MAHARAJA'S TECHNOLOGICAL INSTITUTE,
GOVERNMENT POLYTECHNIC COLLEGE
THRISSUR - 680 020

RANDOM FOREST ALGORITHM

Submitted By:

ANANDU M M
Register No: 2101130452

Department of Computer Engineering 2023-2024

MAHARAJA'S TECHNOLOGICAL INSTITUTE,
GOVERNMENT POLYTECHNIC COLLEGE
THRISSUR - 680 020

Seminar report on

RANDOM FOREST ALGORITHM

Submitted By:

ANANDU M M
Register No: 2101130452

Department of Computer Engineering 2023-2024

MAHARAJA'S TECHNOLOGICAL INSTITUTE,
GOVERNMENT POLYTECHNIC COLLEGE
THRISSUR - 680 020

COMPUTER ENGINEERING

CERTIFICATE

This is to certify that the seminar entitled "RANDOM FOREST ALGORITHM" is submitted by ANANDU M M bearing Reg No. 2101130452 in partial fulfilment of the requirement for the award of the Diploma in Computer Engineering of State Board of Technical Examinations, Kerala for the academic year 2023-2024.

LECT. IN CHARGE HEAD OF THE DEPARTMENT

EXTERNAL EXAMINER INTERNAL EXAMINER

ACKNOWLEDGEMENT

First of all, I am indebted to the **GOD ALMIGHTY** for giving me an opportunity to excel in my efforts to complete this seminar on time.

I am extremely grateful to Mrs. **THAJBI P M**, Head of Department, Department of Computer Engineering, for providing all the required resources for the successful completion of my seminar.

My heartfelt gratitude to my seminar guide Mrs.THAJBI P.M,Head Of Department & Mr.Pradeep Chandran Lecturer, Department of Computer Engineering, for these valuable suggestions and guidance in the preparation of the seminar report.

I will be failing in duty if I do not acknowledge with grateful thanks the authors of the references and other literature referred to in this seminar.

Last but not the least, I am very much thankful to my parents who guides me in every steps that I took

ABSTRACT

Random Forest is a popular machine learning algorithm that belongs to the supervised learning technique. It can be used for both Classification and Regression problems in ML. We know that a forest comprises numerous trees, and the more trees more it will be robust. Similarly, the greater the number of trees in a Random Forest Algorithm, the higher its accuracy and problem-solving ability. Random Forest is a classifier that contains several decision trees on various subsets of the given dataset and takes the average to improve the predictive accuracy of that dataset. It is 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.

INDEX

1. INDRODUCTION

A Random Forest Algorithm is a supervised machine learning algorithm that is extremely popular and is used for Classification and Regression problems in Machine Learning. We know that a forest comprises numerous trees, and the more trees more it will be robust. Similarly, the greater the number of trees in a Random Forest Algorithm, the higher its accuracy and problem-solving ability. Random Forest is a classifier that contains several decision trees on various subsets of the given dataset and takes the average to improve the predictive accuracy of that dataset. It is based on the concept of ensemble learning which is a process of combining multiple classifiers to solve a complex problem and improve the performance of the model.

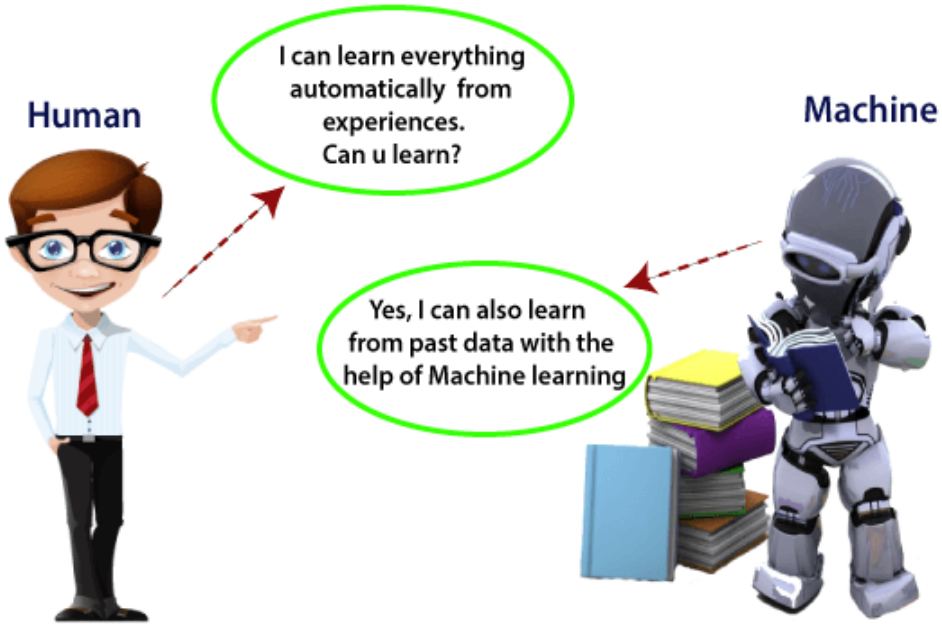
Since the random forest combines multiple trees to predict the class of the dataset, it is possible that some decision trees may predict the correct output, while others may not. But together, all the trees predict the correct output. Therefore, for a better Random forest there should be some actual values in the feature variable of the dataset so that the classifier can predict accurate results rather than a guessed result. The predictions from each tree must have very low correlations.

Hence it takes less training time as compared to other algorithms. And 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. All of these made random forest algorithm a better machine learning algorithm.

2. MACHINE LEARNING

Machine learning is a branch of artificial intelligence (AI) and computer science which focuses on the use of data and algorithms to imitate the way that humans learn, gradually improving its accuracy. Machine learning uses various algorithms for building mathematical models and making predictions using historical data or information. Currently, it is being used for various tasks such as image recognition, speech recognition, email filtering, Facebook auto-tagging, recommender system, and many more.

In the real world, we are surrounded by humans who can learn everything from their experiences with their learning capability, and we have computers or machines which work on our instructions. But can a machine also learn from experiences or past data like a human does? So here comes the role of Machine Learning.



