Business intelligence
Systèmes décisionnels
Informatique décisionnelle

Introduction to data mining

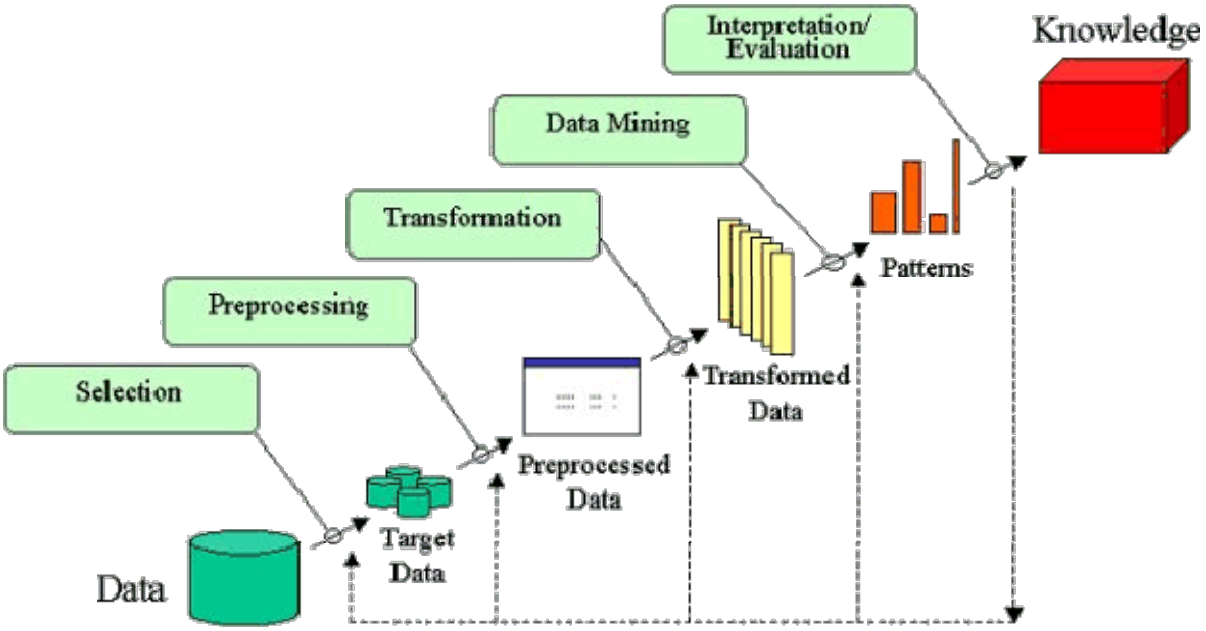
KDD Process [14]

The term Knowledge Discovery in Databases, or KDD for short, refers to the broad process of finding knowledge in data, and emphasizes the "high-level" application of particular data mining methods. It is of interest to researchers in machine learning, pattern recognition, databases, statistics, artificial intelligence, knowledge acquisition for expert systems, and data visualization.

The unifying goal of the KDD process is to extract knowledge from data in the context of large databases.

An Outline of the Steps of the KDD Process [14]

The overall process of finding and interpreting patterns from data involves the repeated application of the following steps:



Source [14]

1. Developing an understanding of
    - the application domain
    - the relevant prior knowledge
    - the goals of the end-user
2. Creating a target data set: selecting a data set, or focusing on a subset of variables, or data samples, on which discovery is to be performed.
3. Data cleaning and preprocessing.
    - Handling Missing Values: Missing values in the dataset can be handled by replacing them with the mean, median, or mode of the corresponding feature or by removing the entire record.
    - Dealing with Outliers: Outliers in the dataset can be detected using various statistical techniques such as z-score analysis, boxplots, and scatterplots. Outliers can be removed from the dataset or replaced with the mean, median, or mode of the corresponding feature.
    - Removal of noise or outliers.

4. Data transformation
   Data transformation help to improve the performance of data mining algorithms, by reducing the dimensionality of the data, and by scaling the data to a common range of values.
    - Finding useful features to represent the data depending on the goal of the task. This can be done using various techniques, such as ***correlation*** analysis. Correlation analysis involves identifying the correlation between the features in the dataset. Features that are highly correlated with each other can be removed, as they do not provide additional information.
    - Using dimensionality reduction or transformation methods to reduce the effective number of variables under consideration or to find invariant representations for the data. A technique such, ***Principal component analysis (PCA)*** identifies the most important features in the dataset and removes the redundant ones.

5. Choosing the data mining task.
    - Deciding whether the goal of the KDD process is classification, regression, clustering, etc.
6. Choosing the data mining algorithm(s).
    - Selecting method(s) to be used for searching for patterns in the data.
    - Deciding which models and parameters may be appropriate.
    - Matching a particular data mining method with the overall criteria of the KDD process.
    - For a classification problem, there are several algorithms available, such as decision trees, support vector machines, and neural networks.

7. Data mining.
    - Searching for patterns of interest in a particular representational form or a set of such representations as classification rules or trees, regression, clustering, and so forth.
8. Interpreting mined patterns.
9. Consolidating discovered knowledge.

The terms knowledge discovery and data mining are distinct.

KDD refers to the overall process of discovering useful knowledge from data. It involves the evaluation and possibly interpretation of the patterns to make the decision of what qualifies as knowledge. It also includes the choice of encoding schemes, preprocessing, sampling, and projections of the data prior to the data mining step.
Data mining refers to the application of algorithms for extracting patterns from data without the additional steps of the KDD process.

The Primary Tasks of Data Mining [14]

The two "high-level" primary goals of data mining, in practice, are prediction and description.

- **Description** focuses on finding human-interpretable patterns describing the data.
- **Prediction** involves using some variables or fields in the database to predict unknown or future values of other variables of interest.

The relative importance of prediction and description for particular data mining applications can vary considerably. However, in the context of KDD, description tends to be more important than prediction. This is in contrast to pattern recognition and machine learning applications (such as speech recognition) where prediction is often the primary goal of the KDD process.

1 Descriptive Data Mining:

This category of data mining is concerned with finding patterns and relationships in the data that can provide insight into the underlying structure of the data. Descriptive data mining is often used to summarize or explore the data, and it can be used to answer questions such as: What are the most common patterns or relationships in the data? Are there any clusters or groups of data points that share common characteristics? What are the outliers in the data, and what do they represent?

Some common techniques used in descriptive data mining include **Cluster** analysis, **Association rule** mining, Visualization

2 Predictive Data Mining: This category of data mining is concerned with developing models that can predict future behavior or outcomes based on historical data. Predictive data mining is often used for classification or regression tasks, and it can be used to answer questions such as: What is the likelihood that a customer will churn? What is the expected revenue for a new product launch? What is the probability of a loan defaulting?
Some common techniques used in predictive data mining include: classification, Regression analysis, Neural networks.

Data mining Tools and Languages:
Data mining Languages: R, SAS, Python, SQL
Data mining Tools: Oracle Data Mining, IBM SPSS Modeler, SAS Enterprise Miner, RapidMiner, Orange, WEKA, Teradata, Qlik, Apache Mahout, ..etc.

Machine learning [1]
A branch of artificial intelligence that provides algorithms enabling machines to learn patterns from historical data to then be able to make predictions on unseen data without being explicitly programmed.
For instance, if we want a computer to recognize images of cats, we don't provide it with specific instructions on what a cat looks like. Instead, we give it thousands of images of cats and let the machine learning algorithm figure out the common patterns and features that define a cat. Over time, as the algorithm processes more images, it gets better at recognizing cats, even when presented with images it has never seen before.

Machine learning can be broadly classified into two types based on the nature of the learning system and the data available: supervised learning, unsupervised learning.

Supervised learning [1]
Supervised learning is the most common type of machine learning. In this approach, the model is trained on a labeled dataset. In other words, the data is accompanied by a label that the model is trying to predict. This could be anything from a category label to a real-valued number.

The model learns a mapping between the input (features) and the output (label) during the training process. Once trained, the model can predict the output for new, unseen data.

Common examples of supervised learning algorithms include linear regression for regression problems and logistic regression, decision trees, and support vector machines for classification problems.

Unsupervised learning
Unsupervised learning, on the other hand, involves training the model on an unlabeled dataset. The model is left to find patterns and relationships in the data on its own.

This type of learning is often used for clustering and dimensionality reduction. Clustering involves grouping similar data points together, while dimensionality reduction involves reducing the number of random variables under consideration by obtaining a set of principal variables.

Common examples of unsupervised learning algorithms include k-means for clustering problems and Principal Component Analysis (PCA) for dimensionality reduction problems.

