ESCOLA SUPERIOR DE TECNOLOGIA IPCA

Machine Learning Algorithms

Fundamentals of Artificial Intelligence

MSc in Applied Artificial Intelligence, 2023-24

Contents

- Topics included
  - Supervised Vs unsupervised Learning
  - Automatic classification
  - Case Based Reasoning
  - Decision Trees
  - Clustering
  - Association Rules

- These slides were based essentially on the following bibliography:
  - Han, J., and Kamber, M. (2011). Data Mining: Concepts and Techniques. 3rd Edition, Morgan Kaufmann Publishers
  - Norvig, P, Russell, S. (2021). Artificial Intelligence: A Modern Approach, 4th Edition. Pearson, ISBN-13: 978-1292401133
  - Provost, F., Fawcett, T. (2013). Data Science for Business: What you need to know about data mining and data-analytic thinking. O'Reilly.

Automatic classification

Automatic classification

Classification aims the automatic assignment of classes/categories

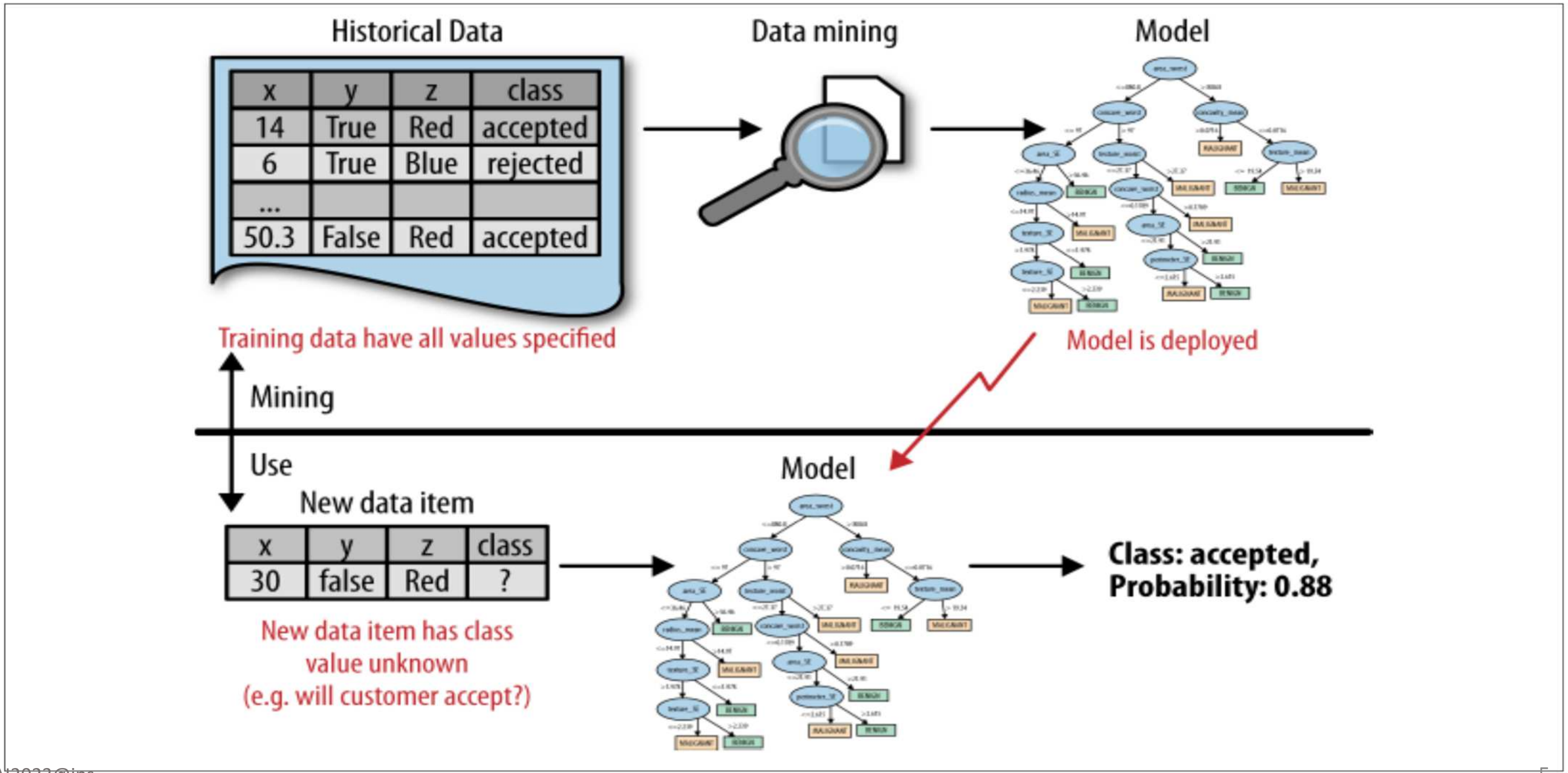
- Automatically assigns a class to the new data
- The class attribute is discrete (few distinct values)
- The model is based on the existing relationships between the remaining attributes and the class attribute.

Model built based on a set of training data (historical records) in 2 steps:

- Step 1: build the classification model
  - Each item belongs to a predefined class
  - Constructed from historical, previously classified, the training data set
- Step 2: use of the model
  - Classify data whose class is unknown

Prediction = classification with continuous values

Model building and use

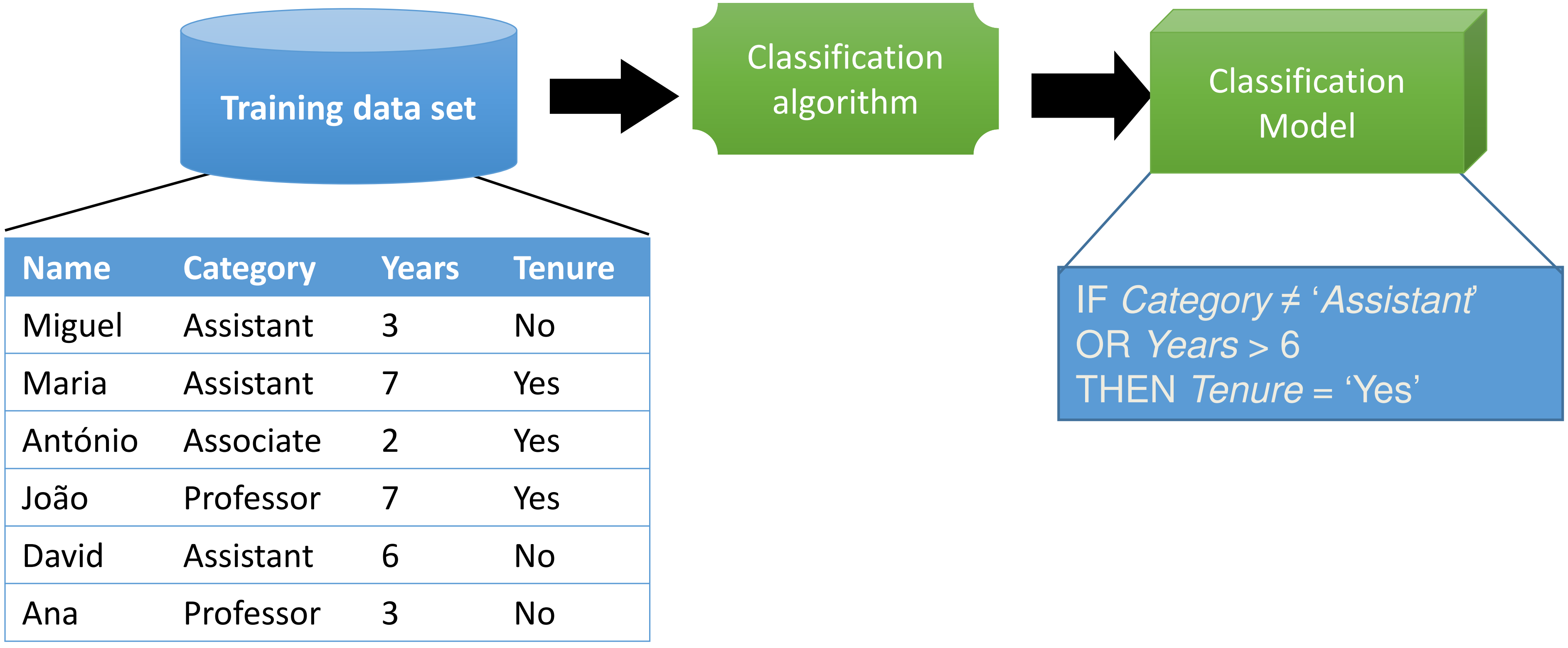


x	y	z	class
14	True	Red	accepted
6	True	Blue	rejected
...			
50.3	False	Red	accepted

x	y	z	class
30	false	Red	?

Historical Data
Data mining
Model
Training data have all values specified
Model is deployed
Mining
Use
Model
New data item
New data item has class value unknown (e.g. will customer accept?)
Class: accepted, Probability: 0.88

