THE UNIVERSITY OF BURDWAN

DEPARTMENT OF STATISTICS

A project work on

Topic : Predict the client will subscribe a term Deposit using ensemble learning model

Course Code: MSST 406

Submitted To The

Department Of Statistics
The University Of BurDwan
M.Sc. SEMESTER –IV EXAMINATION 2023

By

KEYA MONDAL

ROLL NO	:	BUR/ST/2021/009
REG. NO.	:	201801015855 of 2018-19
SESSION	:	2021-2023

CERTIFICATE

This is to certify that KEYA MONDAL (Roll No. - BUR/ST/2021/009) student of M.Sc. semester IV, Department of Statistics from The University of Burdwan has prepared the project work entitled

"Predict the client will subscribe a term Deposit using ensemble learning model" based on direct marketing campaigns of a Portuguese banking institution data.

Teacher's Signature

Place:

Dated:

Table of Contents

ACKNOWLEDGEMENT

Presentation, Inspiration and Motivation have always played a key role in the success any venture.

Primarily I would like to thank the supreme power the Almighty God who is obviously the one who has always guided me to work on the right path of life. I am greatly obliged in taking the opportunity to sincerely thanks our professors for helping and encouraging in this project.

I am also thankful to our respected professor Dr. Arindam Gupta (HOD), Department of Statistics, The University of Burdwan for his continuous support and motivation towards the completion of my project work.

I am also express my sincere thanks and gratitude to all my teachers of our department of their constant encouragement and inspiration in carrying out my project work.

Lastly, I thank almighty, my parents, brother and friends for their constant encouragement without which this project not be possible.

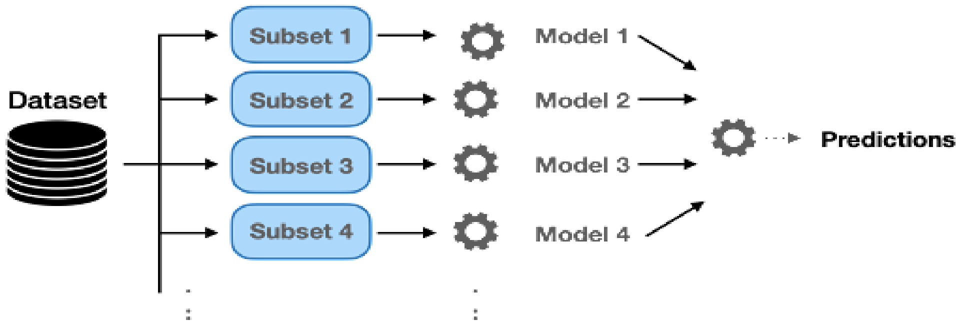
I have no valuable words to express my thanks, but my heart is still full of favours receive from every person.

Initial proposal:

Ensemble models can be used in various ways in banking data analysis to improve predictive performance and decision-making. Here are a few examples: Credit scoring, risk management, fraud detection. I will explain predict the client will subscribed a term deposit and also demonstrate how it can be used to analyze ensemble model using python. I am also built a web page help of html with CSS. If you go to web page you can able to see information about data and data related plot for understand data.

What is ensemble model?

An ensemble is a supervised learning algorithm. Supervised learning algorithms are usually described as performing the task of find a suitable data that will make better predictions with a specific problem. Ensembles combine multiple facts to form a better result. Ensemble modeling is the method of running two or more associated but different models and then combines the results into a single score to improve the accuracy of predictive data and data mining applications. In machine learning, ensemble methods use several algorithms to get better predictive performance.



The main benefits of Ensemble models are:

- Better Forecasting
- More Constant model
- Better results
- Reduces error

Data Acquisition:

<u>Data Description</u>: The data is related with direct marketing campaigns of a Portuguese banking institution. The marketing campaigns were based on phone calls. Often, more than one contact to the same client was required, in order to access if the product (bank term deposit) would be ('yes') or not ('no') subscribed.

DataSource:https://archive.ics.uci.edu/ml/machine-learning-databases/00222/bank-additional.zip

Date Donated : 20-++12-02-14

Domain	Data Characteristics	Area	Number of Row	Number of column
Banking	Multivariate	Business	41188	21

Column Description:

Age (numeric)
Job : type of job (categorical: 'admin.', 'blue-collar', 'entrepreneur', 'housemaid', 'management', 'retired', 'self-employed', 'services', 'student', 'technician', 'unemployed', 'unknown')

Marital : marital status (categorical: 'divorced', 'married', 'single', 'unknown' ; note: 'divorced' means divorced or widowed)

Education (categorical: 'basic.4y', 'basic.6y', 'basic.9y', 'high.school', 'illiterate', 'professional.course', 'university.degree', 'unknown')

Default: has credit in default? (categorical: 'no', 'yes', 'unknown')

Housing: has housing loan? (categorical: 'no', 'yes', 'unknown')

Loan: has personal loan? (categorical: 'no', 'yes', 'unknown')

Related with the last contact of the current campaign:

Contact: contact communication type (categorical: 'cellular','telephone')

Month: last contact month of year (categorical: 'jan', 'feb', 'mar', ..., 'nov', 'dec')

Day_of_week: last contact day of the week (categorical: 'mon','tue','wed','thu','fri')

Duration: last contact duration, in seconds (numeric).

Campaign: number of contacts performed during this campaign and for this client (numeric, includes last contact)

Pdays: number of days that passed by after the client was last contacted from a previous campaign (numeric; 999 means client was not previously contacted)

Previous: number of contacts performed before this campaign and for this client (numeric)

Poutcome: outcome of the previous marketing campaign (categorical: 'failure','nonexistent','success')

Social and economic context attributes

Emp.var.rate: employment variation rate - quarterly indicator(Employment variation rate (emp. var. rate) has negative influence, which means the change of the employment rate will make customers less likely to subscribe a term deposit.

Cons.price.idx: consumer price index - monthly indicator (numeric)(The Consumer Price Index (CPI) measures the change in prices paid by consumers for goods and services.

