K-nearest neighbors (KNN)

Introduction

K-nearest neighbors (KNN) algorithm is a type of supervised ML algorithm which can be used for both classification as well as regression predictive problems. However, it is mainly used for classification predictive problems in industry. The following two properties would define KNN well −

- **Lazy learning algorithm** − KNN is a lazy learning algorithm because it does not have a specialized training phase and uses all the data for training while classification.
- **Non-parametric learning algorithm** − KNN is also a non-parametric learning algorithm because it doesn't assume anything about the underlying data.

Working of KNN Algorithm

K-nearest neighbors (KNN) algorithm uses 'feature similarity' to predict the values of new datapoints which further means that the new data point will be assigned a value based on how closely it matches the points in the training set. We can understand its working with the help of following steps −

Step 1 − For implementing any algorithm, we need dataset. So during the first step of KNN, we must load the training as well as test data.

Step 2 − Next, we need to choose the value of K i.e. the nearest data points. K can be any integer.

Step 3 − For each point in the test data do the following −

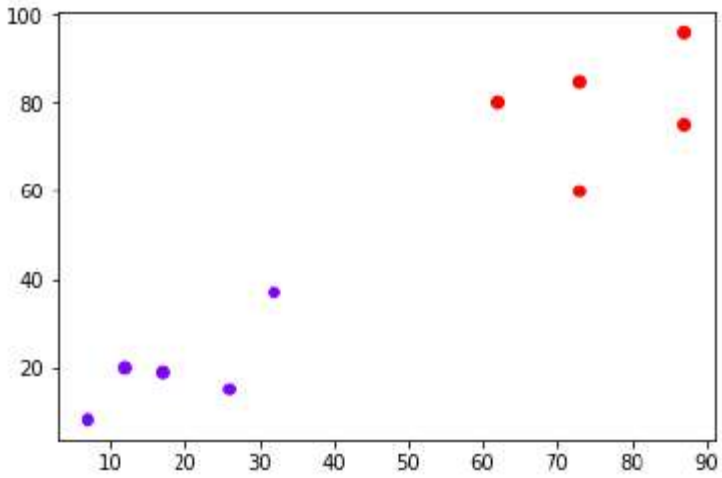
- **3.1** − Calculate the distance between test data and each row of training data with the help of any of the method namely: Euclidean, Manhattan or Hamming distance. The most commonly used method to calculate distance is Euclidean.
- **3.2** − Now, based on the distance value, sort them in ascending order.
- **3.3** − Next, it will choose the top K rows from the sorted array.
- **3.4** − Now, it will assign a class to the test point based on most frequent class of these rows.

Step 4 − End

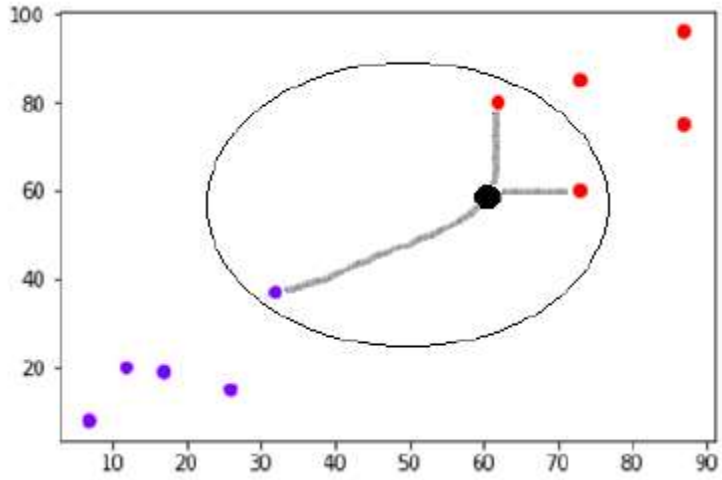
Example

The following is an example to understand the concept of K and working of KNN algorithm −

Suppose we have a dataset which can be plotted as follows −



Now, we need to classify new data point with black dot (at point 60,60) into blue or red class. We are assuming K = 3 i.e. it would find three nearest data points. It is shown in the next diagram −



We can see in the above diagram the three nearest neighbors of the data point with black dot. Among those three, two of them lies in Red class hence the black dot will also be assigned in red class.

Pros and Cons of KNN

Pros

- It is very simple algorithm to understand and interpret.

- It is very useful for nonlinear data because there is no assumption about data in this algorithm.
- It is a versatile algorithm as we can use it for classification as well as regression.
- It has relatively high accuracy but there are much better supervised learning models than KNN.

Cons

- It is computationally a bit expensive algorithm because it stores all the training data.
- High memory storage required as compared to other supervised learning algorithms.
- Prediction is slow in case of big N.
- It is very sensitive to the scale of data as well as irrelevant features.

Applications of KNN

The following are some of the areas in which KNN can be applied successfully −

Banking System

KNN can be used in banking system to predict weather an individual is fit for loan approval? Does that individual have the characteristics similar to the defaulters one?

Calculating Credit Ratings

KNN algorithms can be used to find an individual’s credit rating by comparing with the persons having similar traits.

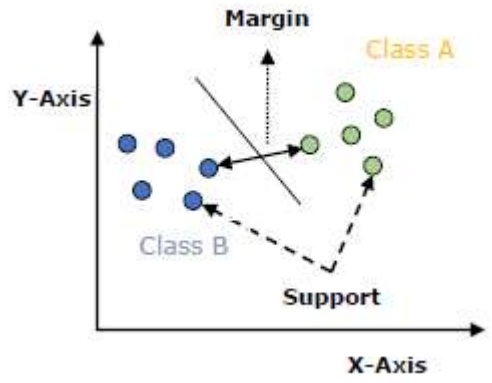
Support vector machines (SVMs)

Introduction to SVM

Support vector machines (SVMs) are powerful yet flexible supervised machine learning algorithms which are used both for classification and regression. But generally, they are used in classification problems. In 1960s, SVMs were first introduced but later they got refined in 1990. SVMs have their unique way of implementation as compared to other machine learning algorithms. Lately, they are extremely popular because of their ability to handle multiple continuous and categorical variables.

Working of SVM

An SVM model is basically a representation of different classes in a hyperplane in multidimensional space. The hyperplane will be generated in an iterative manner by SVM so that the error can be minimized. The goal of SVM is to divide the datasets into classes to find a maximum marginal hyperplane (MMH).



The followings are important concepts in SVM −

- **Support Vectors** − Datapoints that are closest to the hyperplane is called support vectors. Separating line will be defined with the help of these data points.
- **Hyperplane** − As we can see in the above diagram, it is a decision plane or space which is divided between a set of objects having different classes.
- **Margin** − It may be defined as the gap between two lines on the closet data points of different classes. It can be calculated as the perpendicular distance from the line to the support vectors. Large margin is considered as a good margin and small margin is considered as a bad margin.

The main goal of SVM is to divide the datasets into classes to find a maximum marginal hyperplane (MMH) and it can be done in the following two steps −

- First, SVM will generate hyperplanes iteratively that segregates the classes in best way.
- Then, it will choose the hyperplane that separates the classes correctly.

SVM Kernels

In practice, SVM algorithm is implemented with kernel that transforms an input data space into the required form. SVM uses a technique called the kernel trick in which kernel takes a low dimensional input space and transforms it into a higher dimensional space. In simple words, kernel converts non-separable problems into separable problems by adding more dimensions to it. It makes SVM more powerful, flexible and accurate. The following are some of the types of kernels used by SVM.

Linear Kernel

It can be used as a dot product between any two observations. The formula of linear kernel is as below −

$$K(x, x_i) = sum(x * x_i)$$

From the above formula, we can see that the product between two vectors say x & xi is the sum of the multiplication of each pair of input values.

Polynomial Kernel

It is more generalized form of linear kernel and distinguish curved or nonlinear input space. Following is the formula for polynomial kernel −

$$k(X, Xi) = 1 + sum(X * X_i)^d$$

Here d is the degree of polynomial, which we need to specify manually in the learning algorithm.