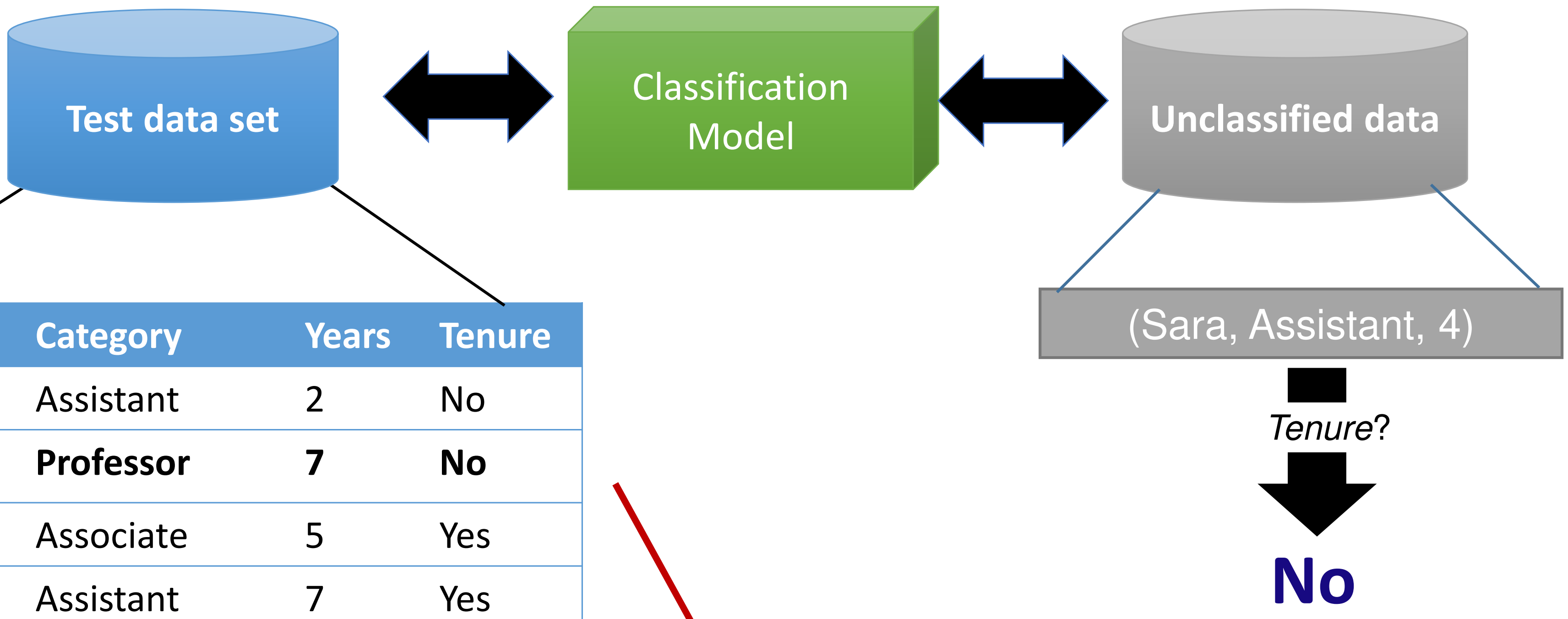
Classification: building the model



Name	Category	Years	Tenure
Miguel	Assistant	3	No
Maria	Assistant	7	Yes
António	Associate	2	Yes
João	Professor	7	Yes
David	Assistant	6	No
Ana	Professor	3	No

IF *Category* ≠ ‘*Assistant*’
OR *Years* > 6
THEN *Tenure* = ‘Yes’

Classification: model use



Name	Category	Years	Tenure
Bruno	Assistant	2	No
Daniel	**Professor**	**7**	**No**
Ana	Associate	5	Yes
Pedro	Assistant	7	Yes

The model will not be always 100% accurate!

Data preparation

- **Data preparation** or data prep is the process of cleaning, structuring and enriching raw data
    - discover features of data and determine the value of dataset
    - structure the data, e.g., splitting columns, pivoting rows or deleting fields
    - clean, identifying data quality issues, such as missing or mismatched values, and apply the appropriate transformation to correct or delete
    - enrich your existing dataset by joining and aggregating multiple data sources
    - validate that the output dataset has the intended structure and content

- **Data wrangling** is the process of converting data into another format for more convenient consumption of the data with the help of semi-automated tools.

Metadata

- Data attributes can be of various types: binary, nominal, ordinal, numeric, dates, etc.

- Common attribute roles in data science:
  - **input** - attributes as model input
  - **target** - output attributes of the model
  - **id/auxiliary** - although useful, should not be used in modelling
  - **ignore** - attributes not used in the modeling process
  - **weight** - weight of each attribute as model entry

- In addition to the data type/role, the metadata developed in the data study shall also include a detailed description of each attribute.

Cleaning of data: transformation

- In many cases it is necessary to convert the data to standard formats, e. g., csv, XML, json, etc.

- Transform the data:
  - Treat null or unknown values
  - Convert the dates to a unified numerical format
  - Discrete numerical data, if necessary
  - Treat errors and outliers
  - Convert nominal attributes of values with sorting to numeric values

Evaluating Classifiers

- How we do this depends on how much data is available
- If there is unlimited data available, then there is no problem
  - Usually, we have less data than we would like so we have make trade-offs

- Evaluation on “small” data: **hold-out testing sets**
  - Hold-out
  - Repeated hold-out

- For "very small" data: **Cross validation**
  - K-fold cross validation
  - Leave-one-out validation

Enough data

- If many (thousands) of examples are available, including several hundred examples from each class,
  - A simple evaluation is sufficient
- **Hold-Out testing** sets
  - Randomly split the available data into a training set (**2/3 of the data**) and a test set (**1/3**)
  - Build a classifier using the train
  - Evaluate it using the test set
- You may get an unfortunate split
  - Holdout estimate can be made more reliable by repeating the process with different subsamples
  - In each iteration, a certain proportion is randomly selected for training (possibly with stratification)
  - The error rates on the different iterations are averaged to yield an overall error rate
- This is called the **repeated holdout method**
  - Still not optimum: the different test sets overlap.
  - Can we prevent overlapping?

Evaluation on “small” data

- The **holdout method** reserves a certain amount for testing (1/3) and uses the remainder for training
  - For “unbalanced” datasets, samples might not be representative
  - When there are few or none instances of some classes
- Stratified sample:
  - advanced version of balancing the data
  - Make sure that each class is represented with approximately equal proportions in both subsets
- **K-fold cross-validation** avoids overlapping test sets
  - First step: data is split into k subsets of equal size
  - Second step: <u>each subset in turn is used for testing</u> and the remainder of available data is used for training
- Often the subsets are stratified before the cross-validation is performed
- The <u>error estimates are averaged</u> to yield an overall error estimate

K-fold cross validation

- The main advantages of k-fold cross validation are that every example is used in testing at some stage and the problem of an unfortunate split is avoided
- Any value can be used for k
  - **K = 10 is most common**
  - Depends on the data set

- Extensive experiments have shown that ten is the best choice to get an accurate estimate
- Stratification reduces the estimate’s variance
- Even better: repeated stratified cross-validation
  - E.g. ten-fold cross-validation is repeated ten times and results are averaged (reduces the variance)

Leave-One-Out Cross-validation

- **Leave-One-Out** is a particular form of cross-validation:
  - Set number of folds to number of training instances
  - i.e., for n training instances, build classifier n times
- Makes best use of the data
  - Involves no random subsampling
- Very computationally expensive
  - Used when there is very "small" data

Case Based Reasoning

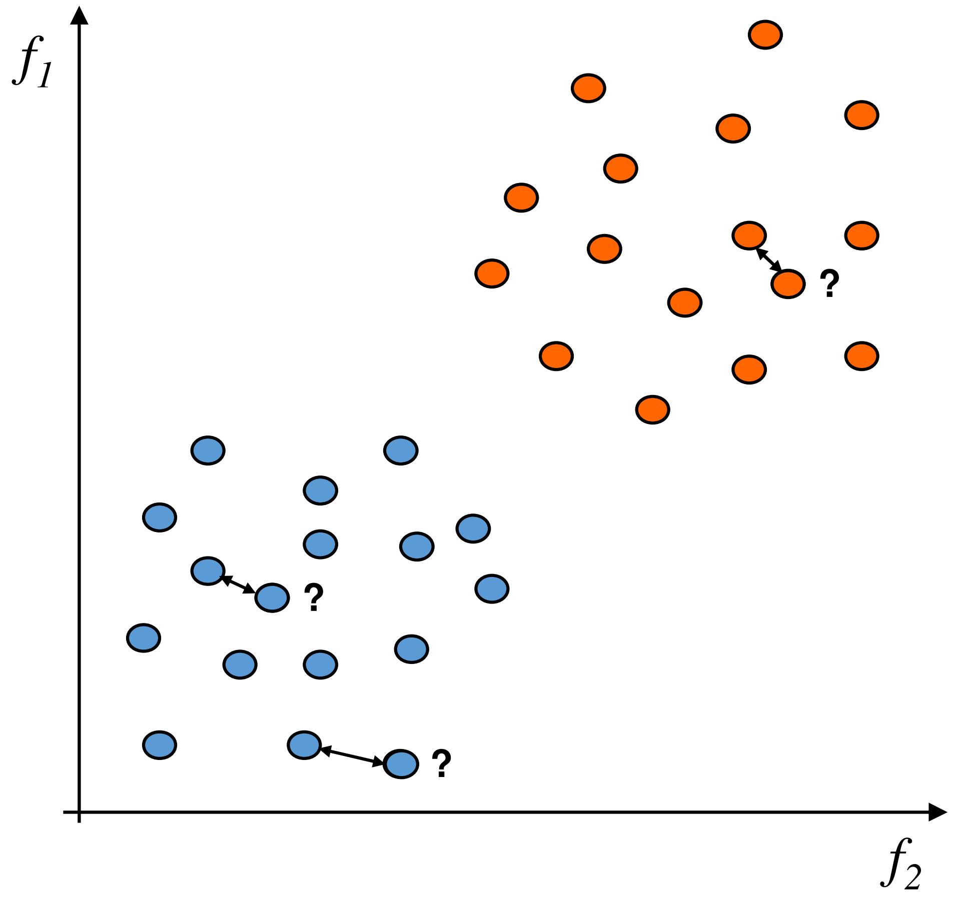
Case Based Reasoning

- One of the simplest approaches
    - Assigns the same class as the most similar object or item.
- The algorithm of the nearest neighbor,
    - k-nearest neighbor (k-NN) algorithm
    - Defined by calculating the distance (Euclidean or other)
- Approach used to classify or predict
    - For discrete values, the k-NN algorithm returns the most common value between the closest training data records
    - For actual values, the k-NN algorithm returns the combination, e. g. average, of the nearest training data records

Example of the nearest neighbor

- **First**, decide which is the case of the data that is closest or similar, calculating the new distance case to all cases or instances of the training set

