Machine Learning and Data Mining

Last update: 25 November 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

Decision 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 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 Analysical Processing (OLAP)** Grouped analysis of multidimensional data. Involves large amount of data. Online Analysical 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

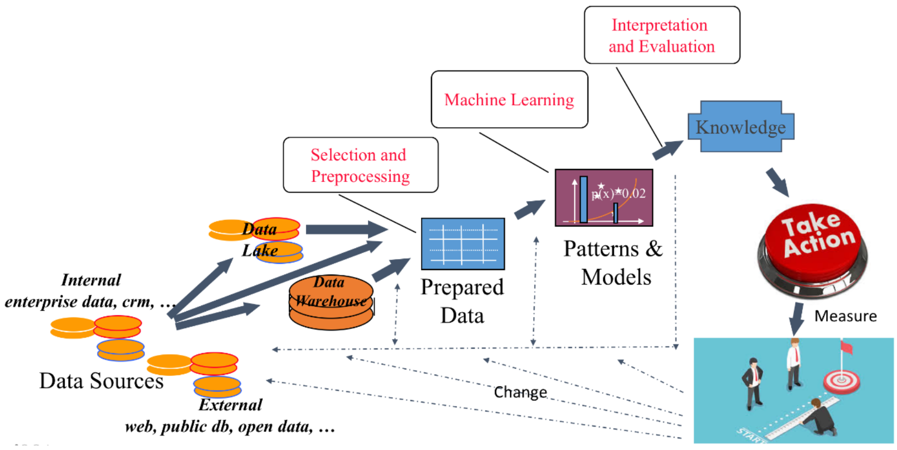


Figure 1.1: Data mining process

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 an 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.

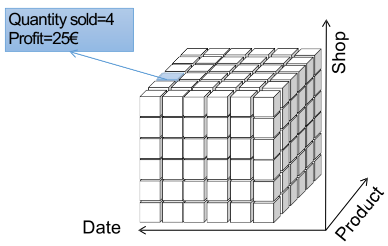
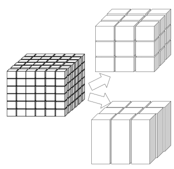


Figure 2.1: OLAP data cube

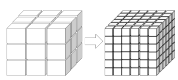
2.1.1 Operators

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



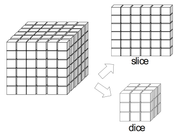
Roll-up

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



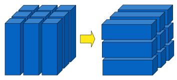
Drill-down

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).



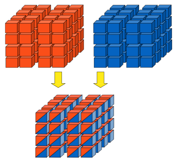
Slide-and-dice

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



Pivot

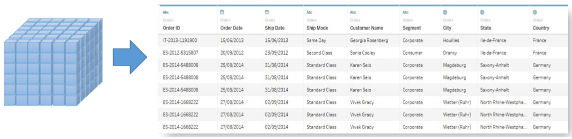
Drill-across Links concepts from different data sources (i.e. `JOIN` in SQL).



Drill-across

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

Drill-through



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)

Inconsistency (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 Merging 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 an 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 redundances (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.

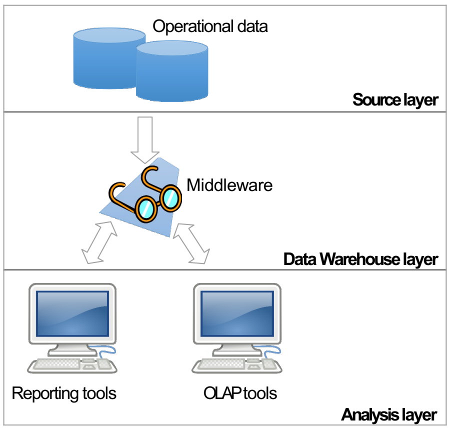
Administrability Easily manageable.