Radial Basis Function (RBF) Kernel

RBF kernel, mostly used in SVM classification, maps input space in indefinite dimensional space. Following formula explains it mathematically −

$$K(x, xi) = exp(-gamma^{*} sum(x - xi\text{-}2))$$

Here, *gamma* ranges from 0 to 1. We need to manually specify it in the learning algorithm. A good default value of *gamma* is 0.1.

As we implemented SVM for linearly separable data, we can implement it in Python for the data that is not linearly separable. It can be done by using kernels.

Pros and Cons of SVM Classifiers

Pros of SVM classifiers

SVM classifiers offers great accuracy and work well with high dimensional space. SVM classifiers basically use a subset of training points hence in result uses very less memory.

Cons of SVM classifiers

They have high training time hence in practice not suitable for large datasets. Another disadvantage is that SVM classifiers do not work well with overlapping classes.

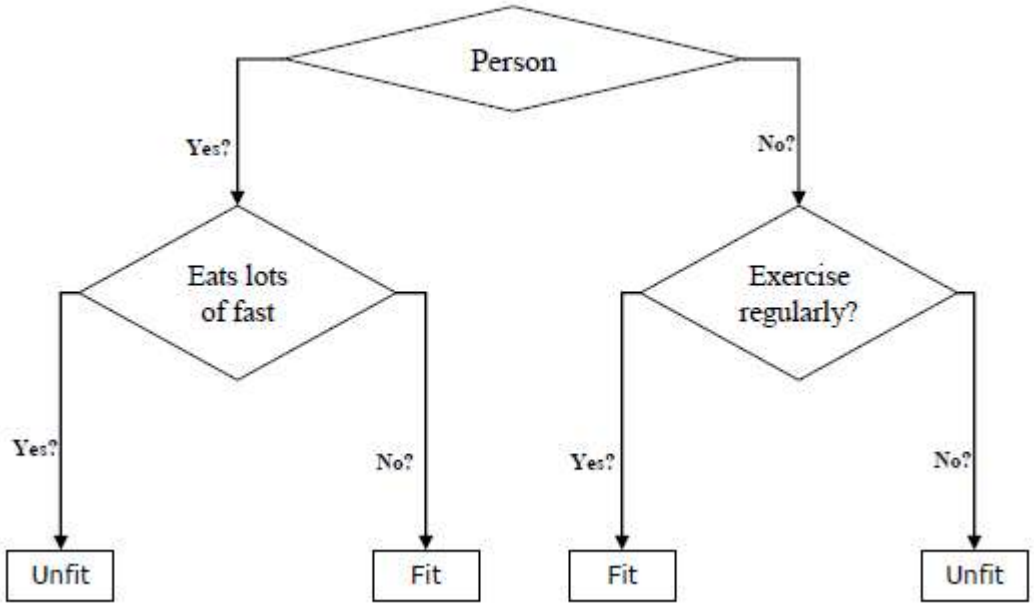
Decision Tree

Introduction to Decision Tree

In general, Decision tree analysis is a predictive modelling tool that can be applied across many areas. Decision trees can be constructed by an algorithmic approach that can split the dataset in different ways based on different conditions. Decisions trees are the most powerful algorithms that falls under the category of supervised algorithms.

They can be used for both classification and regression tasks. The two main entities of a tree are decision nodes, where the data is split and leaves, where we got outcome. The example of a binary tree for predicting whether a person is fit

or unfit providing various information like age, eating habits and exercise habits, is given below −



In the above decision tree, the question are decision nodes and final outcomes are leaves. We have the following two types of decision trees.

- **Classification decision trees** − In this kind of decision trees, the decision variable is categorical. The above decision tree is an example of classification decision tree.
- **Regression decision trees** − In this kind of decision trees, the decision variable is continuous.

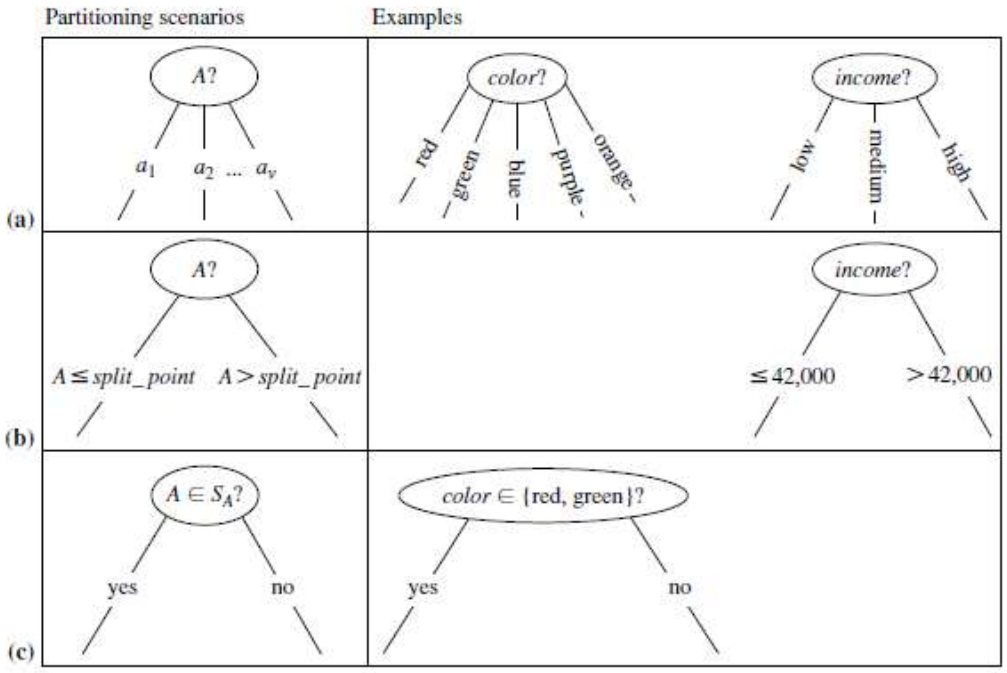
splitting criterion

The splitting criterion also tells us which branches to grow from node *N* with respect to the outcomes of the chosen test. More specifically, the splitting criterion indicates the **splitting attribute** and may also indicate either a **split-point** or a **splitting subset**.

1. *A is discrete-valued:* In this case, the outcomes of the test at node *N* correspond directly to the known values of *A*
2. A is continuous-valued: In this case, the test at node *N* has two possible outcomes, corresponding to the conditions *A* _ *split point* and *A* > *split point*, respectively,

where *split point* is the split-point returned by *Attribute selection method* as part of the splitting criterion.

3. *A is discrete-valued* and a *binary tree*



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
3.Gain Ratio

Information Gain

Let node *N* represent or hold the tuples of partition *D*. The attribute with the highest information gain is chosen as the splitting attribute for node *N*. This attribute minimizes the information needed to classify the tuples in the resulting partitions and reflects the least randomness or “impurity” in these partitions.

Such an approach minimizes the expected number of tests needed to classify a given tuple and guarantees that a simple (but not necessarily the simplest) tree is found.

The expected information needed to classify a tuple in *D* is given by

$$Info(D) = -\sum_{i=1}^{m} p_i \log_2(p_i),$$

How much more information would we still need (after the partitioning) to arrive at an exact classification? This amount is measured by

$$Info_A(D) = \sum_{j=1}^{v} \frac{|D_j|}{|D|} \times Info(D_j).$$

Information gain is defined as the difference between the original information requirement and the new requirement

That is,

$$Gain(A) = Info(D) - Info_A(D).$$

Random Forest Algorithm

Random forest is a supervised learning algorithm which is used for both classification as well as regression. But however, it is mainly used for classification problems. As we know that a forest is made up of trees and more trees means more robust forest. Similarly, random forest algorithm creates decision trees on data samples and then gets the prediction from each of them and finally selects the best solution by means of voting. It is an ensemble method which is better than a single decision tree because it reduces the over-fitting by averaging the result.

