UNIT 2:

Data mining:

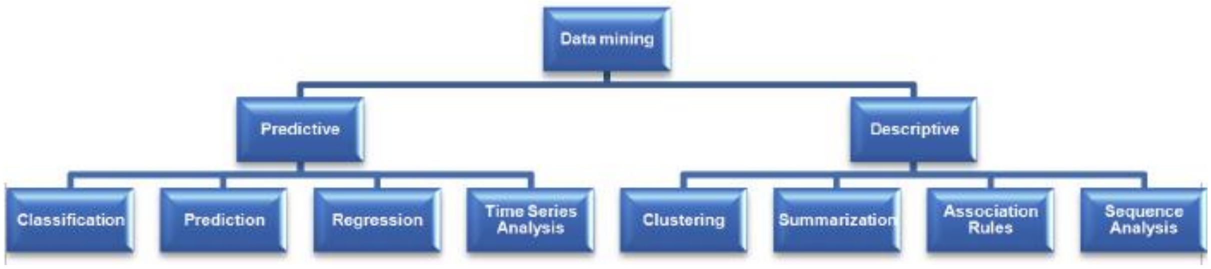
What is Data Mining: -

"Data Mining", that mines the data. In simple words, it is defined as finding hidden insights(information) from the database, extract patterns from the data.

There are different algorithms for different tasks. The function of these algorithms is to fit the model. These algorithms identify the characteristics of data. There are 2 types of models.

1)Predictive model

2)Descriptive model



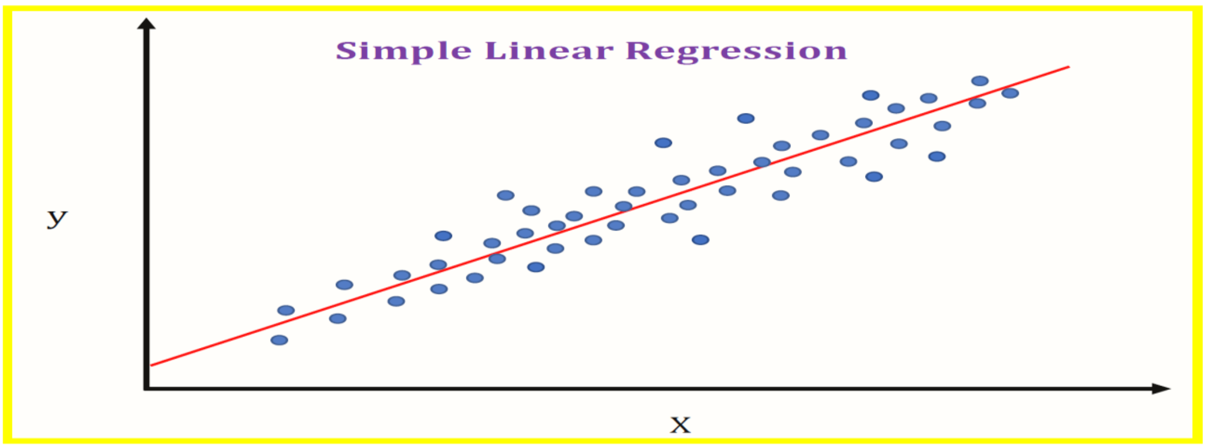
Basic Data Mining Tasks

Under this section, we are going to see some of the mining functions/tasks.

1)Classification
This term comes under supervised learning. Classification algorithms require that the classes should be defined based on variables. Characteristics of data define which class belongs to. Pattern recognition is one of the types of classification problems in which input(pattern) is classified into different classes based on its similarity of defined classes.

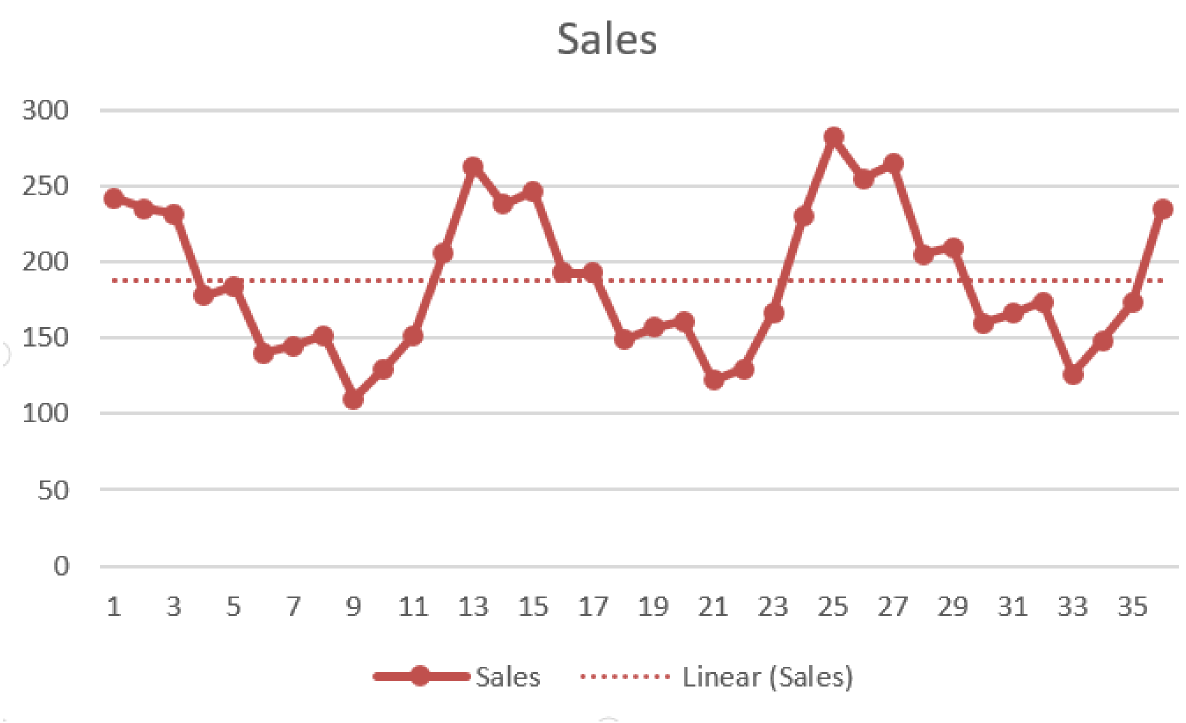
2)Prediction
In real life, we often see predicting future things/values/or else based on past data and present data. Prediction is also a type of classification task. According to the type of application, for example, predicting flood where dependant variables are the water level of the river, its humidity, raining scale, and so on are the attributes.

3)Regression



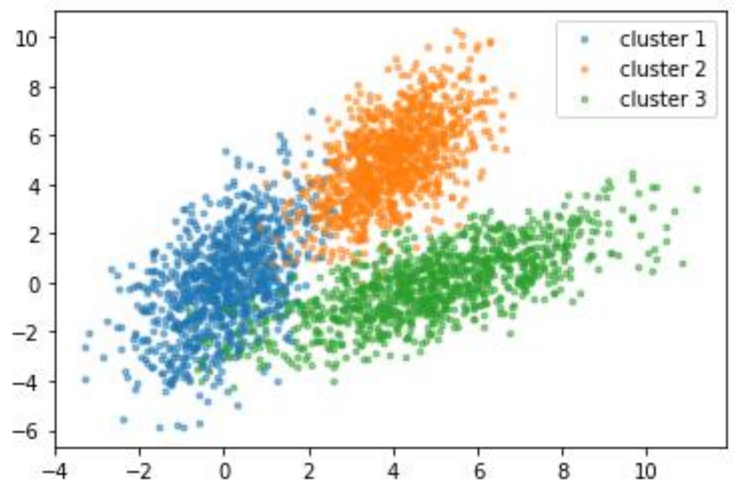
Regression is a statistical technique that is used to determine the relationship between variables(x) and dependant variables(y). There are few types of regression as Linear, Logistic, etc. Linear Regression is used in continuous values(0,1,1.5,….so on) and Logistic Regression is used where there is the possibility of only two events such as pass/fail, true/false, yes/no, etc.

4) Time Series Analysis



In time series analysis, a variable change its value according to time. It means analysis goes under the identifying patterns of data over a period of time. It can be seasonal variation, irregular variation, secular trend, and cyclical fluctuation. For example, annual rainfall, stock market price, etc.

5) Clustering



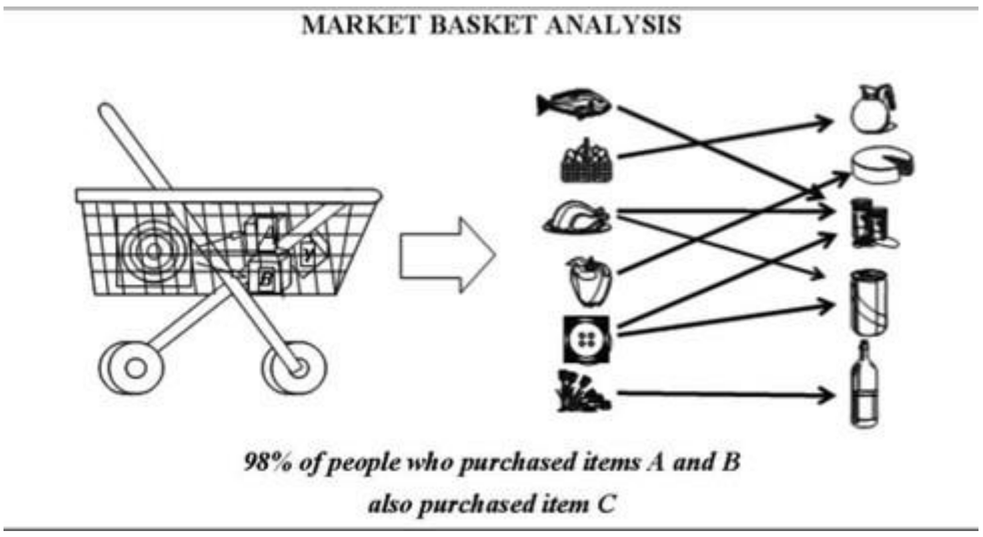
Clustering is the same as classification i.e it groups the data. Clustering comes under unsupervised machine learning. It is a process of partitioning the data into groups based on similar kinds of data.

6) Summarization

Summarization is nothing but characterization or generalization. It retrieves meaningful information from data. It also gives a summary of numeric variables such as mean, mode, median, etc.

7) Association Rules

It's the main task of Data Mining. It helps in finding appropriate patterns and meaningful insights from the database. Association Rule is a model which extracts types of data associations. For example, Market Basket Analysis where association rules are applied to the database to know that which items are purchased together by the customer.



8) Sequence Discovery

It is also called sequential analysis. It is used to discover or find the sequential pattern in data.

Sequential Pattern means the pattern which is purely based on a sequence of time. These patterns are similar to found association rules in database or events are related but its relationship is based only on "Time".

Association rules mining:

Association rule learning is a type of unsupervised learning technique that checks for the dependency of one data item on another data item and maps accordingly so that it can be more profitable. It tries to find some interesting relations or associations among the variables of dataset. It is based on different rules to discover the interesting relations between variables in the database.

The association rule learning is one of the very important concepts of machine learning

, and it is employed in **Market Basket analysis, Web usage mining, continuous production, etc.** Here market basket analysis is a technique used by the various big retailer to discover the associations between items. We can understand it by taking an example of a supermarket, as in a supermarket, all products that are purchased together are put together.

For example, if a customer buys bread, he most likely can also buy butter, eggs, or milk, so these products are stored within a shelf or mostly nearby. Consider the below diagram:

milk
bread
butter
Customer 1
milk
cereal
bread
butter
Customer 2
milk
bread
butter
Customer 3
Sugar
eggs
Customer n

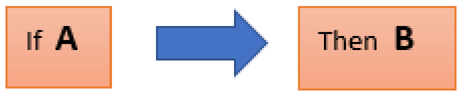
Association rule learning can be divided into three types of algorithms:

1. **Apriori**
2. **Eclat**
3. **F-P Growth Algorithm**

We will understand these algorithms in later chapters.

How does Association Rule Learning work?

Association rule learning works on the concept of If and Else Statement, such as if A then B.



Here the If element is called **antecedent**, and then statement is called as **Consequent**. These types of relationships where we can find out some association or relation between two items is known *as single cardinality*. It is all about creating rules, and if the number of items increases, then cardinality also increases accordingly. So, to measure the associations between thousands of data items, there are several metrics. These metrics are given below:

- **Support**
- **Confidence**
- **Lift**

Let's understand each of them:

Support

Support is the frequency of A or how frequently an item appears in the dataset. It is defined as the fraction of the transaction T that contains the itemset X. If there are X datasets, then for transactions T, it can be written as:

$$\text{Supp(X)}=\frac{Freq(X)}{T}$$

Confidence

Confidence indicates how often the rule has been found to be true. Or how often the items X and Y occur together in the dataset when the occurrence of X is already given. It is the ratio of the transaction that contains X and Y to the number of records that contain X.

$$\text{Confidence}=\frac{Freq(X,Y)}{Freq(X)}$$

Lift

It is the strength of any rule, which can be defined as below formula:

$$\text{Lift}=\frac{Supp(X,Y)}{Supp(X)\times Supp(Y)}$$

It is the ratio of the observed support measure and expected support if X and Y are independent of each other. It has three possible values:

- If **Lift= 1**: The probability of occurrence of antecedent and consequent is independent of each other.
- **Lift>1**: It determines the degree to which the two itemsets are dependent to each other.
- **Lift<1**: It tells us that one item is a substitute for other items, which means one item has a negative effect on another.

Types of Association Rule Learning

Association rule learning can be divided into three algorithms:

Apriori Algorithm

This algorithm uses frequent datasets to generate association rules. It is designed to work on the databases that contain transactions. This algorithm uses a breadth-first search and Hash Tree to calculate the itemset efficiently.

It is mainly used for market basket analysis and helps to understand the products that can be bought together. It can also be used in the healthcare field to find drug reactions for patients.

Eclat Algorithm

Eclat algorithm stands for **Equivalence Class Transformation**. This algorithm uses a depth-first search technique to find frequent itemsets in a transaction database. It performs faster execution than Apriori Algorithm.

F-P Growth Algorithm

The F-P growth algorithm stands for **Frequent Pattern**, and it is the improved version of the Apriori Algorithm. It represents the database in the form of a tree structure that is known as a frequent pattern or tree. The purpose of this frequent tree is to extract the most frequent patterns.

Applications of Association Rule Learning

It has various applications in machine learning and data mining. Below are some popular applications of association rule learning:

- **Market Basket Analysis:** It is one of the popular examples and applications of association rule mining. This technique is commonly used by big retailers to determine the association between items.
- **Medical Diagnosis:** With the help of association rules, patients can be cured easily, as it helps in identifying the probability of illness for a particular disease.
- **Protein Sequence:** The association rules help in determining the synthesis of artificial Proteins.
- It is also used for the **Catalog Design** and **Loss-leader Analysis** and many more other applications.

Naive algorithm

- Naïve Bayes algorithm is a supervised learning algorithm, which is based on **Bayes theorem** and used for solving classification problems.
- It is mainly used in *text classification* that includes a high-dimensional training dataset.
- Naïve Bayes Classifier is one of the simple and most effective Classification algorithms which helps in building the fast machine learning models that can make quick predictions.
- It is a probabilistic classifier, which means it predicts on the basis of the probability of an object.
- Some popular examples of Naïve Bayes Algorithm are spam filtration, Sentimental analysis, and classifying articles.

Why is it called Naïve Bayes?

The Naïve Bayes algorithm is comprised of two words Naïve and Bayes, Which can be described as:

- **Naïve**: It is called Naïve because it assumes that the occurrence of a certain feature is independent of the occurrence of other features. Such as if the fruit is identified on the bases of color, shape, and taste, then red, spherical, and sweet fruit is recognized as an apple. Hence each feature individually contributes to identify that it is an apple without depending on each other.
- **Bayes**: It is called Bayes because it depends on the principle of Bayes' Theorem.

