**Unit 1**

**Algorithm**

An algorithm is **a procedure used for solving a problem or performing a computation**. Algorithms act as an exact list of instructions that conduct specified actions step by step in either hardware- or software-based routines

**Machine learning algorithm**

1. **Linear Regression Algorithm**
2. **Logistic Regression Algorithm**
3. **Decision Tree**
4. **SVM**
5. **Naïve Bayes**
6. **KNN**
7. **K-Means Clustering**
8. **Random Forest**
9. **Apriori**
10. **PCA**

**Linear Regression Algorithm**

**Linear Regression** is a machine learning algorithm based on **supervised learning**. It performs a **regression task**. Regression models a target prediction value based on independent variables. It is mostly used for finding out the relationship between variables and forecasting. Different regression models differ based on – the kind of relationship between dependent and independent variables they are considering, and the number of independent variables getting used. There are many names for a regression’s dependent variable.  It may be called an outcome variable, criterion variable, endogenous variable, or regressand.  The independent variables can be called exogenous variables, predictor variables, or regressors.

Linear regression is used in many different fields, including finance, economics, and psychology, to understand and predict the behavior of a particular variable. For example, in finance, linear regression might be used to understand the relationship between a company’s stock price and its earnings, or to predict the future value of a currency based on its past performance.

One of the most important supervised learning tanks is regression. In regression set of records are present with X and Y values and this values are used to learn a function, so that if you want to predict Y from an unknown X this learn function can be used. In regression we have to find value of Y, So, a function is required which predicts Y given XY is continuous in case of regression.

Here Y is called as criterion variable and X is called as predictor variable. There are many types of functions or modules which can be used for regression. Linear function is the simplest type of function. Here, X may be a single feature or multiple features representing the problem.



Linear regression performs the task to predict a dependent variable value (y) based on a given independent variable (x)). Hence, the name is Linear Regression. In the figure above, X (input) is the work experience and Y (output) is the salary of a person. The regression line is the best fit line for our model.

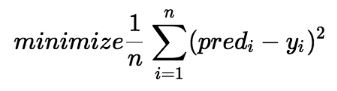
**Hypothesis function for Linear Regression :** 

While training the model we are given : **x:** input training data (univariate – one input variable(parameter)) **y:** labels to data (Supervised learning) When training the model – it fits the best line to predict the value of y for a given value of x. The model gets the best regression fit line by finding the best θ1 and θ2 values. **θ1:** intercept **θ2:** coefficient of x Once we find the best θ1 and θ2 values, we get the best fit line. So when we are finally using our model for prediction, it will predict the value of y for the input value of x.

**How to update θ1 and θ2 values to get the best fit line?**

Linear regression is a powerful tool for understanding and predicting the behavior of a variable, but it has some limitations. One limitation is that it assumes a linear relationship between the independent variables and the dependent variable, which may not always be the case. In addition, linear regression is sensitive to outliers, or data points that are significantly different from the rest of the data. These outliers can have a disproportionate effect on the fitted line, leading to inaccurate predictions.

**Cost Function (J):** By achieving the best-fit regression line, the model aims to predict y value such that the error difference between predicted value and true value is minimum. So, it is very important to update the θ1 and θ2 values, to reach the best value that minimize the error between predicted y value (pred) and true y value (y).



Cost function(J) of Linear Regression is the **Root Mean Squared Error (RMSE)** between predicted y value (pred) and true y value (y). [**Gradient Descent**](https://www.geeksforgeeks.org/gradient-descent-in-linear-regression/)**:** To update θ1 and θ2 values in order to reduce Cost function (minimizing RMSE value) and achieving the best-fit line the model uses Gradient Descent. The idea is to start with random θ1 and θ2 values and then iteratively updating the values, reaching minimum cost.

**Decision Tree**

**Decision Tree** is the most powerful and popular tool for classification and prediction. A Decision tree is a flowchart-like tree structure, where each internal node denotes a test on an attribute, each branch represents an outcome of the test, and each leaf node (terminal node) holds a class label.



*A decision tree for the concept PlayTennis.*

**Construction of Decision Tree:** A tree can be *“learned”* by splitting the source set into subsets based on an attribute value test. This process is repeated on each derived subset in a recursive manner called*recursive partitioning*. The recursion is completed when the subset at a node all has the same value of the target variable, or when splitting no longer adds value to the predictions. The construction of a decision tree classifier does not require any domain knowledge or parameter setting, and therefore is appropriate for exploratory knowledge discovery. Decision trees can handle high-dimensional data. In general decision tree, classifier has good accuracy. Decision tree induction is a typical inductive approach to learn knowledge on classification.

**Short note on Decision Tree:-**

* A decision tree which is also known as prediction tree refers a tree structure to mention the sequences of decisions as well as consequences.
* Considering the input X = (X1, X2,… Xn), the aim is to predict a response or output variable Y.
* Each element in the set (X1, X2,…Xn) is known as input variable.It is possible to achieve the prediction by the process of building a decision tree which has test points as well as branches.
* At each test point, it is decided to select a particular branch and traverse down the tree.
* Ultimately, a final point is reached, and it will be easy to make prediction.
* In a decision tree, all the test points exhibit testing specific input variables (or attributes), and the developed decision tree is represented by the branches.
* Because of flexibility as well as simple visualization, decision trees are mostly probably deployed in data mining applications for the purpose of classification.
* In the decision tree, the input values are considered as categorical or continuous.
* A structure of test points (known as nodes) and branches is established by the decision tree by which the decision being made will be represented.
* Leaf node is the one which do not have further branches. The returning value of leaf nodes is class labels while in some cases they return the probability scores.
* It is possible to convert decision tree into a set of decision rules.
* There are two types of Decision trees: **classification trees and regression trees**
* **Classification trees**are generally applied to output variables which are categorical and mostly binary in nature, for example yes or no, sale or not, and so on.
* Whereas **regression trees** are applied to output variables which are numeric or continuous, for example predicted price of a consumer good.
* In variety of situations, it is possible to apply decision tree. It is easy to represent them in a visual way, and the analogous straightforward.
* Also as the result is a sequence of logical if-then statements, there is no any presence of underlying assumption regarding a linear or nonlinear relationship between the input variables and the response variable.

**Decision Tree Representation:** Decision trees classify instances by sorting them down the tree from the root to some leaf node, which provides the classification of the instance. An instance is classified by starting at the root node of the tree, testing the attribute specified by this node, then moving down the tree branch corresponding to the value of the attribute as shown in the above figure. This process is then repeated for the subtree rooted at the new node.   
The decision tree in above figure classifies a particular morning according to whether it is suitable for playing tennis and returns the classification associated with the particular leaf.(in this case Yes or No).

For example, the instance 

*(Outlook = Sunny, Temperature = Hot, Humidity = High, Wind = Strong )*

would be sorted down the leftmost branch of this decision tree and would therefore be classified as a negative instance.   
In other words, we can say that the decision tree represents a disjunction of conjunctions of constraints on the attribute values of instances.

*(Outlook = Sunny ^ Humidity = Normal) v (Outlook = Overcast) v (Outlook = Rain ^ Wind = Weak)*

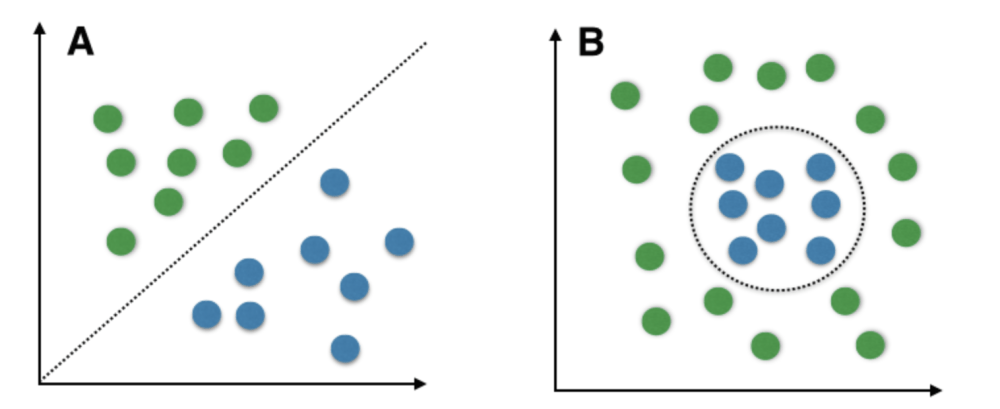
**Support Vector Machines**

Support Vector Machines (SVMs) are a type of supervised learning algorithm that can be used for classification or regression tasks. The main idea behind SVMs is to find a hyperplane that maximally separates the different classes in the training data. This is done by finding the hyperplane that has the largest margin, which is defined as the distance between the hyperplane and the closest data points from each class. Once the hyperplane is determined, new data can be classified by determining on which side of the hyperplane it falls. SVMs are particularly useful when the data has many features, and/or when there is a clear margin of separation in the data.

**What are Support Vector Machines?** Support Vector Machine (SVM) is a relatively simple **Supervised Machine Learning Algorithm** used for classification and/or regression. It is more preferred for classification but is sometimes very useful for regression as well. Basically, SVM finds a hyper-plane that creates a boundary between the types of data. In 2-dimensional space, this hyper-plane is nothing but a line. In SVM, we plot each data item in the dataset in an N-dimensional space, where N is the number of features/attributes in the data. Next, find the optimal hyperplane to separate the data. So by this, you must have understood that inherently, SVM can only perform binary classification (i.e., choose between two classes). However, there are various techniques to use for multi-class problems. **Support Vector Machine for Multi-CLass Problems** To perform SVM on multi-class problems, we can create a binary classifier for each class of the data. The two results of each classifier will be :

* The data point belongs to that class OR
* The data point does not belong to that class.

For example, in a class of fruits, to perform multi-class classification, we can create a binary classifier for each fruit. For say, the ‘mango’ class, there will be a binary classifier to predict if it IS a mango OR it is NOT a mango. The classifier with the highest score is chosen as the output of the SVM. **SVM for complex (Non Linearly Separable)** SVM works very well without any modifications for linearly separable data. **Linearly Separable Data**is any data that can be plotted in a graph and can be separated into classes using a straight line.



*A: Linearly Separable Data B: Non-Linearly Separable Data*

We use **Kernelized SVM** for non-linearly separable data. Say, we have some non-linearly separable data in one dimension. We can transform this data into two dimensions and the data will become linearly separable in two dimensions. This is done by mapping each 1-D data point to a corresponding 2-D ordered pair. So for any non-linearly separable data in any dimension, we can just map the data to a higher dimension and then make it linearly separable. This is a very powerful and general transformation. A **kernel** is nothing but a measure of similarity between data points. The **kernel function** in a kernelized SVM tells you, that given two data points in the original feature space, what the similarity is between the points in the newly transformed feature space. There are various kernel functions available, but two are very popular :

* **Radial Basis Function Kernel (RBF):** The similarity between two points in the transformed feature space is an exponentially decaying function of the distance between the vectors and the original input space as shown below. RBF is the default kernel used in SVM.
* **Polynomial Kernel:**The Polynomial kernel takes an additional parameter, ‘degree’ that controls the model’s complexity and computational cost of the transformation

A very interesting fact is that SVM does not actually have to perform this actual transformation on the data points to the new high dimensional feature space. This is called the **kernel trick**. **The Kernel Trick:**Internally, the kernelized SVM can compute these complex transformations just in terms of similarity calculations between pairs of points in the higher dimensional feature space where the transformed feature representation is implicit. This similarity function, which is mathematically a kind of complex dot product is actually the kernel of a kernelized SVM. This makes it practical to apply SVM when the underlying feature space is complex or even infinite-dimensional. The kernel trick itself is quite complex and is beyond the scope of this article. **Important Parameters in Kernelized SVC ( Support Vector Classifier)**

1. **The Kernel**: The kernel, is selected based on the type of data and also the type of transformation. By default, the kernel is Radial Basis Function Kernel (RBF).
2. **Gamma** : This parameter decides how far the influence of a single training example reaches during transformation, which in turn affects how tightly the decision boundaries end up surrounding points in the input space. If there is a small value of gamma, points farther apart are considered similar. So more points are grouped together and have smoother decision boundaries (maybe less accurate). Larger values of gamma cause points to be closer together (may cause overfitting).
3. **The ‘C’ parameter**: This parameter controls the amount of regularization applied to the data. Large values of C mean low regularization which in turn causes the training data to fit very well (may cause overfitting). Lower values of C mean higher regularization which causes the model to be more tolerant of errors (may lead to lower accuracy).

