#### Naïve Bayes Classification :

1. 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.
2. The fundamental Naive Bayes assumption is that each feature makes an:
   1. Independent : We assume that no pair of features are dependent.
   2. Equal contribution to the outcome : each feature is given the same weight

##### **Bayes' Theorem :**

* 1. Bayes’ Theorem finds the probability of an event occurring given the probability of another event that has already occurred
  2. Now, we apply the Bayes' Theorem in the way ,

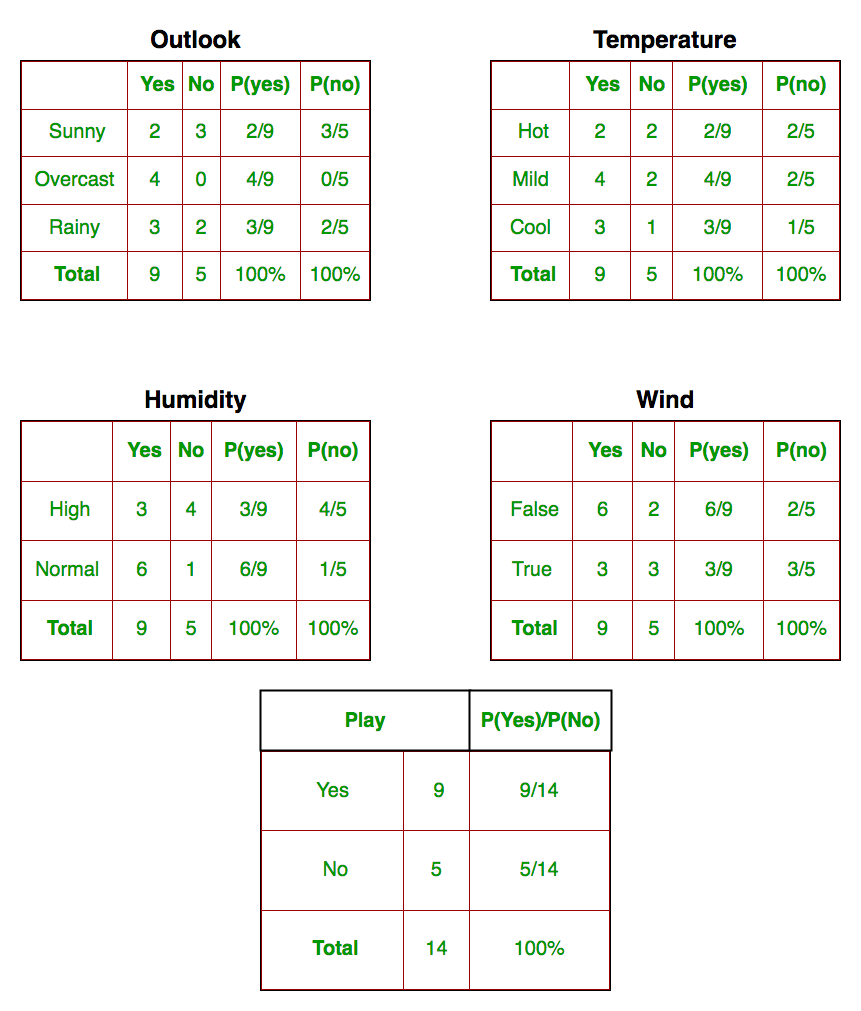
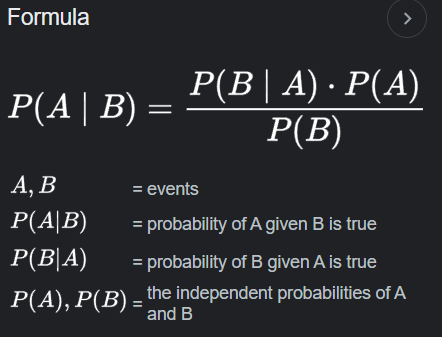




* 2. 





1. Once we apply the Bayes' Theorem we need to find the value of P(y) and P(xi|y)
2. Types of Naïve Bayes:
   1. Gaussian NB : In Gaussian Naive Bayes, continuous values associated with each feature are assumed to be distributed according to a **Gaussian distribution**.



* 1. Multinomial NB : 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.



* 1. Bernoulli NB : In the multivariate Bernoulli event model, features are independent booleans (binary variables) describing inputs.