Some Applications of Machine Learning

- Recommendation systems :Companies like Netflix and Amazon use machine learning to analyze your past behavior and recommend products or movies you might like
- Voice assistants :Voice assistants like Siri, Alexa, and Google Assistant use machine learning to understand your voice commands and provide relevant responses.
- Fraud detection :Banks and credit card companies use machine learning to detect fraudulent transactions. By analyzing patterns of normal and abnormal behavior

Machine learning process [1]

Machine learning process transforms raw data into valuable insights.

Step 1: Data collection

Determine what data is necessary to build the model. Data can be collected from various sources such as databases, text files, images, audio files, or even scraped from the web.

Once collected, the data needs to be prepared for machine learning. This process involves organizing the data in a suitable format, such as a CSV file or a database, and ensuring that the data is relevant to the problem you're trying to solve.

Step 2: Data preprocessing

Data preprocessing is a crucial step in the machine learning process. It involves cleaning the data (removing duplicates, correcting errors), handling missing data (either by removing it or filling it in), and normalizing the data (scaling the data to a standard format).

Preprocessing improves the quality of your data and ensures that your machine learning model can interpret it correctly. This step can significantly improve the accuracy of your model.

Steps you might undertake during this phase

- Standardize data formats and normalize data across different sources.
- Replace incorrect or missing data.
- Enhance and augment data, possibly by using third-party data or multiplying image-based data sets if the core data set is insufficient for training.
- Add dimensions with pre-calculated amounts and aggregate information as needed.

- Remove extraneous and redundant information, known as deduplication, and remove irrelevant data.
- Reduce noise and remove ambiguity.
- Consider anonymizing personal or otherwise sensitive data.
- Select features that identify the most important dimensions and, if necessary, reduce dimensions using a variety of techniques.

Step 3: Choosing the right model
Once the data is prepared, the next step is to choose a machine learning model. There are many types of models to choose from, including linear regression, decision trees, and neural networks. The choice of model depends on the nature of your data and the problem you're trying to solve.

Factors to consider when choosing a model include the size and type of your data, the complexity of the problem, and the computational resources available.

Step 4: Training the model
After choosing a model, the next step is to train it using the prepared data. Training involves feeding the data into the model and allowing it to adjust its internal parameters to better predict the output.

During training, it's important to avoid overfitting (where the model performs well on the training data but poorly on new data) and underfitting (where the model performs poorly on both the training data and new data).

Step 5: Evaluating the model
Once the model is trained, it's important to evaluate its performance before deploying it. This involves testing the model on new data it hasn't seen during training.

Common metrics for evaluating a model's performance include accuracy (for classification problems), precision and recall (for binary classification problems), and mean squared error (for regression problems)

Step 6: Hyperparameter tuning and optimization
After evaluating the model, you may need to adjust its hyperparameters to improve its performance. This process is known as parameter tuning or hyperparameter optimization.

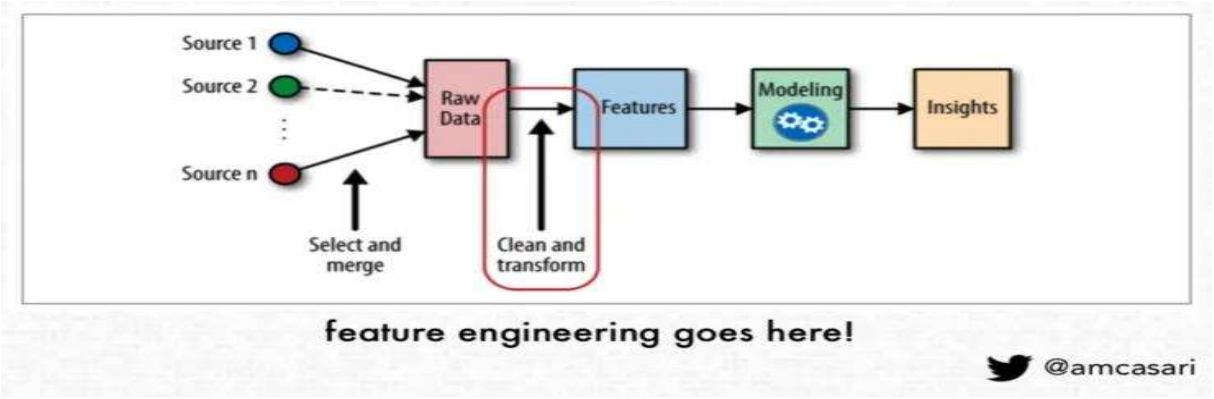
Techniques for hyperparameter tuning include grid search (where you try out different combinations of parameters) and cross validation (where you divide your

data into subsets and train your model on each subset to ensure it performs well on different data).

Step 7: Predictions and deployment
Once the model is trained and optimized, it's ready to make predictions on new data. This process involves feeding new data into the model and using the model's output for decision-making or further analysis.

Deploying the model (operationalizing the model) involves integrating it into a production environment where it can process real-world data and provide real-time insights. This process is often known as MLOps.

Feature Engineering
Feature engineering is the process of transforming raw data into features that are suitable for machine learning models. In other words, it is the process of selecting, extracting, and transforming the most relevant features from the available data to build more accurate and efficient machine learning models.



Feature engineering involves a set of techniques that enable us to create new features by combining or transforming the existing ones.

Feature transformation: is the process of converting one feature type into another, more readable form for a particular model. This consists of transforming continuous into categorical data, or vice-versa.

- **Binning**. This technique essentially transforms continuous, numerical values into categorical features
- **One-hot encoding**. This is the inverse of binning; it creates numerical features from categorical variables. One-hot encoding maps categorical features to binary representations, which are used to map the feature in a matrix or vector space. Literature often refers to this binary representation as a dummy variable.

- **Logarithm transformation** (or log transform) helps to handle skewed data

Feature extraction and selection : Feature extraction is a process that transforms the original dataset into a reduced number of features. In contrast, feature selection finds the element that gives us the most information about the problem. As an example Principal component analysis (PCA) is a common feature extraction method that combines and transforms a dataset's original features to produce new features, called principal components.

Feature scaling

Feature scaling (sometimes called feature normalization) is a standardization technique to rescale features and limit the impact of large scales on models.

- Min-max scaling. Min-max scaling rescales all values for a given feature so that they fall between specified minimum and maximum values, often 0 and 1.
- Z-score scaling. Literature also refers to this as standardization and variance scaling. z-score scaling rescales features so that they have a shared standard deviation of 1 with a mean of 0

Association analysis

Association analysis is a data mining technique used to discover interesting relationships or associations among a set of items in large datasets. The most common machine learning algorithms for association analysis problems, including **Apriori**, Eclat, FP-Growth, FPMax.

Association Mining

Transaction ID	Items
T1	Milk, Bread
T2	Bread, Diaper, Tea, Eggs
T3	Milk, Diaper, Tea, Coke
T4	Bread, Milk, Diaper, Tea
T5	Bread, Milk, Diaper, Coke

Itemset
An itemset is A collection of one or more items. Ex: {Milk, Bread, Diaper}.

Support count (σ) : Frequency of occurrence of an itemset.
Ex: σ({Milk, Bread,Diaper}) = 2
k-itemset : An itemset that contains k items
Support : Fraction of transactions that contain an itemset.
EX: s({Milk, Bread, Diaper}) = 2/5
Frequent Itemset : An itemset whose support is greater than or equal to a minsup threshold
Association Rule : An implication expression of the form X → Y, where X and Y are itemsets.
Ex: {Milk, Diaper} → {Tea}

Rule Evaluation Metrics
Given the association rule: {Milk,Diaper}⇒ {Tea}

Support (s) : Fraction of transactions that contain both X and Y
s= σ({Milk, Diaper, Tea})/|T|=2/5=0.4

Confidence (c): Measures how often items in Y appear in transactions that contain X

c= σ({Milk, Diaper, Tea})/ σ({Milk, Diaper})=2/3=0.67

Association Rule Mining Task

Given a set of transactions T, the goal of association rule mining is to find all rules having
support ≥ minsup threshold
confidence ≥ minconf threshold

Brute-force approach:
– List all possible association rules
– Compute the support and confidence for each rule
– Prune rules that fail the minsup and minconf thresholds ⇒ Computationally prohibitive!

