IML Lab

▼ Simple Linear Regression

- Simple Linear Regression is a type of Regression algorithms that models the relationship between a dependent variable and a single independent variable.
- The relationship shown by a Simple Linear Regression model is linear or a sloped straight line, hence it is called Simple Linear Regression.
- The key point in Simple Linear Regression is that the ***dependent variable must be a continuous/real value***

▼ Implementation

▼ Step-1: Data Pre-processing

- First, we will import the three important libraries, which will help us for loading the dataset, plotting the graphs, and creating the Simple Linear Regression model.
- Next, we will load the dataset into our code:
- After that, we need to extract the dependent and independent variables from the given dataset. The independent variable is years of experience, and the dependent variable is salary.
- Next, we will split both variables into the test set and training set. We have 30 observations, so we will take 20 observations for the training set and 10 observations for the test set. We are splitting our dataset so that we can train our model using a training dataset and then test the model using a test dataset.

▼ Step-2: Fitting the Simple Linear Regression to the Training Set:

- Now the second step is to fit our model to the training dataset.
- To do so, we will import the **LinearRegression** class of the **linear_model** library from the **scikit learn**.
- After importing the class, we are going to create an object of the class named as a **regressor**.

▼ **Step: 3. Prediction of test set result:**

- In this step, we will provide the test dataset (new observations) to the model to check whether it can predict the correct output or not.
- We will create a prediction vector **y_pred**, and **x_pred**, which will contain predictions of test dataset, and prediction of training set respectively.

▼ **Step: 4. visualizing the Training and Testing set results:**

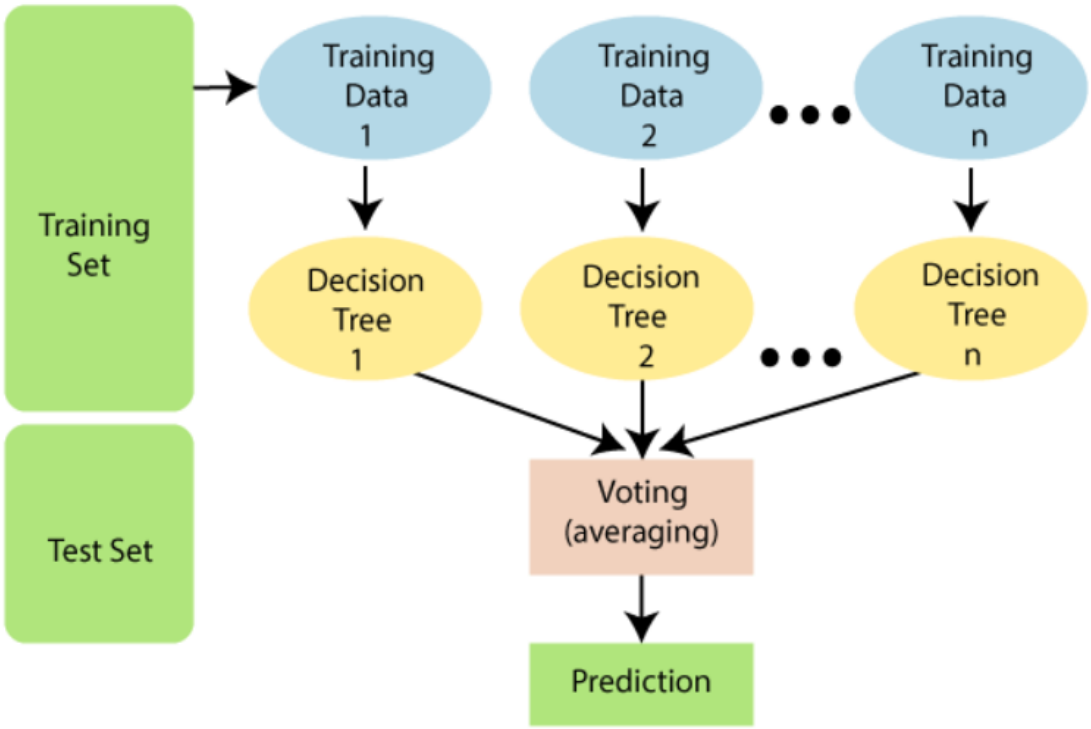
- Now in this step, we will visualize the training and testing set result to see the accuracy of the answers of our model.

▼ **Random Forest Algorithm**

- Random Forest is a classifier that contains a number of decision trees on various subsets of the given dataset and takes the average to improve the predictive accuracy of that dataset.
- The greater number of trees in the forest leads to higher accuracy and prevents the problem of overfitting.