* 1. Complement NB:



e. Categorical NB :



#### 

#### 

#### 

#### 

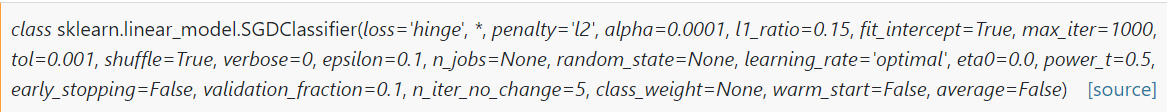
#### 

#### 

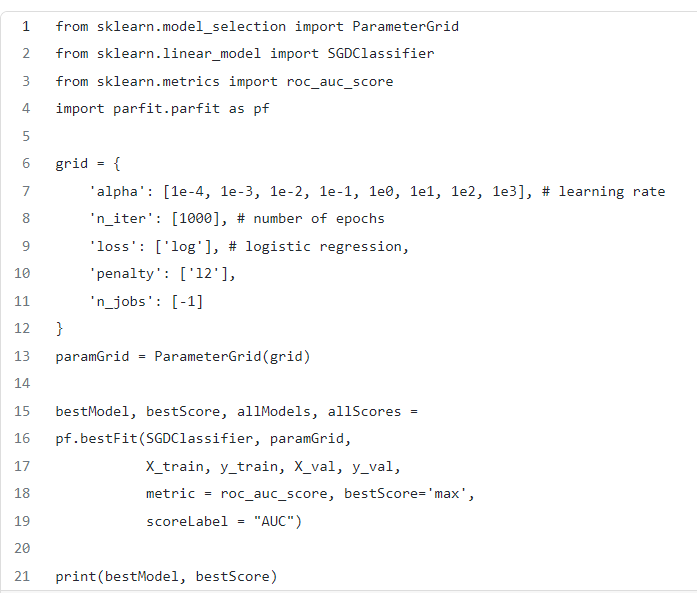
#### 

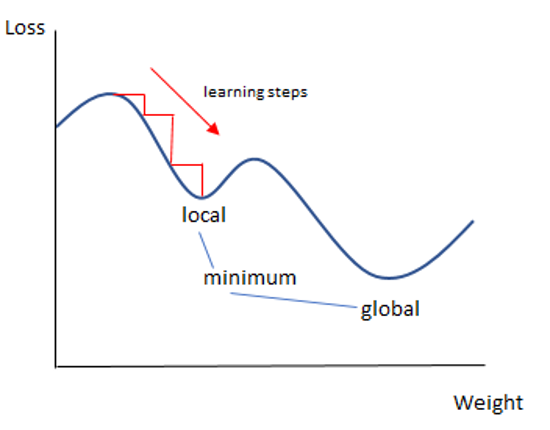
#### 

#### Stochastic Gradient Descent (SGD) Classifier :

******

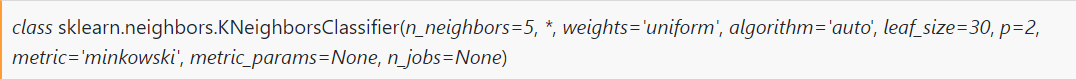
1. SGD Classifier implements regularized linear models with Stochastic Gradient Descent.
2. Stochastic gradient descent considers only 1 random point while changing weights unlike gradient descent which considers the whole training data as such it would be better to use SGDClassifier on larger data sets.
3. The SGDClassifier does not perform as well as the Logistic Regression. It requires some hyper parameter tuning to be done, which is done by 'parfit'.
4. We will use the ‘L2’ penalty for SGDClassifier. Important hyper-parameters to note here are n\_iter and max\_iter.
5. The alpha hyper-parameter serves as both a regularization parameter and the initial learning rate under the default schedule. Lower the value of alpha, higher the value we need for n\_iter.
6. SGDClassifier is a Linear classifiers (SVM, logistic regression, a.o.) with SGD training.