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

2.3.1 Single-layer architecture

Single-layer architecture

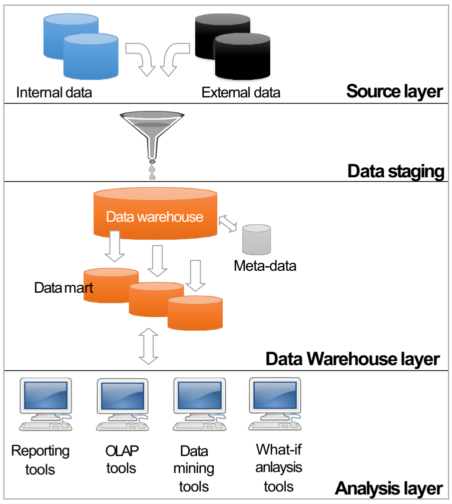
- Minimizes the amount of data stored (i.e. no redundances).
- The source layer is the only physical layer (i.e. no separation).
- A middleware provides the DWH features.



2.3.2 Two-layer architecture

Two-layer architecture

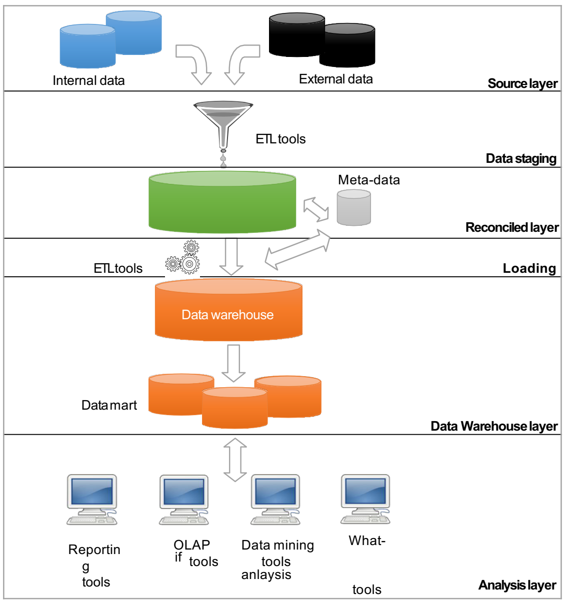
- Source data (source layer) are physically separated from the DWH (data warehouse layer).
- A staging layer applies ETL procedures before populating the DWH.
- The DWH is a centralized repository from which data marts can be created. Metadata repositories store information on sources, staging and data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

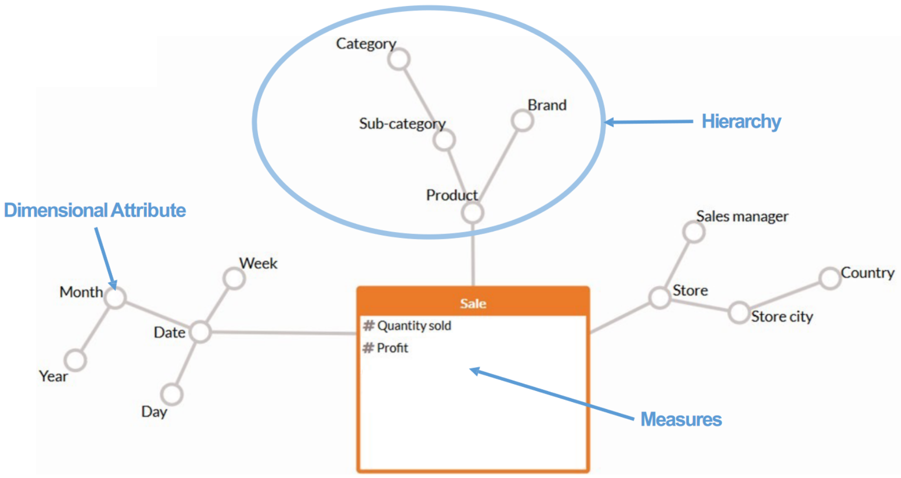


Figure 2.2: Example of DFM

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

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

Level measures

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

Unit measures

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

Aggregation operators can be classified as:

Distributive operators

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

Algebraic operators

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

Holistic operators

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

Additive measure

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

	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

2.4.2 Logical design

Logical design

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

Star schema

Star schema A fact table that contains all the measures and linked to dimensional tables.

