Crash course on ML

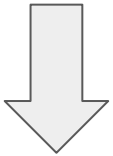
Serafina Di Gioia
Postdoctoral researcher in ML @ICTP

What is ML?

in 1960

use and development of computer systems that are able to learn and adapt without following explicit instructions

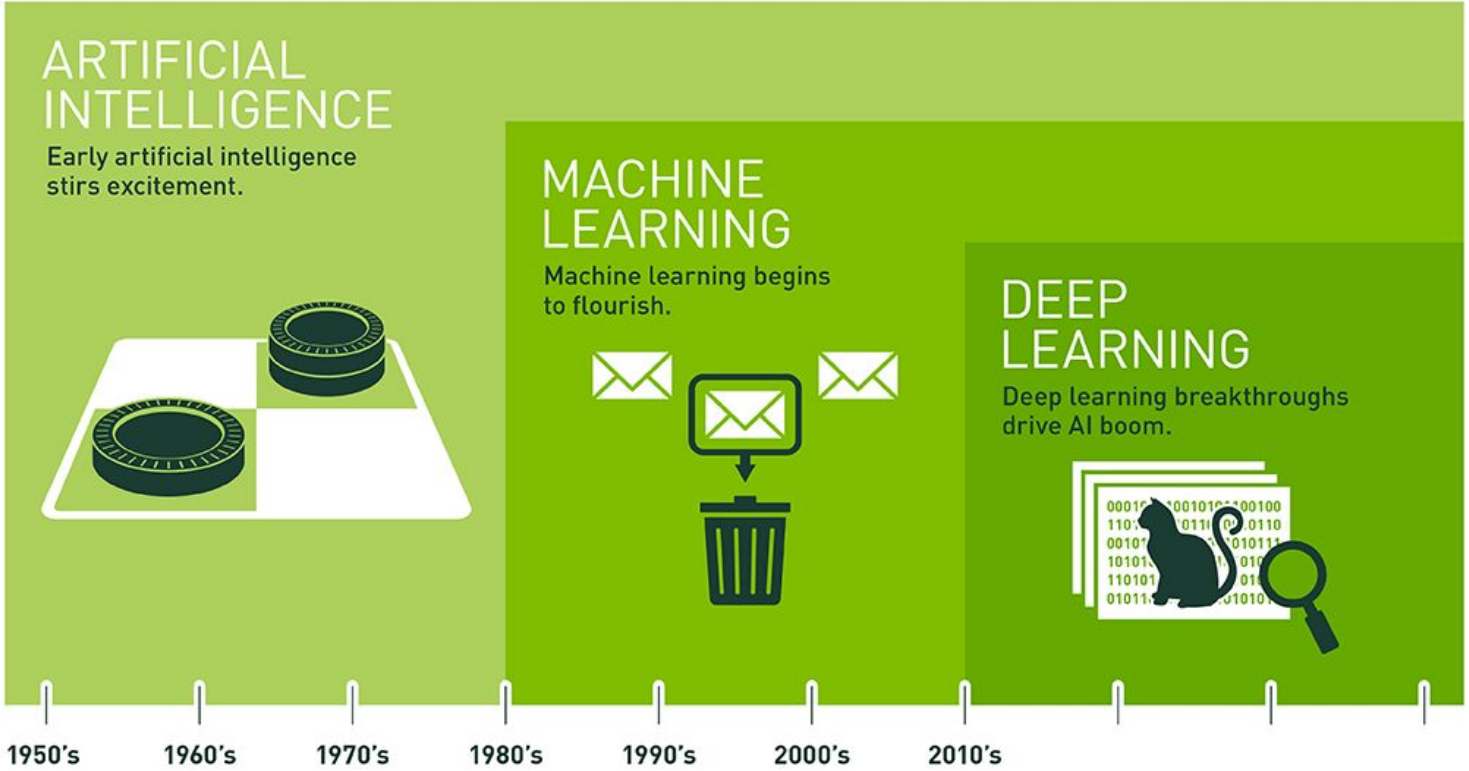
shift in perspective



in 2023

subfield of AI that uses algorithms trained on data to produce adaptable models that can perform a variety of complex tasks (e.g. classification and forecasting) without the need to code specific instructions for different tasks

From LISP to the DL revolution…

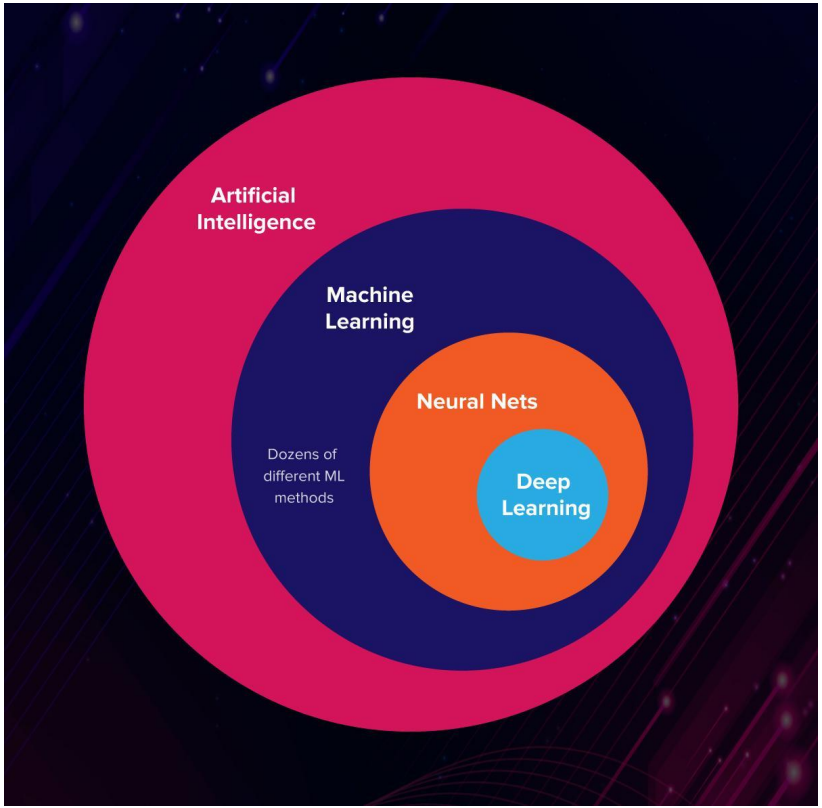


Since an early flush of optimism in the 1950s, smaller subsets of artificial intelligence – first machine learning, then deep learning, a subset of machine learning – have created ever larger disruptions.

Where is DL in the picture?

Deep learning:

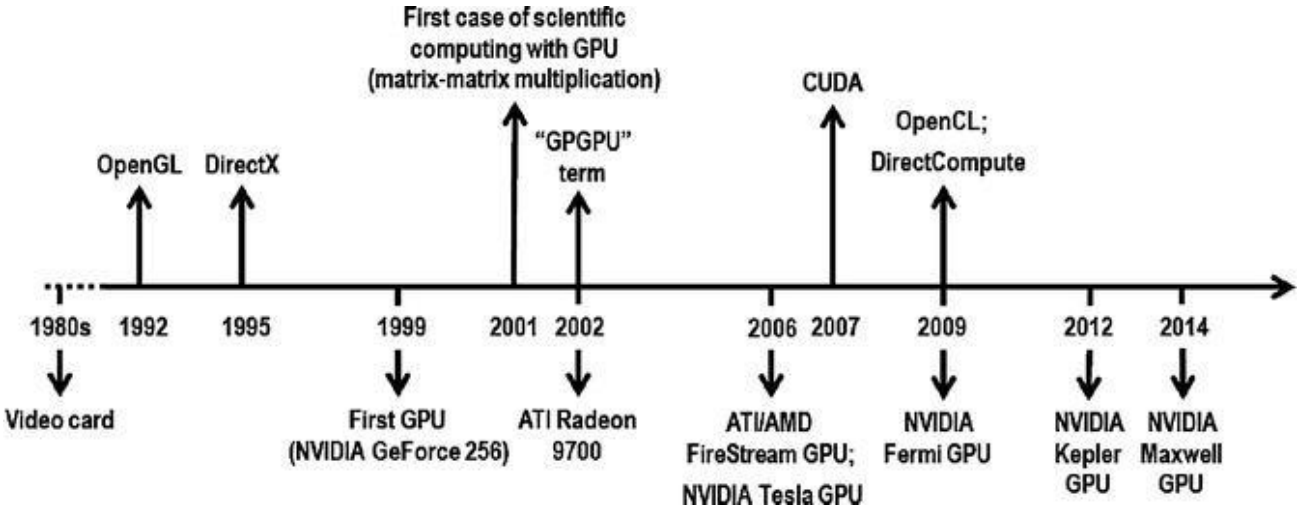
a type of machine learning based on artificial neural networks in which multiple layers of processing are used to extract progressively higher level features from data.



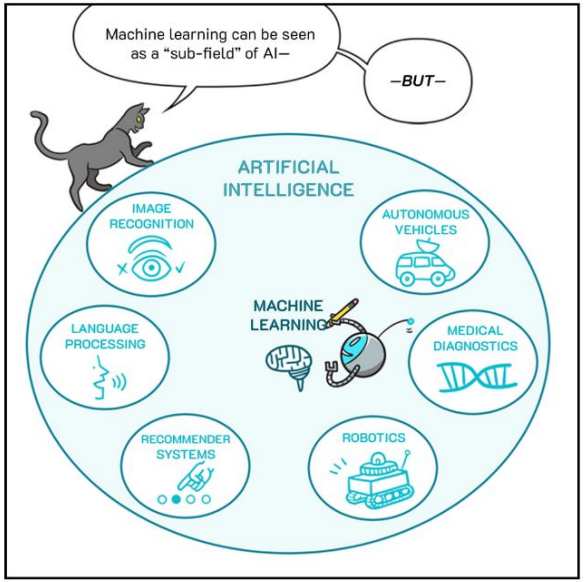
Main ingredients for DL breakthrough

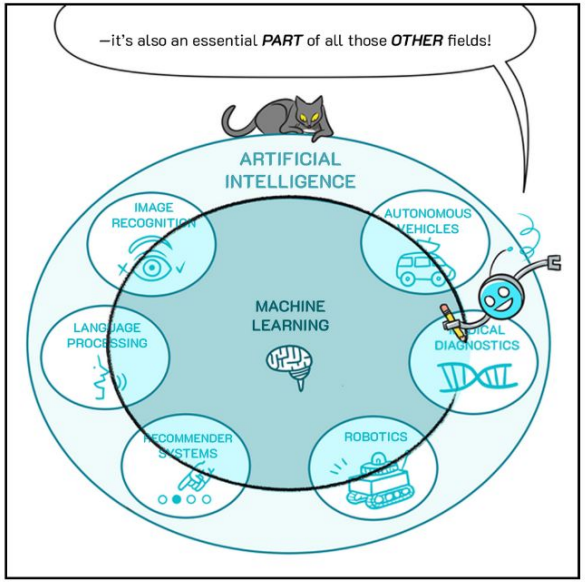
- large datasets available (e.g IMAGENET)
- GPUs development (in particular, CUDA introduction)
- increased involvement of developers from CV and scientific communities

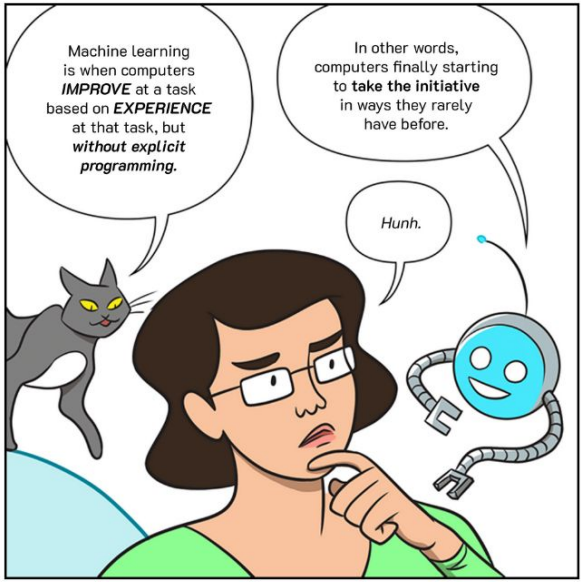
The DL era starts few years after that CUDA came to light



ML in comics

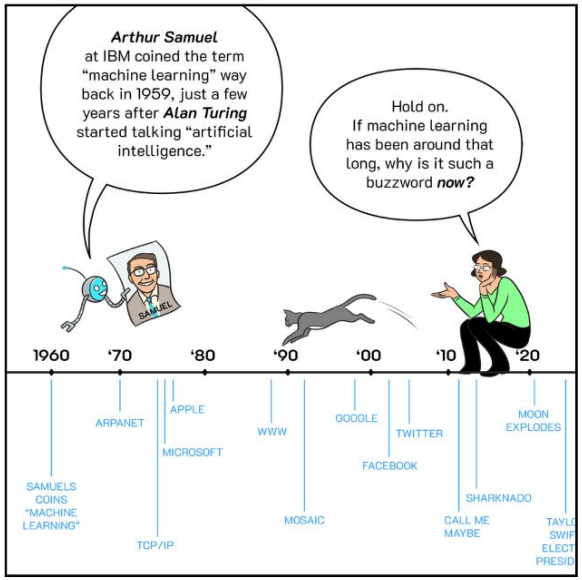


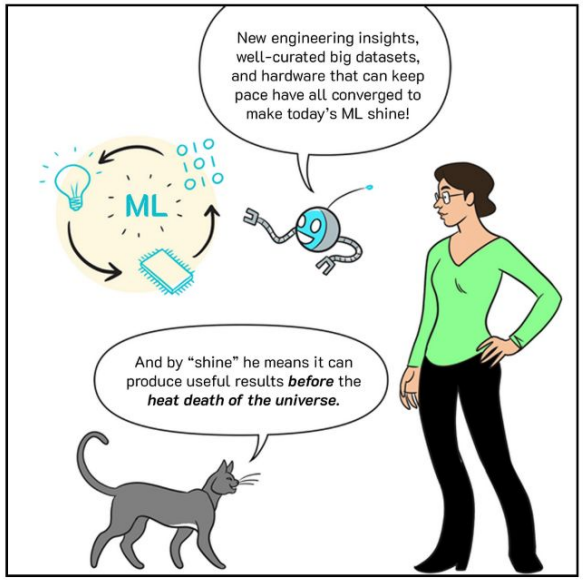


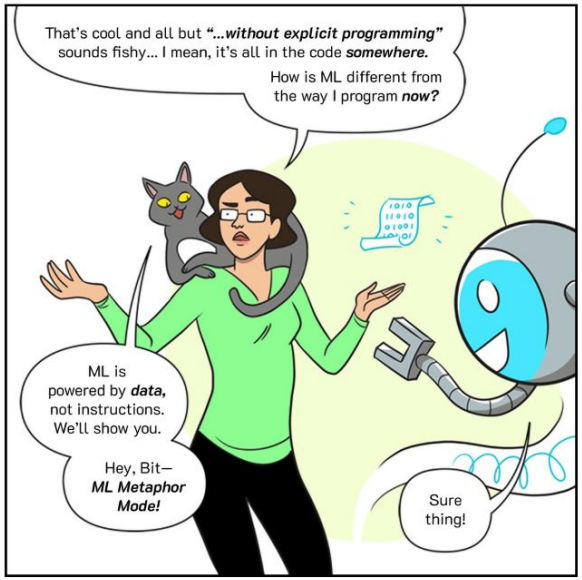


from Google

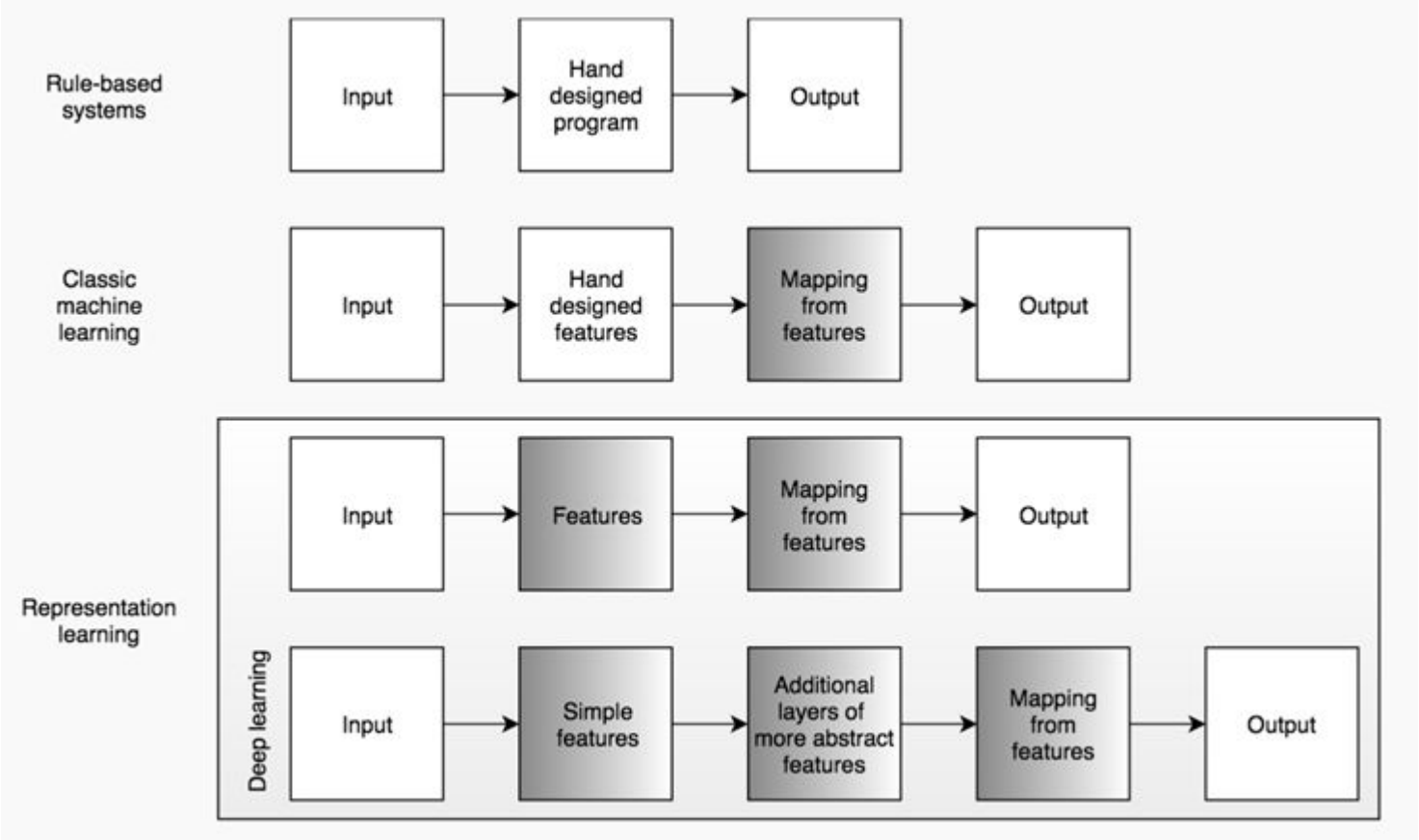
History of ML







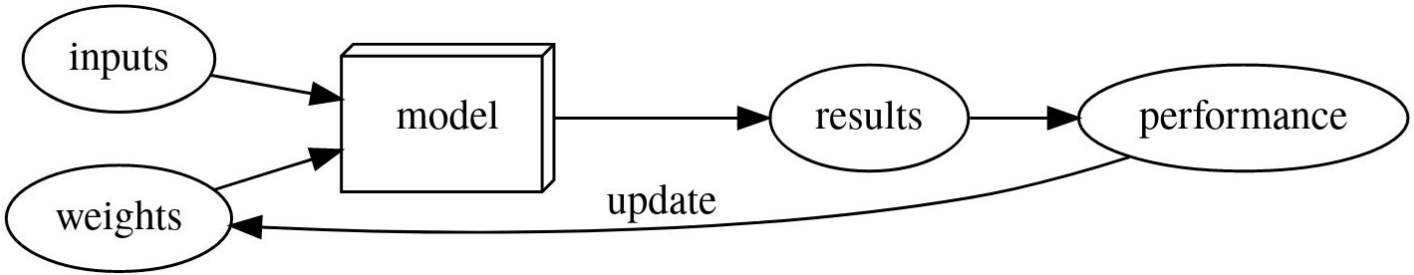
ML/DL vs tradition programming



Building blocks of ML algorithms

ML algorithms have three main components

1. **decision process**: based on some input data, which can be labeled or unlabeled, your algorithm will produce an estimate about a pattern in the data. This estimate can be used to solve a prediction or classification task
2. **error function**: it evaluates the prediction of the model. If there are known examples, an error function can make a comparison to assess the accuracy of the model.
3. **Model Optimization Process**: If the model can fit better to the data points in the training set, then weights are adjusted to reduce the discrepancy between the known example and the model estimate. The algorithm will repeat this “evaluate and optimize” process, updating weights autonomously until a threshold of accuracy has been met.



