Learning from examples

subsymbolic approach to knowledge

Machine Learning & Agent

- ***An agent is learning if it improves its performance after making observations about the world.***

- When the agent is a computer, we call it machine learning:

1. a computer observes some data,
2. builds a model based on the data,
3. and uses the model as both a hypothesis about the world and a piece of software that can solve problems.

Machine Learning & Agent

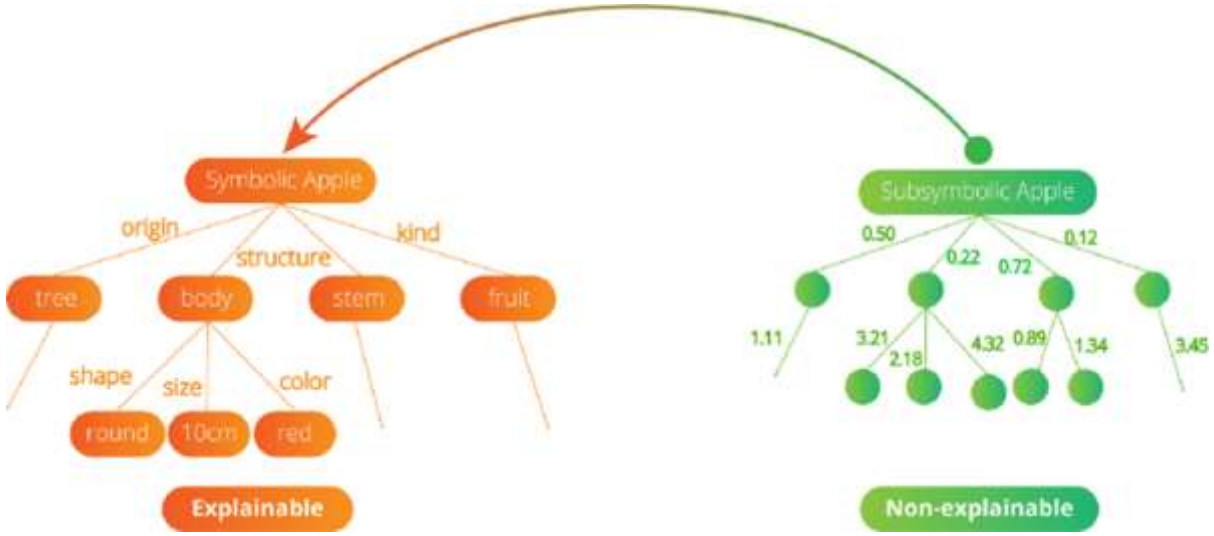
- Any component of an agent program can be improved by machine learning.
  - Consider a self-driving car agent that learns by observing a human driver.
  1. Every time the driver brakes, the agent might learn a condition–action rule for when to brake .
  2. By seeing many camera images that it is told contain buses, it can learn to recognize them.
  3. By trying actions and observing the results—for example, braking hard on a wet road—it can learn the effects of its actions
  - …

- The improvements, and the techniques used to make them, depend on these factors:
  - Which component is to be improved.
  - What prior knowledge the agent has, which influences the model it builds.
  - What data and feedback on that data is available

Machine Learning

- AI models in this paradigm are implicitly represented.
- Implicit representation comes from learning from experience without symbolic representation of rules and properties.

- Instead of clearly defined human-readable relationships, you implement less explainable mathematical equations to solve problems.
- *The ability to mine a good model with limited experience is assumed to give a successful model.*



Example: automated construction of a simple expert system (*decision tree*)

- A system based on production RULES (*IF x THEN y*) which, starting from some FACTS (*data*) tries to demonstrate a hypothesis or reach a conclusion.
- For example, deciding whether or not to play golf.
- VARIABLES:
  - Weather conditions
  - Humidity
  - Wind

DECISIONS (Hp)

- play
- not play

Example: a simple expert system

RULES

1. IF Meteo=‘Sunny’ AND Humidity =‘Low’ THEN Play
2. IF Meteo=‘Sunny’ AND Humidity =‘High’ THEN Don’t play
3. IF Meteo= ‘Rainy’ AND Wind = TRUE THEN Don’t play
4. IF Meteo= ‘Rainy ’ AND Wind = FALSE THEN Play
5. IF Meteo= ‘Overcast’ THEN Play
6. ELSE Play

FACTS

Meteo = ‘Sunny’

Wind = TRUE

Example: a simple expert system

It activates the rules that include the facts of which they are aware

1. IF Meteo=‘Sunny’ AND Humidity =‘Low’ THEN Play
2. IF Meteo=‘Sunny’ AND Humidity =‘High’ THEN Don’t play
3. IF Meteo= ‘Rainy’ AND Wind = TRUE THEN Don’t play
4. IF Meteo= ‘Rainy ’ AND Wind = FALSE THEN Play
5. IF Meteo= ‘Overcast’ THEN Play
6. ELSE Play

To know whether to play or not I have to acquire a new fact …

The "inference engine" then asks:

Humidity ?

If the answer is 'High' the system suggests “not to play”…

Expert system ⇔ Machine Learning

- *I have therefore drawn a conclusion from the available facts.*

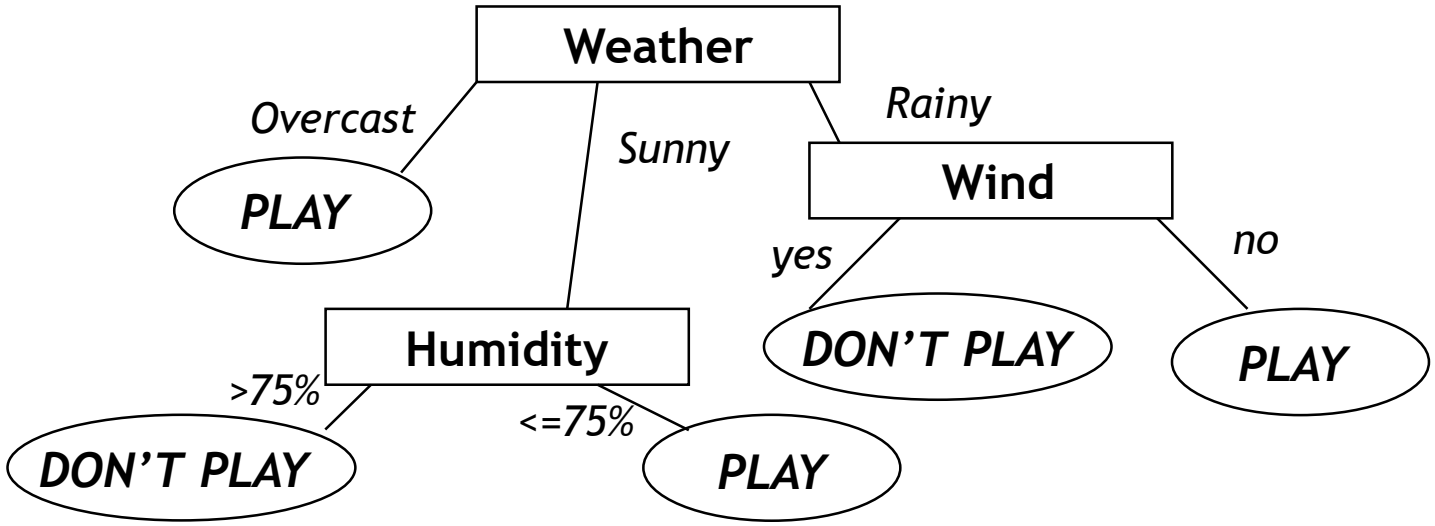
- Can an algorithm build these rules autonomously?
- Yes, for example by using a decision tree

Decision tree

The decision-making process is represented with an inverted logical tree, starting from a root node , where each node is a conditional function.

- each *internal node* represents a test on a property or attribute,
- an *arc* to a child node represents a possible value for that property or attribute,
- a *leaf* the expected final value starting from the values of the other properties, which in the tree is represented by the path from the root node to the leaf node.

Decision tree



Decision tree

It is possible to automatically build a decision tree starting from a series of "examples".

Weather	Wind	Humidity	Decision
Sunny	NO	30-60	**PLAY**
Sunny	YES	90	**DON'T PLAY**
Overcast	NO	50	**PLAY**
Rainy	YES	50	**DON'T PLAY**
Rainy	YES	90	**DON'T PLAY**
Sunny	NO	80	**DON'T PLAY**
Overcast	YES	80	**PLAY**
Rainy	NO	70	**PLAY**

Trainig set:
series of previous situations in which an expert has made the decision to play or not

Inductive learning
(the type of reasoning in the subsymbolic paradigm)

It is used to learn concepts generally expressed with rules whose form is:

«*in situation X perform action Y*»

«*to the observation X it associates the observationY*»

«*to the situation X associates the consequence Y*»

Weather	Wind	Humidity	Decision
Sunny	NO	30-60	**PLAY**
Sunny	YES	90	**DON'T PLAY**
Overcast	NO	50	**PLAY**
Rainy	YES	50	**DON'T PLAY**
Rainy	YES	90	**DON'T PLAY**
Sunny	NO	80	**DON'T PLAY**
Overcast	YES	80	**PLAY**
Rainy	NO	70	**PLAY**

$situation_{1/}observation_1 \rightarrow action_1/observation_1/consequence_1$

...

$situation_{n/}observation_n \rightarrow action_n/observation_n/consequence_n$

$situation_{gen/}observation_{gen} \rightarrow action_{gen}/observation_{gen}/consequence_{gen}$

Learning by experience & Inductive reasoning

- We are never certain something will happen, but we usually know (or can estimate rather well) how likely it is to happen or, at least, what is most likely to happen, based on the ***experience*** we have acquired throughout our life.
- Experience in machine learning means: data in the form of **examples**
  - Explore data to find patterns to understand the hidden laws that regulates the domain

Learning by experience & Inductive reasoning

An inductive learning algorithm performs the following task:

«given a set of examples (samples) of f, define a function h (hypothesis) that approximates f»

- This situation is purely ideal.
- ***Problems***:
  - find the right level of approximationand
  - the observations may be contradictory

- We can speak of **generalization** of the information contained in the (relatively) few examples available in a model that can be reused in all other possible situations of that type.
- Ex. By observing a limited series of cars, I learn to recognize a car in different contexts, i.e. I learn the concept of a car

Inductive reasoning & machines

- "A computer program is said to learn from ***experience E*** with respect to some class of ***tasks T*** and ***performance measure P***, if its performance at tasks in T, as measured by P, improves with experience E". *(Mitchell 1997)*
- *Learning is our means of attaining the ability to perform automatically a task*
- **Task T** : A task that is difficult to be solved with fixed programs written and designed by human beings
- **Experience E**: Collected data that describes the input of our ML system and the main source of information to exploit in order to learn
- **Performance measure P**: How good is the model? Is it able to solve the problem for real? Obviously depending on the task we have to choose a different measure

What is an example in a machine learning task?

- An example is a collection of **features** that have been quantitatively or qualitative measured from some object or event that we want the machine learning system to process.
- Typically we represent an example as a vector $x \varepsilon R^n$ where each entry xi of the vector is a different feature:
  - Examples:
  - Features of an image are usually the values of the pixels in the image
  - Or if we want to predict the price of an house on the basis of some characteristics (n° rooms, garden, position, floor, etc..) that are the features of each house example from which we learn.

What is an Experience E in a machine learning task?

- Experience is drawn from a dataset: A collection of many examples
- One common way of describing a dataset is with a design matrix X:
- A matrix X containing a different example in each row and where each column is a different feature

Let's consider the IRIS dataset (1936): It is a collection of measurements of different parts of 150 iris plants.

$X =$

Sepal length	Sepal Width	Petal length	Petal width
5.1	3.5	1.4	0.2
4.6	3.1	1.5	0.2
7.0	3.2	4.7	1.4
6.9	3.2	5.7	2.3

$y =$

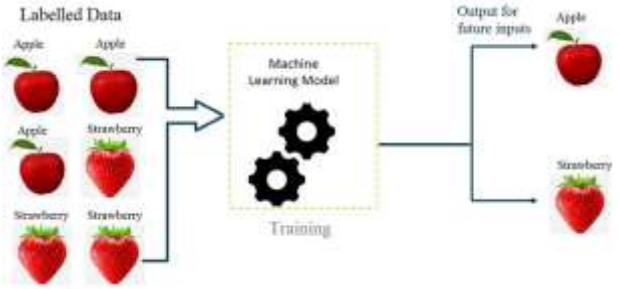
Class(label)
Setosa
Setosa
Versicolor
Virginica

Machine Learning types

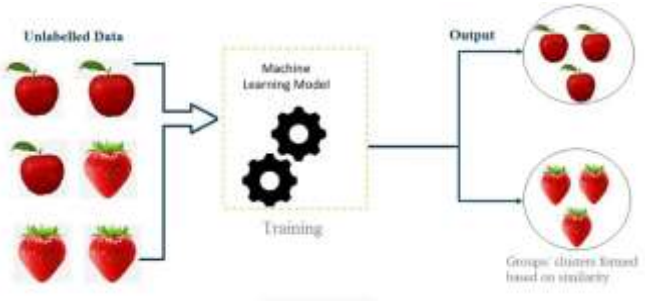
- There are four main types of learning:
- **Supervised Learning**
- **Unsupervised Learning**
- **Semi-supervised learning**
- **Reinforcement learning**

Machine Learning types

- **Supervised Learning**
- The agent observes input-output pairs and learns a function that maps from input to output. An output like this is called a label. A common task of this learning is **classification**