▼ **Image**

The below diagram explains the working of the Random Forest algorithm:

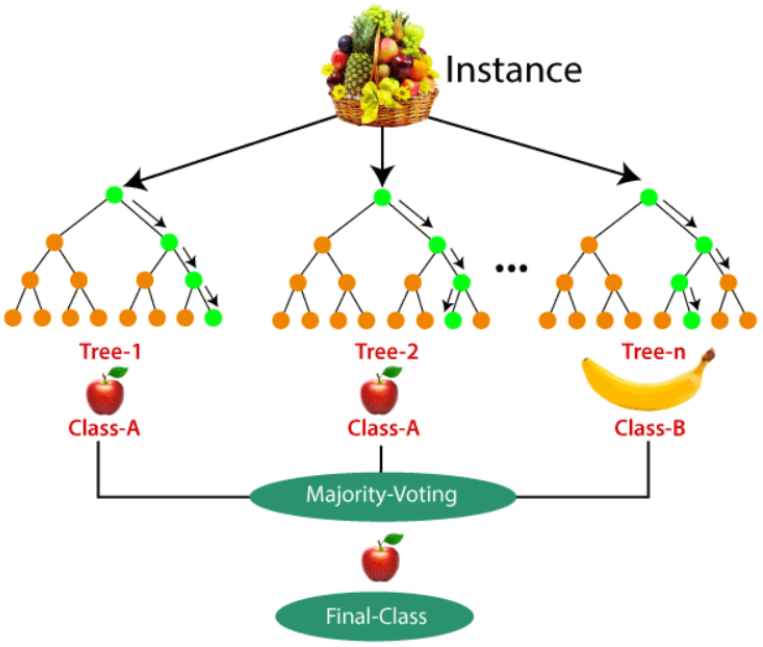


▼ **Implementation**

1. Select random K data points from the training set.
2. Build the decision trees associated with the selected data points (Subsets).
3. Choose the number N for decision trees that you want to build
4. Repeat Step 1 & 2.
5. For new data points, find the predictions of each decision tree, and assign the new data points to the category that wins the majority votes.

▼ **Example of the working**

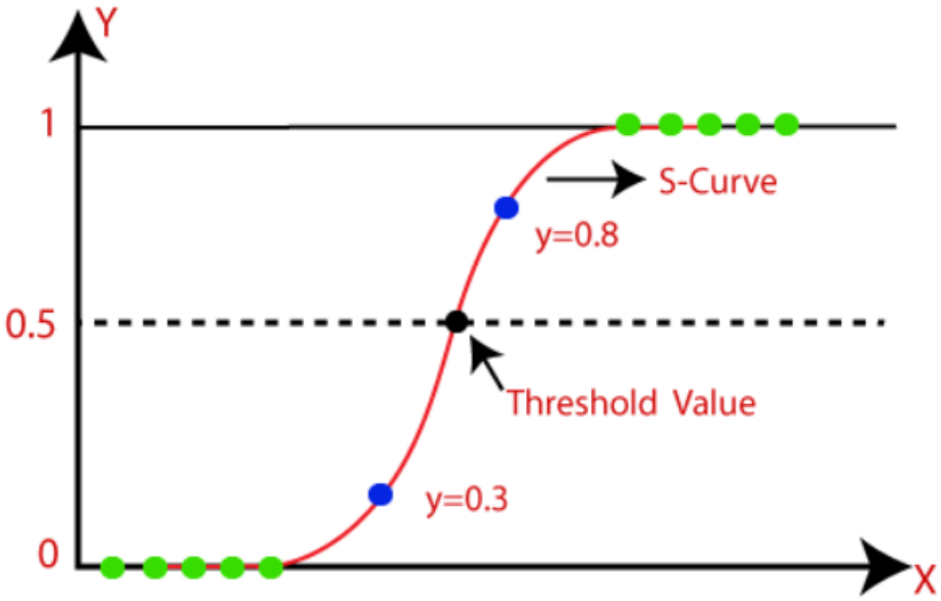
Example: Suppose there is a dataset that contains multiple fruit images. So, this dataset is given to the Random forest classifier. The dataset is divided into subsets and given to each decision tree. During the training phase, each decision tree produces a prediction result, and when a new data point occurs, then based on the majority of results, the Random Forest classifier predicts the final decision. Consider the below image:



▼ Logistic Regression

- Logistic regression predicts the output of a categorical dependent variable. Therefore the outcome must be a categorical or discrete value. It can be either Yes or No, 0 or 1, true or False, etc. but instead of giving the exact value as 0 and 1, **it gives the probabilistic values which lie between 0 and 1**.
- In Logistic regression, instead of fitting a regression line, we fit an "S" shaped logistic function, which predicts two maximum values (0 or 1).