Books on ML

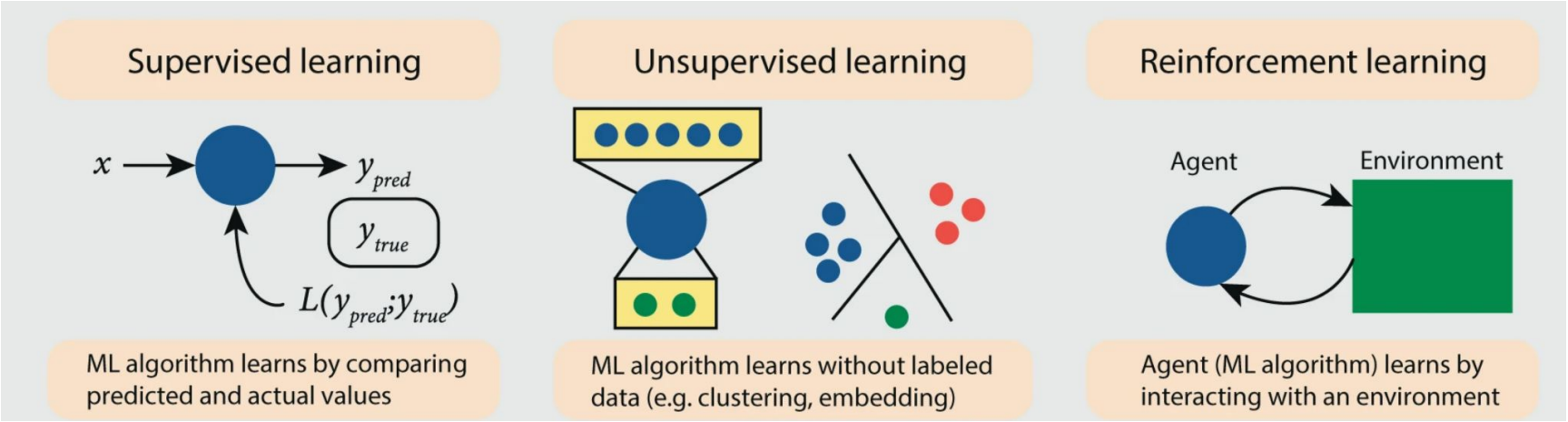
Deep Learning,

by Ian Goodfellow and Yoshua Bengio and Aaron Courville

(free online http://www.deeplearningbook.org/)

Aurelien Geron, Hands-On Machine Learning with Scikit-Learn and TensorFlow

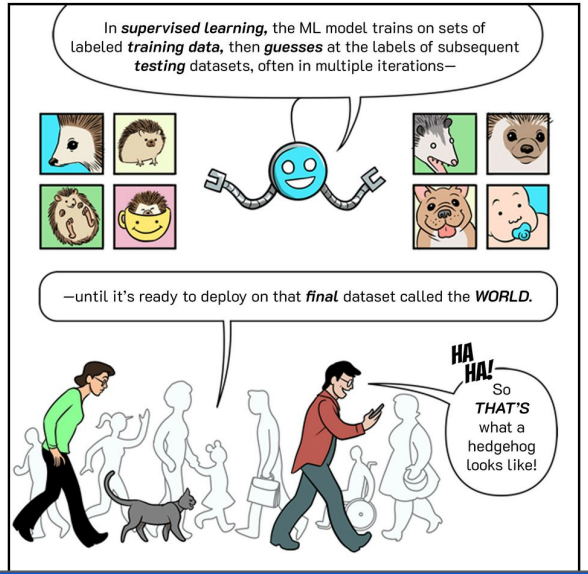
ML branches



Learning type	Model building	Examples
Supervised	Algorithms or models learn from labeled data (task-driven approach)	Classification, regression
Unsupervised	Algorithms or models learn from unlabeled data (Data-Driven Approach)	Clustering, associations, dimensionality reduction
Semi-supervised	Models are built using combined data (labeled + unlabeled)	Classification, clustering
Reinforcement	Models are based on reward or penalty (environment-driven approach)	Classification, control

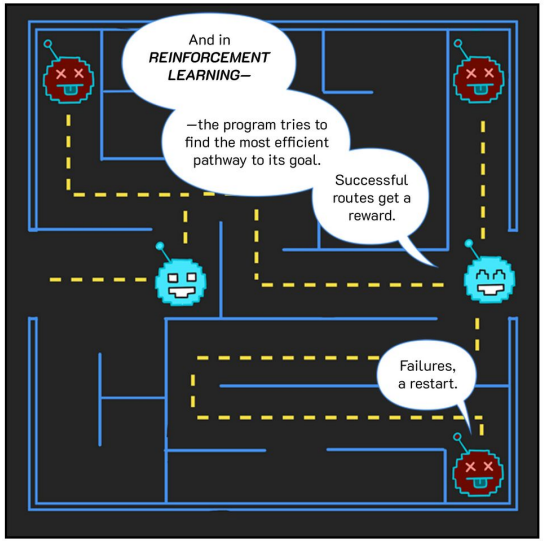
Branches of ML in the comics

Supervised learning



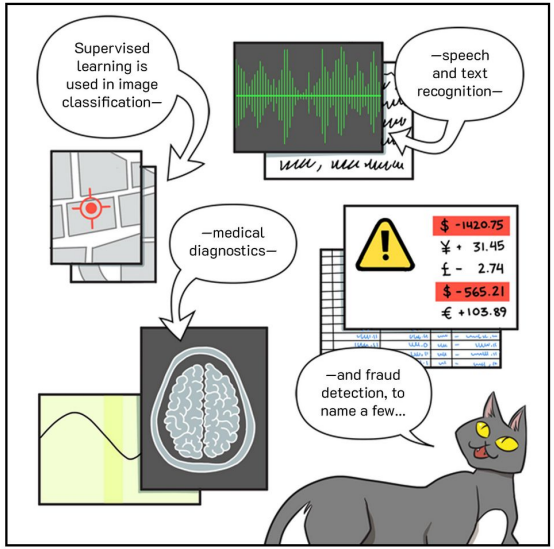
Unsupervised learning

Reinforcement Learning

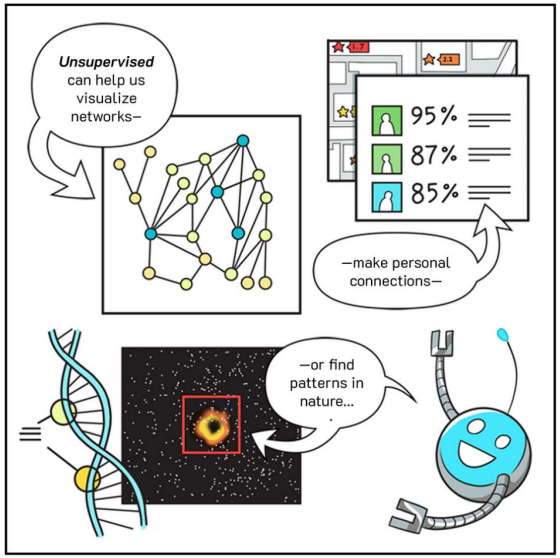


Applications of different methods

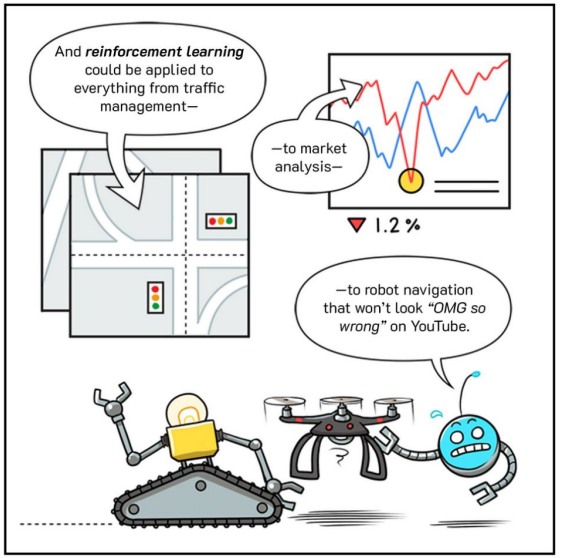
Supervised learning



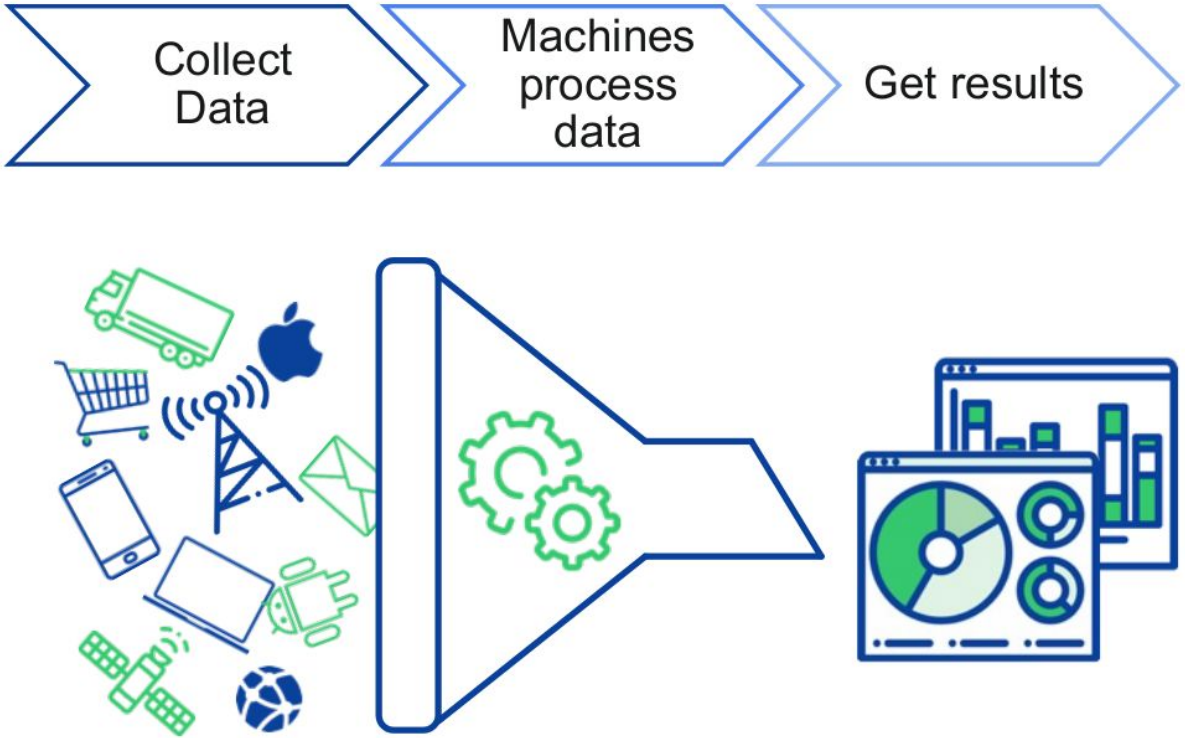
Unsupervised learning



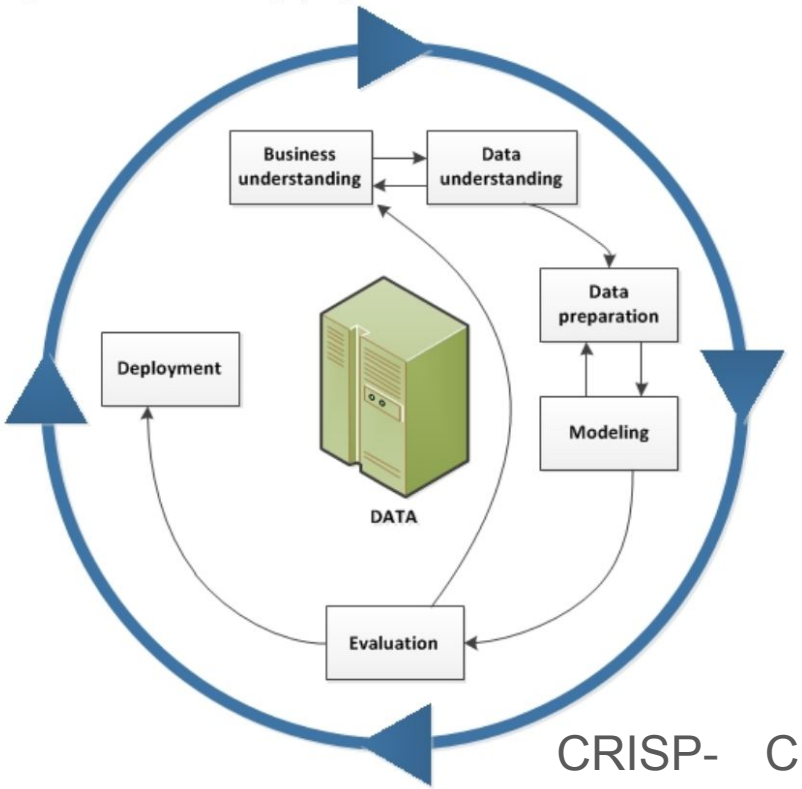
Reinforcement Learning



What people think of ML workflow…



What ML is in reality



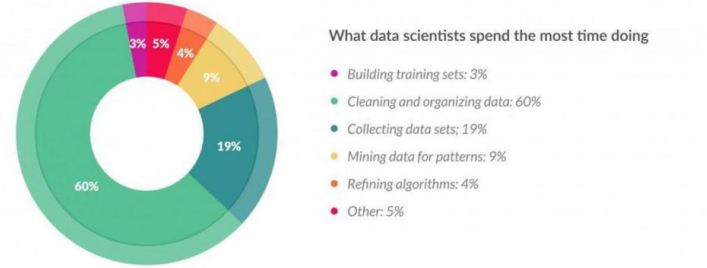
1. **Business understanding**
   - Focus on the business problem in terms of objectives and requirements
   - Translate the business problem into a data-mining problem
2. **Data understanding**
   - Data collection
   - Data exploration: variables, data quality, get first insight into
3. **Data preparation**
   - Data cleaning, selection, transformation …
   - Definition of the final data-set to be used in the analysis
4. **Modelling**
   - Select and optimize models that can better describe the prob
5. **Evaluation**
   - Evaluate the performance of the analysis process in terms of business requirements
   - If the quality of the results matches the business expectation deploy
   - Else: identify which are the business needs that are not yet satisfied and upgrade the analysis
6. **Deployment**

CRISP-DM Cross-Industry Standard Process for Data Mining

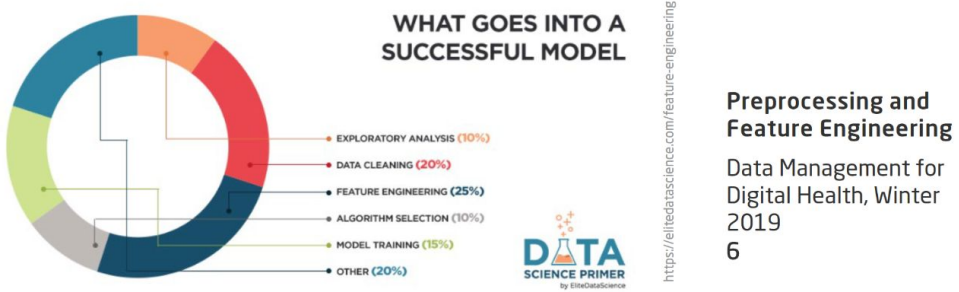
The importance of Exploratory data analysis…

results of surveys compiled by ML scientists some years ago…

but still valid!



Source: https://www.forbes.com/sites/gilpress/2016/03/23/data-preparation-most-time-consuming-least-enjoyable-data-science-task-survey-says/



Preprocessing and Feature Engineering

Data Management for Digital Health, Winter 2019

6

What are data?

Factual information (such as measurements or statistics) used as a basis for reasoning, discussion, or calculation

Information in digital form that can be transmitted or processed

Information output by a sensing device or organ that includes both useful and irrelevant or redundant information

Data types

Structured Data

Quantitative Data

Also called Numerical Data

- **Discrete Data**

e.g. Population, number of attempts, etc.

- **Continuous Data**

e.g. Height, Weight, Age etc.

Qualitative Data

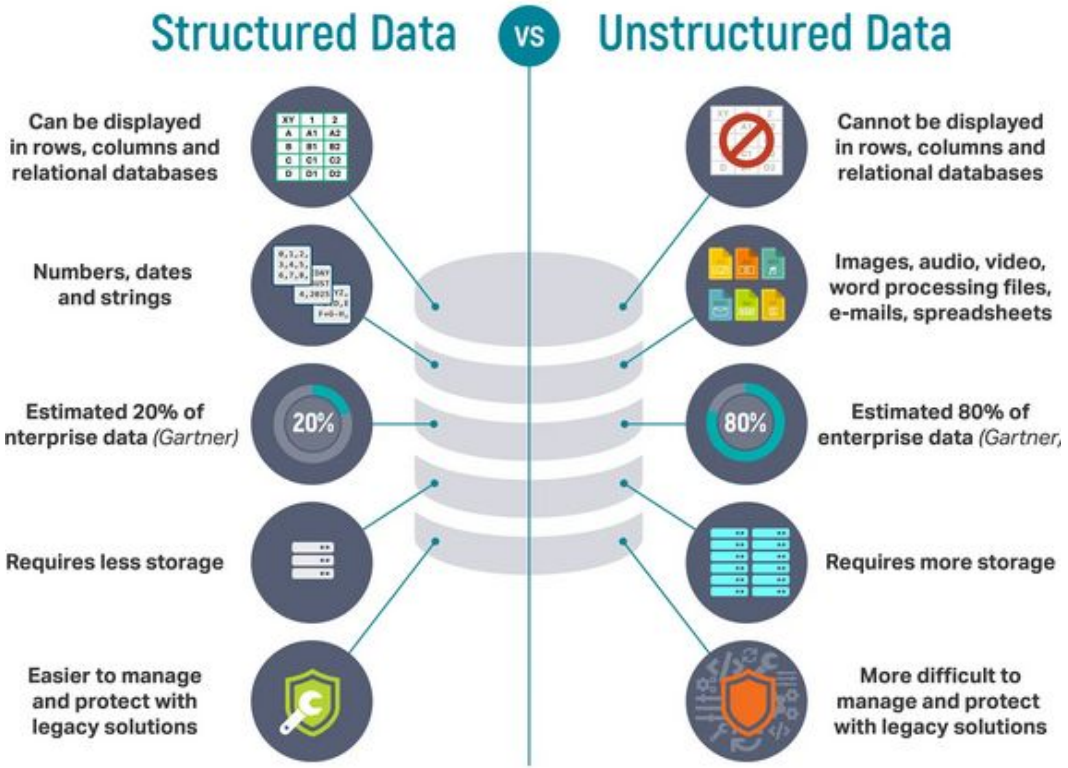
Also called Categorical Data

- **Nominal Data**

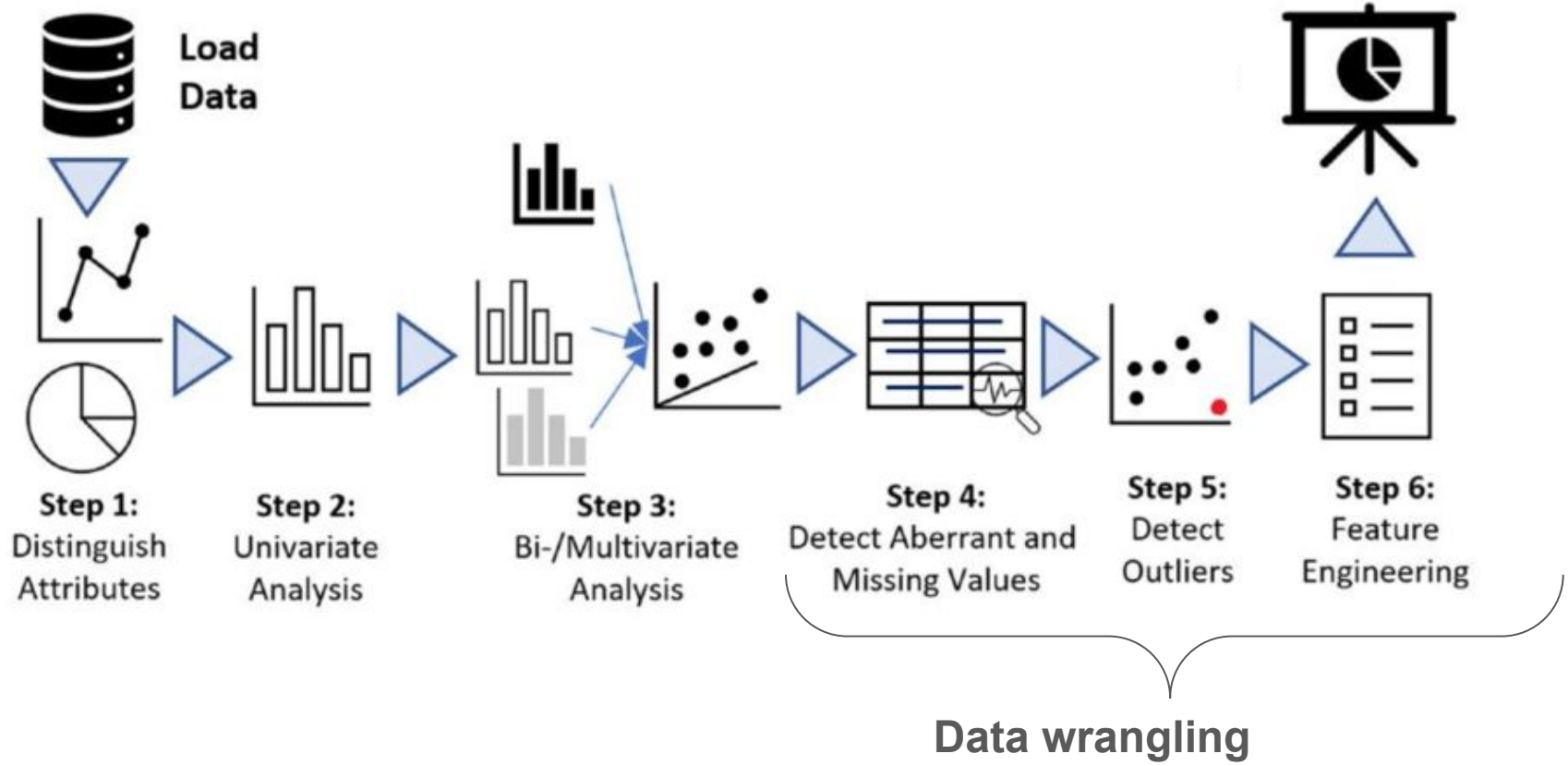
e.g. Male/Female, Black/Brown, etc.

- **Ordinal Data**

e.g. First/Second, Good/Better/Best, etc.



Exploratory Data Analysis (EDA)



Steps of EDA

1. Store data in data structure(s) that will be convenient for exploring/processing

2. Clean/format the data so that:

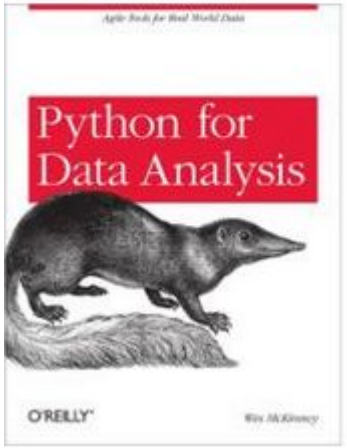
- Each row represents a single object/observation/entry while Each column represents an attribute/property/feature of that entry
- Values are numeric whenever possible
- Columns contain atomic properties that cannot

3. Explore global properties: use histograms, scatter plots, and aggregation functions to summarize the data (can be also step 1)

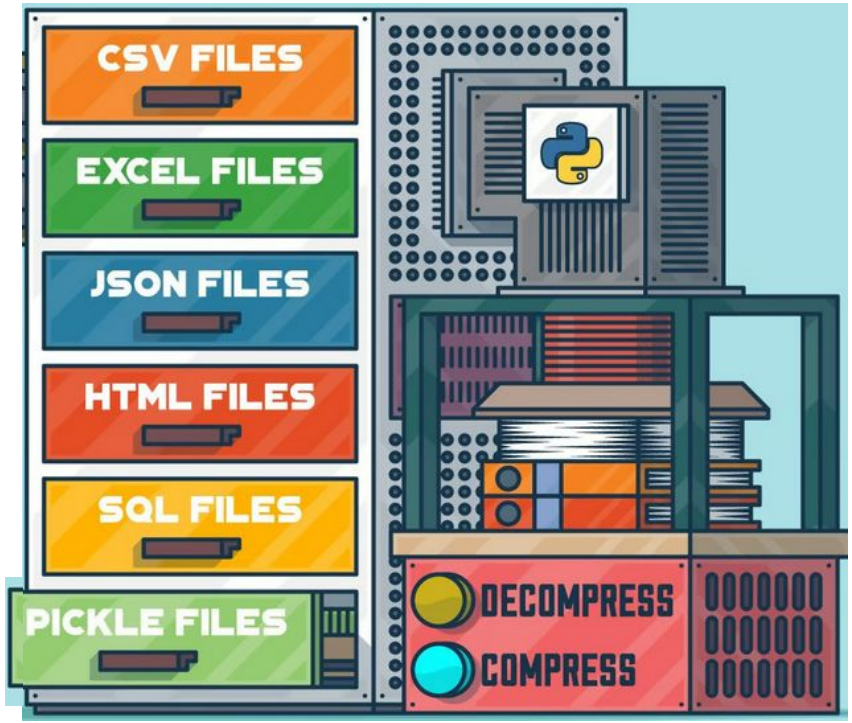
4. Explore group properties: group like-items together to compare subsets of the data (are the comparison results reasonable/expected?)

pandas

- Created in 2008 by Wes McKinney
- acronym for panel data and Python data analysis
- allows to perform data analysis workflow in Python without having to switch to a more domain specific language like R.