"Machine learning enables a machine to automatically learn from data, improve performance from experiences, and predict things without being explicitly programmed"

Machine Learning is said as a subset of artificial intelligence that is mainly concerned with the development of algorithms which allow a computer to learn from the data and past experiences on their own. The term machine learning was first introduced by Arthur Samuel in 1959. We can define it in a summarized way as:

With the help of sample historical data, which is known as training data, machine learning algorithms build a mathematical model that helps in making predictions or decisions without being explicitly programmed. Machine learning brings computer science and statistics together for creating predictive models. Machine learning constructs or uses the algorithms that learn from historical data. The more we will provide the information, the higher will be the performance.

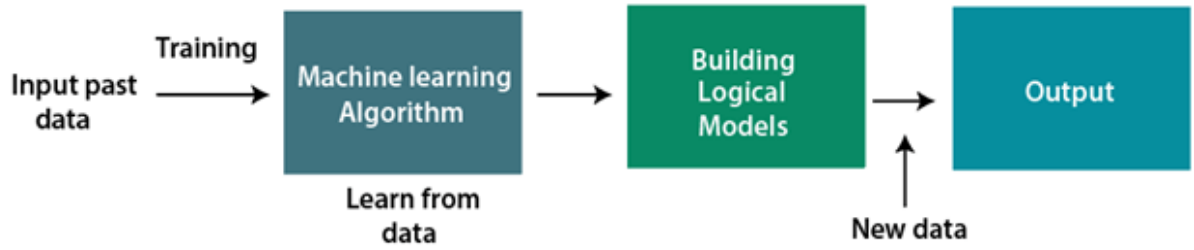
A machine has the ability to learn if it can improve its performance by gaining more data.

2.1 How does Machine Learning work

A Machine Learning system learns from historical data, builds the prediction models, and whenever it receives new data, predicts the output for it. The accuracy of predicted output depends upon the amount of data, as the huge amount of data helps to build a better model which predicts the output more accurately.

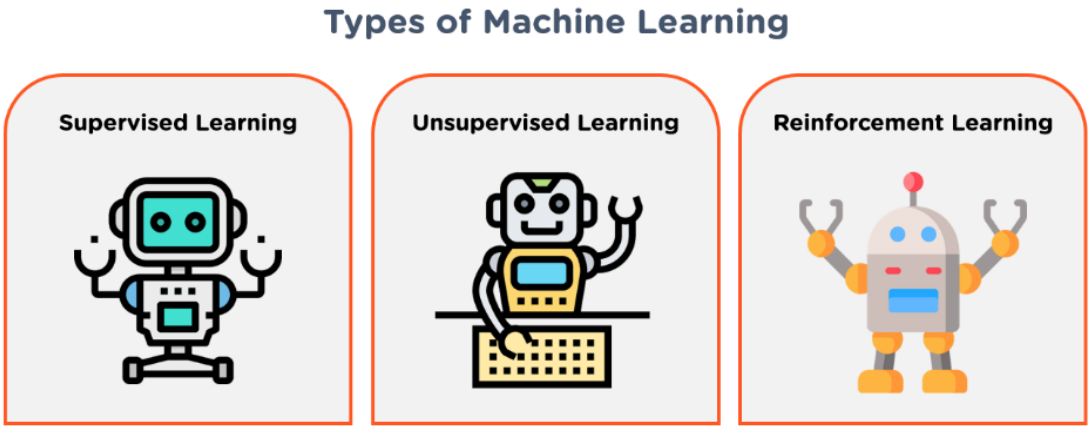
Suppose we have a complex problem, where we need to perform some predictions, so instead of writing a code for it, we just need to feed the data to generic algorithms, and with the help of these algorithms,

machine builds the logic as per the data and predict the output. Machine learning has changed our way of thinking about the problem. The below block diagram explains the working of Machine Learning algorithm:



2.2 Classification of Machine Learning

Machine learning can be classified into three as follows:



2.2.1 Supervised Learning

Supervised learning is a type of machine learning that uses labeled data to train machine learning models. In labeled data, the output is already known. The model just needs to map the inputs to the respective outputs.

It can be of two types:

- Classification
- Regression

Algorithms:

Some of the most popularly used supervised learning algorithms are:

- Linear Regression
- Logistic Regression
- Support Vector Machine
- K Nearest Neighbor
- Decision Tree
- Random Forest
- Naive Bayes

2.2.2 Unsupervised Learning

Unsupervised learning is a type of machine learning that uses unlabeled data to train machines. Unlabeled data doesn't have a fixed output variable. The model learns from the data, discovers the patterns and features in the data, and returns the output.

It can be of two types:

- Clustering
- Association

Algorithms:

Selecting the right algorithm depends on the type of problem you are trying to solve. Some of the common examples of unsupervised learning are:

- K Means Clusterin
- Hierarchical Clustering
- DBSCAN
- Principal Component Analysis

2.2.3 Reinforcement Learning

Reinforcement Learning trains a machine to take suitable actions and maximize its rewards in a particular situation. It uses an agent and an environment to produce actions and rewards. The agent has a start and an end state. But, there might be different paths for reaching the end state, like a maze. In this learning technique, there is no predefined target variable.

Algorithms:

Some of the important reinforcement learning algorithms are:

- Q-learning
- Sarsa
- Monte Carlo
- Deep Q network

3. BACKGROUND

It is 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. It is really similar to decision tree which is a popular machine learning algorithm used for both classification and regression tasks.

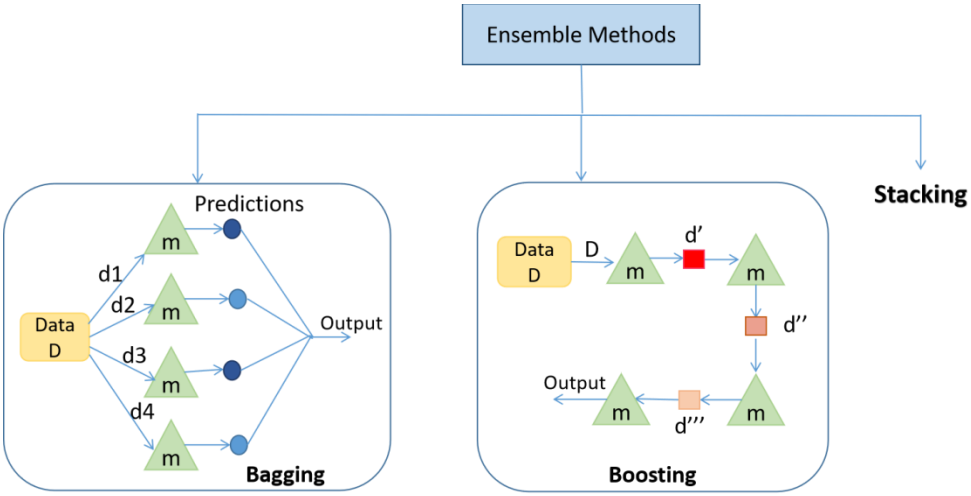
3.1 Ensemble Learning

Ensemble learning is a machine learning technique that combines the predictions of multiple individual models (often called base or weak models) to improve overall predictive performance. The idea behind ensemble learning is that by aggregating the predictions of several models, the weaknesses and errors of individual models can be mitigated, leading to more accurate and robust predictions. Ensemble methods are widely used in various machine learning applications and are known for their effectiveness.