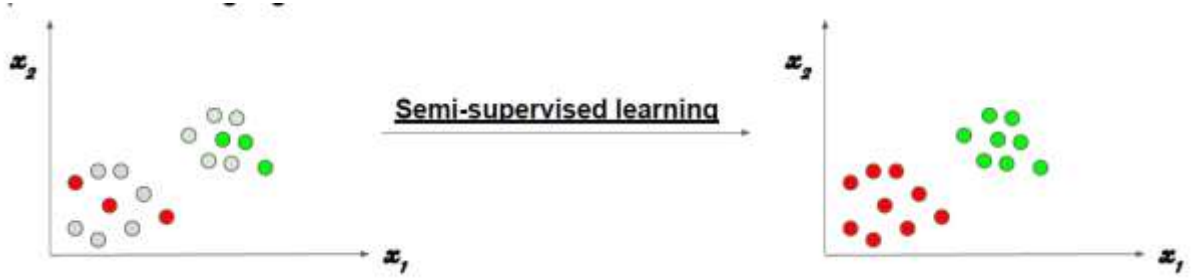
- **Unsupervised Learning**
- the agent learns patterns in the input without any explicit learning feedback. The most common unsupervised learning task is **clustering**: detecting potentially useful clusters of input examples



- **Semi-supervised learning**
- **Reinforcement learning**

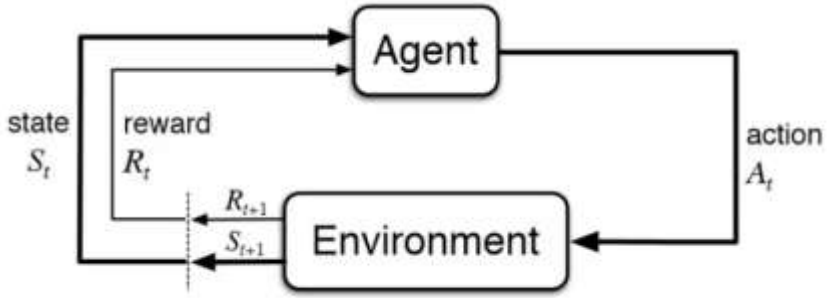
Machine Learning types

- **Supervised Learning**
- **Unsupervised Learning**
- **Semi-supervised learning**
- We have few labelled data so we increase our labelled data considering similarities and proximity (unsupervised) and then we exploit supervised learning algorithms
- **Reinforcement learning**



Machine Learning types

- **Supervised Learning**
- **Unsupervised Learning**
- **Semi-supervised learning**
- **Reinforcement learning**
- the agent learns from a series of reinforcements: **rewards and punishments.**
- For example, at the end of a chess game the agent is told that it has won (a reward) or lost (a punishment).
- It is up to the agent to decide which of the actions prior to the reinforcement were most responsible for it, and to alter its actions to aim towards more rewards in the future

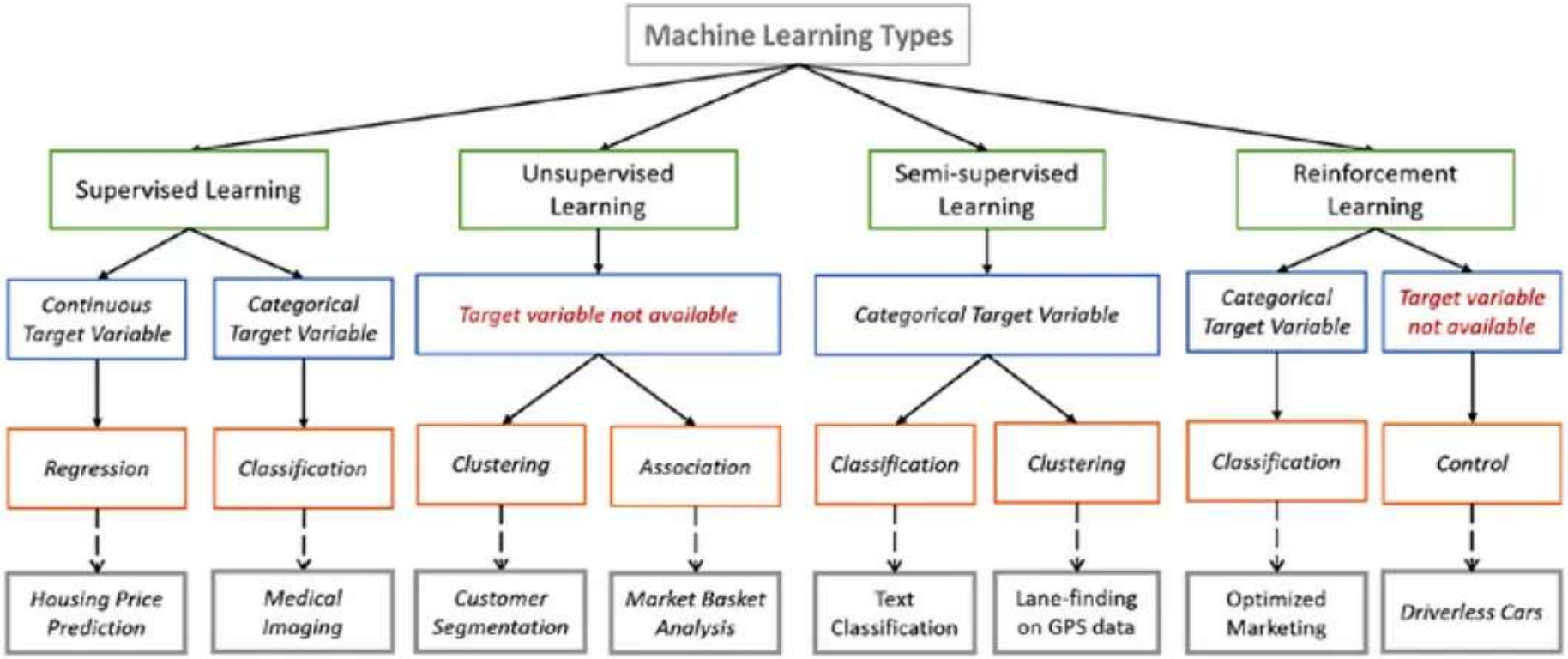


Supervised Learning vs Unsupervised Learning

- Supervised Learning
  - Predictive analytics models => ability to predict the future based on the past
  - Need for labeled data sometimes difficult to find

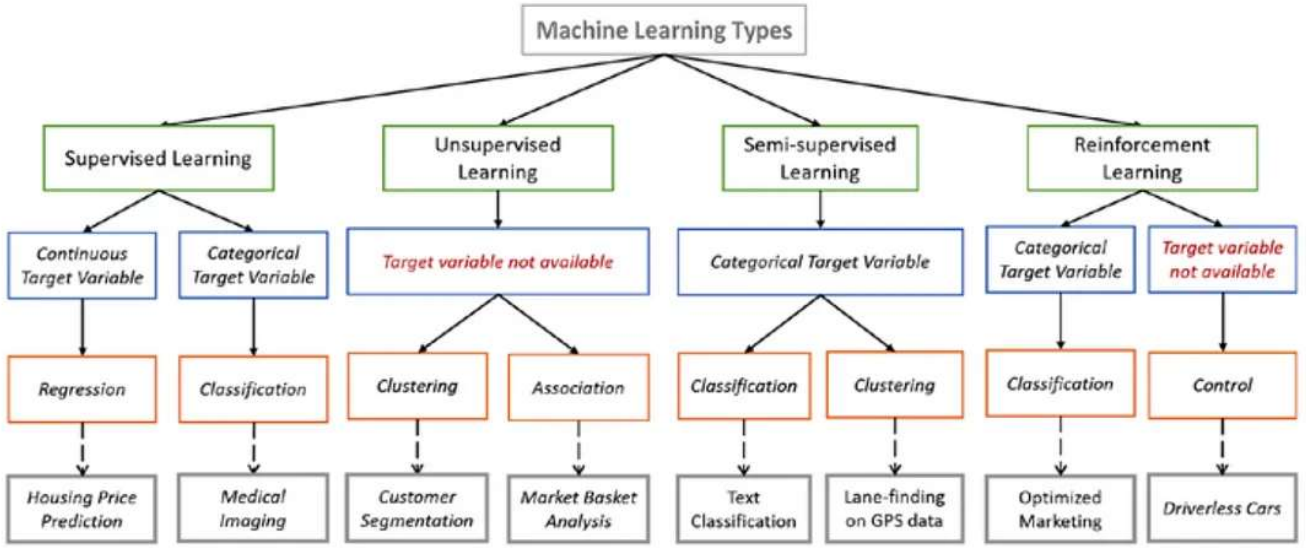
- Unsupervised Learning
  - they do not attempt to find a relationship between predictors and a specific response and therefore are not used to make predictions of any kind.
  - Instead, they are used to find previously unknown forms of organization and representation in data
  - They can reduce the size of data by condensing multiple variables together (dimensional reduction).
    - A typical example of this type is file compression.
    - Compression takes advantage of patterns in the data to represent that same data in a more compact format.
  - Find groups of observations that behave the same way and group them (clustering).

Machine Learning types



Machine Learning types

- **Output:**
  1. Categorical target variable
  2. Continuous target variable
  3. Target variable non available
     - unlabeled dataset



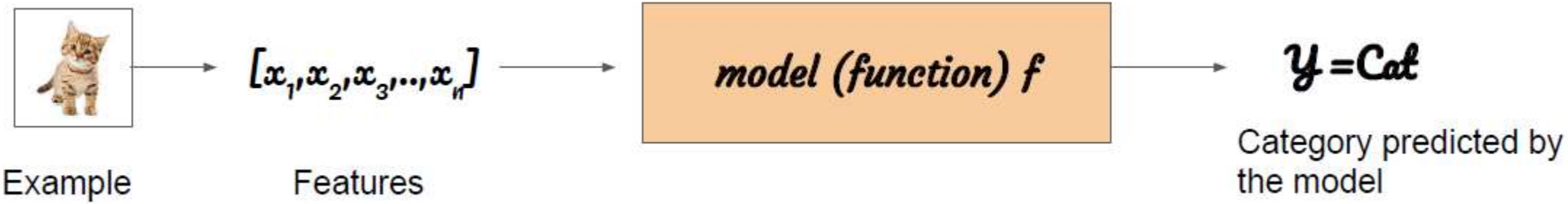
Supervised learning: Classification task

- The system is asked to specify which of k categories some input belongs to.

- To solve this task, the learning algorithm is usually asked to produce a function

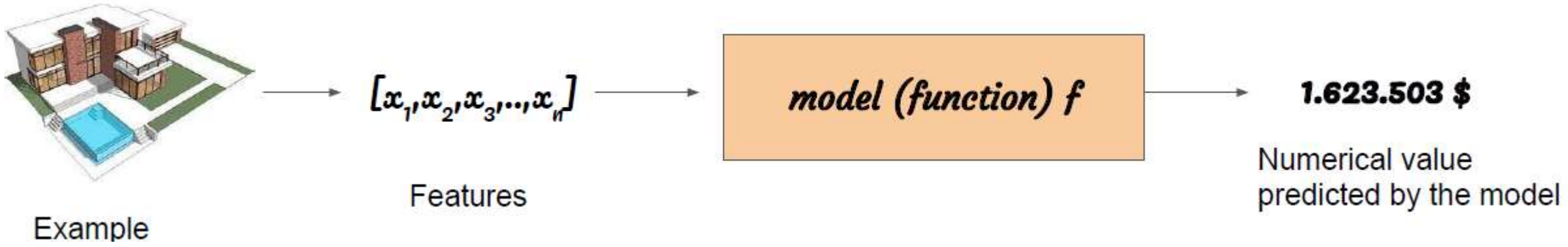
$$y = f(x): R^n \rightarrow \{1,...,k\}$$

- So the model takes an example **x** as input and after some processing **f(x)** it returns a value **y** that is one of the k categories the example x should belong to.



Supervised learning: Regression task

- The system is asked to predict a numerical value given some input.
  - For example: Given the house features we want to predict the price. It is a pure numerical and unbounded value
- To solve this task, the learning algorithm is usually asked to produce a function
- $y = f(x): R^n \rightarrow R$
- So the model takes an example **x** as input and after some processing **f(x)** it returns a value **y** that can be any real value!



Unupervised learning: Clustering task

- The system is asked to group a set of objects in such a way that objects in the same group (called a cluster) are more similar to each other than to those in other groups.
- It is useful to discover latent properties or irregularities among data
- Usually data are unlabelled: we don’t know anything about what is the correct group the example should belong to

Supervised learning

- More formally, the task of supervised learning is this:
  Given a training set of N example input–output pairs
  (x1,y1), (x2,y2), . . . (xN,yN) ,
  where each pair was generated by an unknown function y = f (x),
  discover a function *h* that approximates the true function f .

- The function ***h is called a hypothesis about the world***. It is drawn from a hypothesis space *H* of possible functions.