Files management in Pandas



Data containers in Pandas

Series

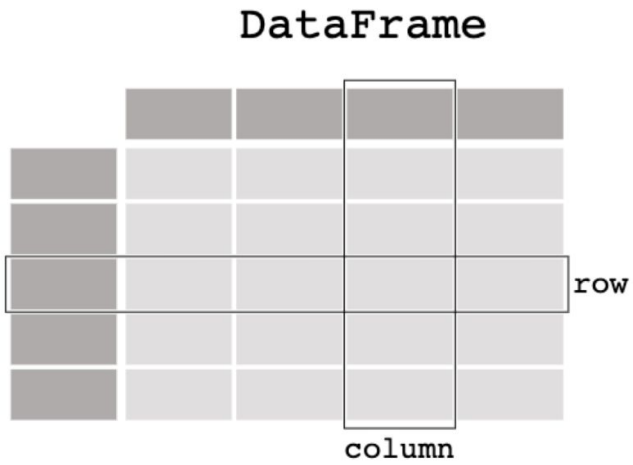
A Series is one-dimensional array-like object containing an array of data (of any NumPy data type) and an associated array of data-labels, called its index.

Dataframe

Tabular, spreadsheet-like data structure containing an ordered collection of columns of potentially different value types (numeric, string, etc.)

it can be regarded as a dict of Series

What a Dataframe looks like…



Visit https://pandas.pydata.org/pandas-docs/stable/getting_started/intro_tutorials/01_table_oriented.html for a more in-depth walkthrough

EDA is an iterative process

First you explore the data graphically

1) Construct graphics to address questions

2) Inspect “answer” and assess new questions

3) Repeat…

keeping an eye on how to transform data appropriately (e.g., invert, log)

Finally, you perform DATA PREPROCESSING

High-level view of dataframes in Pandas

- head() – first N observations
- tail() – last N observations
- describe() – statistics of the quantitative data
- dtypes – the data types of the columns
- columns – names of the columns
- shape – the # of (rows, columns)

EDA with Pandas + seaborn

Summary statistics

df.info()

df.describe()

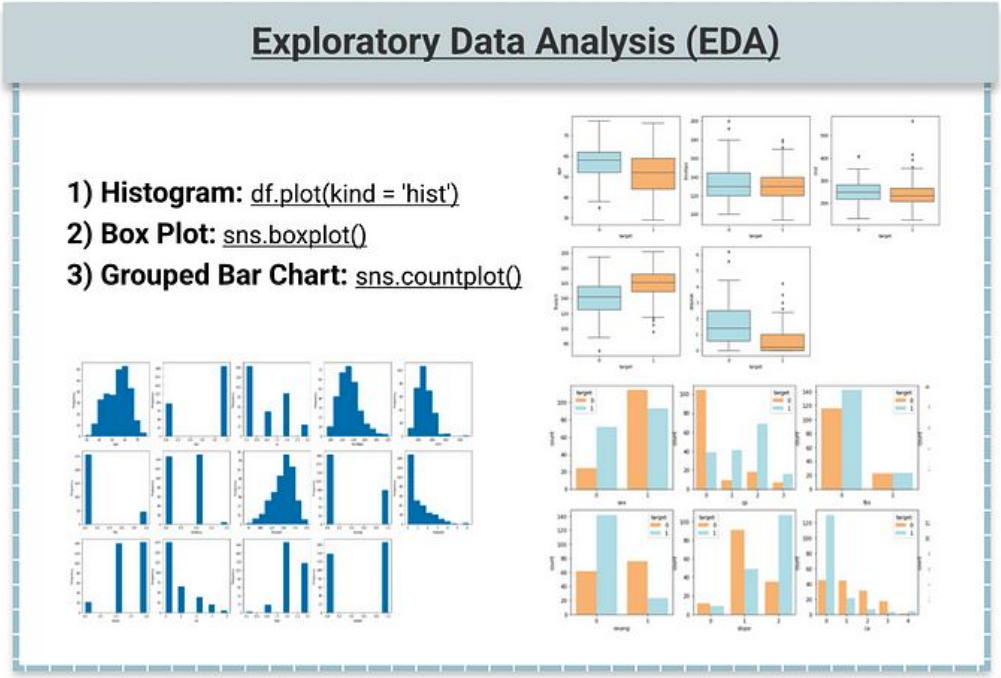
Missing values

.isnull(), .notnull()

Plots

- **Bar charts** - compare different categories.
- **Line charts -** show trends over time or across different categories.
- **Pie charts -** show proportions or percentages of different categories.
- **Histograms -** show the distribution of a single variable.
- **Heatmaps -** show the correlation between different variables.
- **Scatter plots -** show the relationship between two continuous variables.
- **Box plots -** show the distribution of a variable and identify outliers.
- **Violin plots -** check normality assumption

Step 0: visualizing the data

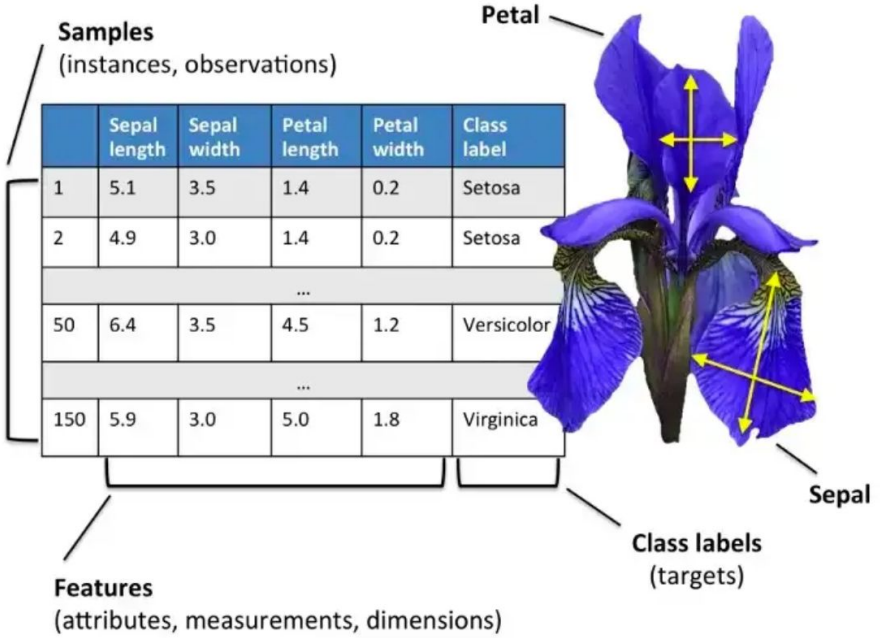


Iris dataset

small classical ML dataset

first work:
Fisher, 1936

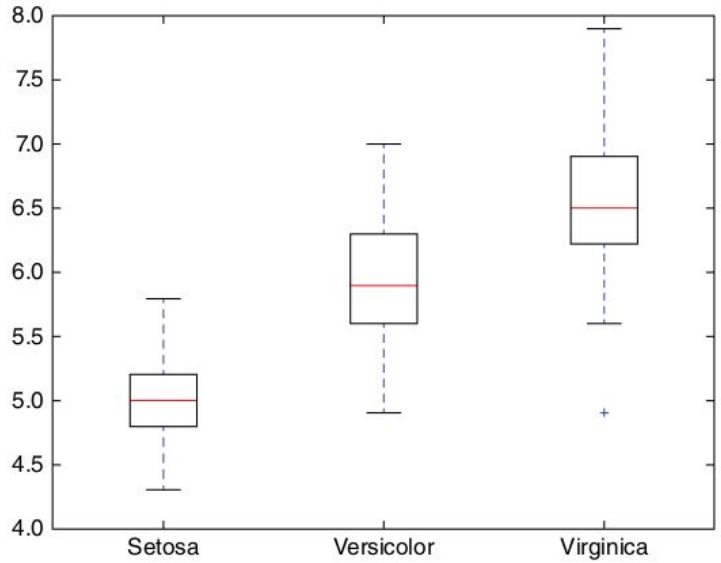
UCI repo:
https://archive.ics.uci.edu/dataset/53/iris



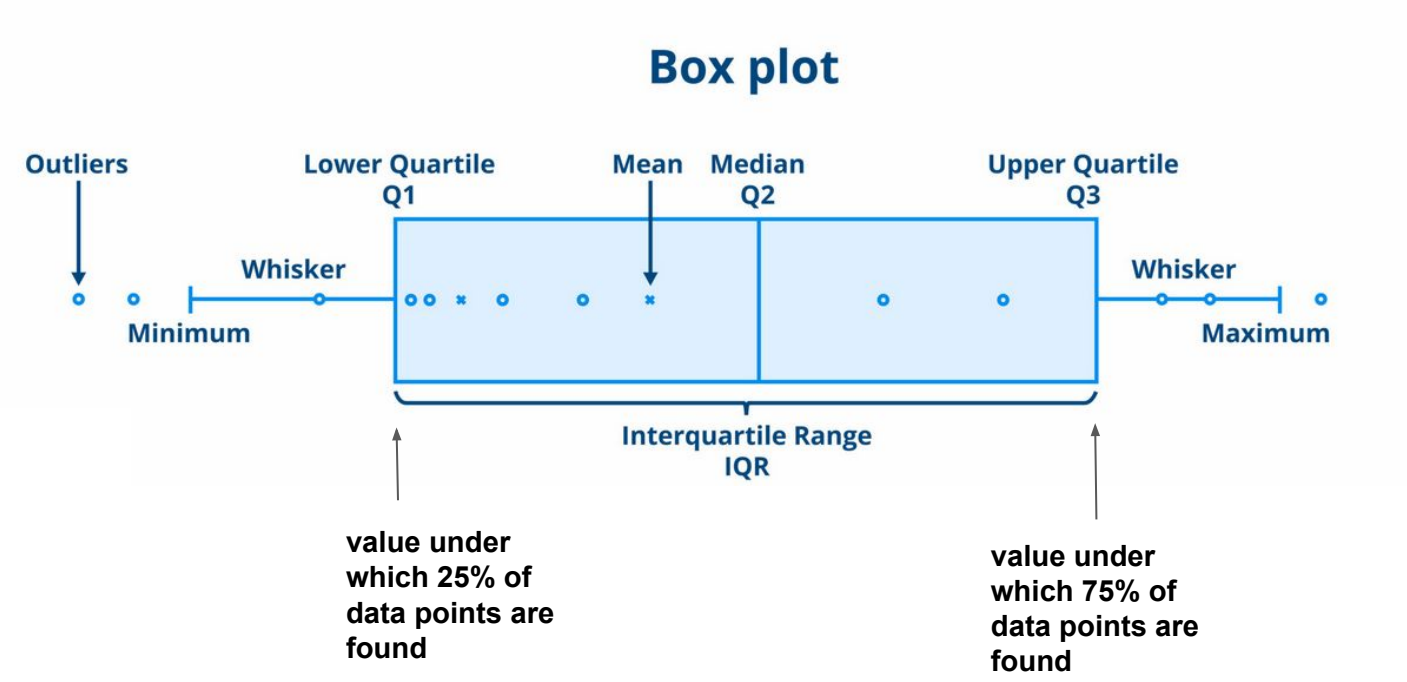
	Sepal length	Sepal width	Petal length	Petal width	Class label
1	5.1	3.5	1.4	0.2	Setosa
2	4.9	3.0	1.4	0.2	Setosa
...					
50	6.4	3.5	4.5	1.2	Versicolor
...					
150	5.9	3.0	5.0	1.8	Virginica

Boxplots

```
col = 'sepal length (cm)'
df['ind'] = pd.Series(df.index).apply(lambda i: i% 50)
df.pivot('ind','species')[col].plot(kind='box')
plt.show()
```



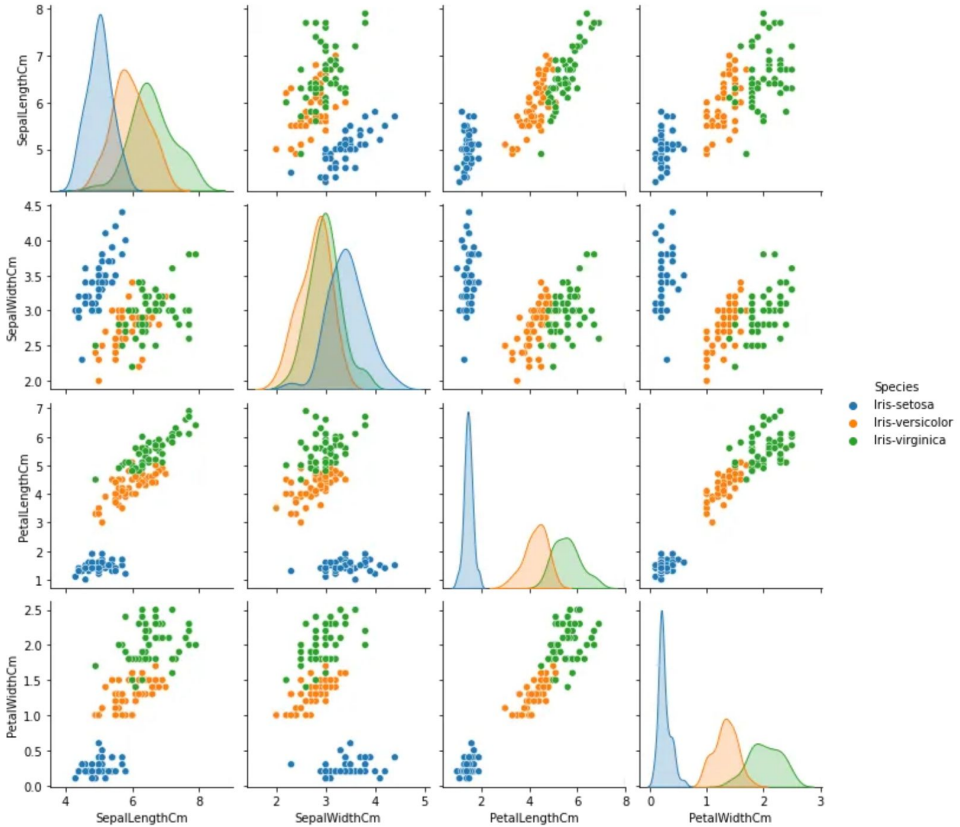
Boxplots


Box plot
Outliers
Lower Quartile
Q1
Mean
Median
Q2
Upper Quartile
Q3
Whisker
Whisker
Minimum
Maximum
Interquartile Range
IQR
value under which 25% of data points are found
value under which 75% of data points are found

Pair plot

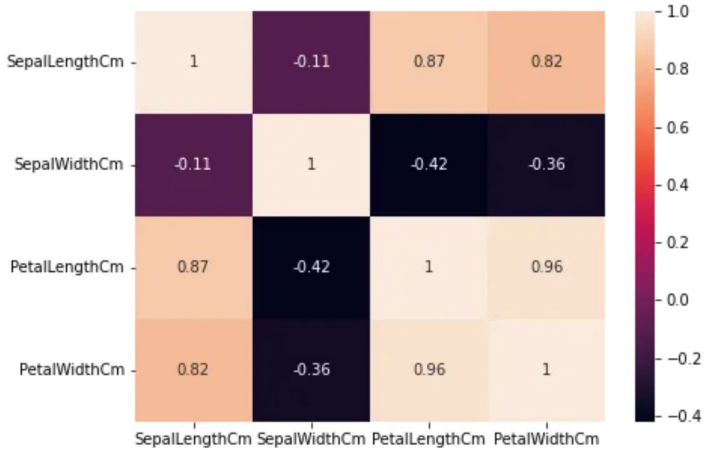
it shows joint and marginal distributions for all pairwise relationships and for each variable, respectively

```
sns.pairplot(data,hue="Species")
plt.show()
```



Plots for Feature association analysis: heatmap

```
plt.figure(figsize=(7,5))
# Plotting the heatmap
sns.heatmap(data.corr(), annot=True)
plt.show()
```

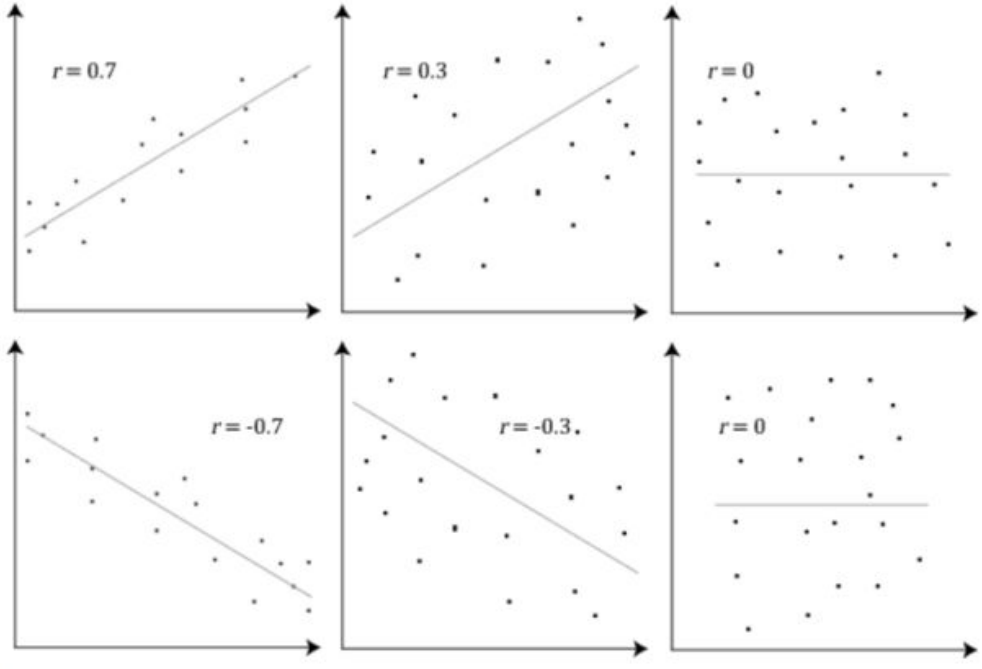


Correlation vs Regression

Correlation analysis: Concerned with measuring the strength and direction of the association between variables. The correlation of X and Y (Y and X).

Linear regression: Concerned with predicting the value of one variable based on (given) the value of the other variable. The regression of Y on X

Pearson Correlation



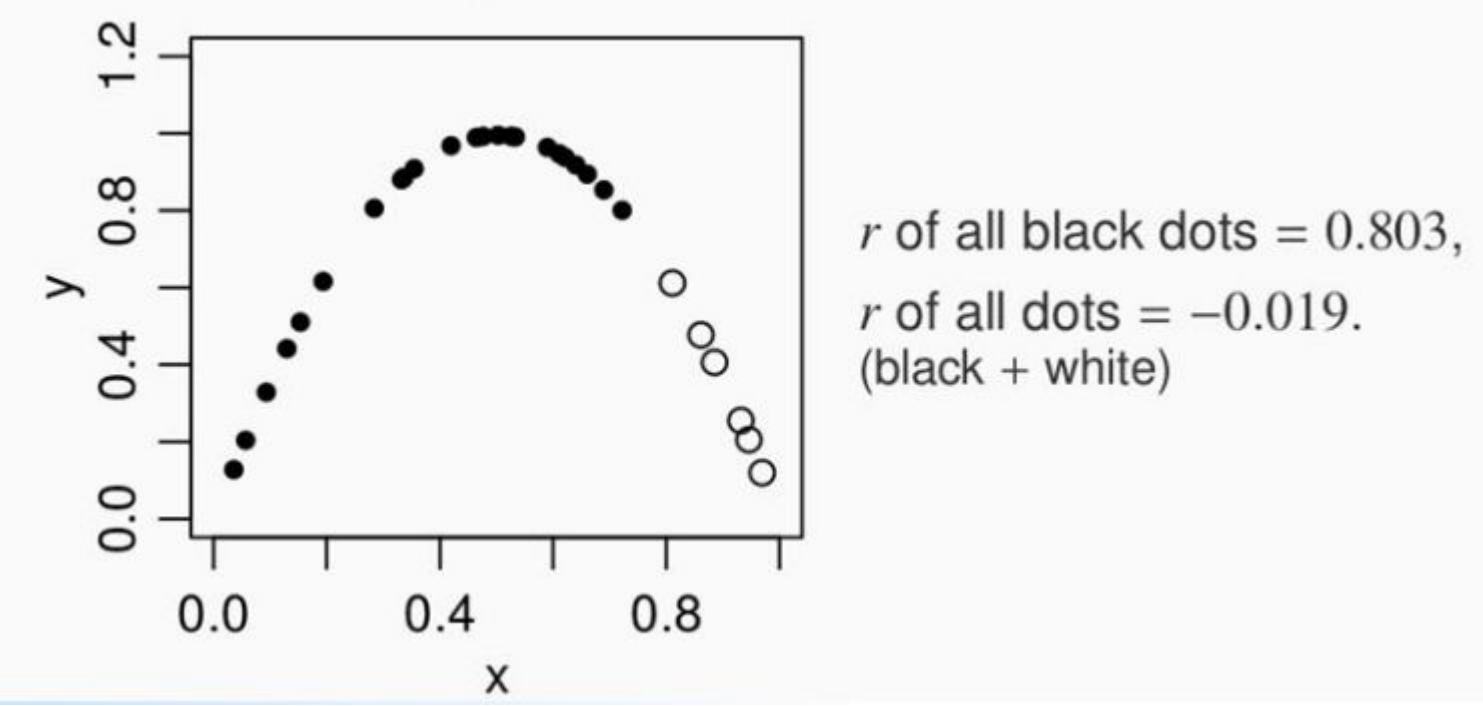
Pearson correlation coefficient

$$r = \frac{1}{n-1} \sum_{i=1}^{n} \left(\frac{x_i - \bar{x}}{s_x} \right) \left(\frac{y_i - \bar{y}}{s_y} \right)$$

Z-score

$$-1 < r(x, y) < 1$$

Pearson cannot measure non-linear correlation!