Bayes' Theorem:

- Bayes' theorem is also known as **Bayes' Rule** or **Bayes' law**, which is used to determine the probability of a hypothesis with prior knowledge. It depends on the conditional probability.
- The formula for Bayes' theorem is given as:

$$P(A|B)=\frac{P(B|A)P(A)}{P(B)}$$

Where,

P(A|B) is Posterior probability: Probability of hypothesis A on the observed event B.

P(B|A) is Likelihood probability: Probability of the evidence given that the probability of a hypothesis is true.

P(A) is Prior Probability: Probability of hypothesis before observing the evidence.

P(B) is Marginal Probability: Probability of Evidence.

Working of Naïve Bayes' Classifier:

Working of Naïve Bayes' Classifier can be understood with the help of the below example:

Suppose we have a dataset of **weather conditions** and corresponding target variable "**Play**". So using this dataset we need to decide that whether we should play or not on a particular day according to the weather conditions. So to solve this problem, we need to follow the below steps:

1. Convert the given dataset into frequency tables.
2. Generate Likelihood table by finding the probabilities of given features.

3. Now, use Bayes theorem to calculate the posterior probability.

Problem: If the weather is sunny, then the Player should play or not?

Solution: To solve this, first consider the below dataset:

	Outlook	**Play**
0	Rainy	Yes
1	Sunny	Yes
2	Overcast	Yes
3	Overcast	Yes
4	Sunny	No
5	Rainy	Yes
6	Sunny	Yes
7	Overcast	Yes
8	Rainy	No
9	Sunny	No
10	Sunny	Yes
11	Rainy	No
12	Overcast	Yes
13	Overcast	Yes

Frequency table for the Weather Conditions:

Weather	Yes	No
Overcast	5	0

Rainy	2	2
Sunny	3	2
Total	10	5

Likelihood table weather condition:

Weather	No	Yes	
Overcast	0	5	5/14= 0.35
Rainy	2	2	4/14=0.29
Sunny	2	3	5/14=0.35
All	4/14=0.29	10/14=0.71	

Applying Bayes'theorem:

P(Yes|Sunny)= P(Sunny|Yes)*P(Yes)/P(Sunny)

P(Sunny|Yes)= 3/10= 0.3

P(Sunny)= 0.35

P(Yes)=0.71

So P(Yes|Sunny) = 0.3*0.71/0.35= **0.60**

P(No|Sunny)= P(Sunny|No)*P(No)/P(Sunny)

P(Sunny|NO)= 2/4=0.5

P(No)= 0.29

P(Sunny)= 0.35

So P(No|Sunny)= 0.5*0.29/0.35 = **0.41**

So as we can see from the above calculation that **P(Yes|Sunny)>P(No|Sunny)**

Hence on a Sunny day, Player can play the game.

Advantages of Naïve Bayes Classifier:

- Naïve Bayes is one of the fast and easy ML algorithms to predict a class of datasets.
- It can be used for Binary as well as Multi-class Classifications.

- It performs well in Multi-class predictions as compared to the other Algorithms.
- It is the most popular choice for **text classification problems**.

Disadvantages of Naïve Bayes Classifier:

- Naive Bayes assumes that all features are independent or unrelated, so it cannot learn the relationship between features.

Applications of Naïve Bayes Classifier:

- It is used for **Credit Scoring**.
- It is used in **medical data classification**.
- It can be used in **real-time predictions** because Naïve Bayes Classifier is an eager learner.
- It is used in Text classification such as **Spam filtering** and **Sentiment analysis**.

Types of Naïve Bayes Model:

There are three types of Naive Bayes Model, which are given below:

- **Gaussian**: The Gaussian model assumes that features follow a normal distribution. This means if predictors take continuous values instead of discrete, then the model assumes that these values are sampled from the Gaussian distribution.
- **Multinomial**: The Multinomial Naïve Bayes classifier is used when the data is multinomial distributed. It is primarily used for document classification problems, it means a particular document belongs to which category such as Sports, Politics, education, etc. The classifier uses the frequency of words for the predictors.
- **Bernoulli**: The Bernoulli classifier works similar to the Multinomial classifier, but the predictor variables are the independent Booleans variables. Such as if a particular word is present or not in a document. This model is also famous for document classification tasks.

Apriori Algorithm

Apriori algorithm refers to the algorithm which is used to calculate the association rules between objects. It means how two or more objects are related to one another. In other words, we can say that the apriori algorithm is an association rule leaning that analyzes that people who bought product A also bought product B.

The primary objective of the apriori algorithm is to create the association rule between different objects. The association rule describes how two or more objects are related to one another. Apriori algorithm is also called frequent pattern mining. Generally, you operate the Apriori algorithm on a database that consists of a huge number of transactions. Let's understand the apriori algorithm with the help of an example; suppose you go to Big Bazar and buy different products. It helps the customers buy their products with ease and increases the sales performance of the Big Bazar. In this tutorial, we will discuss the apriori algorithm with examples.

Introduction

We take an example to understand the concept better. You must have noticed that the Pizza shop seller makes a pizza, soft drink, and breadstick combo together. He also offers a discount to their customers who buy these combos. Do you ever think why does he do so? He thinks that customers who buy pizza also buy soft drinks and breadsticks. However, by making combos, he makes it easy for the customers. At the same time, he also increases his sales performance.

Similarly, you go to Big Bazar, and you will find biscuits, chips, and Chocolate bundled together. It shows that the shopkeeper makes it comfortable for the customers to buy these products in the same place.

The above two examples are the best examples of Association Rules in Data Mining. It helps us to learn the concept of apriori algorithms.

What is Apriori Algorithm?

Apriori algorithm refers to an algorithm that is used in mining frequent products sets and relevant association rules. Generally, the apriori algorithm operates on a database containing a huge number of transactions. For example, the items customers but at a Big Bazar.

Apriori algorithm helps the customers to buy their products with ease and increases the sales performance of the particular store.

Components of Apriori algorithm

The given three components comprise the apriori algorithm.

1. Support
2. Confidence
3. Lift

Let's take an example to understand this concept.

We have already discussed above; you need a huge database containing a large no of transactions. Suppose you have 4000 customers transactions in a Big Bazar. You have to calculate the Support, Confidence, and Lift for two products, and you may say Biscuits and Chocolate. This is because customers frequently buy these two items together.

Out of 4000 transactions, 400 contain Biscuits, whereas 600 contain Chocolate, and these 600 transactions include a 200 that includes Biscuits and chocolates. Using this data, we will find out the support, confidence, and lift.

Support

Support refers to the default popularity of any product. You find the support as a quotient of the division of the number of transactions comprising that product by the total number of transactions. Hence, we get

Support (Biscuits) = (Transactions relating biscuits) / (Total transactions)

= 400/4000 = 10 percent.

Confidence

Confidence refers to the possibility that the customers bought both biscuits and chocolates together. So, you need to divide the number of transactions that comprise both biscuits and chocolates by the total number of transactions to get the confidence.

Hence,

Confidence = (Transactions relating both biscuits and Chocolate) / (Total transactions involving Biscuits)

= 200/400

= 50 percent.

It means that 50 percent of customers who bought biscuits bought chocolates also.

Lift

Consider the above example; lift refers to the increase in the ratio of the sale of chocolates when you sell biscuits. The mathematical equations of lift are given below.

Lift = (Confidence (Biscuits - chocolates)/ (Support (Biscuits)

= 50/10 = 5

It means that the probability of people buying both biscuits and chocolates together is five times more than that of purchasing the biscuits alone. If the lift value is below one, it requires that the people are unlikely to buy both the items together. Larger the value, the better is the combination.

How does the Apriori Algorithm work in Data Mining?

We will understand this algorithm with the help of an example

Consider a Big Bazar scenario where the product set is P = {Rice, Pulse, Oil, Milk, Apple}. The database comprises six transactions where 1 represents the presence of the product and 0 represents the absence of the product.

Transaction ID	**Rice**	**Pulse**	**Oil Milk**	**Apple**

t1	1	1	1	0	0
t2	0	1	1	1	0
t3	0	0	0	1	1
t4	1	1	0	1	0
t5	1	1	1	0	1
t6	1	1	1	1	1

The Apriori Algorithm makes the given assumptions

- All subsets of a frequent itemset must be frequent.
- The subsets of an infrequent item set must be infrequent.
- Fix a threshold support level. In our case, we have fixed it at 50 percent.

Step 1

Make a frequency table of all the products that appear in all the transactions. Now, short the frequency table to add only those products with a threshold support level of over 50 percent. We find the given frequency table.

Product	**Frequency (Number of transactions)**
Rice (R)	4
Pulse(P)	5
Oil(O)	4
Milk(M)	4

The above table indicated the products frequently bought by the customers.

Step 2

Create pairs of products such as RP, RO, RM, PO, PM, OM. You will get the given frequency table.

Itemset	**Frequency (Number of transactions)**

RP	4
RO	3
RM	2
PO	4
PM	3
OM	2

Step 3

Implementing the same threshold support of 50 percent and consider the products that are more than 50 percent. In our case, it is more than 3

Thus, we get RP, RO, PO, and PM

Step 4

Now, look for a set of three products that the customers buy together. We get the given combination.

1. RP and RO give RPO
2. PO and PM give POM

Step 5

Calculate the frequency of the two itemsets, and you will get the given frequency table.

Itemset	**Frequency (Number of transactions)**
RPO	4
POM	3

If you implement the threshold assumption, you can figure out that the customers' set of three products is RPO.

We have considered an easy example to discuss the apriori algorithm in data mining. In reality, you find thousands of such combinations.

How to improve the efficiency of the Apriori Algorithm?

There are various methods used for the efficiency of the Apriori algorithm

Hash-based itemset counting

In hash-based itemset counting, you need to exclude the k-itemset whose equivalent hashing bucket count is least than the threshold is an infrequent itemset.

Transaction Reduction

In transaction reduction, a transaction not involving any frequent X itemset becomes not valuable in subsequent scans.

Apriori Algorithm in data mining

We have already discussed an example of the apriori algorithm related to the frequent itemset generation. Apriori algorithm has many applications in data mining.

The primary requirements to find the association rules in data mining are given below.

Use Brute Force

Analyze all the rules and find the support and confidence levels for the individual rule. Afterward, eliminate the values which are less than the threshold support and confidence levels.

The two-step approaches

The two-step approach is a better option to find the associations rules than the Brute Force method.

Step 1

In this article, we have already discussed how to create the frequency table and calculate itemsets having a greater support value than that of the threshold support.

Step 2

To create association rules, you need to use a binary partition of the frequent itemsets. You need to choose the ones having the highest confidence levels.

In the above example, you can see that the RPO combination was the frequent itemset. Now, we find out all the rules using RPO.

RP-O, RO-P, PO-R, O-RP, P-RO, R-PO

You can see that there are six different combinations. Therefore, if you have n elements, there will be $2^n - 2$ candidate association rules.

Advantages of Apriori Algorithm

- It is used to calculate large itemsets.
- Simple to understand and apply.

Disadvantages of Apriori Algorithms

- Apriori algorithm is an expensive method to find support since the calculation has to pass through the whole database.
- Sometimes, you need a huge number of candidate rules, so it becomes computationally more expensive.

Direct hashing and pruning (DHP):

Hashing & Pruning is very popular association rule mining technique to improve the performance of traditional Apriori algorithm. Hashing technique uses hash function to reduce the size of candidate item set.

Direct Hashing & Pruning (DHP),Perfect Hashing &Pruning (PHP) are the basic hashing algorithms. Many algorithms have been also proposed by researchers like Perfect Hashing Scheme (PHS), Sorting-Indexing and Trimming (SIT),HMFS etc.

THP arranges the item sets into vertical format and then hashed the transactions id (TID) of candidate-k item sets into hash table bucket corresponding to that item set.

Working of DHP algorithm:

Step1: Scan the database to count the support of candidate-(C1) item set and select the items :

count>=min_sup to add into large item set(L1).

Step 2: Now make possible set of candidate-2 item set in each transaction of database (D2). Hash function is applied on each candidate-2 item set to find the corresponding bucket number.

Step 3: Scan database (D2) and hash each item set of transactions into corresponding hash bucket. Some item sets are hashed into same bucket this is called collision problem.

Step4: Select only that candidate-2 item set whose corresponding bucket count>=min_sup. If there is no collision then adds into L2.

a. If there is no collision then add the selected item sets into L2.

b. Else one more scan of database is required to count the support of collided item sets and the item set

Step 5: Now make possible set of candidate-3 item set (D3) and repeat the same procedure until Ck=’

Therefore, it is quite difficult to ensure that both of these given objects refer to the same value or not.

Pros & Cons of DHP algorithm:

1. DHP uses simple hash function to reduce the size of candidate item set.

2. Size of hash table is small which requires less memory to store.

3. There is collision problem in DHP algorithm.

4. More database scans are required to count the support of collided item .

Pruning Methods: In machine learning and data mining, pruning is a technique associated with decision trees. Pruning reduces the size of decision trees by removing parts of the tree that do not provide power to classify instances. Decision trees are the most susceptible out of all the machine learning algorithms to overfitting and effective pruning can reduce this likelihood. This post will go over two techniques to help with overfitting - pre-pruning or early stopping and post-pruning with examples

Pruning or post-pruning:

As the name implies, pruning involves cutting back the tree. After a tree has been built (and in the absence of early stopping discussed below) it may be overfitted. The CART algorithm will repeatedly partition data into smaller and smaller subsets until those final subsets are homogeneous in terms of the outcome variable. In practice this often means that the final subsets (known as the leaves of the tree) each consist of only one or a few data points. The tree has learned the data exactly, but a new data point that differs very slightly might not be predicted well.

Caterogy of Pruning:

1) Minimum error:

The tree is pruned back to the point where the cross-validated error is a minimum. Cross- validation is the process of building a tree with most of the data and then using the remaining part of the data to test the accuracy of the decision tree.

2) Smallest tree:

The tree is pruned back slightly further than the minimum error. Technically the pruning creates a decision tree with cross-validation error within 1 standard error of the minimum error. The smaller tree is more intelligible at the cost of a small increase in error.

Dynamic Itemset Counting

Introduction

- Alternative to Apriori Itemset Generation
- Itemsets are dynamically added and deleted as transactions are read
- Relies on the fact that for an itemset to be frequent, all of its subsets must also be frequent, so we only examine those itemsets whose subsets are all frequent

Algorithm stops after every *M* transactions to add more itemsets.

- **Train analogy:** There are stations every *M* transactions. The passengers are itemsets. Itemsets can get on at any stop as long as they get off at the same stop in the next pass around the database. Only itemsets on the train are counted when they occur in transactions. At the very beginning we can start counting 1-itemsets, at the first station we can start counting some of the 2-itemsets. At the second station we can start counting 3-itemsets as well as any more 2-itemsets that can be counted and so on.

Itemsets are marked in four different ways as they are counted:

- **Solid box:** □ confirmed frequent itemset - an itemset we have finished counting and exceeds the support threshold *minsupp*
- **Solid circle:** ○ confirmed infrequent itemset - we have finished counting and it is below *minsupp*
- **Dashed box:** ⬚ suspected frequent itemset - an itemset we are still counting that exceeds *minsupp*
- **Dashed circle:** ◌ suspected infrequent itemset - an itemset we are still counting that is below *minsupp*

DIC Algorithm

A Java applet which combines DIC, Apriori and Probability Based Objected Interestingness Measures can be found here.