Cons.conf.idx: consumer confidence index - monthly indicator(This consumer confidence indicator provides an indication of future developments of households' consumption and saving, based upon answers regarding their expected financial situation.

Euribor3m: euribor 3 month rate - daily indicator (numeric) (3-month EURIBOR means the rate for deposits in euros for a period of the 3 months, expressed as a percentageNr.employed: number of employees

y - has the client subscribed a term deposit? (binary: 'yes', 'no')

Objective:

- The classification goal is to predict if the client will subscribe (yes/no) a term deposit (variable y).
- I want to see which model is better for this bank data. If someone gives similar data in the future, I can select specific models instead of using all models
- Business impact of bank

Key challenge:

- *drop duration variable because highly affects the output target*
- *convert to category variable*
- Dummy Coding: Dummy coding is a commonly used method for converting a categorical input variable into continuous variable. ‘Dummy’, as the name suggests is a duplicate variable which represents one level of a categorical variable. Presence of a level is represent by 1 and absence is represented by 0. For every level present, one dummy variable will be created
- Split the data into two part train and test

Steps and tasks:

- Import the necessary libraries
- Read the data as a data frame
- Perform basic EDA(exploratory data analysis) which should include the following and print out your insights at every step.
- Shape of the data
- Data type of each attribute
- Prepare the data to train a model
- Train a few standard classification algorithms, note and comment on their performances along different metrics
- Build the ensemble models and compare the results with the base models.

Model Description:

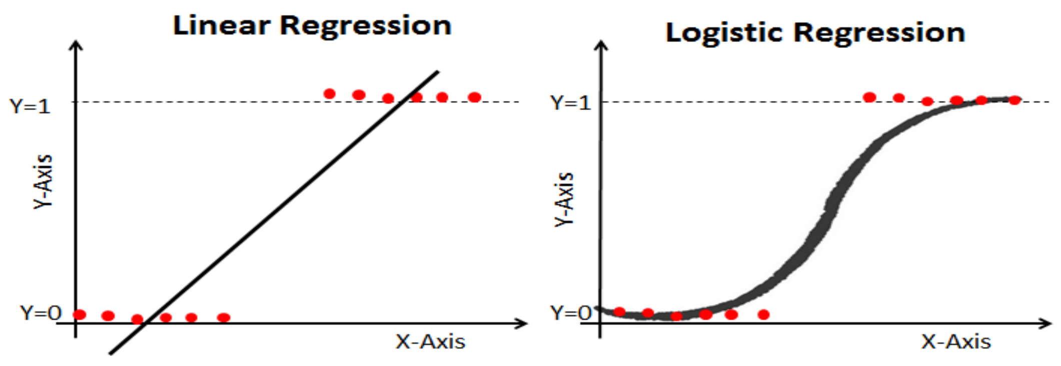
Logistic Regression:

Logistic regression is one of the most popular Machine Learning algorithms, which comes under the Supervised Learning technique. It is used for predicting the categorical dependent variable using a given set of independent variables. 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.

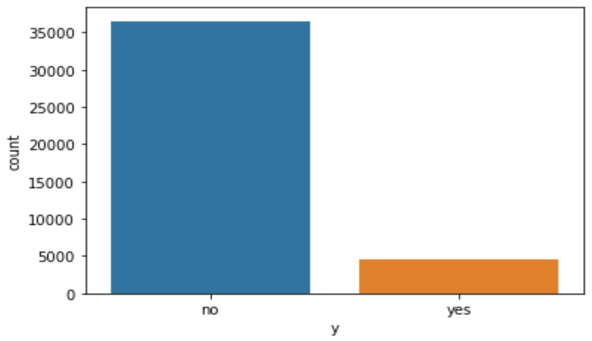
Logistic Function:

$$P = \frac{1}{1 + e^{-(\beta_0 + \beta_1 x)}}$$

The graph of a sigmoid function is as shown below



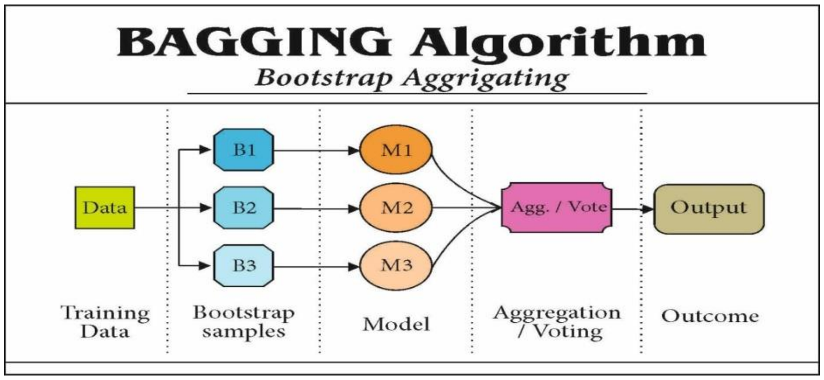
Conclusion from data:



In this dataset plot the Target variable. Here we can see yes appear udder 5000and no occur more than 35000 in Target Column. So we see an imbalanced pattern in the target Variables in bank data. According accuracy we are not used to talk about good or bad models. From confusion matrix we get accuracy, recall, precision and f1 score for Logistic regression are:

Accuracy	recall	precision	f1 score
0.895201	0.217489	0.539889	0.310069

2. Bagging :



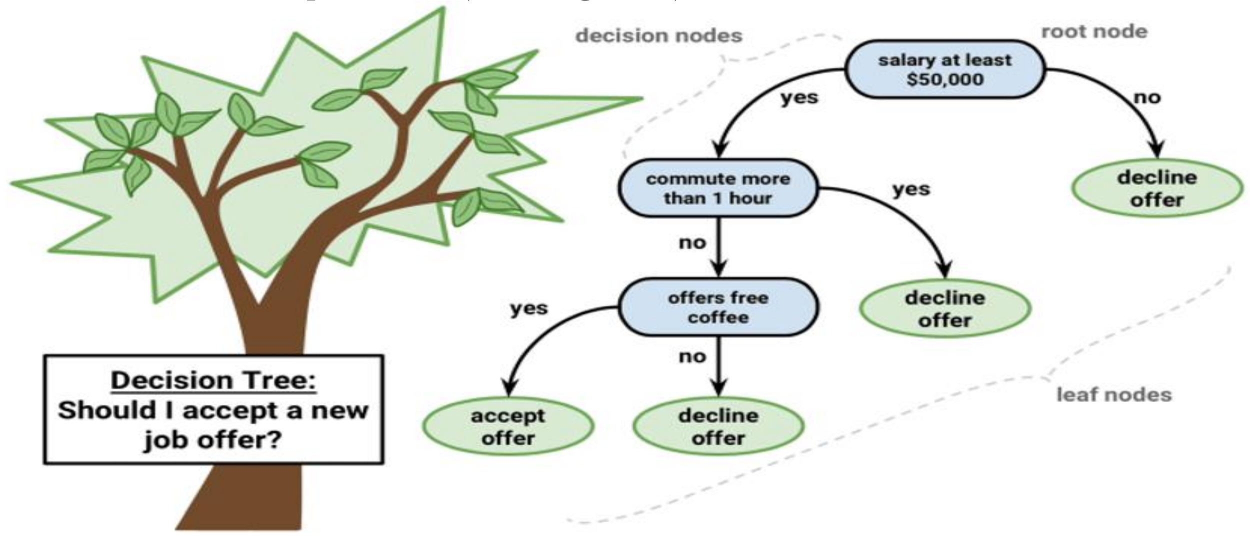
The bagging technique works by creating several subsets of the original training dataset through random sampling with replacement. Each subset is used to train a separate model, often referred to as a base or weak learner. These models are typically trained independently of each other, and they can be of the same type or different types.The key idea behind bagging is that by combining multiple models, the ensemble can reduce the variance and improve the overall performance compared to a single model.

From confusion matrix we get accuracy, recall, precision and f1 score for Bagging are:

Accuracy	recall	precision	f1 score
0.891721	0.278027	0.500000	0.357349

Decision Tree Algorithm:

Decision Tree algorithm belongs to the family of supervised learning algorithms. Unlike other supervised learning algorithms, the decision tree algorithm can be used for solving regression and classification problems too. The goal of using a Decision Tree is to create a training model that can use to predict the class or value of the target variable by learning simple decision rules inferred from prior data(training data).
The goal of using a Decision Tree is to create a training model that can use to predict the class or value of the target variable by learning simple decision rules inferred from prior data(training data).



Root Nodes – It is the node present at the beginning of a decision tree from this node the population starts dividing according to various features Decision.
Nodes – the nodes we get after splitting the root nodes are called Decision Node
Leaf Nodes – the nodes where further splitting is not possible are called leaf nodes or terminal nodes
Sub-tree – just like a small portion of a graph is called sub-graph similarly a sub-section of this decision tree is called sub-tree.
Pruning – is nothing but cutting down some nodes to stop over fitting.

Conclusion from data:

From confusion matrix we get accuracy, recall, precision and f1 score for Decision Tree are:

Accuracy	recall	precision	f1 score
0.900380	0.161435	0.664615	0.259771

Here accuracy value is higher but an imbalanced pattern in the target Variables in bank data so I ignore look at accuracy.

Random Forest Algorithm:

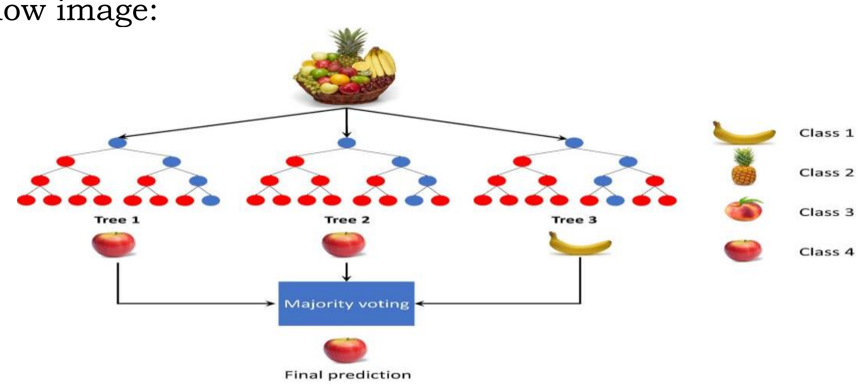
As the name suggests, "Random Forest is a classifier that contains a number of decision trees on various subsets of the given dataset and takes the average to improve the predictive accuracy of that dataset." Instead of relying on one decision tree, the random forest takes the prediction from each tree 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 over fitting.
- 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.

Steps involved in Random Forest Algorithm:

Step-1 – We first make subsets of our original data. We will do row sampling and feature sampling that means we'll select rows and columns with replacement and create subsets of the training dataset
Step- 2 – We create an individual decision tree for each subset we take
Step-3 – Each decision tree will give an output
Step 4 – Final output is considered based on Majority Voting if it's a classification problem and average if it's a regression problem.

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:



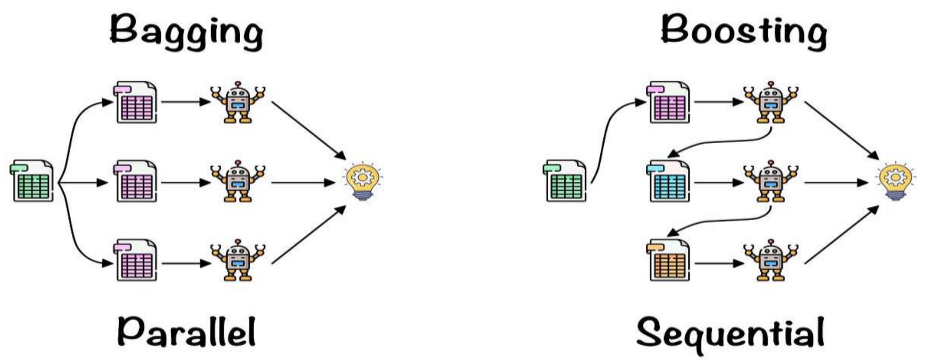
From confusion matrix we get :