Step 1-2: Feature cleaning steps

for continuous variables

- outliers removing
- missing value imputation

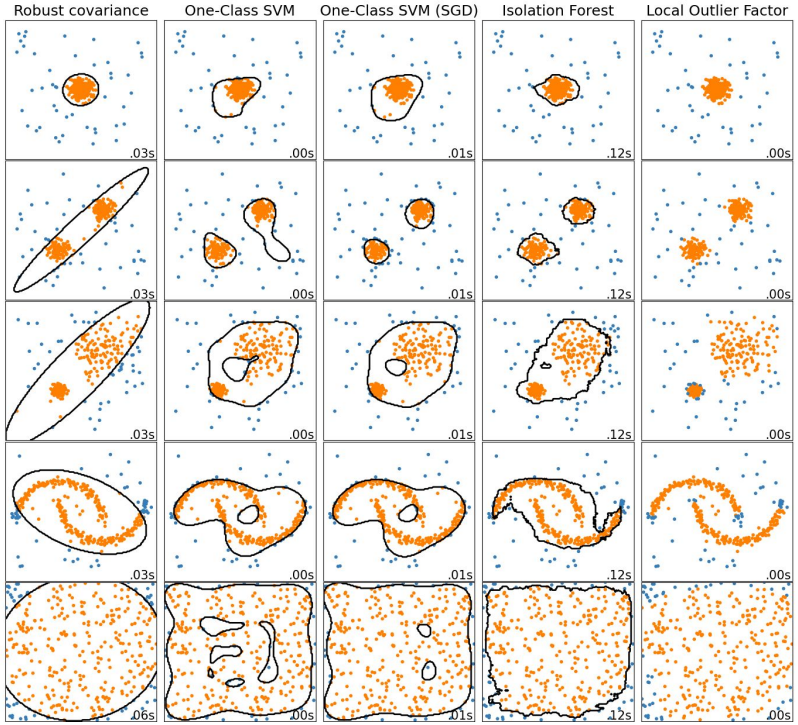
Outliers and novelty detections

(2 faces of anomaly detection)

Many applications require being able to decide whether a new observation belongs to the same distribution as existing observations (it is an *inlier*), or should be considered as different (it is an *outlier*). Often, this ability is used to clean real data sets. Two important distinctions must be made:

outlier detection:	The training data contains outliers which are defined as observations that are far from the others. Outlier detection estimators thus try to fit the regions where the training data is the most concentrated, ignoring the deviant observations.
novelty detection:	The training data is not polluted by outliers and we are interested in detecting whether a **new** observation is an outlier. In this context an outlier is also called a novelty.

Methods for outliers detection in scikit-learn



Step 3-4: Feature engineering steps

for continuous variables

- feature scaling (also called data standardization)
- feature corr. analysis and dimensionality reduction

Feature scaling

Why?

Feature scaling is essential for machine learning algorithms that calculate **distances between data**. If not scaled, the feature with a higher value range starts dominating when calculating distances

Functions in Scikit-learn

1. Min Max Scaler
2. Standard Scaler
3. Max Abs Scaler
4. Robust Scaler
5. Quantile Transformer Scaler
6. Power Transformer Scaler
7. Unit Vector Scaler

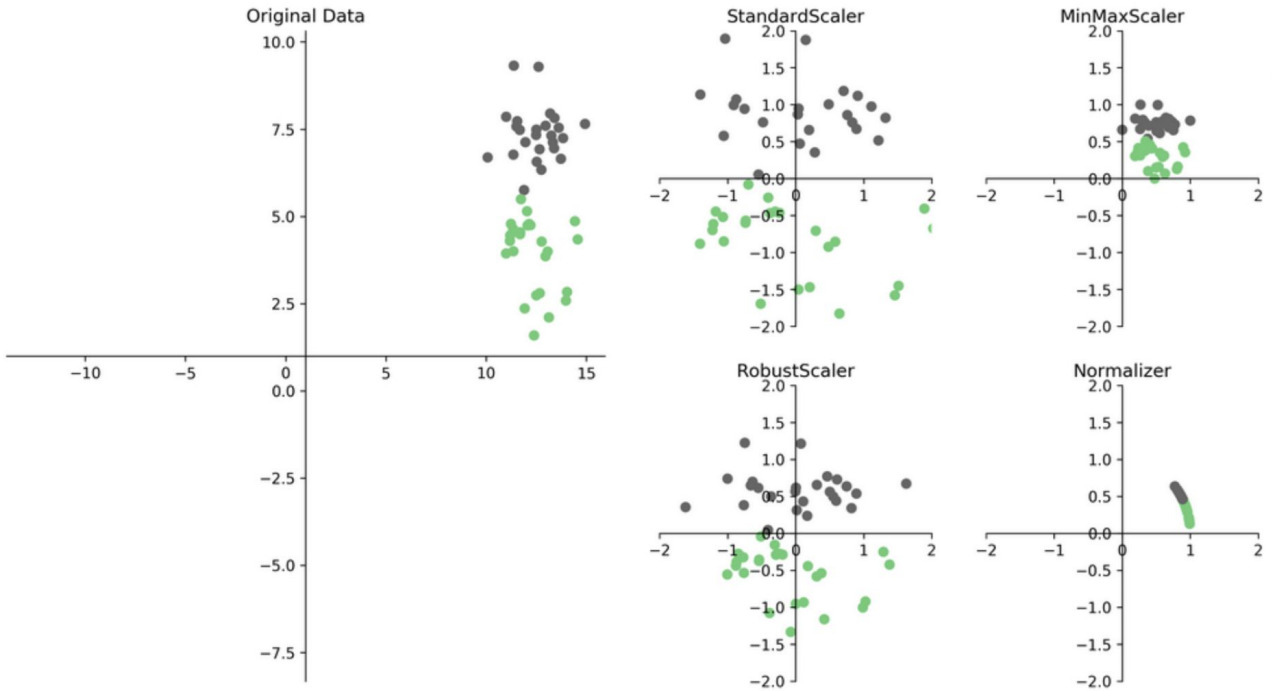
Example of feature scaling in scikit-learn

`QuantileTransformer` provides a non-parametric transformation to map the data to a uniform distribution with values between 0 and 1

```
>>> from sklearn.datasets import load_iris
>>> from sklearn.model_selection import train_test_split
>>> X, y = load_iris(return_X_y=True)
>>> X_train, X_test, y_train, y_test = train_test_split(X, y, random_state=0
>>> quantile_transformer = preprocessing.QuantileTransformer(random_state=0)
>>> X_train_trans = quantile_transformer.fit_transform(X_train)
>>> X_test_trans = quantile_transformer.transform(X_test)
>>> np.percentile(X_train[:, 0], [0, 25, 50, 75, 100])
array([ 4.3,  5.1,  5.8,  6.5,  7.9])
```

Visualizing the rescaling

$$z = (x - u) / s$$

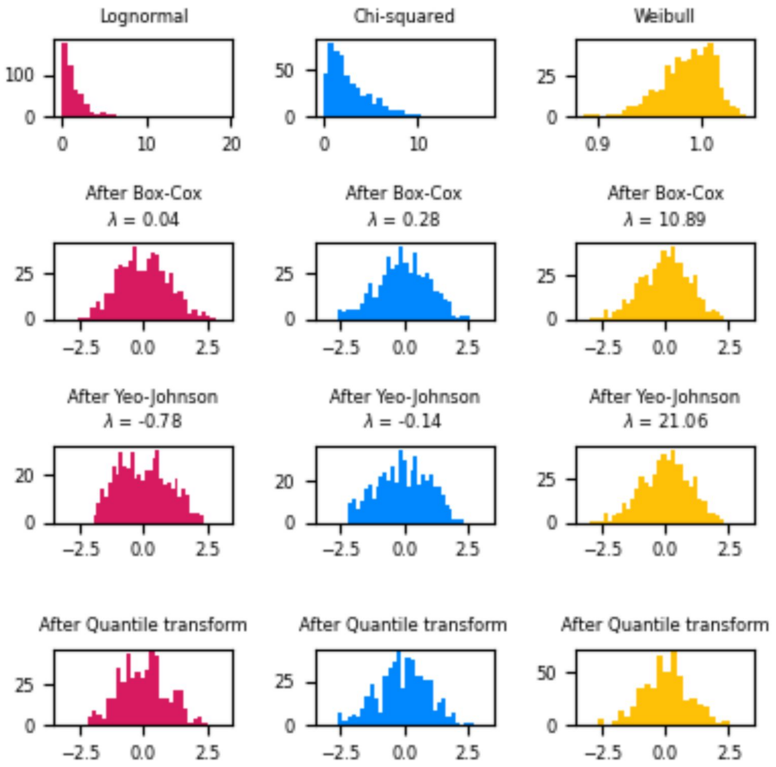


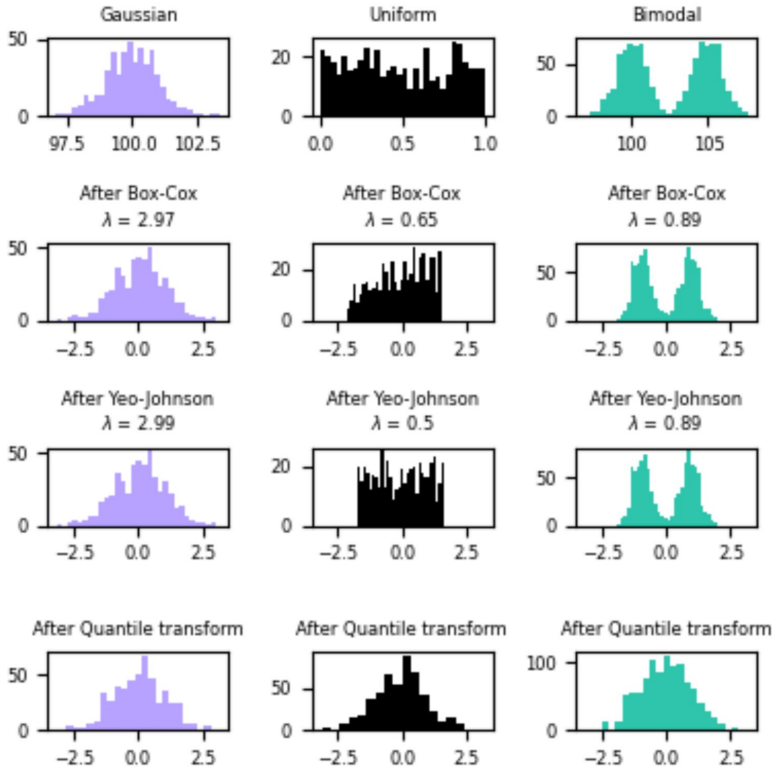
$$X_{new} = \frac{X_i - \min(X)}{\max(x) - \min(X)}$$

$$x' = \frac{x}{||x||}$$

$$X_{new} = \frac{X - X_{median}}{IQR}$$

Visualizing different non-linear transformation in sklearn





Feature preprocessing for categorical variables

The two most common techniques

Label encoding

- ✓ Label encoding returns different values for different classes, bringing in a natural ordering
- ✓ *Recommended when* it's reasonable to assume some sort of (equally spaced) ordering in your categorical feature

country	country
France	1
Italy	0
Spain	2

teaching_level	teaching_level
Beginner	0
Medium	1
Advanced	2

One Hot encoding

- ✓ One hot encoding returns as many dummy variables (0/1 columns) as the classes of the categorical feature
- ✓ *Recommended when* no ordering can be assumed in your categorical feature

country	country_France	country_Italy	country_Spain
France	1	0	0
Italy	0	1	0
Spain	0	0	1

special case: dummy encoding

Sklearn API for estimators (models) VS transformers

```
estimator.fit(X, [y])
```

`estimator.predict`	`estimator.transform`
Classification	Preprocessing
Regression	Dimensionality reduction
Clustering	Feature selection
	Feature extraction

Supervised learning framework

- Training : $\{(\boldsymbol{x}_1, y_1), (\boldsymbol{x}_2, y_2), \ldots, (\boldsymbol{x}_n, y_n)\}$
- $\boldsymbol{x}_i$: input vector

$$\boldsymbol{x}_i = \begin{bmatrix} x_{i,1} \\ x_{i,2} \\ \vdots \\ x_{i,n} \end{bmatrix}, \quad x_{i,j} \in \mathbb{R}$$

- y : response variable
  - $y \in \{-1, 1\}$: binary classification
  - $y \in \mathbb{R}$: regression
  - what we want to be able to predict, having observed some new $\boldsymbol{x}$.

Regression model

A generic regression model connecting j **Explanatory variables** (x_j) to the **response variable** (y), is described by the following formula

$$y = f(\alpha_i, x_j), \qquad \begin{cases} i = 1, n \\ j = 1, m \end{cases}$$

GOALS

PREDICTION & FORECASTING

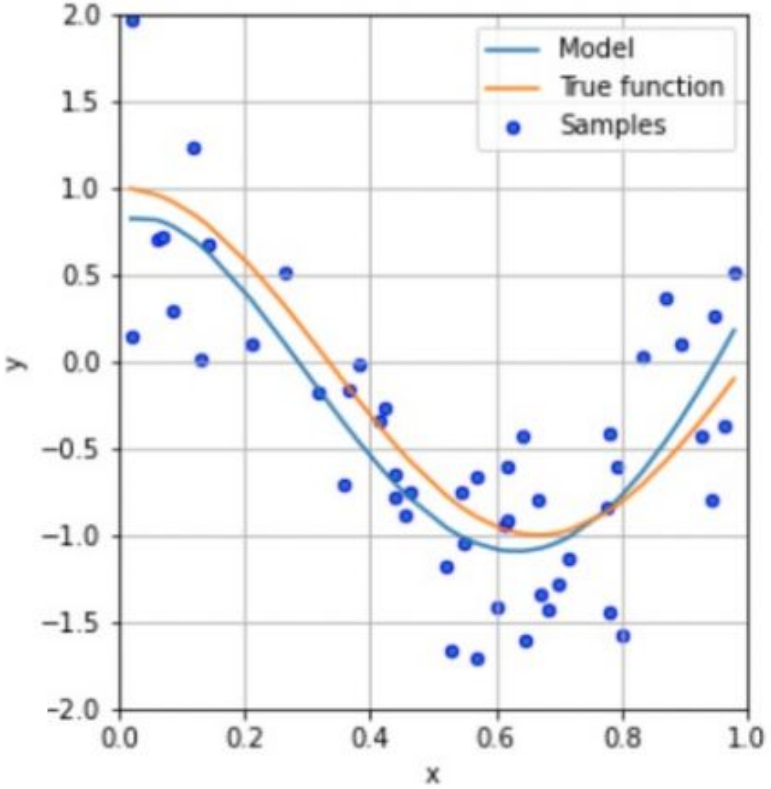
Understanding and modelling the **functional relation** between observations and a target phenomenon means to be able to **predict the behaviour of the phenomenon** in response to a new set of measurements

INTERPRETATION

Depending on the functional relation between the phenomenon and the observations it is possible to **understand the relevance of each variable** in the determination of the phenomenon

Practical definition

A regression model allows to find an approximation of the unknown true function on the basis of a sample of observed data which come from that function (experiment)



Types of regression more used

- **Linear regression**
- **Regularized/penalized linear regression (Lasso, Ridge)**
- **GAM regression**
- **Bayesian Ridge regression**
- **Gaussian process regression (non-parametric)**

The first two regression can be implemented with OLS method or with SGD.

In scikit-learn, this implementations correspond to:

1. LinearRegression, Ridge, Lasso
2. SGDRegressor

When can we apply Linear Regression?

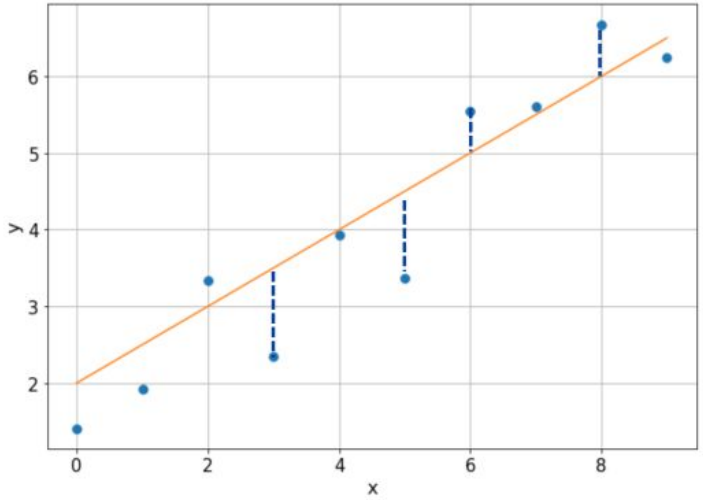
MODEL HYPOTHESES

1. **X is fully deterministic** with $\mathrm{m} \leq n$ and lacks of perfect multicollinearity
2. The error term ε adds *noise* to the linear relationship between the dependent variable and regressors and it is assumes to have 0 mean as it should not depend on any systematic effect: $\mathbb{E}(\boldsymbol{\varepsilon}) = \mathbf{0}$ *
3. The n observations are assumed to be independent ($i.i.d.$ sample), therefore the errors ε_i are not correlated and $\mathbf{Var}(\boldsymbol{\varepsilon_i})$ **is assumed to constant** (σ^2) across all the data instances (**homoscedasticity**) *
4. (*Strong hypothesis –needed for statistical inference on the model parameter estimates)* errors are normally distributed*:* $\boldsymbol{\varepsilon \sim N(0, \sigma^2 I)}$

Linear regression

Model definition: $y = \beta_0 + \beta_1 x + \varepsilon$, where:

- β_0 is the intercept of the model,
- β_1 represents the slope



- The best model is the one that minimises the *sum of squared (SS) residuals* over the n observations:

$$\hat{\beta} = \underset{\beta}{\operatorname{argmin}} \sum_{i=1}^{n} [y_i - f(\beta, x_i)]^2$$

COST FUNCTION

- The algorithm used to find the optimal values $\hat{\beta}_0$ and $\hat{\beta}_1$ is called **OLS** (Ordinary Least Squares)

$$\hat{y} = \hat{\beta}_0 + \hat{\beta}_1 x$$

Ordinary Least Squares

$$\hat{y} = w^T \mathbf{x} + b = \sum_{i=1}^{p} w_i x_i + b$$

$$\min_{w \in \mathbb{R}^p, b \in \mathbb{R}} \sum_{i=1}^{n} ||w^T \mathbf{x}_i + b - y_i||^2$$

Unique solution if $\mathbf{X} = (\mathbf{x}_1, \ldots \mathbf{x}_n)^T$ has full column rank.

Metrics for Model performances

Root Mean squared error

$$RMSE = \sqrt{\frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{n}}$$

Residual standard error

$$RSE = \sqrt{\frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{(n - p - 1)}}$$

R-squared

(coefficient of determination)

$$R^2 = 1 - \frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{n}(y_i - \bar{y})^2} = \underbrace{\frac{SS_{Reg}}{SS_{Tot}} \in [0,1]}_{\text{for linear reg}}$$

MSE vs RMSE vs MAE

- $MSE = \frac{1}{n}\sum_{i=1}^{n}(\hat{y}_i - y_i)^2$ — It assigns more weight to bigger errors, useful when interested in **detecting outliers**
- $RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(\hat{y}_i - y_i)^2}$ — It maintains the same properties as the MSE, but it is in the same scale as y
- $MAE = \frac{1}{n}\sum_{i=1}^{n}|\hat{y}_i - y_i|$ — It is simply the average difference observed in the predicted and actual values across the whole test set, useful if not particularly interested in outliers

Ridge Regression

$$\min_{w \in \mathbb{R}^p, b \in \mathbb{R}} \sum_{i=1}^{n} (w^T \mathbf{x}_i + b - y_i)^2 + \alpha ||w||^2$$

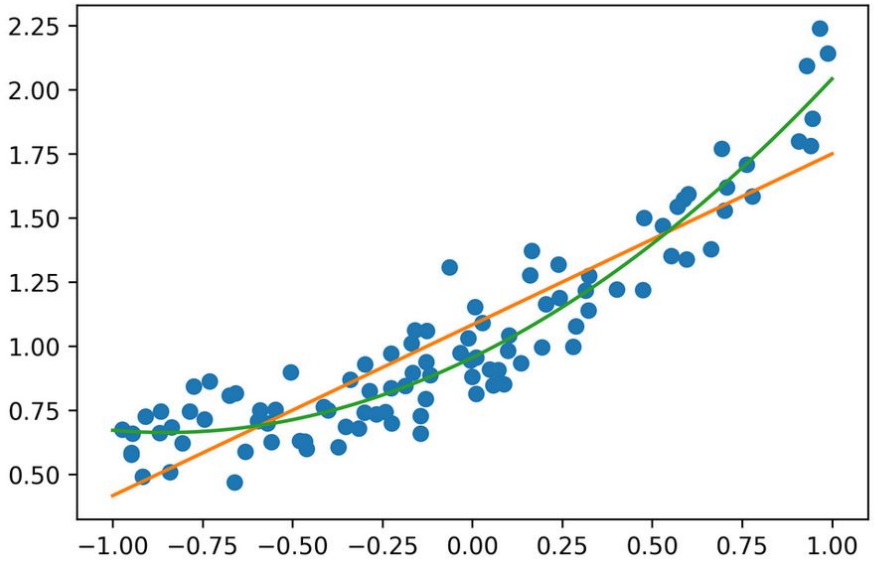
L2 regularization

Always has a unique solution.

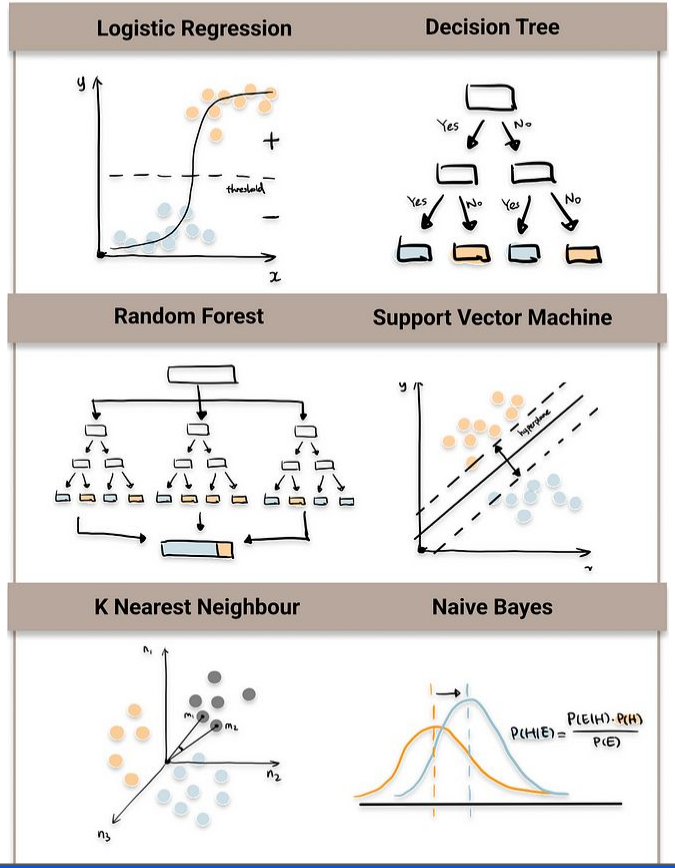
Tuning parameter alpha.