Algorithm:

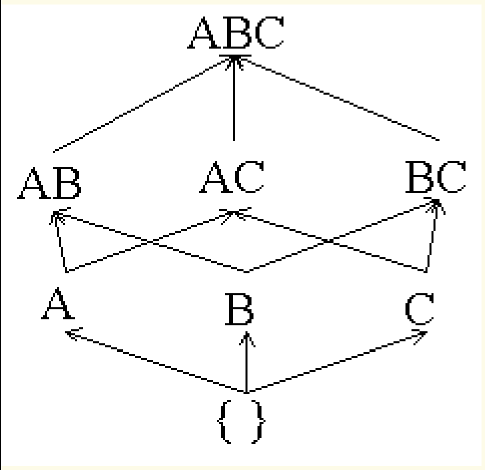
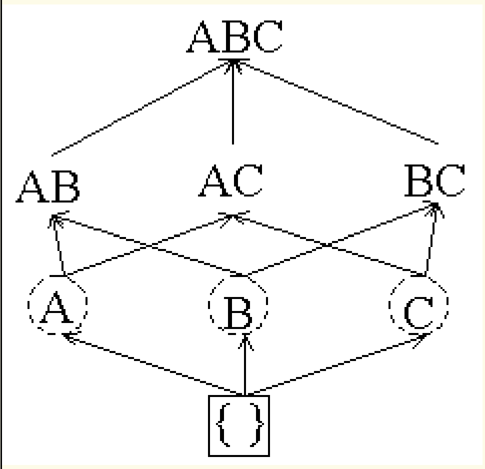
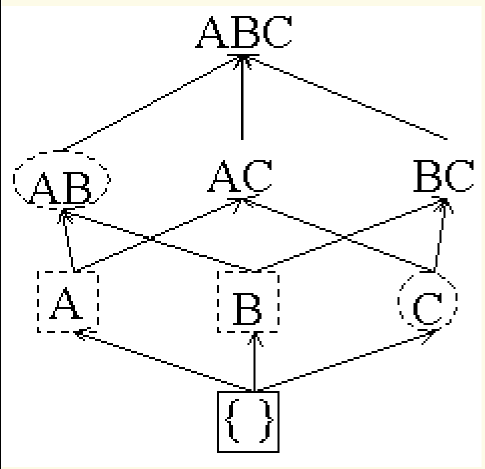
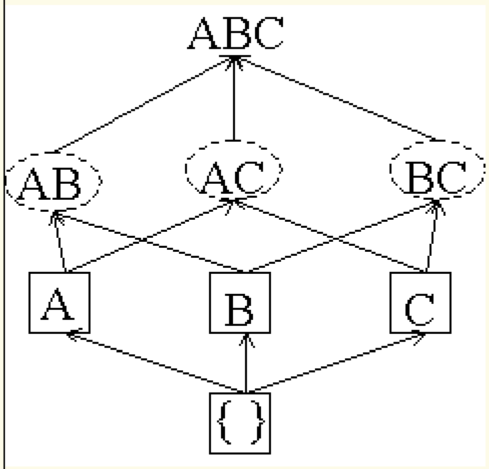
1. Mark the empty itemset with a solid square. Mark all the 1-itemsets with dashed circles. Leave all other itemsets unmarked.
2. While any dashed itemsets remain:
    1. Read *M* transactions (if we reach the end of the transaction file, continue from the beginning). For each transaction, increment the respective counters for the itemsets that appear in the transaction and are marked with dashes.
    2. If a dashed circle's count exceeds *minsupp*, turn it into a dashed square. If any immediate superset of it has all of its subsets as solid or dashed squares, add a new counter for it and make it a dashed circle.
    3. Once a dashed itemset has been counted through all the transactions, make it solid and stop counting it.

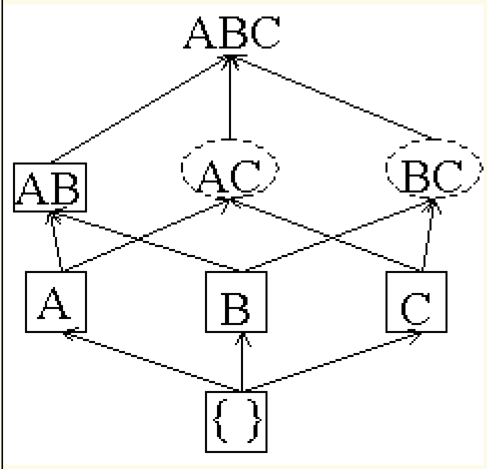
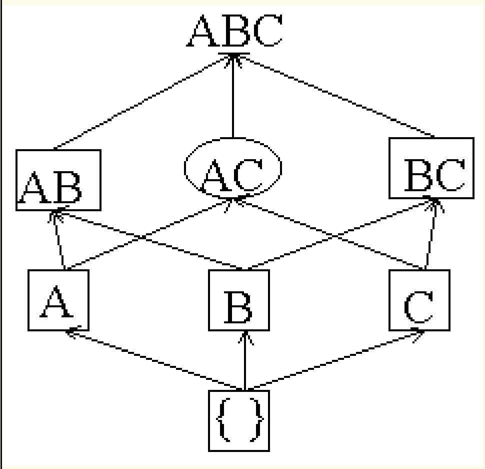
Itemset lattices: An itemset lattice contains all of the possible itemsets for a transaction database. Each itemset in the lattice points to all of its supersets. When represented graphically, a itemset lattice can help us to understand the concepts behind the DIC algorithm.

- Example: *minsupp* = 25% and M = 2.

TID	***A***	***B***	***C***
T1	1	1	0
T2	1	0	0
T3	0	1	1
T4	0	0	0

Transaction Database

Itemset lattice for the above transaction database: 	**Itemset lattice before any transactions are read:**  Counters: A = 0, B = 0, C = 0 Empty itemset is marked with a solid box. All 1-itemsets are marked with dashed circles.
After M transactions are read:  Counters: A = 2, B = 1, C = 0, AB = 0 We change A and B to dashed boxes because their counters are greater than minsup (1) and add a counter for AB because both of its subsets are boxes.	**After 2M transactions are read:**  Counters: A = 2, B = 2, C = 1, AB = 0, AC = 0, BC = 0 C changes to a square because its counter is greater than minsup.A, B and C have been counted all the way through so we stop counting them and make their boxes solid. Add counters

	for AC and BC because their subsets are all boxes.
After 3M transactions read:  Counters: A = 2, B = 2, C = 1, AB = 1, AC = 0, BC = 0 AB has been counted all the way through and its counter satisfies minsup so we change it to a solid box. BC changes to a dashed box.	**After 4M transactions read:**  Counters: A = 2, B = 2, C = 1, AB = 1, AC = 0, BC = 1 AC and BC are counted all the way through. We do not count ABC because one of its subsets is a circle. There are no dashed itemsets left so the algorithm is done.

Mining frequent pattern without candidate generation (FP, growth),

In Data Mining, finding frequent patterns in large databases is very important and has been studied on a large scale in the past few years. Unfortunately, this task is computationally expensive, especially when many patterns exist.

The FP-Growth Algorithm proposed by ***Han in***. This is an efficient and scalable method for mining the complete set of frequent patterns by pattern fragment growth, using an extended prefix-tree structure for storing compressed and crucial information about frequent patterns named frequent-pattern tree (FP-tree). In his study, Han proved that his method outperforms other popular methods for mining frequent patterns, e.g. the Apriori Algorithm and the TreeProjection. In some later works, it was proved that FP-Growth performs better than other methods, including ***Eclat*** and ***Relim***. The popularity and efficiency of the FP-Growth Algorithm contribute to many studies that propose variations to improve its performance.

What is FP Growth Algorithm?

The FP-Growth Algorithm is an alternative way to find frequent item sets without using candidate generations, thus improving performance. For so much, it uses a divide-and-conquer strategy. The core of this method is the usage of a special data structure named frequent-pattern tree (FP-tree), which retains the item set association information.

This algorithm works as follows:

- First, it compresses the input database creating an FP-tree instance to represent frequent items.
- After this first step, it divides the compressed database into a set of conditional databases, each associated with one frequent pattern.
- Finally, each such database is mined separately.

Using this strategy, the FP-Growth reduces the search costs by recursively looking for short patterns and then concatenating them into the long frequent patterns.

In large databases, holding the FP tree in the main memory is impossible. A strategy to cope with this problem is to partition the database into a set of smaller databases (called projected databases) and then construct an FP-tree from each of these smaller databases.

FP-Tree

The frequent-pattern tree (FP-tree) is a compact data structure that stores quantitative information about frequent patterns in a database. Each transaction is read and then mapped onto a path in the FP-tree. This is done until all transactions have been read. Different transactions with common subsets allow the tree to remain compact because their paths overlap.

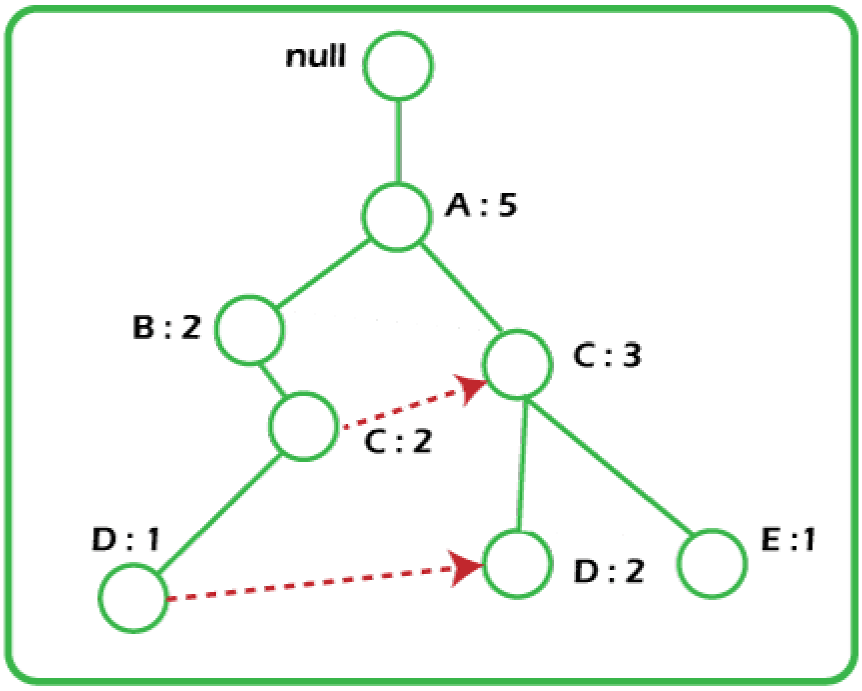
A frequent Pattern Tree is made with the initial item sets of the database. The purpose of the FP tree is to mine the most frequent pattern. Each node of the FP tree represents an item of the item set.

The root node represents null, while the lower nodes represent the item sets. The associations of the nodes with the lower nodes, that is, the item sets with the other item sets, are maintained while forming the tree.

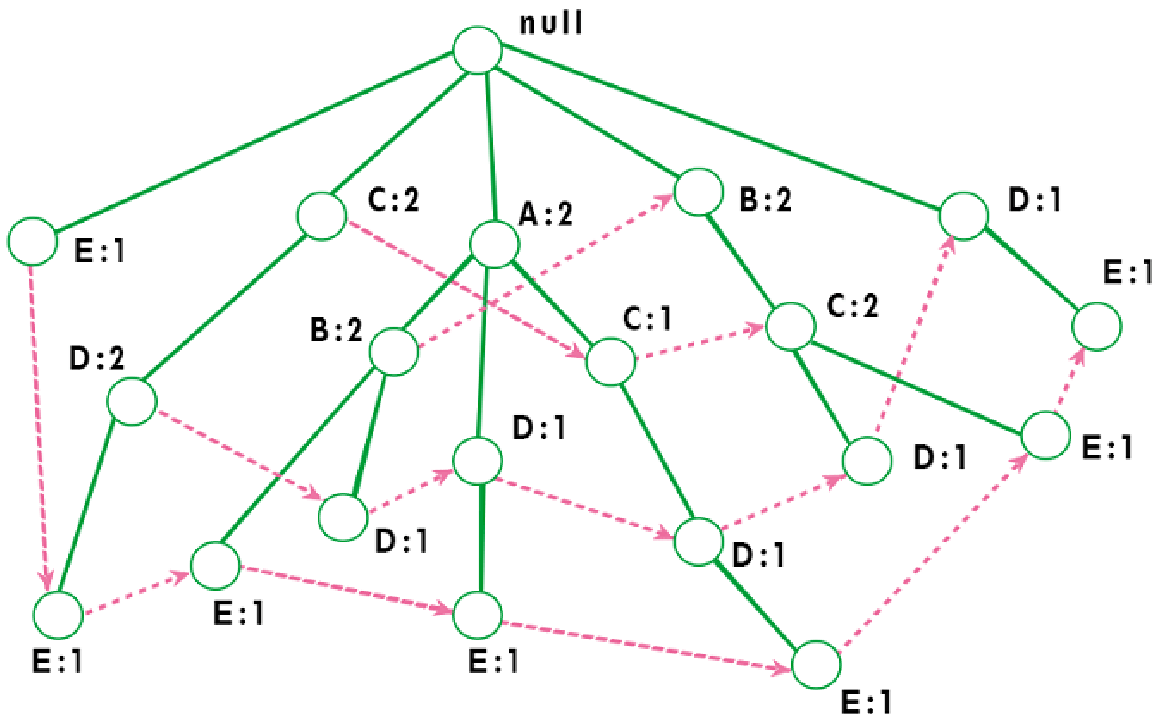
Han defines the FP-tree as the tree structure given below:

1. One root is labelled as "null" with a set of item-prefix subtrees as children and a frequent-item-header table.
2. Each node in the item-prefix subtree consists of three fields:
    - Item-name: registers which item is represented by the node;
    - Count: the number of transactions represented by the portion of the path reaching the node;
    - Node-link: links to the next node in the FP-tree carrying the same item name or null if there is none.
3. Each entry in the frequent-item-header table consists of two fields:
    - Item-name: as the same to the node;
    - Head of node-link: a pointer to the first node in the FP-tree carrying the item name.

Additionally, the frequent-item-header table can have the count support for an item. The below diagram is an example of a best-case scenario that occurs when all transactions have the same itemset; the size of the FP-tree will be only a single branch of nodes.



The worst-case scenario occurs when every transaction has a unique item set. So the space needed to store the tree is greater than the space used to store the original data set because the FP-tree requires additional space to store pointers between nodes and the counters for each item. The diagram below shows how a worst-case scenario FP-tree might appear. As you can see, the tree's complexity grows with each transaction's uniqueness.



Algorithm by Han

The original algorithm to construct the FP-Tree defined by Han is given below:

Algorithm 1: FP-tree construction

Input: A transaction database DB and a minimum support threshold?

Output: FP-tree, the frequent-pattern tree of DB.

Method: The FP-tree is constructed as follows.

1. The first step is to scan the database to find the occurrences of the itemsets in the database. This step is the same as the first step of Apriori. The count of 1-itemsets in the database is called support count or frequency of 1-itemset.
2. The second step is to construct the FP tree. For this, create the root of the tree. The root is represented by null.
3. The next step is to scan the database again and examine the transactions. Examine the first transaction and find out the itemset in it. The itemset with the max count is taken at the top, and then the next itemset with the lower count. It means that the branch of the tree is constructed with transaction itemsets in descending order of count.
4. The next transaction in the database is examined. The itemsets are ordered in descending order of count. If any itemset of this transaction is already present in another branch, then this transaction branch would share a common prefix to the root. This means that the common itemset is linked to the new node of another itemset in this transaction.
5. Also, the count of the itemset is incremented as it occurs in the transactions. The common node and new node count are increased by 1 as they are created and linked according to transactions.
6. The next step is to mine the created FP Tree. For this, the lowest node is examined first, along with the links of the lowest nodes. The lowest node represents the frequency pattern length 1. From this, traverse the path in the FP Tree. This path or paths is called a conditional pattern base.