**Pros of Kernelized SVM:**

1. They perform very well on a range of datasets.
2. They are versatile: different kernel functions can be specified, or custom kernels can also be defined for specific datatypes.
3. They work well for both high and low dimensional data.

**Cons of Kernelized SVM:**

1. Efficiency (running time and memory usage) decreases as the size of the training set increases.
2. Needs careful normalization of input data and parameter tuning.
3. Does not provide a direct probability estimator.
4. Difficult to interpret why a prediction was made.

**Naive Bayes**

**Naive Bayes** classifiers are a collection of classification algorithms based on **Bayes’ Theorem**. It is not a single algorithm but a family of algorithms where all of them share a common principle, i.e. every pair of features being classified is independent of each other.

To start with, let us consider a dataset.

Consider a fictional dataset that describes the weather conditions for playing a game of golf. Given the weather conditions, each tuple classifies the conditions as fit(“Yes”) or unfit(“No”) for playing golf.

Here is a tabular representation of our dataset.

|  | **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 |

The dataset is divided into two parts, namely, **feature matrix** and the **response vector**.

* Feature matrix contains all the vectors(rows) of dataset in which each vector consists of the value of **dependent features**. In above dataset, features are ‘Outlook’, ‘Temperature’, ‘Humidity’ and ‘Windy’.
* Response vector contains the value of **class variable**(prediction or output) for each row of feature matrix. In above dataset, the class variable name is ‘Play golf’.

**Assumption:**

The fundamental Naive Bayes assumption is that each feature makes an:

* independent
* equal

contribution to the outcome.

With relation to our dataset, this concept can be understood as:

* We assume that no pair of features are dependent. For example, the temperature being ‘Hot’ has nothing to do with the humidity or the outlook being ‘Rainy’ has no effect on the winds. Hence, the features are assumed to be **independent**.
* Secondly, each feature is given the same weight(or importance). For example, knowing only temperature and humidity alone can’t predict the outcome accurately. None of the attributes is irrelevant and assumed to be contributing **equally** to the outcome.

**Note:** The assumptions made by Naive Bayes are not generally correct in real-world situations. In-fact, the independence assumption is never correct but often works well in practice.

Now, before moving to the formula for Naive Bayes, it is important to know about Bayes’ theorem.

**Bayes’ Theorem**

Bayes’ Theorem finds the probability of an event occurring given the probability of another event that has already occurred. Bayes’ theorem is stated mathematically as the following equation:

where A and B are events and P(B) ≠ 0.

* Basically, we are trying to find probability of event A, given the event B is true. Event B is also termed as **evidence**.
* P(A) is the **priori** of A (the prior probability, i.e. Probability of event before evidence is seen). The evidence is an attribute value of an unknown instance(here, it is event B).
* P(A|B) is a posteriori probability of B, i.e. probability of event after evidence is seen.

Now, with regards to our dataset, we can apply Bayes’ theorem in following way:

where, y is class variable and X is a dependent feature vector (of size *n*) where:

Just to clear, an example of a feature vector and corresponding class variable can be: (refer 1st row of dataset)

X = (Rainy, Hot, High, False)

y = No

So basically, P(y|X) here means, the probability of “Not playing golf” given that the weather conditions are “Rainy outlook”, “Temperature is hot”, “high humidity” and “no wind”.

**Naive assumption**

Now, its time to put a naive assumption to the Bayes’ theorem, which is, **independence** among the features. So now, we split **evidence** into the independent parts.

Now, if any two events A and B are independent, then,

P(A,B) = P(A)P(B)

Hence, we reach to the result:

which can be expressed as:

Now, as the denominator remains constant for a given input, we can remove that term:

Now, we need to create a classifier model. For this, we find the probability of given set of inputs for all possible values of the class variable *y* and pick up the output with maximum probability. This can be expressed mathematically as:

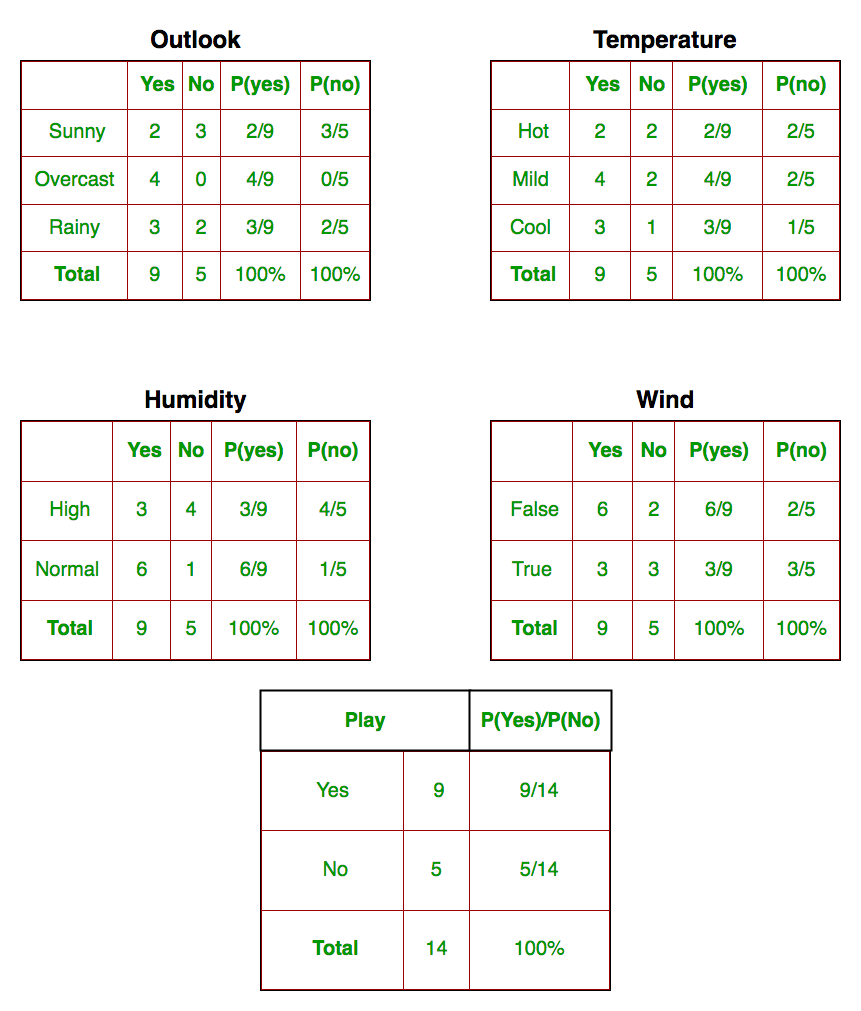
So, finally, we are left with the task of calculating P(y) and P(xi | y).

Please note that P(y) is also called **class probability** and P(xi | y) is called **conditional probability**.

The different naive Bayes classifiers differ mainly by the assumptions they make regarding the distribution of P(xi | y).

Let us try to apply the above formula manually on our weather dataset. For this, we need to do some precomputations on our dataset.

We need to find P(xi | yj) for each xi in X and yj in y. All these calculations have been demonstrated in the tables below:



So, in the figure above, we have calculated P(xi | yj) for each xi in X and yj in y manually in the tables 1-4. For example, probability of playing golf given that the temperature is cool, i.e P(temp. = cool | play golf = Yes) = 3/9.

Also, we need to find class probabilities (P(y)) which has been calculated in the table 5. For example, P(play golf = Yes) = 9/14.

So now, we are done with our pre-computations and the classifier is ready!

Let us test it on a new set of features (let us call it today):

today = (Sunny, Hot, Normal, False)

So, probability of playing golf is given by:

and probability to not play golf is given by:

Since, P(today) is common in both probabilities, we can ignore P(today) and find proportional probabilities as:

and

Now, since

These numbers can be converted into a probability by making the sum equal to 1 (normalization):

and

Since

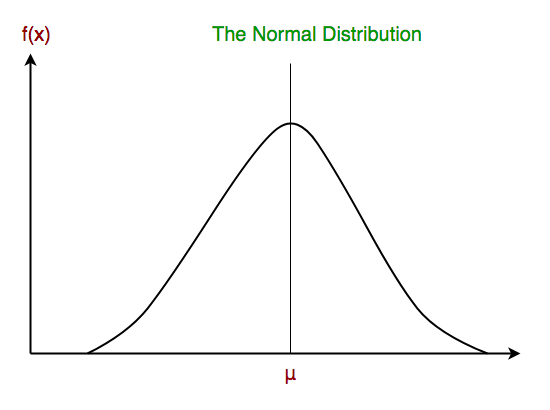
So, prediction that golf would be played is ‘Yes’.

The method that we discussed above is applicable for discrete data. In case of continuous data, we need to make some assumptions regarding the distribution of values of each feature. The different naive Bayes classifiers differ mainly by the assumptions they make regarding the distribution of P(xi | y).

Now, we discuss one of such classifiers here.

**Gaussian Naive Bayes classifier**

In Gaussian Naive Bayes, continuous values associated with each feature are assumed to be distributed according to a **Gaussian distribution**. A Gaussian distribution is also called [Normal distribution](https://en.wikipedia.org/wiki/Normal_distribution). When plotted, it gives a bell shaped curve which is symmetric about the mean of the feature values as shown below:



The likelihood of the features is assumed to be Gaussian, hence, conditional probability is given by:

Other popular Naive Bayes classifiers are:

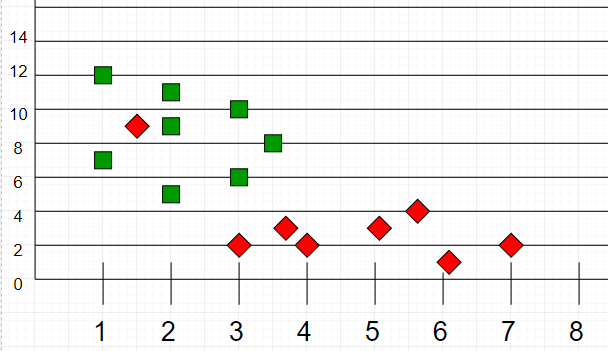
* **Multinomial Naive Bayes**: Feature vectors represent the frequencies with which certain events have been generated by a **multinomial distribution**. This is the event model typically used for document classification.
* **Bernoulli Naive Bayes**: In the multivariate Bernoulli event model, features are independent booleans (binary variables) describing inputs. Like the multinomial model, this model is popular for document classification tasks, where binary term occurrence(i.e. a word occurs in a document or not) features are used rather than term frequencies(i.e. frequency of a word in the document).

