGDK}+\texttt{DGSK}+\texttt{DGDK}}$

**Adjusted rand score** Modification of the rand score to take into account that some agreements may happen by chance.

**Jaccard coefficient** For each class $c$, the Jaccard coefficient is given by:

$$\frac{\texttt{SG}_c\texttt{SK}_c}{\texttt{SG}_c\texttt{SK}_c + \texttt{SG}_c\texttt{DK}_c + \texttt{DG}_c\texttt{SK}_c}$$

## 9.4 K-means

**Algorithm** Clustering algorithm that iteratively improves the centroids. Given the desired number of clusters $K$, the algorithm works as follows: K-means

1. Randomly choose $K$ initial centroids.
2. Each data point belongs to the cluster represented by the nearest centroid.
3. Update the centroids as the centroids of the newly found clusters. Go to 2.

**Distortion** Given: Distortion

- a $D$-dimensional dataset of $N$ points $\mathbf{x}_i$;
- an encoding function $\texttt{encode} : \mathbb{R}^D \to [1, K]$;
- a decoding function $\texttt{decode} : [1, K] \to \mathbb{R}^D$.

Distortion (or inertia) is defined as:

$$\texttt{distortion} = \sum_{i=1}^{N} \big(\mathbf{x}_i - \texttt{decode}(\texttt{encode}(\mathbf{x_i}))\big)^2$$

**Theorem 9.4.1.** To minimize the distortion, it is required that:

1. $\mathbf{x}_i$ is encoded with its nearest center.
2. The center of a point is the centroid of the cluster it belongs to.

Note that k-means alternates points 1 and 2.

*Proof.* The second point is derived by imposing the derivative of `distortion` to 0. □

**Elbow method** Inertia decreases monotonically and can be used to determine an ideal number of clusters. By computing the inertia for varying $K$, a plausible value is the one corresponding to the point where the slope decreases.

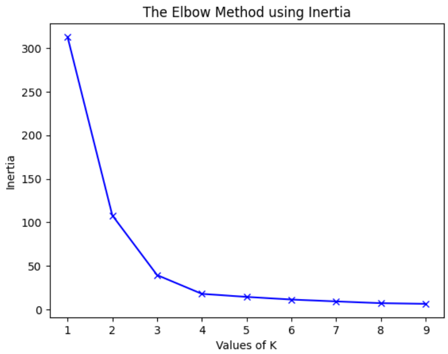



Figure 9.2: Plot of inertia. Possibly good values for $K$ are around 3

The Silhouette score can also be used by selecting the $K$ corresponding to its maximum. Note that, compared to inertia, Silhouette is computationally more expensive.

**Properties**

**Termination** There are a finite number of ways to cluster $N$ objects into $K$ clusters. By construction, at each iteration, the `distortion` is reduced. Therefore, k-means is guaranteed to terminate.

**Non-optimality** The solution found by k-means is not guaranteed to be a global best. The choice of starting points heavily influences the final result. The starting configuration is usually composed of points distant as far as possible.

**Noise** Outliers heavily influence the clustering result. Sometimes, it is useful to remove them.

**Complexity** Given a $D$-dimensional dataset of $N$ points, running k-means for $T$ iterations to find $K$ clusters has complexity $O(TKND)$.

## 9.5 Hierarchical clustering

**Dendrogram** Tree-like structure where the root is a cluster of all the data points and the leaves are clusters with a single data point. Dendrogram

**Agglomerative** Starts with a cluster per data point and iteratively merges them (leaves to root). Uses cluster separation metrics. Agglomerative

**Divisive** Starts with a cluster containing all the data points and iteratively splits them (root to leaves). Uses cluster cohesion metrics. Divisive

**Cluster separation measures** Measure the distance between two clusters $K_i$ and $K_j$.

**Single link** Minimum distance of the points in the two clusters: Single link

$$\texttt{sep}(K_i, K_j) = \min_{\mathbf{x} \in K_i, \mathbf{y} \in K_j} \texttt{dist}(\mathbf{x}, \mathbf{y})$$

Tends to create larger clusters.

**Complete link** Maximum distance of the points in the two clusters: Complete link

$$\texttt{sep}(K_i, K_j) = \max_{\mathbf{x} \in K_i, \mathbf{y} \in K_j} \texttt{dist}(\mathbf{x}, \mathbf{y})$$

Tends to create more compact clusters.