A conditional pattern base is a sub-database consisting of prefix paths in the FP tree occurring with the lowest node (suffix).

7. Construct a Conditional FP Tree, formed by a count of itemsets in the path. The itemsets meeting the threshold support are considered in the Conditional FP Tree.
8. Frequent Patterns are generated from the Conditional FP Tree.

Using this algorithm, the FP-tree is constructed in two database scans. The first scan collects and sorts the set of frequent items, and the second constructs the FP-Tree.

Example

Support threshold=50%, Confidence= 60%

Table 1:

Transaction	**List of items**
T1	I1,I2,I3
T2	I2,I3,I4
T3	I4,I5
T4	I1,I2,I4
T5	I1,I2,I3,I5
T6	I1,I2,I3,I4

Solution: Support threshold=50% => 0.5*6= 3 => min_sup=3

Table 2: Count of each item

Item	**Count**
I1	4
I2	5
I3	4
I4	4

I5	2

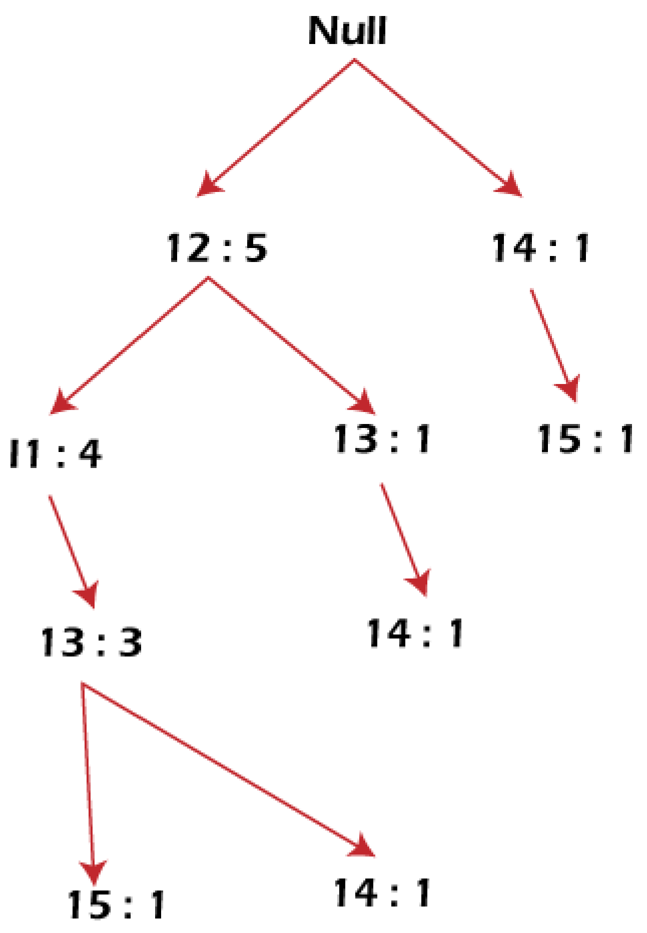
Table 3: Sort the itemset in descending order.

Item	**Count**
I2	5
I1	4
I3	4
I4	4

Build FP Tree

Let's build the FP tree in the following steps, such as:

1. Considering the root node null.
2. The first scan of Transaction T1: I1, I2, I3 contains three items {I1:1}, {I2:1}, {I3:1}, where I2 is linked as a child, I1 is linked to I2 and I3 is linked to I1.
3. T2: I2, I3, and I4 contain I2, I3, and I4, where I2 is linked to root, I3 is linked to I2 and I4 is linked to I3. But this branch would share the I2 node as common as it is already used in T1.
4. Increment the count of I2 by 1, and I3 is linked as a child to I2, and I4 is linked as a child to I3. The count is {I2:2}, {I3:1}, {I4:1}.
5. T3: I4, I5. Similarly, a new branch with I5 is linked to I4 as a child is created.
6. T4: I1, I2, I4. The sequence will be I2, I1, and I4. I2 is already linked to the root node. Hence it will be incremented by 1. Similarly I1 will be incremented by 1 as it is already linked with I2 in T1, thus {I2:3}, {I1:2}, {I4:1}.
7. T5:I1, I2, I3, I5. The sequence will be I2, I1, I3, and I5. Thus {I2:4}, {I1:3}, {I3:2}, {I5:1}.
8. T6: I1, I2, I3, I4. The sequence will be I2, I1, I3, and I4. Thus {I2:5}, {I1:4}, {I3:3}, {I4 1}.

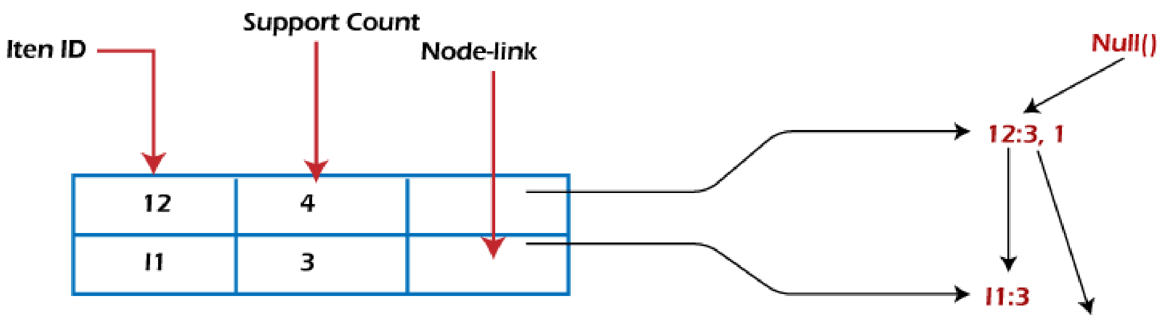


Mining of FP-tree is summarized below:

1. The lowest node item, I5, is not considered as it does not have a min support count. Hence it is deleted.
2. The next lower node is I4. I4 occurs in 2 branches , {I2,I1,I3:,I41},{I2,I3,I4:1}. Therefore considering I4 as suffix the prefix paths will be {I2, I1, I3:1}, {I2, I3: 1} this forms the conditional pattern base.
3. The conditional pattern base is considered a transaction database, and an FP tree is constructed. This will contain {I2:2, I3:2}, I1 is not considered as it does not meet the min support count.
4. This path will generate all combinations of frequent patterns : {I2,I4:2},{I3,I4:2},{I2,I3,I4:2}
5. For I3, the prefix path would be: {I2,I1:3},{I2:1}, this will generate a 2 node FP-tree : {I2:4, I1:3} and frequent patterns are generated: {I2,I3:4}, {I1:I3:3}, {I2,I1,I3:3}.
6. For I1, the prefix path would be: {I2:4} this will generate a single node FP-tree: {I2:4} and frequent patterns are generated: {I2, I1:4}.

Item	**Conditional Pattern Base**	**Conditional FP-tree**	**Frequent Patterns Generated**
I4	{I2,I1,I3:1},{I2,I3:1}	{I2:2, I3:2}	{I2,I4:2},{I3,I4:2},{I2,I3,I4:2}
I3	{I2,I1:3},{I2:1}	{I2:4, I1:3}	{I2,I3:4}, {I1:I3:3}, {I2,I1,I3:3}
I1	{I2:4}	{I2:4}	{I2,I1:4}

The diagram given below depicts the conditional FP tree associated with the conditional node I3.



FP-Growth Algorithm

After constructing the FP-Tree, it's possible to mine it to find the complete set of frequent patterns. Han presents a group of lemmas and properties to do this job and then describes the following FP-Growth Algorithm.

Algorithm 2: FP-Growth

***Input*:** A database DB, represented by FP-tree constructed according to Algorithm 1, and a minimum support threshold?

***Output*:** The complete set of frequent patterns.

***Method*:** Call FP-growth (FP-tree, null).

When the FP-tree contains a single prefix path, the complete set of frequent patterns can be generated in three parts:

1. The single prefix-path P,
2. The multipath Q,
3. And their combinations (lines 01 to 03 and 14).

The resulting patterns for a single prefix path are the enumerations of its subpaths with minimum support. After that, the multipath Q is defined, and the resulting patterns are processed. Finally, the combined results are returned as the frequent patterns found.

Advantages of FP Growth Algorithm

Here are the following advantages of the FP growth algorithm, such as:

- This algorithm needs to scan the database twice when compared to Apriori, which scans the transactions for each iteration.
- The pairing of items is not done in this algorithm, making it faster.
- The database is stored in a compact version in memory.
- It is efficient and scalable for mining both long and short frequent patterns.

Disadvantages of FP-Growth Algorithm

This algorithm also has some disadvantages, such as:

- FP Tree is more cumbersome and difficult to build than Apriori.
- It may be expensive.
- The algorithm may not fit in the shared memory when the database is large.

Difference between Apriori and FP Growth Algorithm

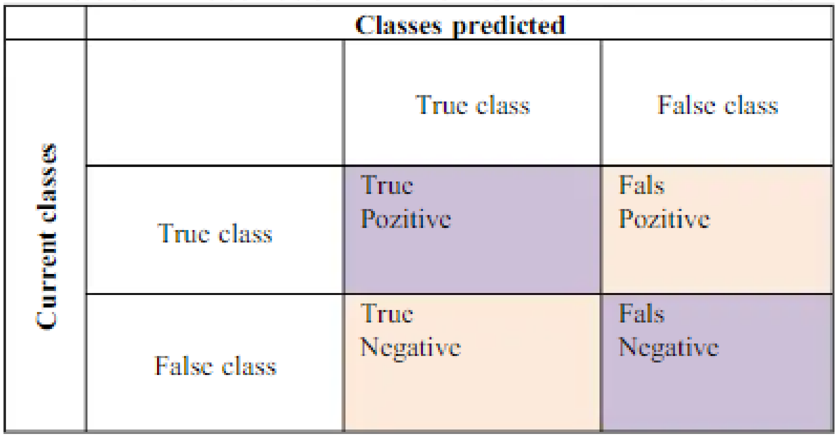
Apriori and FP-Growth algorithms are the most basic FIM algorithms. There are some basic differences between these algorithms, such as:

Apriori	FP Growth
Apriori generates frequent patterns by making the itemsets using pairings such as single item set, double itemset, and triple itemset.	FP Growth generates an FP-Tree for making frequent patterns.
Apriori uses candidate generation where frequent subsets are extended one item at a time.	FP-growth generates a conditional FP-Tree for every item in the data.
Since apriori scans the database in each step, it becomes time-consuming for data where the number of items is larger.	FP-tree requires only one database scan in its beginning steps, so it consumes less time.
A converted version of the database is saved in the memory	A set of conditional FP-tree for every item is saved in the memory
It uses a breadth-first search	It uses a depth-first search

Performance evaluation of algorithms:

Evaluation of classification algorithms is one of the key points in any process of data mining. The most commonly tools used in analyzing the results of classification algorithms applied are: confusion matrix, learning curves and receiver operating curves (ROC).The confusion matrix displays the number of correct and incorrect predictions made by the model compared with the actual classifications in the test data. Confusion matrix for a classifier with two classes True and False is presented in Table 2

Table no 2 **The confusion matrix of a classifier with two classes**



		Classes predicted	
Current classes		True class	False class
	True class	True Pozitive	Fals Pozitive
	False class	True Negative	Fals Negative

The number of correctly predicted values relative to the total number of predicted values specified by Precision parameter that takes values between 0 and 1. Precision equal to 0 indicates that the model has no predictive power, it is not conclusive,

True positive Rate (TP Rate) is the fraction of positive cases predicted as positive and is equivalent to the Recall.

False positive Rate (FP Rate) is the fraction of negative cases predicted as positive.

True negative Rate (TN Rate) is the fraction of negative cases were correctly classified as negative.

False negative Rate (FN Rate) is the fraction of positive cases that were incorrectly classified as negative.

A measure that combines precision and sensitivity is the harmonic mean of the two parameters, F-measure.[2] It is calculated using the formula:

$$F\_Measure = \frac{2 * TP\_Rate * Precision}{TP\_Rate + Precision} \quad (1)$$

The Accuracy is the proportion of the total number of correct predictions and calculated as the ratio between the number of cases correctly classified and the total number of cases.

$$Accuracy = \frac{cases\_on\_main\_diagonal}{number\_of\_totals\_cases} = \frac{TP\ Rate + FN\ Rate}{number\_of\_totals\_cases} \quad (2)$$

The Error indicates the proportion of cases classified incorrectly.

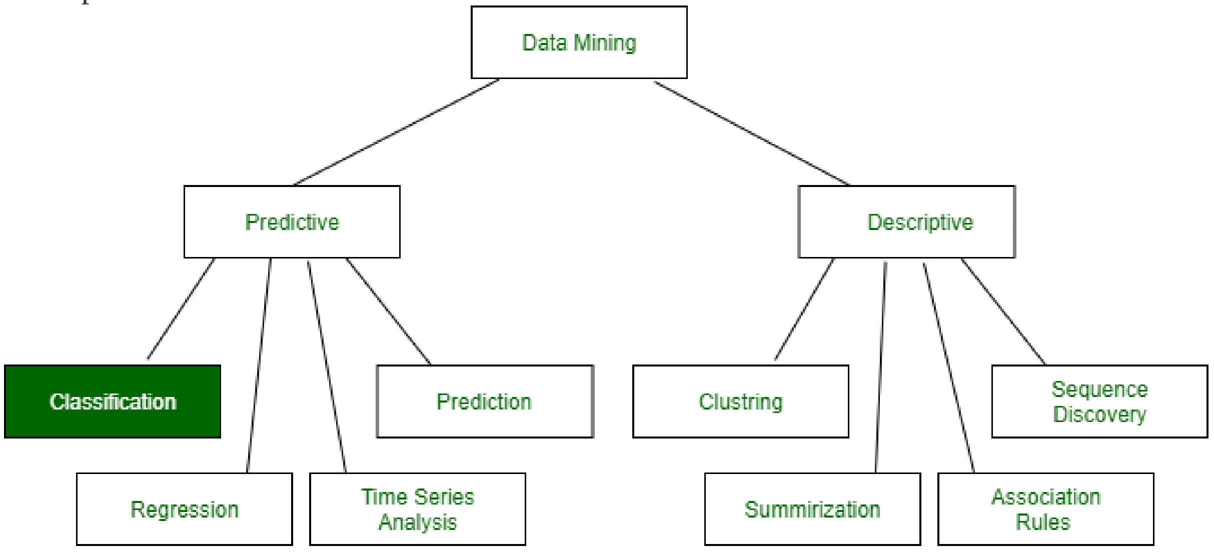
ROC curves (Receiver Operating Characteristic) are widely used in evaluating the results of predictions. These are the cases of false positive rate OX and OY axis rate of true positive cases. These represent on the OX axe the rate offalse positive cases and on the OY axe the rate of true positive cases. Each binary classifier is represented by a point on the graph (FP Rate, TP Rate) Thus,

- The point (0,1) of the ROC plot is the perfect, ideally prediction, classify correctly all the positive and negative cases ;

- The point (1,1) represents a classifier that predicts all cases to be positive;
- The point (1,0) represents a classifier that classifies all instances incorrectly

Classification:

Data Mining: Data mining in general terms means mining or digging deep into data that is in different forms to gain patterns, and to gain knowledge on that pattern. In the process of data mining, large data sets are first sorted, then patterns are identified and relationships are established to perform data analysis and solve problems.