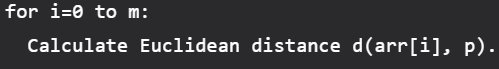


#### 

#### K-Nearest Neighbor :

****

1. This algorithm is used to solve the classification model problems.
2. The K-nearest neighbor or K-NN algorithm basically creates an imaginary boundary to classify the data. When new data points come in, the algorithm will try to predict that to the nearest of the boundary line.
3. Larger k value means smoother curves of separation resulting in less complex models. Whereas, smaller k values tend to overfit the data and result in complex models.
4. 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
5. Algorithm :
6. 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 elements of this array represents a tuple (x, y)

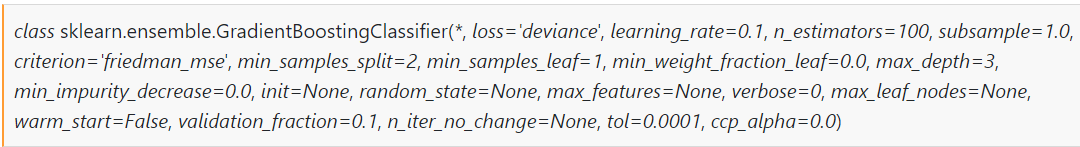


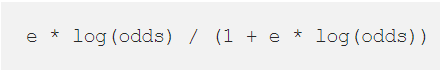
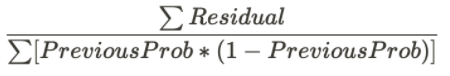
* 1. Make the set S of K smallest distances obtained. Each of these distances corresponds to an already classified data point.
  2. Return the majority label among S

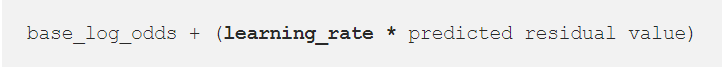
6. Time Complexity : O(N \* logN)

Find if we can find the best K value

#### Gradient Boosting :

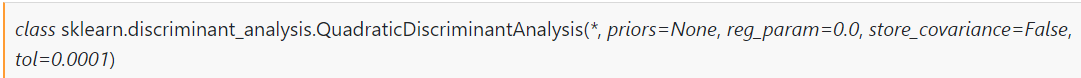


1. In Ensemble Learning, instead of using a single predictor, multiple predictors and training in the data and their results are aggregated, usually giving a better score than using a single model.
2. *Boosting* is a special type of Ensemble Learning technique that works by combining several *weak learners(*predictors with poor accuracy*)* into a strong learner(a model with strong accuracy).
3. In Gradient Boosting, each predictor tries to improve on its predecessor by reducing the errors. It actually fits a new predictor t*o the residual errors made by the previous predictor.*
4. *Algorithm :* 
   1. *In order to make initial predictions on the data, the algorithm will get the log of the odds of the target feature. This is usually the number of True values(values equal to 1) divided by the number of False values(values equal to 0).*
   2. *Once it has the log(odds), we convert that value to a probability by using a logistic function in order to make predictions.*
   3. **
   4. *i. For every instance in the training set, it calculates the residuals for that instance, or, in other words, the observed value minus the predicted value.*
   5. *ii. Once it has done this, it build a new Decision Tree that actually tries to predict the residuals that was previously calculated.*
   6. *Since we have limited number of leaves in the Decision tree we may have an instance where multiple instances fall into same leaf. Unlike Gradient Boosting for Regression, where we could simply average the instance values to get an output value, and leave the single instance as a leaf of its own, we have to transform these values using a formula:*
   7. **
   8. Now, to make new predictions, we do 2 things:
      1. Get the log(odds) prediction for each instance in the training set
      2. Convert that prediction into a probability



1. After we have done this process, we calculate the new residuals of the tree and create a new tree to fit the new residuals. Again, the process is repeated until a certain predefined threshold is reached, or the residuals are negligible.
2. Important Parameters:
   1. **Criterion** : The loss function used to find the optimal feature and threshold to split the data
   2. **learning\_rate** : this parameter scales the contribution of each tree
   3. **max\_depth** : the maximum depth of each tree
   4. **n\_estimators** : the number of trees to construct
   5. **init** (the initial estimator) : By default, it is the log(odds) converted to a probability

#### Quadratic Discriminant Analysis :



1. Quadratic discriminant analysis is quite similar to Linear discriminant analysis except we relaxed the assumption that the mean and covariance of all the classes were equal.
2. Now for each of the class y the covariance matrix is given by :