▼ Image



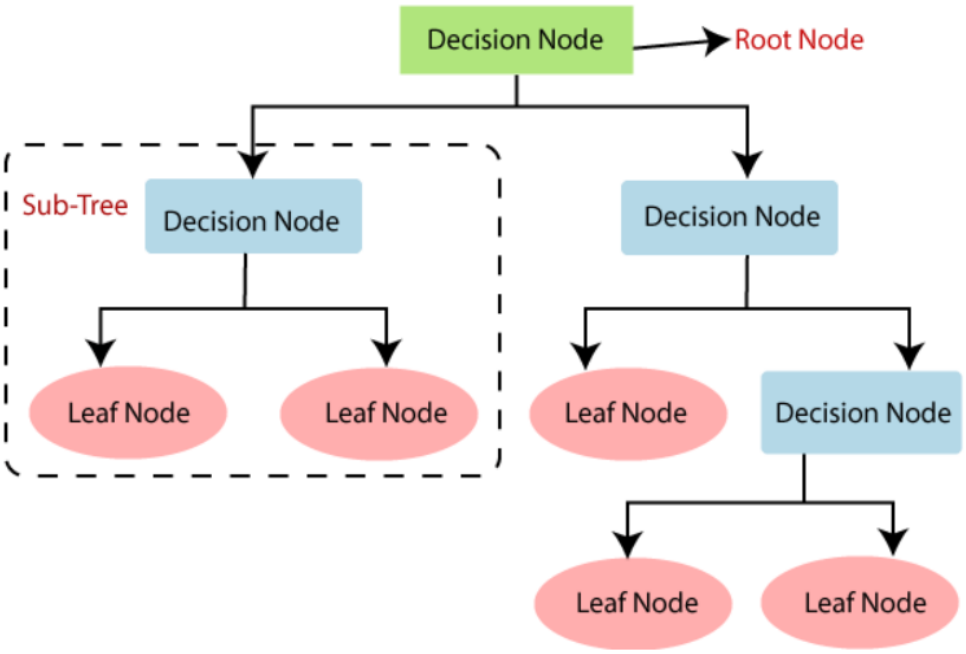
▼ **Implementation**

1. Data Pre-processing step
2. Fitting Logistic Regression to the Training set
3. Predicting the test result
4. Test accuracy of the result(Creation of Confusion matrix)
5. Visualizing the test set result.

▼ **Decision Tree Classification Algorithm**

- In a Decision tree, there are two nodes, which are the **Decision Node** and **Leaf Node.** Decision nodes are used to make any decision and have multiple branches, whereas Leaf nodes are the output of those decisions and do not contain any further branches.
- It is called a decision tree because, similar to a tree, it starts with the root node, which expands on further branches and constructs a tree-like structure.

▼ **Image**



▼ **Decision Tree Terminologies**

- **Root Node:** Root node is from where the decision tree starts. It represents the entire dataset, which further gets divided into two or more homogeneous sets.
- **Leaf Node:** Leaf nodes are the final output node, and the tree cannot be segregated further after getting a leaf node.
- **Splitting:** Splitting is the process of dividing the decision node/root node into sub-nodes according to the given conditions.
- **Branch/Sub Tree:** A tree formed by splitting the tree.
- **Pruning:** Pruning is the process of removing the unwanted branches from the tree.
- **Parent/Child node:** The root node of the tree is called the parent node, and other nodes are called the child nodes.

▼ **Steps for working of the algorithm**

1. Begin the tree with the root node, says S, which contains the complete dataset.

2. Find the best attribute in the dataset using **Attribute Selection Measure (ASM).**
3. Divide the S into subsets that contains possible values for the best attributes.
4. Generate the decision tree node, which contains the best attribute.
5. Recursively make new decision trees using the subsets of the dataset created in step -3. Continue this process until a stage is reached where you cannot further classify the nodes and called the final node as a leaf node.