Accuracy	recall	precision	f1 score
0.891317	0.278774	0.496671	0.496671

Gradient Boosting:

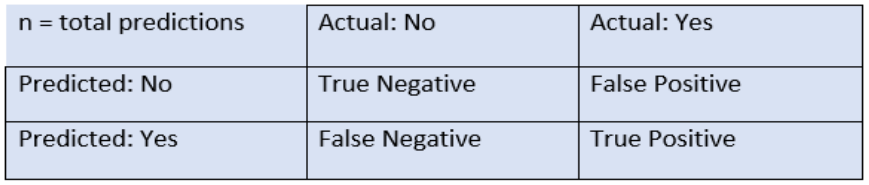
Gradient Boosting Machine (GBM) is one of the most popular forward learning ensemble methods in machine learning.
Gradient Boosting is a powerful boosting algorithm that combines several weak learners into strong learners, in which each new model is trained to minimize the loss function such as mean squared error or cross-entropy of the previous model using gradient descent. In each iteration, the algorithm computes the gradient of the loss function with respect to the predictions of the current ensemble and then trains a new weak model to minimize this gradient.
From confusion matrix we get:

Accuracy	recall	precision	f1 score
0.901109	0.227952	0.617409	0.332969



Confusion Matrix:

The confusion matrix is a matrix used to determine the performance of the classification models for a given set of test data. It can only be determined if the true values for test data are known. The matrix itself can be easily understood, but the related terminologies may be confusing. Since it shows the errors in the model performance in the form of a matrix, hence also known as an error matrix. Some features of Confusion matrix are given below:



n = total predictions	Actual: No	Actual: Yes
Predicted: No	True Negative	False Positive
Predicted: Yes	False Negative	True Positive

Calculations using Confusion Matrix:

Classification Accuracy:

In a confusion matrix, the accuracy value is considered good when it is high. Accuracy represents the overall correctness of the classification model and is calculated by dividing the sum of true positive and true negative predictions by the total number of instances.

$$\text{Accuracy}=\frac{TP+TN}{TP+FP+FN+TN}$$

A high accuracy value indicates that the model has made a significant number of correct predictions compared to incorrect predictions. However, it's important to note that accuracy alone may not provide the complete picture of a model's performance, especially when dealing with imbalanced datasets or when the cost of false positives and false negatives varies.
Here target variable (yes/no) is imbalance so I did not look at accuracy

Precision :
In a confusion matrix, the precision value is considered good when it is high. Precision measures the proportion of correctly predicted positive instances (true positives) out of the total instances predicted as positive (true positives + false positives).
A high precision value indicates that the model has a low rate of false positives, meaning it is accurately identifying positive instances. This is particularly important in scenarios where false positives can have significant consequences or when the cost of misclassifying positive instances is high.

$$\text{Precision}=\frac{TP}{TP+FP}$$

Recall:
In a confusion matrix, the recall value is considered good when it is high. Recall, also known as sensitivity or true positive rate, measures the proportion of correctly predicted positive instances (true positives) out of the total actual positive instances (true positives + false negatives).
A high recall value indicates that the model has a low rate of false negatives, meaning it effectively captures a significant portion of the positive instances in the dataset. This is particularly important when the consequence of missing positive instances is significant or when it is important to identify all positive instances.

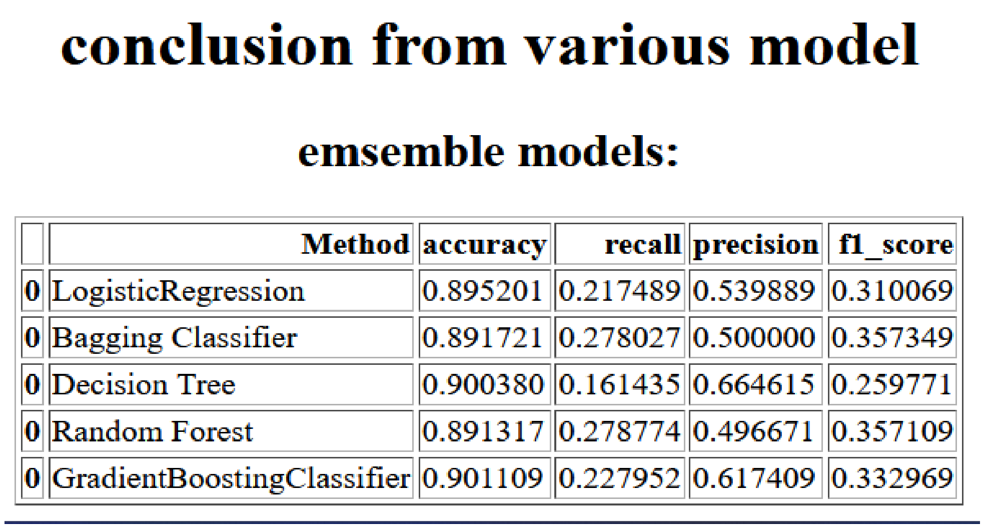
$$\text{Recall}=\frac{TP}{TP+FN}$$

F-measure: when f1 score value is good in confusion matrix the F1 score value is considered good when it is high. The F1 score is the harmonic mean of precision and recall and provides a single metric that balances both measures
A high F1 score indicates that the model has achieved a good balance between precision and recall, suggesting that it is performing well in terms of both minimizing false positives (precision) and capturing positive instances (recall).

$$\text{F-measure} = \frac{2 * Recall * Precision}{Recall + Precision}$$

Conclusion:

If you look at the recall and precision in the table recall is high value for Random Forest and precision value high for decision tree. If you compare both you should look at f1 score . Here f1 score is high for Bagging classifier.
I come to conclusion Bagging Classification and Random Forest is better than Other model. If someone gives similar data like this bank data in the future,
I can select Bagging Classification and Random Forest instead of using all models.