3.1.1 What is ensemble learning?

The ensemble methods in machine learning combine the insights obtained from multiple learning models to facilitate accurate and improved decisions. In learning models, noise, variance, and bias are the major sources of error. The ensemble methods in machine learning help minimize these error-causing factors, thereby ensuring the accuracy and stability of machine learning (ML) algorithms.

The three main classes of ensemble learning methods are bagging, stacking, and boosting, and it is important to both have a detailed understanding of each method and to consider them on your predictive modeling project.



1. Bagging (Bootstrap Aggregating):

- Bagging aims to reduce variance and prevent overfitting. It involves training multiple instances of the same base model on different subsets of the training data, typically by random sampling with replacement (bootstrap).
- The final prediction is made by averaging (for regression) or voting (for classification) the predictions of individual models.

2. Boosting:

- Boosting is an iterative ensemble method that focuses on improving the performance of weak learners over time.

- It assigns different weights to training instances and emphasizes the misclassified ones in subsequent iterations.
- AdaBoost (Adaptive Boosting) and Gradient Boosting are popular boosting algorithms.

3. Stacking (Stacked Generalization):

- Stacking combines predictions from multiple base models using another model, often called a meta-learner or blender.
- The base models' predictions serve as input features for the meta-learner, which then makes the final prediction.
- Stacking can be more complex to implement but often results in improved performance.

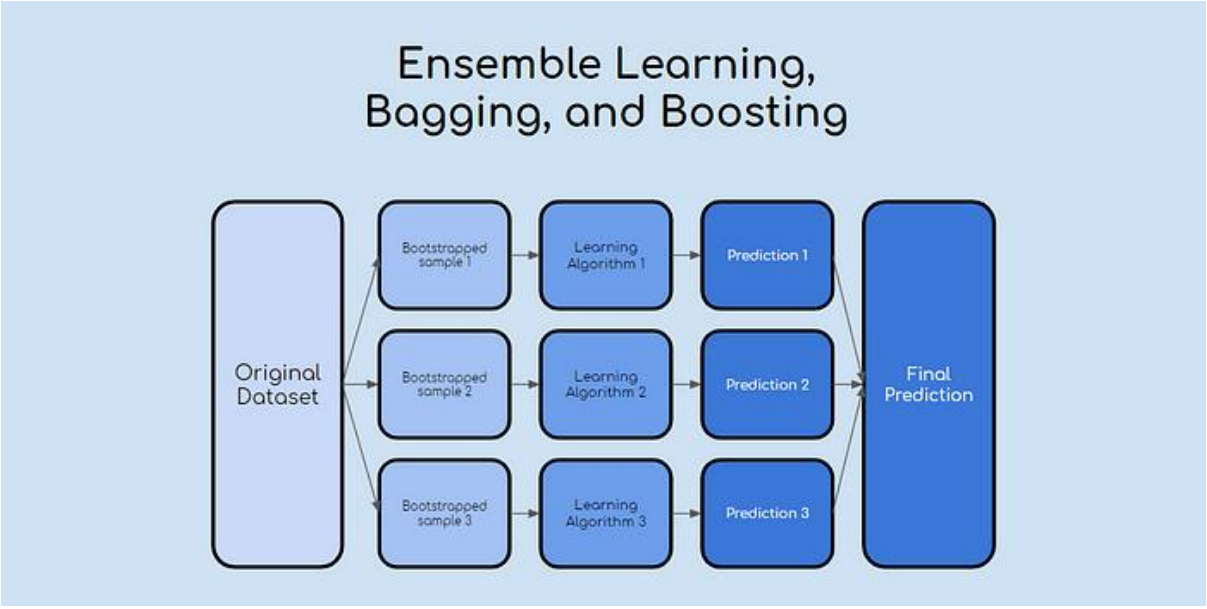
3.1.2 Case study

Assume that you are developing an app for the travel industry. It is obvious that before making the app public, you will want to get crucial feedback on bugs and potential loopholes that are affecting the user experience. What are your available options for obtaining critical feedback? 1) Soliciting opinions from your parents, spouse, or close friends. 2) Asking your co-workers who travel regularly and then evaluating their response. 3) Rolling out your travel and tourism app in beta to gather feedback from non-biased audiences and the travel community.

Think for a moment about what you are doing. You are taking into account different views and ideas from a wide range of people to fix issues that are limiting the user experience. The ensemble neural network and ensemble algorithm do precisely the same thing.

Ensemble learning is valuable in improving model generalization, reducing bias and variance, and achieving better predictive accuracy. The choice of ensemble method depends on the specific problem, data, and the strengths and weaknesses of individual base models. Ensemble techniques are widely used in various domains, including finance, healthcare, natural language processing, and computer vision.

3.2.3 Working



Ensemble learning is a machine learning technique that combines the predictions of multiple models (base models or weak learners) to create a more accurate and robust final prediction. Here's a concise overview of how ensemble learning works:

1. **Diverse Models:** Ensemble learning starts by selecting diverse base models, often of different types or trained on different subsets of data. Diversity among base models is key to capturing different patterns in the data.

2. **Training:** Each base model is trained independently on a portion of the dataset or with specific modifications to encourage diversity. The training process aims to minimize errors or maximize predictive accuracy for each model.

3. **Predictions:** Once trained, the base models are used to make predictions on new or validation data. In classification, each model assigns class labels; in regression, they predict numerical values.

4. **Combining Predictions:** Ensemble methods combine the individual predictions to produce a final prediction. This can be done through techniques like majority voting (classification) or averaging (regression). Some methods use weighted averages or more complex rules.