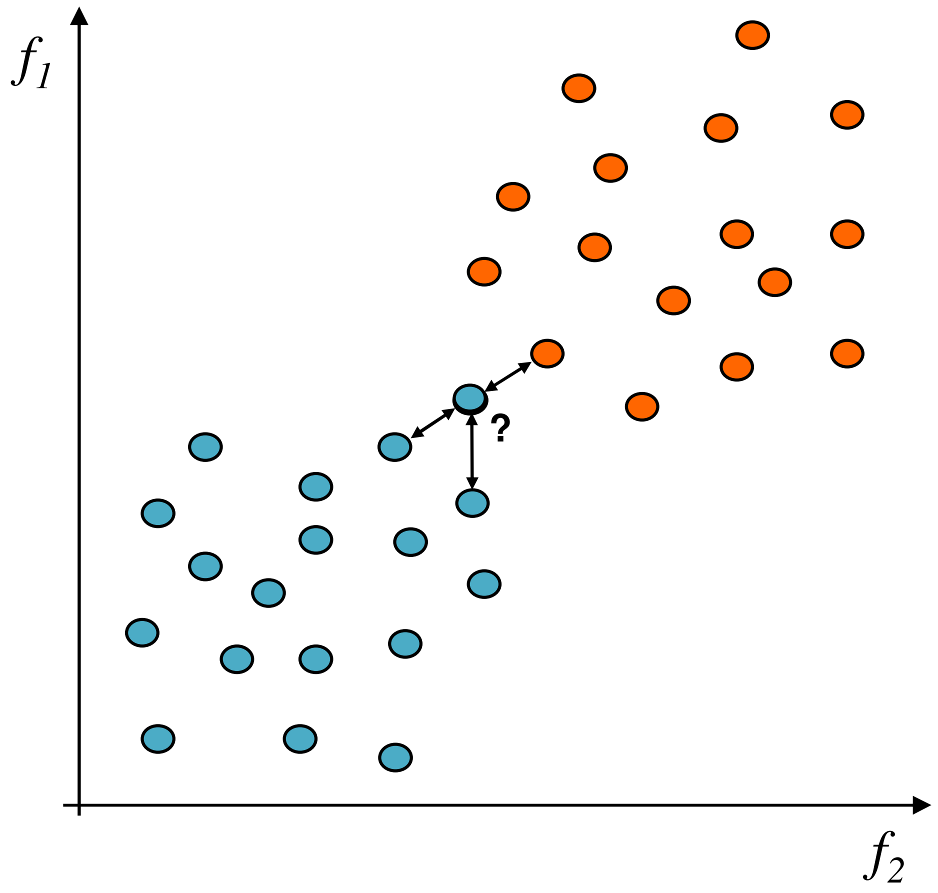
- **Second**, it is attributed to the new case the same class of its closest neighbor



Example of the nearest neighbor (2)

- If 2 instances of the training set of different classes are at the same distance of the new case?

- k-NN (k-Nearest Neighbor)
  - The nearest k neighbors are used to assign the class
  - It is attributed to the dominant class its k neighbors
  - Increased robustness of the algorithm for deviations or outliers



Measures used in k-NN

- In order to determine the next neighbor it is necessary to use a distance measure, according to the specific context of the problem in question
- Euclidean distance is the most widely used formula:

$$d = \sqrt{\sum_{i=1}^{n} (t_i - q_i)^2}$$

where n is the number of characteristics, ti is the i-nth characteristic of the training data set and qi is the i-nth characteristic of the element to be classified.

Summary of k-NN classification

- **Weaknesses**
  - It's not a powerful kind of classification
  - Weak performance, i. e., slow ranking
  - It is not scalable: performance deteriorates tremendously with increased attributes (curse of dimensionality)

- **Strengths**
  - It's one of the easiest methods of classification to understand
  - No prior training required (lazy learning)
  - It's relatively easy to understand
  - It's noise-resistant
  - New cases may be added at any time

Decision trees

Induction of decision trees

- Classification technique widely used as a data mining tool
    - The classification problem is formulated in a tree composed of decision nodes (branches) and classification nodes (leaves)
    - The classification is carried out by navigating the root of the tree until it reaches a leaf that corresponds to the class to be assigned

- A good classification algorithm should create efficient and powerful classification trees.
    - There are lots of classification algorithms
    - ID3 and C4.5 by J. Ross Quinlan are among the best-known algorithms

Algorithm for tree induction

- The basic induction algorithm, type ID3, can be described in the following steps:
  - The decision tree is **built from the root** in a recursive approach
  - At first, all the records/items of the training set are placed at the root
  - All attributes are **discrete** - if they are continuous, they should be discredited
  - Records are split recursively according to the value of their attributes
  - The attributes to be used at each node are selected on the basis of a heuristic or statistical measure, e. g., the **information gain**

- The algorithm will finish the partitioning process when one of the conditions is met:
  - All the records/items of a given node belong to the same class
  - There are no more attributes to continue partitioning, and the sheet is assigned the majority class of the items placed on it
  - There are no more training dataset entries/items to use

Information gain

- Heuristics used as a mechanism for selecting the attribute to be used in each node
  - Used by various algorithms (including ID3 and C4.5)
  - Based on the work of Claude Shannon (Information Gain)
  - The measure of the information is calculated on the basis of entropy

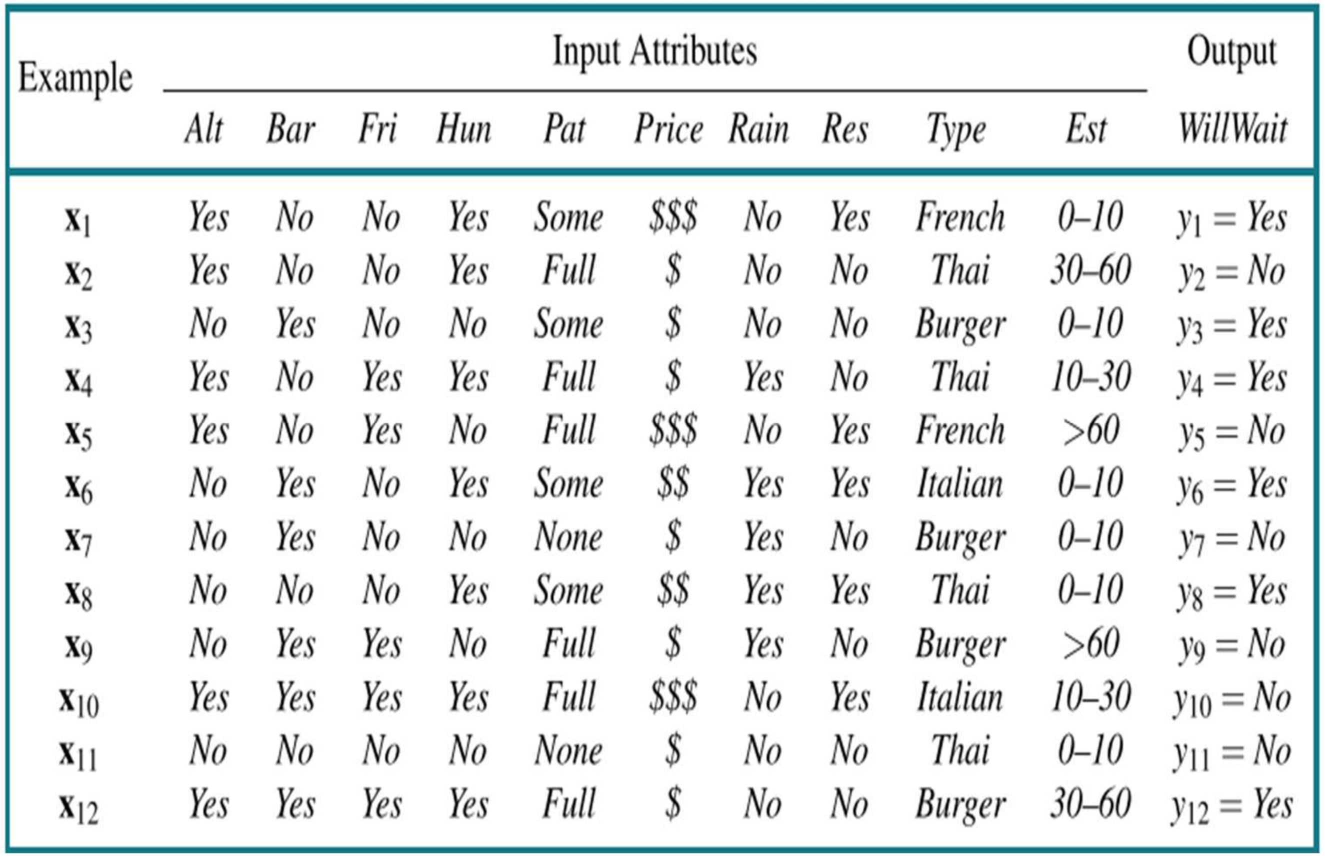
- If data is divided into classes according to the fractions {p1, p2, p3, ..., pm}, then **entropy** is measured as the information needed to classify any arbitrary item or record:

$$E(p_1, p_2, \ldots, p_m) = -\sum_{i=1}^{m} p_i \log_2(p_i)$$

- **Gini index** is another statistical measure used in various implementations of tree induction algorithms. It assumes that all attributes are continuous values.