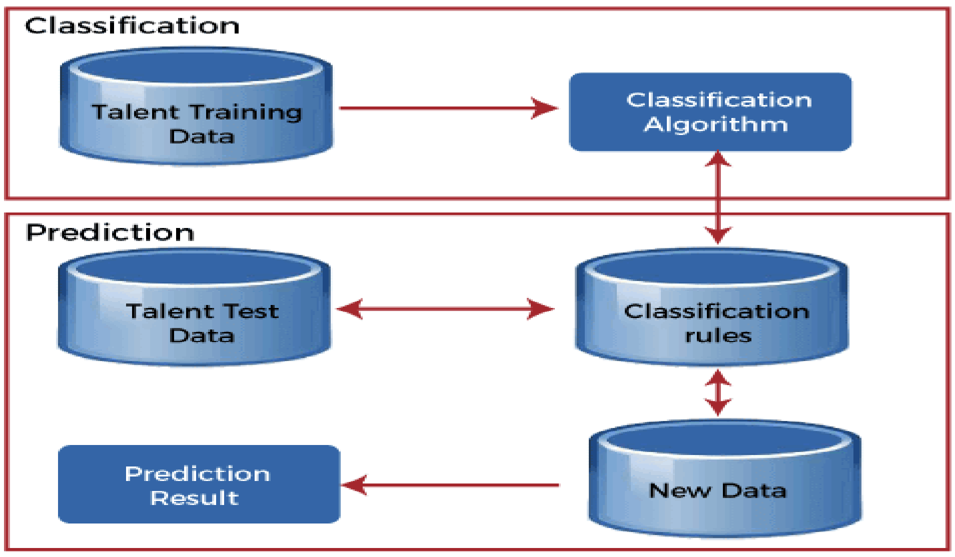
Classification and Predication in Data Mining

There are two forms of data analysis that can be used to extract models describing important classes or predict future data trends. These two forms are as follows:

1. Classification
2. Prediction

We use classification and prediction to extract a model, representing the data classes to predict future data trends. Classification predicts the categorical labels of data with the prediction models. This analysis provides us with the best understanding of the data at a large scale.

Classification models predict categorical class labels, and prediction models predict continuous-valued functions. For example, we can build a classification model to categorize bank loan applications as either safe or risky or a prediction model to predict the expenditures in dollars of potential customers on computer equipment given their income and occupation.



Classification and Prediction Process

What is Classification?

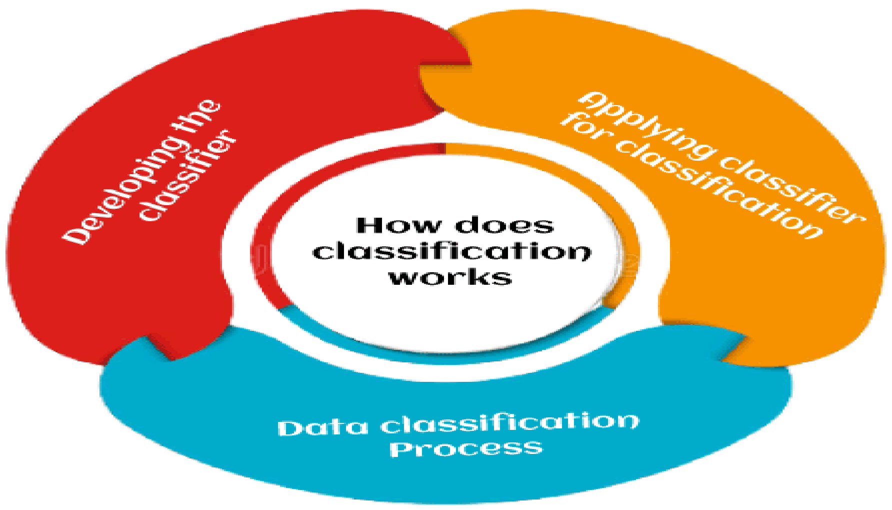
Classification is to identify the category or the class label of a new observation. First, a set of data is used as training data. The set of input data and the corresponding outputs are given to the algorithm. So, the training data set includes the input data and their associated class labels. Using the training dataset, the algorithm derives a model or the classifier. The derived model can be a decision tree, mathematical formula, or a neural network. In classification, when unlabeled data is given to the model, it should find the class to which it belongs. The new data provided to the model is the test data set.

Classification is the process of classifying a record. One simple example of classification is to check whether it is raining or not. The answer can either be yes or no. So, there is a particular number of choices. Sometimes there can be more than two classes to classify. That is called ***multiclass classification***.

The bank needs to analyze whether giving a loan to a particular customer is risky or not. **For example**, based on observable data for multiple loan borrowers, a classification model may be established that forecasts credit risk. The data could track job records, homeownership or leasing, years of residency, number, type of deposits, historical credit ranking, etc. The goal would be credit ranking, the predictors would be the other characteristics, and the data would represent a case for each consumer. In this example, a model is constructed to find the categorical label. The labels are risky or safe.

How does Classification Works?

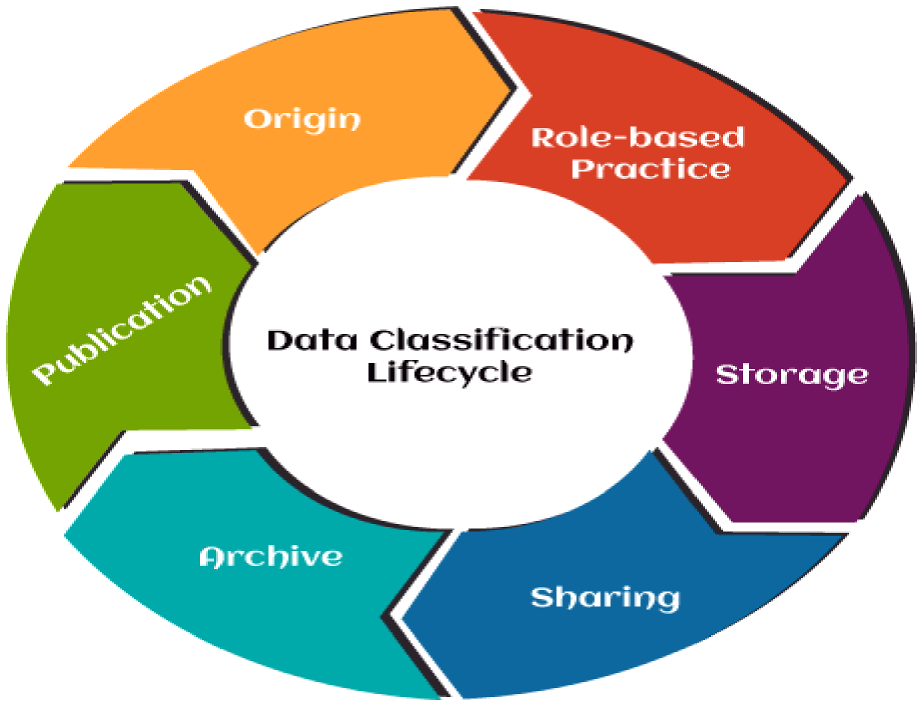
The functioning of classification with the assistance of the bank loan application has been mentioned above. There are two stages in the data classification system: classifier or model creation and classification classifier.


Developing the classifier
Applying classifier for classification
How does classification works
Data classification Process

1. **Developing the Classifier or model creation:** This level is the learning stage or the learning process. The classification algorithms construct the classifier in this stage. A classifier is constructed from a training set composed of the records of databases and their corresponding class names. Each category that makes up the training set is referred to as a category or class. We may also refer to these records as samples, objects, or data points.
2. **Applying classifier for classification:** The classifier is used for classification at this level. The test data are used here to estimate the accuracy of the classification algorithm. If the consistency is deemed sufficient, the classification rules can be expanded to cover new data records. It includes:
    - **Sentiment Analysis:** Sentiment analysis is highly helpful in social media monitoring. We can use it to extract social media insights. We can build sentiment analysis models to read and analyze misspelled words with advanced machine learning algorithms. The accurate trained models provide consistently accurate outcomes and result in a fraction of the time.
    - **Document Classification:** We can use document classification to organize the documents into sections according to the content. Document classification refers to text classification; we can classify the words in the entire document. And with the help of machine learning classification algorithms, we can execute it automatically.
    - **Image Classification:** Image classification is used for the trained categories of an image. These could be the caption of the image, a statistical value, a theme. You can tag images to train your model for relevant categories by applying supervised learning algorithms.
    - **Machine Learning Classification:** It uses the statistically demonstrable algorithm rules to execute analytical tasks that would take humans hundreds of more hours to perform.
3. **Data Classification Process:** The data classification process can be categorized into five steps:
    - Create the goals of data classification, strategy, workflows, and architecture of data classification.
    - Classify confidential details that we store.
    - Using marks by data labelling.
    - To improve protection and obedience, use effects.
    - Data is complex, and a continuous method is a classification.

What is Data Classification Lifecycle?

The data classification life cycle produces an excellent structure for controlling the flow of data to an enterprise. Businesses need to account for data security and compliance at each level. With the help of data classification, we can perform it at every stage, from origin to deletion. The data life-cycle has the following stages, such as:



1. **Origin:** It produces sensitive data in various formats, with emails, Excel, Word, Google documents, social media, and websites.
2. **Role-based practice:** Role-based security restrictions apply to all delicate data by tagging based on in-house protection policies and agreement rules.
3. **Storage:** Here, we have the obtained data, including access controls and encryption.
4. **Sharing:** Data is continually distributed among agents, consumers, and co-workers from various devices and platforms.
5. **Archive:** Here, data is eventually archived within an industry's storage systems.
6. **Publication:** Through the publication of data, it can reach customers. They can then view and download in the form of dashboards.

What is Prediction?

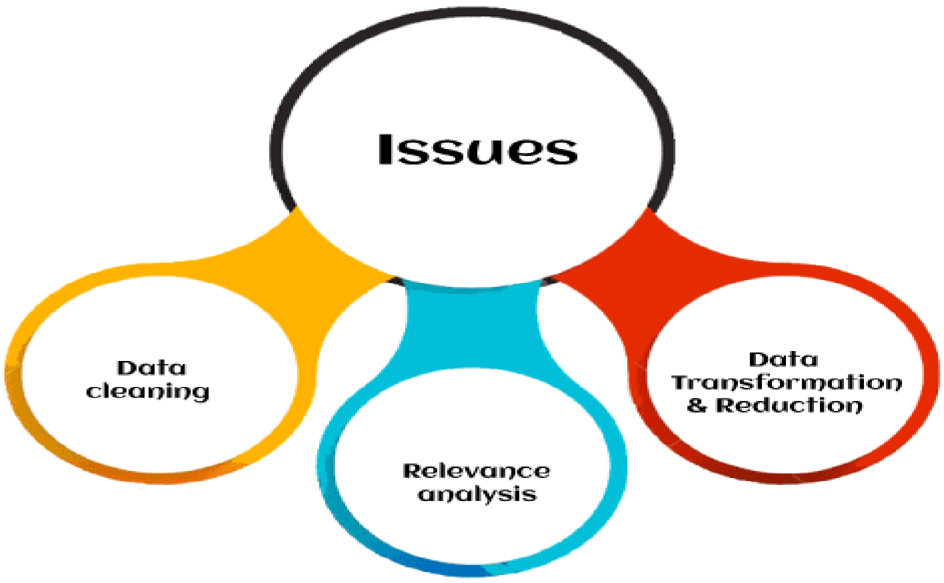
Another process of data analysis is prediction. It is used to find a numerical output. Same as in classification, the training dataset contains the inputs and corresponding numerical output values. The algorithm derives the model or a predictor according to the training dataset. The model should find a numerical output when the new data is given. Unlike in classification, this method does not have a class label. The model predicts a continuous-valued function or ordered value.

Regression is generally used for prediction. Predicting the value of a house depending on the facts such as the number of rooms, the total area, etc., is an example for prediction.

For example, suppose the marketing manager needs to predict how much a particular customer will spend at his company during a sale. We are bothered to forecast a numerical value in this case. Therefore, an example of numeric prediction is the data processing activity. In this case, a model or a predictor will be developed that forecasts a continuous or ordered value function.

Classification and Prediction Issues

The major issue is preparing the data for Classification and Prediction. Preparing the data involves the following activities, such as:

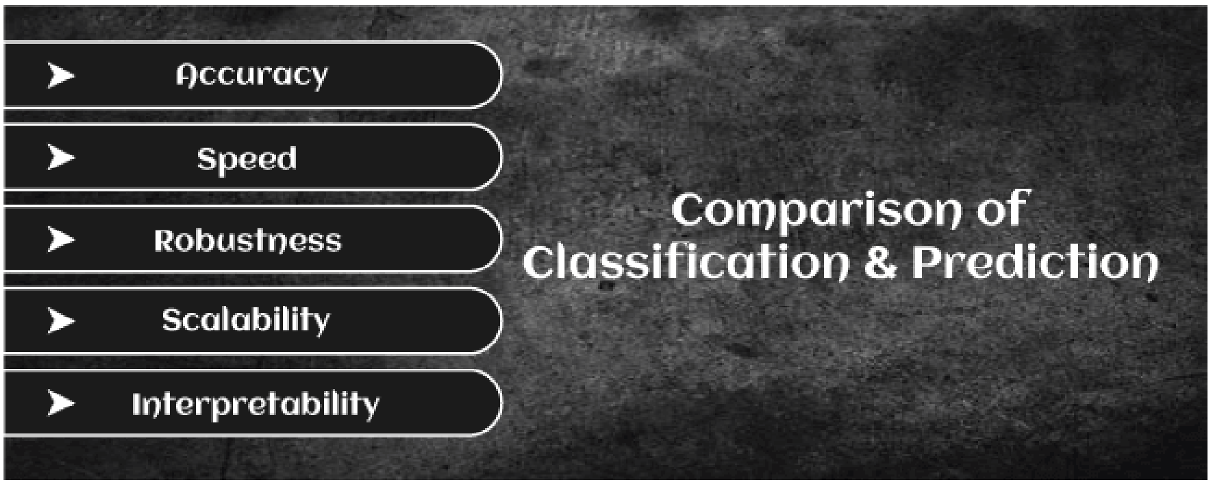


1. **Data Cleaning:** Data cleaning involves removing the noise and treatment of missing values. The noise is removed by applying smoothing techniques, and the problem of missing values is solved by replacing a missing value with the most commonly occurring value for that attribute.
2. **Relevance Analysis:** The database may also have irrelevant attributes. Correlation analysis is used to know whether any two given attributes are related.
3. **Data Transformation and reduction:** The data can be transformed by any of the following methods.
    - **Normalization:** The data is transformed using normalization. Normalization involves scaling all values for a given attribute to make them fall within a small specified range. Normalization is used when the neural networks or the methods involving measurements are used in the learning step.
    - **Generalization:** The data can also be transformed by generalizing it to the higher concept. For this purpose, we can use the concept hierarchies.

NOTE: Data can also be reduced by some other methods such as wavelet transformation, binning, histogram analysis, and clustering.

Comparison of Classification and Prediction Methods

Here are the criteria for comparing the methods of Classification and Prediction, such as:



- **Accuracy:** The accuracy of the classifier can be referred to as the ability of the classifier to predict the class label correctly, and the accuracy of the predictor can be referred to as how well a given predictor can estimate the unknown value.
- **Speed:** The speed of the method depends on the computational cost of generating and using the classifier or predictor.
- **Robustness:** Robustness is the ability to make correct predictions or classifications. In the context of data mining, robustness is the ability of the classifier or predictor to make correct predictions from incoming unknown data.
- **Scalability:** Scalability refers to an increase or decrease in the performance of the classifier or predictor based on the given data.
- **Interpretability:** Interpretability is how readily we can understand the reasoning behind predictions or classification made by the predictor or classifier.