5. Weighting and Meta-Learning (Optional): In some ensembles like boosting, base models' predictions are weighted based on their performance. A meta-learner or meta-model may be used to determine optimal weights.

6. **Final Prediction:** The ensemble produces a final prediction based on the aggregated results from the base models or the meta-learner. The prediction is often more accurate and robust than any individual model's prediction.

7. **Evaluation:** The ensemble's performance is evaluated using validation data or cross-validation to ensure it generalizes well. It helps determine if the ensemble effectively combines the strengths of its base models.

8. **Deployment:** If the ensemble meets performance criteria, it can be deployed for making predictions on new, unseen data, where it benefits from the collective knowledge of the base models.

Ensemble learning enhances predictive accuracy, reduces overfitting, and improves model robustness by combining the insights of multiple models. Common ensemble methods include **Random Forests**, Gradient Boosting, AdaBoost, and various forms of stacking. The choice of ensemble method depends on the problem and data characteristics.

4. RANDOM FOREST ALGORITHM

As the name suggests, "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." Instead of relying on one decision tree, the random forest takes the prediction from each tree and based on the majority votes of predictions, and it predicts the final output. The greater number of trees in the forest leads to higher accuracy and prevents the problem of overfitting. Given below is a sample diagram of Random forest algorithm.

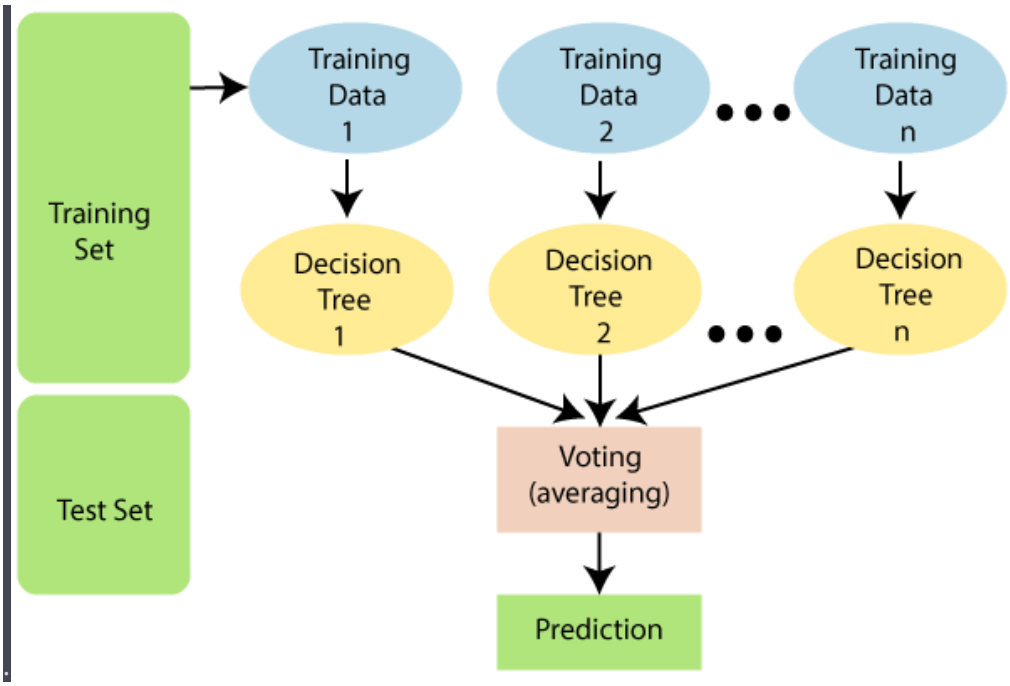
Since the random forest combines multiple trees to predict the class of the dataset, it is possible that some decision trees may predict the correct output, while others may not. But together, all the trees predict the correct output. Therefore, below are two assumptions for a better Random forest classifier:

- There should be some actual values in the feature variable of the dataset so that the classifier can predict accurate results rather than a guessed result.
- The predictions from each tree must have very low correlations.

4.1 Why Random Forest ?

- It takes less training time as compared to other algorithms.
- 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.
- Random Forest can be applied to both classification and regression tasks, making it versatile for a wide range of problems. It can handle categorical and numerical features without requiring extensive preprocessing.
- Reduces Overfitting: Random Forest reduces overfitting, a common issue in machine learning, by averaging predictions from multiple decision trees. Each tree is trained on a random subset of data and features, which helps to reduce model variance.
- Parallelizable: Random Forest is highly parallelizable, meaning it can take advantage of multi-core processors and distributed computing, making it efficient for large-scale datasets.
- Handles Imbalanced Data: It can handle imbalanced datasets, a common scenario in real-world applications, by adjusting class weights during training.

4.2 Working of Random Forest Algorithm



The following steps explain the working Random Forest Algorithm:

Step 1: Select random samples from a given data or training set.

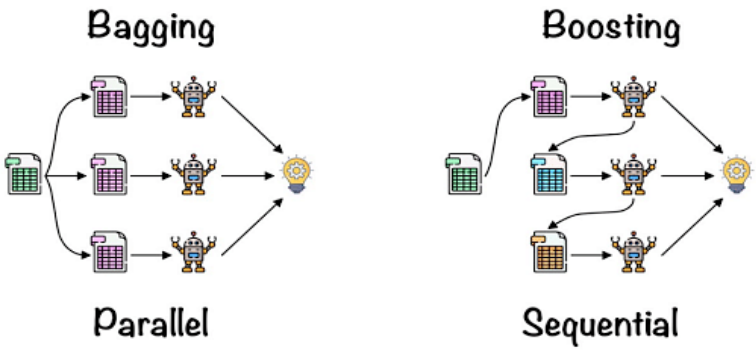
Step 2: This algorithm will construct a decision tree for every training data.

Step 3: Voting will take place by averaging the decision tree.

Step 4: Finally, select the most voted prediction result as the final prediction result.

This combination of multiple models is called Ensemble. Ensemble uses two methods:

1) **Bagging** 2) **Boosting**



4.2.1 Bagging

Bagging is an ensemble technique that involves training multiple base models independently on different subsets of the training data. These subsets are created through bootstrapping, which means random sampling with replacement from the original training dataset.