Two-step approach:
1. Frequent Itemset Generation : Generate all itemsets whose support ≥ minsup
2. Rule Generation : Generate high confidence rules from each frequent itemset, where each rule is a binary partitioning of a frequent itemset

Given the itemset {Milk, Diaper, Tea}

Example of Rules:
{Milk,Diaper} → {Tea} (s=0.4, c=0.67)
{Milk,Tea} → {Diaper} (s=0.4, c=1.0)
{Diaper,Tea} → {Milk} (s=0.4, c=0.67)
{Tea} → {Milk,Diaper} (s=0.4, c=0.67)
{Diaper} → {Milk,Tea} (s=0.4, c=0.5)
{Milk} → {Diaper,Tea} (s=0.4, c=0.5)

All the above Rules originating from the same itemset {Milk, Diaper, Tea}and have identical support but can have different confidence

Apriori Algorithm

– Let k=1
– Generate frequent itemsets of length 1
– Repeat until no new frequent itemsets are identified

- Generate length (k+1) candidate itemsets from length k frequent itemsets
- Prune candidate itemsets containing subsets of length k that are infrequent
- Count the support of each candidate by scanning the DB Eliminate candidate

- Eliminate candidates that are infrequent, leaving only those that are frequent

Classification is learning a function that maps (classifies) a data item into one of several predefined classes. The most common machine learning algorithms for classification problems, including logistic regression, decision trees, random forests, support vector machines, k-nearest neighbours, and naive Bayes.

Classification: Decision Tree

How to Build Decision Tree for Classification – (Using Entropy and Gain)

The ID3 algorithm by J. R. Quinlan. The algorithm uses Entropy and Information Gain to build the tree. (The example is taken from https://kindsonthegenius.com/blog/how-to-build-a-decision-tree-for-classification-step-by-step-procedure-using-entropy-and-gain/)

Attributes				Classes
Outlook	Temperature	Humidity	Windy	Play Golf
Rainy	Hot	High	FALSE	No
Rainy	Hot	High	TRUE	No
Overcast	Hot	High	FALSE	Yes
Sunny	Mild	High	FALSE	Yes
Sunny	Cool	Normal	FALSE	Yes
Sunny	Cool	Normal	TRUE	No
Overcast	Cool	Normal	TRUE	Yes
Rainy	Mild	High	FALSE	No
Rainy	Cool	Normal	FALSE	Yes
Sunny	Mild	Normal	FALSE	Yes
Rainy	Mild	Normal	TRUE	Yes
Overcast	Mild	High	TRUE	Yes
Overcast	Hot	Normal	FALSE	Yes
Sunny	Mild	High	TRUE	No

Figure 1: Exercise on Decision Trees

Final Decision Ttree

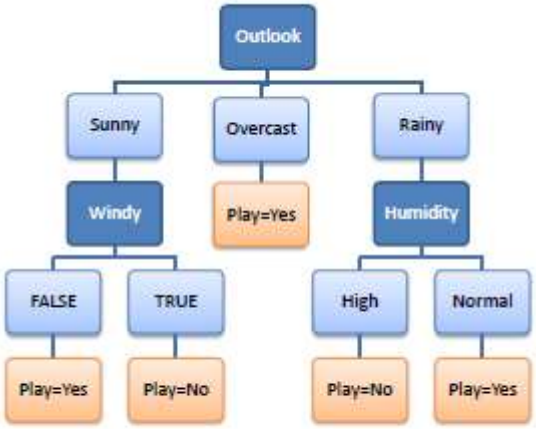
R_1: IF (Outlook=Sunny) AND (Windy=FALSE) THEN Play=Yes

R_2: IF (Outlook=Sunny) AND (Windy=TRUE) THEN Play=No

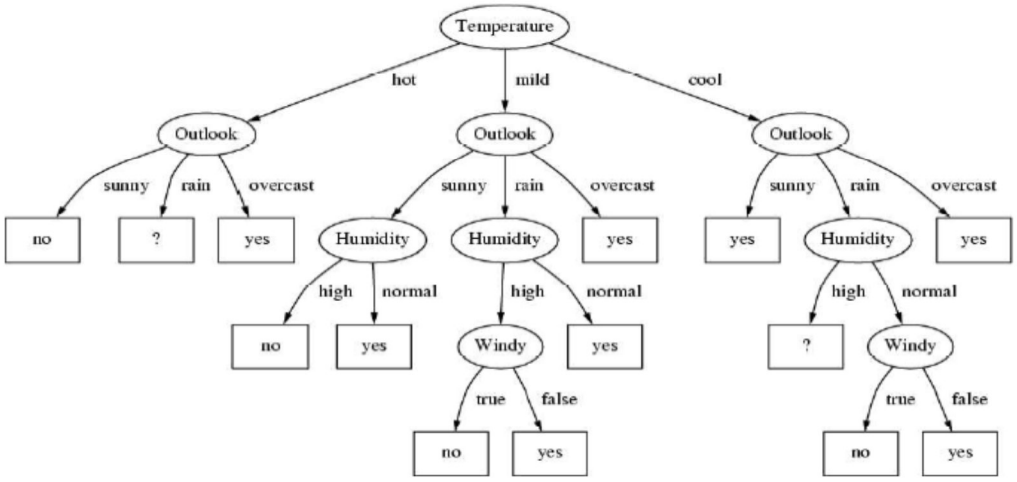
R_3: IF (Outlook=Overcast) THEN Play=Yes

R_4: IF (Outlook=Rainy) AND (Humidity=High) THEN Play=No

R_5: IF (Outlook=Rain) AND (Humidity=Normal) THEN Play=Yes



A Different Decision Tree



Step 1: Determine the Decision Column

For this example **Play Golf** column with classes Yes and No is the decision column.

To determine the rootNode we need to compute the entropy for all columns.

- S is a set of examples
- p_i is the proportion of examples in S that belong to the i-th class

$$E(S) = -p_1 \log p_1 - p_2 \log p_2 \ldots - p_n \log p_n = -\sum_{i=1}^{n} p_i \log p_i$$

To do this, we create a frequency table for the classes (**Play Golf**).

Play Golf(14)	
Yes	No
9	5

Table 2: Frequency Table

Step 2: Calculating Entropy (*Play Golf*)

$$E(PlayGolf) = E(5,9)$$

$$= -\left(\frac{9}{14}\log_2\frac{9}{14}\right) - \left(\frac{5}{14}\log_2\frac{5}{14}\right)$$

$$= -(0.357\ \log_2 0.357) - (0.643\ \log_2 0.643)$$

$$= 0.94$$

Step 3: Calculate Entropy for Other Attributes After Split

For the other four attributes, we need to calculate the entropy after each of the split.

- E(PlayGolf, Outloook)
- E(PlayGolf, Temperature)
- E(PlayGolf, Humidity)
- E(PlayGolf,Windy)

- **Problem:**
  - Entropy only computes the quality of a single (sub-)set of examples
    - corresponds to a single value
  - How can we compute the quality of the entire split?
    - corresponds to an entire attribute
- **Solution:**
  - Compute the weighted average over all sets resulting from the split
    - weighted by their size

$$I(S,A) = \sum_i \frac{|S_i|}{|S|} \cdot E(S_i)$$

To calculate E(PlayGolf, Outlook), we would use the formula below:

		PlayGolf(14)		
		Yes	No	
Outlook	Sunny	3	2	5
	Overcast	4	0	4
	Rainy	2	3	5

Table 3: Frequency Table for Outlook

$$E(PlayGolf, Outlook) = P(Sunny)E(Sunny) + P(Overcast)E(Overcast) + P(Rainy)E(Rainy)$$

$$E(PlayGolf, Outlook) = \frac{5}{14}E(3,2) + \frac{4}{14}E(4,0) + \frac{5}{14}E(2,3)$$

$E(Overcast) = E(4,0)$ $= -\left(\frac{4}{4}\log_2\frac{4}{4}\right) - \left(\frac{0}{4}\log_2\frac{0}{4}\right)$ $= -(0) - (0)$ $= \mathbf{0}$	$E(Sunny) = E(3,2)$ $= -\left(\frac{3}{5}\log_2\frac{3}{5}\right) - \left(\frac{2}{5}\log_2\frac{2}{5}\right)$ $= -(0.60 \log_2 0.60) - (0.40 \log_2 0.40)$ $= -(0.60 * 0.737) - (0.40 * 0.529)$ $= \mathbf{0.971}$