- With alternative vocabulary, we can say that ***h*** is a model of the data, drawn from a model class *H*, or we can say a function drawn from a function class

A model in supervised learning

- A model is mathematical tool that maps our examples (observations) **X** to the desired output **y** in the form

y ~ F(X)

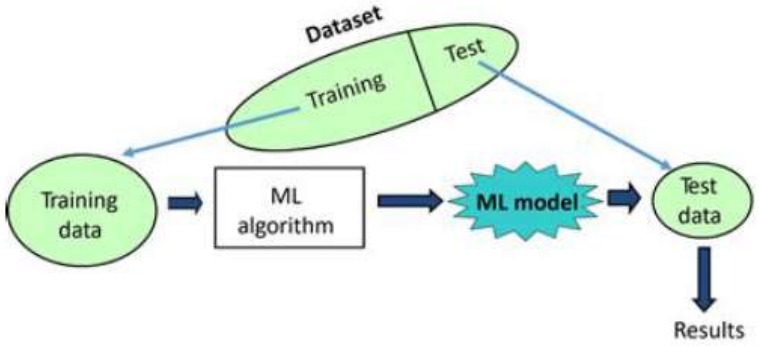
where F() is our model

- F() is a parametric function and we call its parameters **W**

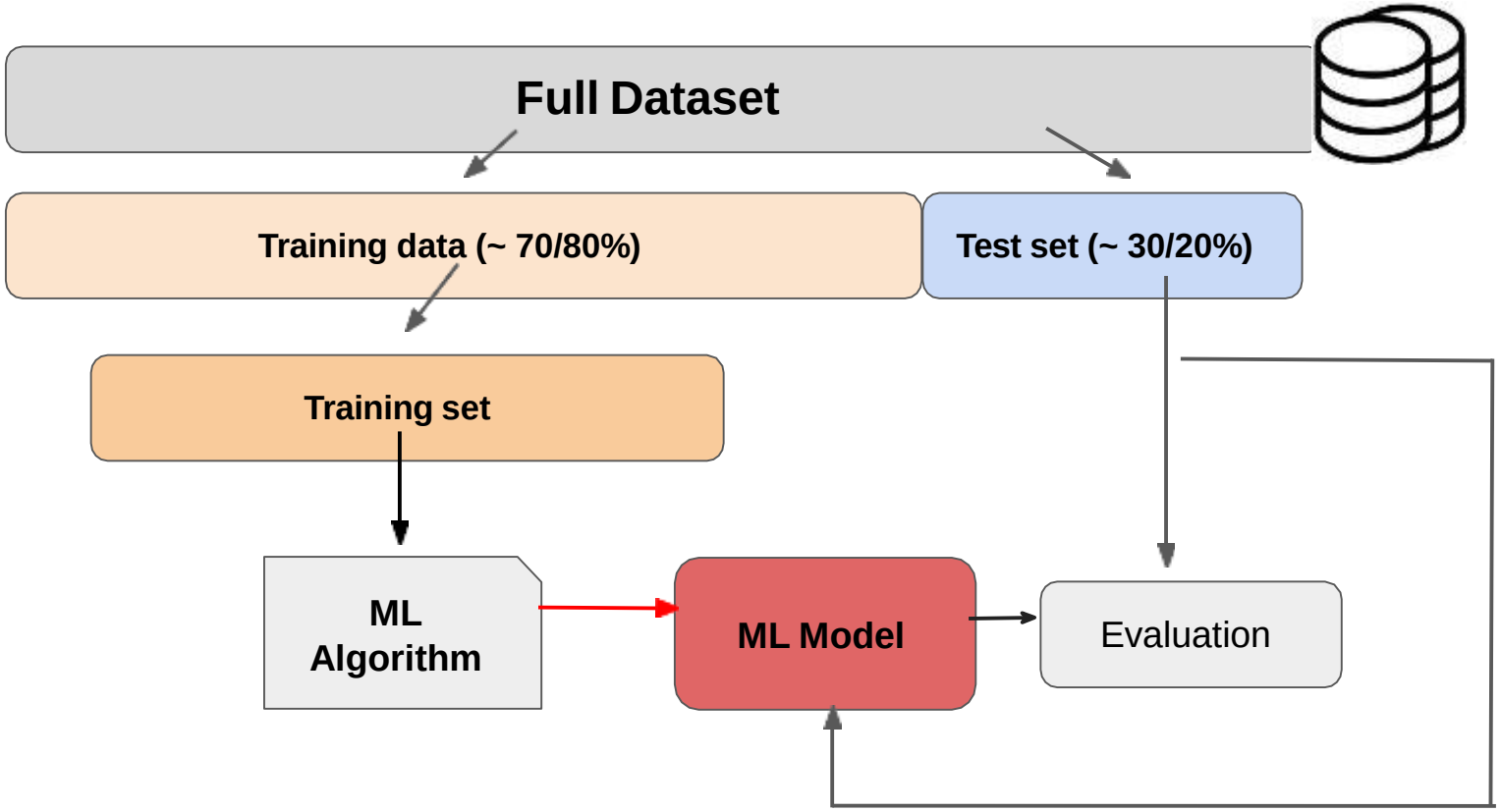
- So, a model is y=F(X ,W) and the goal of learning is to estimate these parameters ***W***

A model in supervised learning

- Once we trained a model we might want to measure its performance for example the accuracy in classification task (proportion of correct predicted examples) or the average error in a regression task
- We want to measure performance on some data that have not been seen during training. Why? ***Because our goal is to build a model that is able to really understand the task and that is able to generalize***
- For this reason we need a separate dataset called Test-set that we never used to train the model
- ***We say that h generalizes well if it accurately predicts the outputs of the test set.***



Training-set & Test-set



Supervised learning with an example of regression

House price prediction

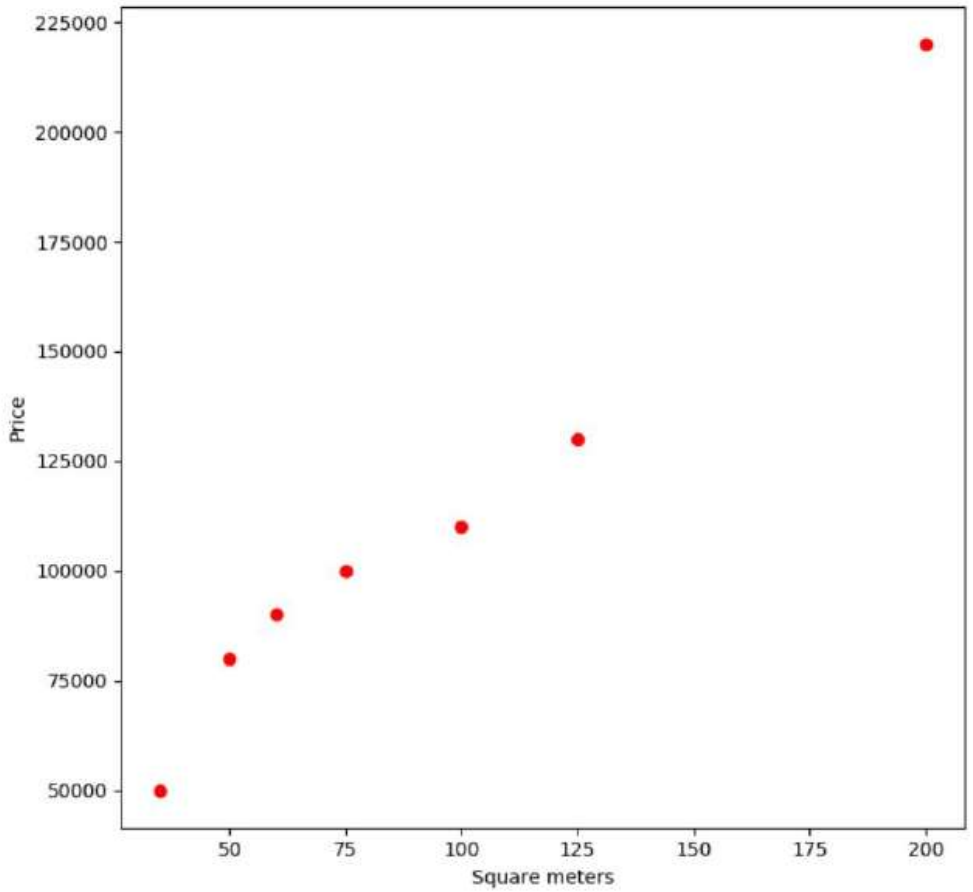
- We want to build an intelligent system that helps to predict prices of houses in a city
- We only have two information: square meters and the price
- Square meters is what we call *a feature*
- Price represents our target output

Mq	Price
50	80,000 $
75	100,000 $
125	130,000 $
35	50,000 $
200	220,000 $
60	90,000 $
100	110,000 $

Supervised learning with an example of regression

House price prediction

- X in this case is represented by square meters column
- Y in this case is the price column
- Our goal is to identify a model that on the basis of these examples can understand the underlying rule of this domain



Supervised learning with an example of regression

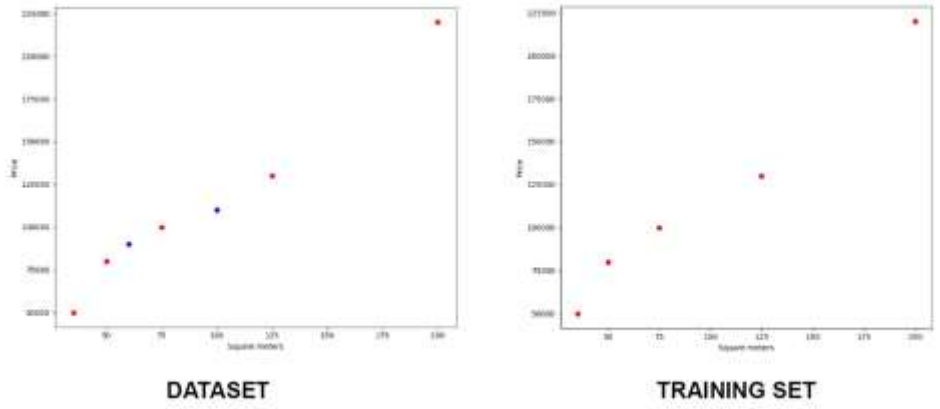
We need

- A task
- A dataset (*to split in training and test set* to evaluate if the model is able to generalize)
- A ML algorithm to build the model
- A metric for this evaluation

Mq	Price
50	80,000 $
75	100,000 $
125	130,000 $
35	50,000 $
200	220,000 $
60	90,000 $
100	110,000 $

Training set

Test set

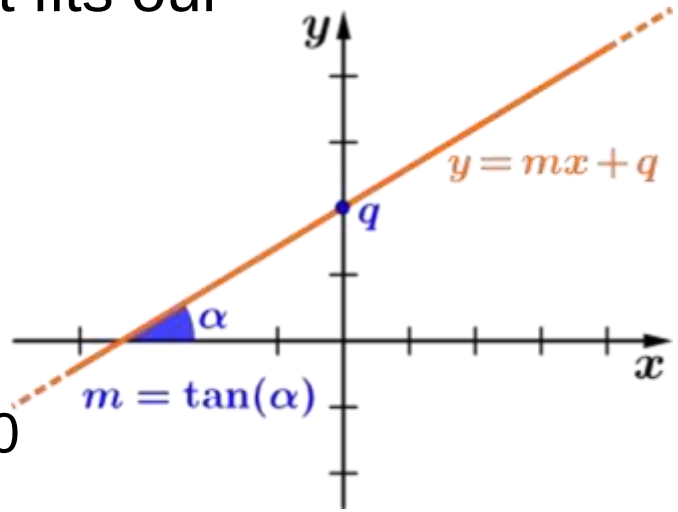


A suggested division is to use the 70 % of data in training and 30 % for the test

Supervised learning with an example of regression

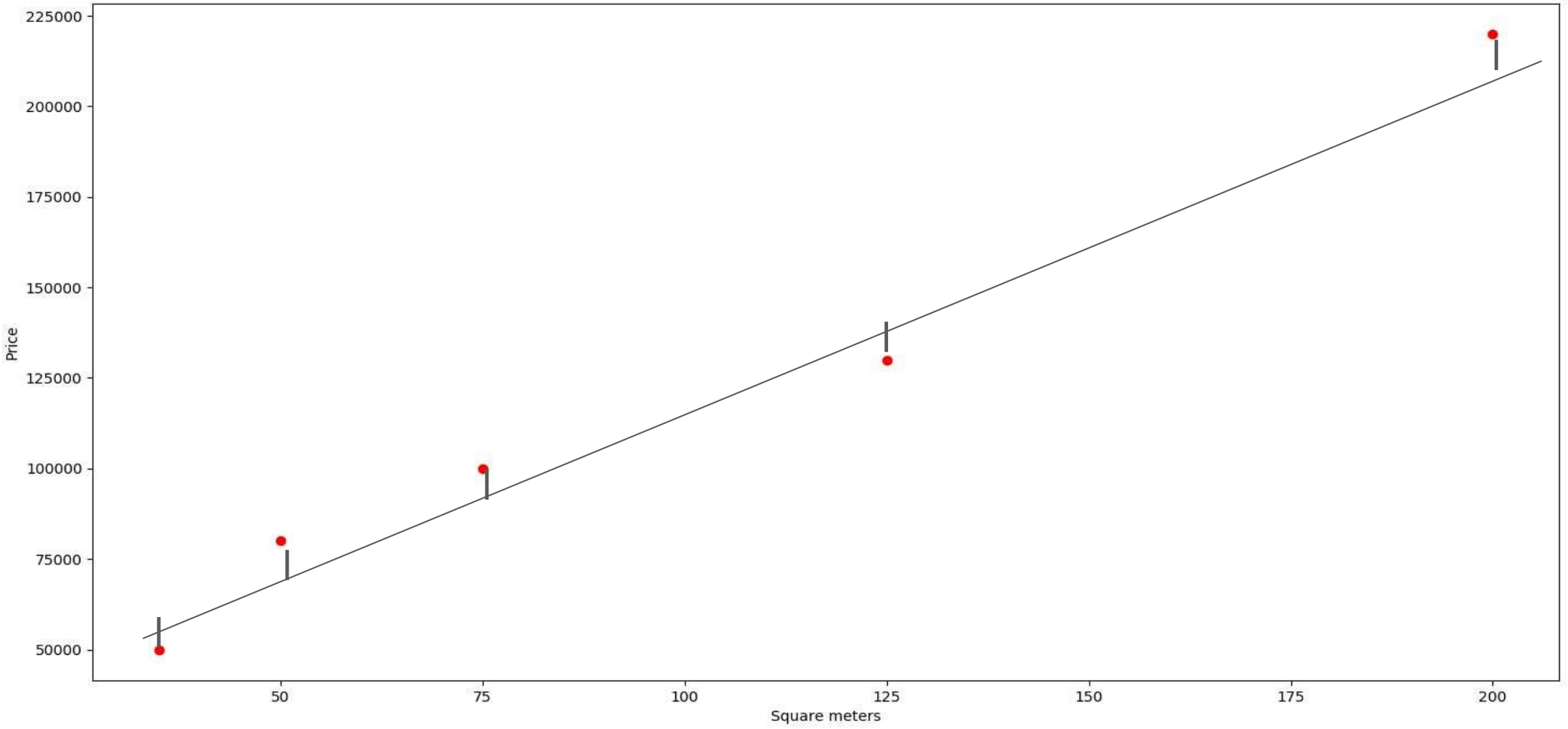
Linear model

- We want to find the most easy linear model that fits our data to get predictions on new data
- In our case let's consider the equation of a line
  - $y = mx + q$
    - Where m is the slope
    - q is the intercept
- If we write that equation as $y = w_1 X + w_0$
  - y is our target output
  - X is our feature
  - W are the two parameters that we have to learn