As we reach to the end of this article, here are some important points to ponder upon:

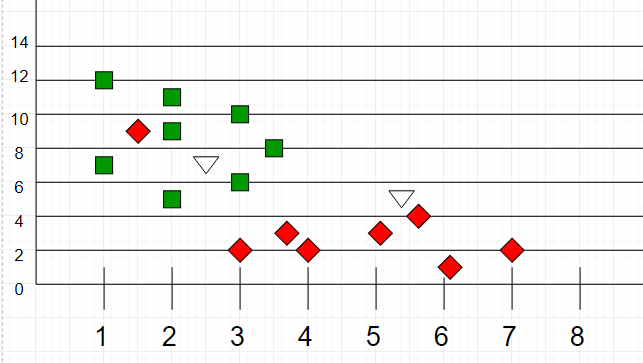
* In spite of their apparently over-simplified assumptions, naive Bayes classifiers have worked quite well in many real-world situations, famously document classification and spam filtering. They require a small amount of training data to estimate the necessary parameters.
* Naive Bayes learners and classifiers can be extremely fast compared to more sophisticated methods. The decoupling of the class conditional feature distributions means that each distribution can be independently estimated as a one dimensional distribution. This in turn helps to alleviate problems stemming from the curse of dimensionality.

**K-Nearest Neighbours**

K-Nearest Neighbours is one of the most basic yet essential classification algorithms in Machine Learning. It belongs to the supervised learning domain and finds intense application in pattern recognition, data mining and intrusion detection.  
It is widely disposable in real-life scenarios since it is non-parametric, meaning, it does not make any underlying assumptions about the distribution of data (as opposed to other algorithms such as [GMM](https://en.wikipedia.org/wiki/Mixture_model), which assume a Gaussian distribution of the given data).  
We are given some prior data (also called training data), which classifies coordinates into groups identified by an attribute.  
As an example, consider the following table of data points containing two features: 



Now, given another set of data points (also called testing data), allocate these points a group by analyzing the training set. Note that the unclassified points are marked as ‘White’.



**Intuition**   
If we plot these points on a graph, we may be able to locate some clusters or groups. Now, given an unclassified point, we can assign it to a group by observing what group its nearest neighbours belong to. This means a point close to a cluster of points classified as ‘Red’ has a higher probability of getting classified as ‘Red’.  
Intuitively, we can see that the first point (2.5, 7) should be classified as ‘Green’ and the second point (5.5, 4.5) should be classified as ‘Red’.  
**Algorithm**   
Let m be the number of training data samples. Let p be an unknown point. 

1. Store the training samples in an array of data points arr[]. This means each element of this array represents a tuple (x, y).

for i=0 to m:

Calculate Euclidean distance d(arr[i], p).

1. Make set S of K smallest distances obtained. Each of these distances corresponds to an already classified data point.
2. Return the majority label among S.

Recommended Problem

K-Nearest Neighbours

[Machine Learning](https://practice.geeksforgeeks.org/explore?page=1&category%5b%5d=Machine%20Learning&sortBy=submissions)

[Directi](https://practice.geeksforgeeks.org/explore?page=1&company%5b%5d=Directi&sortBy=submissions)

[Solve Problem](https://practice.geeksforgeeks.org/problems/k-nearest-neighbours/1?utm_source=gfg&utm_medium=article&utm_campaign=bottom_sticky_on_article" \o "Permalink to K-Nearest Neighbours)

Submission count: 764

## K can be kept as an odd number so that we can calculate a clear majority in the case where only two groups are possible (e.g. Red/Blue). With increasing K, we get smoother, more defined boundaries across different classifications. Also, the accuracy of the above classifier increases as we increase the number of data points in the training set. **K means Clustering**

[Unsupervised Machine Learning learning](https://www.geeksforgeeks.org/supervised-unsupervised-learning/) is the process of teaching a computer to use unlabeled, unclassified data and enabling the algorithm to operate on that data without supervision. Without any previous data training, the machine’s job in this case is to organize unsorted data according to parallels, patterns, and variations.

The goal of [clustering](https://www.geeksforgeeks.org/clustering-in-machine-learning/) is to divide the population or set of data points into a number of groups so that the data points within each group are more comparable to one another and different from the data points within the other groups. It is essentially a grouping of things based on how similar and different they are to one another.

We are given a data set of items, with certain features, and values for these features (like a vector). The task is to categorize those items into groups. To achieve this, we will use the K-means algorithm; an unsupervised learning algorithm. ‘K’ in the name of the algorithm represents the number of groups/clusters we want to classify our items into.

(It will help if you think of items as points in an n-dimensional space).  The algorithm will categorize the items into k groups or clusters of similarity. To calculate that similarity, we will use the euclidean distance as a measurement.

The algorithm works as follows:

1. First, we randomly initialize k points, called means or cluster centroids.
2. We categorize each item to its closest mean and we update the mean’s coordinates, which are the averages of the items categorized in that cluster so far.
3. We repeat the process for a given number of iterations and at the end, we have our clusters.

The “points” mentioned above are called means because they are the mean values of the items categorized in them. To initialize these means, we have a lot of options. An intuitive method is to initialize the means at random items in the data set. Another method is to initialize the means at random values between the boundaries of the data set (if for a feature *x,* the items have values in [0,3], we will initialize the means with values for *x* at [0,3]).

The above algorithm in pseudocode is as follows:

Initialize k means with random values

--> For a given number of iterations:

--> Iterate through items:

--> Find the mean closest to the item by calculating

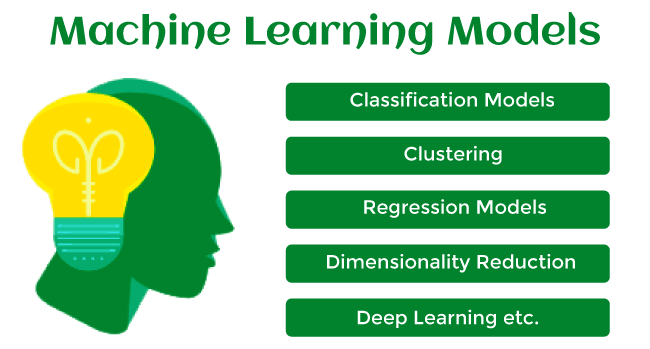
the euclidean distance of the item with each of the means

--> Assign item to mean

--> Update mean by shifting it to the average of the items in that cluster

# **Machine Learning Models**

**A machine learning model is defined as a mathematical representation of the output of the training process.** Machine learning is the study of different algorithms that can improve automatically through experience & old data and build the model. A machine learning model is similar to computer software designed to recognize patterns or behaviors based on previous experience or data. The learning algorithm discovers patterns within the training data, and it outputs an ML model which captures these patterns and makes predictions on new data.



Let's understand an example of the ML model where we are creating an app to recognize the user's emotions based on facial expressions. So, creating such an app is possible by Machine learning models where we will train a model by feeding images of faces with various emotions labeled on them. Whenever this app is used to determine the user's mood, it reads all fed data then determines any user's mood.

Hence, in simple words, we can say that a machine learning model is a simplified representation of something or a process. In this topic, we will discuss different machine learning models and their techniques and algorithms.

## **What is Machine Learning Model?**

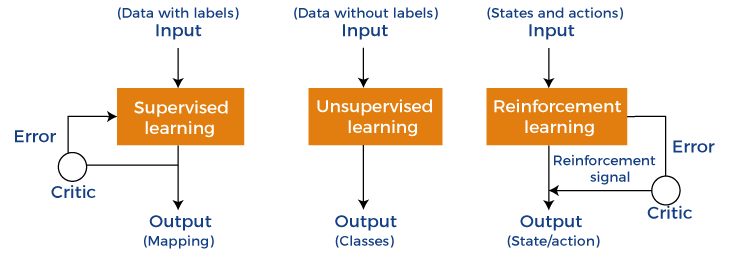
Machine Learning models can be understood as a program that has been trained to find patterns within new data and make predictions. These models are represented as a mathematical function that takes requests in the form of input data, makes predictions on input data, and then provides an output in response. First, these models are trained over a set of data, and then they are provided an algorithm to reason over data, extract the pattern from feed data and learn from those data. Once these models get trained, they can be used to predict the unseen dataset.

There are various types of machine learning models available based on different business goals and data sets.

### Classification of Machine Learning Models:

Based on different business goals and data sets, there are three learning models for algorithms. Each machine learning algorithm settles into one of the three models:

* Supervised Learning
* Unsupervised Learning
* Reinforcement Learning



**Supervised Learning is further divided into two categories:**

* Classification
* Regression

**Unsupervised Learning is also divided into below categories:**

* Clustering
* Association Rule
* Dimensionality Reduction

## **1. Supervised Machine Learning Models**

Supervised Learning is the simplest machine learning model to understand in which input data is called training data and has a known label or result as an output. So, it works on the principle of input-output pairs. It requires creating a function that can be trained using a training data set, and then it is applied to unknown data and makes some predictive performance. Supervised learning is task-based and tested on labeled data sets.

We can implement a supervised learning model on simple real-life problems. For example, we have a dataset consisting of age and height; then, we can build a supervised learning model to predict the person's height based on their age.

Supervised Learning models are further classified into two categories:

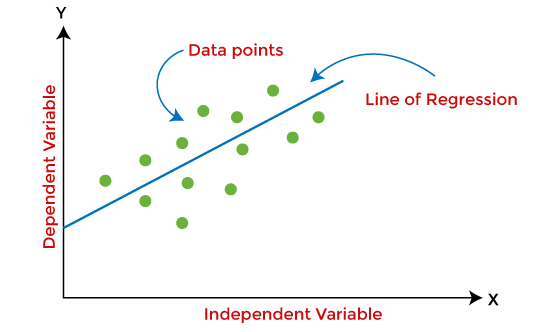
### Regression

In regression problems, the output is a continuous variable. Some commonly used Regression models are as follows:

**a) Linear Regression**

Linear regression is the simplest machine learning model in which we try to predict one output variable using one or more input variables. The representation of linear regression is a linear equation, which combines a set of input values(x) and predicted output(y) for the set of those input values. It is represented in the form of a line:

Y = bx+ c.



The main aim of the linear regression model is to find the best fit line that best fits the data points.

Linear regression is extended to multiple linear regression (find a plane of best fit) and polynomial regression (find the best fit curve).

**b) Decision Tree**

Decision trees are the popular machine learning models that can be used for both regression and classification problems.

A decision tree uses a tree-like structure of decisions along with their possible consequences and outcomes. In this, each internal node is used to represent a test on an attribute; each branch is used to represent the outcome of the test. The more nodes a decision tree has, the more accurate the result will be.

The advantage of decision trees is that they are intuitive and easy to implement, but they lack accuracy.

Decision trees are widely used in **operations research, specifically in decision analysis, strategic planning**, and mainly in machine learning.

**c) Random Forest**

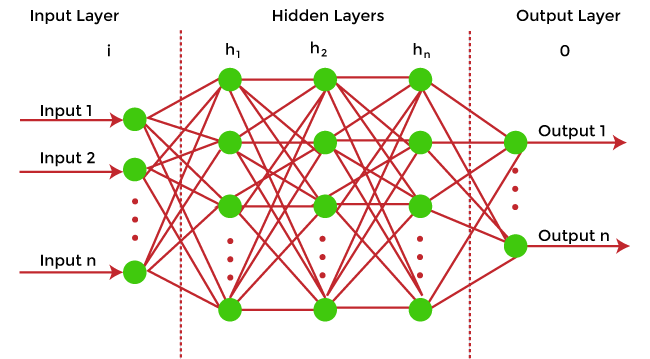
Random Forest is the ensemble learning method, which consists of a large number of decision trees. Each decision tree in a random forest predicts an outcome, and the prediction with the majority of votes is considered as the outcome.