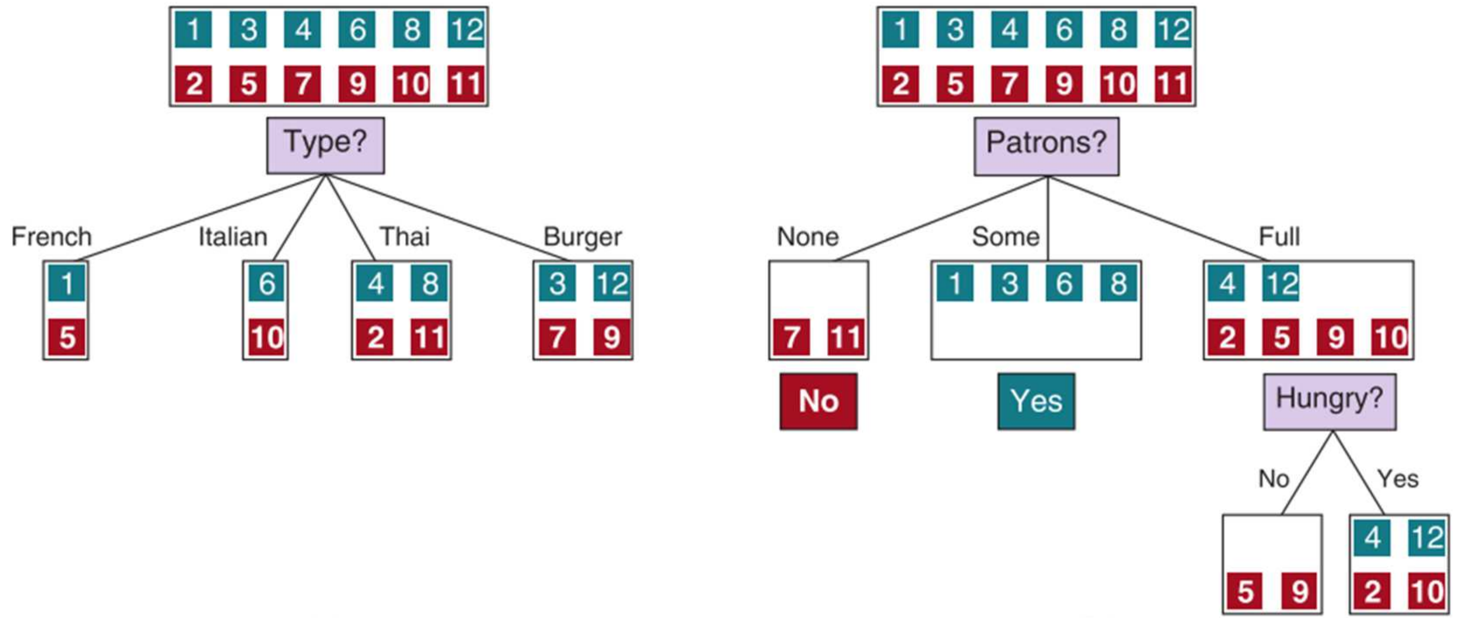
Restaurant: Training data set

1. Alternate: whether there is a suitable alternative restaurant nearby.
2. Bar: whether the restaurant has a comfortable bar area to wait in.
3. Fri/Sat: true on Fridays and Saturdays.
4. Hungry: are we hungry right now?
5. Patrons: how many people are in the restaurant (None, Some, and Full).
6. Price: the price range ($, $$, $$$).
7. Raining: whether it is raining outside.
8. Reservation: whether we made a reservation.
9. Type: the kind of restaurant (French, Italian, Thai, or burger).
10. WaitEstimate: host's wait estimate: 0–10, 10–30, 30–60, or >60 minutes.



Example	Input Attributes										Output
	Alt	*Bar*	*Fri*	*Hun*	*Pat*	*Price*	*Rain*	*Res*	*Type*	*Est*	*WillWait*
$\mathbf{x}_1$	*Yes*	*No*	*No*	*Yes*	*Some*	*\$\$\$*	*No*	*Yes*	*French*	*0–10*	$y_1 = Yes$
$\mathbf{x}_2$	*Yes*	*No*	*No*	*Yes*	*Full*	*\$*	*No*	*No*	*Thai*	*30–60*	$y_2 = No$
$\mathbf{x}_3$	*No*	*Yes*	*No*	*No*	*Some*	*\$*	*No*	*No*	*Burger*	*0–10*	$y_3 = Yes$
$\mathbf{x}_4$	*Yes*	*No*	*Yes*	*Yes*	*Full*	*\$*	*Yes*	*No*	*Thai*	*10–30*	$y_4 = Yes$
$\mathbf{x}_5$	*Yes*	*No*	*Yes*	*No*	*Full*	*\$\$\$*	*No*	*Yes*	*French*	*>60*	$y_5 = No$
$\mathbf{x}_6$	*No*	*Yes*	*No*	*Yes*	*Some*	*\$\$*	*Yes*	*Yes*	*Italian*	*0–10*	$y_6 = Yes$
$\mathbf{x}_7$	*No*	*Yes*	*No*	*No*	*None*	*\$*	*Yes*	*No*	*Burger*	*0–10*	$y_7 = No$
$\mathbf{x}_8$	*No*	*No*	*No*	*Yes*	*Some*	*\$\$*	*Yes*	*Yes*	*Thai*	*0–10*	$y_8 = Yes$
$\mathbf{x}_9$	*No*	*Yes*	*Yes*	*No*	*Full*	*\$*	*Yes*	*No*	*Burger*	*>60*	$y_9 = No$
$\mathbf{x}_{10}$	*Yes*	*Yes*	*Yes*	*Yes*	*Full*	*\$\$\$*	*No*	*Yes*	*Italian*	*10–30*	$y_{10} = No$
$\mathbf{x}_{11}$	*No*	*No*	*No*	*No*	*None*	*\$*	*No*	*No*	*Thai*	*0–10*	$y_{11} = No$
$\mathbf{x}_{12}$	*Yes*	*Yes*	*Yes*	*Yes*	*Full*	*\$*	*No*	*No*	*Burger*	*30–60*	$y_{12} = Yes$

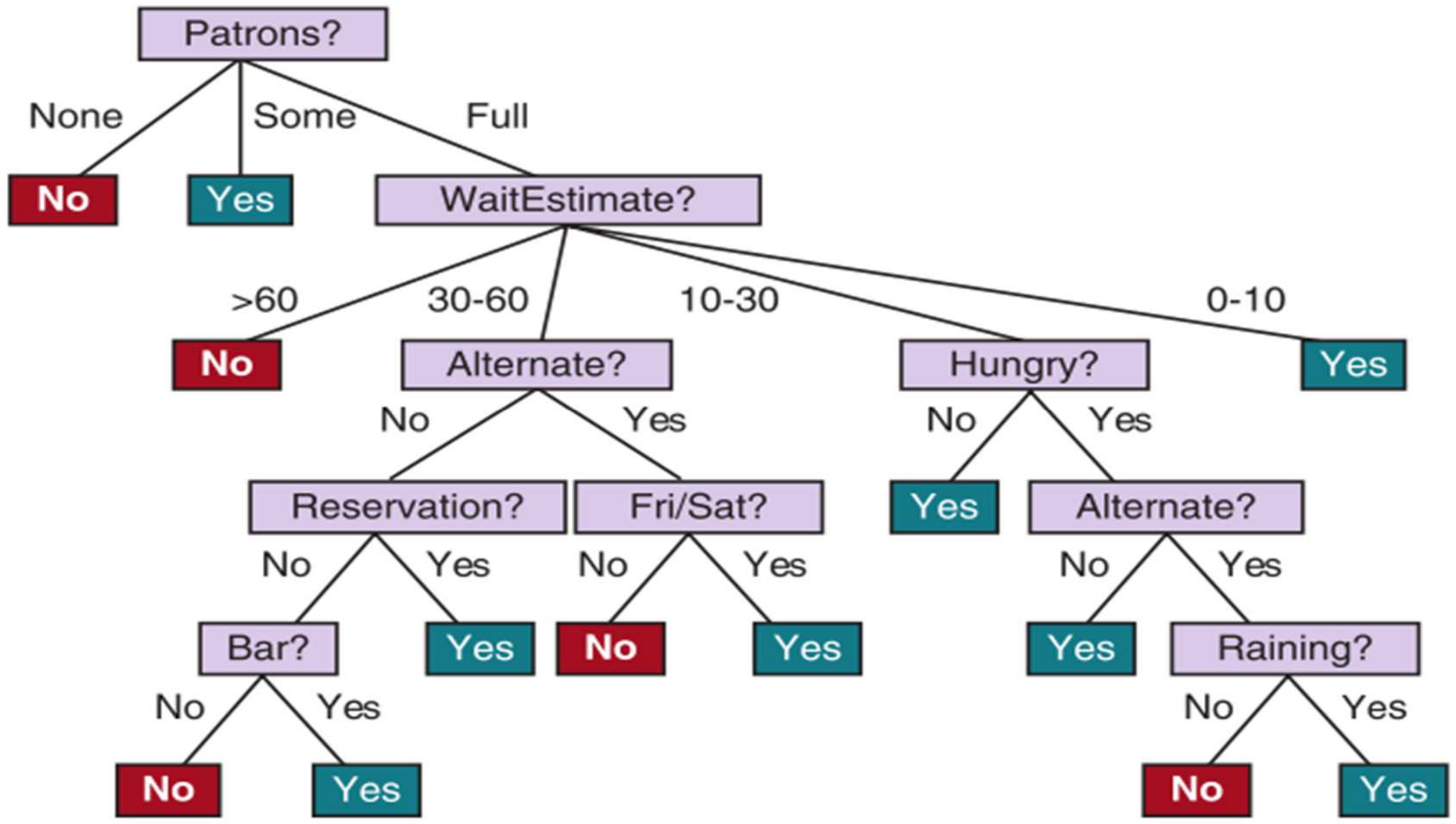
Restaurant: attribute selection



An attribute is selected at each node

- At the root of the tree, the figure compares how the examples are distributed against **type** and **patrons**
- Splitting on **Patrons** does a good job of separating positive and negative examples
- After splitting on Patrons, **Hungry** is a fairly good second test.