E(PlayGolf, Temperature) Calculation

		PlayGolf(14)		
		Yes	No	
Temperature	Hot	2	2	4
	Cold	3	1	4
	Mild	4	2	6

Table 4: Frequency Table for Temperature

$$E(PlayGolf, Temperature) = P(Hot)\,E(2,2) + P(Cold)\,E(3,1) + P(Mild)\,E(4,2)$$

$$E(PlayGolf, Temperature) = \mathbf{4/14} * \mathbf{E}(Hot) + \mathbf{4/14} * \mathbf{E}(Cold) + \mathbf{6/14} * \mathbf{E}(Mild)$$

$$E(PlayGolf, Temperature) = \mathbf{4/14} * \mathbf{E}(2, 2) + \mathbf{4/14} * \mathbf{E}(3, 1) + \mathbf{6/14} * \mathbf{E}(4, 2)$$

$$E(PlayGolf, Temperature) = 4/14 * -(2/4 \log 2/4) - (2/4 \log 2/4) + 4/14 * -(3/4 \log 3/4) - (1/4 \log 1/4) + 6/14 * -(4/6 \log 4/6) - (2/6 \log 2/6)$$

$$E(PlayGolf, Temperature) = 5/14 * 1.0 + 4/14 * 1.811 + 5/14 * 0.918$$

$$= \mathbf{0.911}$$

E(PlayGolf, Humidity) Calculation

		PlayGolf(14)		
		Yes	No	
Humidity	High	3	4	7
	Normal	6	1	7

Table 5: Frequency Table for Humidity

$$E(\text{PlayGolf, Humidity}) = \mathbf{7/14} * E(\text{High}) + \mathbf{7/14} * E(\text{Normal})$$

$$E(\text{PlayGolf, Humidity}) = \mathbf{7/14} * \mathbf{E}(3, 2) + \mathbf{7/14} * \mathbf{E}(4, 0)$$

$$E(\text{PlayGolf, Humidity}) = 7/14 * -(3/7 \log 3/7) - (4/7 \log 4/7)$$
$$+ 7/14 * -(6/7 \log 6/7) - (1/7 \log 1/7)$$

$$E(\text{PlayGolf, Humidity}) = 7/14 * 0.985$$
$$+ 7/14 * 0.592$$
$$= \mathbf{0.788}$$

E(PlayGolf, Windy) Calculation

		PlayGolf(14)		
		Yes	No	
Windy	TRUE	3	3	6
	FALSE	6	2	8

Table 6: Frequency Table for Windy

$$E(\text{PlayGolf, Windy}) = \mathbf{6/14} * \mathbf{E}(\text{True}) + \mathbf{8/14} * \mathbf{E}(\text{False})$$

$$E(\text{PlayGolf, Windy}) = \mathbf{6/14} * \mathbf{E}(3, 3) + \mathbf{8/14} * \mathbf{E}(6, 2)$$

$$E(\text{PlayGolf, Windy}) = 6/14 * -(3/6 \log 3/6) - (3/6 \log 3/6) + 8/14 * -(6/8 \log 6/8) - (2/8 \log 2/8)$$

$$E(\text{PlayGolf, Windy}) = 6/14 * 1.0 + 8/14 * 0.811 = \mathbf{0.892}$$

So now that we have all the entropies for all the four attributes

1. E(PlayGolf, Outloook) = **0.693**
2. E(PlayGolf, Temperature) = **0.911**
3. E(PlayGolf, Humidity) = **0.788**
4. E(PlayGolf,Windy) = **0.892**

Step 4: Calculating Information Gain for Each Split

The information gain is calculated using the formula:

$$Gain(S,T) = Entropy(S) - Entropy(S,T)$$

For example, the information gain after spliting using the Outlook attibute is given by:

$$Gain(PlayGolf, Outlook) = Entropy(PlayGolf) - Entropy(PlayGolf, Outlook)$$

$$= 0.94 - 0.693 = \mathbf{0.247}$$

$$Gain(PlayGolf, Temperature) = Entropy(PlayGolf) - Entropy(PlayGolf, Temparature)$$

$$= 0.94 - 0.911 = \mathbf{0.029}$$

$$Gain(PlayGolf, Humidity) = Entropy(PlayGolf) - Entropy(PlayGolf, Humidity)$$

$$= 0.94 - 0.788 = \mathbf{0.152}$$

$$Gain(PlayGolf, Windy) = Entropy(PlayGolf) - Entropy(PlayGolf, Windy)$$

$$= 0.94 - 0.892 = \mathbf{0.048}$$

Having calculated all the information gain, we now choose the attribute that gives the highest information gain after the split.

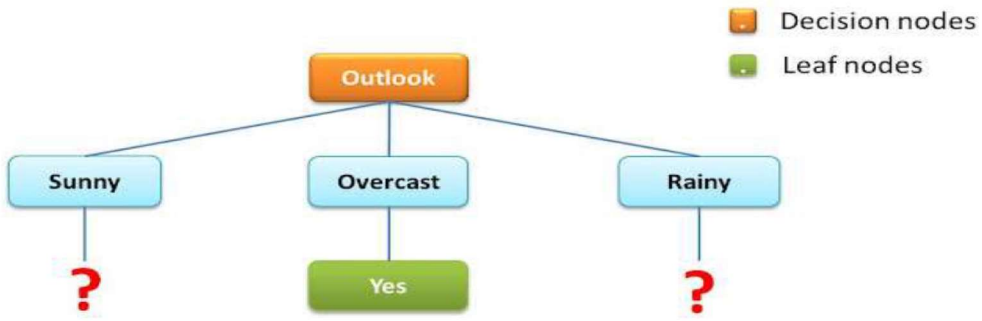


Figure 2: Decision Tree after first split

we need to also split the original table to create sub tables. This sub tables are given in below.

Outlook	Temperature	Humidity	Windy	Play Golf
Sunny	Mild	Normal	FALSE	Yes
Sunny	Mild	High	FALSE	Yes
Sunny	Cool	Normal	FALSE	Yes
Sunny	Cool	Normal	TRUE	No
Sunny	Mild	High	TRUE	No
Overcast	Hot	High	FALSE	Yes
Overcast	Mild	High	TRUE	Yes
Overcast	Hot	Normal	FALSE	Yes
Overcast	Cool	Normal	TRUE	Yes
Rainy	Hot	High	FALSE	No
Rainy	Hot	High	TRUE	No
Rainy	Mild	High	FALSE	No
Rainy	Cool	Normal	FALSE	Yes
Rainy	Mild	Normal	TRUE	Yes

we could see that the Overcast outlook requires no further split because it is just one homogeneous group. So we have a leaf node.

Step 6: Perform Further Splits

The Sunny and the Rainy attributes needs to be split
The Rainy outlook can be split using either Temperature, Humidity or Windy.

As **Humidity** produces homogenous groups for rainy attribute.

Outlook	Temperature	Humidity	Windy	Play Golf
Rainy	Hot	High	FALSE	No
Rainy	Hot	High	TRUE	No
Rainy	Mild	High	FALSE	No
Rainy	Cool	Normal	FALSE	Yes
Rainy	Mild	Normal	TRUE	Yes

Table 8: Split using Humidity

The Rainy attribute could be split using High and Normal attributes and that would give us the tree below.

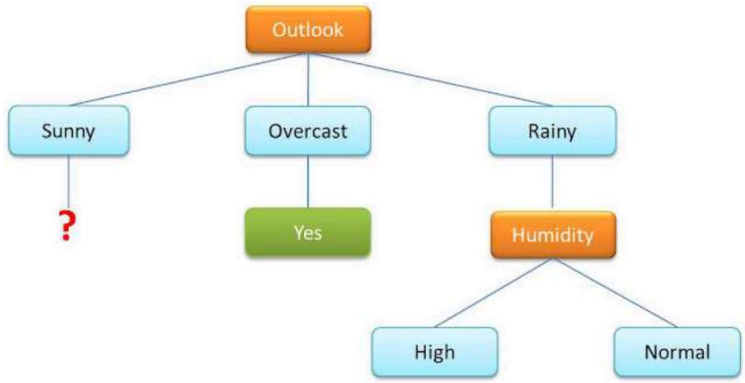


Figure 3: Split using the Humidity Attribute

Let't now go ahead to do the same thing for the Sunny outlook
As **Windy** produces homogenous groups for Sunny attribute.

Outlook	Temperature	Humidity	Windy	Play Golf
Sunny	Mild	Normal	FALSE	Yes
Sunny	Mild	High	FALSE	Yes
Sunny	Cool	Normal	FALSE	Yes
Sunny	Cool	Normal	TRUE	No
Sunny	Mild	High	TRUE	No

Table 9: Split using Windy Attribute

Step 7: Complete the Decision Tree

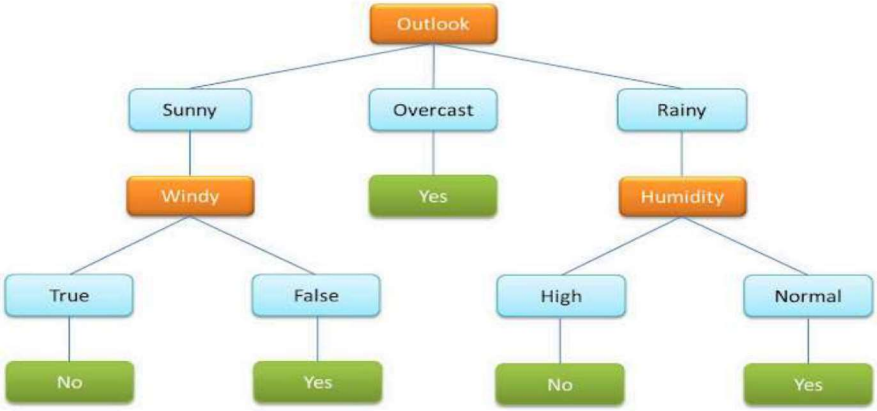


Figure 4: Final Decision Tree

Clustering

Clustering: Given a set of data objects for analysis where, unlike in classification, the class label of each object is not known. *Clustering,* also called data segmentation, is the process of grouping the data into classes or *clusters*, so that objects within a cluster have high similarity in comparison to one another but are very dissimilar to objects in other clusters. Dissimilarities are assessed based on the attribute values describing the objects. Often, distance measures are used.
A cluster is a collection of data objects that are *similar* to one another within the same cluster and are *dissimilar* to the objects in other clusters.
The most common machine learning algorithms for clustering problems including, **K-means**, DBSCAN, Gaussian Mixture Model, BIRCH, Affinity Propagation clustering, OPTICS.