Polynomial feature regression

```
from sklearn.preprocessing import PolynomialFeatures
poly = PolynomialFeatures()
X_bc_poly = poly.fit_transform(X_bc_scaled)
print(X_bc_scaled.shape)
print(X_bc_poly.shape)
```



Classification models



The framework of classification models

- They all share the same schema, by having:
  - a response variable (**target**) of categorical nature,
  - a number of explanatory variables (**features**) which are assumed to be helpful in classifying each observation,
  - and the following objectives

CLASSIFICATION

Predicting the **score or probability of an observation to belong to a given class** depending on a set of properties that describe that observation

INTERPRETATION

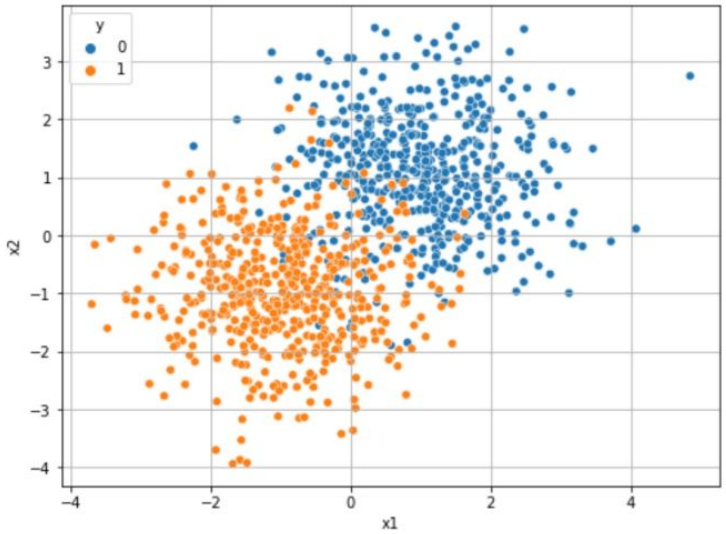
Assessing how much each property is relevant in defining the categories and which set of properties better describe the population belonging to a certain class

Binary classification problem

target

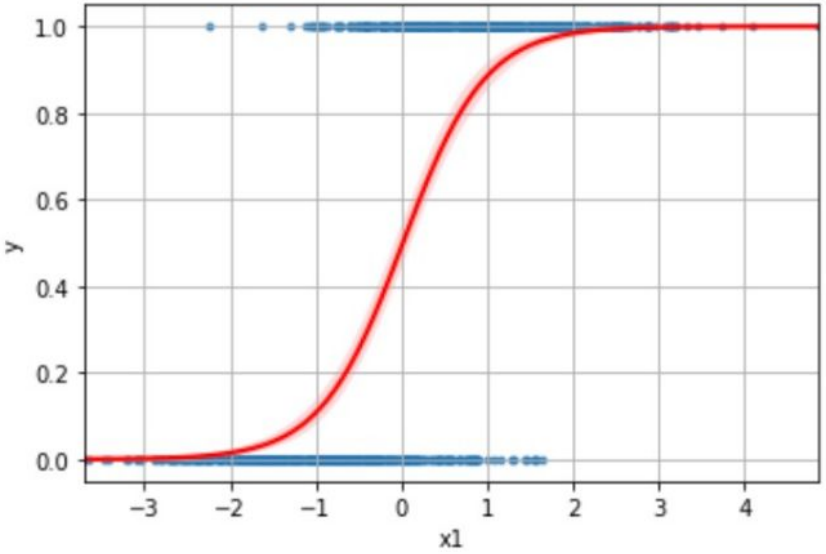
$$y = \begin{cases} 0, & the\ condition\ is\ absent \\ 1, & the\ condition\ is\ present \end{cases}$$

1 is typically called POSITIVE class
0 is called NEGATIVE class



Logistic regression for binary classification

The logistic regression aims at modelling a function with this shape (also called **logistic or sigmoid function**):



- It is bounded in $[0, 1]$ as probabilities also are
- It is S-shaped, being is a cumulative distribution along the variable $\int_{x0}^{x1} pdf(x)\,dx$

$$p(x;\beta) = \frac{e^{\beta_0 + \vec{\beta}\cdot\vec{x}}}{1 + e^{\beta_0 + \vec{\beta}\cdot\vec{x}}}$$

Main components of the logistic regression?

1. A **feature representation** of the input. For each input observation $x^{(i)}$, this will be a vector of features $[x_1, x_2, ..., x_n]$. We will generally refer to feature i for input $x^{(j)}$ as $x_i^{(j)}$, sometimes simplified as x_i, but we will also see the notation f_i, $f_i(x)$, or, for multiclass classification, $f_i(c, x)$.
2. A classification function that computes $\hat{y}$, the estimated class, via $p(y|x)$. In the previous slide we introduced the **sigmoid**
3. An objective function for learning, usually involving minimizing error on training examples. We will introduce the **cross-entropy loss function**.
4. An algorithm for optimizing the objective function. We introduce the **stochastic gradient descent** algorithm.

Stochastic gradient descent

1. Initialize $\mathbf{w} := \mathbf{0} \in \mathbb{R}^m$, $\mathbf{b} := 0$
2. For every training epoch:
   A. For every $\langle \mathbf{x}^{[i]}, y^{[i]} \rangle \in \mathcal{D}$
      (a) $\hat{y}^{[i]} := \sigma\left(\mathbf{x}^{[i]T}\mathbf{w} + b\right)$
      (b) $\nabla_{\mathbf{w}}\mathcal{L} = -\left(y^{[i]} - \hat{y}^{[i]}\right)\mathbf{x}^{[i]}$
          $\nabla_b\mathcal{L} = -\left(y^{[i]} - \hat{y}^{[i]}\right)$
      (c) $\mathbf{w} := \mathbf{w} + \eta \times \left(-\nabla_{\mathbf{w}}\mathcal{L}\right)$
          $b := b + \eta \times \underbrace{\left(-\nabla_b\mathcal{L}\right)}$

learning rate

negative gradient

Note

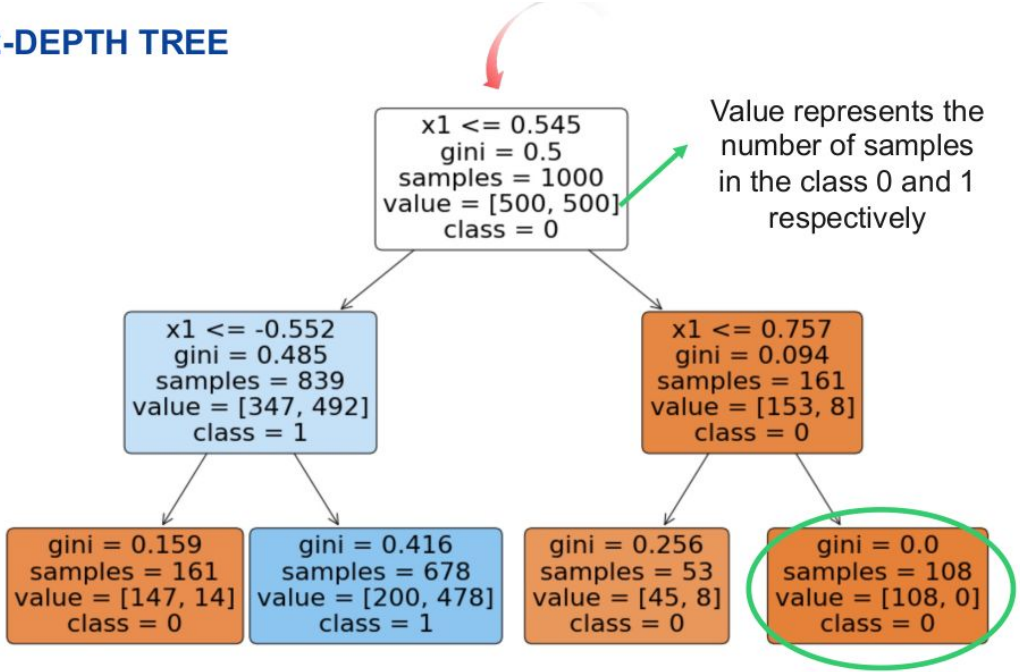
$a - y \Leftrightarrow -\left(y^{[i]} - \hat{y}^{[i]}\right)$

Logistic regression in scikit-learn

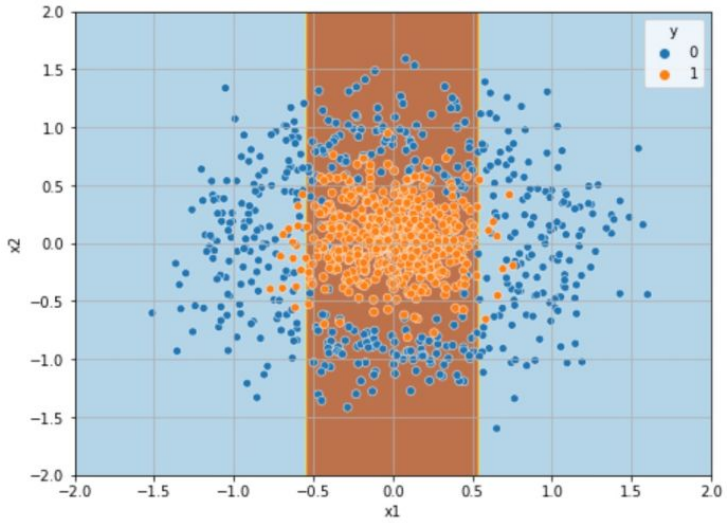
```
# import the necessary libraries
from sklearn.datasets import load_breast_cancer
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score

# load the breast cancer dataset
X, y = load_breast_cancer(return_X_y=True)

# split the train and test dataset
X_train, X_test,\
    y_train, y_test = train_test_split(X, y,
                                       test_size=0.20,
                                       random_state=23)
# LogisticRegression
clf = LogisticRegression(random_state=0)
clf.fit(X_train, y_train)

# Prediction
y_pred = clf.predict(X_test)

acc = accuracy_score(y_test, y_pred)
print("Logistic Regression model accuracy (in %):", acc*100)
```

Example of decision tree on 2D data

2-DEPTH TREE

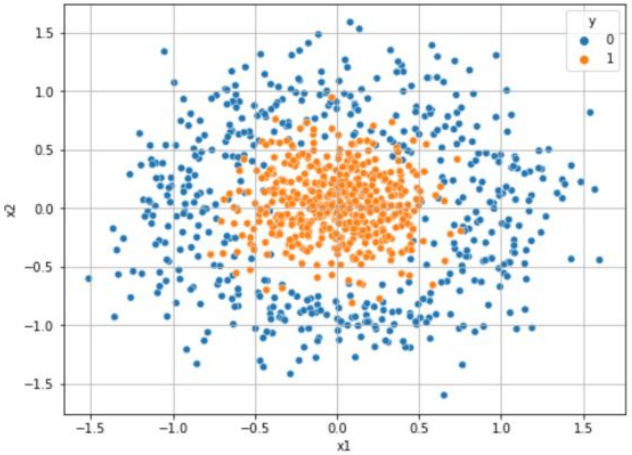


Decision boundaries

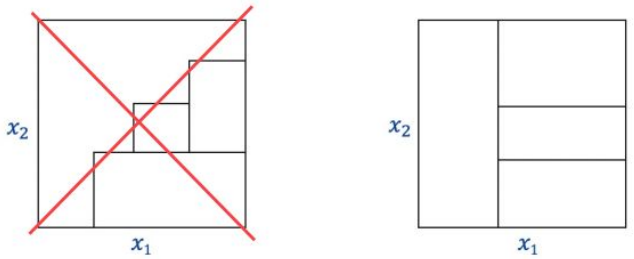


How do decision trees actually works?

TASK: fit a 2-depth tree on the following data



- The algorithm aims at partitioning the two-dimensional features space by **recursive binary splitting (sub-optimal solution)**



- At each node, all the available **features are evaluated**
- A given **objective function** is then optimized to find the most informative feature and point for the split

How do decision trees actually works?

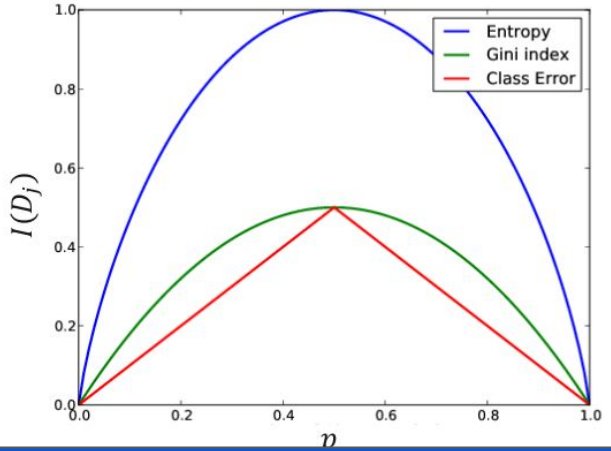
Splitting algorithm

1. Calculate the $I(D)$ of the parent node
2. Calculate the weighted average $I(D)$ of child nodes
3. Subtract the two quantities (estimate IG)
4. Select the split with the highest information gain

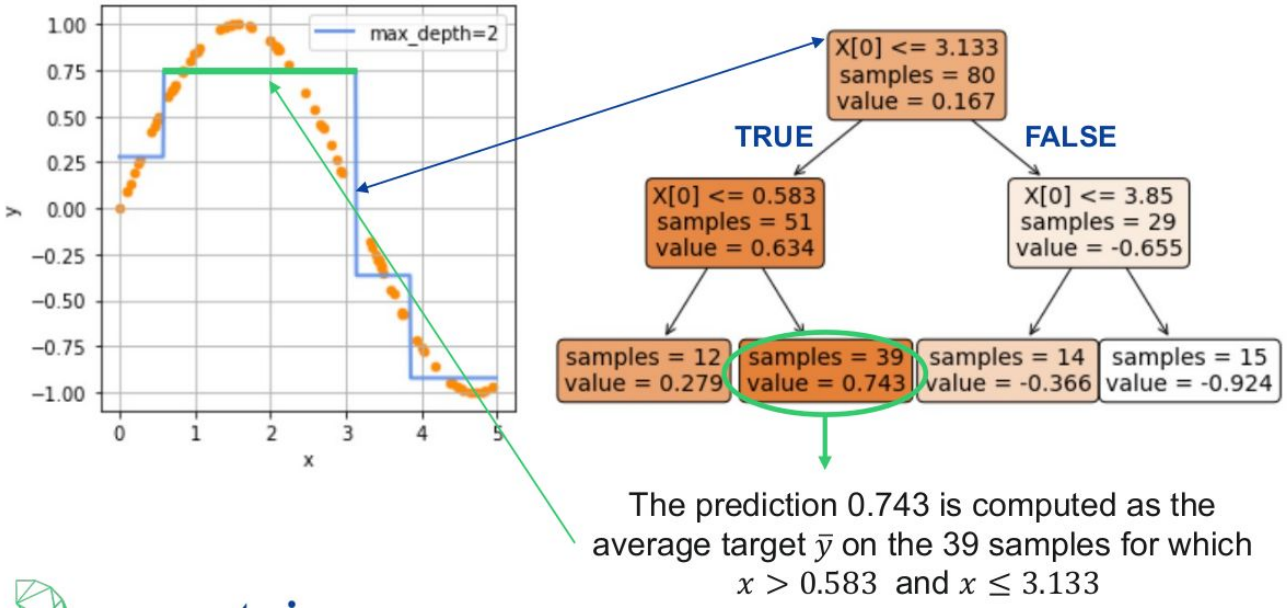
$$IG(D_p) = I(D_p) - \sum_{j=1}^{m} \frac{n_j}{n_p} I(D_j)$$

Gini index: $I(D_j) = \sum_{k=0,1} p_{jk}(1 - p_{jk})$

Entropy: $I(D_j) = -\sum_{k=0,1} p_{jk} \log_2(p_{jk})$



Regression Trees

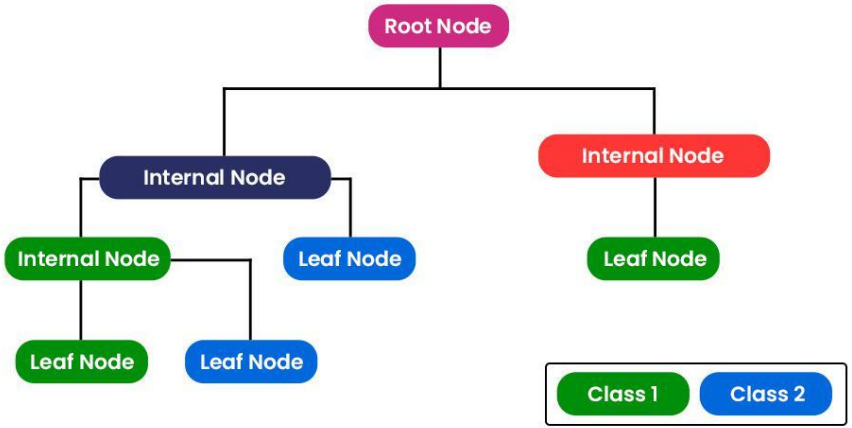


- The **root node** is the top-most node of a decision tree and represents the entire population
- Every **internal node** test a condition on a given attribute
- A **branch** represents the outcome of the test
- Final nodes are called **tree leaves** and represents the final predictions or classifications

Training Decision trees in sklearn -> the CART algo

Scikit-Learn uses CART algorithm to train Decision Trees (also called “growing” trees).

CART(Classification And Regression Trees) is a variation of the decision tree algorithm, first developed by Leo Breiman, Jerome Friedman, Richard Olshen, and Charles Stone in 1984.



CART uses a **greedy approach** to split the data at each node.

It evaluates all possible splits and selects the one that best reduces the impurity of the resulting subsets.

Splitting criterion

For classification: Gini impurity
For regression: residual reduction

Beauty and limitations of Decision Trees

PROS

- It is a very simple, intuitive and interpretable model
- Computationally not expensive
- It automatically does feature selection
- It can approximate almost any function and the tree structure can always be visualized, even in high dimensional space
- It is a suitable model both for regression and classification
- It handles all features type
- It handles missing values

CONS

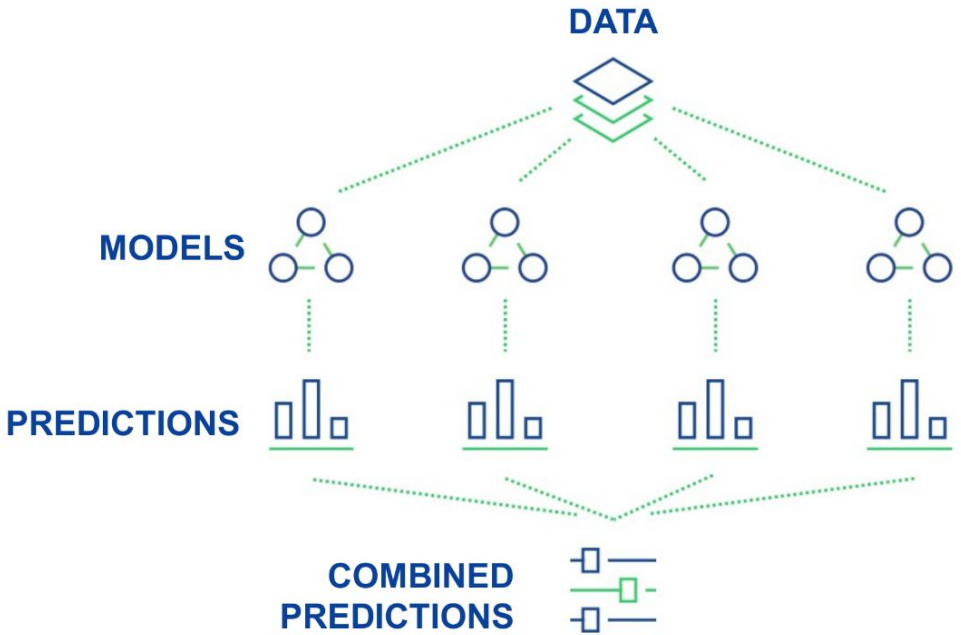
- It easily run in overfitting
- It can only describe the target behavior when the variables are within the range they had in the training set. Anything outside that range will be approximated with the same value (there is no underlying functional form)
- In presence of correlated variables the training could fall in a “local” optimal solution, which might be very different from the “global” one
- It is not stable: small variations in the training set often result in big differences in the structure of the tree

Ensemble models

Models may differ due to

- different sampling of training data
- different features set
- randomized parameters within the algorithm
- different algorithm itself

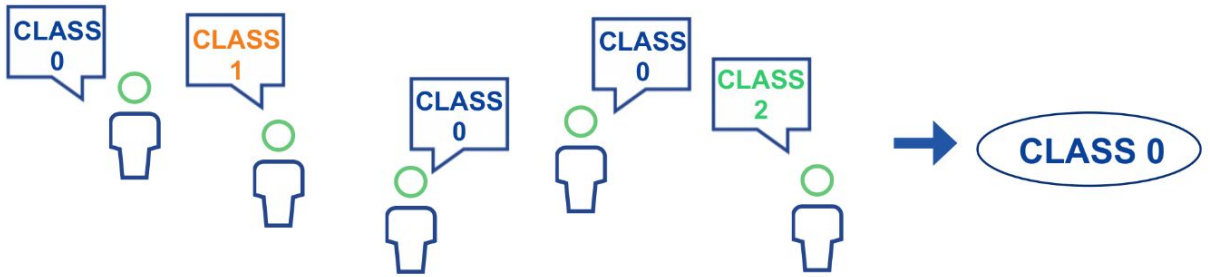
The more they differ, the better is expected to be the quality of their combination, as they capture different aspects of the phenomenon



The power of collective knowledge

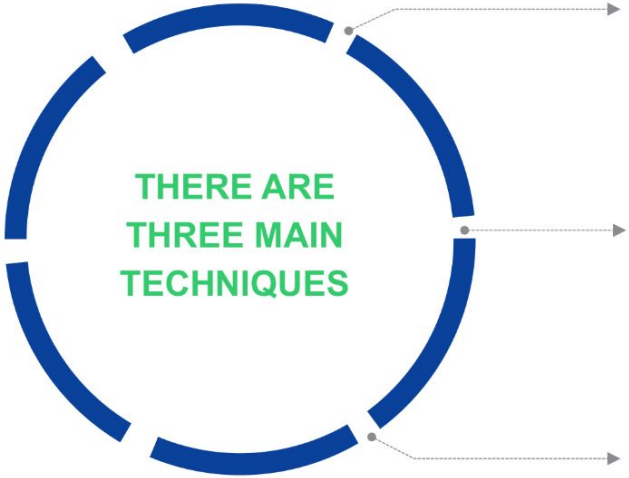
Ensemble models can be used both in regression and classification tasks

- **REGRESSION**: the final predictions can be the average or weighted average of the single models' predictions
- **CLASSIFICATION**: as before, the average or weighted average may be taken to compute the scores. The final classification is based on the **majority vote** among the single classifiers



Wisdom of Crowds
(Surowiecki, 2004)
The collective knowledge of a diverse and independent body of people typically exceeds the knowledge of any single individual, and can be harnessed by voting

Ensemble models



BAGGING → parallel training with different training sets

BOOSTING → sequential training, iteratively re-weighting training examples so current classifier focuses on hard examples

RANDOM FOREST → parallel training with different training sets and feature sets

Bagging

DATASET

Let $Z = \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_n, y_n)\}$ be the complete training set

SAMPLING

Create m subsamples Z_i of size B out of Z using **bootstrapping** techniques (random sampling with replacement)

BASE LEARNER

Let $C_i(x)$ be the classifier trained on the subsample Z_i (**base classifier**)

COMBINED CLASSIFIER

Finally define the **ensemble classifier** as the average prediction over the m base learners:

$$C_{bag}(x) = \frac{1}{m}\sum_{i=1}^{m} C_i(x)$$

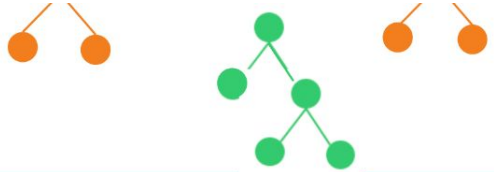
What is a Bootstrap sampling?