A random forest model can be used for both regression and classification problems.

For the classification task, the outcome of the random forest is taken from the majority of votes. Whereas in the regression task, the outcome is taken from the mean or average of the predictions generated by each tree.

**d) Neural Networks**

Neural networks are the subset of machine learning and are also known as artificial neural networks. Neural networks are made up of artificial neurons and designed in a way that resembles the human brain structure and working. Each artificial neuron connects with many other neurons in a neural network, and such millions of connected neurons create a sophisticated cognitive structure.



Neural networks consist of a multilayer structure, containing one input layer, one or more hidden layers, and one output layer. As each neuron is connected with another neuron, it transfers data from one layer to the other neuron of the next layers. Finally, data reaches the last layer or output layer of the neural network and generates output.

Neural networks depend on training data to learn and improve their accuracy. However, a perfectly trained & accurate neural network can cluster data quickly and become a powerful machine learning and AI tool. One of the best-known neural networks is **Google's search algorithm.**

### Classification

Classification models are the second type of Supervised Learning techniques, which are used to generate conclusions from observed values in the categorical form. For example, the classification model can identify if the email is spam or not; a buyer will purchase the product or not, etc. Classification algorithms are used to predict two classes and categorize the output into different groups.

In classification, a classifier model is designed that classifies the dataset into different categories, and each category is assigned a label.

There are two types of classifications in machine learning:

* **Binary classification**: If the problem has only two possible classes, called a binary classifier. For example, cat or dog, Yes or No,
* **Multi-class classification**: If the problem has more than two possible classes, it is a multi-class classifier.

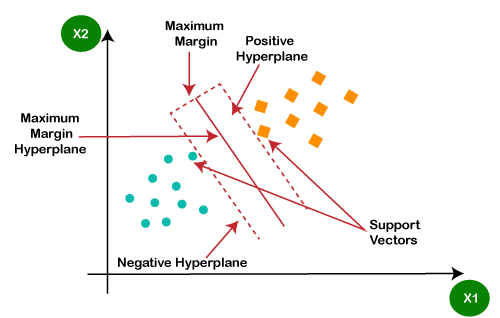
Some popular classification algorithms are as below:

**a) Logistic Regression**

Logistic Regression is used to solve the classification problems in machine learning. They are similar to linear regression but used to predict the categorical variables. It can predict the output in either Yes or No, 0 or 1, True or False, etc. However, rather than giving the exact values, it provides the probabilistic values between 0 & 1.

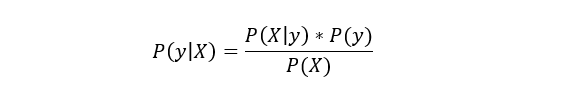
**b) Support Vector Machine**

Support vector machine or SVM is the popular machine learning algorithm, which is widely used for classification and regression tasks. However, specifically, it is used to solve classification problems. The main aim of SVM is to find the best decision boundaries in an N-dimensional space, which can segregate data points into classes, and the best decision boundary is known as Hyperplane. SVM selects the extreme vector to find the hyperplane, and these vectors are known as support vectors.



**c) Naïve Bayes**

Naïve Bayes is another popular classification algorithm used in machine learning. It is called so as it is based on Bayes theorem and follows the naïve(independent) assumption between the features which is given as:

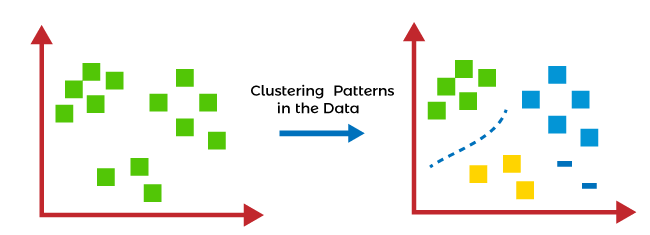


Each naïve Bayes classifier assumes that the value of a specific variable is independent of any other variable/feature. For example, if a fruit needs to be classified based on color, shape, and taste. So yellow, oval, and sweet will be recognized as mango. Here each feature is independent of other features.

## **2. Unsupervised Machine learning models**

Unsupervised Machine learning models implement the learning process opposite to supervised learning, which means it enables the model to learn from the unlabeled training dataset. Based on the unlabeled dataset, the model predicts the output. Using unsupervised learning, the model learns hidden patterns from the dataset by itself without any supervision.

Unsupervised learning models are mainly used to perform three tasks, which are as follows:

* **Clustering**  
  Clustering is an unsupervised learning technique that involves clustering or groping the data points into different clusters based on similarities and differences. The objects with the most similarities remain in the same group, and they have no or very few similarities from other groups.  
  Clustering algorithms can be widely used in different tasks such as **Image segmentation, Statistical data analysis, Market segmentation**, etc.  
  Some commonly used Clustering algorithms are K-means Clustering, hierarchal Clustering, DBSCAN, etc.  
  
* **Association Rule Learning**  
  Association rule learning is an unsupervised learning technique, which finds interesting relations among variables within a large dataset. The main aim of this learning algorithm is to find the dependency of one data item on another data item and map those variables accordingly so that it can generate maximum profit. This algorithm is mainly applied in **Market Basket analysis, Web usage mining, continuous production**, etc.  
  Some popular algorithms of Association rule learning are **Apriori Algorithm, Eclat, FP-growth algorithm.**
* **Dimensionality Reduction**  
  The number of features/variables present in a dataset is known as the dimensionality of the dataset, and the technique used to reduce the dimensionality is known as the dimensionality reduction technique.  
  Although more data provides more accurate results, it can also affect the performance of the model/algorithm, such as overfitting issues. In such cases, dimensionality reduction techniques are used.  
  "**It is a process of converting the higher dimensions dataset into lesser dimensions dataset ensuring that it provides similar information**."  
  Different dimensionality reduction methods such **as PCA(Principal Component Analysis), Singular Value Decomposition, etc.**

## **Reinforcement Learning**

In reinforcement learning, the algorithm learns actions for a given set of states that lead to a goal state. It is a feedback-based learning model that takes feedback signals after each state or action by interacting with the environment. This feedback works as a reward (positive for each good action and negative for each bad action), and the agent's goal is to maximize the positive rewards to improve their performance.

The behavior of the model in reinforcement learning is similar to human learning, as humans learn things by experiences as feedback and interact with the environment.

Below are some popular algorithms that come under reinforcement learning:

* **Q-learning:** Q-learning is one of the popular model-free algorithms of reinforcement learning, which is based on the Bellman equation.

It aims to learn the policy that can help the AI agent to take the best action for maximizing the reward under a specific circumstance. It incorporates Q values for each state-action pair that indicate the reward to following a given state path, and it tries to maximize the Q-value.

* **State-Action-Reward-State-Action (SARSA):** SARSA is an On-policy algorithm based on the Markov decision process. It uses the action performed by the current policy to learn the Q-value. The SARSA algorithm stands **for State Action Reward State Action, which symbolizes the tuple (s, a, r, s', a').**
* **Deep Q Network:** DQN **or Deep Q Neural network is Q-learning** within the neural network. It is basically employed in a big state space environment where defining a Q-table would be a complex task. So, in such a case, rather than using Q-table, the neural network uses Q-values for each action based on the state.

## **Training Machine Learning Models**

Once the Machine learning model is built, it is trained in order to get the appropriate results. To train a machine learning model, one needs a huge amount of pre-processed data. Here pre-processed data means data in structured form with reduced null values, etc. If we do not provide pre-processed data, then there are huge chances that our model may perform terribly.

## **How to choose the best model?**

In the above section, we have discussed different machine learning models and algorithms. But one most confusing question that may arise to any beginner that "which model should I choose?". So, the answer is that it depends mainly on the business requirement or project requirement. Apart from this, it also depends on associated attributes, the volume of the available dataset, the number of features, complexity, etc. However, in practice, it is recommended that we always start with the simplest model that can be applied to the particular problem and then gradually enhance the complexity & test the accuracy with the help of parameter tuning and cross-validation.

## **Difference between Machine learning model and Algorithms**

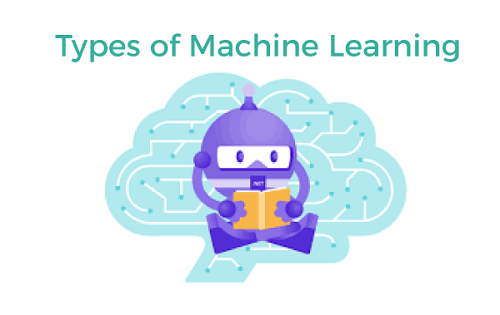
One of the most confusing questions among beginners is that are machine learning models, and algorithms are the same? Because in various cases in machine learning and data science, these two terms are used interchangeably.

The answer to this question is No, and the machine learning model is not the same as an algorithm. In a simple way, an **ML algorithm is like a procedure or method that runs on data to discover patterns from it** and generate the model. At the same time, a **machine learning model is like a computer program that generates output or makes predictions**. More specifically, when we train an algorithm with data, it becomes a model.

1. Machine Learning ModelModel = Model Data + Prediction Algorithm

# **Types of Machine Learning**

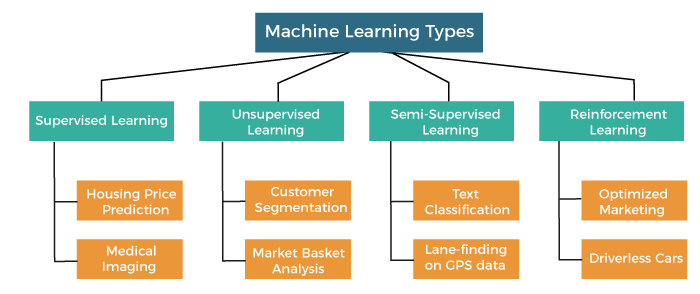
**Machine learning is a subset of AI, which enables the machine to automatically learn from data, improve performance from past experiences, and make predictions**. Machine learning contains a set of algorithms that work on a huge amount of data. Data is fed to these algorithms to train them, and on the basis of training, they build the model & perform a specific task.



These ML algorithms help to solve different business problems like Regression, Classification, Forecasting, Clustering, and Associations, etc.

Based on the methods and way of learning, machine learning is divided into mainly four types, which are:

1. Supervised Machine Learning
2. Unsupervised Machine Learning
3. Semi-Supervised Machine Learning
4. Reinforcement Learning



In this topic, we will provide a detailed description of the types of Machine Learning along with their respective algorithms:

Play Video

## **1. Supervised Machine Learning**

As its name suggests, [Supervised machine learning](https://www.javatpoint.com/supervised-machine-learning) is based on supervision. It means in the supervised learning technique, we train the machines using the "labelled" dataset, and based on the training, the machine predicts the output. Here, the labelled data specifies that some of the inputs are already mapped to the output. More preciously, we can say; first, we train the machine with the input and corresponding output, and then we ask the machine to predict the output using the test dataset.