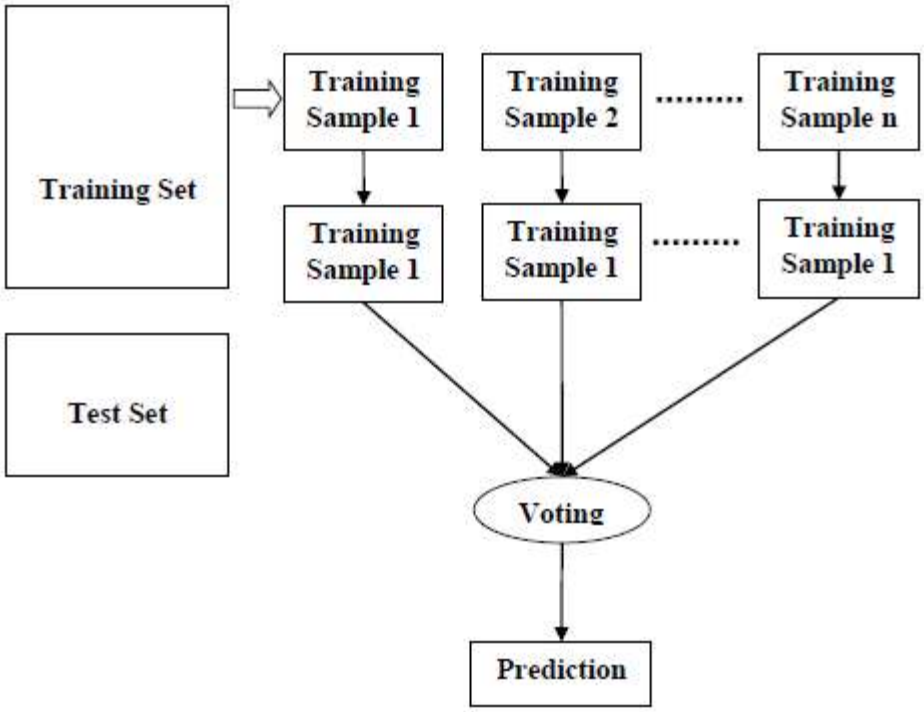
Working of Random Forest Algorithm

We can understand the working of Random Forest algorithm with the help of following steps −

- **Step 1** − First, start with the selection of random samples from a given dataset.
- **Step 2** − Next, this algorithm will construct a decision tree for every sample. Then it will get the prediction result from every decision tree.
- **Step 3** − In this step, voting will be performed for every predicted result.

- **Step 4** − At last, select the most voted prediction result as the final prediction result.

The following diagram will illustrate its working −



Pros

The following are the advantages of Random Forest algorithm −

- It overcomes the problem of overfitting by averaging or combining the results of different decision trees.
- Random forests work well for a large range of data items than a single decision tree does.
- Random forest has less variance then single decision tree.
- Random forests are very flexible and possess very high accuracy.
- Scaling of data does not require in random forest algorithm. It maintains good accuracy even after providing data without scaling.
- Random Forest algorithms maintains good accuracy even a large proportion of the data is missing.

Cons

The following are the disadvantages of Random Forest algorithm −

- Complexity is the main disadvantage of Random forest algorithms.
- Construction of Random forests are much harder and time-consuming than decision trees.
- More computational resources are required to implement Random Forest algorithm.

CONFUSION MATRIX

It is the easiest way to measure the performance of a classification problem where the output can be of two or more type of classes. A confusion matrix is nothing but a table with two dimensions viz. “Actual” and “Predicted” and furthermore, both the dimensions have “True Positives (TP)”, “True Negatives (TN)”, “False Positives (FP)”, “False Negatives (FN)” as shown below −

The explanation of the terms associated with confusion matrix are as follows −

- **True Positives (TP)** − It is the case when both actual class & predicted class of data point is 1.
- **True Negatives (TN)** − It is the case when both actual class & predicted class of data point is 0.
- **False Positives (FP)** − It is the case when actual class of data point is 0 & predicted class of data point is 1.
- **False Negatives (FN)** − It is the case when actual class of data point is 1 & predicted class of data point is 0.

		Predicted class		
		yes	*no*	Total
Actual class	*yes*	*TP*	*FN*	*P*
	no	*FP*	*TN*	*N*
	Total	*P′*	*N′*	$P + N$

EXAMPLE

Classes	buys_computer = yes	buys_computer = no	Total	Recognition (%)
buys_computer = yes	6954	46	7000	99.34
buys_computer = no	412	2588	3000	86.27
Total	7366	2634	10,000	95.42

Metrics for Evaluating Classifier Performance

Measure	Formula
accuracy, recognition rate	$\frac{TP+TN}{P+N}$
error rate, misclassification rate	$\frac{FP+FN}{P+N}$
sensitivity, true positive rate, recall	$\frac{TP}{P}$
specificity, true negative rate	$\frac{TN}{N}$
precision	$\frac{TP}{TP+FP}$
F, F_1, F-score, harmonic mean of precision and recall	$\frac{2 \times precision \times recall}{precision + recall}$
F_β, where β is a non-negative real number	$\frac{(1+\beta^2) \times precision \times recall}{\beta^2 \times precision + recall}$

The **accuracy** of a classifier on a given test set is the percentage of test set tuples that are correctly classified by the classifier. That is,

$$accuracy = \frac{TP + TN}{P + N}.$$

error rate or **misclassification rate** of a classifier, *M*, which is simply 1-*accuracy*(M), where *accuracy*(M) is the accuracy of *M*. This also can be computed as

$$error\ rate = \frac{FP + FN}{P + N}.$$

We now consider the **class imbalance problem**, where the main class of interest is rare. That is, the data set distribution reflects a significant majority of the negative class and a minority positive class. For example, in fraud detection applications, the class of interest (or positive class) is *"fraud,"* which occurs much less frequently. The **sensitivity** and **specificity** measures can be used to measure accuracy.

$$sensitivity = \frac{TP}{P}$$

$$specificity = \frac{TN}{N}.$$

It can be shown that accuracy is a function of sensitivity and specificity:

$$accuracy = sensitivity\frac{P}{(P+N)} + specificity\frac{N}{(P+N)}.$$

The *precision* and *recall* measures are also widely used in classification. **Precision** can be thought of as a measure of *exactness* (i.e., what percentage of tuples labeled as positive are actually such), whereas **recall** is a measure of *completeness* (what percentage of positive tuples are labeled as such). If recall seems familiar, that's because it is the same as sensitivity (or the *true positive rate*). These measures can be computed as

$$precision = \frac{TP}{TP + FP}$$

$$recall = \frac{TP}{TP + FN} = \frac{TP}{P}.$$

An alternative way to use precision and recall is to combine them into a single measure. This is the approach of the *F* measure (also known as the *F*1 score or *F*-score)

$$F = \frac{2 \times precision \times recall}{precision + recall}$$

$$F_\beta = \frac{(1 + \beta^2) \times precision \times recall}{\beta^2 \times precision + recall},$$

Naïve Bayes Classifier 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.