**Average link** Average distance of the points in the two clusters: Average link

$$\texttt{sep}(K_i, K_j) = \frac{1}{|K_i| \cdot |K_j|} \sum_{\mathbf{x} \in K_i, \mathbf{y} \in K_j} \texttt{dist}(\mathbf{x}, \mathbf{y})$$

**Centroid-based** Distance between the centroids of the two clusters. Centroid-based

**Ward's method** Let $K_m$ be the cluster obtained by merging $K_i$ and $K_j$. The distance between $K_i$ and $K_j$ is determined as: Ward's method

$$\texttt{sep}(K_i, K_j) = \texttt{SSE}(K_m) - \big(\texttt{SSE}(K_i) + \texttt{SSE}(K_j)\big)$$

### 9.5.1 Agglomerative clustering

#### Algorithm

Agglomerative clustering

1. Initialize a cluster for each data point.
2. Compute the distance matrix between each cluster.
3. Merge the two clusters with the lowest separation, drop their values from the distance matrix and add a row/column for the newly created cluster.
4. Go to point 2. if the number of clusters is greater than one.

After the construction of the dendrogram, a cut can be performed at a user-defined level. A cut near the root will result in few bigger clusters. A cut near the leaves will result in numerous smaller clusters. Cut

#### Properties

**Complexity** Space complexity of $O(N^2)$ to store the distance matrix.

Time complexity of $O(N^3)$ ($O(N)$ iterations with a $O(N^2)$ search for the pair to merge and $O(N)$ to recompute the distance matrix) that can be reduced to $O(N^2 \log(N))$ when using indexing.

## 9.6 Density-based clustering

Consider as clusters the high-density areas of the data space.

**Grid-based** Split the data space into a grid and count the number of points in each tile.

**Object-centered** Count, for each point, the number of neighbors within a radius.

### 9.6.1 DBSCAN

**Neighborhood** Given a radius $\varepsilon$, the neighborhood of a point $\mathbf{x}$ are the points within an $\varepsilon$-sphere centered on $\mathbf{x}$. Neighborhood

**Core point** Given a minimum number of neighbors $m$, a point $\mathbf{x}$ is a core point if it has at least $m$ neighbors. Core point

**Border point** A point $\mathbf{x}$ is a border point if it is not a core point. Border point

**Directly density reachable** A point $\mathbf{p}$ is directly density reachable from $\mathbf{q}$ iff: Directly density reachable

- $\mathbf{q}$ is a core point.
- $\mathbf{q}$ is a neighbor of $\mathbf{p}$.

**Density reachable** A point $\mathbf{p}$ is density reachable from $\mathbf{q}$ iff: Density reachable

- $\mathbf{q}$ is a core point.
- There exists a sequence of points $\mathbf{s}_1, \dots, \mathbf{s}_z$ such that:
  - $\mathbf{s}_1$ is directly density reachable from $\mathbf{q}$.
  - $\mathbf{s}_{i+1}$ is directly density reachable from $\mathbf{s}_i$.
  - $\mathbf{p}$ is directly density reachable from $\mathbf{s}_z$.

**Density connected** A point $\mathbf{p}$ is density connected to $\mathbf{q}$ iff there exists a point $\mathbf{s}$ such that both $\mathbf{p}$ and $\mathbf{q}$ are density reachable from $\mathbf{s}$. Density connected

**Algorithm** Determine clusters as maximal sets of density connected points. Border points not density connected to any core point are labeled as noise. DBSCAN

In other words, what happens is the following:

- Neighboring core points are part of the same cluster.
- Border points are part of the cluster of their nearest core point neighbor.
- Border points without a core point neighbor are noise.

**Properties**

**Robustness** Able to find clusters of any shape and detect noise.

**Hyperparameters** Sensible to the choice of the radius $\varepsilon$ and minimum neighbors $m$.

**K-distance method**

1. Determine for each point its $k$-distance as the distance to its $k$-nearest neighbors.
2. Sort the points by decreasing $k$-distance and plot them.
3. Use as possible $\varepsilon$ the values around the area where the slope decreases (similarly to the elbow method).

**Complexity** Complexity of $O(N^2)$, reduced to $O(N \log N)$ if using spatial indexing.

### 9.6.2 DENCLUE

**Kernel density estimation** Statistical method to estimate the distribution of a dataset through a function. Kernel density estimation

**Kernel function** Symmetric and monotonically decreasing function to describe the influence of a data point on its neighbors. Kernel function

A typical kernel function is the Gaussian.

**Overall density function** The overall density of the dataset is obtained as the sum of the kernel function evaluated at each data point.