Let's understand supervised learning with an example. Suppose we have an input dataset of cats and dog images. So, first, we will provide the training to the machine to understand the images, such as the **shape & size of the tail of cat and dog, Shape of eyes, colour, height (dogs are taller, cats are smaller), etc.** After completion of training, we input the picture of a cat and ask the machine to identify the object and predict the output. Now, the machine is well trained, so it will check all the features of the object, such as height, shape, colour, eyes, ears, tail, etc., and find that it's a cat. So, it will put it in the Cat category. This is the process of how the machine identifies the objects in Supervised Learning.

**The main goal of the supervised learning technique is to map the input variable(x) with the output variable(y).** Some real-world applications of supervised learning are **Risk Assessment, Fraud Detection, Spam filtering,** etc.

### Categories of Supervised Machine Learning

Supervised machine learning can be classified into two types of problems, which are given below:

* **Classification**
* **Regression**

### a) Classification

Classification algorithms are used to solve the classification problems in which the output variable is categorical, such as "**Yes" or No, Male or Female, Red or Blue, etc**. The classification algorithms predict the categories present in the dataset. Some real-world examples of classification algorithms are **Spam Detection, Email filtering, etc.**

Some popular classification algorithms are given below:

* **Random Forest Algorithm**
* **Decision Tree Algorithm**
* **Logistic Regression Algorithm**
* **Support Vector Machine Algorithm**

### b) Regression

Regression algorithms are used to solve regression problems in which there is a linear relationship between input and output variables. These are used to predict continuous output variables, such as market trends, weather prediction, etc.

Some popular Regression algorithms are given below:

* **Simple Linear Regression Algorithm**
* **Multivariate Regression Algorithm**
* **Decision Tree Algorithm**
* **Lasso Regression**

### Advantages and Disadvantages of Supervised Learning

**Advantages:**

* Since supervised learning work with the labelled dataset so we can have an exact idea about the classes of objects.
* These algorithms are helpful in predicting the output on the basis of prior experience.

**Disadvantages:**

* These algorithms are not able to solve complex tasks.
* It may predict the wrong output if the test data is different from the training data.
* It requires lots of computational time to train the algorithm.

### Applications of Supervised Learning

Some common applications of Supervised Learning are given below:

* **Image Segmentation:**  
  Supervised Learning algorithms are used in image segmentation. In this process, image classification is performed on different image data with pre-defined labels.
* **Medical Diagnosis:**  
  Supervised algorithms are also used in the medical field for diagnosis purposes. It is done by using medical images and past labelled data with labels for disease conditions. With such a process, the machine can identify a disease for the new patients.
* **Fraud Detection -** Supervised Learning classification algorithms are used for identifying fraud transactions, fraud customers, etc. It is done by using historic data to identify the patterns that can lead to possible fraud.
* **Spam detection -** In spam detection & filtering, classification algorithms are used. These algorithms classify an email as spam or not spam. The spam emails are sent to the spam folder.
* **Speech Recognition -** Supervised learning algorithms are also used in speech recognition. The algorithm is trained with voice data, and various identifications can be done using the same, such as voice-activated passwords, voice commands, etc.

## **2. Unsupervised Machine Learning**

[Unsupervised learnin](https://www.javatpoint.com/unsupervised-machine-learning)g is different from the Supervised learning technique; as its name suggests, there is no need for supervision. It means, in unsupervised machine learning, the machine is trained using the unlabeled dataset, and the machine predicts the output without any supervision.

In unsupervised learning, the models are trained with the data that is neither classified nor labelled, and the model acts on that data without any supervision.

**The main aim of the unsupervised learning algorithm is to group or categories the unsorted dataset according to the similarities, patterns, and differences.** Machines are instructed to find the hidden patterns from the input dataset.

Let's take an example to understand it more preciously; suppose there is a basket of fruit images, and we input it into the machine learning model. The images are totally unknown to the model, and the task of the machine is to find the patterns and categories of the objects.

So, now the machine will discover its patterns and differences, such as colour difference, shape difference, and predict the output when it is tested with the test dataset.

### Categories of Unsupervised Machine Learning

Unsupervised Learning can be further classified into two types, which are given below:

* **Clustering**
* **Association**

### 1) Clustering

The clustering technique is used when we want to find the inherent groups from the data. It is a way to group the objects into a cluster such that the objects with the most similarities remain in one group and have fewer or no similarities with the objects of other groups. An example of the clustering algorithm is grouping the customers by their purchasing behaviour.

Some of the popular clustering algorithms are given below:

* **K-Means Clustering algorithm**
* **Mean-shift algorithm**
* **DBSCAN Algorithm**
* **Principal Component Analysis**
* **Independent Component Analysis**

### 2) Association

Association rule learning is an unsupervised learning technique, which finds interesting relations among variables within a large dataset. The main aim of this learning algorithm is to find the dependency of one data item on another data item and map those variables accordingly so that it can generate maximum profit. This algorithm is mainly applied in **Market Basket analysis, Web usage mining, continuous production**, etc.

Some popular algorithms of Association rule learning are **Apriori Algorithm, Eclat, FP-growth algorithm.**

### Advantages and Disadvantages of Unsupervised Learning Algorithm

**Advantages:**

* These algorithms can be used for complicated tasks compared to the supervised ones because these algorithms work on the unlabeled dataset.
* Unsupervised algorithms are preferable for various tasks as getting the unlabeled dataset is easier as compared to the labelled dataset.

**Disadvantages:**

* The output of an unsupervised algorithm can be less accurate as the dataset is not labelled, and algorithms are not trained with the exact output in prior.
* Working with Unsupervised learning is more difficult as it works with the unlabelled dataset that does not map with the output.

### Applications of Unsupervised Learning

* **Network Analysis:** Unsupervised learning is used for identifying plagiarism and copyright in document network analysis of text data for scholarly articles.
* **Recommendation Systems:** Recommendation systems widely use unsupervised learning techniques for building recommendation applications for different web applications and e-commerce websites.
* **Anomaly Detection:** Anomaly detection is a popular application of unsupervised learning, which can identify unusual data points within the dataset. It is used to discover fraudulent transactions.
* **Singular Value Decomposition:** Singular Value Decomposition or SVD is used to extract particular information from the database. For example, extracting information of each user located at a particular location.

## **3. Semi-Supervised Learning**

**Semi-Supervised learning is a type of Machine Learning algorithm that lies between Supervised and Unsupervised machine learning**. It represents the intermediate ground between Supervised (With Labelled training data) and Unsupervised learning (with no labelled training data) algorithms and uses the combination of labelled and unlabeled datasets during the training period.

**A**lthough Semi-supervised learning is the middle ground between supervised and unsupervised learning and operates on the data that consists of a few labels, it mostly consists of unlabeled data. As labels are costly, but for corporate purposes, they may have few labels. It is completely different from supervised and unsupervised learning as they are based on the presence & absence of labels.

**To overcome the drawbacks of supervised learning and unsupervised learning algorithms, the concept of Semi-supervised learning is introduced**. The main aim of [semi-supervised learning](https://www.javatpoint.com/semi-supervised-learning) is to effectively use all the available data, rather than only labelled data like in supervised learning. Initially, similar data is clustered along with an unsupervised learning algorithm, and further, it helps to label the unlabeled data into labelled data. It is because labelled data is a comparatively more expensive acquisition than unlabeled data.

We can imagine these algorithms with an example. Supervised learning is where a student is under the supervision of an instructor at home and college. Further, if that student is self-analysing the same concept without any help from the instructor, it comes under unsupervised learning. Under semi-supervised learning, the student has to revise himself after analyzing the same concept under the guidance of an instructor at college.

### Advantages and disadvantages of Semi-supervised Learning

**Advantages:**

* It is simple and easy to understand the algorithm.
* It is highly efficient.
* It is used to solve drawbacks of Supervised and Unsupervised Learning algorithms.

**Disadvantages:**

* Iterations results may not be stable.
* We cannot apply these algorithms to network-level data.
* Accuracy is low.

**Unit 4**

**Hadoop**

Hadoop is an Apache open source framework written in java that allows distributed processing of large datasets across clusters of computers using simple programming models. The Hadoop framework application works in an environment that provides distributed *storage* and *computation* across clusters of computers. Hadoop is designed to scale up from single server to thousands of machines, each offering local computation and storage.

## **Hadoop Architecture**

At its core, Hadoop has two major layers namely −

* Processing/Computation layer (MapReduce), and
* Storage layer (Hadoop Distributed File System).



## **MapReduce**

MapReduce is a parallel programming model for writing distributed applications devised at Google for efficient processing of large amounts of data (multi-terabyte data-sets), on large clusters (thousands of nodes) of commodity hardware in a reliable, fault-tolerant manner. The MapReduce program runs on Hadoop which is an Apache open-source framework.

## **Hadoop Distributed File System**

The Hadoop Distributed File System (HDFS) is based on the Google File System (GFS) and provides a distributed file system that is designed to run on commodity hardware. It has many similarities with existing distributed file systems. However, the differences from other distributed file systems are significant. It is highly fault-tolerant and is designed to be deployed on low-cost hardware. It provides high throughput access to application data and is suitable for applications having large datasets.

Apart from the above-mentioned two core components, Hadoop framework also includes the following two modules −

* **Hadoop Common** − These are Java libraries and utilities required by other Hadoop modules.
* **Hadoop YARN** − This is a framework for job scheduling and cluster resource management.

## **How Does Hadoop Work?**

It is quite expensive to build bigger servers with heavy configurations that handle large scale processing, but as an alternative, you can tie together many commodity computers with single-CPU, as a single functional distributed system and practically, the clustered machines can read the dataset in parallel and provide a much higher throughput. Moreover, it is cheaper than one high-end server. So this is the first motivational factor behind using Hadoop that it runs across clustered and low-cost machines.

Hadoop runs code across a cluster of computers. This process includes the following core tasks that Hadoop performs −

* Data is initially divided into directories and files. Files are divided into uniform sized blocks of 128M and 64M (preferably 128M).
* These files are then distributed across various cluster nodes for further processing.
* HDFS, being on top of the local file system, supervises the processing.
* Blocks are replicated for handling hardware failure.
* Checking that the code was executed successfully.
* Performing the sort that takes place between the map and reduce stages.
* Sending the sorted data to a certain computer.
* Writing the debugging logs for each job.

## **Advantages of Hadoop**

* Hadoop framework allows the user to quickly write and test distributed systems. It is efficient, and it automatic distributes the data and work across the machines and in turn, utilizes the underlying parallelism of the CPU cores.
* Hadoop does not rely on hardware to provide fault-tolerance and high availability (FTHA), rather Hadoop library itself has been designed to detect and handle failures at the application layer.
* Servers can be added or removed from the cluster dynamically and Hadoop continues to operate without interruption.
* Another big advantage of Hadoop is that apart from being open source, it is compatible on all the platforms since it is Java based.

**NoSQL**

NoSQL is a type of database management system (DBMS) that is designed to handle and store large volumes of unstructured and semi-structured data. Unlike traditional relational databases that use tables with pre-defined schemas to store data, NoSQL databases use flexible data models that can adapt to changes in data structures and are capable of scaling horizontally to handle growing amounts of data.

The term NoSQL originally referred to “non-SQL” or “non-relational” databases, but the term has since evolved to mean “not only SQL,” as NoSQL databases have expanded to include a wide range of different database architectures and data models.

**NoSQL databases are generally classified into four main categories:**

1. Document databases: These databases store data as semi-structured documents, such as JSON or XML, and can be queried using document-oriented query languages.
2. Key-value stores: These databases store data as key-value pairs, and are optimized for simple and fast read/write operations.
3. Column-family stores: These databases store data as column families, which are sets of columns that are treated as a single entity. They are optimized for fast and efficient querying of large amounts of data.
4. Graph databases: These databases store data as nodes and edges, and are designed to handle complex relationships between data.
5. NoSQL databases are often used in applications where there is a high volume of data that needs to be processed and analyzed in real-time, such as social media analytics, e-commerce, and gaming. They can also be used for other applications, such as content management systems, document management, and customer relationship management.