Example

RID	age	income	student	credit_rating	Class: buys_computer
1	youth	high	no	fair	no
2	youth	high	no	excellent	no
3	middle_aged	high	no	fair	yes
4	senior	medium	no	fair	yes
5	senior	low	yes	fair	yes
6	senior	low	yes	excellent	no
7	middle_aged	low	yes	excellent	yes
8	youth	medium	no	fair	no
9	youth	low	yes	fair	yes
10	senior	medium	yes	fair	yes
11	youth	medium	yes	excellent	yes
12	middle_aged	medium	no	excellent	yes
13	middle_aged	high	yes	fair	yes
14	senior	medium	no	excellent	no

The tuple we wish to classify is

$$\boldsymbol{X} = (age = youth,\ income = medium,\ student = yes,\ credit\_rating = fair)$$

We need to maximize $P(\boldsymbol{X}|C_i)P(C_i)$, for $i = 1, 2$. $P(C_i)$, the prior probability of each class, can be computed based on the training tuples:

$$P(buys\_computer = yes) = 9/14 = 0.643$$
$$P(buys\_computer = no) = 5/14 = 0.357$$

To compute $P(\boldsymbol{X}|C_i)$, for $i = 1, 2$, we compute the following conditional probabilities:

$$P(age = youth \mid buys\_computer = yes) = 2/9 = 0.222$$
$$P(age = youth \mid buys\_computer = no) = 3/5 = 0.600$$
$$P(income = medium \mid buys\_computer = yes) = 4/9 = 0.444$$
$$P(income = medium \mid buys\_computer = no) = 2/5 = 0.400$$
$$P(student = yes \mid buys\_computer = yes) = 6/9 = 0.667$$
$$P(student = yes \mid buys\_computer = no) = 1/5 = 0.200$$
$$P(credit\_rating = fair \mid buys\_computer = yes) = 6/9 = 0.667$$
$$P(credit\_rating = fair \mid buys\_computer = no) = 2/5 = 0.400$$

Using these probabilities, we obtain

$$\begin{aligned} P(\boldsymbol{X}|buys\_computer = yes) &= P(age = youth \mid buys\_computer = yes) \\ &\quad \times P(income = medium \mid buys\_computer = yes) \\ &\quad \times P(student = yes \mid buys\_computer = yes) \\ &\quad \times P(credit\_rating = fair \mid buys\_computer = yes) \\ &= 0.222 \times 0.444 \times 0.667 \times 0.667 = 0.044. \end{aligned}$$

Similarly,

$$P(\boldsymbol{X}|buys\_computer = no) = 0.600 \times 0.400 \times 0.200 \times 0.400 = 0.019.$$

To find the class, C_i, that maximizes $P(\boldsymbol{X}|C_i)P(C_i)$, we compute

$$P(\boldsymbol{X}|buys\_computer = yes)P(buys\_computer = yes) = 0.044 \times 0.643 = 0.028$$
$$P(\boldsymbol{X}|buys\_computer = no)P(buys\_computer = no) = 0.019 \times 0.357 = 0.007$$

Therefore, the naïve Bayesian classifier predicts $buys\_computer = yes$ for tuple $\boldsymbol{X}$. ■

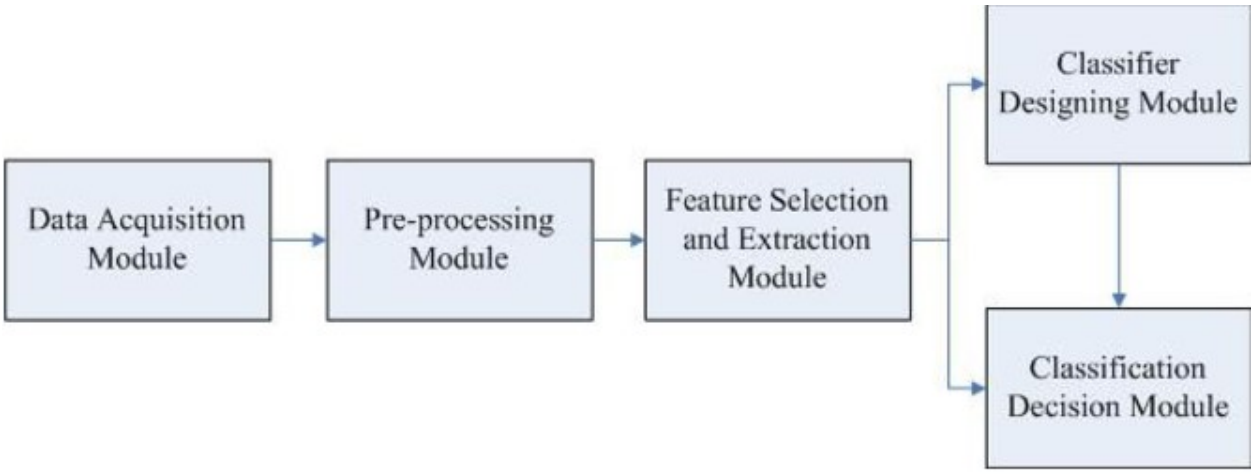
PATTERNS, FEATURES, PATTER REPRESENTATION

Pattern is everything around in this digital world. A pattern can either be seen physically or it can be observed mathematically by applying algorithms.

Example: The colors on the clothes, speech pattern etc. In computer science, a pattern is represented using vector feature values.

Pattern recognition is the process of recognizing patterns by using machine learning algorithm. Pattern recognition can be defined as the classification of data based on knowledge already gained or on statistical information extracted from patterns and/or their representation. One of the important aspects of the pattern recognition is its application potential.

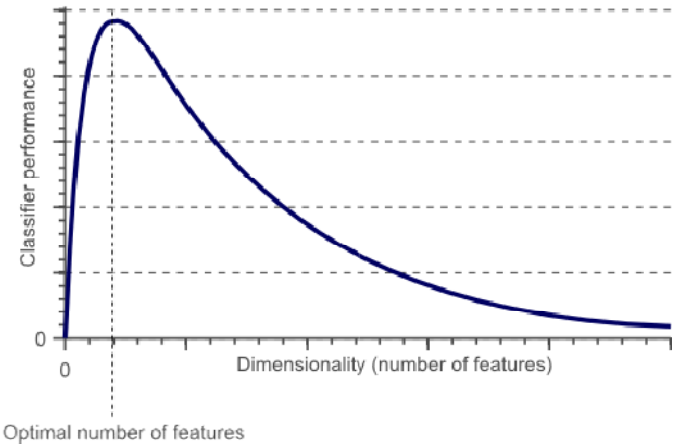
Examples: Speech recognition, speaker identification, multimedia document recognition. In a typical pattern recognition application, the raw data is processed and converted into a form that is amenable for a machine to use. Pattern recognition involves classification and cluster of patterns.



CURSE OF DIMENSIONALITY

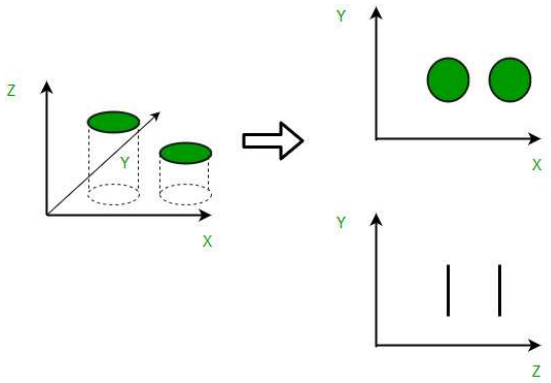
Handling the high-dimensional data is very difficult in practice, commonly known as the *curse of dimensionality*. If the dimensionality of the input dataset increases, any machine learning algorithm and model becomes more complex. As the number of features increases, the number of samples also gets increased proportionally, and the chance of overfitting also increases. If the machine learning model is trained on high-dimensional data, it becomes overfitted and results in poor performance.

Hence, it is often required to reduce the number of features, which can be done with dimensionality reduction.



DIMENSIONALITY REDUCTION

In machine learning classification problems, there are often too many factors on the basis of which the final classification is done. These factors are basically variables called features. The higher the number of features, the harder it gets to visualize the training set and then work on it. Sometimes, most of these features are correlated, and hence redundant. This is where dimensionality reduction algorithms come into play. Dimensionality reduction is the process of reducing the number of random variables under consideration, by obtaining a set of principal variables. It can be divided into feature selection and feature extraction.