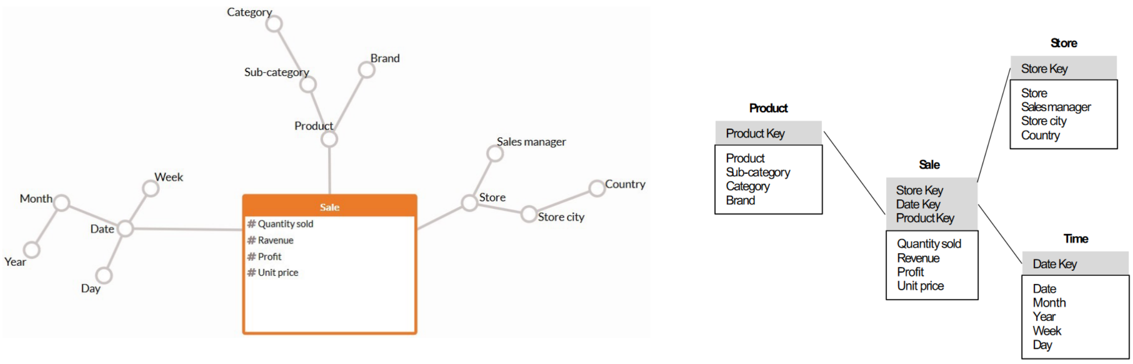


Figure 2.4: Example of star schema

Snowflake schema

Snowflake schema A star schema variant with partially normalized dimension tables.

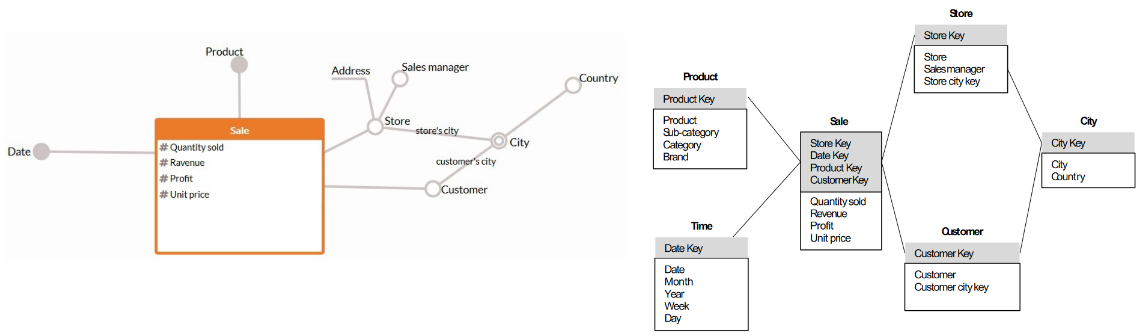


Figure 2.5: Example of snowflake schema

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 require low latency. More expensive HW/SW.

Hot storage

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

Cold storage

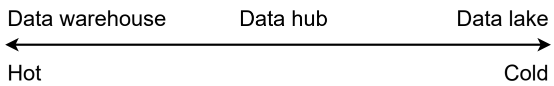


Figure 3.1: Data storage technologies

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 **Performance**	High storage cost Fast queries	Low storage cost 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 stores 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.

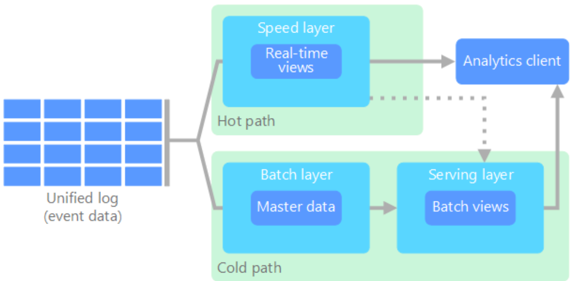


Figure 3.2: Lambda lake architecture

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).