However, NoSQL databases may not be suitable for all applications, as they may not provide the same level of data consistency and transactional guarantees as traditional relational databases. It is important to carefully evaluate the specific needs of an application when choosing a database management system.

**NoSQL** originally referring to non SQL or non relational is a database that provides a mechanism for storage and retrieval of data. This data is modeled in means other than the tabular relations used in relational databases. Such databases came into existence in the late 1960s, but did not obtain the NoSQL moniker until a surge of popularity in the early twenty-first century. NoSQL databases are used in real-time web applications and big data and their use are increasing over time.

* NoSQL systems are also sometimes called Not only SQL to emphasize the fact that they may support SQL-like query languages. A NoSQL database includes simplicity of design, simpler horizontal scaling to clusters of machines and finer control over availability. The data structures used by NoSQL databases are different from those used by default in relational databases which makes some operations faster in NoSQL. The suitability of a given NoSQL database depends on the problem it should solve.
* NoSQL databases, also known as “not only SQL” databases, are a new type of database management system that have gained popularity in recent years. Unlike traditional relational databases, NoSQL databases are designed to handle large amounts of unstructured or semi-structured data, and they can accommodate dynamic changes to the data model. This makes NoSQL databases a good fit for modern web applications, real-time analytics, and big data processing.
* Data structures used by NoSQL databases are sometimes also viewed as more flexible than relational database tables. Many NoSQL stores compromise consistency in favor of availability, speed and partition tolerance. Barriers to the greater adoption of NoSQL stores include the use of low-level query languages, lack of standardized interfaces, and huge previous investments in existing relational databases.
* Most NoSQL stores lack true ACID(Atomicity, Consistency, Isolation, Durability) transactions but a few databases, such as MarkLogic, Aerospike, FairCom c-treeACE, Google Spanner (though technically a NewSQL database), Symas LMDB, and OrientDB have made them central to their designs.
* Most NoSQL databases offer a concept of eventual consistency in which database changes are propagated to all nodes so queries for data might not return updated data immediately or might result in reading data that is not accurate which is a problem known as stale reads. Also some NoSQL systems may exhibit lost writes and other forms of data loss. Some NoSQL systems provide concepts such as write-ahead logging to avoid data loss.
* One simple example of a NoSQL database is a document database. In a document database, data is stored in documents rather than tables. Each document can contain a different set of fields, making it easy to accommodate changing data requirements
* For example, “Take, for instance, a database that holds data regarding employees.”. In a relational database, this information might be stored in tables, with one table for employee information and another table for department information. In a document database, each employee would be stored as a separate document, with all of their information contained within the document.
* NoSQL databases are a relatively new type of database management system that have gained popularity in recent years due to their scalability and flexibility. They are designed to handle large amounts of unstructured or semi-structured data and can handle dynamic changes to the data model. This makes NoSQL databases a good fit for modern web applications, real-time analytics, and big data processing.

**Key Features of NoSQL :**

1. **Dynamic schema:** NoSQL databases do not have a fixed schema and can accommodate changing data structures without the need for migrations or schema alterations.
2. **Horizontal scalability:** NoSQL databases are designed to scale out by adding more nodes to a database cluster, making them well-suited for handling large amounts of data and high levels of traffic.
3. **Document-based:**Some NoSQL databases, such as MongoDB, use a document-based data model, where data is stored in semi-structured format, such as JSON or BSON.
4. **Key-value-based:** Other NoSQL databases, such as Redis, use a key-value data model, where data is stored as a collection of key-value pairs.
5. **Column-based:**Some NoSQL databases, such as Cassandra, use a column-based data model, where data is organized into columns instead of rows.
6. **Distributed and high availability:** NoSQL databases are often designed to be highly available and to automatically handle node failures and data replication across multiple nodes in a database cluster.
7. **Flexibility:** NoSQL databases allow developers to store and retrieve data in a flexible and dynamic manner, with support for multiple data types and changing data structures.
8. **Performance:**NoSQL databases are optimized for high performance and can handle a high volume of reads and writes, making them suitable for big data and real-time applications.

**Advantages of NoSQL:** There are many advantages of working with NoSQL databases such as MongoDB and Cassandra. The main advantages are high scalability and high availability.

1. **High scalability :** NoSQL databases use sharding for horizontal scaling. Partitioning of data and placing it on multiple machines in such a way that the order of the data is preserved is sharding. Vertical scaling means adding more resources to the existing machine whereas horizontal scaling means adding more machines to handle the data. Vertical scaling is not that easy to implement but horizontal scaling is easy to implement. Examples of horizontal scaling databases are MongoDB, Cassandra, etc. NoSQL can handle a huge amount of data because of scalability, as the data grows NoSQL scale itself to handle that data in an efficient manner.
2. **Flexibility:** NoSQL databases are designed to handle unstructured or semi-structured data, which means that they can accommodate dynamic changes to the data model. This makes NoSQL databases a good fit for applications that need to handle changing data requirements.
3. **High availability :** Auto replication feature in NoSQL databases makes it highly available because in case of any failure data replicates itself to the previous consistent state.
4. **Scalability:** NoSQL databases are highly scalable, which means that they can handle large amounts of data and traffic with ease. This makes them a good fit for applications that need to handle large amounts of data or traffic
5. **Performance:** NoSQL databases are designed to handle large amounts of data and traffic, which means that they can offer improved performance compared to traditional relational databases.
6. **Cost-effectiveness:** NoSQL databases are often more cost-effective than traditional relational databases, as they are typically less complex and do not require expensive hardware or software.

**Disadvantages of NoSQL:** NoSQL has the following disadvantages.

1. **Lack of standardization** :  There are many different types of NoSQL databases, each with its own unique strengths and weaknesses. This lack of standardization can make it difficult to choose the right database for a specific application
2. **Lack of ACID compliance :** NoSQL databases are not fully ACID-compliant, which means that they do not guarantee the consistency, integrity, and durability of data. This can be a drawback for applications that require strong data consistency guarantees.
3. **Narrow focus :** NoSQL databases have a very narrow focus as it is mainly designed for storage but it provides very little functionality. Relational databases are a better choice in the field of Transaction Management than NoSQL.
4. **Open-source :** NoSQL is open-source database. There is no reliable standard for NoSQL yet. In other words, two database systems are likely to be unequal.
5. **Lack of support for complex queries :**NoSQL databases are not designed to handle complex queries, which means that they are not a good fit for applications that require complex data analysis or reporting.
6. **Lack of maturity :** NoSQL databases are relatively new and lack the maturity of traditional relational databases. This can make them less reliable and less secure than traditional databases.
7. **Management challenge :** The purpose of big data tools is to make the management of a large amount of data as simple as possible. But it is not so easy. Data management in NoSQL is much more complex than in a relational database. NoSQL, in particular, has a reputation for being challenging to install and even more hectic to manage on a daily basis.
8. **GUI is not available :** GUI mode tools to access the database are not flexibly available in the market.
9. **Backup :** Backup is a great weak point for some NoSQL databases like MongoDB. MongoDB has no approach for the backup of data in a consistent manner.
10. **Large document size :** Some database systems like MongoDB and CouchDB store data in JSON format. This means that documents are quite large (BigData, network bandwidth, speed), and having descriptive key names actually hurts since they increase the document size.

**Types of NoSQL database:** Types of NoSQL databases and the name of the databases system that falls in that category are:

1. **Graph Databases**: Examples – Amazon Neptune, Neo4j
2. **Key value store:** Examples – Memcached, Redis, Coherence
3. **Tabular:** Examples – Hbase, Big Table, Accumulo
4. **Document-based:** Examples – MongoDB, CouchDB, Cloudant

**When should NoSQL be used:**

1. When a huge amount of data needs to be stored and retrieved.
2. The relationship between the data you store is not that important
3. The data changes over time and is not structured.
4. Support of Constraints and Joins is not required at the database level
5. The data is growing continuously and you need to scale the database regularly to handle the data.

In conclusion, NoSQL databases offer several benefits over traditional relational databases, such as scalability, flexibility, and cost-effectiveness. However, they also have several drawbacks, such as a lack of standardization, lack of ACID compliance, and lack of support for complex queries. When choosing a database for a specific application, it is important to weigh the benefits and drawbacks carefully to determine the best fit.

# **CAP Theorem in Hadoop**

## What is CAP Theorem?

CAP theorem is designed for distributed file systems(collection of interconnected nodes).CAP Theorem also known as Brewer’s theorem and used to distributed consistency.It contains follwing three technical terms for distributed systems.

**C** – Consistency

**A** – Availability

**P** – Partition Tolerance

**Consistency:**

When you read data it will give same data how many times read and server send response each and every request but systems always consistent when read data.(all node having same data)

**Availability:**

It means all requests give response and no error accured in this systems.

**Partition Tolerance:**

All functions run all time when more nodes not responsive and commnication break between two nodes

Distributed systems statisfy any two terms only and not satisfy three terms

## [Selecting Two options in CAP Theorem:](http://www.credosystemz.com/training-in-chennai/best-big-data-training-in-chennai/)

**CP – Consistency/Partition Tolerance:**

It wait for response form partioned nodes and that data are timeout error.system give data and it distributed across the hadoop cluster.System return error that based on you desire.It is good for marketing because which system does not work partition data will tagged to user.

**AP – Availability/Partition Tolerance:**

It returns the recent data from the systems and it allows write options that processed when partion resolved.Availability compeling the data when come external error come from system functions.Final decision is ***software trade off*** in that systems.In this process any network problem occurs control options are there in your hand only because network problem is only temporary.

**CA – Consistency/Availability:**

All realational database are CA.In CA all nodes are always in contact.Partion occurs in system that system will be blocked.

## [Spaces Defined by CAP:](http://www.credosystemz.com/training-in-chennai/best-big-data-training-in-chennai/)

**NoSpace:**

It is called empty set because there is no any database engines.It not satisty any theorem properties

**CD Space:**

Space between consistency and availability.In CD Space distribution does not exist and find NoSql database.Relational database are postioned in this space

**CT Space:**

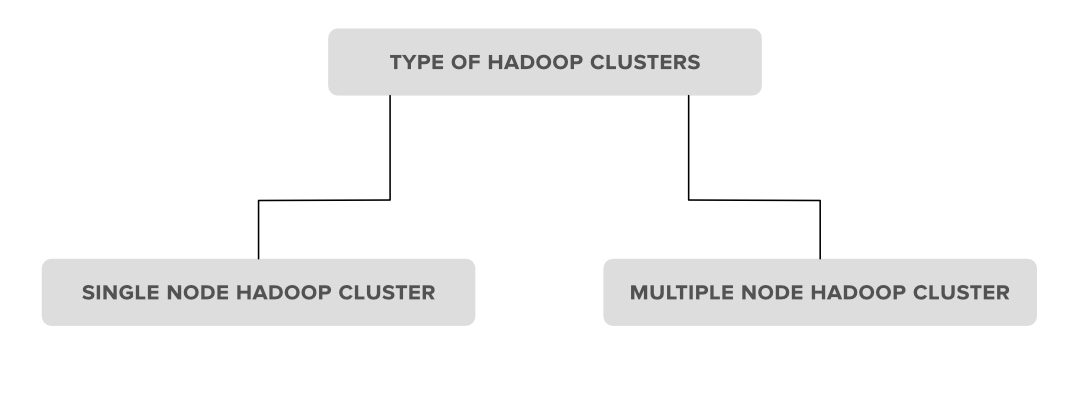
It don’t have any consistency and it will favor of availability.This is a relative database and cannot guarntee between nodes

**DT Space:**

It offers some consistency and leaving certain levels of availability.In this space database not response the client enquires.