Components of Dimensionality Reduction
There are two components of dimensionality reduction:

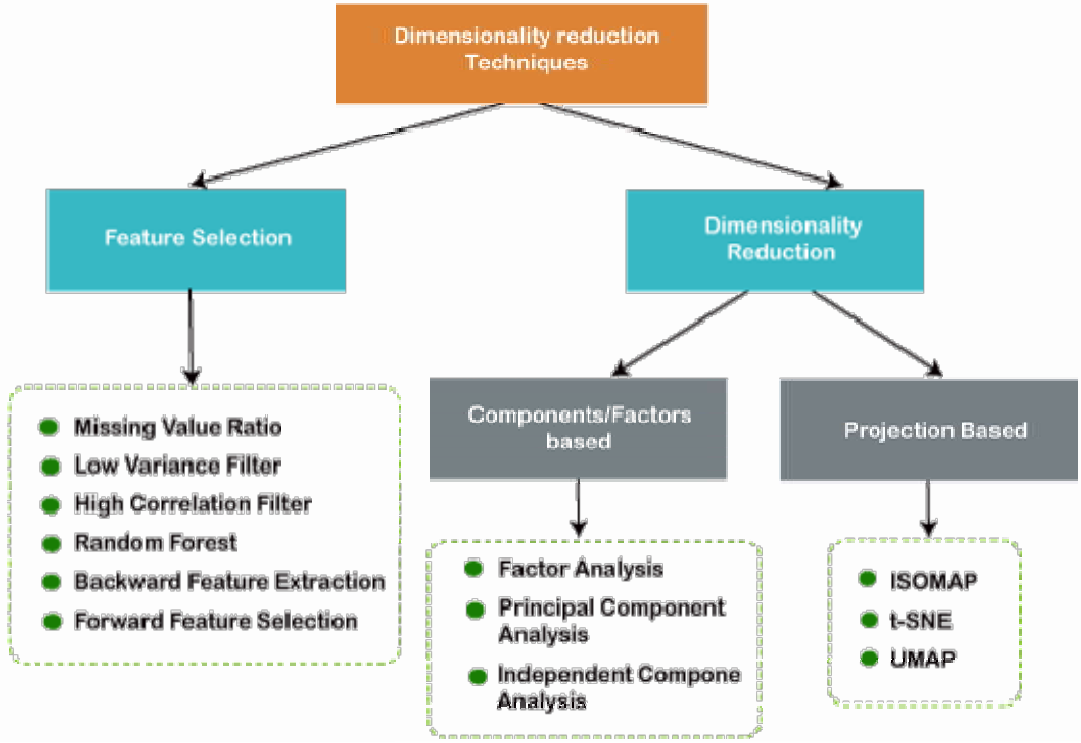
- Feature selection: In this, we try to find a subset of the original set of variables, or features, to get a smaller subset which can be used to model the problem. It usually involves three ways:

1. Filter
2. Wrapper
3. Embedded

- Feature extraction: This reduces the data in a high dimensional space to a lower dimension space, i.e. a space with lesser no. of dimensions.

  Methods of Dimensionality Reduction
  The various methods used for dimensionality reduction include:

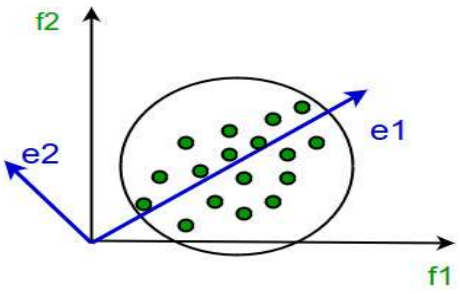
- Principal Component Analysis (PCA)
- Linear Discriminant Analysis (LDA)
- Generalized Discriminant Analysis (GDA)



-

Principal Component Analysis

This method was introduced by Karl Pearson. It works on a condition that while the data in a higher dimensional space is mapped to data in a lower dimension space, the variance of the data in the lower dimensional space should be maximum.



It involves the following steps:

- Construct the covariance matrix of the data.
- Compute the eigenvectors of this matrix.
- Eigenvectors corresponding to the largest eigenvalues are used to reconstruct a large fraction of variance of the original data.
  Hence, we are left with a lesser number of eigenvectors, and there might have been some data loss in the process. But, the most important variances should be retained by the remaining eigenvectors.

SUPERVISED AND UNSUPERVISED LEARNING

Supervised Learning	Unsupervised Learning
Supervised learning algorithms are trained using labeled data.	Unsupervised learning algorithms are trained using unlabeled data.
Supervised learning model takes direct feedback to check if it is predicting correct output or not.	Unsupervised learning model does not take any feedback.
Supervised learning model predicts the output.	Unsupervised learning model finds the hidden patterns in data.
In supervised learning, input data is provided to the model along with the output.	In unsupervised learning, only input data is provided to the model.
The goal of supervised learning is to train the model so that it can predict the output when it is given new data.	The goal of unsupervised learning is to find the hidden patterns and useful insights from the unknown dataset.
Supervised learning needs supervision to train the model.	Unsupervised learning does not need any supervision to train the model.

Supervised learning can be categorized in Classification and Regression problems.	Unsupervised Learning can be classified in Clustering and Associations problems.
Supervised learning model produces an accurate result.	Unsupervised learning model may give less accurate result as compared to supervised learning.
It includes various algorithms such as Linear Regression, Logistic Regression, Support Vector Machine, Multi-class Classification, Decision tree, Bayesian Logic, etc.	It includes various algorithms such as Clustering, KNN, and Apriori algorithm.

CLASSIFICATION—LINEAR AND NON-LINEAR

Classification Algorithms can be divided into the Mainly two category:

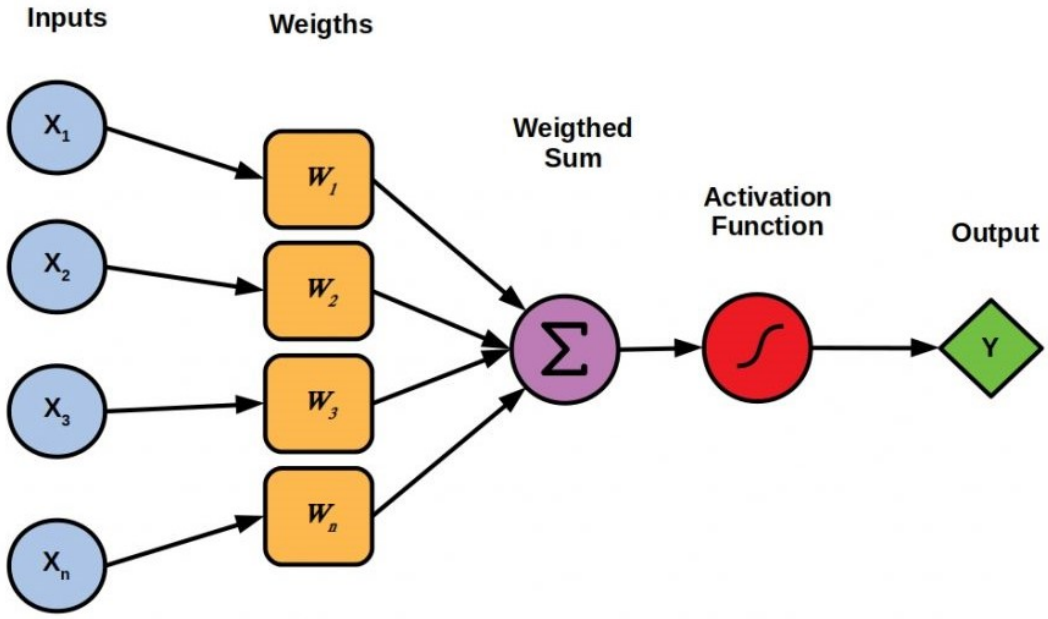
- Linear Models
  - Logistic Regression
  - Support Vector Machines
- Non-linear Models
  - K-Nearest Neighbours
  - Kernel SVM
  - Naïve Bayes
  - Decision Tree Classification
  - Random Forest Classification

PERCEPTRON

Perceptron is an algorithm used for supervised learning of binary classifiers. Binary classifiers decide whether an input, usually represented by a series of vectors, belongs to a specific class. a perceptron is a single-layer neural network. They consist of four main parts including input values, weights and bias, net sum, and an activation function.

The process begins by taking all the input values and multiplying them by their weights. Then, all of these multiplied values are added together to create the weighted sum. The weighted sum is then applied to the activation function, producing the perceptron's output. The activation function plays the integral role of ensuring the output is mapped between required values such as (0,1) or

(-1,1). It is important to note that the weight of an input is indicative of the strength of a node. Similarly, an input's bias value gives the ability to shift the activation function curve up or down.