Difference between Classification and Prediction

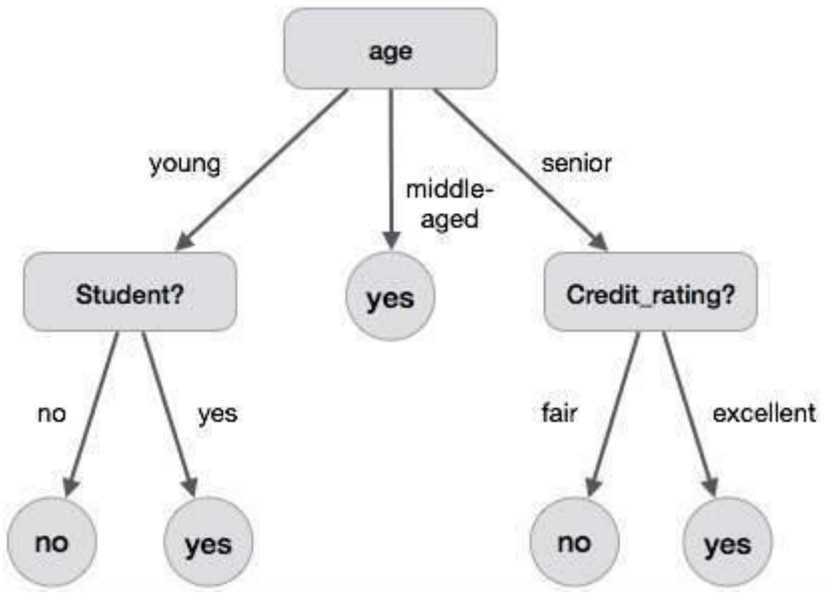
The decision tree, applied to existing data, is a classification model. We can get a class prediction by applying it to new data for which the class is unknown. The assumption is that the new data comes from a distribution similar to the data we used to construct our decision tree. In many instances, this is a correct assumption, so we can use the decision tree to build a predictive model. Classification of prediction is the process of finding a model that describes the classes or concepts of information. The purpose is to predict the class of objects whose class label is unknown using this model. Below are some major differences between classification and prediction.

Classification	Prediction
Classification is the process of identifying which category a new observation belongs to based on a training data set containing observations whose category membership is known.	Predication is the process of identifying the missing or unavailable numerical data for a new observation.
In classification, the accuracy depends on finding the class label correctly.	In prediction, the accuracy depends on how well a given predictor can guess the value of a predicated attribute for new data.
In classification, the model can be known as the classifier.	In prediction, the model can be known as the predictor.
A model or the classifier is constructed to find the categorical labels.	A model or a predictor will be constructed that predicts a continuous-valued function or ordered value.
For example, the grouping of patients based on their medical records can be considered a classification.	**For example**, We can think of prediction as predicting the correct treatment for a particular disease for a person.

Decision tree :

A decision tree is a structure that includes a root node, branches, and leaf nodes. Each internal node denotes a test on an attribute, each branch denotes the outcome of a test, and each leaf node holds a class label. The topmost node in the tree is the root node.

The following decision tree is for the concept buy_computer that indicates whether a customer at a company is likely to buy a computer or not. Each internal node represents a test on an attribute. Each leaf node represents a class.



The benefits of having a decision tree are as follows −

- It does not require any domain knowledge.
- It is easy to comprehend.
- The learning and classification steps of a decision tree are simple and fast.

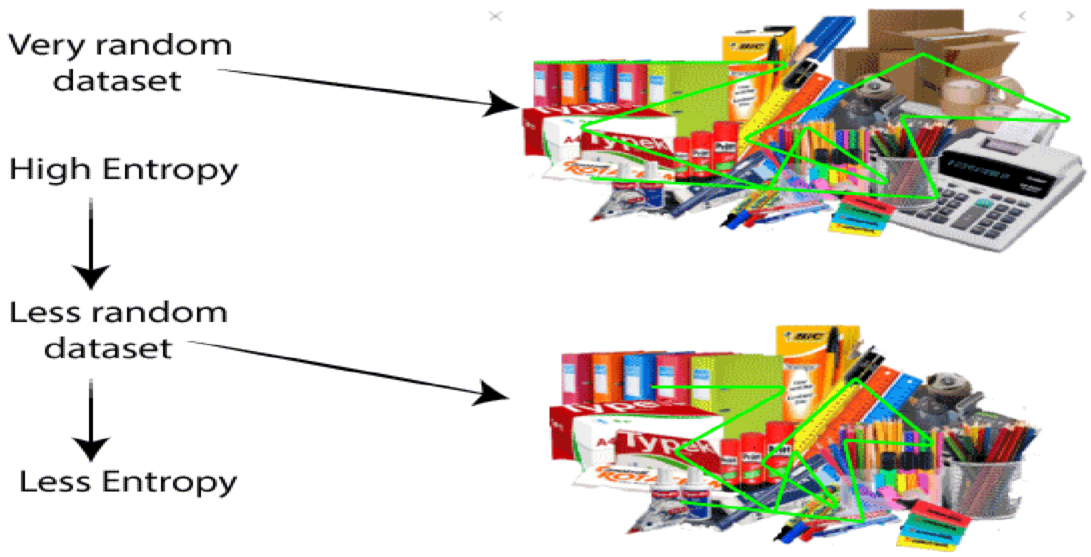
Decision Tree Induction

Decision Tree is a supervised learning method used in data mining for classification and regression methods. It is a tree that helps us in decision-making purposes. The decision tree creates classification or regression models as a tree structure. It separates a data set into smaller subsets, and at the same time, the decision tree is steadily developed. The final tree is a tree with the decision nodes and leaf nodes. A decision node has at least two branches. The leaf nodes show a classification or decision. We can't accomplish more split on leaf nodes-The uppermost decision node in a tree that relates to the best predictor called the root node. Decision trees can deal with both categorical and numerical data.

Key factors:

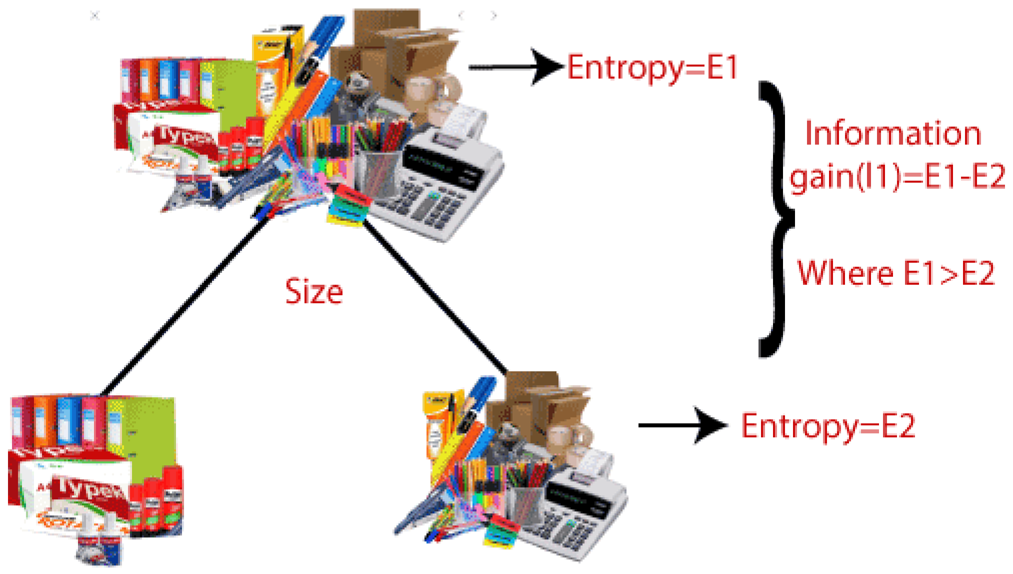
Entropy:

Entropy refers to a common way to measure impurity. In the decision tree, it measures the randomness or impurity in data sets.



Information Gain:

Information Gain refers to the decline in entropy after the dataset is split. It is also called **Entropy Reduction**. Building a decision tree is all about discovering attributes that return the highest data gain.



In short, a decision tree is just like a flow chart diagram with the terminal nodes showing decisions. Starting with the dataset, we can measure the entropy to find a way to segment the set until the data belongs to the same class.

Why are decision trees useful?

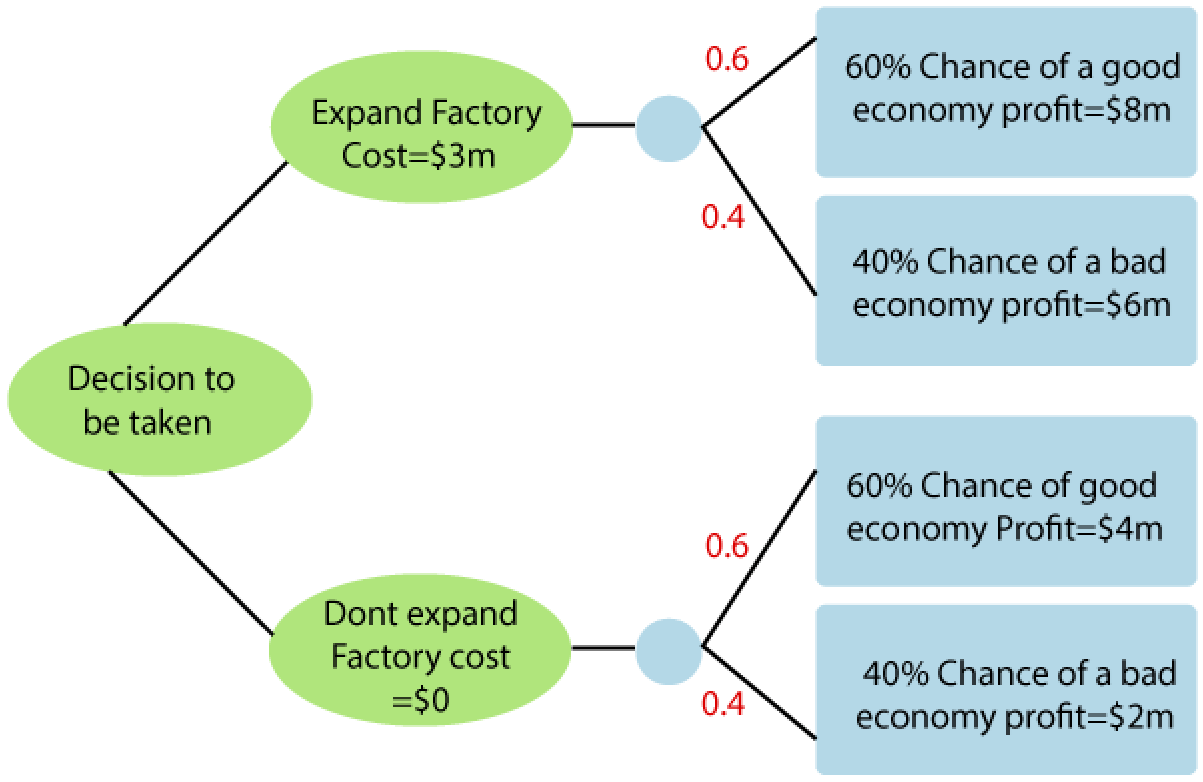
It enables us to analyze the possible consequences of a decision thoroughly.

It provides us a framework to measure the values of outcomes and the probability of accomplishing them.

It helps us to make the best decisions based on existing data and best speculations.

In other words, we can say that a decision tree is a hierarchical tree structure that can be used to split an extensive collection of records into smaller sets of the class by implementing a sequence of simple decision rules. A decision tree model comprises a set of rules for portioning a huge heterogeneous population into smaller, more homogeneous, or mutually exclusive classes. The attributes of the classes can be any variables from nominal, ordinal, binary, and quantitative values, in contrast, the classes must be a qualitative type, such as categorical or ordinal or binary. In brief, the given data of attributes together with its class, a decision tree creates a set of rules that can be used to identify the class. One rule is implemented after another, resulting in a hierarchy of segments within a segment. The hierarchy is known as the **tree**, and each segment is called a **node**. With each progressive division, the members from the subsequent sets become more and more similar to each other. Hence, the algorithm used to build a decision tree is referred to as recursive partitioning. The algorithm is known as **CART** (Classification and Regression Trees)

Consider the given example of a factory where


Expand Factory Cost=$3m
0.6
60% Chance of a good economy profit=$8m
0.4
40% Chance of a bad economy profit=$6m
Decision to be taken
0.6
60% Chance of good economy Profit=$4m
Dont expand Factory cost =$0
0.4
40% Chance of a bad economy profit=$2m

Expanding factor costs \$3 million, the probability of a good economy is 0.6 (60%), which leads to \$8 million profit, and the probability of a bad economy is 0.4 (40%), which leads to \$6 million profit.

Not expanding factor with 0\$ cost, the probability of a good economy is 0.6(60%), which leads to \$4 million profit, and the probability of a bad economy is 0.4, which leads to \$2 million profit.

The management teams need to take a data-driven decision to expand or not based on the given data.

Net Expand = (0.6 *8 + 0.4*6) - 3 = \$4.2M

Net Not Expand = (0.6*4 + 0.4*2) - 0 = \$3M

\$4.2M > \$3M,therefore the factory should be expanded.

Decision tree Algorithm:

The decision tree algorithm may appear long, but it is quite simply the basis algorithm techniques is as follows:

The **algorithm** is based on three parameters**: D, attribute_list, and Attribute _selection_method.**

Generally, we refer to **D** as a **data partition.**

Initially, **D** is the entire set of **training tuples** and their related **class levels** (input training data).

The parameter **attribute_list** is a set of **attributes** defining the tuples.

Attribute_selection_method specifies a **heuristic process** for choosing the attribute that "best" discriminates the given tuples according to **class**.

Attribute_selection_method process applies an **attribute selection measure**.

Advantages of using decision trees:

A decision tree does not need scaling of information.

Missing values in data also do not influence the process of building a choice tree to any considerable extent.

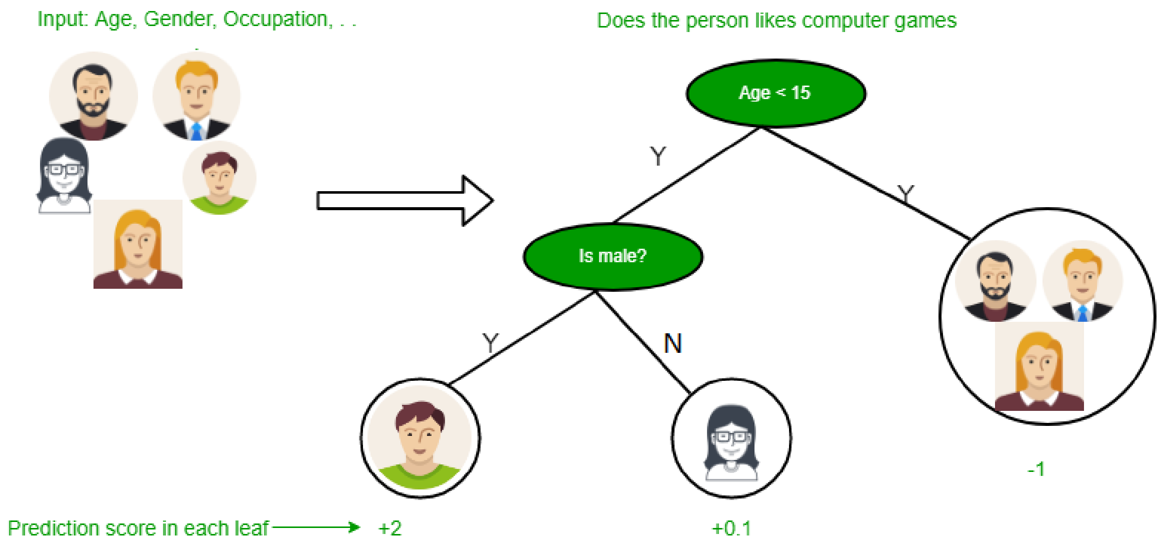
A decision tree model is automatic and simple to explain to the technical team as well as stakeholders.

Compared to other algorithms, decision trees need less exertion for data preparation during pre-processing.

A decision tree does not require a standardization of data.