- If $X_0 = 1$ we can write the equation as $\mathbf{Y = W^T X}$

Supervised learning with an example of regression



Supervised learning with an example of regression

- We want w_0 and w_1 to have a line that has the minimum distance from our point
- Let's call the output of our linear model $y^* = F(x, W)$
- We want the best W that minimize the following cost function:

$$J(w) = \frac{1}{2m} \sum_{i=1}^{m} (y_i - y_i^*)^2$$

Loss function L

  - Where m is the number of the training examples

- To minimize this function we should compute the derivative (gradient) of L with respect to W and find where this derivative is equal to 0
  - $\nabla_w L = 0$

Supervised learning with an example of regression

- We have a $J(w_0, w_1)$ cost function and we want to find the parameters w_0, w_1 that better minimizes J
  - We start with a random choice of W
  - Keep changing W to reduce J(W) until we hopefully end up at a minimum
  - Weights are updated simultaneously

Repeat until convergence {

$$w_J := w_j - \alpha \frac{d}{dw_j} J(w_0, w_1) \quad \text{for j=0 and j=1}$$

Where α is the learning rate

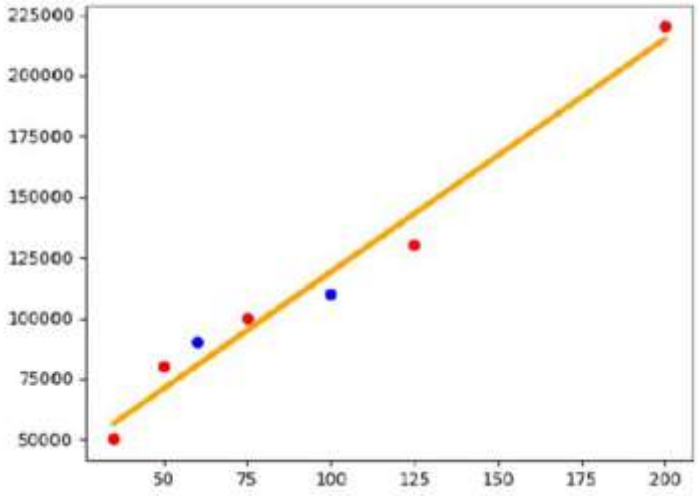
}

Supervised learning with an example of regression

Linear Regression

- $Y = W^T X$
  - Where X is the feature matrix (examples) and Y is the column vector with the target values

- It computes the gradient of the loss and estimates the most performing line



How good is our model ?

- **Root mean square error**

Good datasets for good models

- A model ‘learnt from data’ is generally as good as the data from which it has been derived

- A good dataset should be
  - Large
  - Correct (affected by noise as little as possible)
  - Consistent (examples must not be contradictory and a pattern should exist)
  - Well balanced (all classes adequately represented)

I.I.D Assumptions

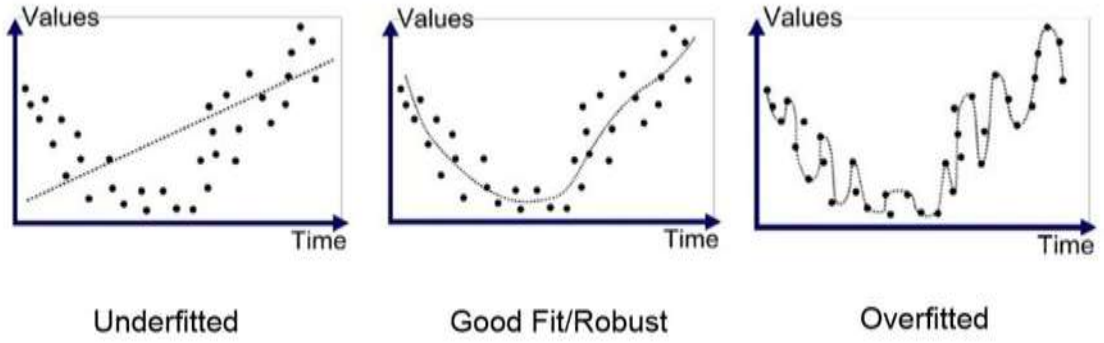
- The training and test data are generated by a probability distribution over datasets called the “data-generating process”

- We typically make a set of assumptionsknown collectivelyas the i.i.d assumptions:
  - The examples in the dataset are **<u>independent</u>** from each other and that the training set and test set are **<u>identically distributed</u>** (***I.I.D***)

  - The training and test set are drawn from the same probability distribution. Commonly known also as the **<u>Ideal Generator hypothesis</u>**: It exist a probability distribution of every phenomenon and our dataset is only a set of observations we collected about this distribution. We hope that our dataset is representative of the entire distribution!

Overfitting VS underfitting

- Underfitting occurs when the model is not able to obtain a sufficiently low error value on the training-set: probably few data or the model is not the one correct to fit data.
- Overfitting occurs when the gap between training error and test error is too large
- *We want to absolutely avoid overfitting* (the learning algorithm memorizes the training set instead of learning general rules, the model performs well on the training data, but it does not generalize well).



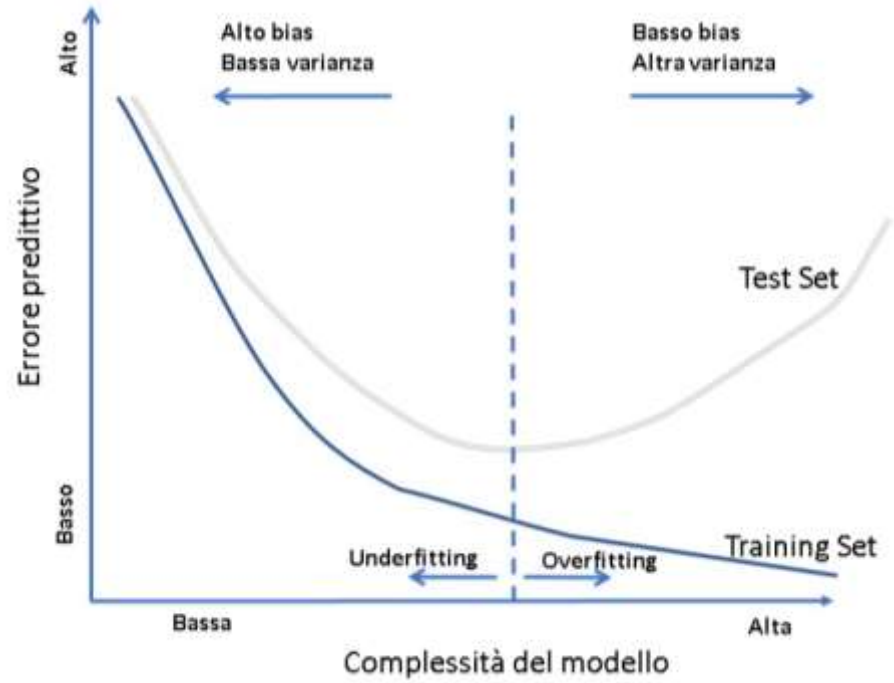
Overfitting VS underfitting

- ***Bias***

represents how far, on average, a model's predictions are from reality

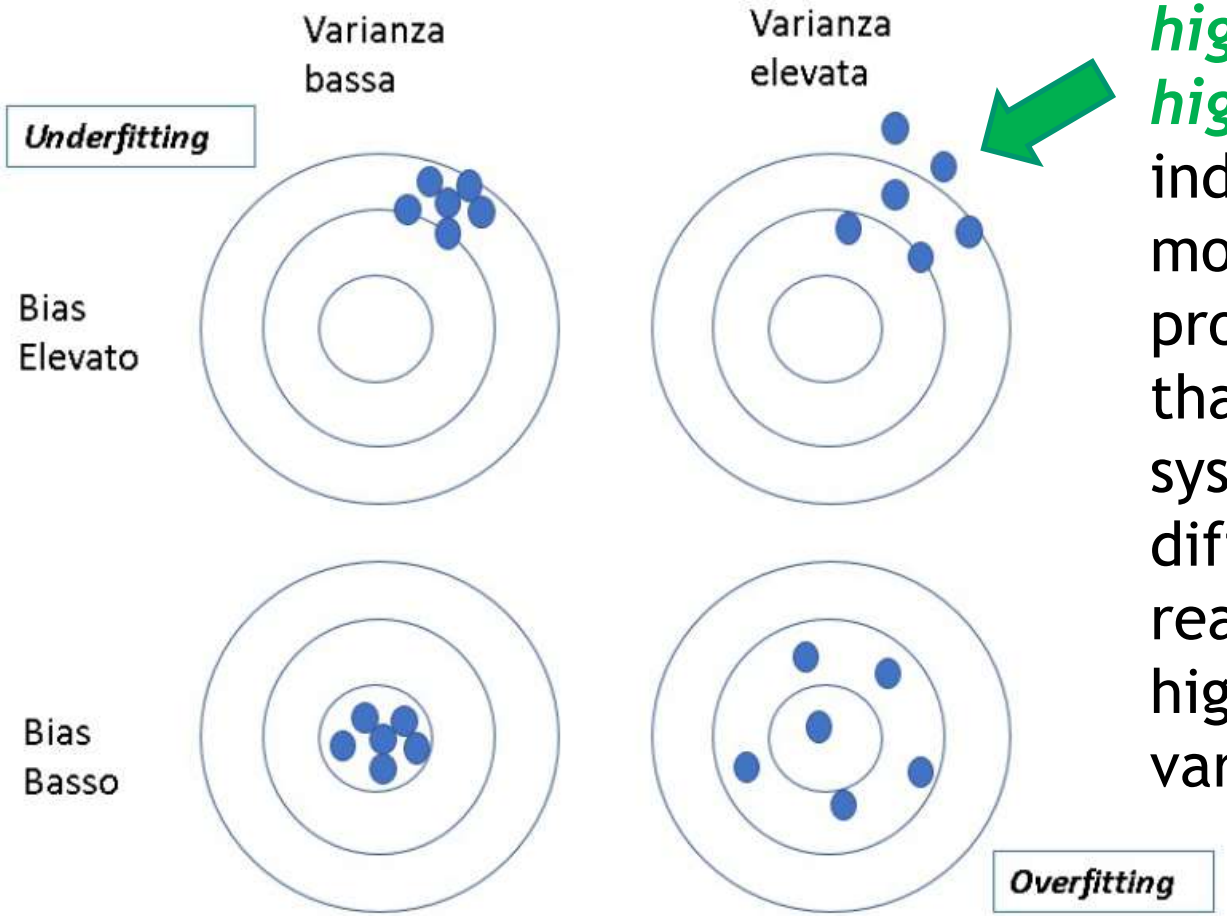
- ***Variance***

indicates how much the estimates vary around the mean.



Underfitting: in this case the model is too simple to have a good predictive performance on average.

Overfitting VS underfitting



high bias
high variance
indicate errors in modeling such as to produce results that are systematically different from reality and with a high degree of variability

optimal situation
low bias
low variance

Overfitting VS underfitting

- Suppose we have a probability distribution p(x,y) and we sample from it repeatedly to generate the training set and the test set.

- We train a model, and for some fixed value w:
  - If the expected training set error is exactly the same as the expected test set error we are ok since data are drawn (in theory) from the same distribution
  - If test set error is “greater” than training set error: **probably overfitting**
  - If training error is big itself: **probably underfitting**

- The factors determining how well a machine learning algorithm will perform are its ability to:
  - Make the training error small
  - Make the gap between training and test error small