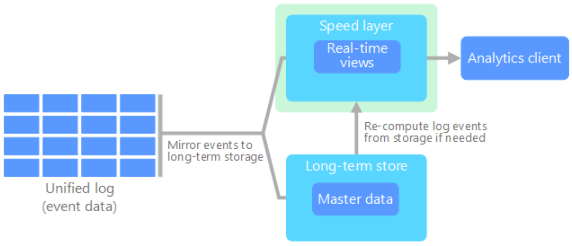


Figure 3.3: Kappa lake architecture

3.4.3 Delta lake

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

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

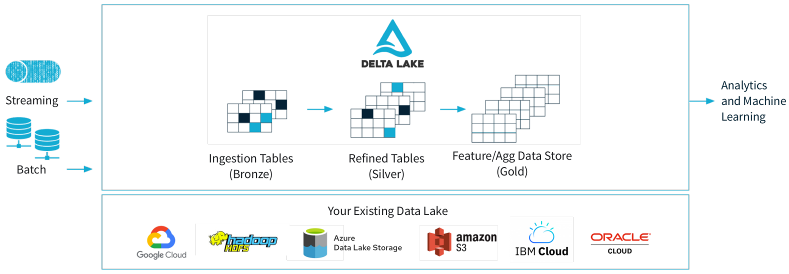


Figure 3.4: Delta lake architecture

3.5 Metadata

Metadata are 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

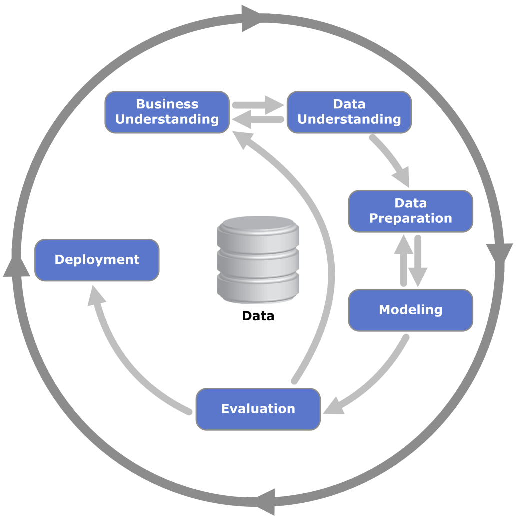


Figure 4.1: CRISP-DM workflow

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 Modelling

- Select modelling technique. Modelling
- 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 groupping 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, instead of `null`, to mark missing data).

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 weight them by their probability.

Duplicated data Data that may be merged.

Duplicated data

6 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

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

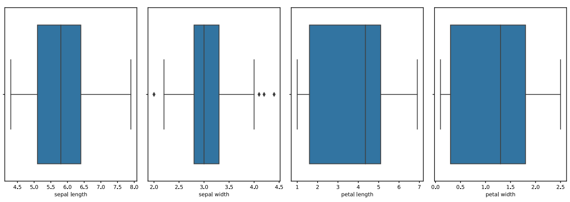
Vapnik-Chervonenkis dimension A dataset with N elements defines 2^N learning problems. A model $\mathcal{M}$ has Vapnik-Chervonenkis (VC) dimension N if it is able to solve all the possible learning problems with N elements.

Vapnik-Chervonenkis dimension

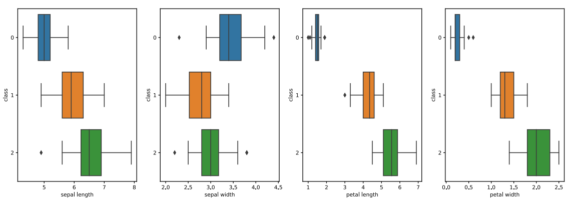
Example. A straight line has VC dimension 3.