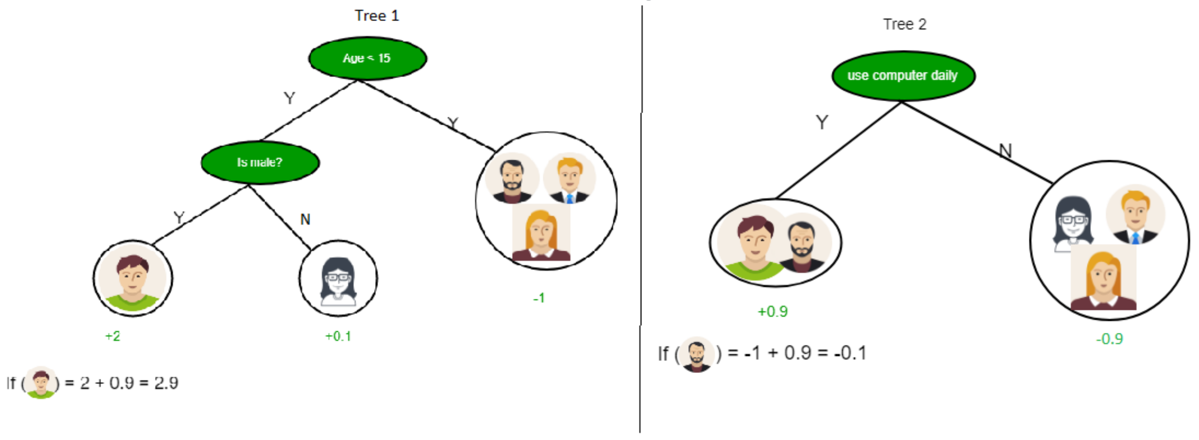
Decision Tree Introduction (Algorithms) with example

- Decision tree algorithm falls under the category of supervised learning. They can be used to solve both regression and classification problems.
- Decision tree uses the tree representation to solve the problem in which each leaf node corresponds to a class label and attributes are represented on the internal node of the tree.
- We can represent any boolean function on discrete attributes using the decision tree.


Input: Age, Gender, Occupation, . .
Does the person likes computer games
Age < 15
Y
Y
Is male?
Y
N
-1
Prediction score in each leaf
+2
+0.1

Below are some assumptions that we made while using decision tree:

- At the beginning, we consider the whole training set as the root.
- Feature values are preferred to be categorical. If the values are continuous then they are discretized prior to building the model.
- On the basis of attribute values records are distributed recursively.
- We use statistical methods for ordering attributes as root or the internal node.



As you can see from the above image that Decision Tree works on the Sum of Product form which is also known as *Disjunctive Normal Form*. In the above image, we are predicting the use of computer in the daily life of the people.
In Decision Tree the major challenge is to identification of the attribute for the root node in each level. This process is known as attribute selection. We have two popular attribute selection measures:

1. Information Gain
2. Gini Index

1. Information Gain

When we use a node in a decision tree to partition the training instances into smaller subsets the entropy changes. Information gain is a measure of this change in entropy. ***Definition***: Suppose S is a set of instances, A is an attribute, S_v is the subset of S with A = v, and Values (A) is the set of all possible values of A, then

Entropy
Entropy is the measure of uncertainty of a random variable, it characterizes the impurity of an arbitrary collection of examples. The higher the entropy more the information content.
Definition: Suppose S is a set of instances, A is an attribute, S_v is the subset of S with A = v, and Values (A) is the set of all possible values of A, then

Example:
For the set X = {a,a,a,b,b,b,b,b}

Total instances: 8

Instances of b: 5

Instances of a: 3

$$= -[0.375 * (-1.415) + 0.625 * (-0.678)]$$

$$=-(-0.53-0.424)$$

$$= 0.954$$

Building Decision Tree using Information Gain

The essentials:

- Start with all training instances associated with the root node
- Use info gain to choose which attribute to label each node with
- *Note:* No root-to-leaf path should contain the same discrete attribute twice
- Recursively construct each subtree on the subset of training instances that would be classified down that path in the tree.

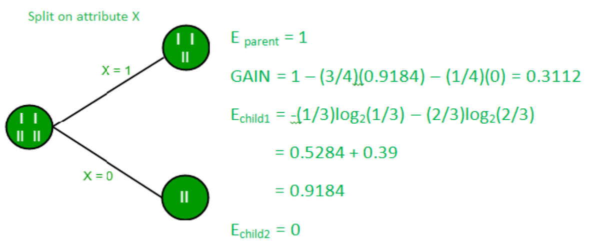
The border cases:

- If all positive or all negative training instances remain, label that node "yes" or "no" accordingly
- If no attributes remain, label with a majority vote of training instances left at that node
- If no instances remain, label with a majority vote of the parent's training instances

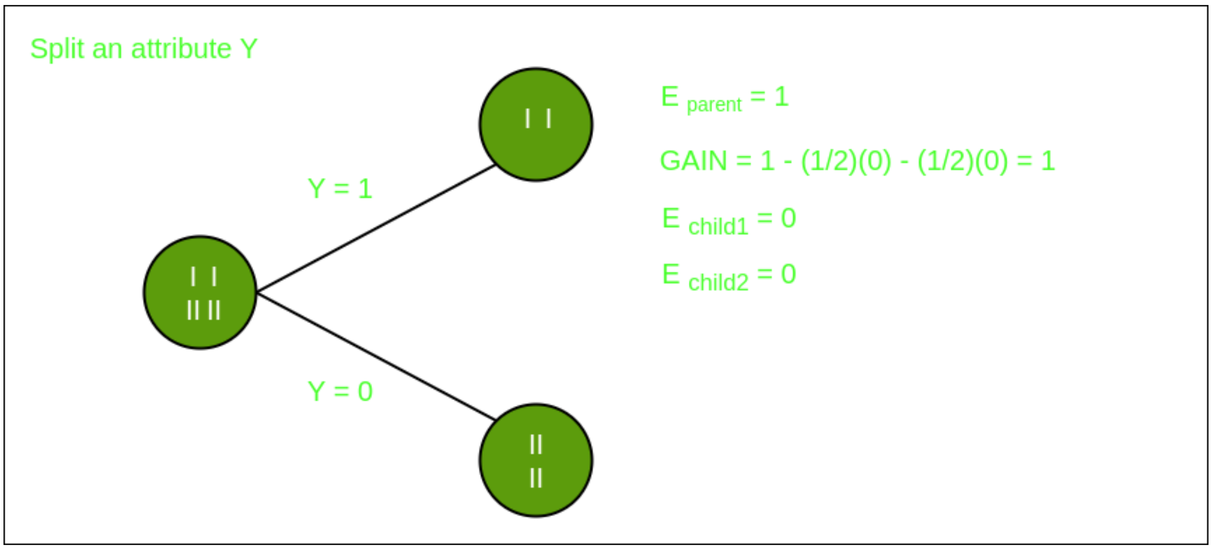
Example:
Now, lets draw a Decision Tree for the following data using Information gain.
Training set: 3 features and 2 classes

X	Y	Z	C
1	1	1	I
1	1	0	I
0	0	1	II
1	0	0	II

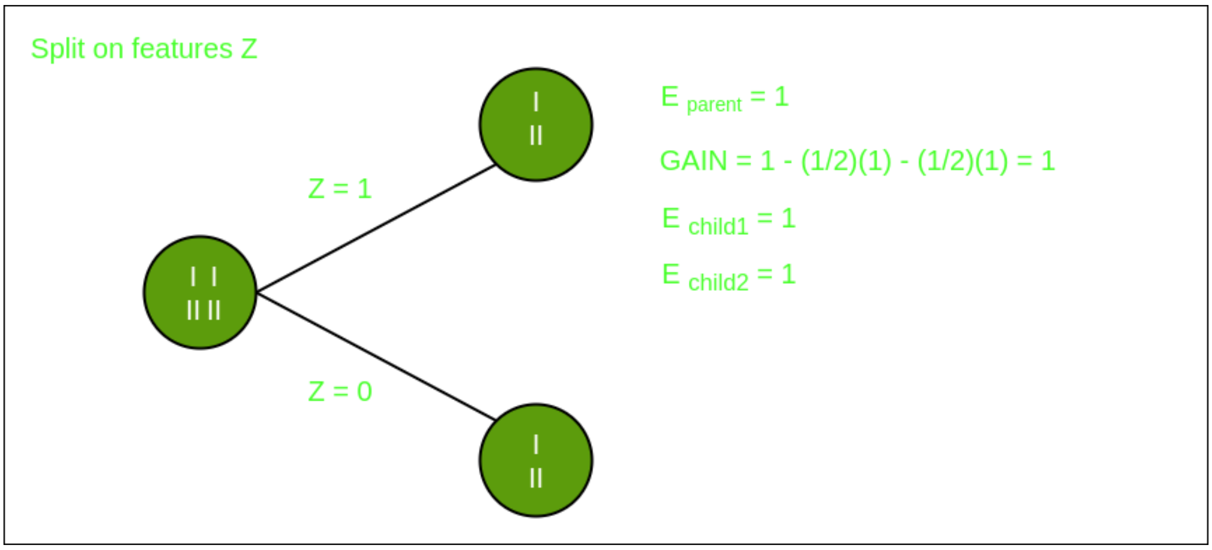
Here, we have 3 features and 2 output classes.
To build a decision tree using Information gain. We will take each of the feature and calculate the information for each feature.


Split on attribute X
X = 1
X = 0
E parent = 1
GAIN = 1 – (3/4)(0.9184) – (1/4)(0) = 0.3112
Echild1 = -(1/3)log2(1/3) – (2/3)log2(2/3)
= 0.5284 + 0.39
= 0.9184
Echild2 = 0

Split on feature X

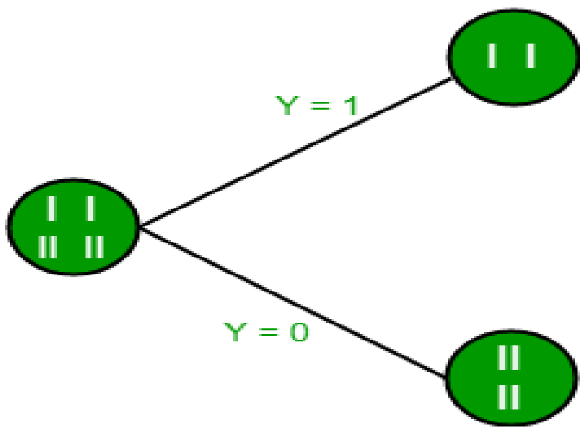


Split on feature Y



Split on feature Z

From the above images we can see that the information gain is maximum when we make a split on feature Y. So, for the root node best suited feature is feature Y. Now we can see that while splitting the dataset by feature Y, the child contains pure subset of the target variable. So we don't need to further split the dataset. The final tree for the above dataset would be look like this:



2. Gini Index

- Gini Index is a metric to measure how often a randomly chosen element would be incorrectly identified.
- It means an attribute with lower Gini index should be preferred.
- Sklearn supports "Gini" criteria for Gini Index and by default, it takes "gini" value.
- The Formula for the calculation of the Gini Index is given below.

Example:
Lets consider the dataset in the image below and draw a decision tree using gini index.

Index	A	B	C	D	E
1	4.8	3.4	1.9	0.2	positive
2	5	3	1.6	1.2	positive
3	5	3.4	1.6	0.2	positive
4	5.2	3.5	1.5	0.2	positive
5	5.2	3.4	1.4	0.2	positive
6	4.7	3.2	1.6	0.2	positive
7	4.8	3.1	1.6	0.2	positive
8	5.4	3.4	1.5	0.4	positive
9	7	3.2	4.7	1.4	negative
10	6.4	3.2	4.7	1.5	negative
11	6.9	3.1	4.9	1.5	negative

Index	A	B	C	D	E
12	5.5	2.3	4	1.3	negative
13	6.5	2.8	4.6	1.5	negative
14	5.7	2.8	4.5	1.3	negative
15	6.3	3.3	4.7	1.6	negative
16	4.9	2.4	3.3	1	negative

In the dataset above there are 5 attributes from which attribute E is the predicting feature which contains 2(Positive & Negative) classes. We have an equal proportion for both the classes. In Gini Index, we have to choose some random values to categorize each attribute. These values for this dataset are:

A	B	C	D
>= 5	>= 3.0	>= 4.2	>= 1.4
< 5	< 3.0	< 4.2	< 1.4

Calculating Gini Index for Var A:

Value >= 5: 12

Attribute A >= 5 & class = positive:

Attribute A >= 5 & class = negative:

Gini(5, 7) = 1 –
Value < 5: 4

Attribute A < 5 & class = positive:

Attribute A < 5 & class = negative:
Gini(3, 1) = 1 –
By adding weight and sum each of the gini indices:

Calculating Gini Index for Var B:
Value >= 3: 12

Attribute B >= 3 & class = positive:

Attribute B >= 5 & class = negative:

Gini(5, 7) = 1 –
Value < 3: 4

Attribute A < 3 & class = positive:

Attribute A < 3 & class = negative: Gini(3, 1) = 1 –

By adding weight and sum each of the gini indices:

Using the same approach we can calculate the Gini index for C and D attributes.

	Positive	Negative
For A|>= 5.0	5	7
|<5	3	1

Gini Index of A = 0.45825

	Positive	Negative
For B|>= 3.0	8	4
|< 3.0	0	4

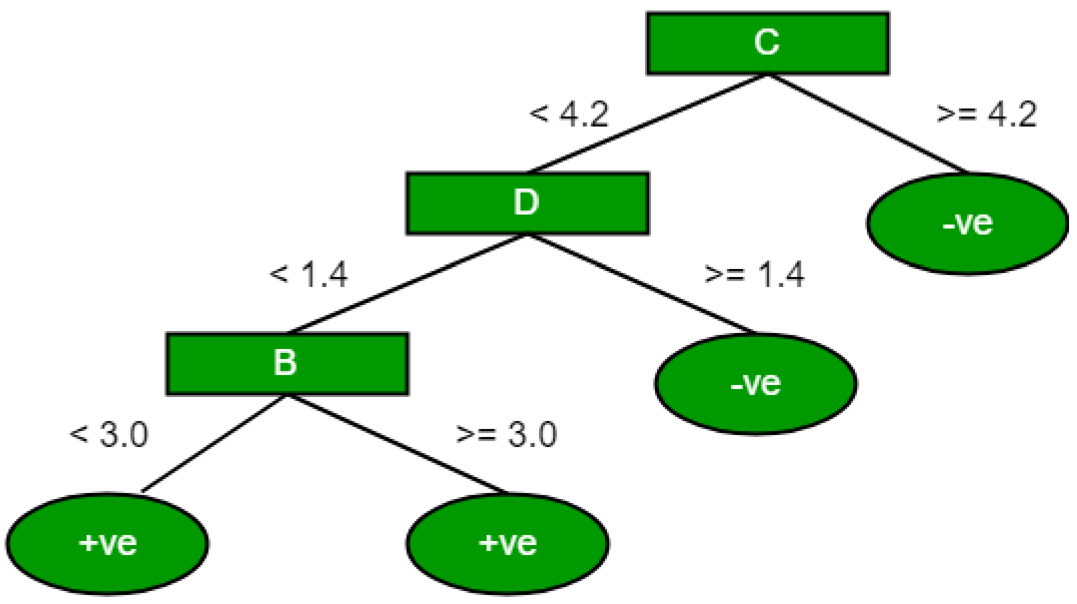
Gini Index of B= 0.3345

	Positive	Negative
For C|>= 4.2	0	6
|< 4.2	8	2

Gini Index of C= 0.2

	Positive	Negative
For D|>= 1.4	0	5
|< 1.4	8	3

Gini Index of D= 0.273


Decision tree for above dataset

The most notable types of decision tree algorithms are:-

1. **Iterative Dichotomiser 3 (ID3):** This algorithm uses Information Gain to decide which attribute is to be used classify the current subset of the data. For each level of the tree, information gain is calculated for the remaining data recursively.
2. **C4.5:** This algorithm is the successor of the ID3 algorithm. This algorithm uses either Information gain or Gain ratio to decide upon the classifying attribute. It is a direct improvement from the ID3 algorithm as it can handle both continuous and missing attribute values.
3. **Classification and Regression Tree(CART):** It is a dynamic learning algorithm which can produce a regression tree as well as a classification tree depending upon the dependent variable.