Bootstrap samples are expected to **approximately** be **representative and independent samples** of the true data distribution (almost $i.i.d.$)

- For n big enough, sampling from Z is like sampling from the true data distribution
- If $n >> B$ the samples should not be much correlated

We are thus assuming to fit and then average m independent models…

Base/weak classifiers



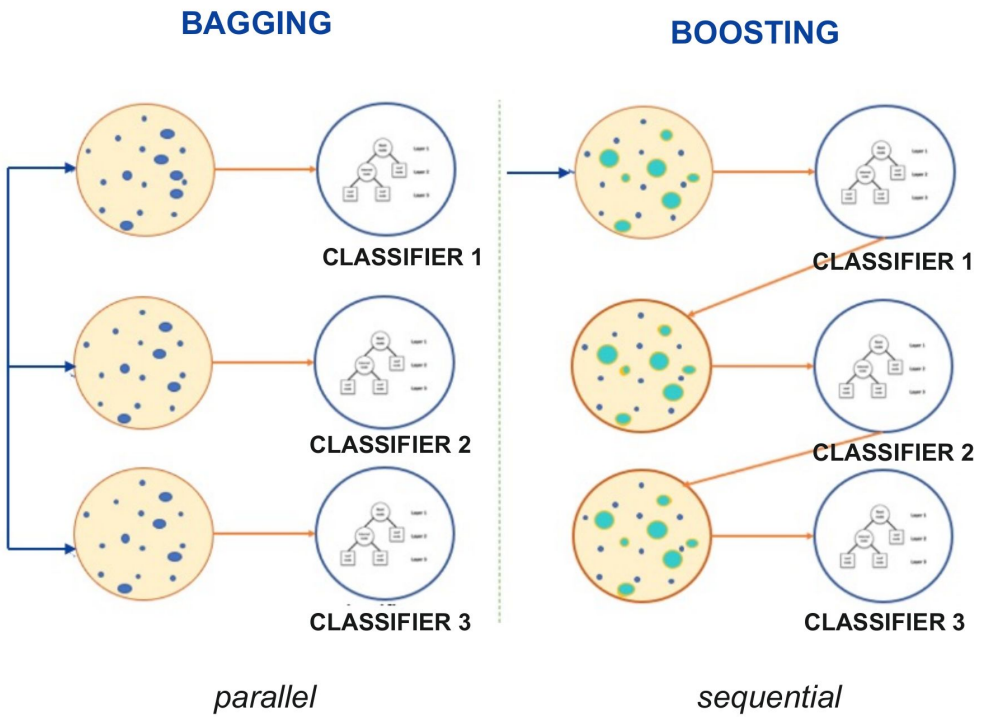
- We call **base or weak classifier** the model used as **building block of the ensamble model**
- The base classifier is trained on **subsamples** of the training data, so that:
  - it should learn the details of the specific subsample
  - does not need to be much granular (e.g. a very deep decision tree) to learn details because the subsample has a relatively small size, hence less information to draw from (→ ***weak***)
- As it is trained multiple times, the **computational cost** of training the model is a relevant aspect to consider
- A very widespread choice is the **decision tree**
  - It is a highly unstable model (**high variance**: any little perturbation of the training data may result in significantly different tree structures)
  - Notoriously very fast to run

Boosting

What differs is the **strategy** used to generate the subsets:

- *bagging* uses bootstrap sampling,
- *boosting,* at each step, increases the probability of picking events that were misclassified at the previous step

In the boosting model, each *weak classifier* is mostly focused on **learning how to correct the errors** done by the previous *weak classifier*

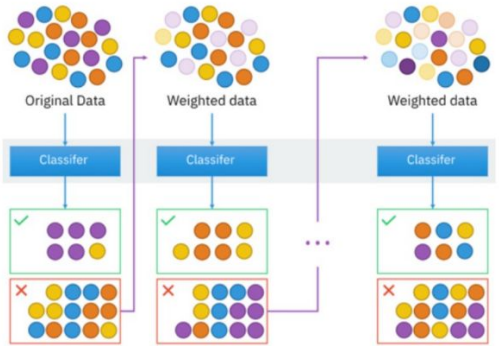


Boosting

The algorithm

INITIALIZATION

- Let $Z = \{(x_1, y_1), \ldots, (x_n, y_n)\}$ be the complete training set with $y_i \in \{-1, +1\}$
- Weights: ${w_i}^1 = 1/n$ with $i = 1, \ldots, n$ where n is the total number of events in the training set, while the 1 sup-index denotes the 1-st iteration of the algorithm



Extract a sample Z_j of the dataset Z
bootstrapping with the probabilities ${w_i}^j$

For $j = 1, \ldots, m$
(m is the total number of weak classifiers)

Train the base model $C_j(x)$ on Z_j

Compute the **classification error** err_j on the complete dataset Z weighting each event by ${w_i}^j$

$$err_j = \frac{\sum_{i=1}^{n} {w_i}^j I\left(C_j(x_i) \neq y_i\right)}{\sum_{i=1}^{n} {w_i}^j}$$

Update and then rescale the vector of weights

$${w_i}^{j+1} = {w_i}^j e^{-y_i \alpha_j C_j(x_i)}$$

where $\alpha_j = log\left(\frac{1-err_j}{err_j}\right)$

Best classifier in the boosting algorithm

The **final *strong classifier*** $C(x)$ is computed as the weighted average of the m *weak classifiers*, where weights are defined by the α_j:

$$C(x) = sign\left[\sum_{j=1}^{m} \alpha_j C_j(x)\right]$$

- Note that α_j is inversely proportional to the misclassification error: $\alpha_j = log\left(\frac{1-err_j}{err_j}\right)$, hence we're giving **more weight to those base learners that best classify the data**

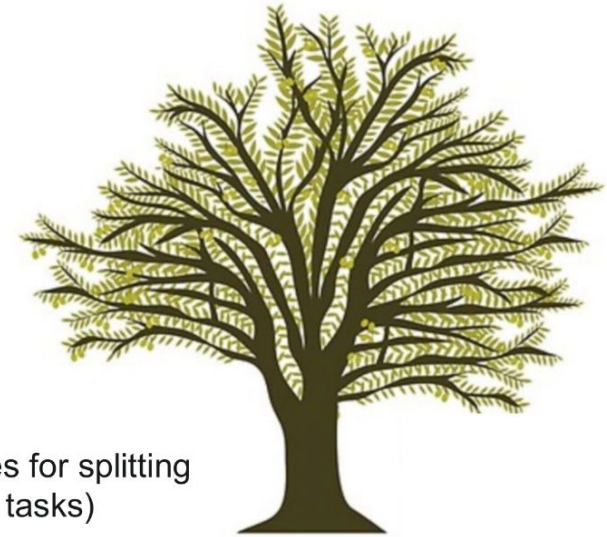
*As the base learner is trained in an **adaptive way** to focus of the hardest-to-classify data instances, boosting has been proved to **reduce both the variance and the bias** of the base classifier*

Random Forest

- **Random forest** is a substantail modification of *bagging* that builds a large collection of ***de-correlated trees*** and then averages them (it is therefore specifically developed for trees as base learners)

The idea is to **improve the variance reduction** of *bagging* **by reducing the correlation of the trees**

To do so, in the tree-growing process ***features are randomly selected***



- Before each split, $m \leq p$ of the input features are selected at random as candidates for splitting (a common choice is $m = \sqrt{p}$ for classification tasks and $m = \lfloor p/3 \rfloor$ for regression tasks)

What kind of problems we solve with ensemble models?

What kind of problems we solve with ensemble models?

Averaging weak learners outputs do not change the expected value (bias of the model) but **reduces its variance**

(just like averaging $i.i.d.$ random variables preserve expected value but reduces variance)

A variance reduction **decreases the prediction error** of the final model (bias-variance tradeoff)

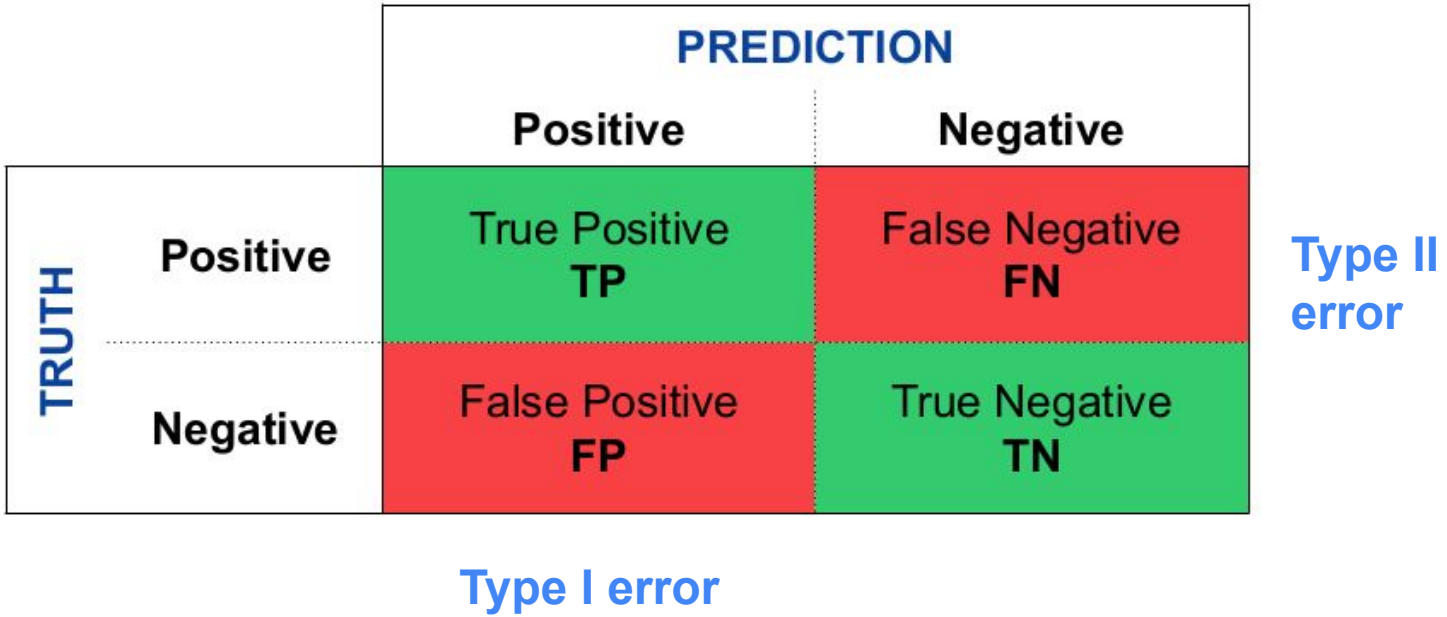
$$Err(x) = Bias^2 + Variance + Irreducible\ Error$$

On which models bagging will perform especially well?

On low bias, high variance models (e.g. trees)

Model evaluation

Confusion matrix



Evaluation metrics

F1 score

$$2\frac{precision \cdot recall}{precision + recall} = \frac{2 \cdot TP}{2 \cdot TP + FP + FN}$$

FPR (False Positive Rate)

- *What is the proportion of misclassified instances out of all the true negatives?*
- $\frac{FP}{FP+TN}$ (should be as low as possible)

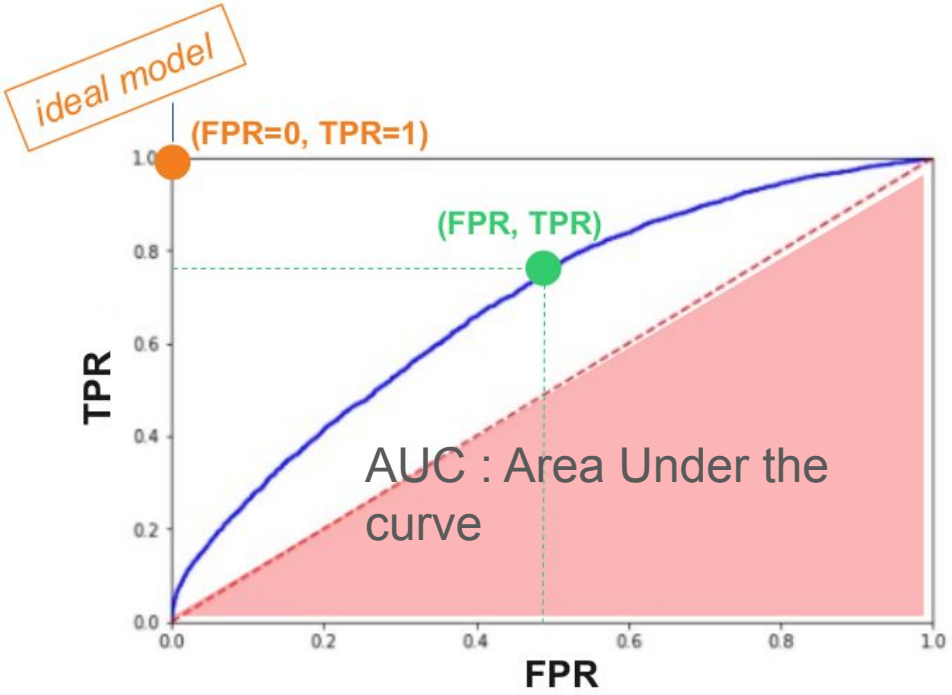
RECALL or TPR (True Positive Rate)

- *What proportion of actual positives is identified correctly?*
- $\frac{TP}{TP+FN}$ (should be high as possible)

PRECISION

- *What proportion of positive predictions is actually correct?*
- $\frac{TP}{TP+FP}$ (should be high as possible)

ROC curve

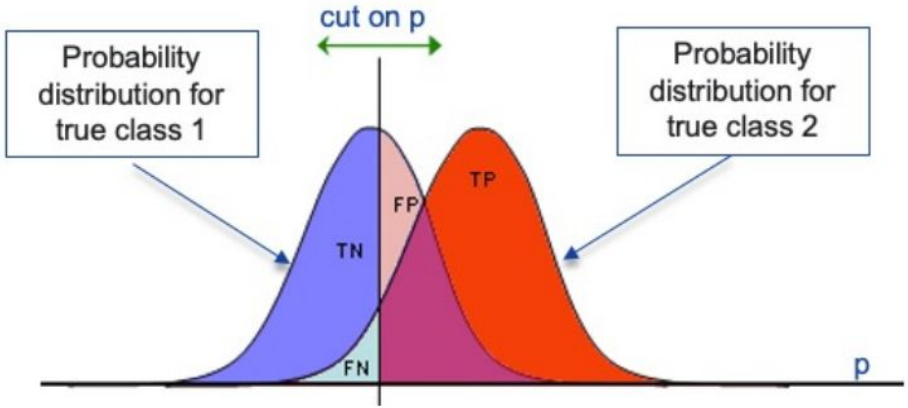


The AUC can be interpreted as the probability that the model assigns a higher score to a random positive example than to a random negative example

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

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

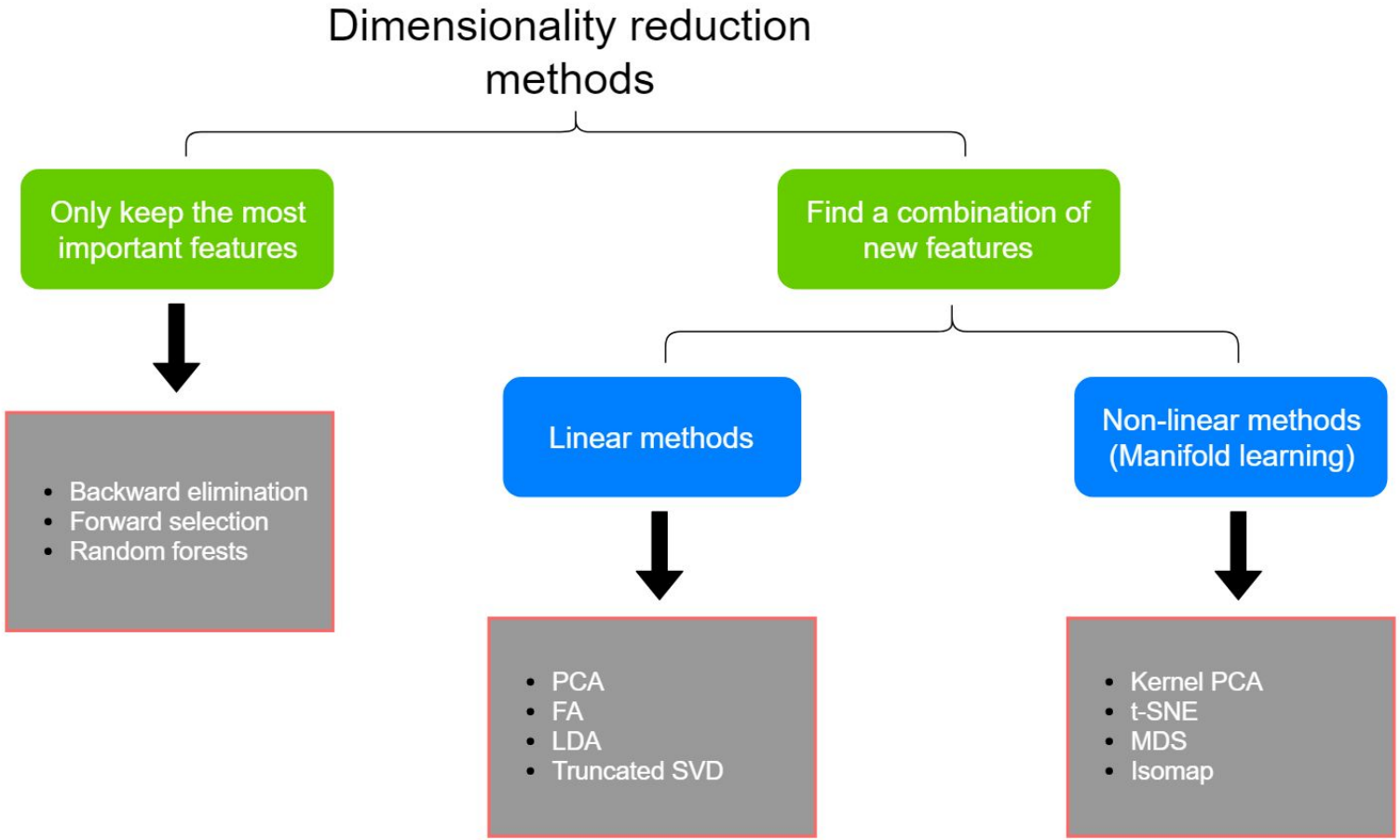
What's the effect of classification threshold on the prediction?



- Lowering the classification threshold classifies more items as positives
  → Both FP and TP increase
- Increasing the classification threshold classifies more items as negatives
  → Both TN and FN increase

Unsupervised learning algorithms

- Manifold Learning (Dim. Red. & ID est.):
  - PCA
  - K-PCA
  - ISOMAP
  - t-SNE
  - Autoencoders
- Density Estimation:
  - Histograms
  - Kernel Density Estimation
  - k-Nearest Neighbor
  - Generative Adversarial NN
- Clustering:
  - k-means/c-means, kernel k-means, spectral clustering…
  - Hierarchical clustering
  - Density Based clustering
  - Self Organizig Maps


Dimensionality reduction methods
Only keep the most important features
Find a combination of new features
Backward elimination
Forward selection
Random forests
Linear methods
Non-linear methods (Manifold learning)
PCA
FA
LDA
Truncated SVD
Kernel PCA
t-SNE
MDS
Isomap

Useful tools for dim. reduction/ feature analysis:

- Variance $\mathrm{Var}(X) = \mathrm{E}\left[(X - \mathrm{E}[X])^2\right]$
- Correlation $P_{xy} = \frac{\sigma_{xy}}{\sigma_x \sigma_y} = \frac{\sum (x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\left(\sum (x_i - \bar{x})^2\right)\left(\sum (y_i - \bar{y})^2\right)}}$
- Covariance Matrix
- Mutual information $I(i) = \int_{x_i} \int_{y} p(x_i, y) \log \frac{p(x_i, y)}{p(x_i)p(y)}$

Covariance matrix

Covariance matrix for 3D data

$$C = \begin{bmatrix} cov(x,x) & cov(x,y) & cov(x,z) \\ cov(y,x) & cov(y,y) & cov(y,z) \\ cov(z,x) & cov(z,y) & cov(z,z) \end{bmatrix}$$

Variances

- Diagonal contains the variances of x, y and z

• cov(x,y) = cov(y,x) hence matrix is symmetrical about the diagonal

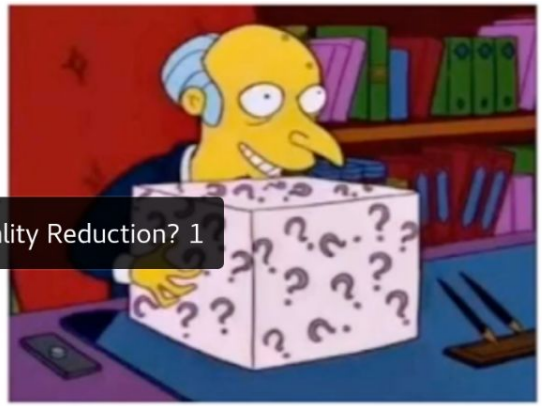
• N-dimensional data will result in NxN covariance matrix

Dimensionality reduction

It’s motivated by the assumption that all the important information on the data are in a N-dim space, with N smaller than the dimensionality of input data

Finding the intrinsic dimension of the input data is the Holy Grail for most ML researchers

$$A \in \mathbb{R}^{n \times m}$$



$$\longrightarrow \quad A' \in \mathbb{R}^{n \times k}$$

$$(k < m)$$

Motivations for dimensionality reduction

- Data Visualization
- Data Compression
- Noise Reduction
- Data Classification
- Trend Analysis
- Factor Analysis

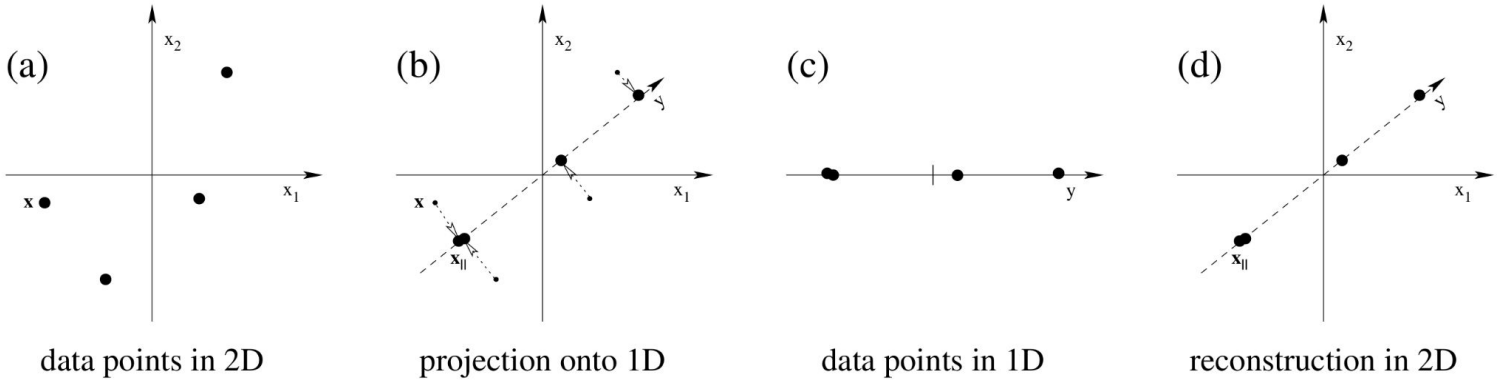
Diving into PCA (Principal component analysis)

How could we find the smallest subspace of the -D space that

keeps the most information about the original data?