Data exploration

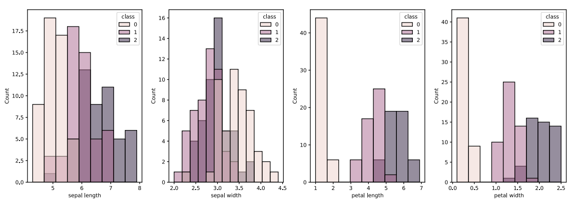
Data exploration



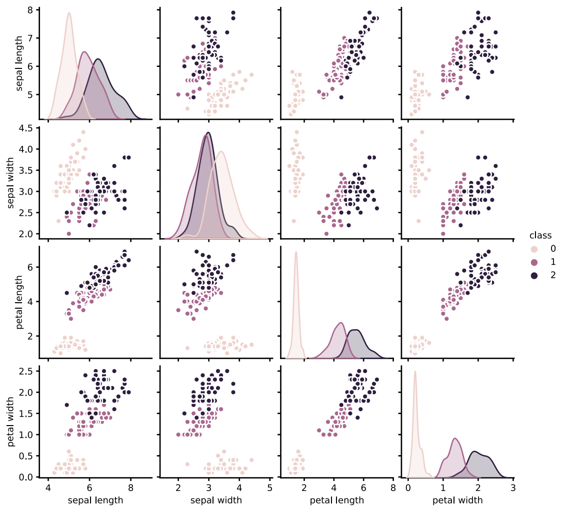
(a) Iris dataset general boxplot



(b) Iris dataset class boxplot



(c) Iris dataset histograms



(d) Iris dataset pairplots

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

6.1 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:

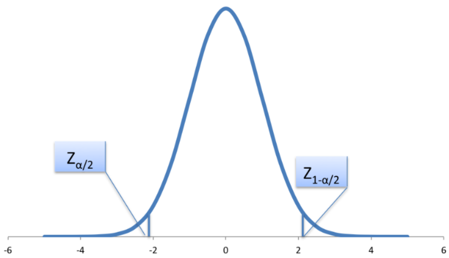
- Noisy data.
- Lack of representative instances.

6.1.1 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 ($x \pm$ 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 $f =$ (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 α (i.e. the probability of a wrong estimate), we want that:

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

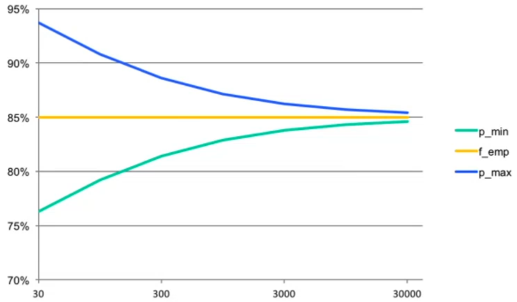
In other words, we want the middle term to have a high probability to be between the $\frac{\alpha}{2}$ and $(1-\frac{\alpha}{2})$ quantiles of the gaussian.



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 α. For a pessimistic estimate, $\pm$ becomes a $+$. Vice versa, for a optimistic estimate, $\pm$ becomes a $-$.
As N is at the denominator, this means that for large values of N, the uncertainty becomes smaller.



6.1.2 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 if 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

At 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 final model.

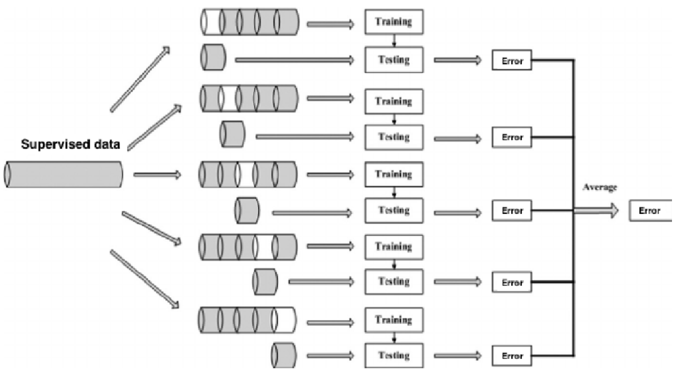


Figure 6.2: Cross validation example

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

6.1.3 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 positive 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

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

6.1.4 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.

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

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

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

κ-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.

Cost sensitive learning Assign a cost to the errors. 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).

6.1.5 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.

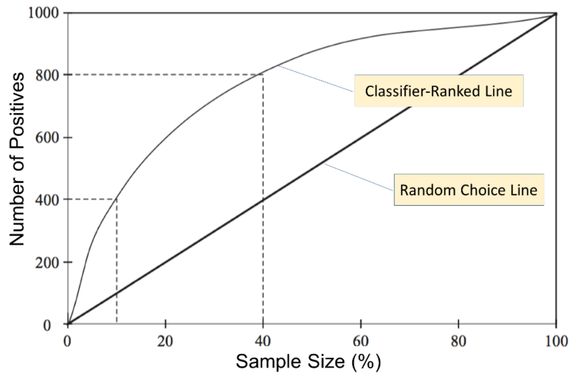


Figure 6.3: Example of lift chart

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 represent the false positive rate while the y-axis represent 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.