The Validation set & Hyperparameter tuning

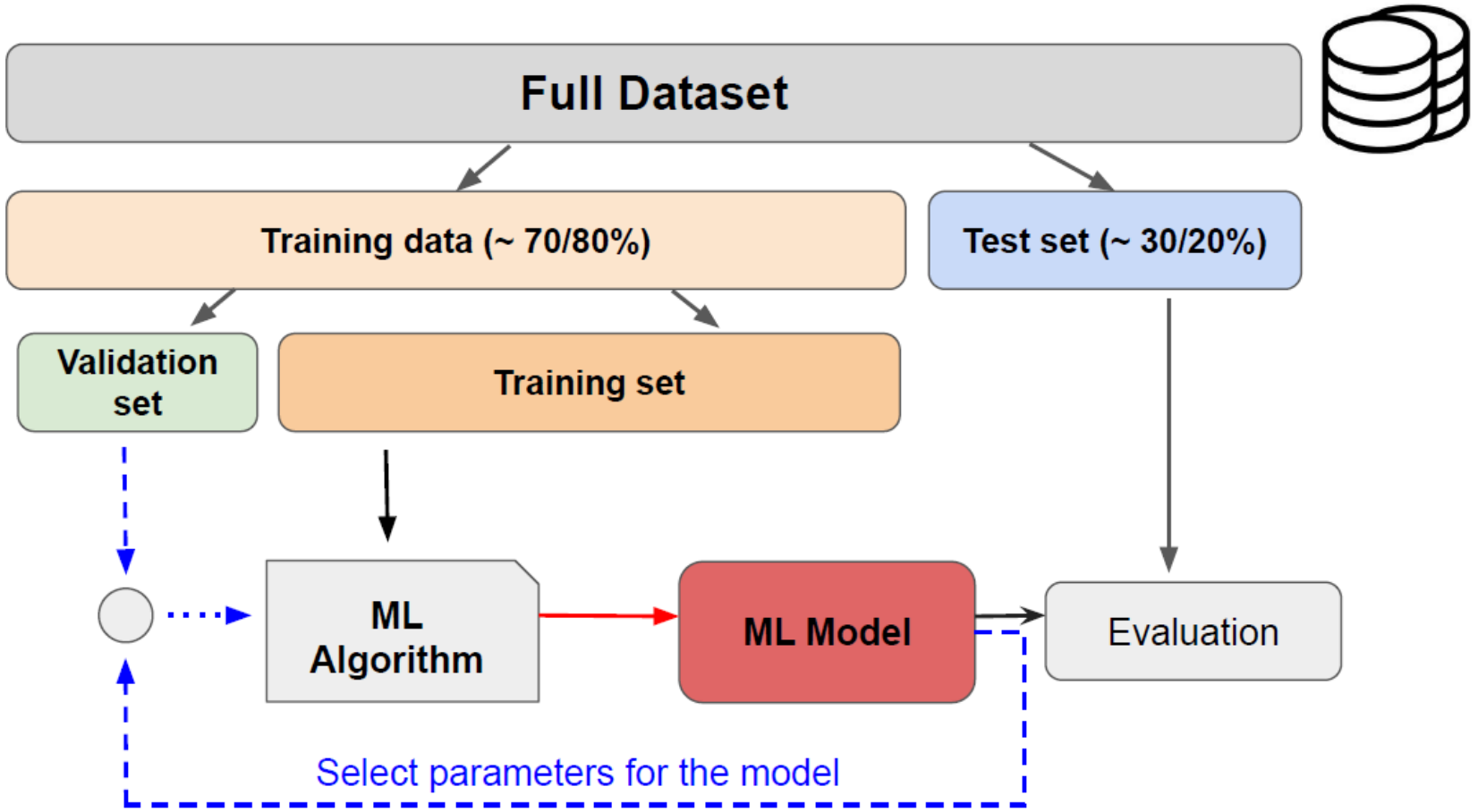
- When we have to select a model or try different values of any hyperparameter to design our algorithm or to control the model's capacity we cannot use the test-set!

- That's why the test-set is necessary to measure the final generalization error of our model and our model has not to be designed "suited" to our test-set, otherwise we cannot measure the generalization error!

- So we introduce a third set called **Validation set**

- Just to recap:
  - **Training Set**: the dataset from which we learn
  - **Validation set:** A dataset we use to tune model's parameters
  - **Test Set**: A dataset which must include patterns describing the same problem which do not belong to the training set and that you MUST NEVER USE until the end of training and design of the model!
    - The test set is used to verify whether what has been learnt in the training phase can be correctly **generalized** on new data

Data set: training, validation, test



Some problems with the data

- Examples should be a representative sample of the problem
- ✓ Number of examples (is the sample large enough?) Distribution of the examples (are the data distributed in such a way as to represent each "interesting" region of the space considered?)
- ➢ ***In general, fault situations cannot be remedied and, above all, they cannot be verified before learning***

- Examples must be 'correct‘
- ✓ Contradictions or obviously incorrect examples should be avoided
- ➢ **It is not always possible to satisfy this condition (e.g. signal with noise) but critical situations can be remedied**

Some problems with the data

- If the objective is to find the relationships existing between the variables, then one must start from the assumption that a relationship actually exists between these variables
  - ***Algorithms are unable to communicate that such a relationship does not actually exist***
- The machine is very clever, but struggles to put things into context
  - *The task of the human being to evaluate these metrics and communicate the results*

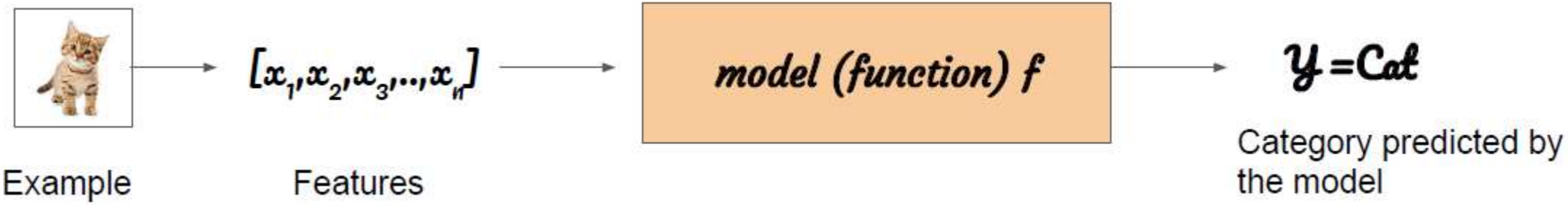
Supervised learning: Classification task

- The system is asked to specify which of k categories some input belongs to.

- To solve this task, the learning algorithm is usually asked to produce a function

$$y = f(x): R^n \rightarrow \{1,...,k\}$$

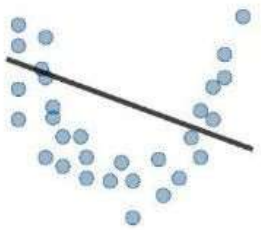
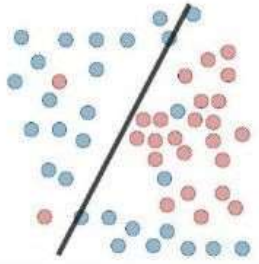
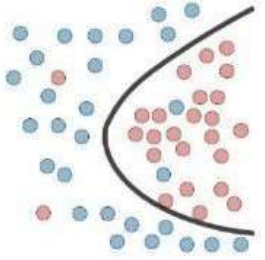
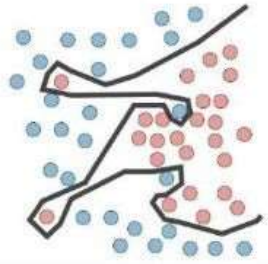
- So the model takes an example **x** as input and after some processing **f(x)** it returns a value **y** that is one of the k categories the example x should belong to.



Model for classification task

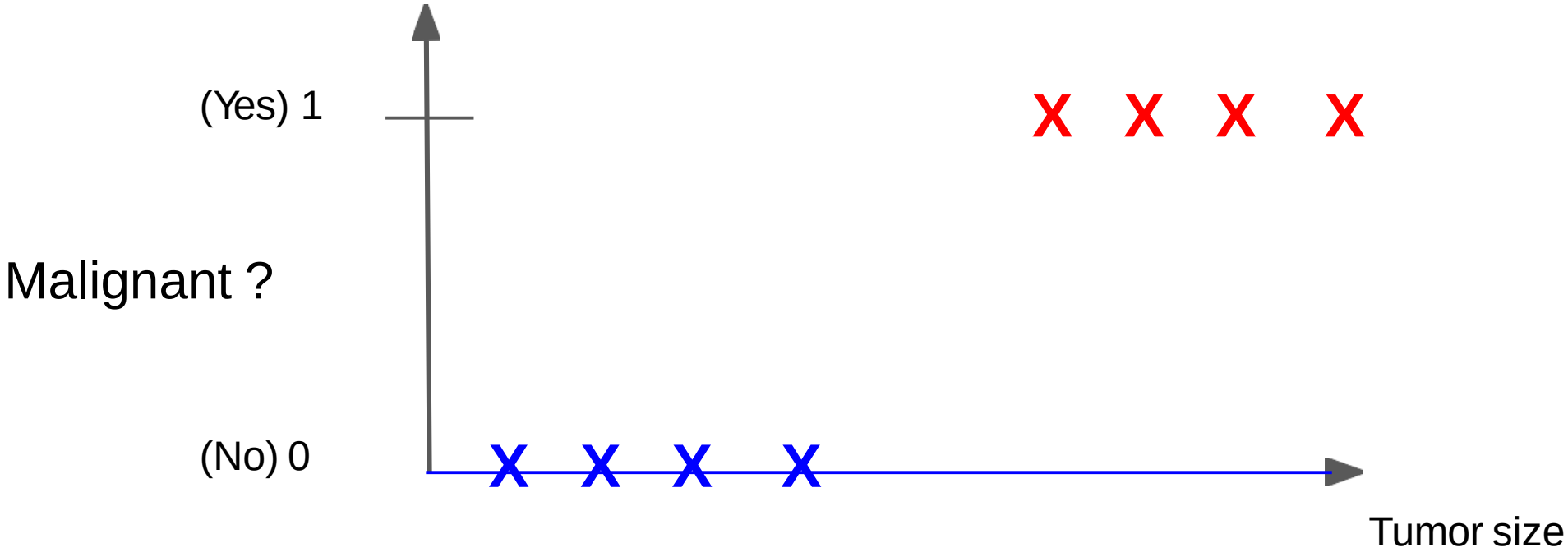
- We want to estimate a parametric function $F(X,W)$ that maps our input features X to one or more of our target labels in y, where y is an element of a limited set and is called class.

- The goal is to find the **decision boundaries** in the features space

	Underfitting	Just right	Overfitting
Symptoms	- High training error - Training error close to test error - High bias	- Training error slightly lower than test error	- Low training error - Training error much lower than test error - High variance
Regression			
Classification			

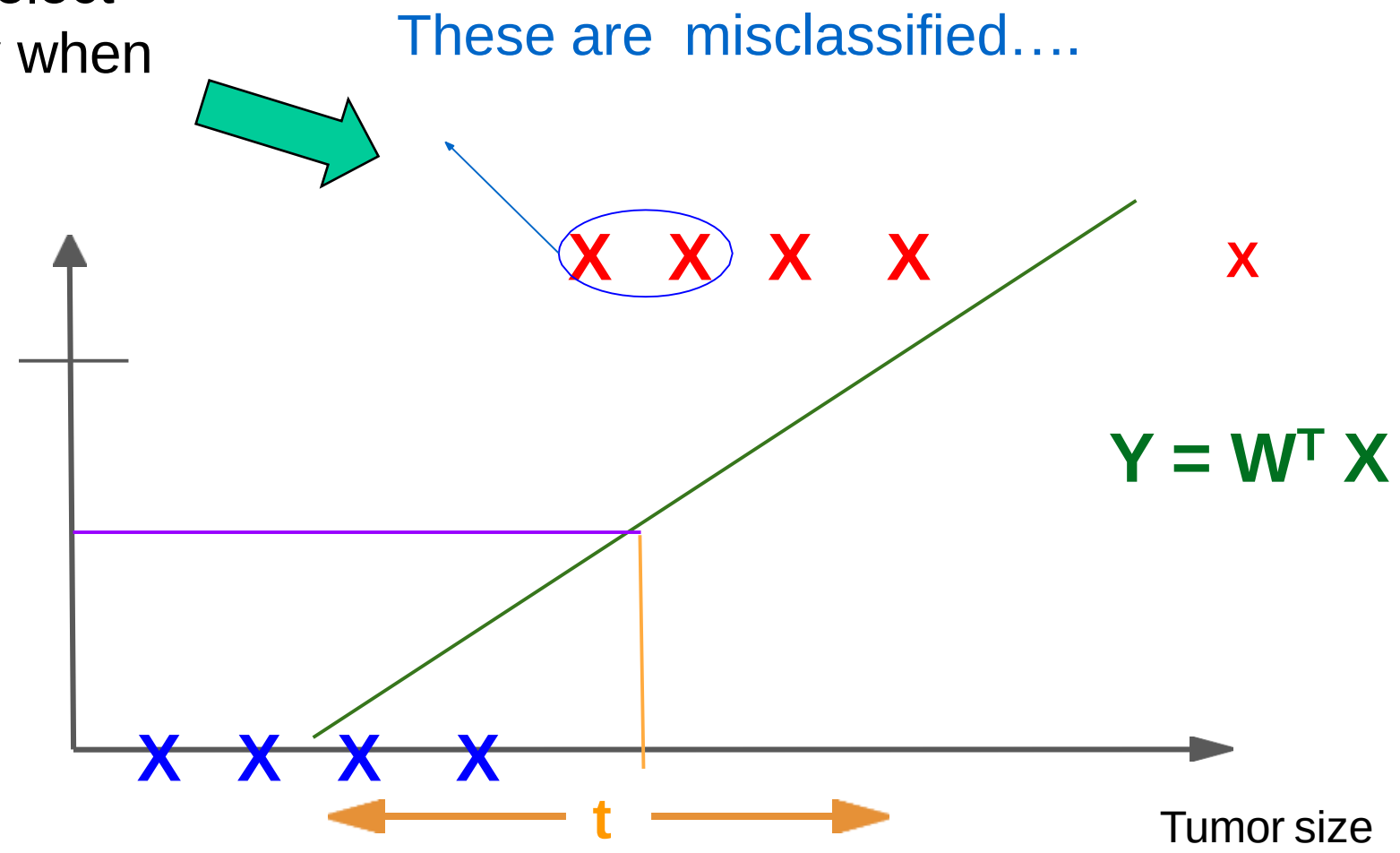
Can we adapt Linear Regression for a classification task?

- We want to train a model to predict if a tumor is benign or malignant using a single feature (tumor size)
- $y \in \{0,1\}$: 0 is malignant , 1 is benign
- X has only one feature



Can we adapt Linear Regression for a classification task?