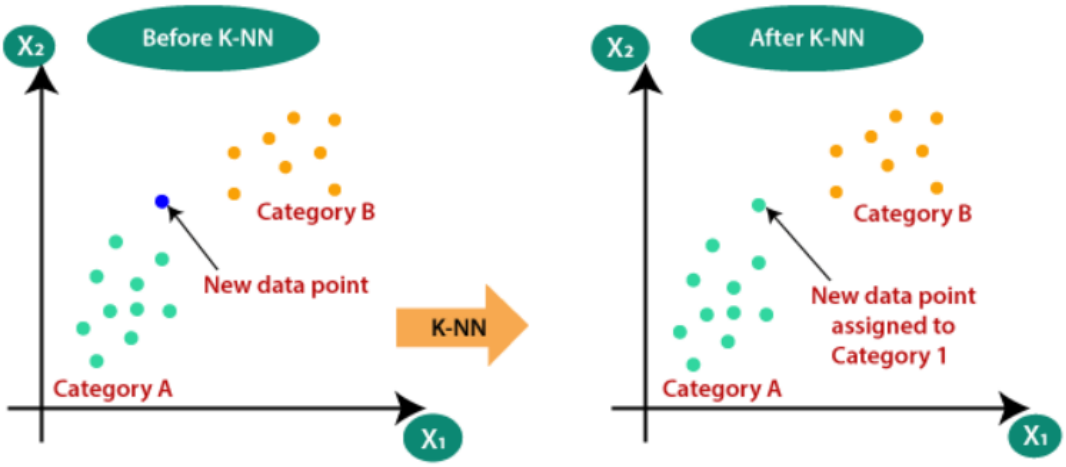
▼ **Implementation**

1. **Data Pre-processing step**
2. **Fitting a Decision-Tree algorithm to the Training set**
3. **Predicting the test result**
4. **Test accuracy of the result(Creation of Confusion matrix)**
5. **Visualizing the test set result.**

▼ **K-Nearest Neighbor(KNN) Algorithm**

- K-NN algorithm assumes the similarity between the new case/data and available cases and put the new case into the category that is most similar to the available categories.
- K-NN algorithm stores all the available data and classifies a new data point based on the similarity. This means when new data appears then it can be easily classified into a well suite category by using K- NN algorithm.

▼ **Image**



▼ **How does algo work**

1. Select the number K of the neighbors
2. Calculate the Euclidean distance of **K number of neighbors**
3. Take the K nearest neighbors as per the calculated Euclidean distance.
4. Among these k neighbors, count the number of the data points in each category.\
5. Assign the new data points to that category for which the number of the neighbor is maximum.
6. Our model is ready.

▼ **Implementation**

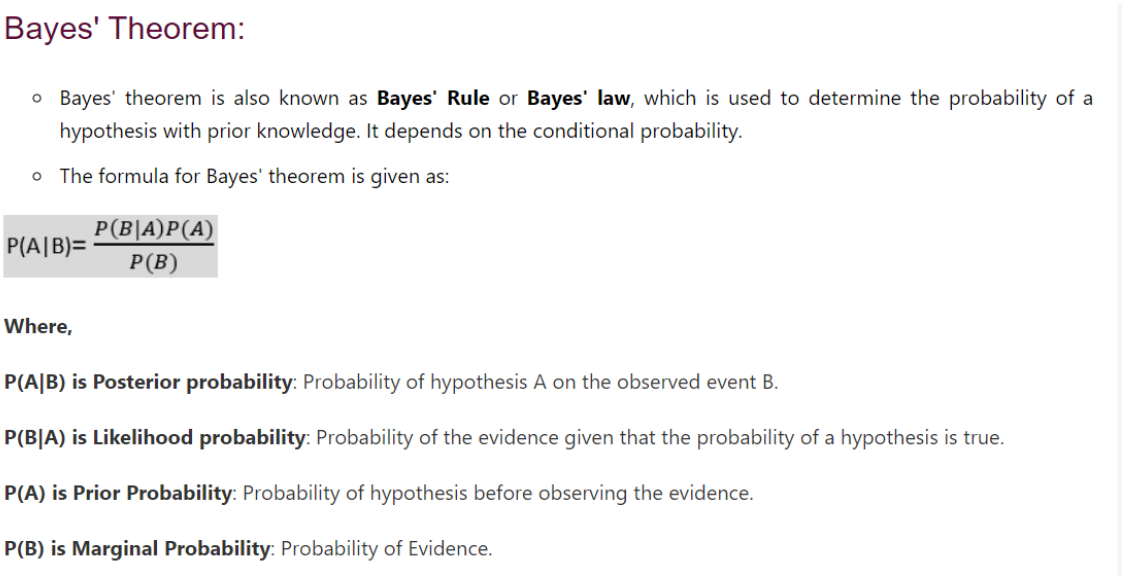
1. **Data Pre-processing step**
2. **Fitting a Decision-Tree algorithm to the Training set**
3. **Predicting the test result**
4. **Test accuracy of the result(Creation of Confusion matrix)**
5. **Visualizing the test set result.**

▼ **Naïve Bayes Classifier Algorithm**

- Naïve Bayes algorithm is a supervised learning algorithm, which is based on **Bayes theorem** and used for solving classification problems.