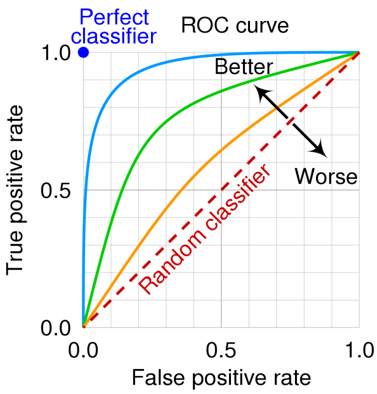


Figure 6.4: Example of ROC curves

6.2 Decision trees

6.2.1 Information theory

Shannon theorem Let $\boldsymbol{X} = \{\mathbf{v}_1, \dots, \mathbf{v}_V\}$ be a data source where each of the possible value 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

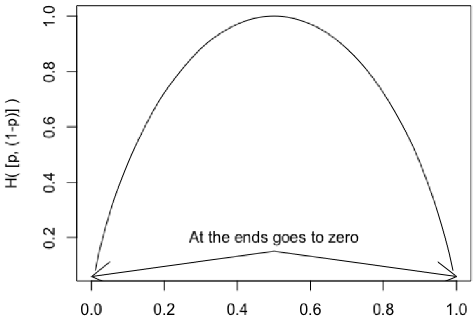
$$H(\boldsymbol{X}) = -\sum_j p_j \log_2(p_j)$$

$H(\boldsymbol{X})$ 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 an 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

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

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

Information gain

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

When $H(c \,|\, d : t)$ is low, $IG(c \,|\, d : t)$ is high as splitting with threshold t result in purer groups. Vice versa, when $H(c \,|\, d : t)$ is high, $IG(c \,|\, d : t)$ 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:

$$IG(c \,|\, d) = \max_t IG(c \,|\, d : t)$$

6.2.2 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.

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.

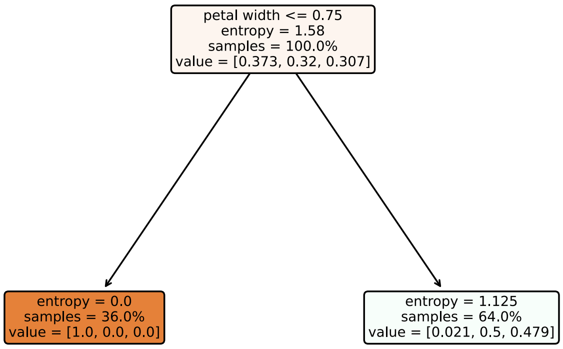


Figure 6.5: Example of decision tree

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 6.2.1.

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

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

Misclassification error has a bias toward the major class.

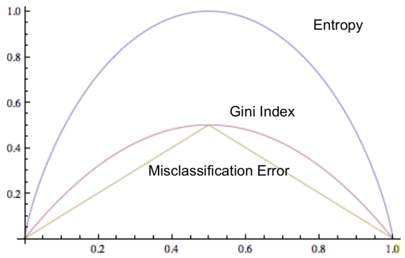


Figure 6.6: Comparison of impurity measures

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.

6.2.3 Complexity

Given a dataset $\boldsymbol{X}$ of N instances and D attributes, each level of the tree requires to evaluate all the 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.

6.2.4 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.

6.3 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.

6.3.1 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

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

6.3.2 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 uses this feature when computing the products chain during inference. Smoothing methods can be used to avoid this problem.

Laplace smoothing

Laplace smoothing Given:

- α 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.
- 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 α is 1. When $\alpha = 0$, there is no smoothing. For higher values of α, the smoothed feature gain more importance when computing the priors.

Missing values

Missing values Naive Bayes is robust to 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.

Gaussian assumption

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.