Clustering can also be used for outlier detection, where outliers (values that are "far away" from any cluster) may be more interesting than common cases. Applications of outlier detection include the detection of credit card fraud and the monitoring of criminal activities in electronic commerce. For example, exceptional cases in credit card transactions, such as very expensive and frequent purchases, may be of interest as possible fraudulent activity. Clustering may serve as a preprocessing step for other algorithms, such as characterization, attribute subset selection, and classification, which would then operate on the detected clusters and the selected attributes or features.

A Categorization of Major Clustering Methods

In general, the major clustering methods can be classified into the following categories.

Partitioning methods: Given a database of *n* objects or data tuples, a partitioning method constructs *k* partitions of the data, where each partition represents a cluster, where each group must contain at least one object, and each object must belong to exactly one group.

To achieve global optimality in partitioning-based clustering would require the exhaustive enumeration of all of the possible partitions. Instead, most applications adopt one of a few popular heuristic methods, such as
(1) the *k-means* algorithm, where each cluster is represented by the **mean** value of the objects in the cluster,
(2) the *k-medoids* algorithm, where each cluster is represented by one of the objects located near the center of the cluster.
These heuristic clustering methods work well for finding **spherical-shaped** clusters. To find clusters with complex shapes and for clustering very large data sets, partitioning-based methods need to be extended.

Hierarchical methods: A hierarchical method creates a hierarchical decomposition of the given set of data objects. A hierarchical method can be classified as being either *agglomerative* or *divisive*, based on how the hierarchical decomposition is formed. The *agglomerative approach*, also called the *bottom-up* approach, starts with each object forming a separate group. It successively merges the objects or groups that are close to one another, until all of the groups are merged into one (the topmost level of the hierarchy), or until a termination condition holds. The *divisive approach*, also called the *top-down* approach, starts with all of the objects in the same cluster. In each successive iteration, a cluster is split up into smaller clusters, until eventually each object is in one cluster, or until a termination condition holds.

Density-based methods: Their general idea is to continue growing the given cluster as long as the density (number of objects or data points) in the "neighborhood" exceeds some threshold; that is, for each data point within a given cluster, the neighborhood of a given radius has to contain at least a minimum number of points. Such a method can be used to filter out noise (outliers) and discover clusters of arbitrary shape.
DBSCAN and its extension, OPTICS, are typical density-based methods that grow clusters according to a density-based connectivity analysis.

Grid-based methods: Grid-based methods quantize the object space into a finite number of cells that form a grid structure. All of the clustering operations are performed on the grid structure (i.e., on the quantized space). The main advantage of this approach is its fast processing time, which is typically independent of the number of data objects and dependent only on the number of cells in each dimension in the quantized space.
STING is a typical example of a grid-based method.

Model-based methods: Model-based methods hypothesize a model for each of the clusters and find the best fit of the data to the given model. A model-based algorithm may locate clusters by constructing a density function that reflects the spatial distribution of the data points. It also leads to a way of automatically determining the number of clusters based on standard statistics, taking "noise" or outliers into account and thus yielding robust clustering methods.
EMis an algorithm that performs expectation-maximization analysis based on statistical modeling. COBWEB is a conceptual learning algorithm that performs probability analysis and takes *concepts* as a model for clusters. SOM (or self-organizing feature map) is a neural network-based algorithm that clusters by mapping high dimensional data into a 2-D or 3-D feature map, which is also useful for data visualization.

Partitioning Methods

Given *D*, a data set of *n* objects, and *k*, the number of clusters to form, a partitioning algorithm organizes the objects into *k* partitions ($k <= n$), where each partition represents a cluster. The clusters are formed to optimize an objective partitioning criterion, such as a dissimilarity function based on distance, so that the objects within a cluster are "similar," whereas the objects of different clusters are "dissimilar" in terms of the data set attributes.

The most well-known and commonly used partitioning methods are *k-means*, *k-medoids*, and their variations.

Centroid-Based Technique: The *k*-Means Method

The *k*-means algorithm takes the input parameter, *k*, and partitions a set of *n* objects into *k* clusters so that the resulting **intracluster similarity** is **high** but the **intercluster similarity** is **low**. Cluster similarity is measured in regard to the ***mean*** value of the objects in a cluster, which can be viewed as the cluster's *centroid* or ***center of gravity.***

Algorithm: *k*-means. The *k*-means algorithm for partitioning, where each cluster's center is represented by the mean value of the objects in the cluster.

Input:

- *k*: the number of clusters,
- *D*: a data set containing *n* objects.

Output: A set of *k* clusters.

Method:

(1) arbitrarily choose *k* objects from *D* as the initial cluster centers;
(2) **repeat**
(3) (re)assign each object to the cluster to which the object is the most similar, based on the mean value of the objects in the cluster;
(4) update the cluster means, i.e., calculate the mean value of the objects for each cluster;
(5) **until** no change;

Figure 7.2 The *k*-means partitioning algorithm.

Another variant to *k*-means is the *k*-modes method, which extends the *k*-means paradigm to cluster categorical data by replacing the means of clusters with **modes**, using new dissimilarity measures to deal with **categorical** objects and a **frequency-based** method to update modes of clusters. The *k*-means and the *k*-modes methods can be integrated to cluster data with mixed numeric and categorical values.

K Means Numerical Example

https://people.revoledu.com/kardi/tutorial/kMean/NumericalExample.htm

Suppose we have several objects (4 types of medicines) and each object have two attributes or features as shown in table below. Our goal is to group these objects into K=2 group of medicine based on the two features (pH and weight index).

	weight index (x)	pH(Y)
Medicine A	1	1
Medicine B	2	1
Medicine C	4	3
Medicine D	5	4

Each medicine represents one point with two attributes (X, Y)

1. *Initial value of centroids* : Suppose we use medicine A and medicine B as the first
centroids. Let c1 and c2 denote the coordinate of the centroids, then c1=(1,1) and c2=(2,1)

2. *Objects-Centroids distance* : we calculate the distance between cluster centroid to each object. Let us use Euclidean distance, then we have distance matrix at iteration 0 is

$$\mathbf{D}^0 = \begin{bmatrix} 0 & 1 & 3.61 & 5 \\ 1 & 0 & 2.83 & 4.24 \end{bmatrix} \quad \begin{matrix} \mathbf{c}_1 = (1,1) & group-1 \\ \mathbf{c}_2 = (2,1) & group-2 \end{matrix}$$

$$\begin{matrix} A & B & C & D \end{matrix}$$

$$\begin{bmatrix} 1 & 2 & 4 & 5 \\ 1 & 1 & 3 & 4 \end{bmatrix} \begin{matrix} X \\ Y \end{matrix}$$

Each column in the distance matrix symbolizes the object. The first row of the distance matrix corresponds to the distance of each object to the first centroid and the second row is the distance of each object to the second centroid. For example, distance from medicine C = (4, 3) to the first centroid

$\mathbf{c}_1 = (1,1)$ is $\sqrt{(4-1)^2+(3-1)^2} = 3.61$

and its distance to the second centroid

$\mathbf{c}_2 = (2,1)$ is $\sqrt{(4-2)^2+(3-1)^2} = 2.83$

3. *Objects clustering* : We assign each object based on the minimum distance. Thus, medicine A is assigned to group 1, medicine B to group 2, medicine C to group 2 and medicine D to group 2. The element of Group matrix below is 1 if and only if the object is assigned to that group.

$$\mathbf{G}^0 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 \end{bmatrix} \quad \begin{matrix} group-1 \\ group-2 \end{matrix}$$

$$\begin{matrix} A & B & C & D \end{matrix}$$

4. *Iteration-1, determine centroids* : Knowing the members of each group, now we compute the new centroid of each group based on these new memberships. Group 1 only has one member thus the centroid remains in c1(1,1)