1. By adding the following term and solving, the Quadratic Discriminant function is given by:



##### **Linear Discriminant Analysis(LDA) :**

* 1. In a classification problem represented by Bayes’ Probability Distribution, LDA does it differently by trying to model the distribution of X given the predictor class:

* 1. In LDA, we assume that P(X | Y = k) can be estimated to the multivariate Normal distribution that is given by the following equation:





* 1. The final equation after the assumptions is:

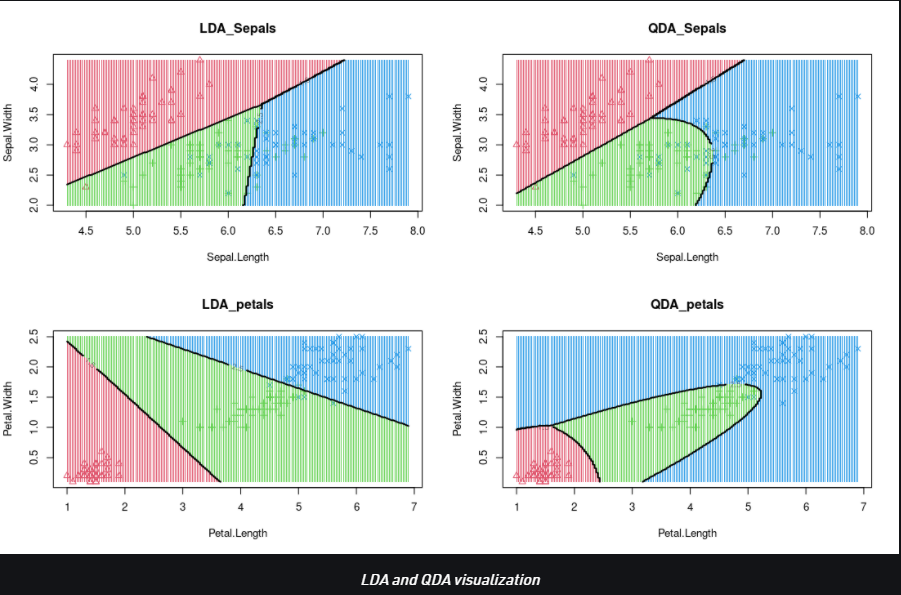


* 1. Taking Log on both sides gives us a decision boundary
     1. For two classes we get a linear function of X, where both class give equal value

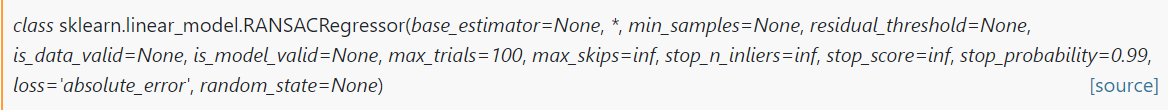




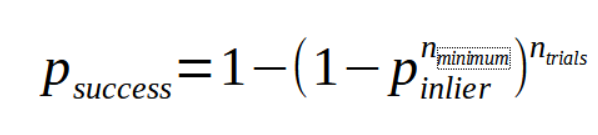
* + 1. For (K>2), we need to estimate the pK means, pK variance, K prior proportions and 

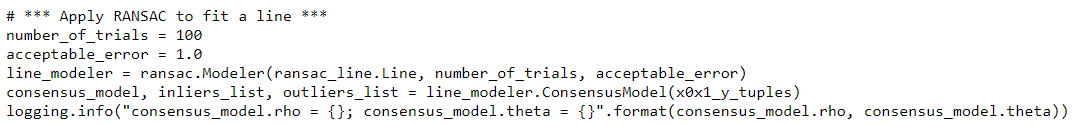


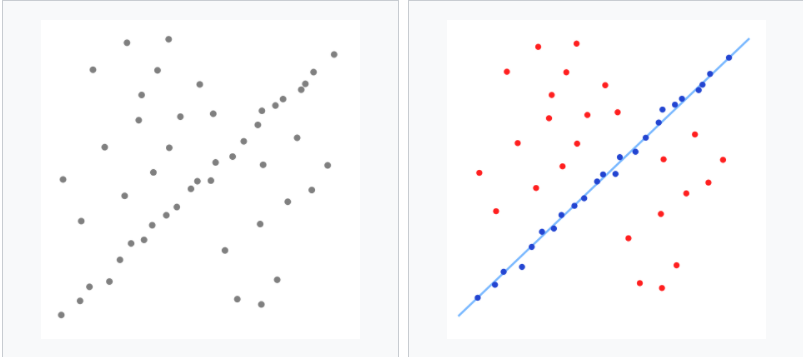
#### Random Sample Consensus :