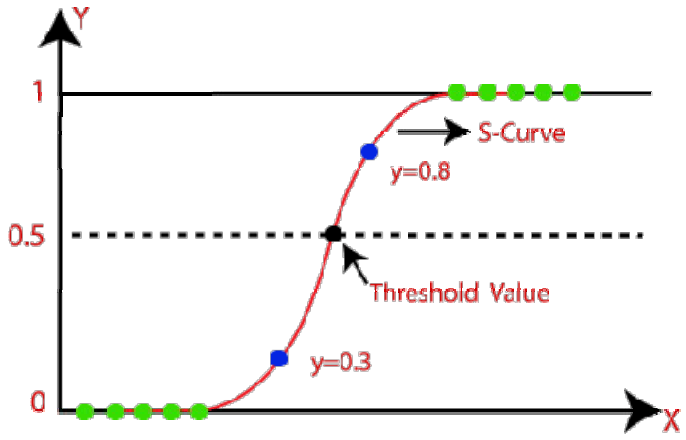


As a simplified form of a neural network, specifically a single-layer neural network, perceptrons play an important role in binary classification. This means the perceptron is used to classify data into two parts, hence binary. Sometimes, perceptrons are also referred to as linear binary classifiers for this reason.

LOGISTIC REGRESSION

- Logistic regression is one of the most popular Machine Learning algorithms, which comes under the Supervised Learning technique. It is used for predicting the categorical dependent variable using a given set of independent variables.
- Logistic regression predicts the output of a categorical dependent variable. Therefore the outcome must be a categorical or discrete value. It can be either Yes or No, 0 or 1, true or False, etc. but instead of giving the exact value as 0 and 1, it gives the probabilistic values which lie between 0 and 1.
- Logistic Regression is much similar to the Linear Regression except that how they are used. Linear Regression is used for solving Regression problems, whereas Logistic regression is used for solving the classification problems.

- In Logistic regression, instead of fitting a regression line, we fit an "S" shaped logistic function, which predicts two maximum values (0 or 1).
- The curve from the logistic function indicates the likelihood of something such as whether the cells are cancerous or not, a mouse is obese or not based on its weight, etc.
- Logistic Regression is a significant machine learning algorithm because it has the ability to provide probabilities and classify new data using continuous and discrete datasets.
- Logistic Regression can be used to classify the observations using different types of data and can easily determine the most effective variables used for the classification. The below image is showing the logistic function:



Logistic Regression Equation:

The Logistic regression equation can be obtained from the Linear Regression equation. The mathematical steps to get Logistic Regression equations are given below:

- We know the equation of the straight line can be written as:

$$y= b_0 + b_1x_1 + b_2x_2 + b_3x_3 + \cdots + b_nx_n$$

- In Logistic Regression y can be between 0 and 1 only, so for this let's divide the above equation by (1-y):

$$\frac{y}{1-y}$$; 0 for y= 0, and infinity for y=1

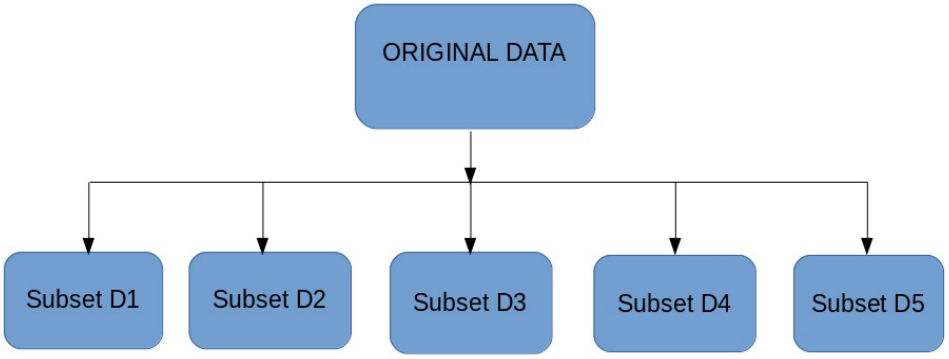
Type of Logistic Regression:

On the basis of the categories, Logistic Regression can be classified into three types:

- Binomial: In binomial Logistic regression, there can be only two possible types of the dependent variables, such as 0 or 1, Pass or Fail, etc.
- Multinomial: In multinomial Logistic regression, there can be 3 or more possible unordered types of the dependent variable, such as "cat", "dogs", or "sheep"
- Ordinal: In ordinal Logistic regression, there can be 3 or more possible ordered types of dependent variables, such as "low", "Medium", or "High".

BOOSTING AND BAGGING

- Bagging (or Bootstrap Aggregation), is a simple and very powerful ensemble method. Bagging is the application of the Bootstrap procedure to a high-variance machine learning algorithm, typically decision trees.
- The idea behind bagging is combining the results of multiple models (for instance, all decision trees) to get a generalized result. Now, bootstrapping comes into picture.
- Bagging (or Bootstrap Aggregating) technique uses these subsets (bags) to get a fair idea of the distribution (complete set). The size of subsets created for bagging may be less than the original set.
- It can be represented as follows:

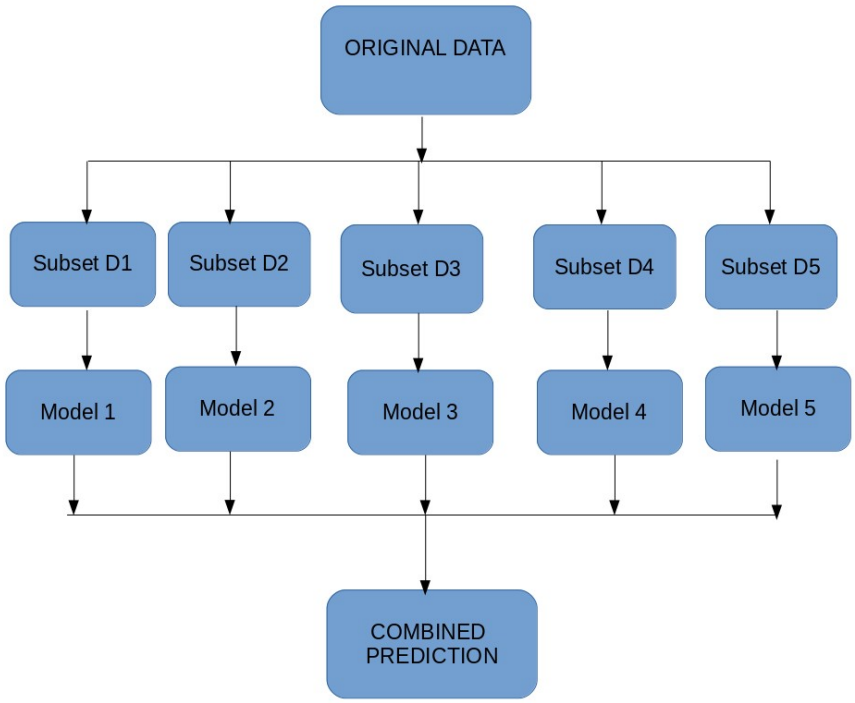


Bagging works as follows:-

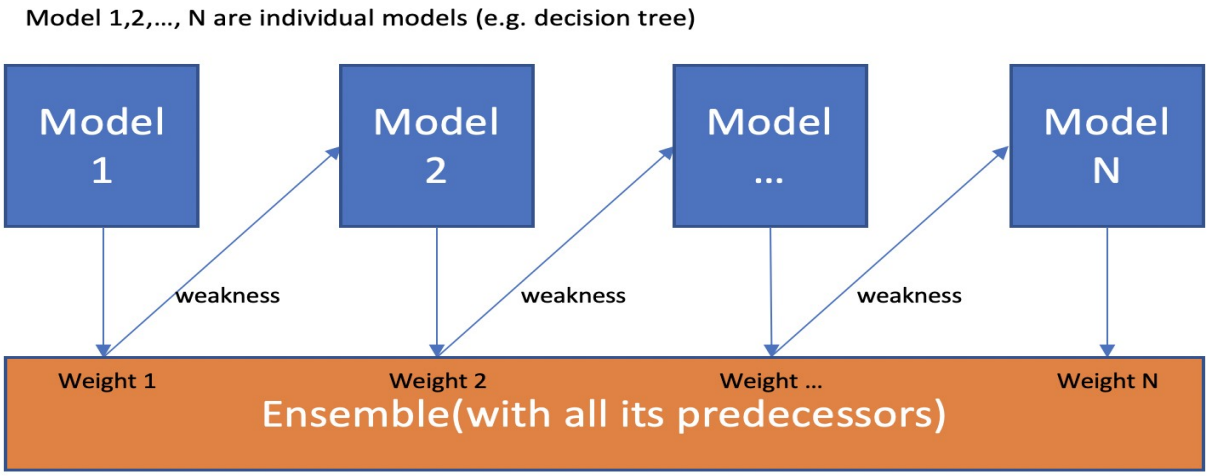
1. Multiple subsets are created from the original dataset, selecting observations with replacement.
2. A base model (weak model) is created on each of these subsets.

3. The models run in parallel and are independent of each other.
4. The final predictions are determined by combining the predictions from all the models.

Now, bagging can be represented diagrammatically as follows



- Boosting is a sequential process, where each subsequent model attempts to correct the errors of the previous model. The succeeding models are dependent on the previous model.
- In this technique, learners are learned sequentially with early learners fitting simple models to the data and then analyzing data for errors. In other words, we fit consecutive trees (random sample) and at every step, the goal is to solve for net error from the prior tree.
- When an input is misclassified by a hypothesis, its weight is increased so that next hypothesis is more likely to classify it correctly. By combining the whole set at the end converts weak learners into better performing model.
- Let's understand the way boosting works in the below steps.
    1. A subset is created from the original dataset.
    2. Initially, all data points are given equal weights.
    3. A base model is created on this subset.
    4. This model is used to make predictions on the whole dataset.



- Errors are calculated using the actual values and predicted values.
- The observations which are incorrectly predicted, are given higher weights. (Here, the three misclassified blue-plus points will be given higher weights)
- Another model is created and predictions are made on the dataset. (This model tries to correct the errors from the previous model)
- Similarly, multiple models are created, each correcting the errors of the previous model.
- The final model (strong learner) is the weighted mean of all the models (weak learners).

CLUSTERING---PARTITIONAL AND HIERARCHICAL; K-MEANS CLUSTERING

Cluster analysis, or clustering, is an unsupervised machine learning task.

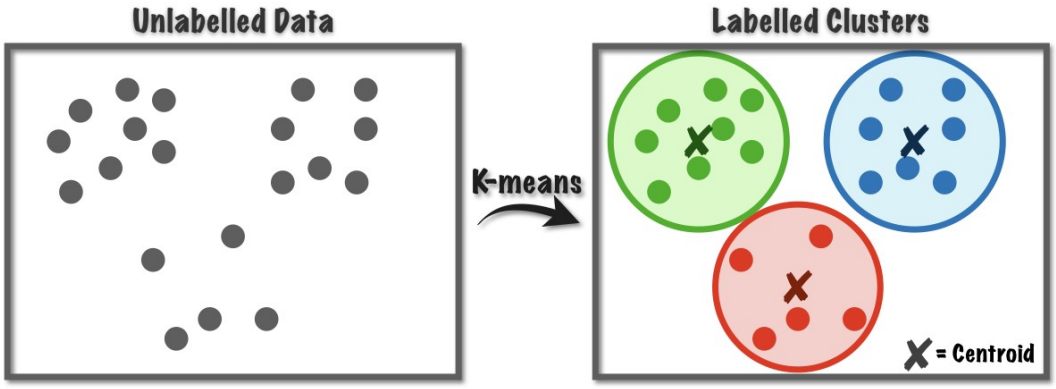
It involves automatically discovering natural grouping in data. Unlike supervised learning, clustering algorithms only interpret the input data and find natural groups or clusters.

1. Examples of Clustering Algorithms
1. BIRCH
2. DBSCAN
3. K-Means
4. Spectral Clustering
5. Gaussian Mixture Model

K-MEANS

K-means clustering algorithm computes the centroids and iterates until we it finds optimal centroid. It assumes that the number of clusters are already known. It is also called flat clustering algorithm. The number of clusters identified from data by algorithm is represented by 'K' in K-means.

In this algorithm, the data points are assigned to a cluster in such a manner that the sum of the squared distance between the data points and centroid would be minimum. It is to be understood that less variation within the clusters will lead to more similar data points within same cluster.



Working of K-Means Algorithm

Step 1 − First, we need to specify the number of clusters, K, need to be generated by this algorithm.

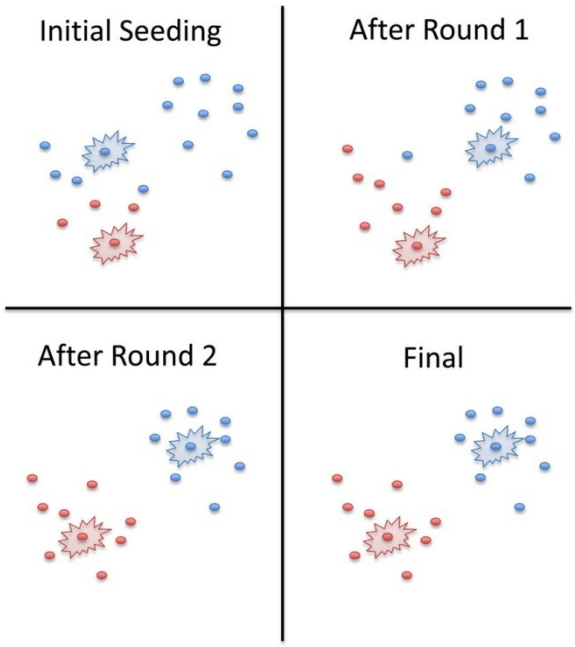
Step 2 − Next, randomly select K data points and assign each data point to a cluster. In simple words, classify the data based on the number of data points.

Step 3 − Now it will compute the cluster centroids.

Step 4 − Next, keep iterating the following until we find optimal centroid which is the assignment of data points to the clusters that are not changing any more

- 4.1 − First, the sum of squared distance between data points and centroids would be computed.
- 4.2 − Assign each data point to the cluster that is closer than other cluster (centroid).
- 4.3 − At last compute the centroids for the clusters by taking the average of all data points of that cluster.

K-means follows Expectation-Maximization approach to solve the problem. The Expectation-step is used for assigning the data points to the closest cluster and the Maximization-step is used for computing the centroid of each cluster.



Applications of K-Means Clustering Algorithm

- Market segmentation
- Document Clustering
- Image segmentation
- Image compression
- Customer segmentation
- Analyzing the trend on dynamic data

EVALUATION METRICS :

Root mean square error or root mean square deviation is one of the most commonly used measures for evaluating the quality of predictions. It shows how far predictions fall from measured true values using Euclidean distance.

To compute RMSE, calculate the residual (difference between prediction and truth) for each data point, compute the norm of residual for each data point, compute the mean of residuals and take the square root of that mean. RMSE is commonly used in supervised learning applications, as RMSE uses and needs true measurements at each predicted data point.

Root mean square error can be expressed as

$$RMSE = \sqrt{\frac{\sum_{i=1}^{N} \|y(i) - \hat{y}(i)\|^2}{N}},$$

where N is the number of data points, y(i) is the i-th measurement, and ŷ(i) is its corresponding prediction.

Mean Absolute Error (MAE)

Mean Absolute Error (also called L1 loss) is one of the most simple yet robust loss functions used for regression models.