Group 2 now has three members, thus the centroid is the average coordinate among the three members

$$c_2 = \left(\frac{2+4+5}{3}, \frac{1+3+4}{3}\right) = \left(\frac{11}{3}, \frac{8}{3}\right)$$

5. *Iteration-1, Objects-Centroids distances* : The next step is to compute the distance of all objects to the new centroids. Similar to step 2, we have distance matrix at iteration 1 is

$$D^1 = \begin{bmatrix} 0 & 1 & 3.61 & 5 \\ 3.14 & 2.36 & 0.47 & 1.89 \end{bmatrix} \quad \begin{matrix} c_1 = (1,1) & group-1 \\ c_2 = \left(\frac{11}{3}, \frac{8}{3}\right) & group-2 \end{matrix}$$

$$\begin{matrix} A & B & C & D \end{matrix}$$

$$\begin{bmatrix} 1 & 2 & 4 & 5 \\ 1 & 1 & 3 & 4 \end{bmatrix} \quad \begin{matrix} X \\ Y \end{matrix}$$

6. *Iteration-1, Objects clustering:* Similar to step 3, we assign each object based on the minimum distance. Based on the new distance matrix, we move the medicine B to Group 1 while all the other objects remain. The Group matrix is shown below

$$G^1 = \begin{bmatrix} 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \end{bmatrix} \quad \begin{matrix} group-1 \\ group-2 \end{matrix}$$

$$\begin{matrix} A & B & C & D \end{matrix}$$

7. *Iteration 2, determine centroids:* Now we repeat step 4 to calculate the new centroids coordinate based on the clustering of previous iteration. Group1 and group 2 both has two members, thus the new centroids are

$$c_1 = \left(\frac{1+2}{2}, \frac{1+1}{2}\right) = \left(1\tfrac{1}{2}, 1\right) \quad \text{and} \quad c_2 = \left(\frac{4+5}{2}, \frac{3+4}{2}\right) = \left(4\tfrac{1}{2}, 3\tfrac{1}{2}\right)$$

8. *Iteration-2, Objects-Centroids distances* : Repeat step 2 again, we have new distance matrix at iteration 2 as

$$D^2 = \begin{bmatrix} 0.5 & 0.5 & 3.20 & 4.61 \\ 4.30 & 3.54 & 0.71 & 0.71 \end{bmatrix} \quad \begin{matrix} c_1 = \left(1\frac{1}{2}, 1\right) & group-1 \\ c_2 = \left(4\frac{1}{2}, 3\frac{1}{2}\right) & group-2 \end{matrix}$$

$$\begin{matrix} A & B & C & D \end{matrix}$$

$$\begin{bmatrix} 1 & 2 & 4 & 5 \\ 1 & 1 & 3 & 4 \end{bmatrix} \quad \begin{matrix} X \\ Y \end{matrix}$$

9. *Iteration-2, Objects clustering:* Again, we assign each object based on the minimum distance.

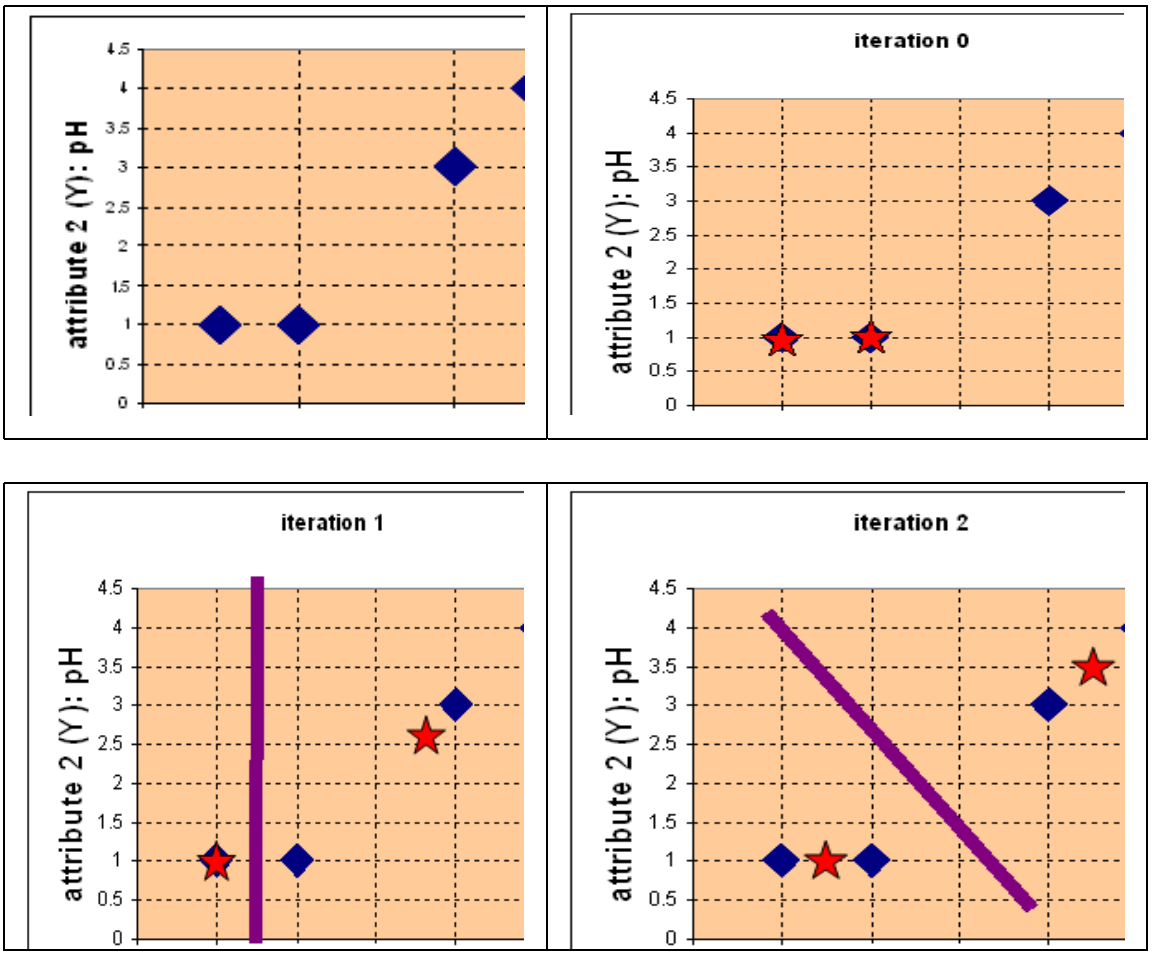
$$G^2 = \begin{bmatrix} 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \end{bmatrix} \quad \begin{matrix} group-1 \\ group-2 \end{matrix}$$

$$\begin{matrix} A & B & C & D \end{matrix}$$

We obtain result that $G^2=G^1$. Comparing the grouping of last iteration and this iteration reveals that the objects does not move group anymore. Thus, the computation of the k-mean clustering has reached its stability and no more iteration is needed. We get the final grouping as the results

	weight index (x)	pH(Y)	Group
Medicine A	1	1	1

Medicine B	2	1	1
Medicine C	4	3	2
Medicine D	5	4	2



Hierarchical Methods

A hierarchical clustering method works by grouping data objects into a tree of clusters. Hierarchical clustering methods can be further classified as either *agglomerative* or *divisive*, depending on whether the hierarchical decomposition is formed in a bottom-up (merging) or top-down (splitting) fashion. The quality of a pure hierarchical clustering method suffers from its inability to perform adjustment once a merge or split decision has been executed. That is, if a particular merge or split decision later turns out to have been a poor choice, the method cannot backtrack and correct it.