1. **Random sample consensus** (**RANSAC**) is an [iterative method](https://en.wikipedia.org/wiki/Iterative_method) to estimate parameters of a mathematical model from a set of observed data that contains [outliers](https://en.wikipedia.org/wiki/Outliers), when outliers are to be accorded no influence on the values of the estimates. Therefore, it also can be interpreted as an outlier detection method.
2. RANSAC uses [repeated random sub-sampling](https://en.wikipedia.org/wiki/Cross-validation_(statistics)#Repeated_random_sub-sampling_validation).
3. Assumptions :
   1. A basic assumption is that the data consists of "inliers", i.e., data whose distribution can be explained by some set of model parameters, though may be subject to noise, and "outliers" which are data that do not fit the model.
   2. It also assumes that, given a (usually small) set of inliers, there exists a procedure which can estimate the parameters of a model that optimally explains or fits this data.
4. A simple example is [fitting a line](https://en.wikipedia.org/wiki/Regression_analysis) in two dimensions to a set of observations.
   1. Assuming that this set contains both inliers and outliers, points which cannot be fitted to this line, a [simple least squares method](https://en.wikipedia.org/wiki/Ordinary_least_squares) for line fitting will generally produce a line with a bad fit to the data including inliers and outliers, because it is fitted to all points including the outliers.
   2. RANSAC instead attempts to exclude the outliers and find a linear model that only uses the inliers in its calculation by fitting linear models to several random samplings of the data and returning the model that has the best fit to a subset of the data.
   3. The probability of the algorithm succeeding depends on the proportion of inliers in the data as well as the choice of several algorithm parameters.
5. The basic idea of the algorithm is to generate candidate models from multiple randomly selected tuples of observations







#### 

| ***Algorithms*** | ***When to use*** |
| --- | --- |
| *Naive Bayes’ Classification* | *-> Multi-class prediction*  *-> small dataset which satisfies assumptions*  *-> Majority of features as categorical* |
| *SGD* | *-> Fast and easy to fit algo thus lower computational requirements* |
| *K-nearest Neighbor* | *-> Small dataset* |
| *Gradient Boosting* | *-> Large and complex dataset*  *->*  *-> Gives better result compared to counterpart but difficult to fit*  *-> Lowers bias-error* |
| *QDA* | *-> Non-linear variables* |
| *RANSAC* | *-> Outliers are very spread out and effects the best fit line* |

#### 

#### 

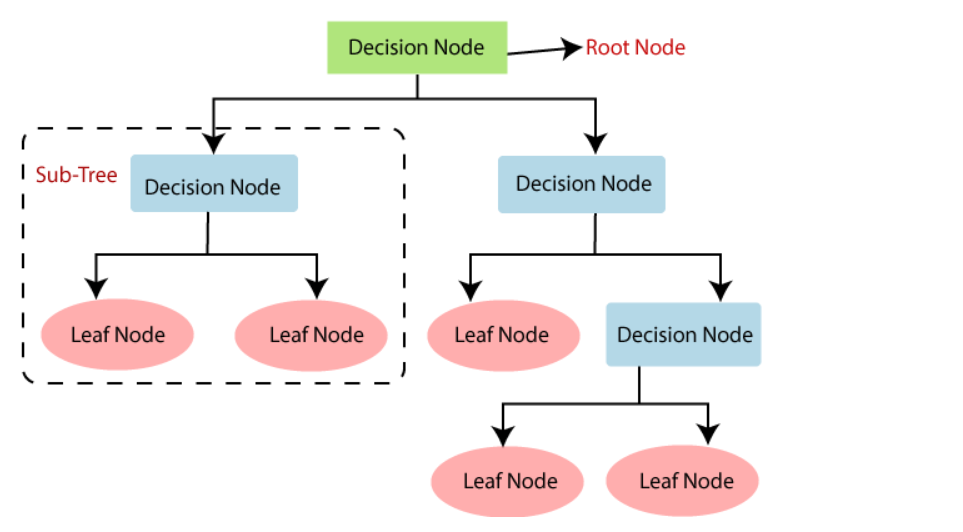
***Decision Tree:***

* A **Supervised learning technique**that can be used for both classification and Regression problems, but mostly it is preferred for solving Classification problems. A decision tree simply asks a question, and based on the answer (Yes/No), it further splits the tree into subtrees.