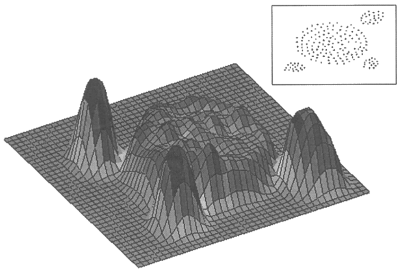


Figure 9.3: Example of density function from a set of points (top right) using a Gaussian kernel

**Algorithm** Given a threshold $\xi$, DENCLUE works as follows: DENCLUE

1. Derive a density function of the dataset.
2. Identify local maximums and consider them as density attractors.
3. Associate to each data point the density attractor in the direction of maximum increase.
4. Points associated with the same density attractor are part of the same cluster.
5. Remove clusters with a density attractor lower than $\xi$.
6. Merge clusters connected through a path of points whose density is greater or equal to $\xi$ (e.g. in Figure 9.3 the center area will result in many small clusters that can be merged with an appropriate $\xi$).

**Properties**

**Robustness** Able to recognize clusters of different shapes and handle noise.

**High dimension weakness** Does not perform well with high-dimensional data with different densities.

**Complexity** Computational complexity of $O(N^2)$.

## 9.7 Model-based clustering

Assuming that the attributes are independent random variables, model-based clustering finds a set of distributions (one per cluster) that describe the data.

### Gaussian mixture (expectation maximization)

**Algorithm** Gaussian mixture

1. Select an initial set of parameters for the distributions.
2. Expectation step: for each data point, compute its probability to belong to each distribution.
3. Maximization step: tweak the parameters to maximize the likelihood (i.e. move the Gaussian towards the center of the cluster).
4. Go to point 2. until convergence.

# 10 Association rules

## 10.1 Frequent itemset

**Itemset** Collection of one or more items (e.g. $\{\text{milk}, \text{bread}, \text{diapers}\}$). Itemset

**K-itemset** Itemset with $k$ items. K-itemset

**Support count** Number of occurrences of an itemset in a dataset. Support count

**Example.**
Given the following transactions:

| | |
|---|---|
| 1 | bread, milk |
| 2 | beer, bread, diaper, eggs |
| 3 | beer, coke, diaper, milk |
| **4** | **beer, bread, diaper, milk** |
| **5** | **bread, coke, diaper, milk** |

The support count of the itemset containing bread, diapers and milk is:

$$\sigma(\{\text{bread}, \text{diapers}, \text{milk}\}) = 2$$

**Association rule** Given two itemsets $A$ and $C$, an association rule has form: Association rule

$$A \to C$$

It means that there are transactions in the dataset where $A$ and $C$ co-occur. Note that it is not strictly a logical implication.

### Metrics

**Support** Given $N$ transactions, the support of an itemset $A$ is: Support

$$\texttt{sup}(A) = \frac{\sigma(A)}{N}$$

The support of an association rule $A \to C$ is:

$$\texttt{sup}(A \to C) = \texttt{sup}(A \cup C) = \frac{\sigma(A \cup C)}{N}$$

Low support implies random associations.

**Frequent itemset** Itemset whose support is at least a given threshold. Frequent itemset

**Confidence** Given an association rule $A \to C$, its confidence is given by: Confidence

$$\texttt{conf}(A \to C) = \frac{\sigma(A \cup C)}{\sigma(A)} \in [0, 1]$$

Low confidence implies low reliability.

**Theorem 10.1.1.** The confidence of $A \to C$ can be computed given the supports of $A \to C$ and $A$:

$$\texttt{conf}(A \to C) = \frac{\texttt{sup}(A \to C)}{\texttt{sup}(A)}$$

**Association rule mining** Given $N$ transactions and two thresholds `min_sup` and `min_conf`, association rule mining finds all the rules $A \rightarrow C$ such that:

Association rule mining

$$\texttt{sup}(A \rightarrow C) \geq \texttt{min\_sup}$$
$$\texttt{conf}(A \rightarrow C) \geq \texttt{min\_conf}$$

This can be done in two steps:

1. Determine the itemsets with support $\geq$ `min_sup` (frequent itemsets).
2. Determine the association rules with confidence $\geq$ `min_conf`.

Frequent itemset generation
Rule generation

## 10.2 Frequent itemset generation

### 10.2.1 Brute force

Given $D$ items, there are $2^D$ possible itemsets. To compute the support of a single itemset, the complexity is $O(NW)$ where $N$ is the number of transactions and $W$ is the width of the largest transaction. Listing all the itemsets and computing their support have an exponential complexity of $O(NW2^D)$.