Restaurant: decision tree model

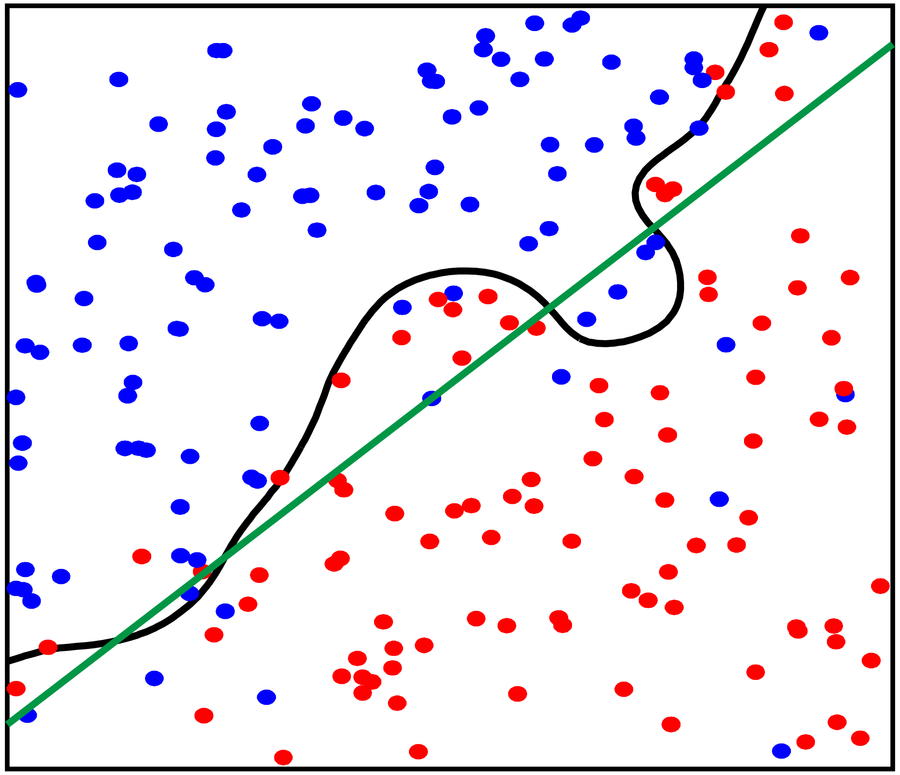


Overfitting

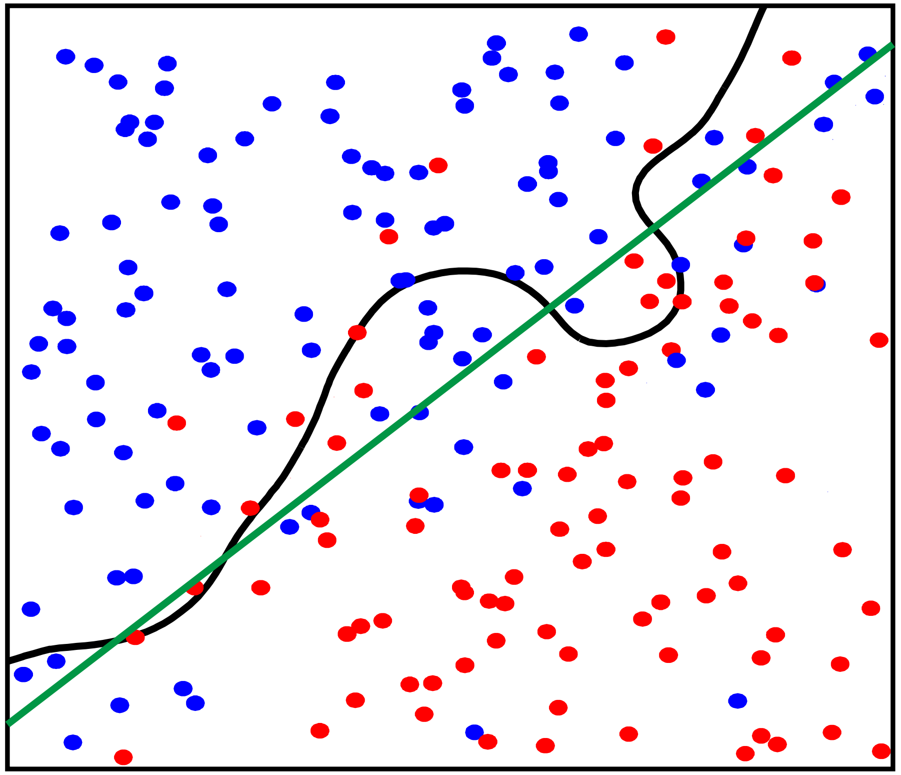
- Typical problem of the classification functionality to avoid, as it causes
    - Reduction of the model accuracy
    - Generates more branches that will reflect the anomalies caused by noise and outliers
- The fact that the tree becomes more complex will reduce the speed of the model in classification

Overfitting example

Training data set



Test data set



Pruning the tree

- In order to eliminate the overfitting in decision trees, two approaches can be followed
  - Pre-pruning - avoids building branches when the precision gain in the training data set is below a certain threshold
  - Post-pruning - after the tree is fully induced, the branches with lower precision gain are removed

- Posterior pruning allows several pruning simulations to be performed and the best pruned tree to be selected.
- Early pruning is faster, because it is performed in the construction of the tree, but generates worse models

Automatic classification Summary

- One of the most used techniques in machine learning.
- Challenges
  - Improving scalability for data sets with hundreds of attributes and millions of records
  - Reducing the classification time

- Reasons for using decision trees
  - Speed in the construction of the models in relation to other techniques
  - The model is easy to convert to an understandable set of rules
  - Rules can be easily transformed into SQL language
  - The accuracy of the classification is comparable to other techniques

Clustering

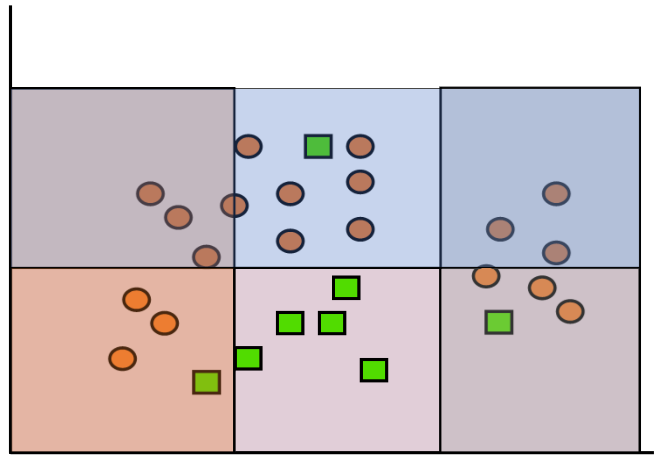
Cluster analysis

- A segment or cluster is a set of similar or related objects
  - Objects in the same segment share features with each other
  - Objects from different segments have distinctive features that make them dissimilar

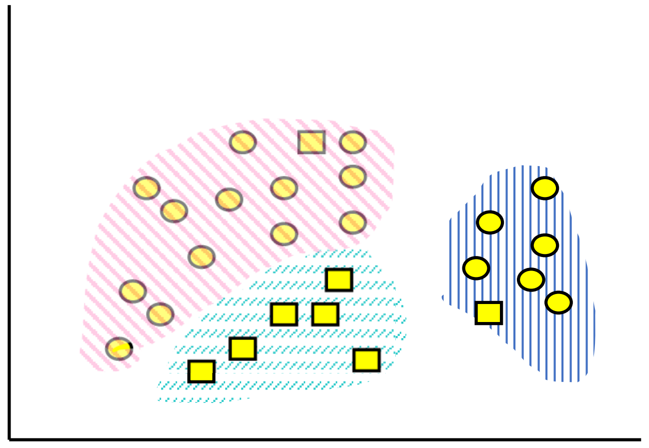
- Segmentation, known as clustering or cluster analysis, aims to group similar objects according to the similarities found between their attributes
  - Used as a primary functionality of data mining, e.g., to organize customers into segments
  - It can be used as a preprocessing technique for other algorithms, e. g. discretize continuous attributes in the induction of classification trees.

Classification versus clustering

- **Classification**
  - Supervised learning
  - It builds a model that assigns a class to an unclassified record from a list of predefined values for that class



- **Clustering**
  - Unsupervised learning
  - The classes are not predefined
  - Can be seen as natural classification, where classes are not predefined
  - Classes are generated from the data



Clustering quality

- A good segmentation method should produce high quality clusters
  - high similarity between the elements of the same cluster
  - low similarity between elements of different clusters

- The similarity between two items can be measured as a distance function, e. g. dist(i,j), were different weights can be associated to each attribute or variable
  - In most cases, it is not easy to define whether a certain degree of similarity is good enough.

- Another measure of quality is the ability to uncover hidden patterns in the data.
- Benchmaking between different segmentation methods can be used to assess quality.

Clustering requirements for method selection

- **Scalability** (number of records/items)
- Ability to handle different **types of attributes**
- Ability to handle **dynamic data**
- Discovery of **arbitrarily shaped** clusters
- Ability to **handle noise** and outliers
- Insensitive to the entry order of attributes
- **High dimensionality** (number of attributes)
- Incorporate user-specific constraints
- Ease of understanding and usability of results

Distance function

- In the simplest case of a numerical attribute A, the distance can be the function: Dist(X,Y) = A(X) - A(Y)
- If we consider that each item has n numerical attributes, the Euclidean distance can be used

$$Dist(X,Y) = \sqrt{(X_1 - Y_1)^2 + (X_2 - Y_2)^2 + \cdots + (X_n - Y_n)^2}$$

- Nominal attributes ➔ distance function that assigns the value 1 if the attribute is different; value 0 if the attribute is the same
- The importance of each attribute is usually different ➔ so, define different weights for each attribute.

Partitioning method

- It is one of many existing approaches for clustering.
- Partitioning aims to partition a set of n objects in a set of k clusters in order to obtain the minimum of the **sum of the squares of the distance of the p elements of the same C_i cluster to its center m_i**
- Given the value of k, it is necessary to find a set of k clusters that optimizes the partitioning criterion adopted
- To obtain the overall optimum it is necessary to thoroughly compare all possible combinations. Among others, K-Means and K-Medoids heuristic methods are used:
- In **K-Means**, each cluster is represented by its center obtained through an average function (mean) of the attributes of the elements that make up the cluster
- No **k-medoids** or PAM - Partition Around Medoids - each cluster is represented by one of the cluster objects

K-Means clustering algorithm

The K-Means method requires that the number of clusters to be generated (k) be provided in the start of the process.