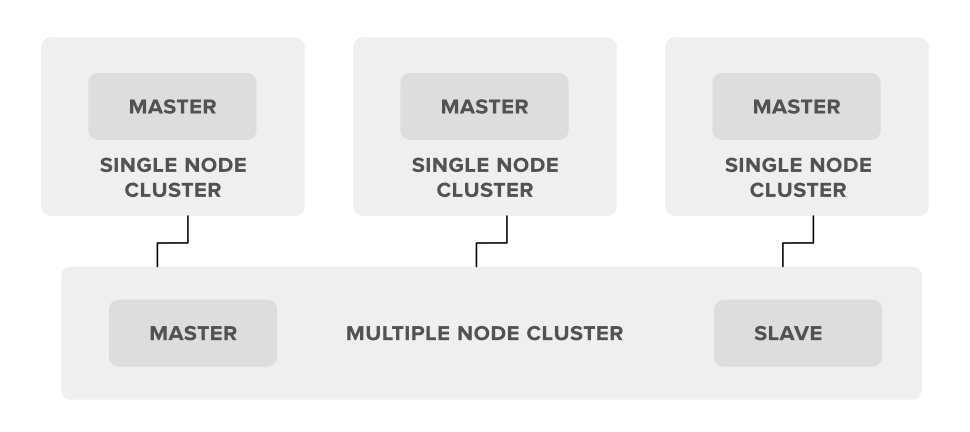
### Types of Hadoop clusters

**1. Single Node Hadoop Cluster**  
**2. Multiple Node Hadoop Cluster**



**1. Single Node Hadoop Cluster:** In Single Node Hadoop Cluster as the name suggests the cluster is of an only single node which means all our Hadoop Daemons i.e. Name Node, Data Node, Secondary Name Node, Resource Manager, Node Manager will run on the same system or on the same machine. It also means that all of our processes will be handled by only single JVM(Java Virtual Machine) Process Instance.

**2. Multiple Node Hadoop Cluster:**In multiple node Hadoop clusters as the name suggests it contains multiple nodes. In this kind of cluster set up all of our Hadoop Daemons, will store in different-different nodes in the same cluster setup. In general, in multiple node Hadoop cluster setup we try to utilize our higher processing nodes for Master i.e. Name node and Resource Manager and we utilize the cheaper system for the slave Daemon’s i.e.Node Manager and Data Node.



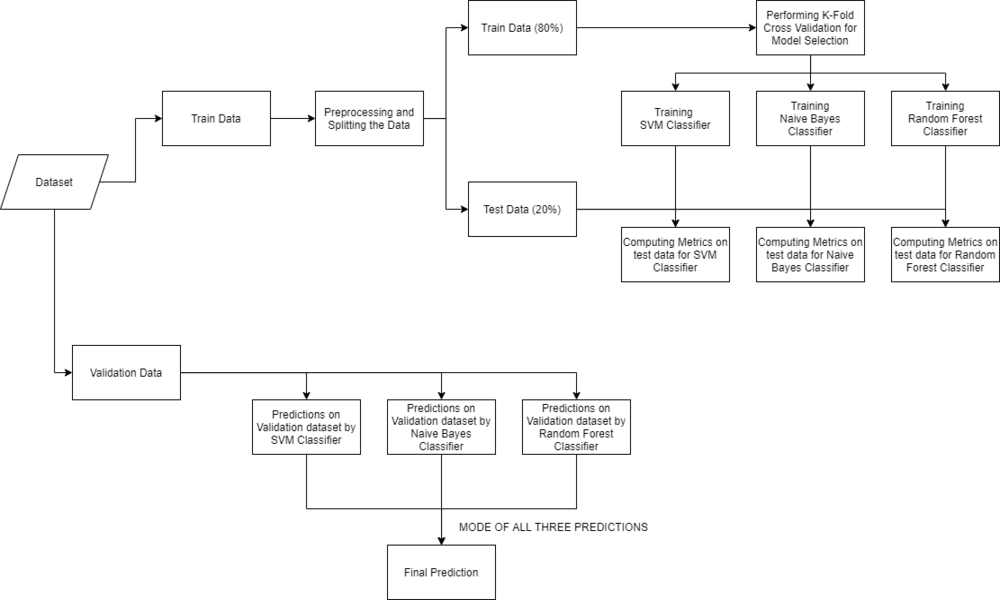
# **Disease Prediction Using Machine Learning**

**Approach:**

* **Gathering the Data:**Data preparation is the primary step for any machine learning problem. We will be using a [dataset](https://www.kaggle.com/kaushil268/disease-prediction-using-machine-learning)from Kaggle for this problem. This dataset consists of two CSV files one for training and one for testing. There is a total of 133 columns in the dataset out of which 132 columns represent the symptoms and the last column is the prognosis.
* **Cleaning the Data:**Cleaning is the most important step in a machine learning project. The quality of our data determines the quality of our machine-learning model. So it is always necessary to clean the data before feeding it to the model for training. In our dataset all the columns are numerical, the target column i.e. prognosis is a string type and is encoded to numerical form using a [label encoder](https://www.geeksforgeeks.org/ml-label-encoding-of-datasets-in-python/).
* **Model Building:**After gathering and cleaning the data, the data is ready and can be used to train a machine learning model. We will be using this cleaned data to train the Support Vector Classifier, Naive Bayes Classifier, and Random Forest Classifier. We will be using a [confusion matrix](https://www.geeksforgeeks.org/confusion-matrix-machine-learning/) to determine the quality of the models.
* **Inference:**After training the three models we will be predicting the disease for the input symptoms by combining the predictions of all three models. This makes our overall prediction more robust and accurate.

At last, we will be defining a function that takes symptoms separated by commas as input, predicts the disease based on the symptoms by using the trained models, and returns the predictions in a JSON format.

**Implementation:**



Make sure that the [Training](https://media.geeksforgeeks.org/wp-content/cdn-uploads/20210826192554/Training.csv) and [Testing](https://media.geeksforgeeks.org/wp-content/cdn-uploads/20210826192633/Testing.csv) are downloaded and the train.csv, test.csv are put in the dataset folder. Open jupyter notebook and run the code individually for better understanding.

* Python3

|  |
| --- |
| # Importing libraries  import numpy as np  import pandas as pd  from scipy.stats import mode  import matplotlib.pyplot as plt  import seaborn as sns  from sklearn.preprocessing import LabelEncoder  from sklearn.model\_selection import train\_test\_split, cross\_val\_score  from sklearn.svm import SVC  from sklearn.naive\_bayes import GaussianNB  from sklearn.ensemble import RandomForestClassifier  from sklearn.metrics import accuracy\_score, confusion\_matrix    %matplotlib inline |

## **Reading the dataset**

Firstly we will be loading the dataset from the folders using the pandas library. While reading the dataset we will be dropping the null column. This dataset is a clean dataset with no null values and all the features consist of 0’s and 1s. Whenever we are solving a classification task it is necessary to check whether our target column is balanced or not. We will be using a bar plot, to check whether the dataset is balanced or not.

* Python3

|  |
| --- |
| # Reading the train.csv by removing the  # last column since it's an empty column  DATA\_PATH = "dataset/Training.csv"  data = pd.read\_csv(DATA\_PATH).dropna(axis = 1)    # Checking whether the dataset is balanced or not  disease\_counts = data["prognosis"].value\_counts()  temp\_df = pd.DataFrame({      "Disease": disease\_counts.index,      "Counts": disease\_counts.values  })    plt.figure(figsize = (18,8))  sns.barplot(x = "Disease", y = "Counts", data = temp\_df)  plt.xticks(rotation=90)  plt.show() |

**Output:**



From the above plot, we can observe that the dataset is a balanced dataset i.e. there are exactly 120 samples for each disease, and no further balancing is required. We can notice that our target column i.e. prognosis column is of object datatype, this format is not suitable to train a machine learning model. So, we will be using a label encoder to convert the prognosis column to the numerical datatype. Label Encoder converts the labels into numerical form by assigning a unique index to the labels. If the total number of labels is n, then the numbers assigned to each label will be between 0 to n-1.

* Python3

|  |
| --- |
| # Encoding the target value into numerical  # value using LabelEncoder  encoder = LabelEncoder()  data["prognosis"] = encoder.fit\_transform(data["prognosis"]) |

## **Splitting the data for training and testing the model**

Now that we have cleaned our data by removing the Null values and converting the labels to numerical format, It’s time to split the data to train and test the model. We will be splitting the data into 80:20 format i.e. 80% of the dataset will be used for training the model and 20% of the data will be used to evaluate the performance of the models.

* Python3

|  |
| --- |
| X = data.iloc[:,:-1]  y = data.iloc[:, -1]  X\_train, X\_test, y\_train, y\_test =train\_test\_split(    X, y, test\_size = 0.2, random\_state = 24)    print(f"Train: {X\_train.shape}, {y\_train.shape}")  print(f"Test: {X\_test.shape}, {y\_test.shape}") |

**Output:**

Train: (3936, 132), (3936,)

Test: (984, 132), (984,)

### ****Model Building****

After splitting the data, we will be now working on the modeling part. We will be using K-Fold cross-validation to evaluate the machine-learning models. We will be using Support Vector Classifier, Gaussian Naive Bayes Classifier, and Random Forest Classifier for cross-validation. Before moving into the implementation part let us get familiar with k-fold cross-validation and the machine learning models.

* **K-Fold Cross-Validation:**K-Fold cross-validation is one of the cross-validation techniques in which the whole dataset is split into k number of subsets, also known as folds, then training of the model is performed on the k-1 subsets and the remaining one subset is used to evaluate the model performance.
* **Support Vector Classifier:**Support Vector Classifier is a discriminative classifier i.e. when given a labeled training data, the algorithm tries to find an optimal hyperplane that accurately separates the samples into different categories in hyperspace.
* **Gaussian Naive Bayes Classifier:**It is a probabilistic machine learning algorithm that internally uses Bayes Theorem to classify the data points.
* **Random Forest Classifier:**Random Forest is an ensemble learning-based supervised machine learning classification algorithm that internally uses multiple decision trees to make the classification. In a random forest classifier, all the internal decision trees are weak learners, and the outputs of these weak decision trees are combined i.e. mode of all the predictions is as the final prediction.