How it works:

- Random subsets of data are created with replacement (so some data points may be repeated, and others may be omitted).
- Each base model is trained on one of these subsets.
- The predictions from all the base models are then combined, typically by averaging (for regression problems) or using a majority vote (for classification problems).

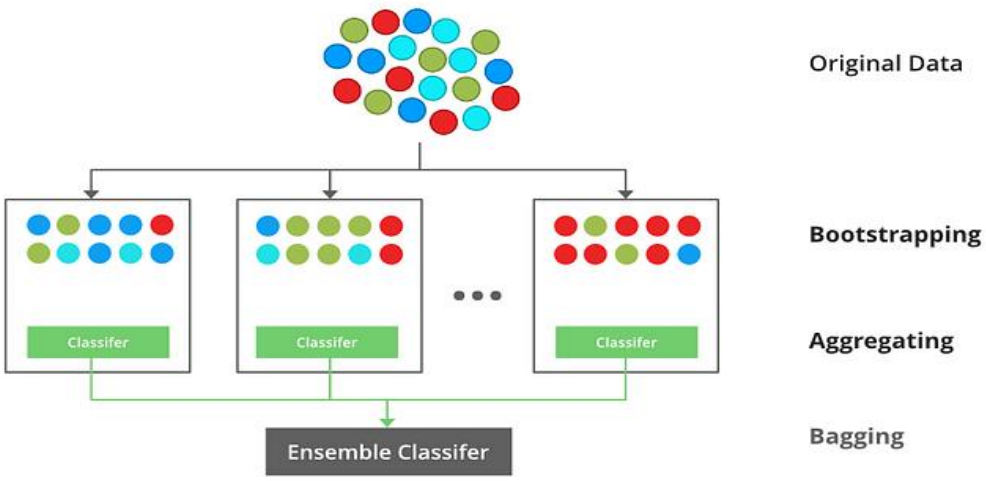
4.2.2 Boosting

Boosting is an ensemble technique that aims to improve the accuracy of a model by giving more weight to the samples that are difficult to classify. It builds base models sequentially, where each subsequent model focuses on correcting the mistakes made by the previous ones.

How it works:

- Initially, all data points are assigned equal weights.
- A base model is trained on the training data, and its predictions are evaluated.
- Weights are adjusted for misclassified data points, making the misclassified points more important for the next base model.
- The process is repeated, with each new model focusing on the previously misclassified data points.
- The final ensemble is a weighted combination of these base models.

Bagging is also known as Bootstrap Aggregation used by random forest. The process begins with any original random data. After arranging, it is organised into samples known as Bootstrap Sample. This process is known as Bootstrapping. Further, the models are trained individually, yielding different results known as Aggregation. In the last step, all the results are combined, and the generated output is based on majority voting. This step is known as Bagging and is done using an Ensemble Classifier.



4.3 Important Terms to Know

There are different ways that the Random Forest algorithm makes data decisions, and consequently, there are some important related terms to know. Some of these terms include:

Entropy :- It is a measure of randomness or unpredictability in the data set.

Information Gain :- A measure of the decrease in the entropy after the data set is split is the information gain.

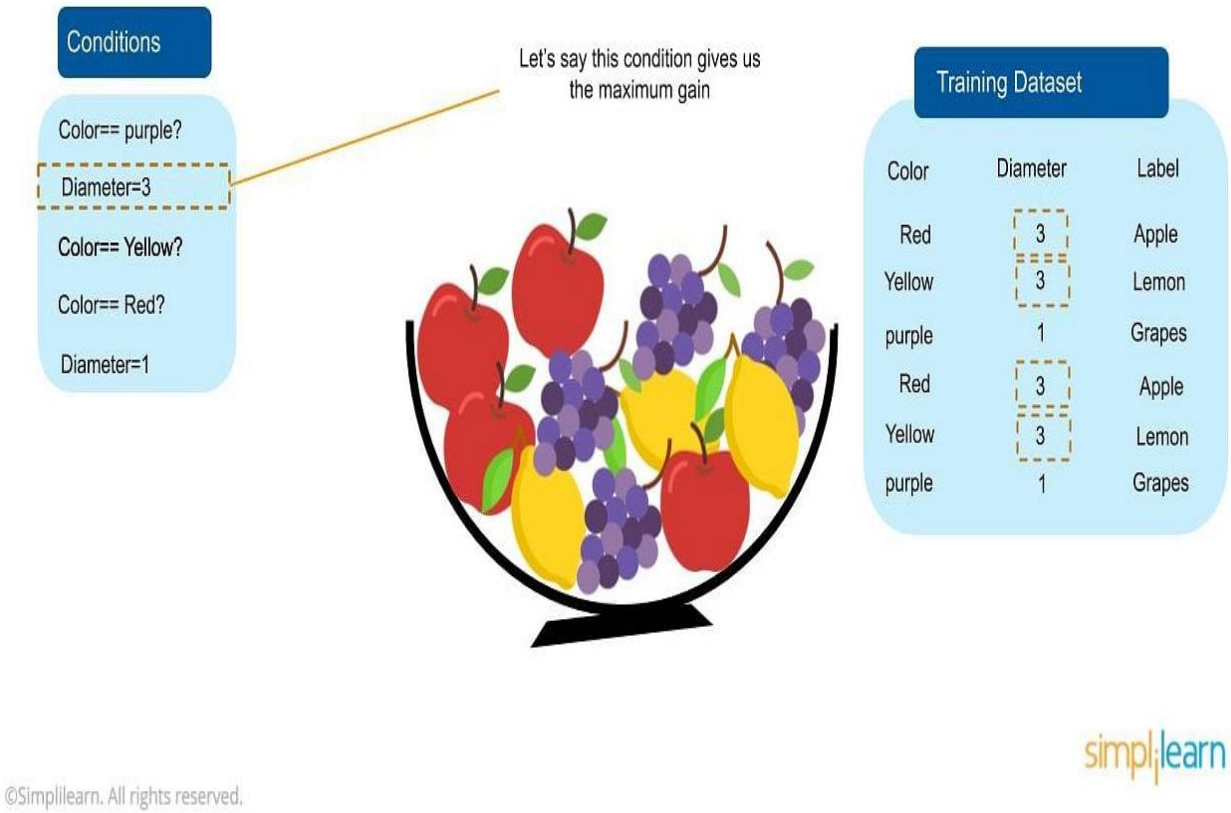
Leaf Node :- A leaf node is a node that carries the classification or the decision.