### 10.2.2 Apriori principle

**Theorem 10.2.1.** If an itemset is frequent, then all of its subsets are frequent.

Apriori principle

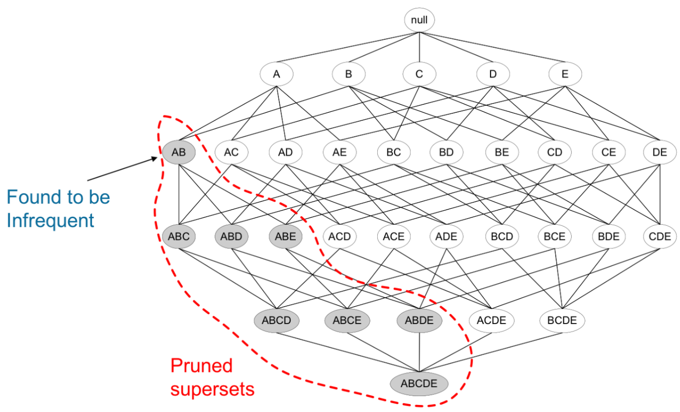
*Proof.* By the definition of support, it holds that:

$$\forall X, Y : (X \subseteq Y) \Rightarrow (\texttt{sup}(X) \geq \texttt{sup}(Y))$$

In other words, the support metric is anti-monotone. □

**Corollary 10.2.1.1.** If an itemset is infrequent, then all of its supersets are infrequent.

**Example.**



**Algorithm 3** Apriori principle

```
def candidatesGeneration(freq_itemsets_k):
    candidate_itemsets_{k+1} = selfJoin(freq_itemsets_k)
    for itemset in candidate_itemsets_{k+1}:
        for sub in subsetsOfSize(k, itemset):
            if sub not in freq_itemsets_k:
                candidate_itemsets_{k+1}.remove(itemset)
    return candidate_itemsets_{k+1}

def aprioriItemsetGeneration(transactions, min_sup):
    freq_itemsets_1 = itemsetsOfSize(1, transactions)
    k = 1
    while freq_itemsets_1 is not null:
        candidate_itemsets_{k+1} = candidatesGeneration(freq_itemsets_k)
        freq_itemsets_{k+1} = {c ∈ candidate_itemsets_{k+1} | sup(c) ≥ min_sup}
        k += 1
    return freq_itemsets_k
```

**Complexity** The complexity of the apriori principle depends on:

- The choice of the support threshold.
- The number of unique items.
- The number and the width of the transactions.

## 10.3 Rule generation

### 10.3.1 Brute force

Given a frequent $k$-itemset $L$, there are $2^k - 2$ possible association rules ($-2$ as $L \rightarrow \varnothing$ and $\varnothing \rightarrow L$ can be ignored). For each possible rule, it is necessary to compute the confidence. The overall complexity is exponential.

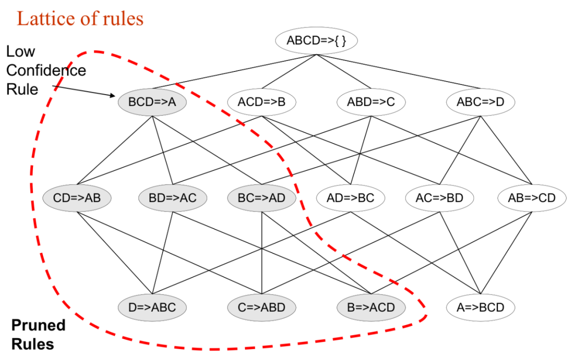
### 10.3.2 Apriori principle

Apriori principle

**Theorem 10.3.1.** Without loss of generality, consider an itemset $\{A, B, C, D\}$. It holds that:

$$\texttt{conf}(ABC \rightarrow D) \geq \texttt{conf}(AB \rightarrow CD) \geq \texttt{conf}(A \rightarrow BCD)$$

**Example.**



## 10.4 Interestingness measures

**Contingency table** Given an association rule $A \rightarrow C$, its contingency table is defined as: Contingency table

| | $C$ | $\overline{C}$ | |
|---|---|---|---|
| $A$ | $\mathcal{P}(A \wedge C)$ | $\mathcal{P}(A \wedge \overline{C})$ | $\mathcal{P}(A)$ |
| $\overline{A}$ | $\mathcal{P}(\overline{A} \wedge C)$ | $\mathcal{P}(\overline{A} \wedge \overline{C})$ | $\mathcal{P}(\overline{A})$ |
| | $\mathcal{P}(C)$ | $\mathcal{P}(\overline{C})$ | 100 |