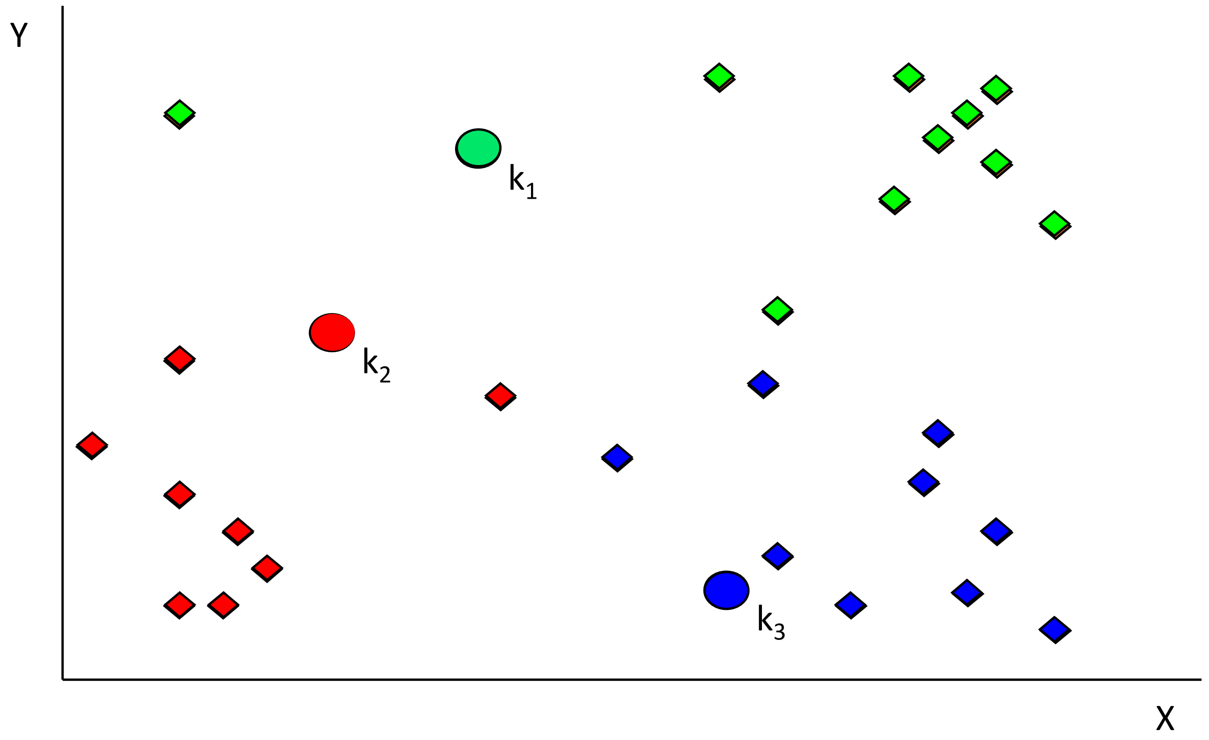
Given the value of k, the method works as follows:

- Post k random cluster centers
- Affect each object to the cluster whose center is closest to it
- Move the center of the cluster to its midpoint, calculated according to the objects that constitute it
- Repeat steps 2 and 3 as long as there are objects closer to a cluster other than your own

k-Means algorithm example (1, 2)

1) Launch k=3
random centres (seeds)

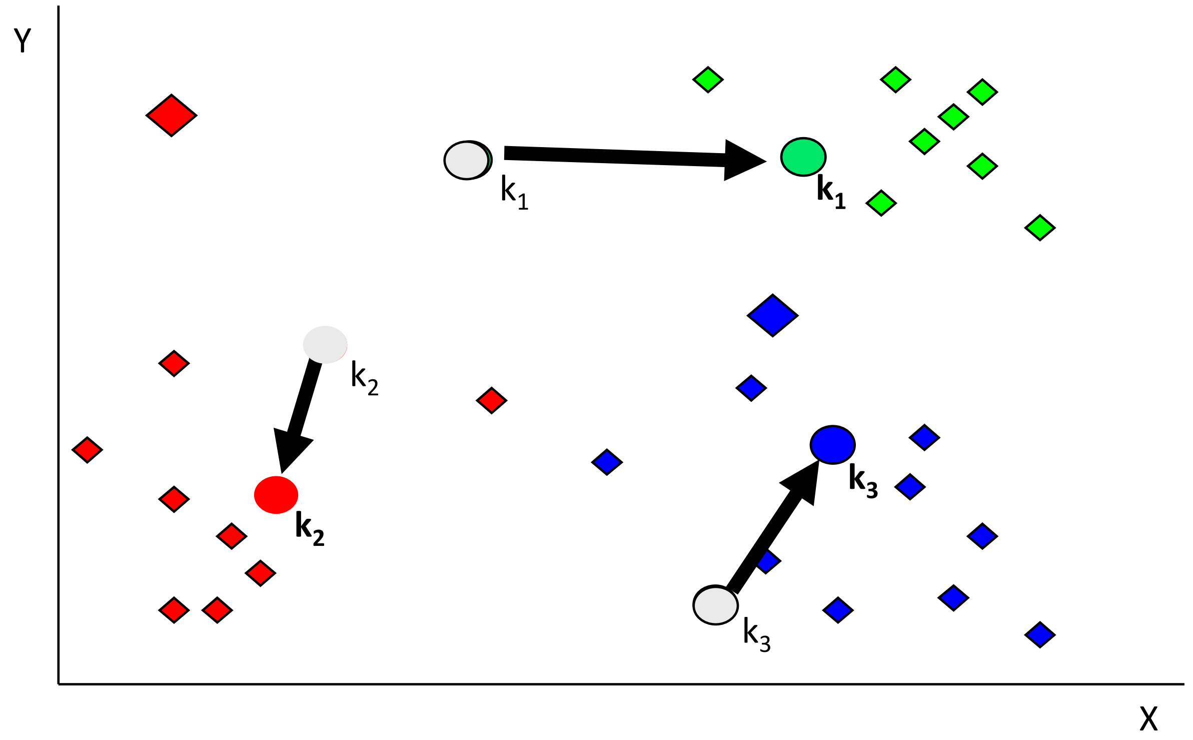
2) Attach each item
to the nearest seed



k-Means algorithm example (3, 22)

3) Move the center of the
    cluster to the center of its items

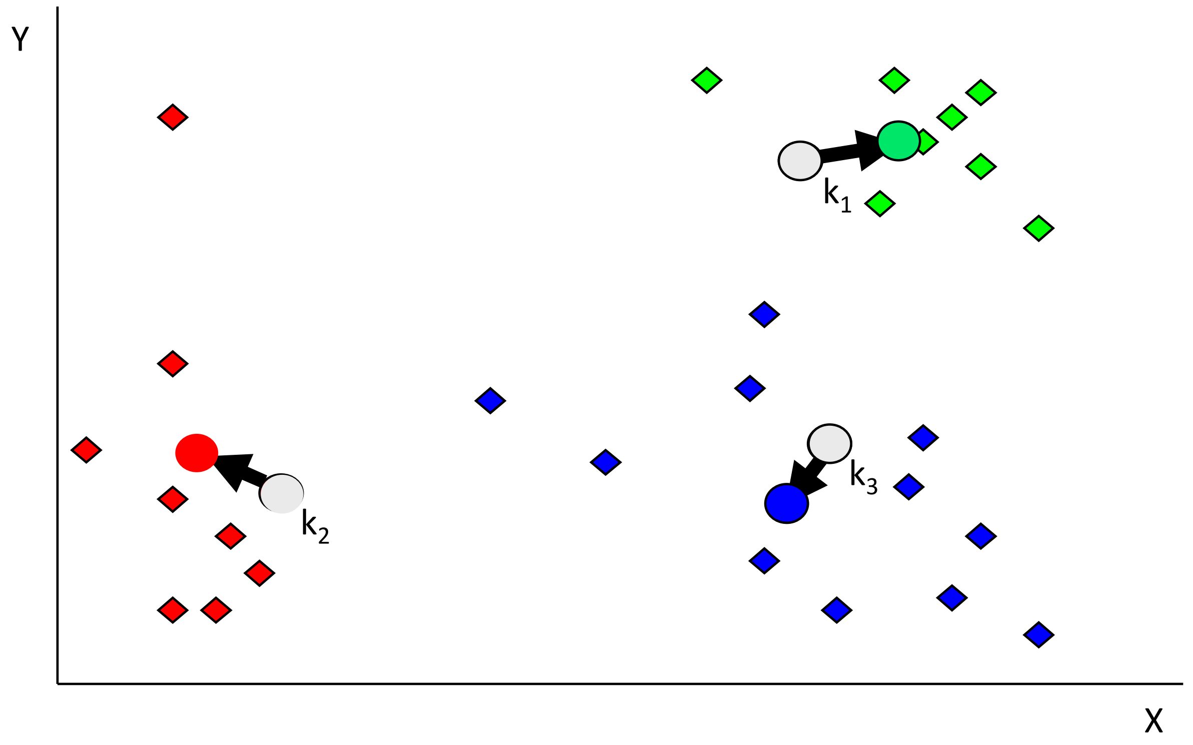
2*) Re-affect the points
    to the nearest cluster center



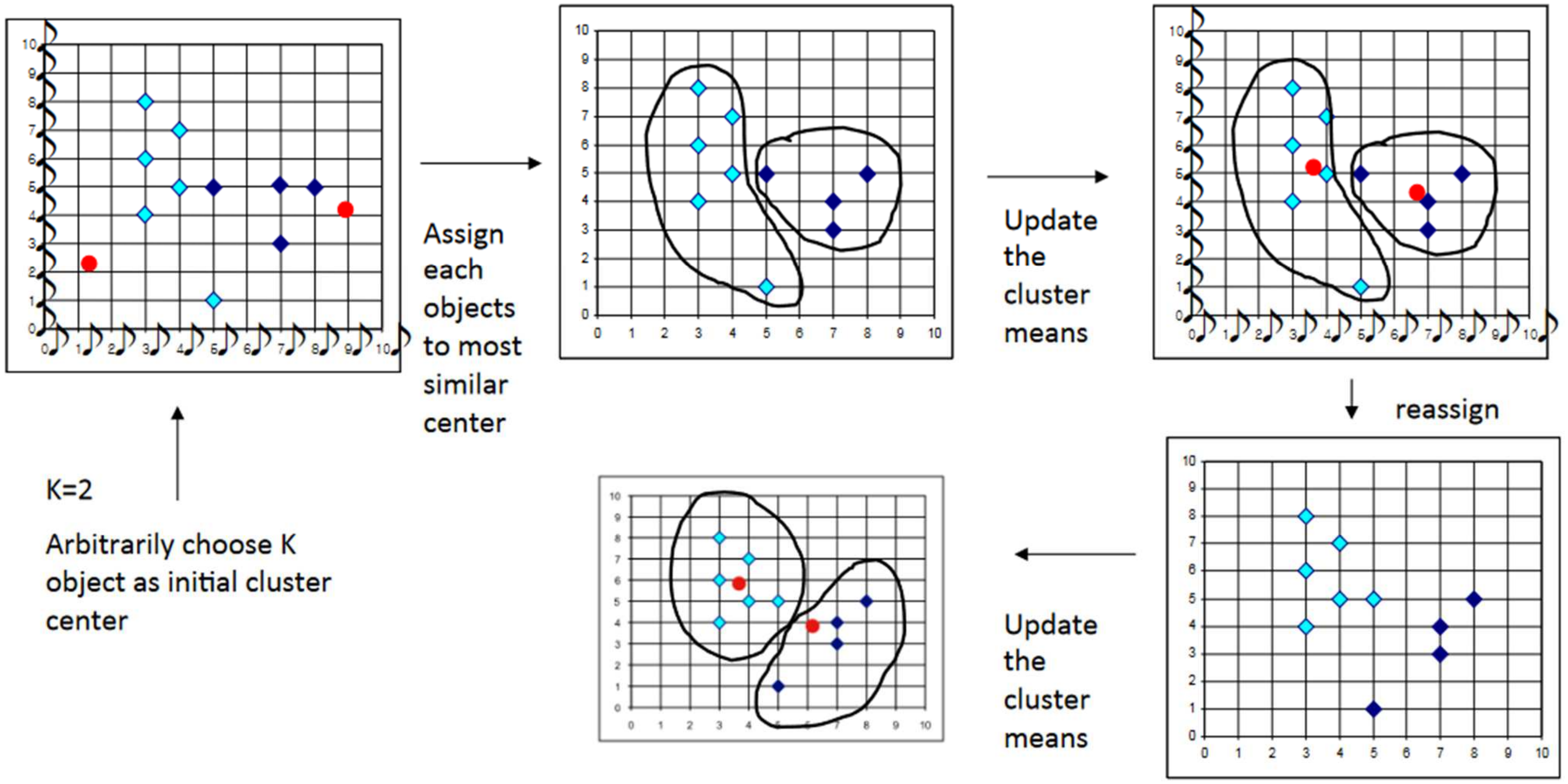
k-Means algorithm example (32,4)