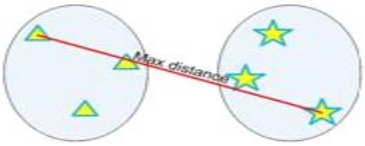
Linkages between Objects

The rule of hierarchical clustering lie on how objects should be grouped into clusters. Given a distance matrix, linkages between objects can be computed through a criterion to compute distance between groups. Most common & basic criteria are

Single Linkage: minimum distance criterion

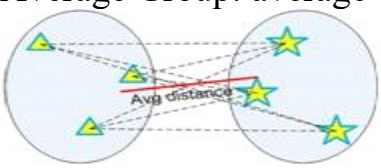
$$d_{A\to B} = \min_{\substack{\forall i\in A \\ \forall j\in B}} \left(d_{ij}\right)$$

Complete Linkage: maximum distance criterion



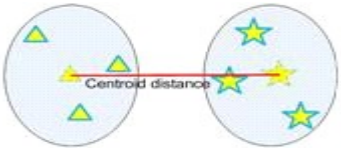
$$d_{A\to B} = \max_{\substack{\forall i\in A \\ \forall j\in B}} \left(d_{ij}\right)$$

Average Group: average distance criterion



$$d_{A\to B} = \operatorname{average}_{\substack{\forall i\in A \\ \forall j\in B}} \left(d_{ij}\right)$$

Centroid distance criterion



$$d_{A\to B} = \left\|\mathbf{c}_A - \mathbf{c}_B\right\| = \frac{1}{n_i n_j} \sum_{\substack{\forall i\in A \\ \forall j\in B}} \left(d_{ij}\right)$$

Ward: minimize variance of the merge cluster

Ward's distance between clusters C_i and C_j is the difference between the total within cluster sum of squares for the two clusters separately, and the within cluster sum of squares resulting from merging the two clusters in cluster C_{ij}

$$D_w(C_i, C_j) = \sum_{x\in C_i}(x-r_i)^2 + \sum_{x\in C_j}(x-r_j)^2 - \sum_{x\in C_{ij}}(x-r_{ij})^2$$

r_i: centroid of C_i ; r_j : centroid of C_j ; r_{ij}: centroid of C_{ij}

As a general formula, Lance–Williams algorithm:
Distance between existing cluster k with n_k objects and newly formed cluster (r, s) is given as:

$$d_{k\to(r,s)} = \alpha_r d_{k\to r} + \alpha_s d_{k\to s} + \beta d_{r\to s} + \gamma \left| d_{k\to r} - d_{k\to s} \right|$$

The values of the parameters are given in the table below.

	α_r		α_s	β	γ
Single Link	1/2	1/2	0	-1/2	
Complete Link	1/2	1/2	0	1/2	
Unweighted pair group method average (UPGMA)	$\frac{n_r}{n_r+n_s}$	$\frac{n_s}{n_r+n_s}$	0	0	
weighted pair group method average (WPGMA)	1/2	1/2	0	0	
unweighted pair group method centroid (UPGMC)	$\frac{n_r}{n_r+n_s}$	$\frac{n_s}{n_r+n_s}$	$\frac{-n_r n_s}{(n_r+n_s)^2}$	0	
weighted pair group method centroid (WPGMC)	1/2	1/2	-1/4	0	
Ward's method	$\frac{n_r+n_k}{n_r+n_s+n_k}$	$\frac{n_s+n_k}{n_r+n_s+n_k}$	$\frac{-n_k}{n_r+n_s+n_k}$	0	

Numerical Example of Hierarchical Clustering
https://people.revoledu.com/kardi/tutorial/Clustering/Numerical%20Example.htm

Minimum distance clustering is also called as single linkage hierarchical clustering or nearest neighbor clustering. Distance between two clusters is defined by the minimum distance between objects of the two clusters, as shown below.
For example, we have given an input distance matrix of size 6 by 6. This distance matrix was calculated based on the object features.

Dist	A	B	C	D	E	F
A	0.00	0.71	5.66	3.61	4.24	3.20
B	0.71	0.00	4.95	2.92	3.54	2.50
C	5.66	4.95	0.00	2.24	1.41	2.50
D	3.61	2.92	2.24	0.00	1.00	0.50
E	4.24	3.54	1.41	1.00	0.00	1.12
F	3.20	2.50	2.50	0.50	1.12	0.00

We have 6 objects and we put each object into one cluster. Thus, in the beginning we have 6 clusters. Our goal is to group those 6 clusters such that at the end of the

iterations, we will have only single cluster consists of the whole six original objects.
In each step of the iteration, we find the closest pair clusters. In this case, the closest cluster is between cluster F and D with shortest distance of 0.5. Thus, we group cluster D and F into cluster (D, F). Then we update the distance matrix (see distance matrix below). Distance between ungrouped clusters will not change from the original distance matrix. Now the problem is how to calculate distance between newly grouped clusters (D, F) and other clusters?

Min Distance (Single Linkage)

Dist	A	B	C	D, F	E
A	0.00	0.71	5.66	?	4.24
B	0.71	0.00	4.95	?	3.54
C	5.66	4.95	0.00	?	1.41
D, F	?	?	?	0.00	?
E	4.24	3.54	1.41	?	0.00

That That is exactly where the linkage rule comes into effect. Using **single linkage**, we specify minimum distance between original objects of the two clusters. Using the input distance matrix, distance between cluster (D, F) and cluster A is computed as

$$d_{(D,F)\to A} = \min\left(d_{DA}, d_{FA}\right) = \min\left(3.61, 3.20\right) = 3.20$$

Distance between cluster (D, F) and cluster B is

$$d_{(D,F)\to B} = \min\left(d_{DB}, d_{FB}\right) = \min\left(2.92, 2.50\right) = 2.50$$

Similarly, distance between cluster (D, F) and cluster C is

$$d_{(D,F)\to C} = \min\left(d_{DC}, d_{FC}\right) = \min\left(2.24, 2.50\right) = 2.24$$

Finally, distance between cluster E and cluster (D, F) is calculated as

$$d_{E\to(D,F)} = \min\left(d_{ED}, d_{EF}\right) = \min\left(1.00, 1.12\right) = 1.00$$

Then, the updated distance matrix becomes

Min Distance (Single Linkage)

Dist	A	B	C	D, F	E
A	0.00	0.71	5.66	3.20	4.24
B	0.71	0.00	4.95	2.50	3.54
C	5.66	4.95	0.00	2.24	1.41
D, F	**3.20**	**2.50**	**2.24**	0.00	1.00
E	4.24	3.54	1.41	**1.00**	0.00

Looking at the lower triangular updated distance matrix, we found out that the closest distance between cluster B and cluster A is now 0.71. Thus, we group cluster A and cluster B into a single cluster name (A, B).

Now we update the distance matrix. Aside from the first row and first column, all the other elements of the new distance matrix are not changed.

Dist	A,B	C	(D, F)	E
A,B	0	?	?	?
C	?	0	2.24	1.41
(D, F)	?	2.24	0	1.00
E	?	1.41	1.00	0

Using the input distance matrix (size 6 by 6), distance between cluster C and cluster (D, F) is computed as

$$d_{C\to(A,B)} = \min\left(d_{CA}, d_{CB}\right) = \min\left(5.66, 4.95\right) = 4.95$$

Distance between cluster (D, F) and cluster (A, B) is the minimum distance between all objects involves in the two clusters

$$d_{(D,F)\to(A,B)} = \min\left(d_{DA}, d_{DB}, d_{FA}, d_{FB}\right) = \min\left(3.61, 2.92, 3.20, 2.50\right) = 2.50$$

Similarly, distance between cluster E and (A, B) is

$$d_{E\to(A,B)} = \min\left(d_{EA}, d_{EB}\right) = \min\left(4.24, 3.54\right) = 3.54$$

Then the updated distance matrix is

Min Distance (Single Linkage)

Dist	A,B	C	(D, F)	E
A,B	0	4.95	2.50	3.54
C	**4.95**	0	2.24	1.41
(D, F)	**2.50**	2.24	0	1.00
E	**3.54**	1.41	1.00	0

Observing the lower triangular of the updated distance matrix, we can see that the closest distance between clusters happens between cluster E and (D, F) at distance 1.00. Thus, we cluster them together into cluster ((D, F), E).
The updated distance matrix is given below.

Min Distance (Single Linkage)

Dist	(A,B)	C	(D, F), E
(A,B)	0.00	4.95	2.50
C	4.95	0.00	1.41
(D, F), E	**2.50**	**1.41**	0.00

Distance between cluster ((D, F), E) and cluster (A, B) is calculated as

$$d_{((D,F),E)\to(A,B)} = \min\left(d_{DA}, d_{DB}, d_{FA}, d_{FB}, d_{EA}, d_{EB}\right) = \min\left(3.61, 2.92, 3.20, 2.50, 4.24, 3.54\right) = 2.50$$

Distance between cluster ((D, F), E) and cluster C yields the minimum distance of 1.41. This distance is computed as

$$d_{((D,F),E)\to C} = \min\left(d_{DC}, d_{FC}, d_{EC}\right) = \min\left(2.24, 2.50, 1.41\right) = 1.41$$

After that, we merge cluster ((D, F), E) and cluster C into a new cluster name (((D, F), E), C).