- It is a probabilistic classifier, which means it predicts on the basis of the probability of an object.

▼ **Bayes Theorem**



Bayes' Theorem:

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

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

Where,

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

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

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

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

▼ **Working**

1. Convert the given dataset into frequency tables.
2. Generate Likelihood table by finding the probabilities of given features.
3. Now, use Bayes theorem to calculate the posterior probability.

▼ **Implementation**

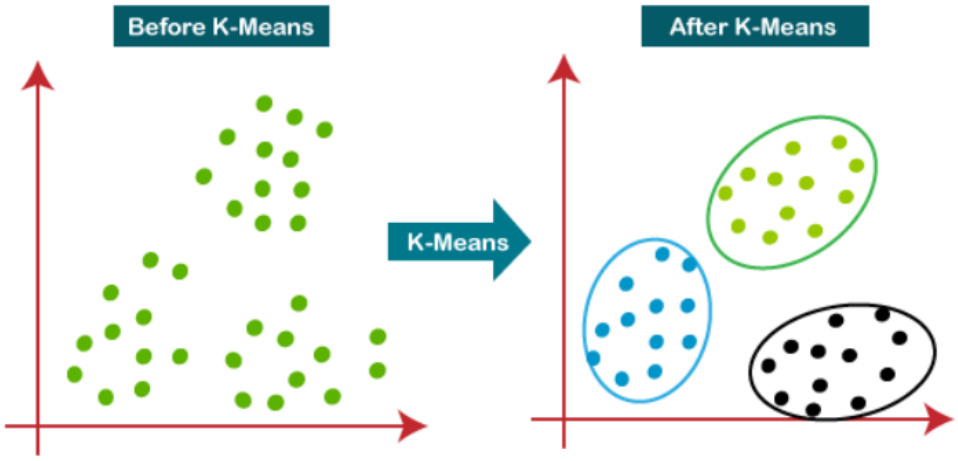
1. **Data Pre-processing step**
2. **Fitting a Decision-Tree algorithm to the Training set**
3. **Predicting the test result**
4. **Test accuracy of the result(Creation of Confusion matrix)**
5. **Visualizing the test set result.**

▼ **K-Means Clustering Algorithm**

- It allows us to cluster the data into different groups and a convenient way to discover the categories of groups in the unlabeled dataset on its own without the need for any training.

- It is a centroid-based algorithm, where each cluster is associated with a centroid. The main aim of this algorithm is to minimize the sum of distances between the data point and their corresponding clusters.

▼ **Image**



▼ **Working**

1. Select the number K to decide the number of clusters.
2. Select random K points or centroids. (It can be other from the input dataset).
3. Assign each data point to their closest centroid, which will form the predefined K clusters.
4. Calculate the variance and place a new centroid of each cluster.
5. Repeat the third steps, which means reassign each datapoint to the new closest centroid of each cluster.
6. If any reassignment occurs, then go to step-4 else go to FINISH.
7. The model is ready.

▼ **Implementation**

1. **Data Pre-processing step**
2. **Fitting a Decision-Tree algorithm to the Training set**

3. **Predicting the test result**
4. **Test accuracy of the result(Creation of Confusion matrix)**
5. **Visualizing the test set result.**

▼ **How to Compare Machine Learning Models and Algorithms**

▼ **Loss Functions and Metrics for Regression:**

- **Mean Square Error:** measures the average of the squares of the errors or deviations, that is, the difference between the estimated and true value. It aids in imposing higher weights on outliers, thus reducing the issue of overfitting.
- **Mean Absolute Error:** It's the absolute difference between the estimated value and true value. It decreases the weight for outlier errors when compared to the mean squared error.
- **Smooth Absolute Error:** It's the absolute difference between the estimated value and true value for the predictions lying close to the real value, and it's the square of the difference between the estimated and the true values of the outliers (or points far off from predicted values). Essentially, it's a combination of MSE and MAE.