Attribute Selection Measures

* **Information Gain**
* **Gini Index**

**Gini index is used for node split, lower the gini index better the feature for the node split.**



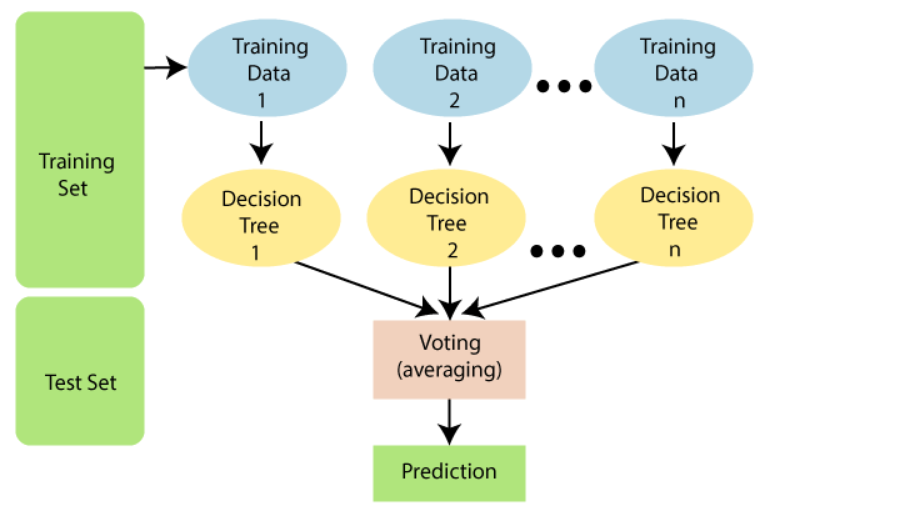
* There is less requirement of data cleaning compared to other algorithms.
* The decision tree contains lots of layers, which makes it complex.
* It may have an overfitting issue, which can be resolved using the **Random Forest algorithm.**
* For more class labels, the computational complexity of the decision tree may increase.

***Random Forest :***

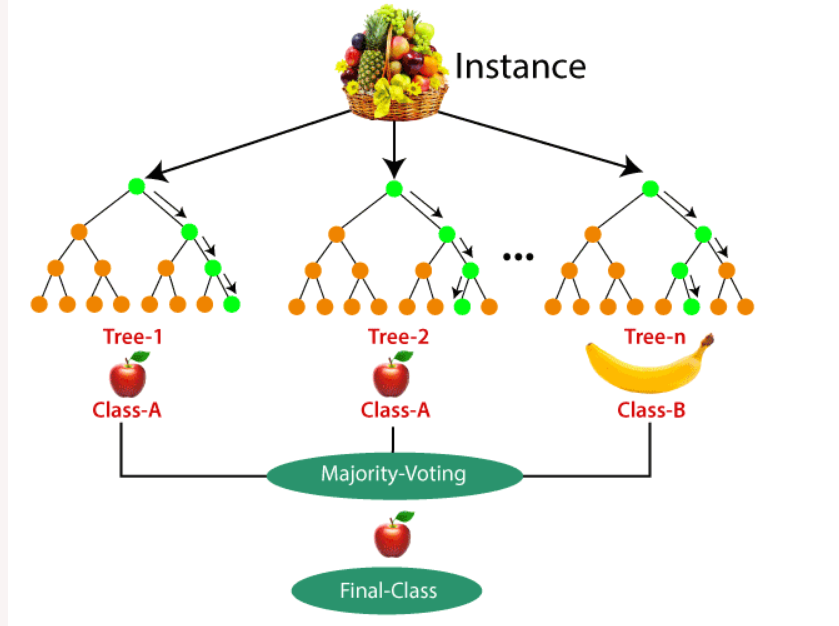
supervised learning technique, can be used for both Classification and Regression problems.

based on the concept of **ensemble learning,** which is a process of *combining multiple classifiers to solve a complex problem and to improve the performance of the model.*