- We can find the best $Y = W^T X$ from our dataset and then select a threshold over y to classify when the tumor is malignant or not



- Moreover, linear regression returns a y that is not bounded between 0 and 1

Logistic regression

- We want a $h_W(X)$ bounded: $0 <= h_W(X) <= 1$, the simplicity of a linear regression and a good fit to our binary classification task
  - $y = h(X) = \sigma(W^T X)$

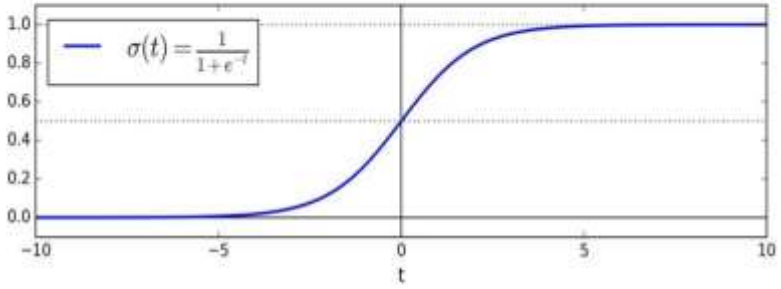
Logistic regression (sigmoid)

- output is a number between 0 and 1

$$h_W(x) = \frac{1}{1+e^{-W^T x}}$$

Logistic function

$$\sigma(t) = \frac{1}{1 + \exp(-t)}$$



Logistic regression: a classification task

- Logistic regression (despite the name) is a classification algorithm
- It estimates the probability that an instance belongs to a class or not (Binary classification)
- Just like a linear regression model, a logistic regression computes a weighted sum of the input features plus a bias term
- But instead of returning the result directly, it outputs the logistic of this result
  - So when the probability that an instance belongs to the positive class is computed, the $\mathbf{y}^*$ can be computed easily:
    - Class 0 (or negative) if $p < 0.5$
    - Class 1 (or positive) if $p >= 0.5$

When possible labels are more than two, several binary classifier are built to identify the correct class

Logistic function: classification task

- Suppose the case we have the dataset in figure, with 2 features X_1 and X_2
  - We trained a Logistic regression and we found that the best parameters W are respectively W_0= -3 , W_1=1 , W_2=1

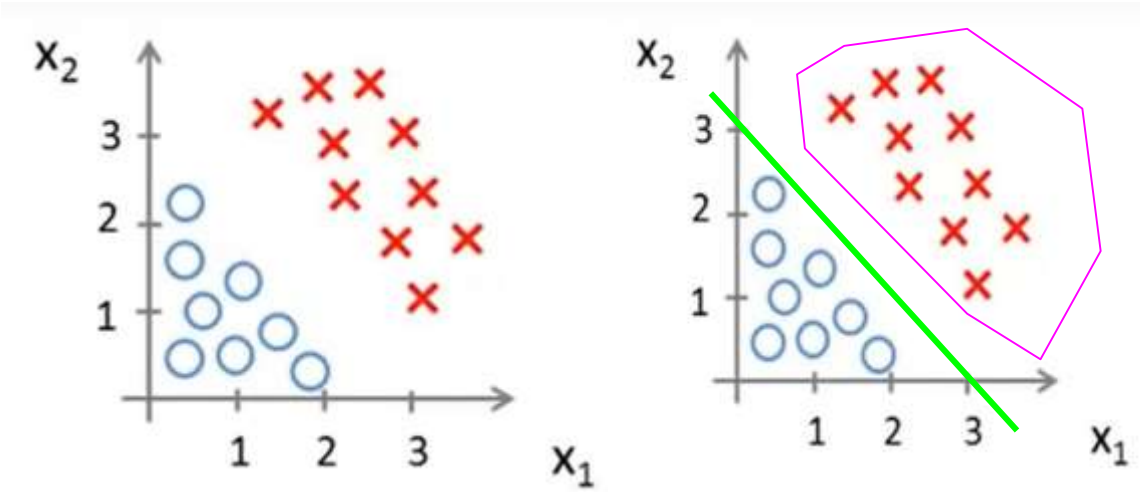
$$h_W(X) = g(w_0 + w_1x_1 + w_2x_2)$$

- We predict y=1

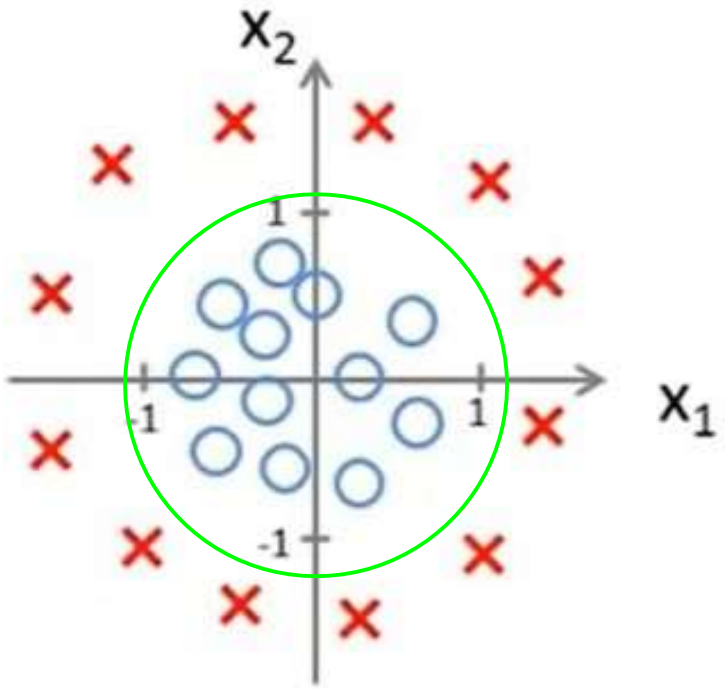
when $-3 + x_1 + x_2 >= 0$

  - So when

    $x_1 + x_2 >= 3$



Non-linear decision boundaries



$$h_W(X) = g(w_0 + w_1 x_1 + w_2 x_2 + w_3 x_1^2 + w_4 x_2^2)$$

How to evaluate a classification task?

- Confusion matrix
  - Accuracy
  - Precision
  - Recall
  - F-1 measure

How to evaluate a classification task?

Example: Iris plants dataset

Number of Instances

150 (50 in each of three classes)

Number of Attributes

4 numeric, predictive features and the class

Features Information

- sepal length in cm
- sepal width in cm
- petal length in cm
- petal width in cm
- **class:**
  - Iris-Setosa
  - Iris-Versicolor
  - Iris-Virginica

Confusion matrix

Total		Predicted		
		Iris-setosa	Iris-versicolor	Iris-virginica
Real class	Iris setosa	14	1	1
	Iris-versicolor	0	16	0
	Iris-virginica	0	1	15

Accuracy = (14+16+15)/48= 0,9375

Accuracy

- The accuracy is defined as the ratio among the number of correct predictions over the total number of predictions
- The accuracy is often expressed as a percentage
- *If the test-set is unbalanced we must use other metrics*

- Suppose a dataset has 800 images of cats and 200 images of dogs
- We use 100 images for the test, but in this test-set we randomly selected:
  - 90 cats
  - 10 dogs

- The accuracy on the test-set is 90%
  - What does this accuracy mean?
  - If I use a “dummy” script that says every time “cat” without any learning it reaches the same accuracy
  - But if the test-set were balanced the “dummy” script would reach only 50% of accuracy

Accuracy: Balanced test-set required

Test-set MUST BE BALANCED if you want to measure the accuracy

- **What about the training-set ?** Do we have to balance also it ?
  - It depends, if the training-set is unbalanced probably the less representative class will be less learnt by the model
  - It means that the model will probably predict that class less than the others because it will be less confident with that class
  - Usually it is better to have an almost balanced training-set, but sometimes we could desire to predict a label less than another

If unbalanced test …. Precision

In a binary classification we can divide predictions in True Positive (TP), True Negative (TN), False positive (FP), False Negative (FN)

		Predicted	
		Negative	Positive
Real class	Negative	TN	FP
	Positive	FN	TP

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

- It is like an accuracy on a single class
- Alone does not give enough information since a trivial way to have a perfect (precision = 1/1 = 100%) is to use a test-set with just one example
- So usually precision is used along with another metric called **Recall** that takes into account the False negative

Precision, Recall and F-1 Score

		Predicted	
		Negative	Positive
Real class	Negative	TN	FP
	Positive	FN	TP

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

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

$$F_1 = \frac{2}{\frac{1}{Precision}+\frac{1}{Recall}}$$

- Precision can be thought as the accuracy of the positive predictions
- Recall (also known as Sensitivity or True positive rate) is the ratio of positive instances that are correctly detected by the classifier
- F1 is the harmonic mean of precision and recall. Whereas the regular mean treats all values equally, the harmonic one gives much weight to low values. As a consequence we can get a high F1 score only if both precision and recall are high

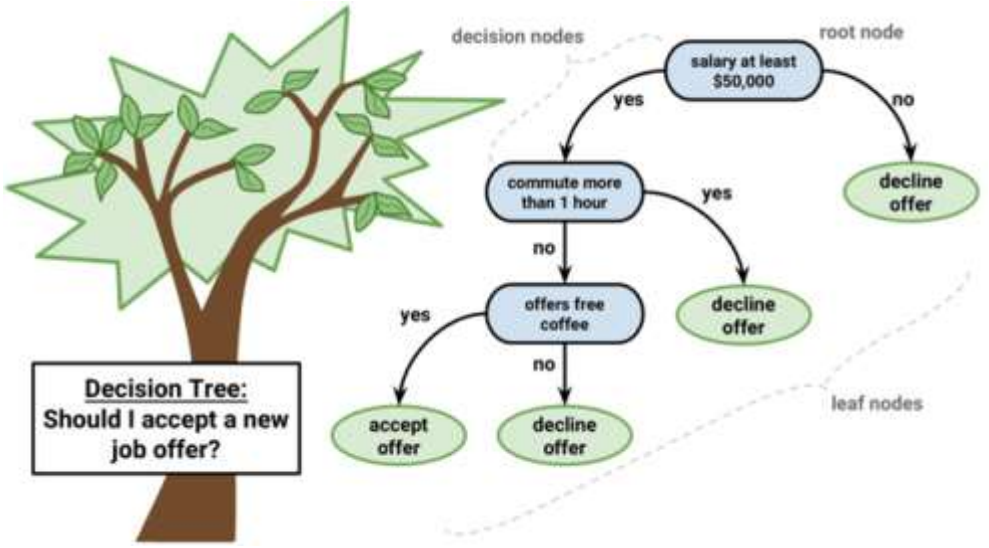
Model for classification task: decision tree

- The algorithm is based on the concept of entropy in information theory
  - In Information Theory, it measures the average information content of a message (Shannon's theorems) and it is zero when all messages are identical

- Gini: impurity metric
  - A node is "pure" if Gini=0
  - If all training instances it applies to belong to the same class

$$G_i = 1 - \sum_{k=1}^{n} p_{i,k}^{\ 2}$$

- $p_{i,k}$ is the ratio of class k instances among the training instances in the i^{th} node.



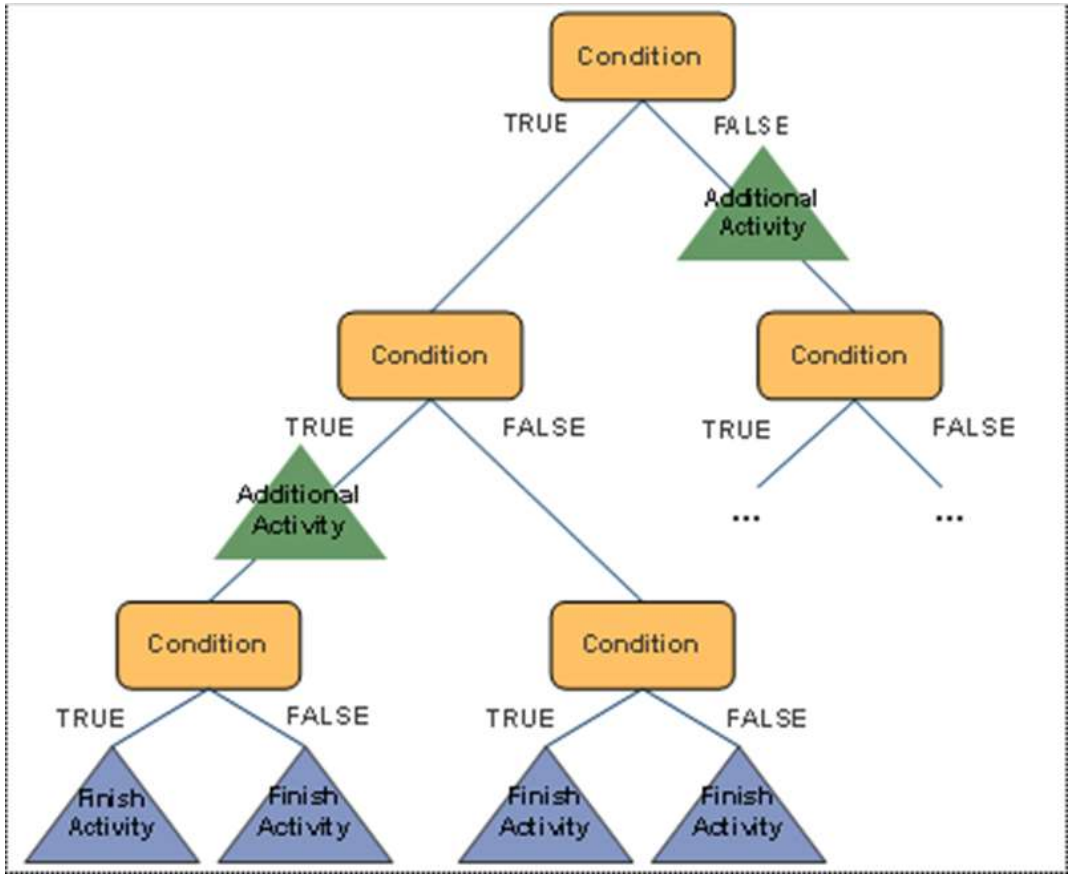
Model for classification task: decision tree

The advantages of having a decision tree are as follows:

- It does not require any domain knowledge.
- It's easy to understand.
- The learning and classification steps of a decision tree are quick and easy.

Disadvantages:

- They are unstable, which means that a small change in the data can lead to a large change in the structure of the optimal decision tree.
- They are often relatively inaccurate



Model for classification task: Naive Bayes

- It is based on the probability of an element to belong to a certain class
- It determines the class to which it belongs based on the conditional probabilities for all classes based on the attributes of the various elements.
- The correct classification occurs when the conditional probability of a certain class C with respect to the attributes is maximum.