Simpler solution: **Principal Component Analysis**

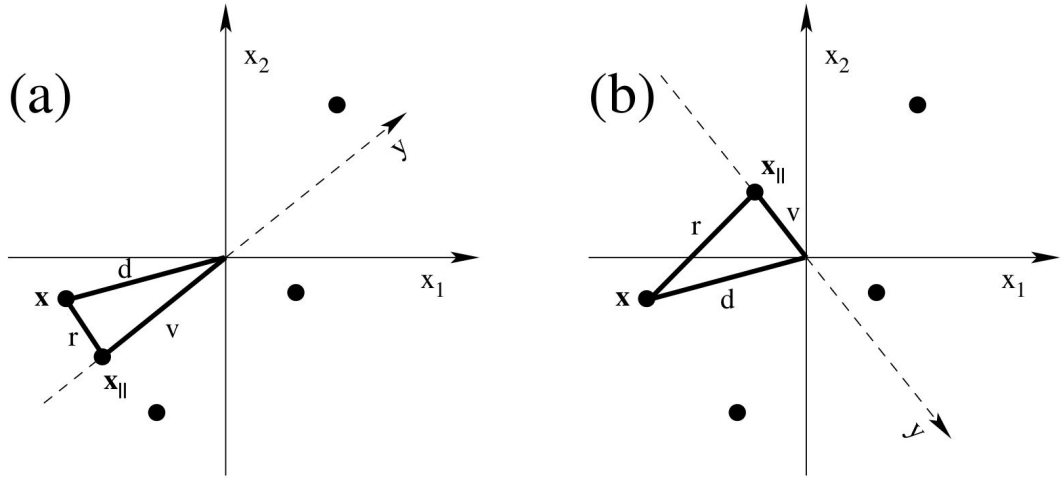
PCA for linear dim. reduction from 2D to 1D



With PCA we project the input data onto a lower dim. space such that we have the minimal reconstruction error

$$E := \left\langle \|\mathbf{x}^{\mu} - \mathbf{x}_{||}^{\mu}\|^2 \right\rangle_{\mu}$$

Maximal variance = minimum R_E



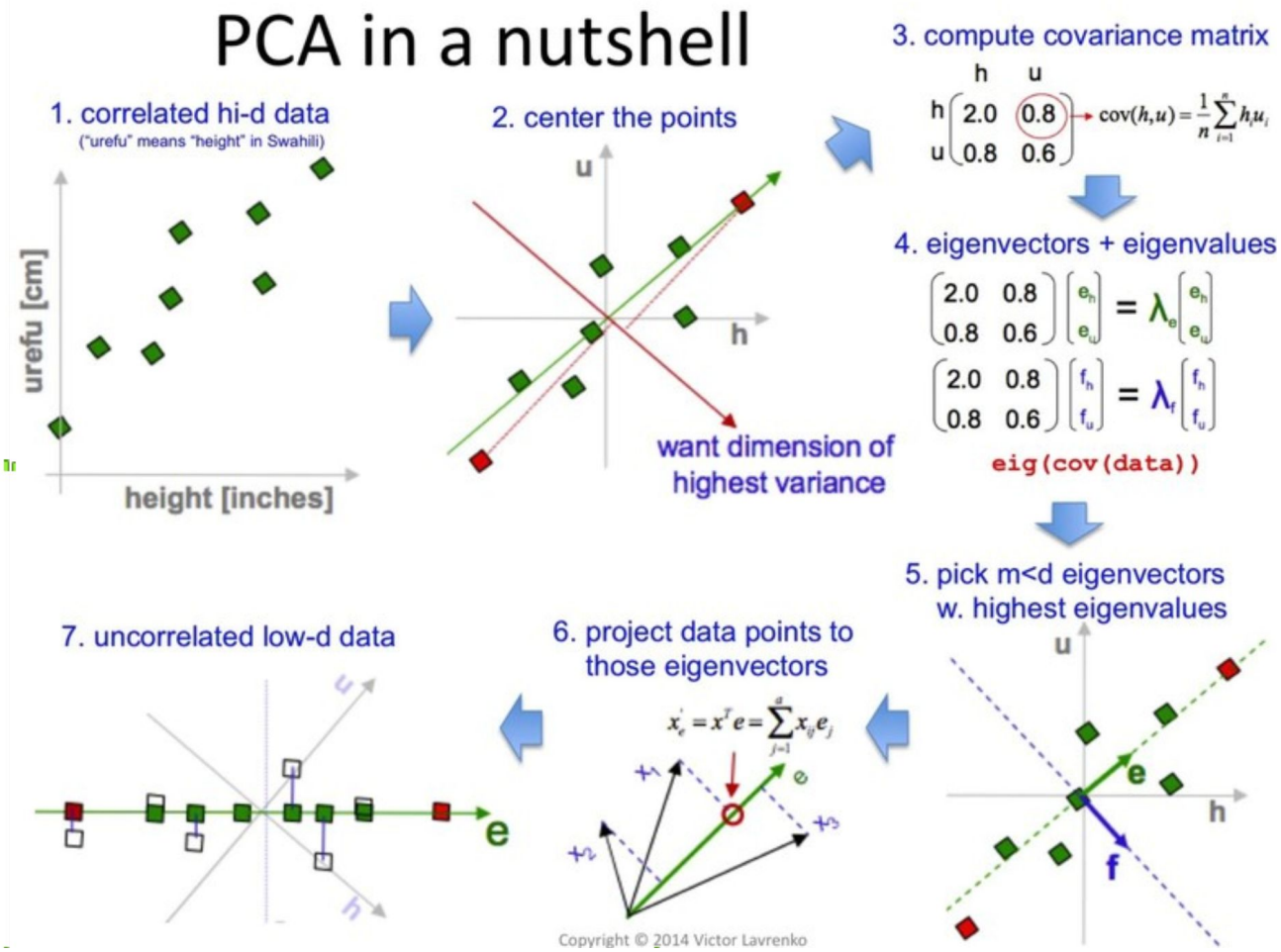
Looking at the 2D case we can see how the sum of the variance and the reconstruction error is constantly equal to d (Pitagora) and this means that we can maximize the variance to reduce the reconstruction error

PCA steps

Given m observations of n-dim column vectors , to perform PCA we follow the 6 steps below:

1. Compute the mean vector $\vec{\mu} = \frac{1}{m}\sum_{j=1}^{m}\vec{x}^{(j)}$
2. Compute the covariance matrix
3. Compute the sorted eigenvalue/eigenvector pairs (Diagonalize/SVD)
4. Choose the number of dimensions (k) in which project the data
5. Project the input data onto the selected k eigenvectors

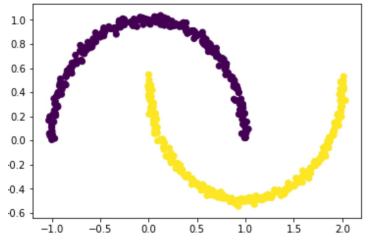
PCA in a nutshell



Problem with PCA

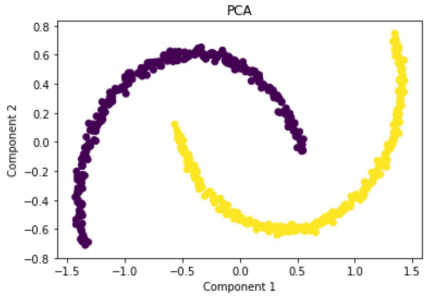
If we consider non-linear problems PCA fails in reducing the input data dimensionality

```
from sklearn.decomposition import PCA
pca = PCA(n_components=2)
X_pca = pca.fit_transform(X)

plt.title("PCA")
plt.scatter(X_pca[:, 0], X_pca[:, 1], c=y)
plt.xlabel("Component 1")
plt.ylabel("Component 2")
```



PCA codes applied to the input data

```
from sklearn.decomposition import PCA
pca = PCA(n_components=2)
X_pca = pca.fit_transform(X)

plt.title("PCA")
plt.scatter(X_pca[:, 0], X_pca[:, 1], c=y)
plt.xlabel("Component 1")
plt.ylabel("Component 2")
plt.show()
```



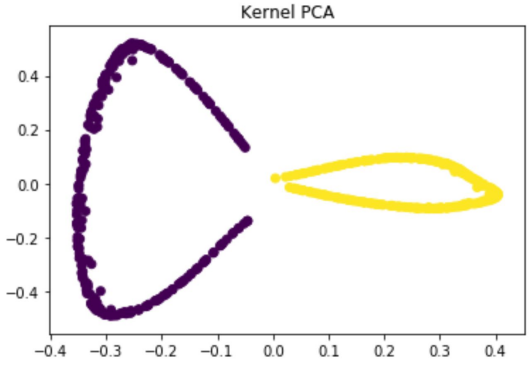
we need to use a kernel method -> kernel-PCA

Problem with PCA

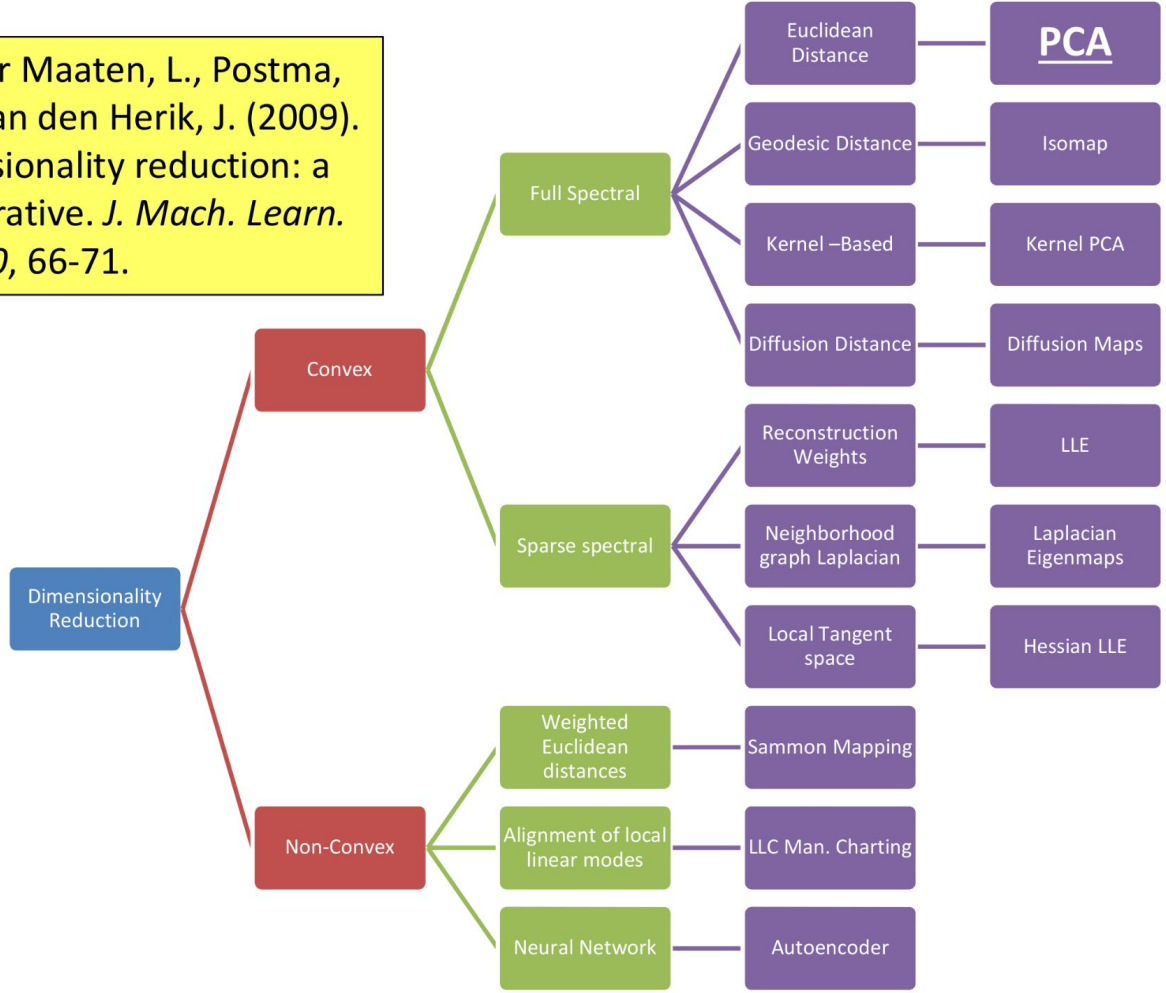
If we consider non-linear problems PCA fails in reducing the input data dimensionality

we need to use a kernel method -> kernel-PCA

```
from sklearn.decomposition import KernelPCA
kpca = KernelPCA(kernel='rbf', gamma=15)
X_kpca = kpca.fit_transform(X)

plt.title("Kernel PCA")
plt.scatter(X_kpca[:, 0], X_kpca[:, 1], c=y)
plt.show()
```



Van Der Maaten, L., Postma, E., & Van den Herik, J. (2009). Dimensionality reduction: a comparative. *J. Mach. Learn. Res.*, *10*, 66-71.



kernel PCA

Kernel PCA is an extension of PCA that allows for the separability of nonlinear data by making use of kernels. The basic idea behind it is to project the linearly inseparable data onto a higher dimensional space where it becomes linearly separable.

Kernel PCA can be summarized as a 4 step process [1]:

1. Construct the kernel matrix (K) from the training dataset
2. Estimate the Gram matrix
3. Estimate the eigenvalues of the Gram matrix

Advantages of kPCA

1. Non-linearity: Kernel PCA can capture non-linear patterns in the data that are not possible with traditional linear PCA.
2. Robustness: Kernel PCA can be more robust to outliers and noise in the data, as it considers the global structure of the data, rather than just local distances between data points.
3. Versatility: Different types of kernel functions can be used in kernel PCA to suit different types of data and different objectives.
4. Kernel PCA can handle nonlinear relationships between the input features, allowing for more accurate dimensionality reduction and feature extraction compared to traditional linear PCA.
5. It can preserve the most important information in high-dimensional datasets while reducing the dimensionality of the data, making it easier to visualize and analyze.
6. Kernel PCA can be used for a variety of tasks, including data visualization, clustering, and classification.

CLUSTERING

Clustering tries to separate data “naturally”, in such a way that ***similar*** elements lay in the same cluster while ***dissimilar*** elements belong to a different one

Similarity

pairwise function of the features. Its definition depends on the nature of input data. It can be seen as a distance (but it does not always corresponds to metric distances!)

Quantitative Features: *Metric Distances*

- Minkowski distance:

$$d_{ij} = \left(\sum_{l=1}^{d} \left|x_{il} - x_{jl}\right|^{p}\right)^{1/p}$$

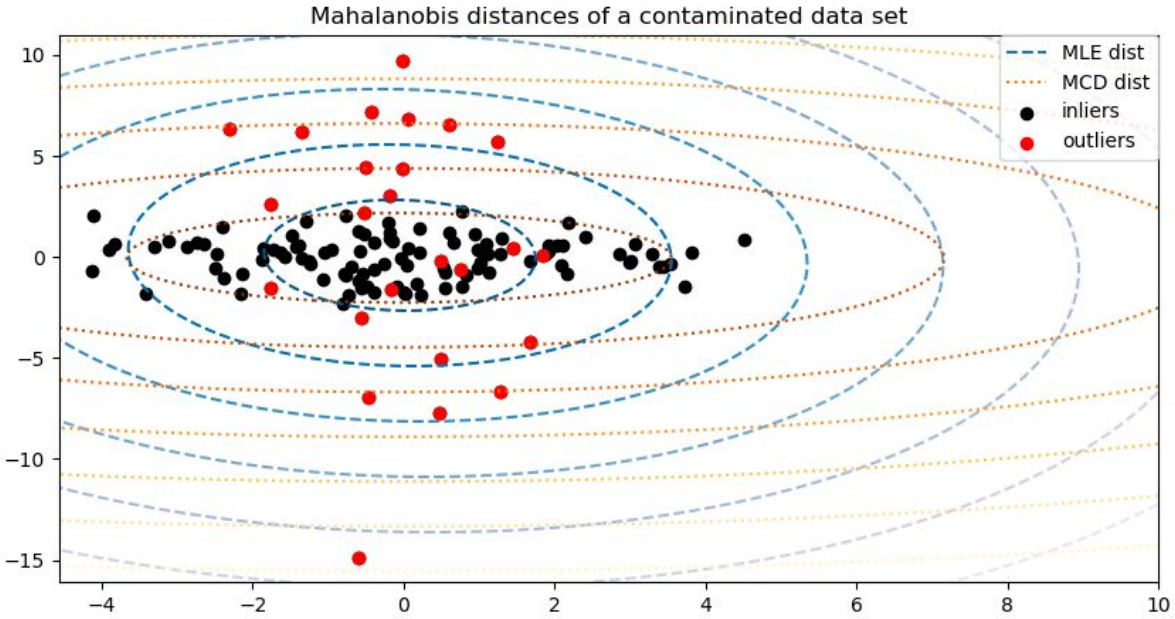
  - Special cases are:
    - Euclidean (p=2)
    - City-block (p=1)
    - Sup (p→∞) . Eqv to $d_{ij} = \max_{l} \left|x_{il} - x_{jl}\right|$

- Mahalanobis distance

$$d_{ij} = \left(x_i - x_j\right)^{T} \mathbb{C}^{-1} \left(x_i - x_j\right)$$

Invariant with respect to any non-singular linear transformation of the coordinates. $\mathbb{C}$ is the covariance matrix.

Outlier detections with Mahalanobis distance



Quantitative Features: *Not metric distances*

- Pearson correlation:

$$d_{ij} = \frac{1-r_{ij}}{2}; r_{ij} = \frac{\sum_{k=1}^{m}(x_{k,i}-\bar{x_i})(x_{k,j}-\overline{x_j})}{\sqrt{\sum_{k=1}^{m}(x_{k,i}-\bar{x_i})^2 \sum_{k=1}^{m}(x_{k,j}-\overline{x_j})^2}}$$

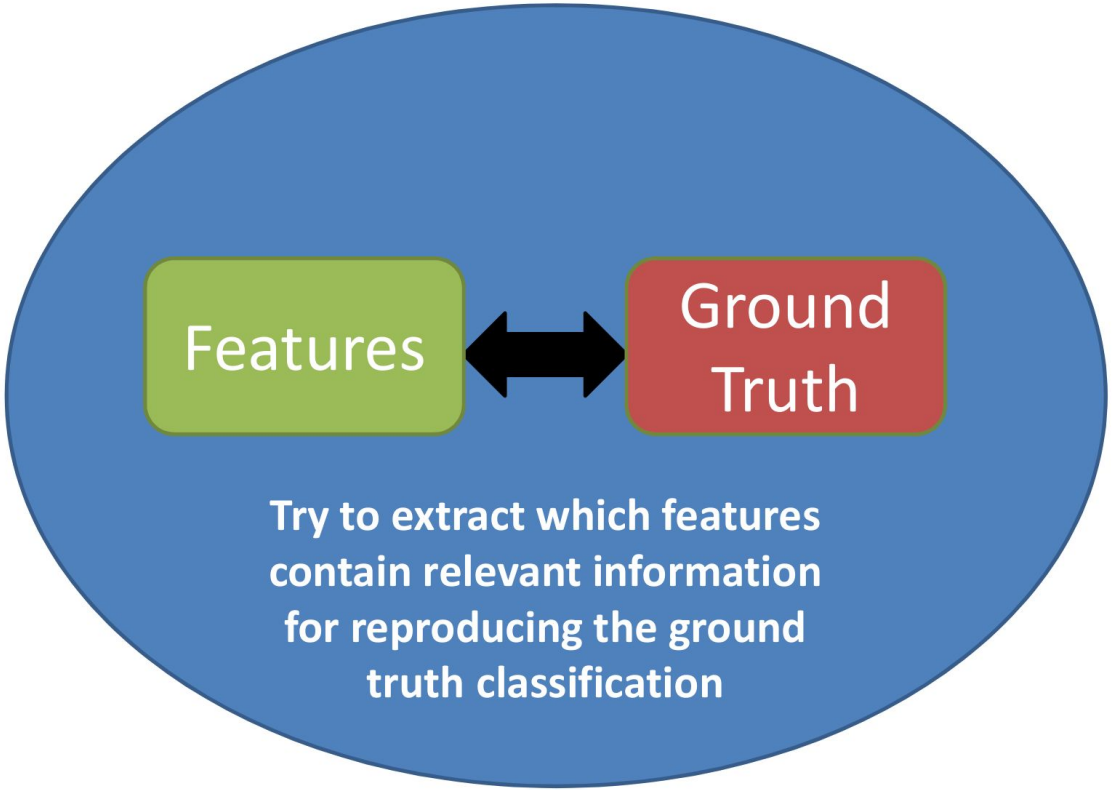
- Point Symmetry distance:

$$d_{ij} = \min_{k \neq i} \frac{\|(x_i - x_j) + (x_k - x_j)\|}{\|(x_i - x_j)\| + \|(x_k - x_j)\|}$$

- Cosine similarity:

$$S_{ij} = \frac{{x_i}^T \cdot x_j}{\|x_j\| \|x_i\|}$$

Feature selection



Techniques for Feature Selection (1)
Variable Ranking

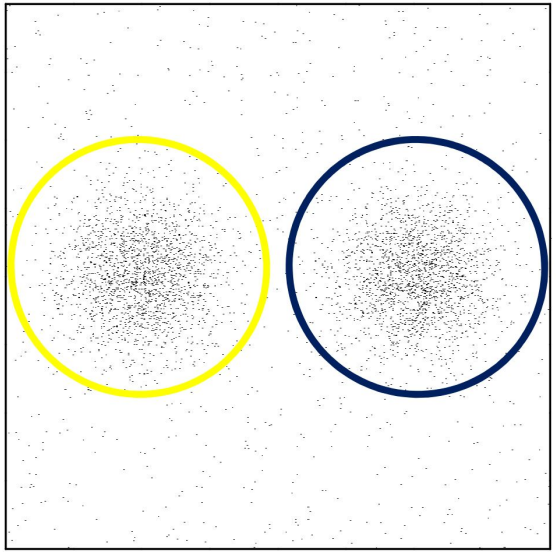
- Which variables explain better the labels?
- Quantify it and sort by the score
  - Linear correlation coefficient (R).
  - Single variable classifier. Jaccard index, F-score, etc.