Decision Node :- A node that has two or more branches.

Root Node :- The root node is the topmost decision node, which is where you have all of your data.

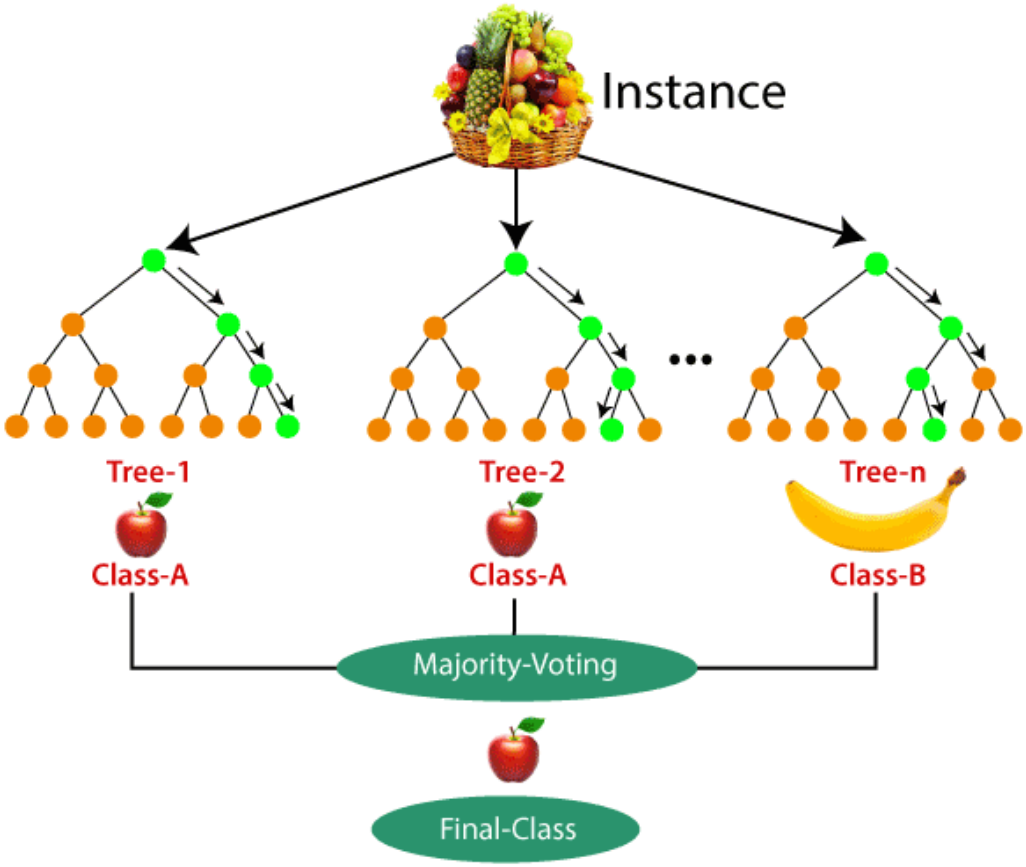
4.3.1 Case Study 1

Let's say we want to classify the different types of fruits in a bowl based on various features, but the bowl is cluttered with a lot of options. You would create a training dataset that contains information about the fruit, including colors, diameters, and specific labels (i.e., apple, grapes, etc.) You would then need to split the data by sorting out the smallest piece so that you can split it in the biggest way possible. You might want to start by splitting your fruits by diameter and then by color. You would want to keep splitting until that particular node no longer needs it, and you can predict a specific fruit with 100 percent accuracy.



4.3.2 Case Study 2

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:



4.4 Important Hyperparameters

Hyperparameters are used in random forests to either enhance the performance and predictive power of models or to make the model faster.

- n_estimators: Number of trees built by the algorithm before averaging the products.
- max_features: Maximum number of features random forest uses before considering splitting a node.
- mini_sample_leaf: Determines the minimum number of leaves required to split an internal node.

The following hyperparameters are used to increase the speed of the model:

- n_jobs: Conveys to the engine how many processors are allowed to use. If the value is 1, it can use only one processor, but if the value is -1,, there is no limit.
- random_state: Controls randomness of the sample. The model will always produce the same results if it has a definite value of random state and if it has been given the same hyperparameters and the same training data.
- oob_score: OOB (Out Of the Bag) is a random forest cross-validation method. In this, one-third of the sample is not used to train the data but to evaluate its performance.

5. Python Implementation Of Random Forest Algorithm

For implementing random forest algorithm we need a dataset. Just consider a dataset user_data for an example.

Implementation steps are given below:

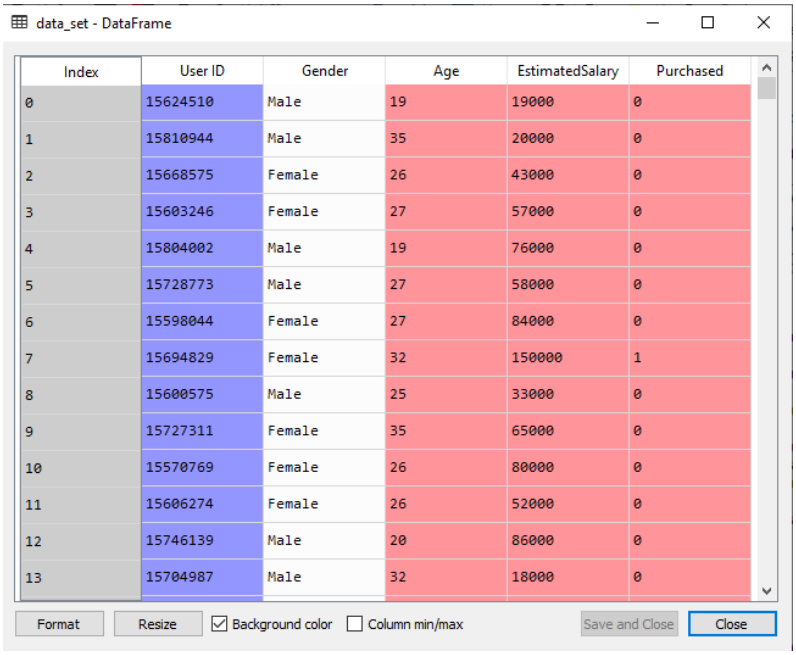
- Data Pre-processing step
- Fitting the Random forest algorithm to the Training set
- Predicting the test result
- Test accuracy of the result (Creation of Confusion matrix)
- Visualizing the test set result.
- Visualizing the training set result.