- The algorithm has the following strengths:
  - Works well in case of "noise" in a data part.
  - Tends not to consider irrelevant attributes.
  - Model training is much easier than other algorithms.
- The reverse of the coin is represented by the assumption of the independence of the attributes, which may not be present in reality

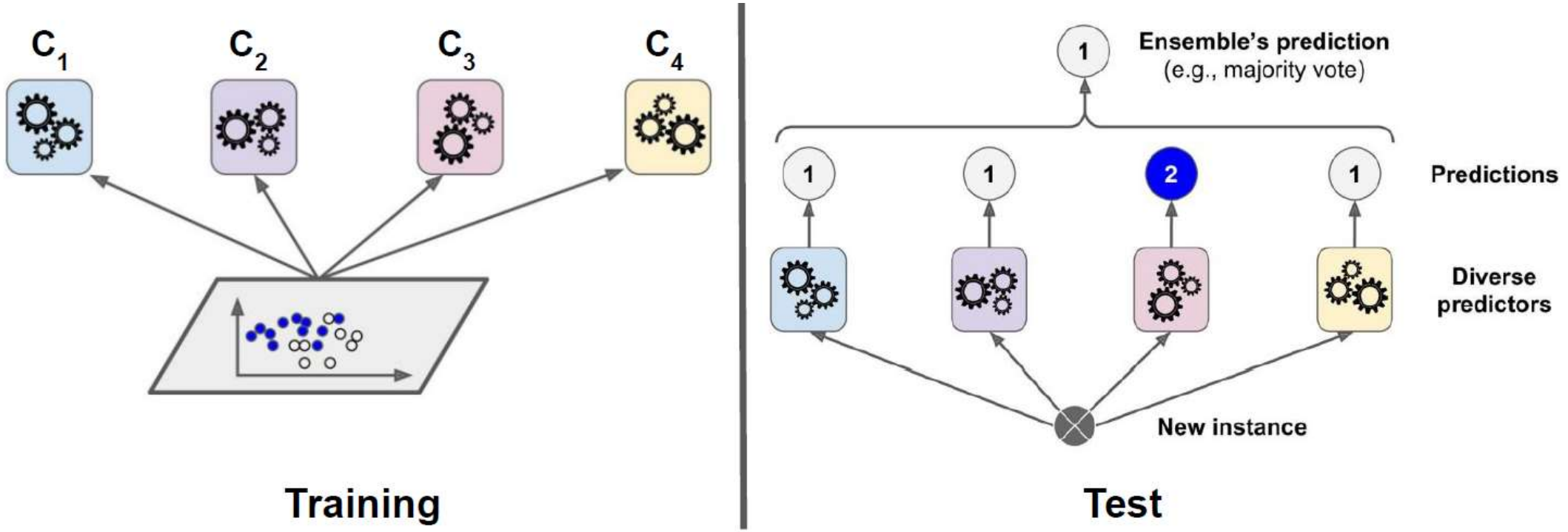
Ensemble Learning

- When you are looking for an answer to a complex problem it is better to ask to different experts and then aggregate their answer (Wisdom of the crowd)

- ***Key idea: aggregate the predictions of a group of predictors to get better predictions than the best single predictor***

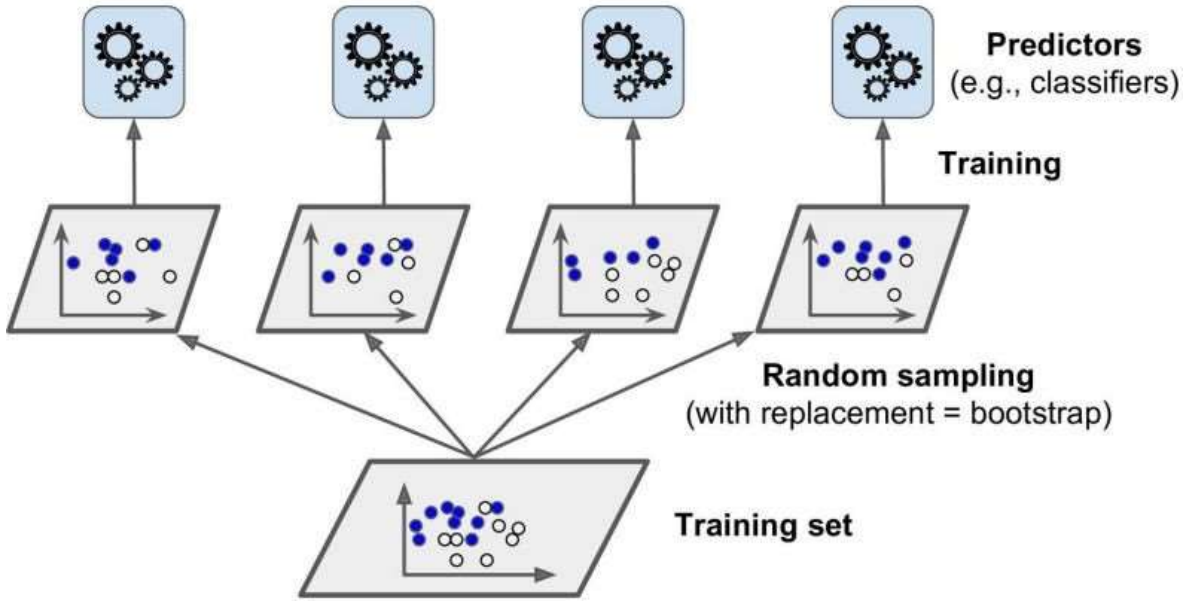
- The idea of ensemble learning is to select a collection, or ensemble, of hypotheses, h1,h2, . . . ,hn, and combine their predictions by averaging, voting, or by another level of machine learning.
- We call the individual hypotheses ***base models*** and their combination an ***ensemble model***

Ensemble Learning



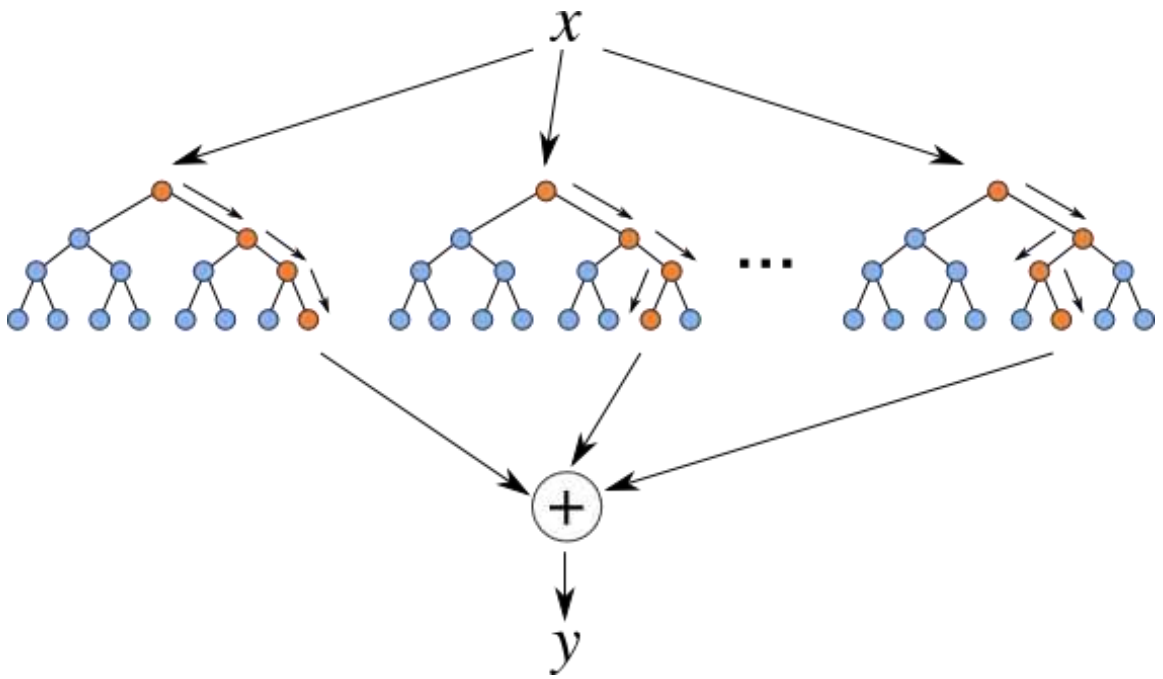
Bootstrap aggregating (Bagging)

- Only one training-set and only one algorithm
- Train the classifiers on different random subsets of the training-set
- When sampling is performed with replacement, this method is called **bootstrap**
- Each individual predictor has a higher bias than if it were trained on the original dataset, but aggregation reduces both bias and variance



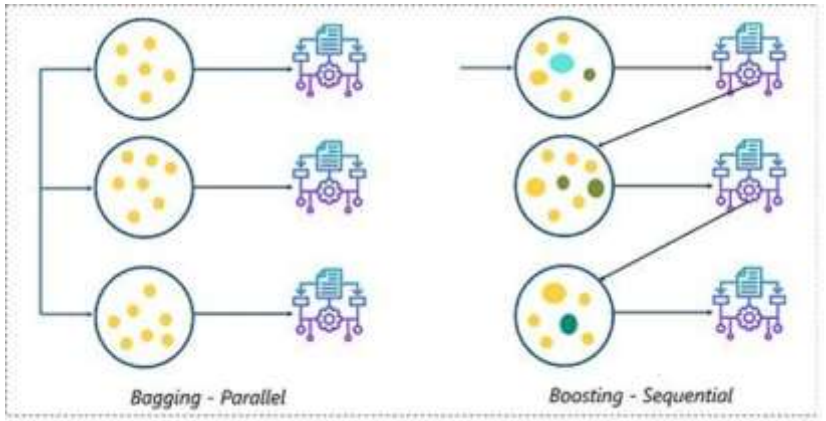
Random forest

- Random forest is a bagging-based model where we make an aggregation of decision trees
- It also sub-samples a fraction of the features when fitting a decision tree to each bootstrap sample



Boosting

- Similar to bagging, but now the idea is to train predictors sequentially
- Each predictor tries to correct its predecessor
    - Many boosting methods available:
        - Adaptive boosting (AdaBoost)
        - Gradient boosting
        - XGboost (Extreme gradient boosting)



weighted training set : each example has an associated weight wj ≥ 0 that describes how much the example should count during training.
=> if one example had a weight of 3 and the other examples all had a weight of 1, that would be equivalent to having 3 copies of the one example in the training set.

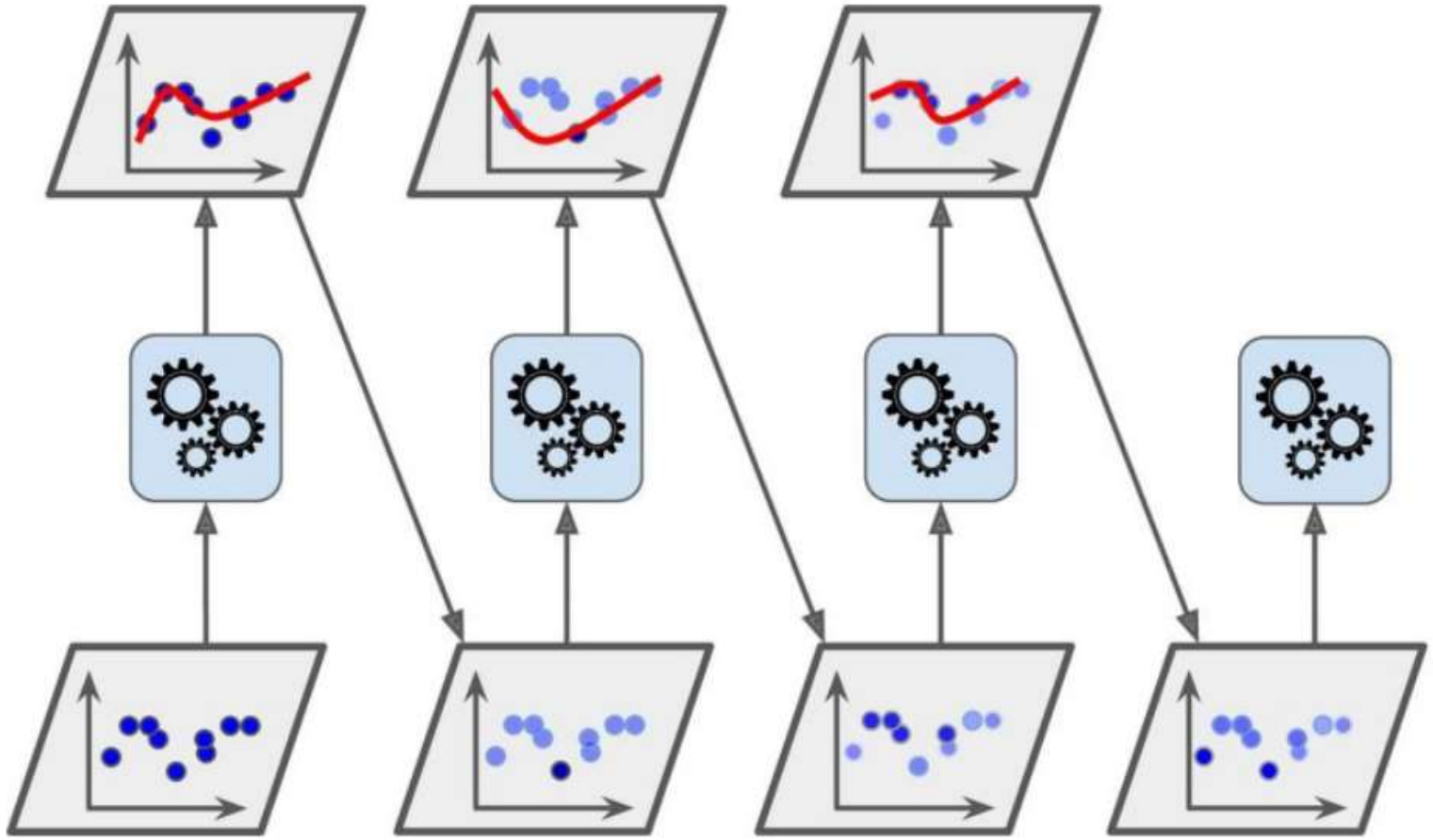
AdaBoost

- To correct a predecessor one way is to pay attention to the training instances that the predecessor under-fitted