**The greater number of trees in the forest leads to higher accuracy and prevents the problem of overfitting.**



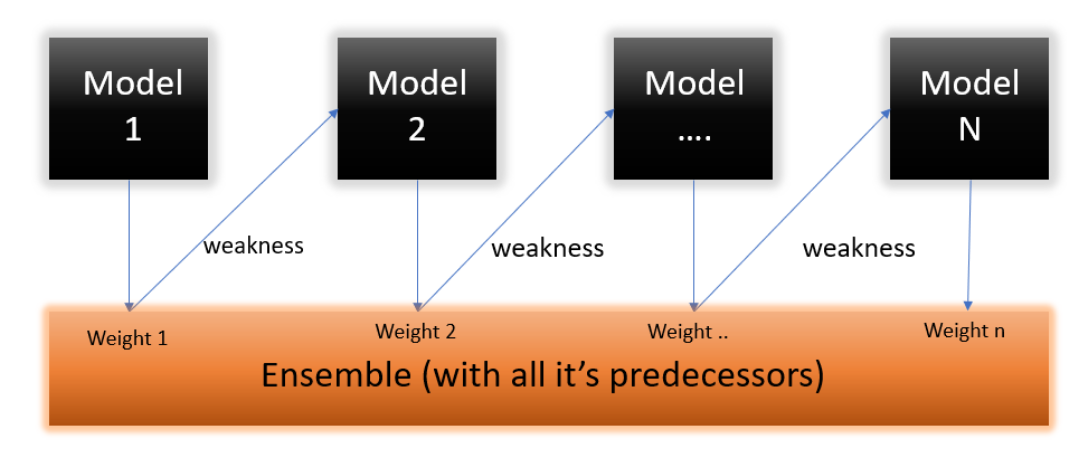
* It predicts output with high accuracy, even for the large dataset it runs efficiently.
* It can also maintain accuracy when a large proportion of data is missing.
* Are we sure about missing data? Missing data is okay
* Remove outlier? Too many Outlier we need to remove



* It is capable of handling large datasets with high dimensionality.
* It enhances the accuracy of the model and prevents the overfitting issue.
* Not suitable for Regression tasks.

***AdaBoostClassifier:***

Ensemble model for classification, the training data should be of high-quality, remove outlier from training data, Noisy data, specifically noise in the output variable can be problematic. If possible, attempt to isolate and clean these from your training dataset.



Basic idea is to create stumps of depth =1, and use the error of the 1st stump to reassign weight that will be used by the next stump and so on.

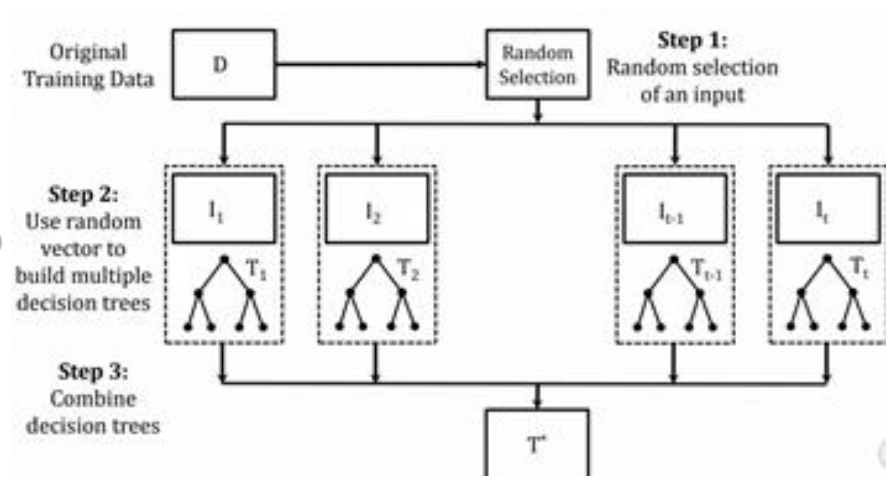
Link for reference: <https://www.analyticsvidhya.com/blog/2021/09/adaboost-algorithm-a-complete-guide-for-beginners/>

***ExtraTreesClassifier:***

Similar to random forest but faster in comparison which uses a number of decision tree at the back end and each decision tree is uncorrelated to each other. Main difference between Rf and ETC is the split is based randomly whereas for RF it is based on GINI or entropy.