The updated distance matrix is shown in the figure below

Min Distance (Single Linkage)

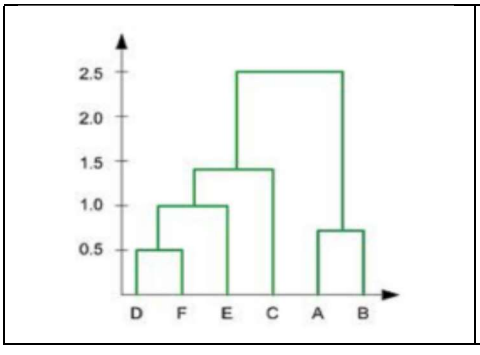
Dist	(A,B)	(D, F), E),C
(A,B)	0.00	2.50
((D, F), E),C	2.50	0.00

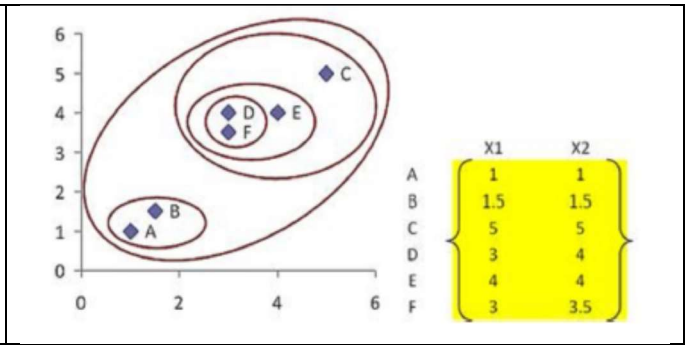
The minimum distance of 2.5 is the result of the following computation

$$d_{(((D,F),E),C)\rightarrow(A,B)} = \min\left(d_{DA}, d_{DB}, d_{FA}, d_{FB}, d_{EA}, d_{EB}, d_{CA}, d_{CB}\right)$$

$$d_{(((D,F),E),C)\rightarrow(A,B)} = \min\left(3.61, 2.92, 3.20, 2.50, 4.24, 3.54, 5.66, 4.95\right) = 2.50$$

Now if we merge the remaining two clusters, we will get only single cluster contain the whole 6 objects. Thus, our computation is finished. We summarized the results of computation as follow:
In the beginning we have 6 clusters: 1. A, B, C, D, E and F
2. We merge cluster D and F into cluster (D, F) at distance **0.50**
3. We merge cluster A and cluster B into (A, B) at distance **0.71**
4. We merge cluster E and (D, F) into ((D, F), E) at distance **1.00**
5. We merge cluster ((D, F), E) and C into (((D, F), E), C) at distance **1.41**
6. We merge cluster (((D, F), E), C) and (A, B) into ((((D, F), E), C), (A, B)) at distance **2.50**
7. The last cluster contain all the objects, thus conclude the computation
Using this information, we can now draw the final results of a dendogram. The dendogram is drawn based on the distances to merge the clusters above.
The hierarchy is given as (((D, F), E),C), (A,B). We can also plot the clustering hierarchy into XY space





	X1	X2
A	1	1
B	1.5	1.5
C	5	5
D	3	4
E	4	4
F	3	3.5

Regression is learning a function which maps a data item to a real-valued prediction variable. The function can be achieved using Ordinary Least Squares or Gradient Descent.

Linear regression

In simple linear regression, we predict scores on one variable from the scores on a second variable. The variable we are predicting is called the criterion variable and is referred to as Y. The variable we are basing our predictions on is called the predictor variable and is referred to as X. When there is only one predictor variable, the prediction method is called simple regression. In simple linear regression, the topic of this section, the predictions of Y when plotted as a function of X form a straight line.

The example data in Table 1 are plotted in Figure 1. You can see that there is a positive relationship between X and Y. If you were going to predict Y from X, the higher the value of X, the higher your prediction of Y.

Table 1. Example data.

X	Y
1.00	1.00
2.00	2.00
3.00	1.30
4.00	3.75
5.00	2.25

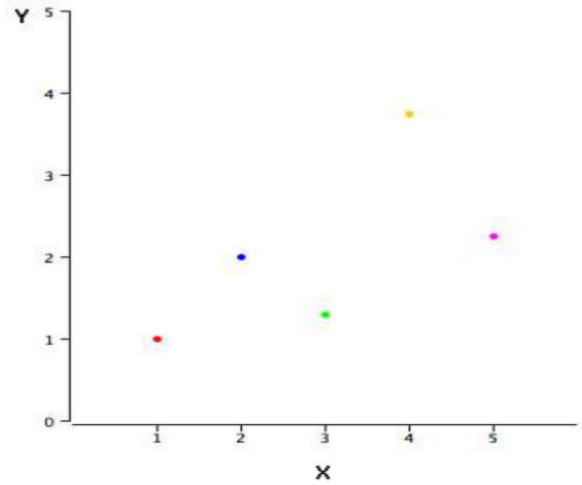


Figure 1. A scatter plot of the example data.

Linear regression consists of finding the best-fitting straight line through the points. The best-fitting line is called a regression line. The black diagonal line in Figure 2 is the regression line and consists of the predicted score on Y for each possible value of X. The vertical lines from the points to the regression line represent the errors of prediction. As you can see, the red point is very near the regression line; its error of prediction is small. By contrast, the yellow point is much higher than the regression line and therefore its error of prediction is large.

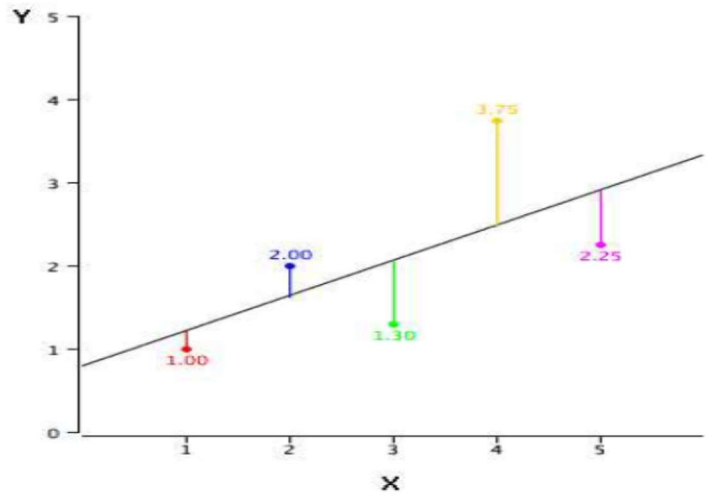


Figure 2. A scatter plot of the example data. The black line consists of the predictions, the points are the actual data, and the vertical lines between the points and the black line represent errors of prediction.

The error of prediction for a point is the value of the point minus the predicted value (the value on the line). Table 2 shows the predicted values (Y') and the errors of prediction (Y-Y'). For example, the first point has a Y of 1.00 and a predicted Y (called Y') of 1.21. Therefore, its error of prediction is -0.21.

Table 2. Example data.

X	Y	Y'	Y-Y'	$(Y-Y')^2$
1.00	1.00	1.210	-0.210	0.044
2.00	2.00	1.635	0.365	0.133
3.00	1.30	2.060	-0.760	0.578
4.00	3.75	2.485	1.265	1.600
5.00	2.25	2.910	-0.660	0.436

You may have noticed that we did not specify what is meant by "best-fitting line." By far, the most commonly-used criterion for the best-fitting line is the line that minimizes the sum of the squared errors of prediction.That is the criterion that was used to find the line in Figure 2.
The last column in Table 2 shows the squared errors of prediction. The sum of the squared errors of prediction shown in Table 2 is lower than it would be for any other regression line.

The formula for a regression line is

$$Y' = bX + A$$

where Y' is the predicted score, b is the slope of the line, and A is the Y intercept.

The **least squares approach** chooses b and A to minimize the **RSS**(residual sum of squares)

$$\mathrm{RSS} = e_1^2 + e_2^2 + \cdots + e_n^2,$$

e=Y-Y'

The equation for the line in Figure 2 is

$$Y' = 0.425X + 0.785$$

For X = 1,

$$Y' = (0.425)(1) + 0.785 = 1.21.$$

For X = 2,

$$Y' = (0.425)(2) + 0.785 = 1.64.$$

Computing the Regression Line

In the age of computers, the regression line is typically computed with statistical software. However, the calculations are relatively easy, and are given here for anyone who is interested. The calculations are based on the statistics shown in Table 3. M_X is the mean of X, M_Y is the mean of Y, s_X is the standard deviation of X, s_Y is the *standard deviation* of Y, and r is the *correlation* between X and Y.

Table 3. Statistics for computing the regression line.

M_X	M_Y	s_X	s_Y	r
3	2.06	1.581	1.072	0.627

The slope (b) can be calculated as follows:	the intercept (A) can be calculated as
$b = r\, s_Y/s_X$	$A = M_Y - bM_X.$

For these data,

$$b = (0.627)(1.072)/1.581 = 0.425$$

$$A = 2.06 - (0.425)(3) = 0.785$$

$r = \frac{\sum xy}{\sqrt{\sum x^2 \sum y^2}}$	$\sigma^2 = \frac{\sum (X-\mu)^2}{N}$	$s^2 = \frac{\sum (X-M)^2}{N-1}$

[1]https://www.datacamp.com/blog/what-is-machine-learning