MAE takes the average sum of the absolute differences between the actual and the predicted values. For a data point x_i and its predicted value y_i, n being the total number of data points in the dataset, the mean absolute error is defined as:

$$\text{MAE} = \frac{\sum_{i=1}^{n} |y_i - x_i|}{n}$$

Coefficient of Determination (R Squared)

- The coefficient of determination is the square of the correlation(r), thus it ranges from 0 to 1.
- With linear regression, the coefficient of determination is equal to the square of the correlation between the x and y variables.
- If R^2 is equal to 0, then the dependent variable cannot be predicted from the independent variable.
- If R^2 is equal to 1, then the dependent variable can be predicted from the independent variable without any error.
- If R^2 is between 0 and 1, then it indicates the extent that the dependent variable can be predictable. If R^2 of 0.10 means, it is 10 percent of the variance in the y variable is predicted from the x variable. If 0.20 means, 20 percent of the variance in the y variable is predicted from the x variable, and so on.

The value of R^2 shows whether the model would be a good fit for the given data set.

$$r = \frac{n(\Sigma xy) - (\Sigma x)(\Sigma y)}{\sqrt{[\, n\Sigma x^2 - (\Sigma x)^2 \,][\, n\Sigma y^2 - (\Sigma y)^2 \,]}}$$

TRAINING AND TESTING A CLASSIFIER

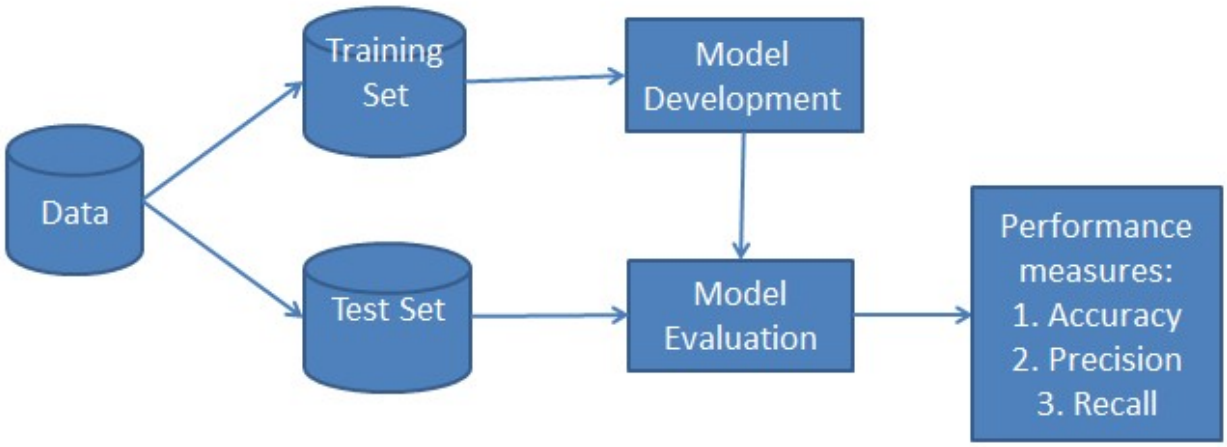
Training and Testing is a phenomena through which a system gets trained and becomes adaptable to give result in an accurate manner. Learning is the most important phase as how well the system performs on the data provided to the system depends on which algorithms used on the data. Entire dataset is divided into two categories, one which is used in training the model i.e. Training set and the other that is used in testing the model after training, i.e. Testing set.

- Trainingset:

  Training set is used to build a model. It consists of the set of images that are used to train the system. Training rules and algorithms used give relevant information on how to associate input data with output decision. The system is trained by applying these algorithms on the dataset, all the relevant information is extracted from the data and results are obtained. Generally, 80% of the data of the dataset is taken for training data

- Testingset:

  Testing data is used to test the system. It is the set of data which is used to verify whether the system is producing the correct output after being trained or not. Generally, 20% of the data of the dataset is used for testing.



CROSS-VALIDATION

Cross-Validation

Cross-validation is a technique in which we train our model using the subset of the data-set and then evaluate using the complementary subset of the data-set.

The three steps involved in cross-validation are as follows :

1. Reserve some portion of sample data-set.

2. Using the rest data-set train the model.
3. Test the model using the reserve portion of the data-set.

Methods of Cross Validation

- Validation
- LOOCV (Leave One Out Cross Validation)
- K-Fold Cross Validation

HANDLING- EXPLORATORY DATA ANALYSIS (EDA)

Steps in Data Exploration and Preprocessing:

1. Identification of variables and data types
2. Analyzing the basic metrics
3. Non-Graphical Univariate Analysis
4. Graphical Univariate Analysis
5. Bivariate Analysis
6. Variable transformations
7. Missing value treatment
8. Outlier treatment
9. Correlation Analysis
10. Dimensionality Reduction

ROC CURVE

An ROC curve (receiver operating characteristic curve) is a graph showing the performance of a classification model at all classification thresholds. This curve plots two parameters:

- True Positive Rate
- False Positive Rate

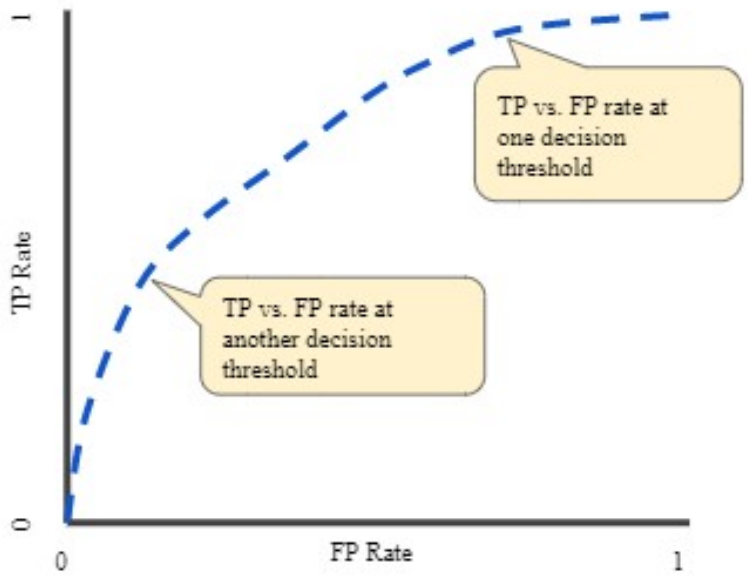
True Positive Rate (TPR) is a synonym for recall and is therefore defined as follows:

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

False Positive Rate (FPR) is defined as follows:

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

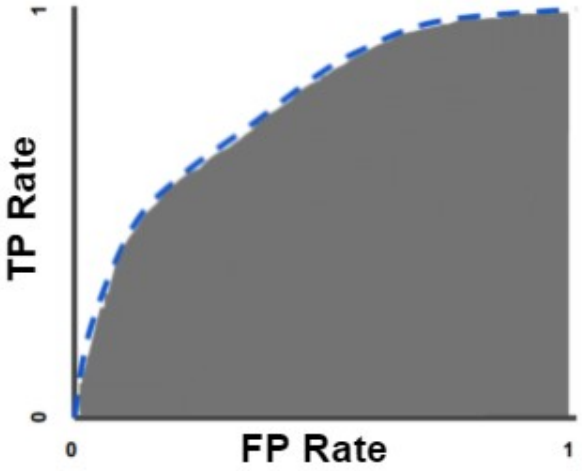
An ROC curve plots TPR vs. FPR at different classification thresholds. Lowering the classification threshold classifies more items as positive, thus increasing both False Positives and True Positives. The following figure shows a typical ROC curve.



To compute the points in an ROC curve, we could evaluate a logistic regression model many times with different classification thresholds, but this would be inefficient. Fortunately, there's an efficient, sorting-based algorithm that can provide this information for us, called AUC.

AUC: Area Under the ROC Curve

AUC stands for "Area under the ROC Curve." That is, AUC measures the entire two-dimensional area underneath the entire ROC curve (think integral calculus) from (0,0) to (1,1).



(COST FUNCTIONS : same as evaluation functions)