5.1 Data Pre-processing Step

Below is the code for the pre-processing step:

```
# importing libraries
import numpy as nm
import matplotlib.pyplot as mtp
import pandas as pd

#importing datasets
data_set= pd.read_csv('user_data.csv')

#Extracting Independent and dependent Variable
x= data_set.iloc[:, [2,3]].values
y= data_set.iloc[:, 4].values

# Splitting the dataset into training and test set.
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test= train_test_split(x, y, test_size= 0.25, random_state=0)

#feature Scaling
from sklearn.preprocessing import StandardScaler
st_x= StandardScaler()
x_train= st_x.fit_transform(x_train)
x_test= st_x.transform(x_test)
```

We have processed the data when we have loaded the dataset:



Index	User ID	Gender	Age	EstimatedSalary	Purchased
0	15624510	Male	19	19000	0
1	15810944	Male	35	20000	0
2	15668575	Female	26	43000	0
3	15603246	Female	27	57000	0
4	15804002	Male	19	76000	0
5	15728773	Male	27	58000	0
6	15598044	Female	27	84000	0
7	15694829	Female	32	150000	1
8	15600575	Male	25	33000	0
9	15727311	Female	35	65000	0
10	15570769	Female	26	80000	0
11	15606274	Female	26	52000	0
12	15746139	Male	20	86000	0
13	15704987	Male	32	18000	0

5.2 Fitting the Random Forest algorithm to the training set:

Now we will fit the Random forest algorithm to the training set. To fit it, we will import the RandomForestClassifier class from the sklearn.ensemble library. The code is given below:

```
#Fitting Decision Tree classifier to the training set
from sklearn.ensemble import RandomForestClassifier
classifier= RandomForestClassifier(n_estimators= 10, criterion="entropy")
classifier.fit(x_train, y_train)
```

In the above code, the classifier object takes below parameters:

- **n_estimators=** The required number of trees in the Random Forest. The default value is 10. We can choose any number but need to take care of the overfitting issue.
- **criterion=** It is a function to analyze the accuracy of the split. Here we have taken "entropy" for the information gain.

Output:

```
RandomForestClassifier(bootstrap=True, class_weight=None, criterion='entropy',
                       max_depth=None, max_features='auto', max_leaf_nodes=None,
                       min_impurity_decrease=0.0, min_impurity_split=None,
                       min_samples_leaf=1, min_samples_split=2,
                       min_weight_fraction_leaf=0.0, n_estimators=10,
                       n_jobs=None, oob_score=False, random_state=None,
                       verbose=0, warm_start=False)
```

5.3 Predicting the test result

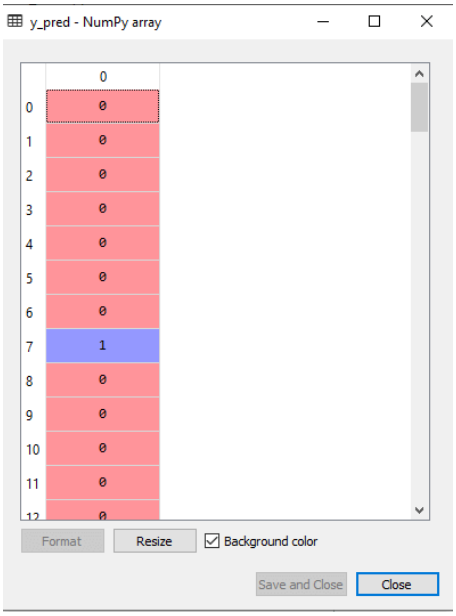
Since our model is fitted to the training set, so now we can predict the test result. For prediction, we will create a new prediction vector y_pred . Below is the code for it:

```
#Predicting the test set result
y_pred= classifier.predict(x_test)
```

The prediction vector is given as:

Output:

The prediction vector is given as:



By checking the above prediction vector and test set real vector, we can determine the incorrect predictions done by the classifier.

5.4 Test accuracy of the result (Creation of Confusion matrix)

Now we will create the confusion matrix to determine the correct and incorrect predictions. Below is the code for it:

```
#Creating the Confusion matrix
from sklearn.metrics import confusion_matrix
cm= confusion_matrix(y_test, y_pred)
```

Output:

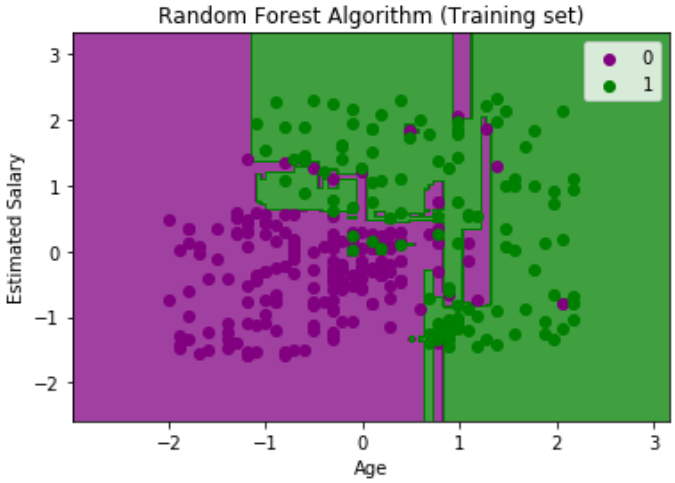
As we can see in the above matrix, there are 4+4= 8 incorrect predictions and 64+28= 92 correct predictions.