Techniques for Feature Selection (1)
Information Theoretic Ranking

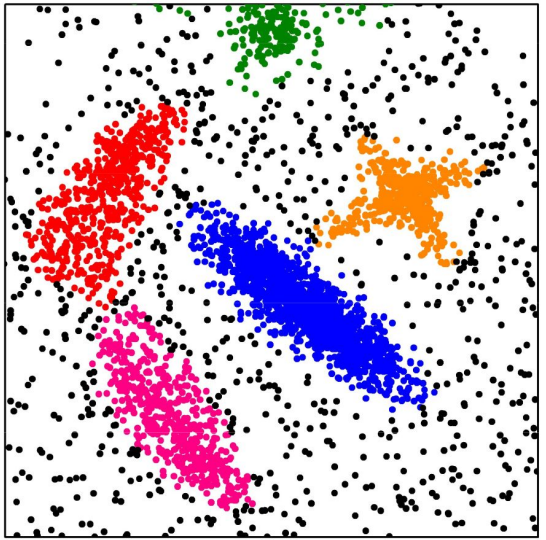
$$I(i) = \int_{x_i} \int_{y} p(x_i, y) \log \frac{p(x_i, y)}{p(x_i)p(y)}$$

- A kind of single variable classifier.
- In non-continuous variables, the integrals become sums
- Extensible to continuous variables by non parametric density estimation
- Using a Gaussian distribution for estimating the density will lead to a similar criteria to the correlation coefficient.
- Is a formalization of the intuition that the higher the joint distribution, the higher the mutual information, i.e. the higher should it be in the rank.

What is a cluster?

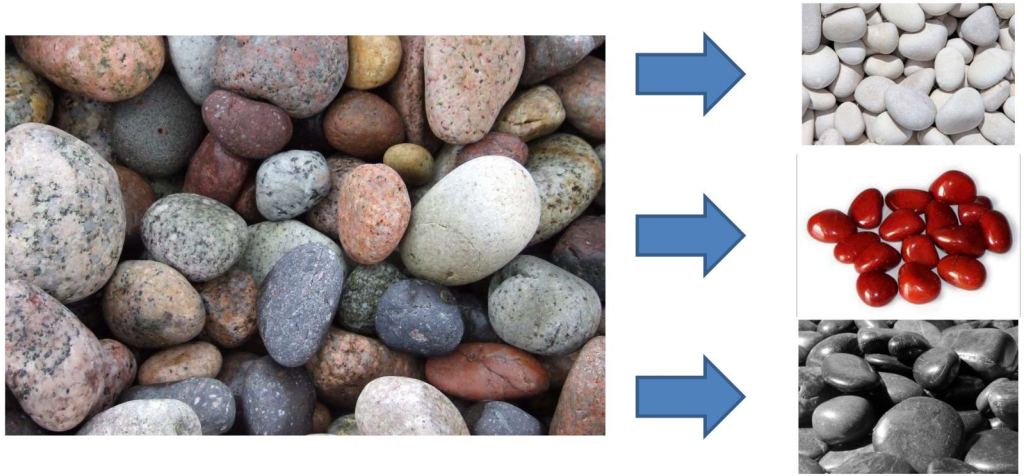


Other cluster examples…

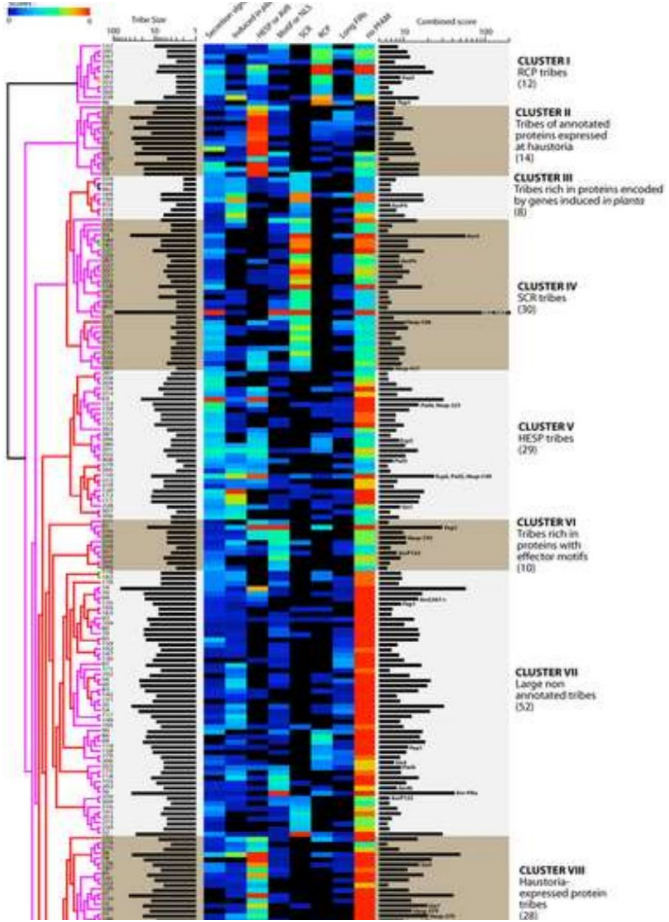


Clustering stone collections

Cluster by light wavelength



Are they nice or ugly?
Color classification?



Clustering proteins

Hierarchical clustering of the secretome reveals clusters of secreted protein families as high priority effector candidates.

Saunders DGO, Win J, Cano LM, Szabo LJ, Kamoun S, et al. (2012) Using Hierarchical Clustering of Secreted Protein Families to Classify and Rank Candidate Effectors of Rust Fungi. PLoS ONE 7(1): e29847. doi:10.1371/journal.pone.0029847 http://journals.plos.org/plosone/article?id=info:doi/10.1371/journal.pone.0029847

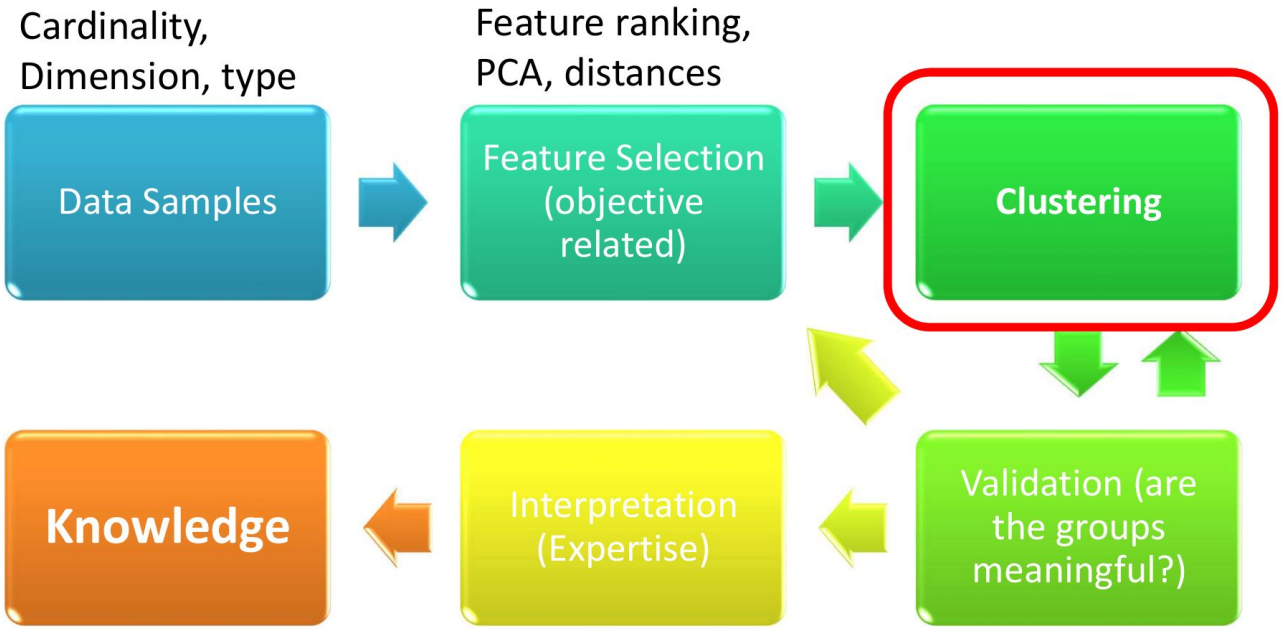
By clustering and classifying plant proteins detect protein groups (families) that will probably act against mold

Clustering definition

> *"Understanding our world requires conceptualizing the similarities and differences between the entities that compose it" (Tyron and Bailey, 1970).*

- Clustering ≠ Classification
  - Clustering generate groups
  - Classification generate groups <u>& labels</u>

Clustering procedure



Categories of clustering algorithms

- Flat clustering performs a hard partition of the input data
- Fuzzy clustering is a flat clustering algorithm with soft element assignation
- Hierarchical clustering generates a tree instead of a single partition. Can be agglomerative (joining elements) or divisive (dividing the dataset)

Considerations on clustering

The result of the clustering process depends on:

- your cluster definition
- the metric adopted to measure distances
- the feature you choose to consider to cluster data

Most common flat clustering algorithm…

K-means: A flat clustering algorithm

MacQueen, J. (1967, June). Some methods for classification and analysis of multivariate observations. In Proceedings of the fifth Berkeley symposium on mathematical statistics and probability (Vol. 1, No. 14, pp. 281-297).

Basic concept of K-MEANS

Attempts to minimize the intracluster distance

while maximizing the intercluster distance.

- It is based on the concept of cluster centroid,

i.e. the average position of the cluster elements.

- Still widely used.
- It can be easily parallelized and linearized.
- User must provide the number of desired clusters (k)

K-means algorithm

Input: cluster size *k*, instances $\{\overrightarrow{x_i}\}_{i=1}^{n}$
Output: cluster membership assignments $\{z_i\}_{i=1}^{n}$

1. Initialize *k* cluster centroids $\{\overrightarrow{c_l}\}_{l=1}^{k}$ (randomly from the data set).
2. Repeat until no instance changes its cluster membership:
   - Decide the cluster membership of instances by assigning them to the nearest cluster centroid

     $$z_i = argmin_l(d(\overrightarrow{c_l}, \overrightarrow{x_i}))$$ *Minimize intra distance*
   - Update the *k* cluster centroids based on the assigned cluster membership

     $$\overrightarrow{c_l} = \frac{\sum_{i=1}^{n} \delta_{z_{il}} \overrightarrow{x_i}}{\sum_{i=1}^{n} \delta_{z_{il}}}$$ *Maximize inter distance*

Loss function/Objective function

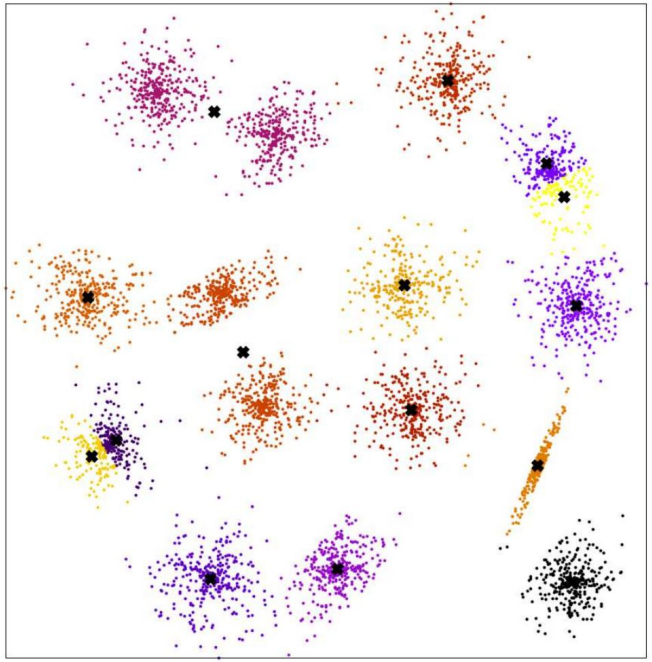
- Loss function:

$$O(z) = \sum_{l=1}^{k} \sum_{i=1}^{n} \delta(z_{i,} l) \quad \|\overrightarrow{x_i} - \overrightarrow{c_l}\|^2$$

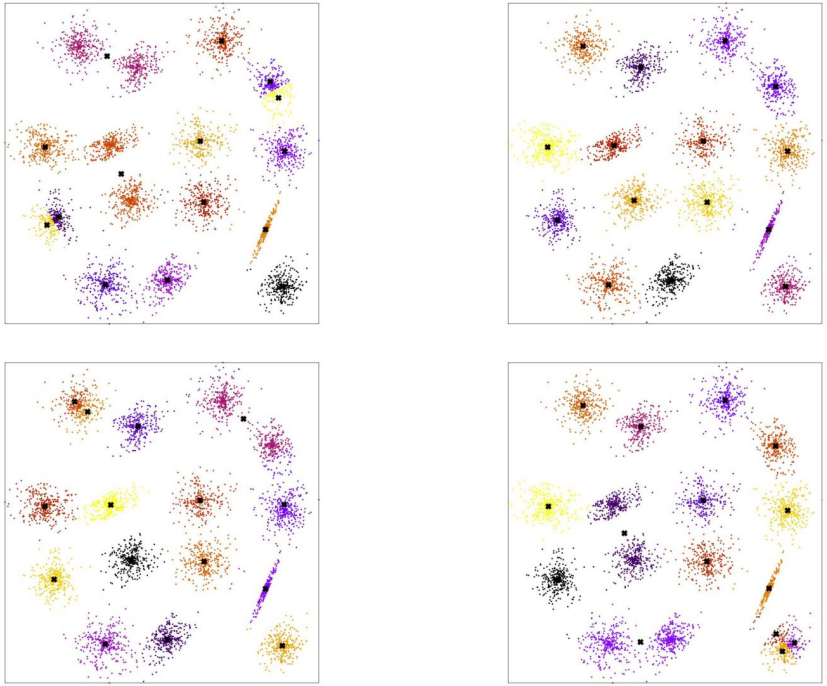
- z is an array with n components that reflects the assignation in clusters.
- $\overrightarrow{c_l}$ is the vector with the coordinates of the l-th cluster centroid. $\overrightarrow{c_l} = \frac{\sum_{i=1}^{n} \delta_{z_{il}} \overrightarrow{x_i}}{\sum_{i=1}^{n} \delta_{z_{il}}}$

K-means steps

- Randomly pick *k* centers.
- Assign each point to its nearest center.
- Recompute centers.
- Iterate…



Different initialization impact



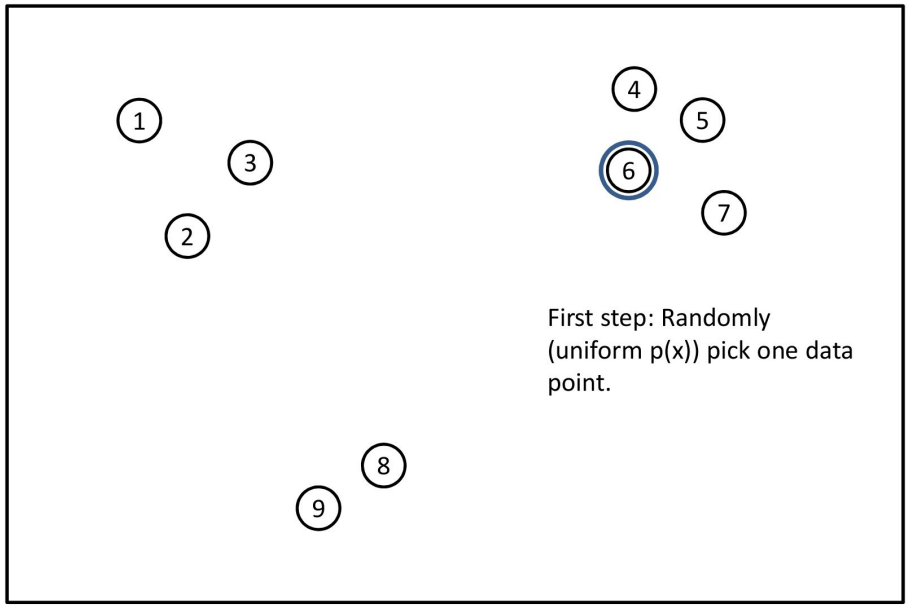
K-means weaknesses

Initialization over sensitive (local optimization) →k- means++
• Which k-employing →Scree test
• Sensitive to outliers →k-medoids
• employs Euclidean distance →k-medoids
• Only for spherical clusters →kernel k-means
(advanced)

Better initialization: *k*-means++

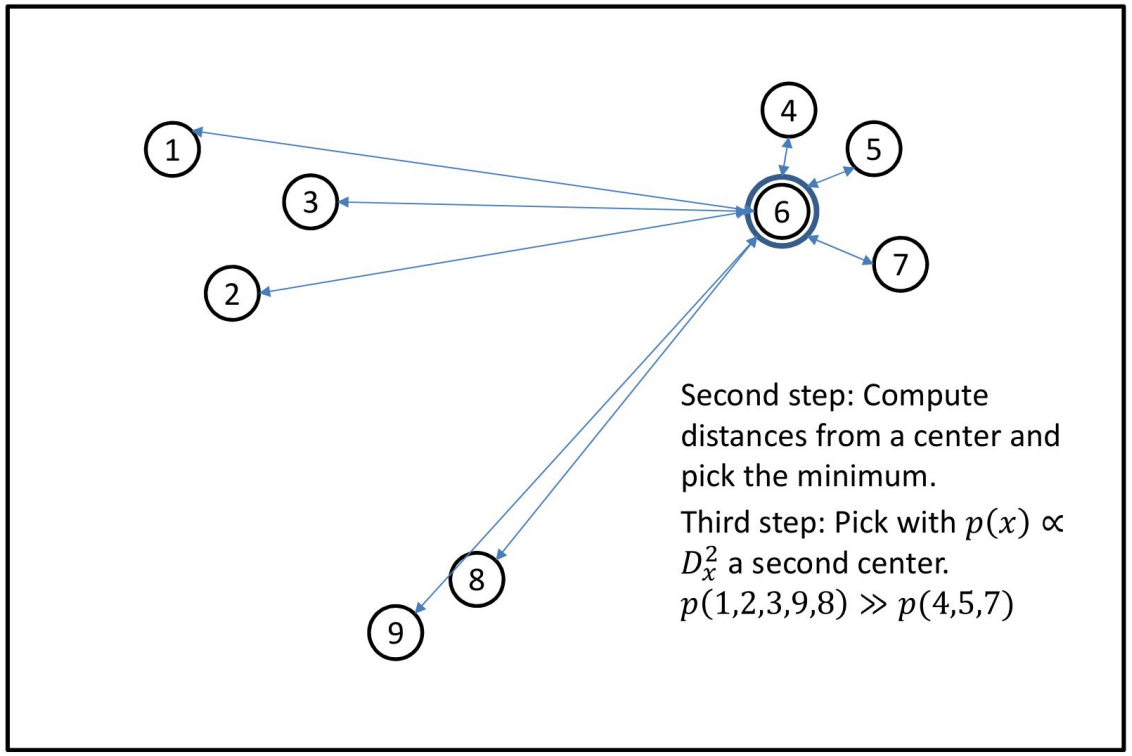
1. Choose the first cluster center at random
2. Repeat until all *k* centers have been found
   - For each instance compute $\mathrm{D_x} = \min_{\mathrm{k}}[d(x, c_k)]$
   - Choose a new cluster center with probability $p(x) \propto D_x^2$ ← *new center should be far away from existing centers*
3. Run *k*-means with selected centers as initialization

Steps of K-means ++

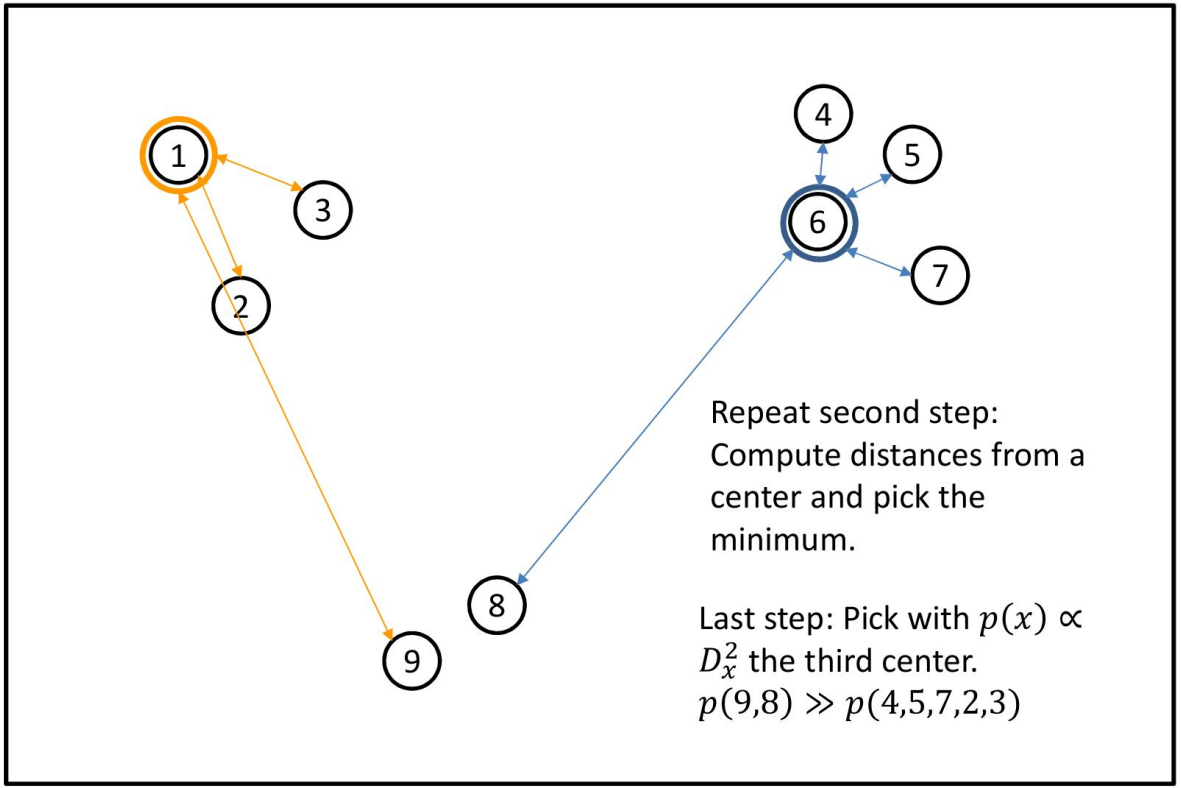


Step 2

K-means ++



Step 2


1
2
3
4
5
6
7
8
9
Repeat second step:
Compute distances from a center and pick the minimum.
Last step: Pick with $p(x) \propto D_x^2$ the third center.
$p(9,8) \gg p(4,5,7,2,3)$

References

S. Dridi, **Unsupervised Learning - A Systematic Literature Review**