**Remark.** Confidence can be misleading.

**Example.**
Given the following contingency table:

| | coffee | $\overline{\text{coffee}}$ | |
|---|---|---|---|
| tea | 15 | 5 | 20 |
| $\overline{\text{tea}}$ | 75 | 5 | 80 |
| | 90 | 10 | 100 |

We have that:

$$\texttt{conf}(\text{tea} \rightarrow \text{coffee}) = \frac{\texttt{sup}(\text{tea}, \text{coffee})}{\texttt{sup}(\text{tea})} = \frac{15}{20} = 0.75$$

But, we also have that:

$$\mathcal{P}(\text{coffee}) = 0.9 \qquad \mathcal{P}\left(\text{coffee} \mid \overline{\text{tea}}\right) = \frac{75}{80} = 0.9375$$

So, despite the high confidence of (tea $\rightarrow$ coffee), the probability of coffee increases in absence of tea.

### 10.4.1 Statistical-based measures

Measures that take into account the statistical independence of the items.

**Lift** Lift

$$\texttt{lift}(A \rightarrow C) = \frac{\texttt{conf}(A \rightarrow C)}{\texttt{sup}(C)} = \frac{\mathcal{P}(A \wedge C)}{\mathcal{P}(A)\mathcal{P}(C)}$$

If $\texttt{lift}(A \rightarrow C) = 1$, then $A$ and $C$ are independent.

**Leverage** Leverage

$$\texttt{leve}(A \rightarrow C) = \texttt{sup}(A \cup C) - \texttt{sup}(A)\texttt{sup}(C) = \mathcal{P}(A \wedge C) - \mathcal{P}(A)\mathcal{P}(C)$$

If $\texttt{leve}(A \rightarrow C) = 0$, then $A$ and $C$ are independent.

**Conviction** Conviction

$$\texttt{conv}(A \rightarrow C) = \frac{1 - \texttt{sup}(C)}{1 - \texttt{conf}(A \rightarrow C)} = \frac{\mathcal{P}(A)(1 - \mathcal{P}(C))}{\mathcal{P}(A) - \mathcal{P}(A \wedge C)}$$

| **Metric** | **Interpretation** |
|---|---|
| High support | The rule applies to many transactions. |
| High confidence | The chance that the rule is true for some transactions is high. |
| High lift | Low chance that the rule is just a coincidence. |
| High conviction | The rule is violated less often compared to the case when the antecedent and consequent are independent. |

Table 10.1: Intuitive interpretation of the measures

## 10.5 Multi-dimensional association rules

**Mono-dimensional events** Represented as transactions. Each event contains items that appear together.

Mono-dimensional events

**Multi-dimensional events** Represented as tuples. Each event contains the values of its attributes.

Multi-dimensional events

**Mono/Multi-dimensional equivalence** Mono-dimensional events can be converted into multi-dimensional events and vice versa.

Equivalence

To transform quantitative attributes, it is usually useful to discretize them.

**Example** (Multi to mono)**.**

| Id | co2 | tin_oxide |
|---|---|---|
| 1 | high | medium |
| 2 | medium | low |

$\rightarrow$

| Id | Transaction |
|---|---|
| 1 | {co2/high, tin_oxide/medium} |
| 2 | {co2/medium, tin_oxide/low} |

**Example** (Mono to multi)**.**

| Id | a | b | c | d |
|---|---|---|---|---|
| 1 | yes | yes | no | no |
| 2 | yes | no | yes | no |

$\leftarrow$

| Id | Transaction |
|---|---|
| 1 | {a, b} |
| 2 | {a, c} |

## 10.6 Multi-level association rules

Organize items into a hierarchy.

**Specialized to general** Generally, the support of the rule increases.

Specialized to general

**Example.** From (apple → milk) to (fruit → dairy)

**General to specialized** Generally, the support of the rule decreases.

General to specialized

**Example.** From (fruit → dairy) to (apple → milk)

**Redundant level** A more specialized rule in the hierarchy is redundant if its confidence is similar to the one of a more general rule.

Redundant level

**Multi-level association rule mining** Run association rule mining on different levels of abstraction (general to specialized). At each level, the support threshold is decreased.

Multi-level association rule mining

`<end of course>`