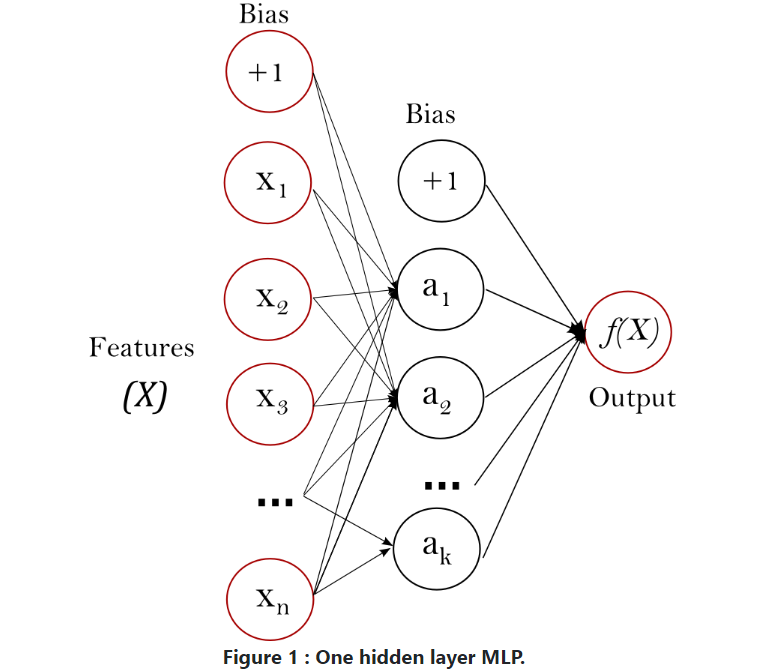
ETC also used all the records of the samples whereas RF takes subset of the samples. i.e each tree is constructed on all the records in the data but split is done randomly.

Use this if random forest doesn’t give good result



***MLPClassifier:***

Type of neural network for supervised learning, it learns a non-linear function unlike logistic regression and this method can be used for classification and regression.



Disadvantages:

* MLP requires tuning a number of hyperparameters such as the number of hidden neurons, layers, and iterations.
* MLP is sensitive to feature scaling. so, it is highly recommended to scale your data.
* Works well with large input data.
* Similar accuracy can be achieved with smaller data.
* This type of network are having more than 3 layers and its used to classify the data which is not linear
* These kinds of networks are fully connected with every node.
* These networks are extensively used for speech recognition and other machine learning technologies

Why use this compared to nn?

It is a type of nn so it is similar in behaviour.

https://www.ibm.com/in-en/cloud/learn/neural-networks#toc-types-of-n-YgdI1-Kt

#### References :

* Naive Bayes’ : <https://www.geeksforgeeks.org/naive-bayes-classifiers/>
* SGD : https://towardsdatascience.com/how-to-make-sgd-classifier-perform-as-well-as-logistic-regression-using-parfit-cc10bca2d3c4
* K-Nearest Neighbor : <https://www.geeksforgeeks.org/k-nearest-neighbours/?ref=lbp>
* Gradient Boosting : <https://towardsdatascience.com/gradient-boosting-classification-explained-through-python-60cc980eeb3d>
* Quadratic Discriminant Analysis : <https://www.geeksforgeeks.org/quadratic-discriminant-analysis/>
* RANSAC : https://towardsdatascience.com/random-sample-consensus-helps-you-filter-those-pesky-outliers-9dbfbb8b668f
* Documentation
  + Naive Bayes’ Classifier : <https://scikit-learn.org/stable/modules/naive_bayes.html>
  + SGD Classifier : <https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.SGDClassifier.html>
  + K-Nearest-Neighbor Algorithm: <https://scikit-learn.org/stable/modules/generated/sklearn.neighbors.KNeighborsClassifier.html>
  + Gradient Boosting : <https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.GradientBoostingClassifier.html>
  + Quadratic Discriminant Analysis : <https://scikit-learn.org/stable/modules/generated/sklearn.discriminant_analysis.QuadraticDiscriminantAnalysis.html>
  + Random Sample Consensus Regressor : https://scikit-learn.org/stable/modules/generated/sklearn.linear\_model.RANSACRegressor.html