**Decision Tree**

A decision tree is a hierarchical model for making decisions or predictions. It is a graphic representation of a number of options and their results. The tree is made up of nodes, branches, and leaves. When a decision must be made in line with specific requirements, which are often expressed as questions, the nodes act as decision points. The branches show the possible consequences of the decision, while the leaves represent the final results or projections. Which branch should be taken depends on a test condition at each internal node. The test condition is typically expressed as a simple logical condition, such as "Is the temperature higher than 70 degrees?" The answer to this question indicates which branch of the tree. As we move down the tree, the number of branches grows, and the inquiries become more specific and intricate. The end decision or forecast is represented by a leaf node, which is where we eventually arrive. Decision trees can be used to address the challenges of grouping, regression, and classification. Because they are easy to understand and interpret and can be utilized with categorical and continuous data, they are frequently employed in data mining and machine learning.

**Random Forest**

A well-liked ensemble learning technique in machine learning is random forest. It integrates different decision trees to create a prediction that is more precise and stable. At training time, the random forest algorithm creates a large number of decision trees. The following are the results of a survey conducted by the National Center for Education Statistics (NCES) on the use of the e-learning platform. This action is referred to as bagging (bootstrap aggregating). The random forest algorithm collects all of the individual trees' forecasts during prediction and then outputs the prediction that is most frequently made.

When opposed to a single decision tree, the key benefit of a random forest is that it lessens overfitting. The random forest can better generalize to unknown data since it uses several trees to capture the intricate relationships in the data. For both classification and regression problems, the random forest can be used. The class with the highest votes from the individual trees is what the random forest produces when classifying data. The result of regression is the average of the predicted values obtained from each individual tree. Several real-world applications, such as those in banking, medicine, and image identification, make extensive use of random forests. For supervised learning tasks, it is regarded as one of the most potent machine-learning algorithms

**KNN**

A non-parametric machine learning approach called KNN (k-Nearest Neighbors) is employed for both classification and regression problems. It is based on the premise that similar items are close to each other in a high-dimensional space. The KNN algorithm first trains on a dataset with labels, where each sample has a corresponding label. The algorithm determines the separation between a new sample and each sample in the training dataset while making a prediction. After choosing the k closest samples based on the distance measure, the algorithm predicts the label for the new sample using the label of the majority of the k nearest neighbors. The Manhattan distance or Euclidean distance are two distance metrics that the KNN technique can use to determine the separation between two samples. Cross-validation or other optimization techniques are frequently used to choose the value of k, or the number of neighbors to take into account. One of the main advantages of KNN is how simple it is to use and interpret. It works with both categorical and continuous data and makes no assumptions about how the underlying data are distributed. The dimensionality curse may occur when there are many features since KNN may be computationally expensive for huge datasets. In real-world settings, KNN is extensively used in picture identification, natural language processing, and recommendation systems.

**Naïve Bayes**

The Bayes theorem, which states that the likelihood of a hypothesis (such as a class label) given evidence (such as input features), is proportional to the likelihood of the evidence given the hypothesis and the prior probability of the hypothesis, is the foundation of the probabilistic machine learning algorithm known as Naive Bayes. The "naive" assumption behind Naive Bayes is that, given the class label, the characteristics are conditionally independent. This indicates that the algorithm makes the assumption that the existence or absence of one characteristic has no bearing on the likelihood of existence or absence of an other feature. The Naive Bayes method uses the Bayes theorem to determine the posterior probability of each class label given the input data during prediction. The algorithm then chooses the class label that has the highest posterior probability as the input sample prediction. Naive Bayes is an efficient and straightforward technique that can deal with missing values and high-dimensional data. It may be applied to text classification problems in natural language processing, binary classification problems, and multiclass classification problems. The assumption of conditional independence, which could not hold in some circumstances, is one of the fundamental drawbacks of the Naive Bayes model. Nonetheless, in many real-world situations, it has been demonstrated that Naive Bayes works effectively.

**LSTM**

Long Short-Term Memory (LSTM) is a form of recurrent neural network (RNN) architecture that is frequently applied to sequence-to-sequence learning applications, including voice and natural language processing. The primary distinction between LSTM and conventional RNNs is that LSTM contains a memory cell with long-term storage capabilities, enabling it to recognize long-term dependencies in the input sequence. Three gates—the input gate, the forget gate, and the output gate—control the memory cell. A series of inputs are sent to the LSTM network during training, and the input gate chooses the data to add to the memory cell. The forget gate determines which information to remove from the memory cell, and the output gate determines which information to output from the memory cell to the next layer in the network. In order to build deeper architectures, LSTM can be stacked, which enables it to recognize more intricate patterns in input sequences. It can also be bidirectional, meaning that it analyzes the input sequence in both forward and backward directions, allowing it to capture dependencies in both ways. Numerous real-world applications, including sentiment analysis, speech recognition, and language translation, have demonstrated the effectiveness of LSTM. Nevertheless, it may be computationally costly and requires a big quantity of training data to learn successfully.