- The new predictor will focus more on the hard cases (adaptive)
  a. A first base predictor is trained and used to make predictions on the training-set
  b. The relative weight of misclassified training instances is then increased
  c. A second classifier is trained using the updated weights and so on...

  d. A weight is assigned to each classifier
  e. Once all predictors are trained, the ensemble makes predictions like bagging, except that predictors have different weights depending on their accuracy on the training-set

AdaBoost

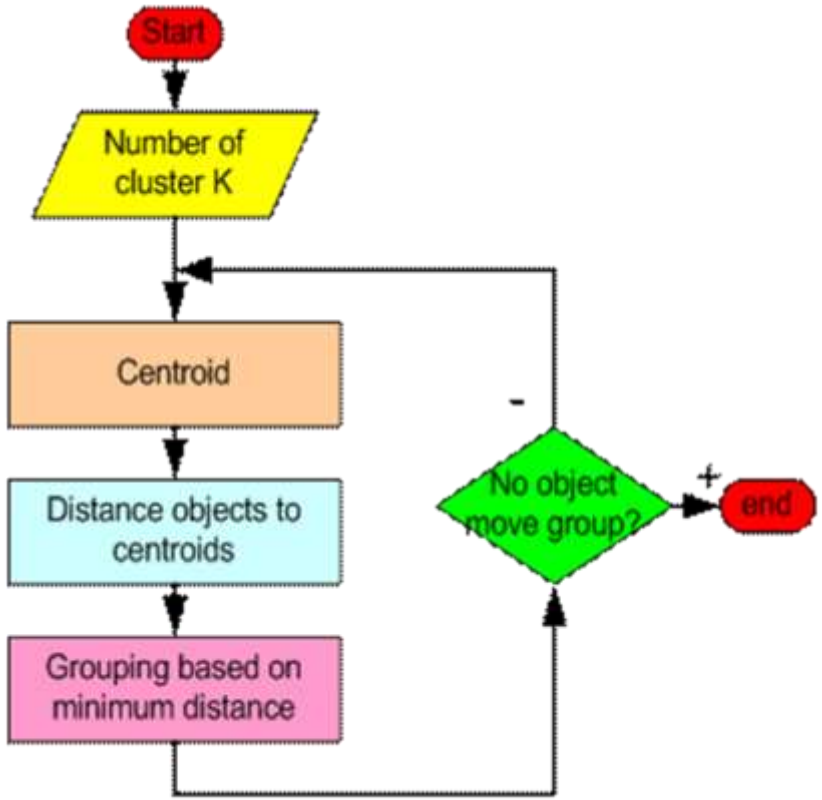


Unupervised learning: Clustering task

- The system is asked to group a set of objects in such a way that objects in the same group (called a cluster) are more similar to each other than to those in other groups.
- It is useful to discover latent properties or irregularities among data
- Usually data are unlabelled: We don’t know anything about what is the correct group the example should belong to

K-means Clustering

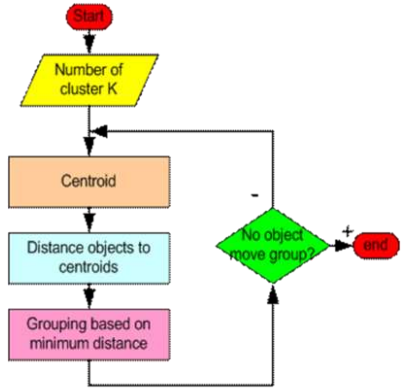
- Strengths
  - Simple iterative method
  - User provides “K”
- Weaknesses
  - Often too simple → bad results
  - Difficult to guess the correct “K”



K-means Clustering

Basic Algorithm:

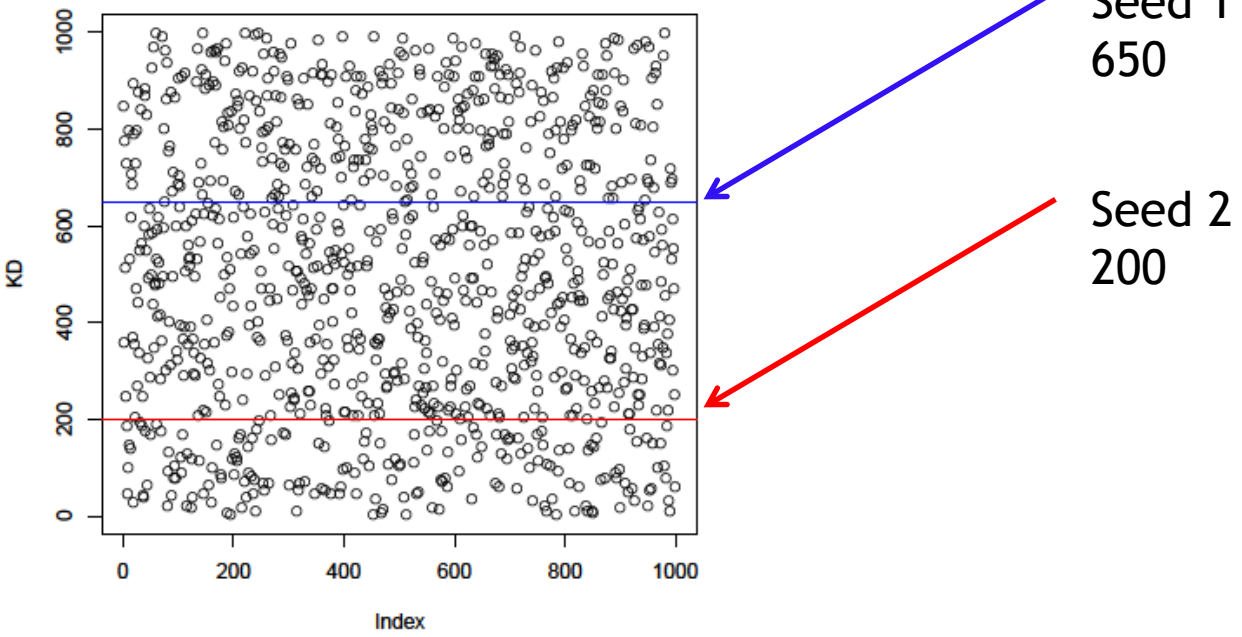
- Step 0: select K
- Step 1: randomly select initial cluster seeds



An initial cluster seed represents the "mean value" of its cluster.
In figure:

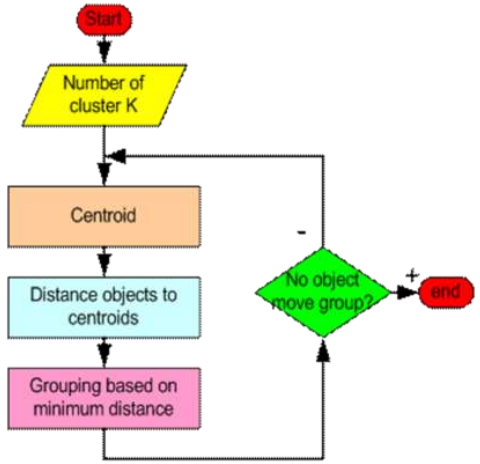
Cluster seed 1 = 650

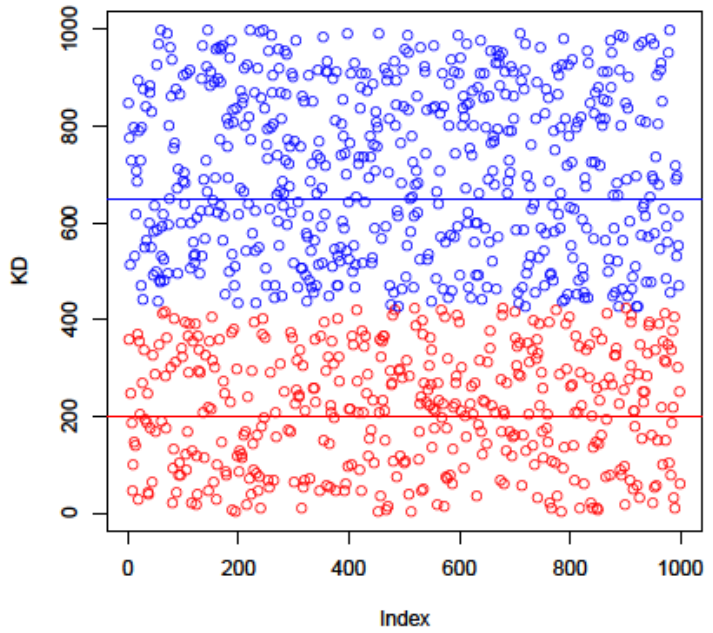
Cluster seed 2 = 200



K-means Clustering

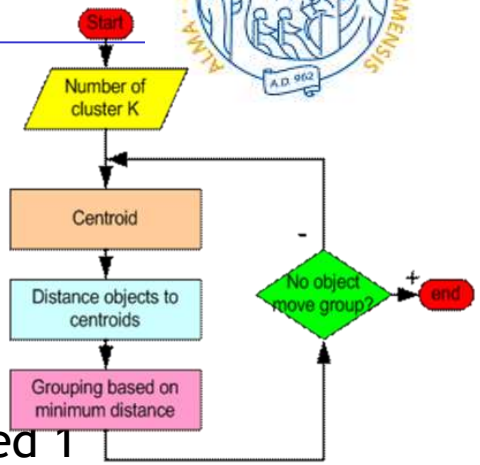
- Step 2: calculate distance from each object to each cluster seed.
- What type of distance should we use?
  - Squared Euclidean distance
- Step 3: Assign each object to the closest cluster

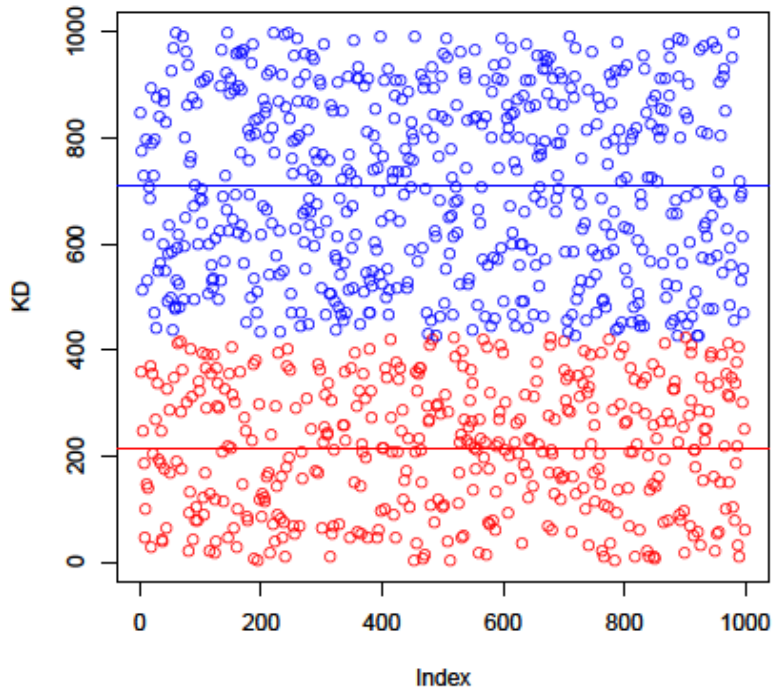




K-means Clustering

- Step 4: Compute the new centroid for each cluster



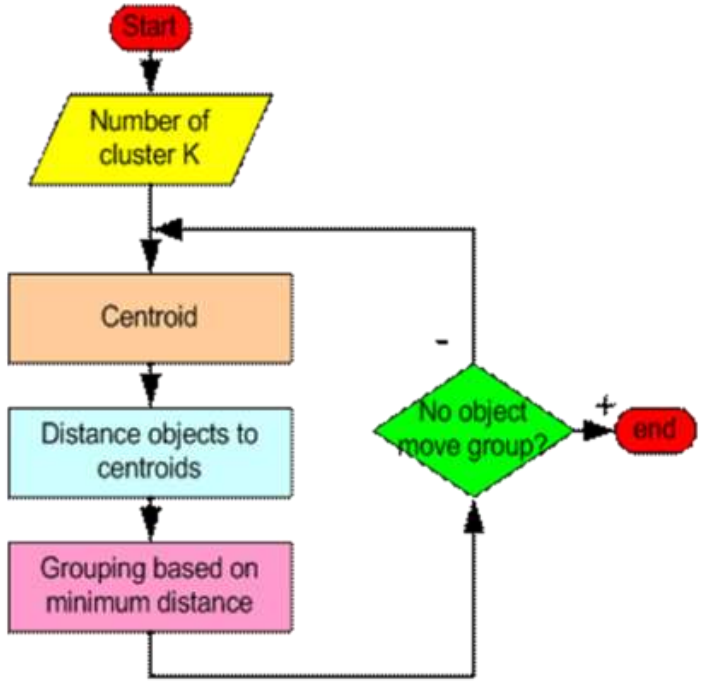


Cluster Seed 1
708.9

Cluster Seed 2
214.2

K-means Clustering

- Iterate:
  - Calculate distance from objects to cluster centroids.
  - Assign objects to closest cluster
  - Recalculate new centroids
- Stop based on convergence criteria
  - No change in clusters
  - Max iterations



ML in python

- https://scikit-learn.org/stable/