3*) Move the center of the cluster
to the midpoint of your items

4) If there are no reallocations,
end iteration,
else go back to step 2*



The K-Means Clustering Method


K=2
Arbitrarily choose K object as initial cluster center
Assign each objects to most similar center
Update the cluster means
reassign
Update the cluster means

What Is the problem of the K-Means?

- The k-means algorithm is sensitive to outliers
  - Since an object with an extremely large value may substantially distort the distribution of the data.

- Instead of mean, use medians of each cluster
  - Mean of 1, 3, 5, 7, 9 is **5**
  - Mean of 1, 3, 5, 7, 1009 is **205**
  - Median of 1, 3, 5, 7, 1009 is **5**
  - Median advantage: not affected by extreme values

- **K-Medoids** do not use mean value of the object in a cluster as a reference point
  - Instead, it uses the medoid of each cluster
  - The medoid is the most centrally located item in a cluster

K-means summary

Good points

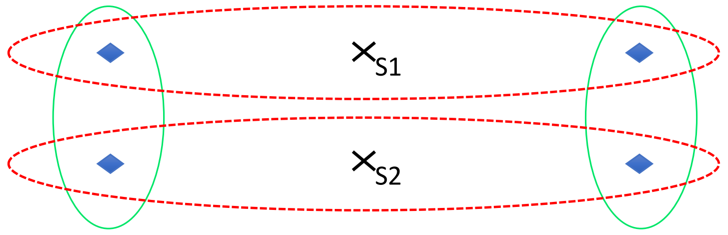
- Simple and easy to understand
- Good efficiency
- Items are automatically allocated to clusters

More examples of clustering algorithms

https://www.kdnuggets.com/2018/06/5-clustering-algorithms-data-scientists-need-know.html

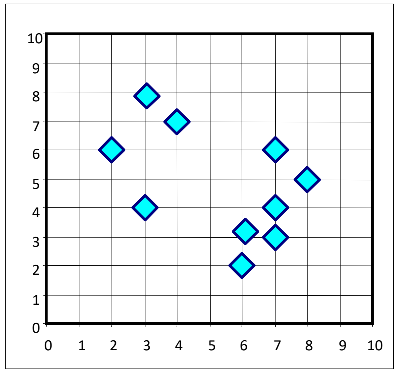
Bad points

- Applicable only to numeric attributes, where you can define the average
- It is necessary to specify k
- Influenced by noise and outliers
- All items are forced to belong to a cluster
- Not applicable when the clusters are non-convex-shaped
- Local search problem

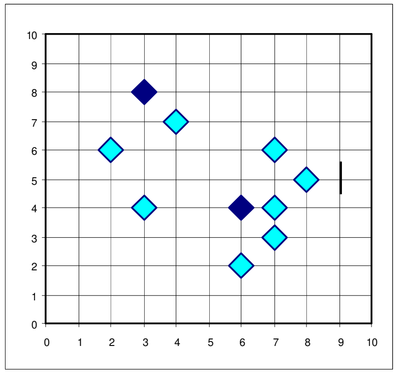


PAM: A Typical K-Medoids Algorithm

K=2

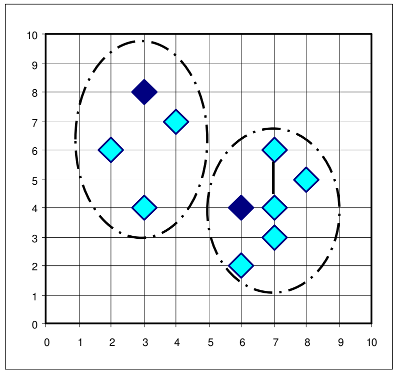


Arbitrary choose k object as initial medoids

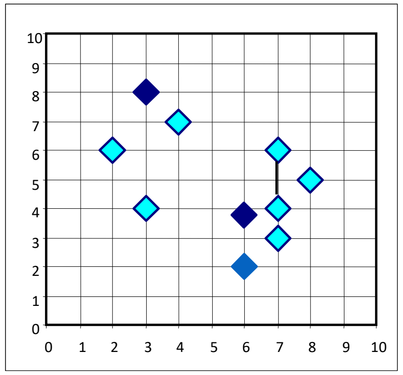


Assign each remaining object to nearest medoids

Total Cost = 20

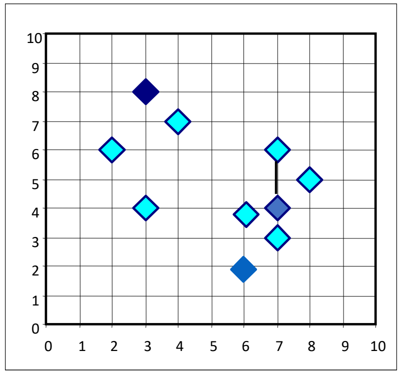


Randomly select a non medoid object,O_{ramdom}



Compute total cost of swapping

Total Cost = 18



Swapping O and O_{ramdom}

If quality is improved.

Do loop

Until no change

The K-Medoid Clustering Method

- K-Medoids Clustering: Find representative objects (medoids) in clusters
  - PAM (Partitioning Around Medoids, Kaufmann & Rousseeuw 1987)
    - Starts from an initial set of medoids and iteratively replaces one of the medoids by one of the non-medoids if it improves the total distance of the resulting clustering
    - PAM works effectively for small data sets, but does not scale well for large data sets (due to the computational complexity)
- Efficiency improvement on PAM
  - CLARA (Kaufmann & Rousseeuw, 1990): PAM on samples
  - CLARANS (Ng & Han, 1994): Randomized re-sampling

Association rules

Association rules

Goal	find patterns, associations or between items or objects
Typical source	Transactional databases or repositories with records of transactions or events
Frequent pattern	set of items or sequence that occurs the minimum value stipulated
Motivation	E.g. Identify what types of DNA are most sensitive to a new drug.

Itemset

- Be I a set of items {I1, I2, I3,..., Im}; be D a transaction database where each T transaction is a set of items such that T⊆I
  - So, if A is a set of items, transaction T is said to contain A if and only if A⊆T
  - An association rule is an implication **A⇒B, where A⊆I, B⊆I and A∩B=∅**

- A set of items with k elements is abbreviated to k-itemset

- Frequency of occurrence of a set of items
  - Corresponds to the number of transactions containing this set of items
  - Can be abbreviated by frequency, support value or count
  - A set of items is often said to satisfy the minimum support value.

Association rule support and confidence.

- A rule A⇒B is said to be valid in a set of transactions D with support S and confidence C

- Support
  - Corresponds to the % transactions in D that contain both sets of items A and B
  - Support (A⇒B) = Support (A∪B)
    = probability P(A∪B) = nr_transactions(A∪B) / total_transactions()

- Confidence
  - Corresponds to the % transactions in D containing B of total transactions A
  - Confidence(A⇒B) = conditional probability P(B|A) = support(AUC)/support(A) =
    = P(B|A) = nr_transactions(A∪B) / nr_transactions(A)

- A rule of association is said to be strong if it meets the minimum values of support and confidence.

Association rule example

Transaction ID	Sold items
10	A, B, C
20	A, C
30	A, D
40	B, E, F

Frequent pattern	Support
{A}	75%
{B}	50%
{C}	50%
{A, C}	50%

Rule A => C
Support (A => C) = nr_transactions(AUC) / total_transactions() = 50%
Confidence (A => C) = nr_transactions(AUC) / nr_transactions(A) = 66,7%

Rule C => A
Support (C => A) = nr_transactions(AUC) / total_transactions() = 50%
Confidence (C => A) = nr_transactions(AUC) / nr_transactions(C) = 100%

Association rules mining

- Very simple principle consisting of the sequence of **two steps**:
  - Find all frequent item sets with support > *minimum support*
  - Generate the association rules from the selected sets of items that satisfy the *minimum support* and *confidence*.

- The combination explosion is a challenge, due to the high number of item sets
  - A long set of items contains many smaller sets of items
  - E. g., a frequent set of items with 100 elements may contain ....

$$\binom{100}{1} + \binom{100}{2} + \ldots + \binom{100}{100} = 2^{100} - 1 \approx 1.27 \times 10^{30}$$

Apriori algorithm

- Any subset of a set of frequent items will also have to be frequent
  - E. g., if the set {beer, diapers, peanuts} is frequent,
    then {beer, diapers} will also have to be frequent
  - all transactions that contain {beer, diapers, peanuts} also contain {beer, diapers}.

- Principle of pruning
  - **If a set of items is not frequent their supersets are also not frequent**

- The Apriori algorithm implements a candidate generation and testing approach based on the following method:
  - Generate sets of k+1 size items from the frequent sets of k-items
  - Test the generated sets with the existing data in the database.