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:

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

6.4 Perceptron

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 $w_1, \dots, w_n, w_b$.

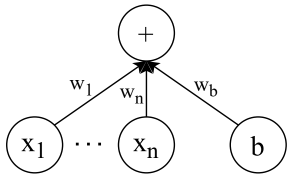


Figure 6.7: Example of perceptron

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

$$w_1x_1 + \ldots + w_nx_n + w_bb = \begin{cases} \texttt{positive} & \text{if} > 0 \\ \texttt{negative} & \text{if} < 0 \end{cases}$$

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

6.4.1 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.

6.5 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

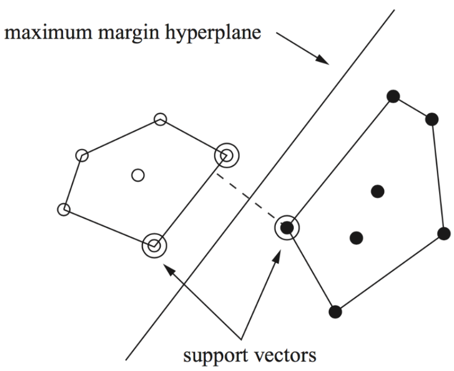


Figure 6.8: Maximum margin hyperplane of linearly separable data

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 margine. 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

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

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

6.5.1 Kernel trick

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.

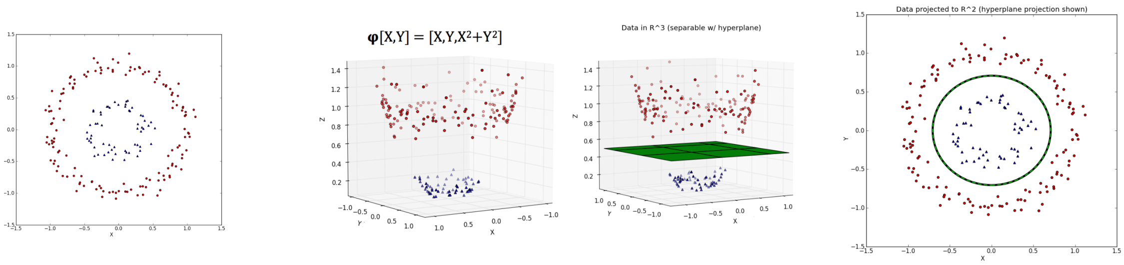


Figure 6.9: Example of mapping from $\mathbb{R}^2$ to $\mathbb{R}^3$

The kernel trick allows to avoid to explicitly map 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 γ, r and d are parameters.

Radial based function $K(x, y) = \exp(-\gamma\|x - y\|^2)$, where γ 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`

6.5.2 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.

6.5.3 Characteristics

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

6.6 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

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.

6.6.1 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 has been 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

6.7 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

6.8 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.

6.9 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).

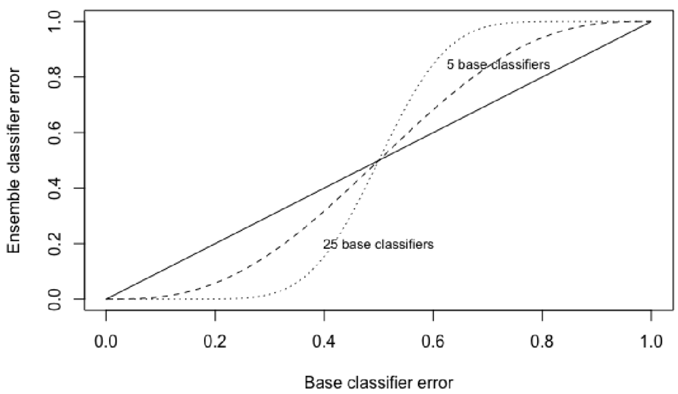


Figure 6.10: Relationship between the error of base classifiers and ensemble models

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.

6.9.1 Random forests

Random forests

Different decision trees trained on a different random sampling of the training set and different subset 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.

7 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.
- It is suited to compare different models.

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

Note that:

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

Polynomial regression Find a polynomial instead of a hyperplane.

Polynomial regression