5.5 Visualizing the training set result.

Here we will visualize the training set result. To visualize the training set result we will plot a graph for the Random forest classifier. The classifier will predict yes or No for the users who have either Purchased or Not purchased the SUV car. Given below is the code for that.

```
from matplotlib.colors import ListedColormap
x_set, y_set = x_train, y_train
x1, x2 = nm.meshgrid(nm.arange(start = x_set[:, 0].min() - 1, stop = x_set[:, 0].max() + 1, step  =0.01),
nm.arange(start = x_set[:, 1].min() - 1, stop = x_set[:, 1].max() + 1, step = 0.01))
mtp.contourf(x1, x2, classifier.predict(nm.array([x1.ravel(), x2.ravel()]).T).reshape(x1.shape),
alpha = 0.75, cmap = ListedColormap(('purple','green' )))
mtp.xlim(x1.min(), x1.max())
mtp.ylim(x2.min(), x2.max())
for i, j in enumerate(nm.unique(y_set)):
    mtp.scatter(x_set[y_set == j, 0], x_set[y_set == j, 1],
        c = ListedColormap(('purple', 'green'))(i), label = j)
mtp.title('Random Forest Algorithm (Training set)')
mtp.xlabel('Age')
mtp.ylabel('Estimated Salary')
mtp.legend()
mtp.show()
```

Output:



The above image is the visualization result for the Random Forest classifier working with the training set result. It is very much similar to the Decision tree classifier.

Each data point corresponds to each user of the user_data, and the purple and green regions are the prediction regions. The purple region is classified for the users who did not purchase the SUV car, and the green region is for the users who purchased the SUV.

So, in the Random Forest classifier, we have taken 10 trees that have predicted Yes or NO for the Purchased variable. The classifier took the majority of the predictions and provided the result.

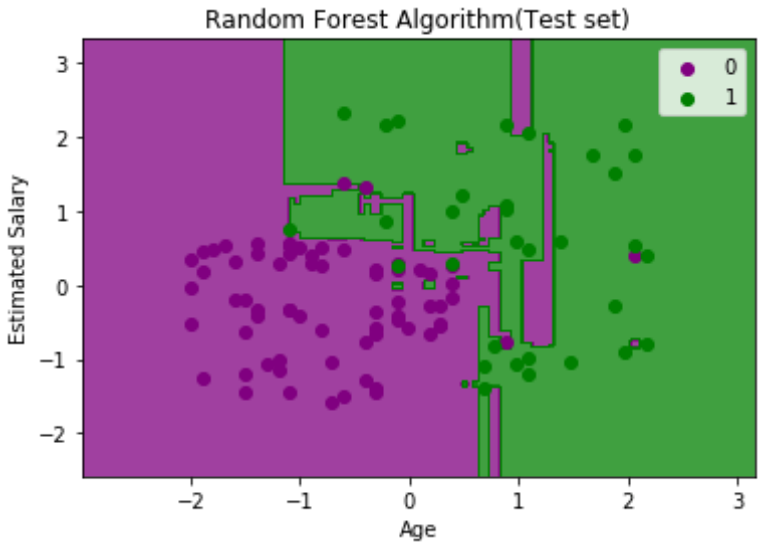
5.6. Visualizing the test set result

Now we will visualize the test set result. Below is the code for it:

```
#Visulaizing the test set result

from matplotlib.colors import ListedColormap
x_set, y_set = x_test, y_test
x1, x2 = nm.meshgrid(nm.arange(start = x_set[:, 0].min()  1, stop = x_set[:, 0].max() + 1, step  =0.01),

nm.arange(start = x_set[:, 1].min() - 1, stop = x_set[:, 1].max() + 1, step = 0.01))
mtp.contourf(x1, x2, classifier.predict(nm.array([x1.ravel(), x2.ravel()]).T).reshape(x1.shape),

alpha = 0.75, cmap = ListedColormap(('purple','green' )))
mtp.xlim(x1.min(), x1.max())
mtp.ylim(x2.min(), x2.max())
for i, j in enumerate(nm.unique(y_set)):
    mtp.scatter(x_set[y_set == j, 0], x_set[y_set == j, 1],
        c = ListedColormap(('purple', 'green'))(i), label = j)
mtp.title('Random Forest Algorithm(Test set)')
```

```
mtp.xlabel('Age')
mtp.ylabel('Estimated Salary')
mtp.legend()
mtp.show()
```

Output:



The above image is the visualization result for the test set. We can check that there is a minimum number of incorrect predictions (8) without the Overfitting issue. We will get different results by changing the number of trees in the classifier.

6. Applications of Random Forest Algorithm

Some of the applications of Random Forest Algorithm are listed below:

1. Banking: It predicts a loan applicant's solvency. This helps lending institutions make a good decision on whether to give the customer loan or not. They are also being used to detect fraudsters.
2. Health Care: Health professionals use random forest systems to diagnose patients. Patients are diagnosed by assessing their previous medical history. Past medical records are reviewed to establish the proper dosage for the patients.
3. Stock Market: Financial analysts use it to identify potential markets for stocks. It also enables them to remember the behaviour of stocks.
4. E-Commerce: Through this system, e-commerce vendors can predict the preference of customers based on past consumption behaviour.

7. When to avoid Random forest algorithm

Random Forests Algorithms are not ideal in the following situations:

1. Extrapolation: Random Forest regression is not ideal in the extrapolation of data. Unlike linear regression, which uses existing observations to estimate values beyond the observation range.
2. Sparse Data: Random Forest does not produce good results when the data is sparse. In this case, the subject of features and bootstrapped sample will have an invariant space. This will lead to unproductive spills, which will affect the outcome.

8. Advantages & Disadvantages

Advantages:

- Can perform both Regression and classification tasks.
- Produces good predictions that can be understood easily.
- Can handle large data sets efficiently.
- Provides a higher level of accuracy in predicting outcomes over the decision algorithm.

Disadvantages:

- While using a Random Forest Algorithm, more resources are required for computation.
- It Consumes more time compared to the decision tree algorithm.
- Less intuitive when we have an extensive collection of decision trees.
- Extremely complex and requires more computational resources.

9. Conclusion

Random forest is a great choice if anyone wants to build the model fast and efficiently, as one of the best things about the random forest is it can handle missing values. It is one of the best techniques with high performance, widely used in various industries for its efficiency. It can handle binary, continuous, and categorical data. Overall, random forest is a fast, simple, flexible, and robust model with some limitations.

REFERENCES

1. Wikipedia page on Random forest algorithm :- https://en.wikipedia.org/wiki/Random_forest
2. Javapoint tutorials page on Random forest algorithm :- https://www.javatpoint.com/machine-learning-random-forest-algorithm
3. Simplilearn tutorial page on Random Forest Algorithm :- https://www.simplilearn.com/tutorials/machine-learning-tutorial/random-forest-algorithm
4. Simplilearn youtube channel