Candidates' generation

- Generation performed in two steps
  - Generation - join the K size sets
  - Pruning - pruning is done by eliminating the infrequent sets
- Test candidate sets of size k+1 in the database

- Example of candidate generation:
  - Generation - L3={abc, abd, acd, ace, bcd}, combination L3*L3
  - From the combination of abc and abd you get **abcd**,
  - From the combination of acd and ace you get **acde**

- Pruning - **acde** is removed because ade is not in L3, being
  - The candidate set C4={abcd} is obtained
  - C4 which will have to be tested with the database records before moving to L4

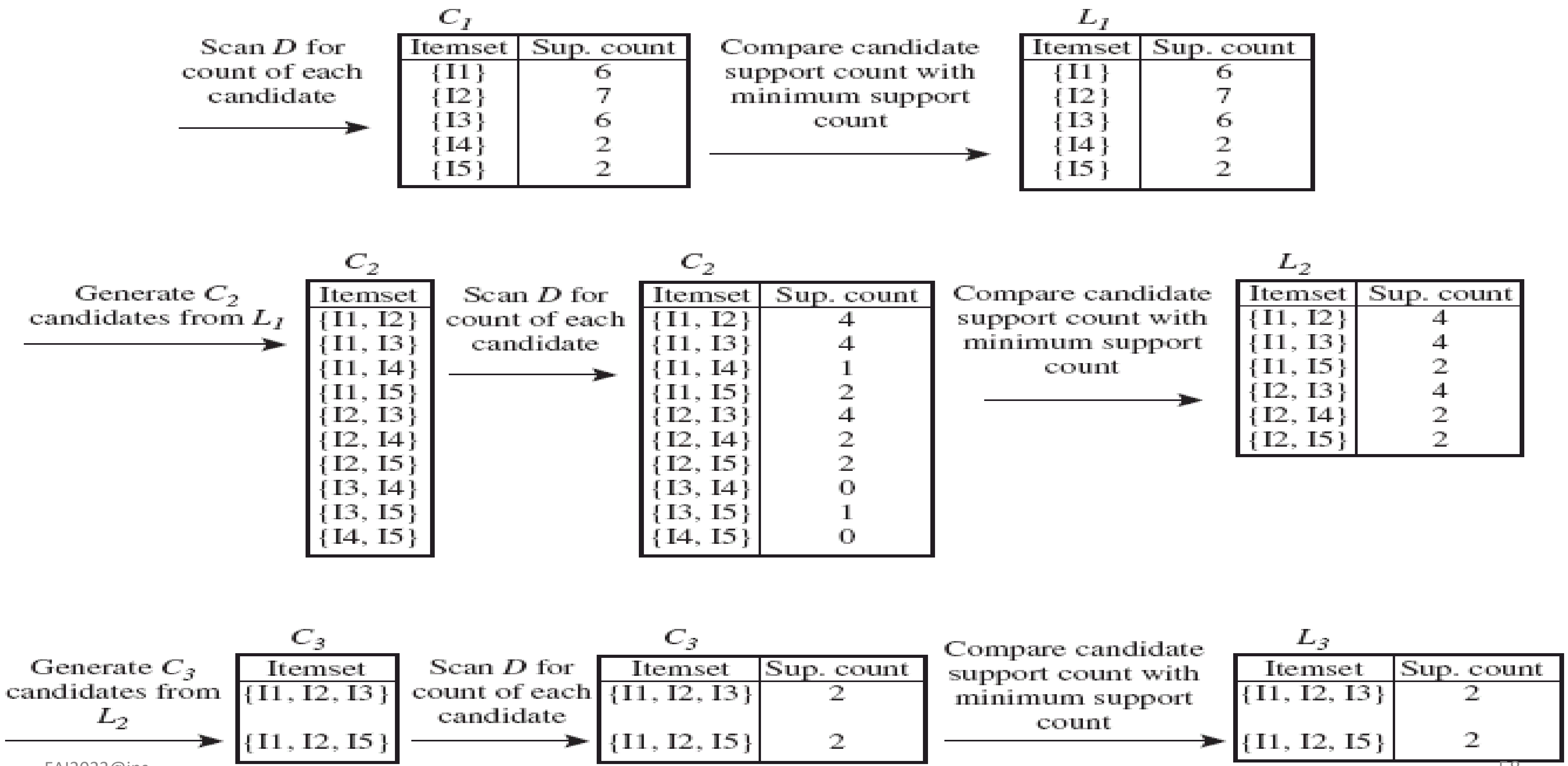
Apriori algorithm demo

TID	List of item_IDs
T100	Beer, Crisps, Milk
T200	Crisps, Bread
T300	Crisps, Nappies
T400	Beer, Crisps, Bread
T500	Beer, Nappies
T600	Crisps, Nappies
T700	Beer, Nappies
T800	Beer, Crisps, Nappies, Milk
T900	Beer, Crisps, Nappies

Items found in the transactions

ID	Item
I1	Beer
I2	Crisps
I3	Nappies
I4	Bread
I5	Milk

Apriori algorithm demo (2)



Scan D for count of each candidate $\rightarrow$

C_1

Itemset	Sup. count
{I1}	6
{I2}	7
{I3}	6
{I4}	2
{I5}	2

Compare candidate support count with minimum support count $\rightarrow$

L_1

Itemset	Sup. count
{I1}	6
{I2}	7
{I3}	6
{I4}	2
{I5}	2

Generate C_2 candidates from L_1 $\rightarrow$

C_2

Itemset
{I1, I2}
{I1, I3}
{I1, I4}
{I1, I5}
{I2, I3}
{I2, I4}
{I2, I5}
{I3, I4}
{I3, I5}
{I4, I5}

Scan D for count of each candidate $\rightarrow$

C_2

Itemset	Sup. count
{I1, I2}	4
{I1, I3}	4
{I1, I4}	1
{I1, I5}	2
{I2, I3}	4
{I2, I4}	2
{I2, I5}	2
{I3, I4}	0
{I3, I5}	1
{I4, I5}	0

Compare candidate support count with minimum support count $\rightarrow$

L_2

Itemset	Sup. count
{I1, I2}	4
{I1, I3}	4
{I1, I5}	2
{I2, I3}	4
{I2, I4}	2
{I2, I5}	2

Generate C_3 candidates from L_2 $\rightarrow$

C_3

Itemset
{I1, I2, I3}
{I1, I2, I5}

Scan D for count of each candidate $\rightarrow$

C_3

Itemset	Sup. count
{I1, I2, I3}	2
{I1, I2, I5}	2

Compare candidate support count with minimum support count $\rightarrow$

L_3

Itemset	Sup. count
{I1, I2, I3}	2
{I1, I2, I5}	2

Challenges of rule mining

- Challenges / problems
  - Generation of a high number of candidates
  - Carrying out extensive database queries
  - Tedious count of number of support transactions
  - Full access to transaction records (full-scan) has high computational costs

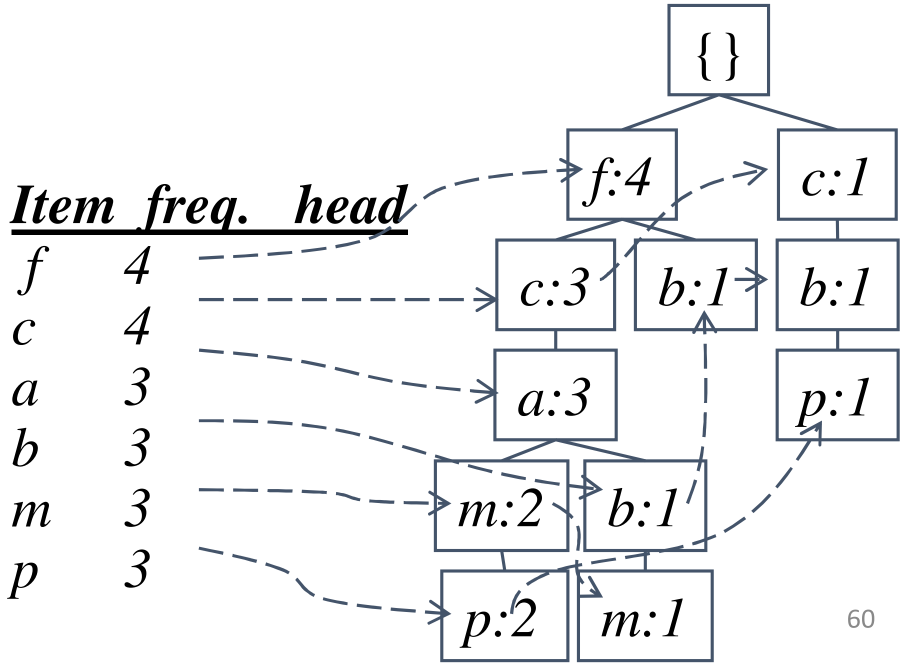
- Improvements to the Apriori algorithm
  - Reduction in the number of database accesses
  - Reduction in the number of candidates
  - Improvement of the process of counting support to the various candidates.

FT-Growth Algorithm

Improves efficiency by generating candidates. Make longer patterns from short patterns using frequent local items. Steps for building the FP-tree:

- Counting of frequent size one items (e.g. support > 10%)
- Sorts the frequent items in descending order
- Go through the database for the 2nd time and build the FP-tree

T_ID	Itens vendidos	Itens frequentes (ordenados)
100	{*f, a, c, d, g, i, m, p*}	{*f, c, a, m, p*}
200	{*a, b, c, f, l, m, o*}	{*f, c, a, b, m*}
300	{*b, f, h, j, o*}	{*f, b*}
400	{*b, c, k, s, p*}	{*c, b, p*}
500	{*a, f, c, e, l, p, m, n*}	{*f, c, a, m, p*}



Association performance metrics

For the association rule A⇒C, we have the following metrics:

- **Support**: relative frequency of A and C transactions.

  support(A⇒C) = support(A∪C)

- **Confidence**: confidence of the association rule.

  Confidence(A⇒C) = support(A∪C)/support(A)

- **Lift**: shows the interest of the rule.

  Interest (A⇒C) = confidence (A⇒C)/support(C)

- **Leverage**: values the frequency over the interest.

  Leverage(A⇒C) = support (A⇒C) - support(A)* support(C)

- **Conviction**: measures the effect of the right side not being true:

  Conviction(A⇒C) = support(A) * support(C) / support(A∪C)

Thank you!