Naïve Bayes method

What is a classifier?

A classifier is a machine learning model that is used to discriminate different objects based on certain features.

Principle of Naive Bayes Classifier:

A Naive Bayes classifier is a probabilistic machine learning model that's used for classification task. The crux of the classifier is based on the Bayes theorem.

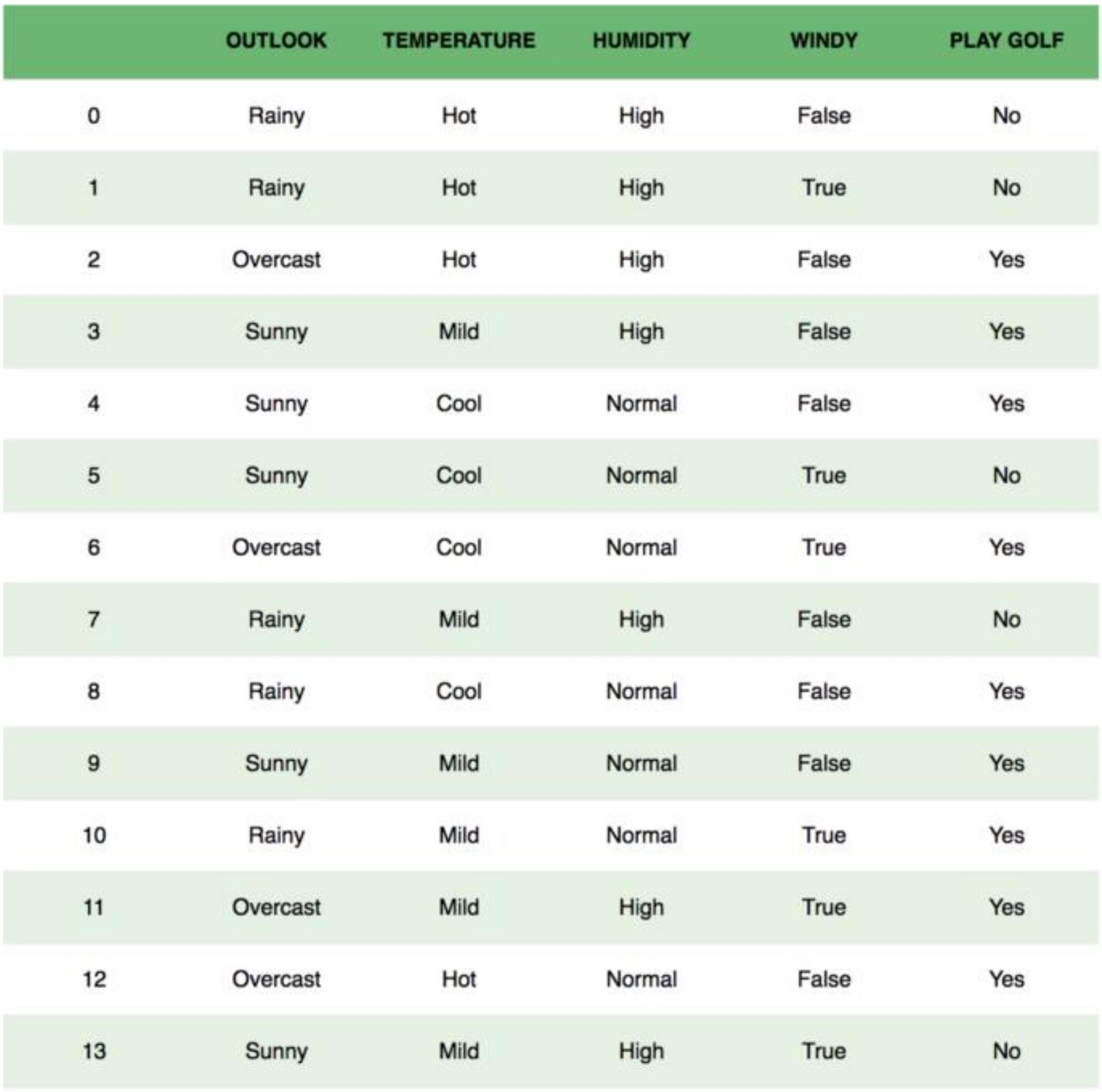
Bayes Theorem:

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)}$$

Using Bayes theorem, we can find the probability of **A** happening, given that **B** has occurred. Here, **B** is the evidence and **A** is the hypothesis. The assumption made here is that the predictors/features are independent. That is presence of one particular feature does not affect the other. Hence it is called naive.

Example:

Let us take an example to get some better intuition. Consider the problem of playing golf. The dataset is represented as below.



	OUTLOOK	TEMPERATURE	HUMIDITY	WINDY	PLAY GOLF
0	Rainy	Hot	High	False	No
1	Rainy	Hot	High	True	No
2	Overcast	Hot	High	False	Yes
3	Sunny	Mild	High	False	Yes
4	Sunny	Cool	Normal	False	Yes
5	Sunny	Cool	Normal	True	No
6	Overcast	Cool	Normal	True	Yes
7	Rainy	Mild	High	False	No
8	Rainy	Cool	Normal	False	Yes
9	Sunny	Mild	Normal	False	Yes
10	Rainy	Mild	Normal	True	Yes
11	Overcast	Mild	High	True	Yes
12	Overcast	Hot	Normal	False	Yes
13	Sunny	Mild	High	True	No

We classify whether the day is suitable for playing golf, given the features of the day. The columns represent these features and the rows represent individual entries. If we take the first row of the dataset, we can observe that is not suitable for playing golf if the outlook is rainy, temperature is hot, humidity is high and it is not windy. We make two assumptions here, one as stated above we consider that these predictors are independent. That is, if the temperature is hot, it does not necessarily mean that the humidity is high. Another assumption made here is that all the predictors have an equal effect on the outcome. That is, the day being windy does not have more importance in deciding to play golf or not.

According to this example, Bayes theorem can be rewritten as:

$$P(y|X) = \frac{P(X|y)P(y)}{P(X)}$$

The variable **y** is the class variable(play golf), which represents if it is suitable to play golf or not given the conditions. Variable **X** represent the parameters/features.

X is given as,

$$X = (x_1, x_2, x_3, ..., x_n)$$

Here x_1,x_2….x_n represent the features, i.e they can be mapped to outlook, temperature, humidity and windy. By substituting for **X** and expanding using the chain rule we get,

$$P(y|x_1, ..., x_n) = \frac{P(x_1|y)P(x_2|y)...P(x_n|y)P(y)}{P(x_1)P(x_2)...P(x_n)}$$

Now, you can obtain the values for each by looking at the dataset and substitute them into the equation. For all entries in the dataset, the denominator does not change, it remain static. Therefore, the denominator can be removed and a proportionality can be introduced.

$$P(y|x_1, ..., x_n) \propto P(y) \prod_{i=1}^{n} P(x_i|y)$$

In our case, the class variable(**y**) has only two outcomes, yes or no. There could be cases where the classification could be multivariate. Therefore, we need to find the class **y** with maximum probability.

$$y = argmax_y P(y) \prod_{i=1}^{n} P(x_i|y)$$

Using the above function, we can obtain the class, given the predictors.

Types of Naive Bayes Classifier:

Multinomial Naive Bayes:

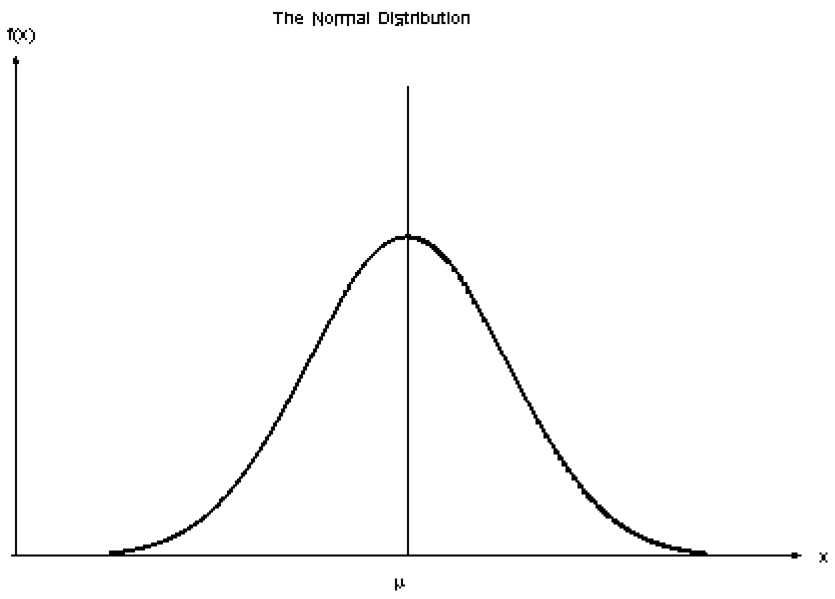
This is mostly used for document classification problem, i.e whether a document belongs to the category of sports, politics, technology etc. The features/predictors used by the classifier are the frequency of the words present in the document.

Bernoulli Naive Bayes:

This is similar to the multinomial naive bayes but the predictors are boolean variables. The parameters that we use to predict the class variable take up only values yes or no, for example if a word occurs in the text or not.

Gaussian Naive Bayes:

When the predictors take up a continuous value and are not discrete, we assume that these values are sampled from a gaussian distribution.



Gaussian Distribution(Normal Distribution)

Since the way the values are present in the dataset changes, the formula for conditional probability changes to,

$$P(x_i|y) = \frac{1}{\sqrt{2\pi\sigma_y^2}} exp\left(-\frac{(x_i-\mu_y)^2}{2\sigma_y^2}\right)$$

Estimating predictive accuracy of classification method:

Classifier Accuracy

Evaluating & estimating the accuracy of classifiers is important in that it allows one to evaluate how accurately a given classifier will label future data, that, is, data on which the classifier has not been trained.

For example, suppose you used data from previous sales to train a classifier to predict customer purchasing behavior.

You would like an estimate of how accurately the classifier can predict the purchasing behavior of future customers, that is, future customer data on which the classifier has not been trained.

Accuracy estimates to help in the comparison of different classifiers.

Methods To Find Accuracy Of The Classifiers

- Holdout Method
- Random Subsampling
- K-fold Cross-Validation
- Bootstrap Methods

Estimating The Classifier Accuracy

Holdout & Random Subsampling

The holdout method is what we have alluded to so far in our discussions about accuracy.

In this method, the given data are randomly partitioned into two independent sets, a training set, and a test set.

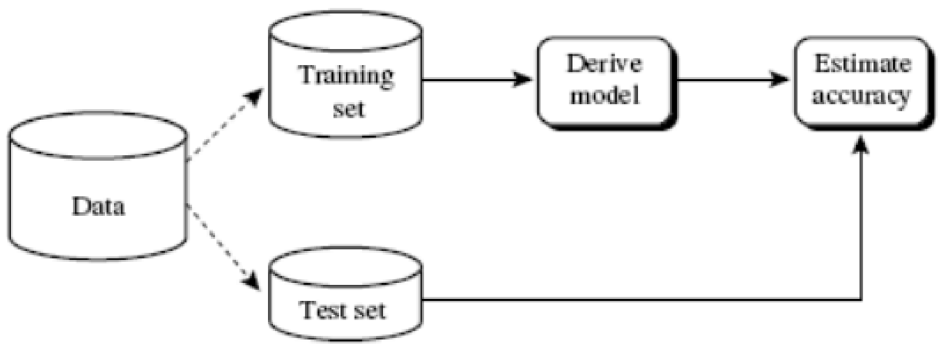
Typically, two-thirds of the data are allocated to the training set, and the remaining one-third is allocated to the test set.

The training set is used to derive the model, whose accuracy is estimated with the test set.

The estimate is pessimistic because only a portion of the initial data is used to derive the model.

Random subsampling is a variation of the holdout method in which the holdout method is repeated k times.

The overall accuracy estimate is taken as the average of the accuracies obtained from each iteration. (For prediction, we can take the average of the predictor error rates.)



Estimating accuracy with the holdout method.

Cross-Validation

In k-**fold cross-validation**, the initial data are randomly partitioned into k mutually exclusive subsets or "folds," D1, D2,....., Dk, each of approximately equal size.

Training and testing are performed k times. In iteration i, partition Di is reserved as the test set, and the remaining partitions are collectively used to train the model.

That is, in the first iteration, subsets D2..., Dk collectively serves as the training set to obtain a first model, which is tested on D1; the second iteration is trained on subsets D1, D3,..., Dk and tested on D2; and so on.

Unlike the holdout and random subsampling methods above, here, each sample is used the same number of times for training and once for testing.

For classification, the accuracy estimate is the overall number of correct classifications from the k iterations, divided by the total number of tuples in the initial data.

For prediction, the error estimate can be computed as the total loss from the k iterations, divided by the total number of initial tuples.

Bootstrapping

Unlike the accuracy estimation methods mentioned above, the bootstrap method samples the given training tuples uniformly with replacement.

That is, each time a tuple is selected, it is equally likely to be selected again and readded to the training set.

For instance, imagine a machine that randomly selects tuples for our training set. In sampling with replacement, the machine is allowed to select the same tuple more than once.

Ensemble Methods - Increasing The Accuracy

Are there general strategies for improving classifier and predictor accuracy?

YES, Bagging and boosting are two such techniques.

Bagging

We first take an intuitive look at how bagging works as a method of increasing accuracy.

For ease of explanation, we will assume at first that our model is a classifier. Suppose that you are a patient and would like to have a diagnosis made based on your symptoms.

Instead of asking one doctor, you may choose to ask several. If a certain diagnosis occurs more than any of the others, you may choose this as the final or best diagnosis.

That is, the final diagnosis is made based on a majority vote, where each doctor gets an equal vote. Now replace each doctor by a classifier, and you have the basic idea behind bagging.

Intuitively, a majority vote made by a large group of doctors may be more reliable than a majority vote made by a small group.

Given a set, D, of d tuples, bagging works as follows.

For iteration i (i = 1, 2,..., k), a training set, Di, of d tuples is sampled with replacement from the original set of tuples, D.

Note that the term bagging stands for bootstrap aggregation.

A classifier model, Mi, is learned for each training set, Di.

To classify an unknown tuple, X, each classifier, Mi, returns its class prediction, which counts as one vote.

The bagged classifier, M, counts the votes and assigns the class with the most votes to X.

Bagging can be applied to the prediction of continuous values by taking the average value of each prediction for a given test tuple.

The bagged classifier often has significantly greater accuracy than a single classifier derived from D, the original training data.

Boosting

We now look at the ensemble method of boosting. As in the previous section, suppose that as a patient, you have certain symptoms.

Instead of consulting one doctor, you choose to consult several.

Suppose you assign weights to the value or worth of each doctor’s diagnosis, based on the accuracies of previous diagnoses they have made.

The final diagnosis is then a combination of the weighted diagnoses. This is the essence behind boosting.

In boosting, weights are assigned to each training tuple.

A series of k classifiers is iteratively learned. After a classifier Mi is learned, the weights are updated to allow the subsequent classifier, Mi+1, to “pay more attention” to the training tuples that were misclassified by Mi.

The final boosted classifier, M, combines the votes of each individual classifier, where the weight of each classifier’s vote is a function of its accuracy.

The boosting algorithm can be extended for the prediction of continuous values.