**Using K-Fold Cross-Validation for model selection**

* Python3

|  |
| --- |
| # Defining scoring metric for k-fold cross validation  def cv\_scoring(estimator, X, y):      return accuracy\_score(y, estimator.predict(X))    # Initializing Models  models = {      "SVC":SVC(),      "Gaussian NB":GaussianNB(),      "Random Forest":RandomForestClassifier(random\_state=18)  }    # Producing cross validation score for the models  for model\_name in models:      model = models[model\_name]      scores = cross\_val\_score(model, X, y, cv = 10,                               n\_jobs = -1,                               scoring = cv\_scoring)      print("=="\*30)      print(model\_name)      print(f"Scores: {scores}")      print(f"Mean Score: {np.mean(scores)}") |

**Output:**

*============================================================*

*SVC*

*Scores: [1. 1. 1. 1. 1. 1. 1. 1. 1. 1.]*

*Mean Score: 1.0*

*============================================================*

*Gaussian NB*

*Scores: [1. 1. 1. 1. 1. 1. 1. 1. 1. 1.]*

*Mean Score: 1.0*

*============================================================*

*Random Forest*

*Scores: [1. 1. 1. 1. 1. 1. 1. 1. 1. 1.]*

*Mean Score: 1.0*

From the above output, we can notice that all our machine learning algorithms are performing very well and the mean scores after k fold cross-validation are also very high. To build a robust model we can combine i.e. take the mode of the predictions of all three models so that even one of the models makes wrong predictions and the other two make correct predictions then the final output would be the correct one. This approach will help us to keep the predictions much more accurate on completely unseen data. In the below code we will be training all the three models on the train data, checking the quality of our models using a confusion matrix, and then combine the predictions of all three models.

**Building robust classifier by combining all models:**

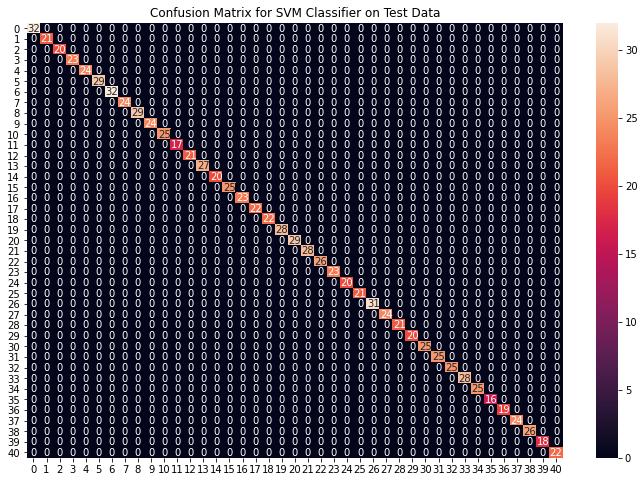
* Python3

|  |
| --- |
| # Training and testing SVM Classifier  svm\_model = SVC()  svm\_model.fit(X\_train, y\_train)  preds = svm\_model.predict(X\_test)    print(f"Accuracy on train data by SVM Classifier\  : {accuracy\_score(y\_train, svm\_model.predict(X\_train))\*100}")    print(f"Accuracy on test data by SVM Classifier\  : {accuracy\_score(y\_test, preds)\*100}")  cf\_matrix = confusion\_matrix(y\_test, preds)  plt.figure(figsize=(12,8))  sns.heatmap(cf\_matrix, annot=True)  plt.title("Confusion Matrix for SVM Classifier on Test Data")  plt.show()    # Training and testing Naive Bayes Classifier  nb\_model = GaussianNB()  nb\_model.fit(X\_train, y\_train)  preds = nb\_model.predict(X\_test)  print(f"Accuracy on train data by Naive Bayes Classifier\  : {accuracy\_score(y\_train, nb\_model.predict(X\_train))\*100}")    print(f"Accuracy on test data by Naive Bayes Classifier\  : {accuracy\_score(y\_test, preds)\*100}")  cf\_matrix = confusion\_matrix(y\_test, preds)  plt.figure(figsize=(12,8))  sns.heatmap(cf\_matrix, annot=True)  plt.title("Confusion Matrix for Naive Bayes Classifier on Test Data")  plt.show()    # Training and testing Random Forest Classifier  rf\_model = RandomForestClassifier(random\_state=18)  rf\_model.fit(X\_train, y\_train)  preds = rf\_model.predict(X\_test)  print(f"Accuracy on train data by Random Forest Classifier\  : {accuracy\_score(y\_train, rf\_model.predict(X\_train))\*100}")    print(f"Accuracy on test data by Random Forest Classifier\  : {accuracy\_score(y\_test, preds)\*100}")    cf\_matrix = confusion\_matrix(y\_test, preds)  plt.figure(figsize=(12,8))  sns.heatmap(cf\_matrix, annot=True)  plt.title("Confusion Matrix for Random Forest Classifier on Test Data")  plt.show() |

**Output:**

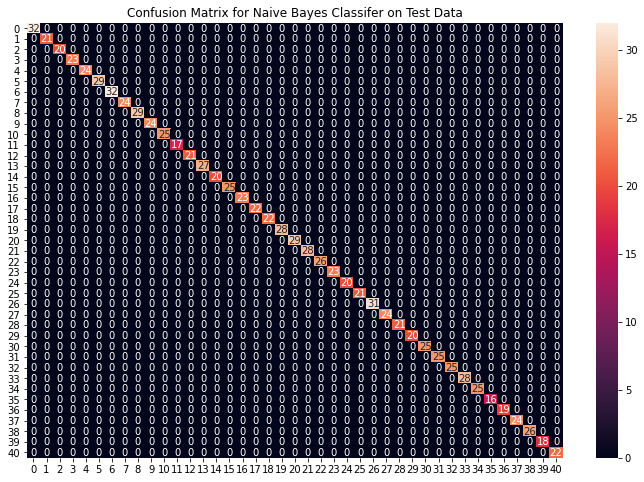
Accuracy on train data by SVM Classifier: 100.0

Accuracy on test data by SVM Classifier: 100.0



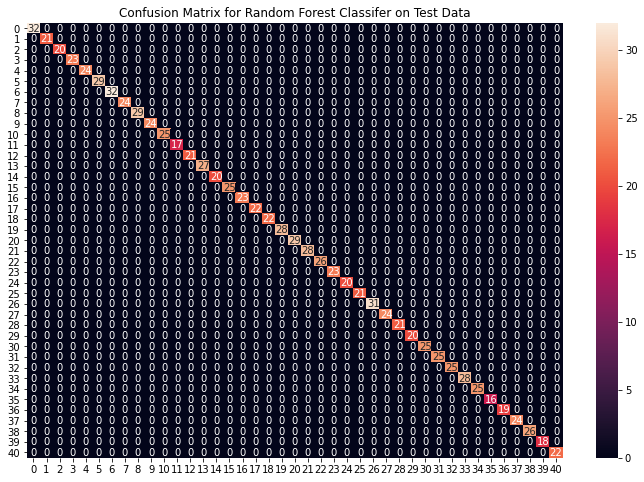
Accuracy on train data by Naive Bayes Classifier: 100.0

Accuracy on test data by Naive Bayes Classifier: 100.0



Accuracy on train data by Random Forest Classifier: 100.0

Accuracy on test data by Random Forest Classifier: 100.0



From the above confusion matrices, we can see that the models are performing very well on the unseen data. Now we will be training the models on the whole train data present in the dataset that we downloaded and then test our combined model on test data present in the dataset.

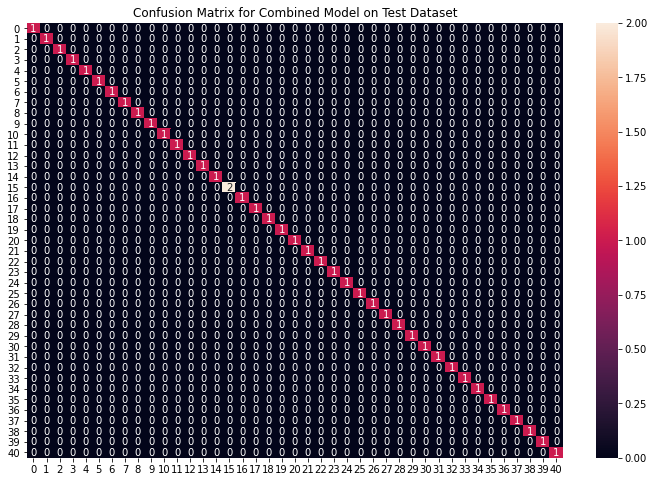
**Fitting the model on whole data and validating on the Test dataset:**

* Python3

|  |
| --- |
| # Training the models on whole data  final\_svm\_model = SVC()  final\_nb\_model = GaussianNB()  final\_rf\_model = RandomForestClassifier(random\_state=18)  final\_svm\_model.fit(X, y)  final\_nb\_model.fit(X, y)  final\_rf\_model.fit(X, y)    # Reading the test data  test\_data = pd.read\_csv("./dataset/Testing.csv").dropna(axis=1)    test\_X = test\_data.iloc[:, :-1]  test\_Y = encoder.transform(test\_data.iloc[:, -1])    # Making prediction by take mode of predictions  # made by all the classifiers  svm\_preds = final\_svm\_model.predict(test\_X)  nb\_preds = final\_nb\_model.predict(test\_X)  rf\_preds = final\_rf\_model.predict(test\_X)    final\_preds = [mode([i,j,k])[0][0] for i,j,                 k in zip(svm\_preds, nb\_preds, rf\_preds)]    print(f"Accuracy on Test dataset by the combined model\  : {accuracy\_score(test\_Y, final\_preds)\*100}")    cf\_matrix = confusion\_matrix(test\_Y, final\_preds)  plt.figure(figsize=(12,8))    sns.heatmap(cf\_matrix, annot = True)  plt.title("Confusion Matrix for Combined Model on Test Dataset")  plt.show() |

**Output:**

Accuracy on Test dataset by the combined model: 100.0



We can see that our combined model has classified all the data points accurately. We have come to the final part of this whole implementation, we will be creating a function that takes symptoms separated by commas as input and outputs the predicted disease using the combined model based on the input symptoms.

**Creating a function that can take symptoms as input and generate predictions for disease**

* Python3

|  |
| --- |
| symptoms = X.columns.values    # Creating a symptom index dictionary to encode the  # input symptoms into numerical form  symptom\_index = {}  for index, value in enumerate(symptoms):      symptom = " ".join([i.capitalize() for i in value.split("\_")])      symptom\_index[symptom] = index    data\_dict = {      "symptom\_index":symptom\_index,      "predictions\_classes":encoder.classes\_  }    # Defining the Function  # Input: string containing symptoms separated by commas  # Output: Generated predictions by models  def predictDisease(symptoms):      symptoms = symptoms.split(",")        # creating input data for the models      input\_data = [0] \* len(data\_dict["symptom\_index"])      for symptom in symptoms:          index = data\_dict["symptom\_index"][symptom]          input\_data[index] = 1        # reshaping the input data and converting it      # into suitable format for model predictions      input\_data = np.array(input\_data).reshape(1,-1)        # generating individual outputs      rf\_prediction = data\_dict["predictions\_classes"][final\_rf\_model.predict(input\_data)[0]]      nb\_prediction = data\_dict["predictions\_classes"][final\_nb\_model.predict(input\_data)[0]]      svm\_prediction = data\_dict["predictions\_classes"][final\_svm\_model.predict(input\_data)[0]]        # making final prediction by taking mode of all predictions      final\_prediction = mode([rf\_prediction, nb\_prediction, svm\_prediction])[0][0]      predictions = {          "rf\_model\_prediction": rf\_prediction,          "naive\_bayes\_prediction": nb\_prediction,          "svm\_model\_prediction": svm\_prediction,          "final\_prediction":final\_prediction      }      return predictions    # Testing the function  print(predictDisease("Itching,Skin Rash,Nodal Skin Eruptions")) |

**Output:**

{

'rf\_model\_prediction': 'Fungal infection',

'naive\_bayes\_prediction': 'Fungal infection',

'svm\_model\_prediction': 'Fungal infection',

'final\_prediction': 'Fungal infection'

}

***Note:****The symptoms that are given as input to the function should be exactly the same among the 132 symptoms in the dataset.*