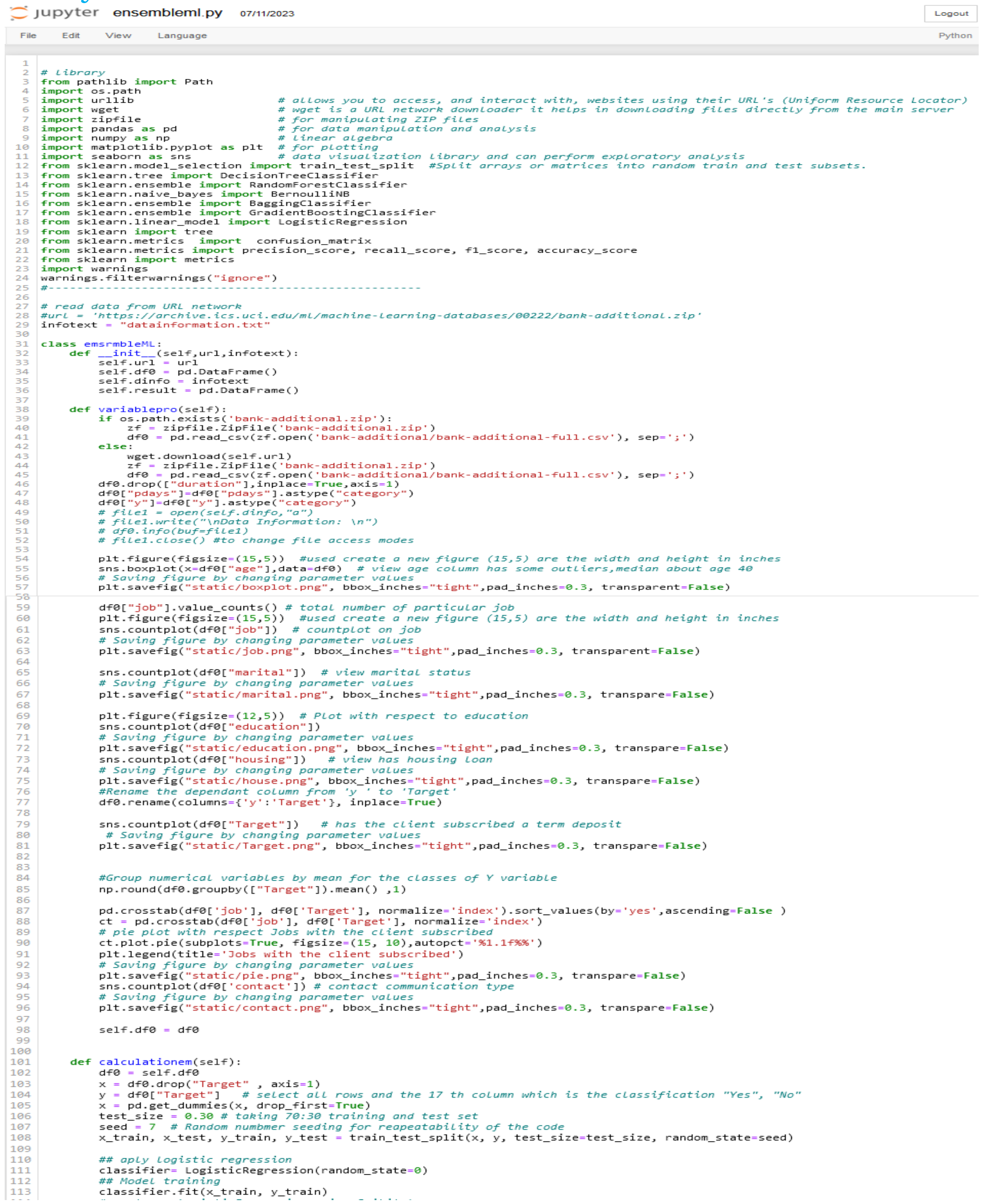


conclusion from various model

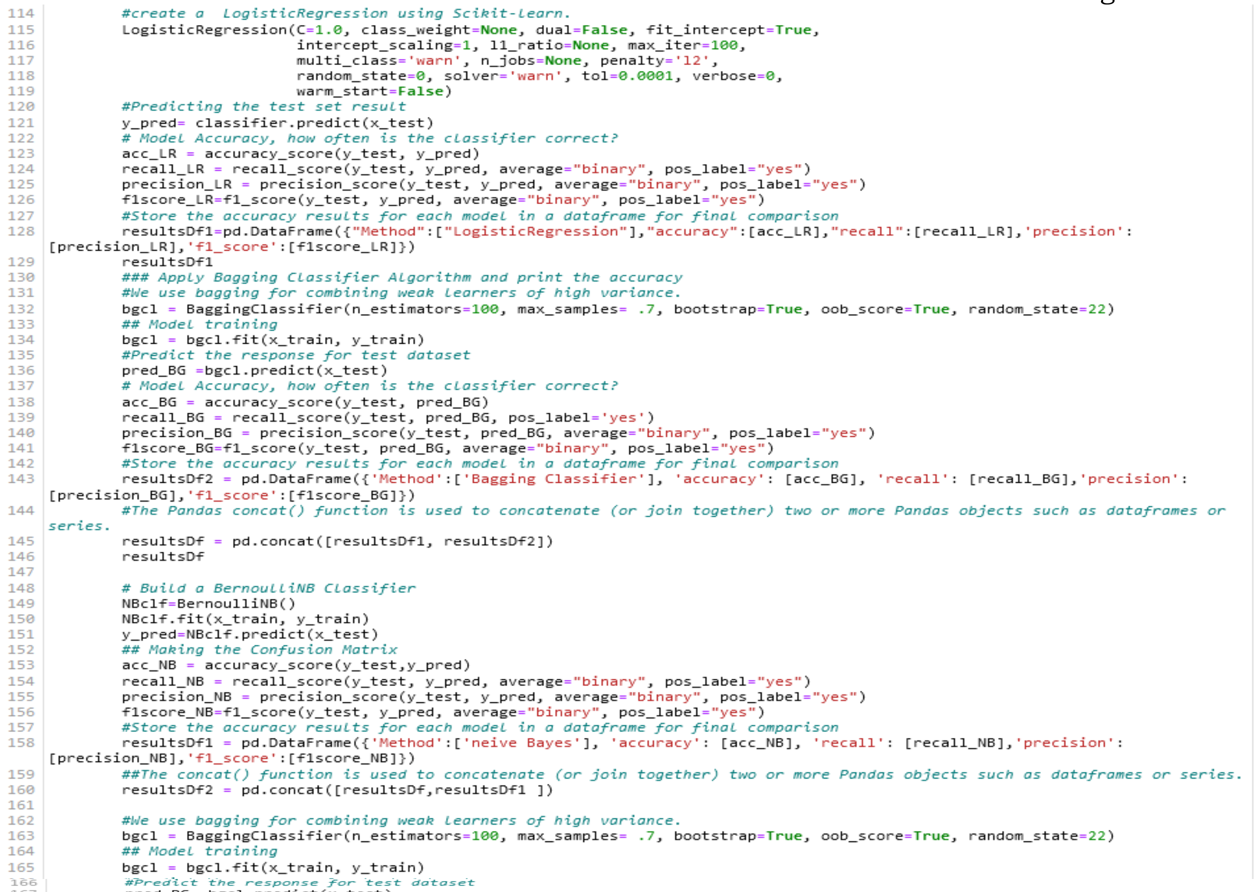
emsemble models:

	Method	accuracy	recall	precision	f1_score
0	LogisticRegression	0.895201	0.217489	0.539889	0.310069
0	Bagging Classifier	0.891721	0.278027	0.500000	0.357349
0	Decision Tree	0.900380	0.161435	0.664615	0.259771
0	Random Forest	0.891317	0.278774	0.496671	0.357109
0	GradientBoostingClassifier	0.901109	0.227952	0.617409	0.332969

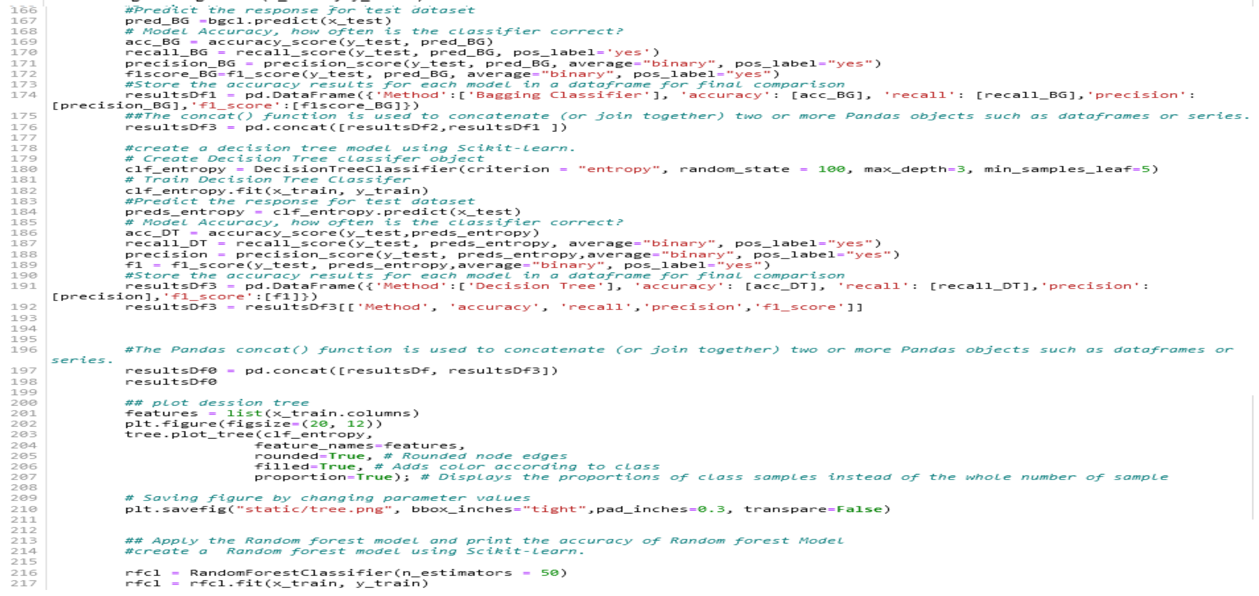
### Python code Details:



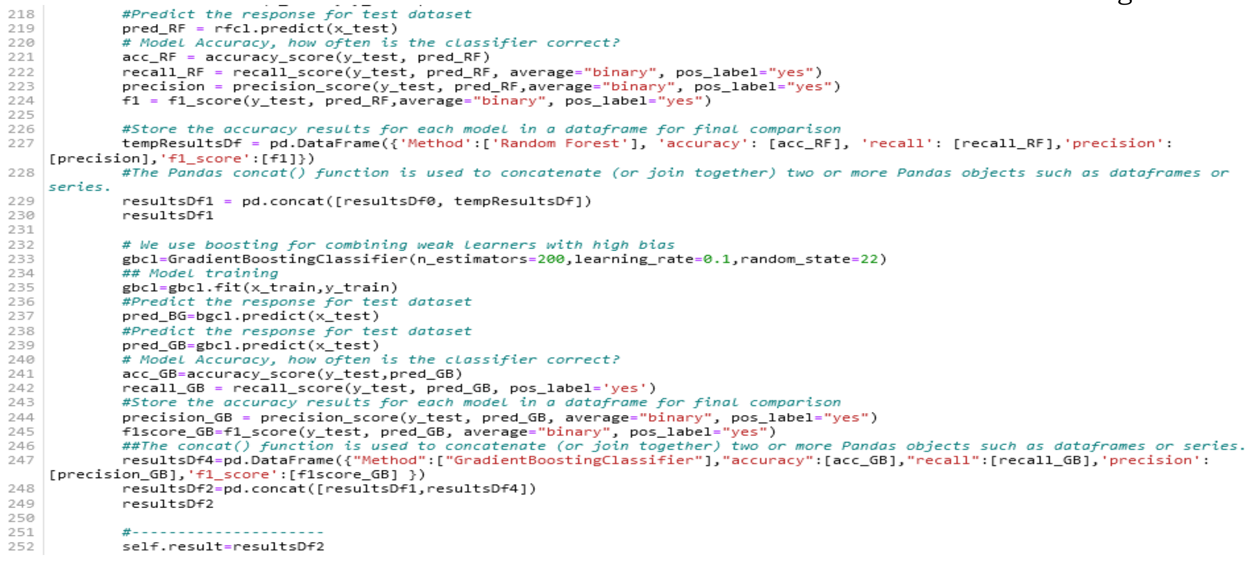
```

# Library
from pathlib import Path
import os.path
import urllib                    # allows you to access, and interact with, websites using their URL's (Uniform Resource Locator)
import wget                      # wget is a URL network downloader it helps in downloading files directly from the main server
import zipfile                   # for manipulating ZIP files
import pandas as pd              # for data manipulation and analysis
import numpy as np               # linear algebra
import matplotlib.pyplot as plt  # for plotting
import seaborn as sns            # data visualization library and can perform exploratory analysis
from sklearn.model_selection import train_test_split  #Split arrays or matrices into random train and test subsets.
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.naive_bayes import BernoulliNB
from sklearn.ensemble import BaggingClassifier
from sklearn.ensemble import GradientBoostingClassifier
from sklearn.linear_model import LogisticRegression
from sklearn import tree
from sklearn.metrics  import  confusion_matrix
from sklearn.metrics import precision_score, recall_score, f1_score, accuracy_score
from sklearn import metrics
import warnings
warnings.filterwarnings("ignore")
#-----------------------------------------------------

# read data from URL network
#url = 'https://archive.ics.uci.edu/ml/machine-learning-databases/00222/bank-additional.zip'
infotext = "datainformation.txt"

class emsrmbleML:
    def __init__(self,url,infotext):
        self.url = url
        self.df0 = pd.DataFrame()
        self.dinfo = infotext
        self.result = pd.DataFrame()

    def variablepro(self):
        if os.path.exists('bank-additional.zip'):
            zf = zipfile.ZipFile('bank-additional.zip')
            df0 = pd.read_csv(zf.open('bank-additional/bank-additional-full.csv'), sep=';')
        else:
            wget.download(self.url)
            zf = zipfile.ZipFile('bank-additional.zip')
            df0 = pd.read_csv(zf.open('bank-additional/bank-additional-full.csv'), sep=';')
        df0.drop(["duration"],inplace=True,axis=1)
        df0["pdays"]=df0["pdays"].astype("category")
        df0["y"]=df0["y"].astype("category")
        # file1 = open(self.dinfo,"a")
        # file1.write("\nData Information: \n")
        # df0.info(buf=file1)
        # file1.close() #to change file access modes

        plt.figure(figsize=(15,5))  #used create a new figure (15,5) are the width and height in inches
        sns.boxplot(x=df0["age"],data=df0)  # view age column has some outliers,median about age 40
        # Saving figure by changing parameter values
        plt.savefig("static/boxplot.png", bbox_inches="tight",pad_inches=0.3, transparent=False)

        df0["job"].value_counts() # total number of particular job
        plt.figure(figsize=(15,5))  #used create a new figure (15,5) are the width and height in inches
        sns.countplot(df0["job"])  # countplot on job
        # Saving figure by changing parameter values
        plt.savefig("static/job.png", bbox_inches="tight",pad_inches=0.3, transparent=False)

        sns.countplot(df0["marital"])  # view marital status
        # Saving figure by changing parameter values
        plt.savefig("static/marital.png", bbox_inches="tight",pad_inches=0.3, transpare=False)

        plt.figure(figsize=(12,5))  # Plot with respect to education
        sns.countplot(df0["education"])
        # Saving figure by changing parameter values
        plt.savefig("static/education.png", bbox_inches="tight",pad_inches=0.3, transpare=False)
        sns.countplot(df0["housing"])   # view has housing loan
        # Saving figure by changing parameter values
        plt.savefig("static/house.png", bbox_inches="tight",pad_inches=0.3, transpare=False)
        #Rename the dependant column from 'y ' to 'Target'
        df0.rename(columns={'y':'Target'}, inplace=True)

        sns.countplot(df0["Target"])   # has the client subscribed a term deposit
         # Saving figure by changing parameter values
        plt.savefig("static/Target.png", bbox_inches="tight",pad_inches=0.3, transpare=False)


        #Group numerical variables by mean for the classes of Y variable
        np.round(df0.groupby(["Target"]).mean() ,1)

        pd.crosstab(df0['job'], df0['Target'], normalize='index').sort_values(by='yes',ascending=False )
        ct = pd.crosstab(df0['job'], df0['Target'], normalize='index')
        # pie plot with respect Jobs with the client subscribed
        ct.plot.pie(subplots=True, figsize=(15, 10),autopct='%1.1f%%')
        plt.legend(title='Jobs with the client subscribed')
        # Saving figure by changing parameter values
        plt.savefig("static/pie.png", bbox_inches="tight",pad_inches=0.3, transpare=False)
        sns.countplot(df0['contact']) # contact communication type
        # Saving figure by changing parameter values
        plt.savefig("static/contact.png", bbox_inches="tight",pad_inches=0.3, transpare=False)

        self.df0 = df0



    def calculationem(self):
        df0 = self.df0
        x = df0.drop("Target" , axis=1)
        y = df0["Target"]   # select all rows and the 17 th column which is the classification "Yes", "No"
        x = pd.get_dummies(x, drop_first=True)
        test_size = 0.30 # taking 70:30 training and test set
        seed = 7  # Random numbmer seeding for reapeatability of the code
        x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=test_size, random_state=seed)

        ## aply logistic regression
        classifier= LogisticRegression(random_state=0)
        ## Model training
        classifier.fit(x_train, y_train)
```



```
        #create a  LogisticRegression using Scikit-Learn.
        LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True,
                           intercept_scaling=1, l1_ratio=None, max_iter=100,
                           multi_class='warn', n_jobs=None, penalty='l2',
                           random_state=0, solver='warn', tol=0.0001, verbose=0,
                           warm_start=False)
        #Predicting the test set result
        y_pred= classifier.predict(x_test)
        # Model Accuracy, how often is the classifier correct?
        acc_LR = accuracy_score(y_test, y_pred)
        recall_LR = recall_score(y_test, y_pred, average="binary", pos_label="yes")
        precision_LR = precision_score(y_test, y_pred, average="binary", pos_label="yes")
        f1score_LR=f1_score(y_test, y_pred, average="binary", pos_label="yes")
        #Store the accuracy results for each model in a dataframe for final comparison
        resultsDf1=pd.DataFrame({"Method":["LogisticRegression"],"accuracy":[acc_LR],"recall":[recall_LR],'precision':
[precision_LR],'f1_score':[f1score_LR]})
        resultsDf1
        ### Apply Bagging Classifier Algorithm and print the accuracy
        #We use bagging for combining weak learners of high variance.
        bgcl = BaggingClassifier(n_estimators=100, max_samples= .7, bootstrap=True, oob_score=True, random_state=22)
        ## Model training
        bgcl = bgcl.fit(x_train, y_train)
        #Predict the response for test dataset
        pred_BG =bgcl.predict(x_test)
        # Model Accuracy, how often is the classifier correct?
        acc_BG = accuracy_score(y_test, pred_BG)
        recall_BG = recall_score(y_test, pred_BG, pos_label='yes')
        precision_BG = precision_score(y_test, pred_BG, average="binary", pos_label="yes")
        f1score_BG=f1_score(y_test, pred_BG, average="binary", pos_label="yes")
        #Store the accuracy results for each model in a dataframe for final comparison
        resultsDf2 = pd.DataFrame({'Method':['Bagging Classifier'], 'accuracy': [acc_BG], 'recall': [recall_BG],'precision':
[precision_BG],'f1_score':[f1score_BG]})
        #The Pandas concat() function is used to concatenate (or join together) two or more Pandas objects such as dataframes or
series.
        resultsDf = pd.concat([resultsDf1, resultsDf2])
        resultsDf

        # Build a BernoulliNB Classifier
        NBclf=BernoulliNB()
        NBclf.fit(x_train, y_train)
        y_pred=NBclf.predict(x_test)
        ## Making the Confusion Matrix
        acc_NB = accuracy_score(y_test,y_pred)
        recall_NB = recall_score(y_test, y_pred, average="binary", pos_label="yes")
        precision_NB = precision_score(y_test, y_pred, average="binary", pos_label="yes")
        f1score_NB=f1_score(y_test, y_pred, average="binary", pos_label="yes")
        #Store the accuracy results for each model in a dataframe for final comparison
        resultsDf1 = pd.DataFrame({'Method':['neive Bayes'], 'accuracy': [acc_NB], 'recall': [recall_NB],'precision':
[precision_NB],'f1_score':[f1score_NB]})
        ##The concat() function is used to concatenate (or join together) two or more Pandas objects such as dataframes or series.
        resultsDf2 = pd.concat([resultsDf,resultsDf1 ])

        #We use bagging for combining weak learners of high variance.
        bgcl = BaggingClassifier(n_estimators=100, max_samples= .7, bootstrap=True, oob_score=True, random_state=22)
        ## Model training
        bgcl = bgcl.fit(x_train, y_train)
```



```
        #Predict the response for test dataset
        pred_BG =bgcl.predict(x_test)
        # Model Accuracy, how often is the classifier correct?
        acc_BG = accuracy_score(y_test, pred_BG)
        recall_BG = recall_score(y_test, pred_BG, pos_label='yes')
        precision_BG = precision_score(y_test, pred_BG, average="binary", pos_label="yes")
        f1score_BG=f1_score(y_test, pred_BG, average="binary", pos_label="yes")
        #Store the accuracy results for each model in a dataframe for final comparison
        resultsDf1 = pd.DataFrame({'Method':['Bagging Classifier'], 'accuracy': [acc_BG], 'recall': [recall_BG],'precision':
[precision_BG],'f1_score':[f1score_BG]})
        ##The concat() function is used to concatenate (or join together) two or more Pandas objects such as dataframes or series.
        resultsDf3 = pd.concat([resultsDf2,resultsDf1 ])

        #create a decision tree model using Scikit-Learn.
        # Create Decision Tree classifer object
        clf_entropy = DecisionTreeClassifier(criterion = "entropy", random_state = 100, max_depth=3, min_samples_leaf=5)
        # Train Decision Tree Classifer
        clf_entropy.fit(x_train, y_train)
        #Predict the response for test dataset
        preds_entropy = clf_entropy.predict(x_test)
        # Model Accuracy, how often is the classifier correct?
        acc_DT = accuracy_score(y_test,preds_entropy)
        recall_DT = recall_score(y_test, preds_entropy, average="binary", pos_label="yes")
        precision = precision_score(y_test, preds_entropy,average="binary", pos_label="yes")
        f1 = f1_score(y_test, preds_entropy,average="binary", pos_label="yes")
        #Store the accuracy results for each model in a dataframe for final comparison
        resultsDf3 = pd.DataFrame({'Method':['Decision Tree'], 'accuracy': [acc_DT], 'recall': [recall_DT],'precision':
[precision],'f1_score':[f1]})
        resultsDf3 = resultsDf3[['Method', 'accuracy', 'recall','precision','f1_score']]



        #The Pandas concat() function is used to concatenate (or join together) two or more Pandas objects such as dataframes or
series.
        resultsDf0 = pd.concat([resultsDf, resultsDf3])
        resultsDf0

        ## plot dession tree
        features = list(x_train.columns)
        plt.figure(figsize=(20, 12))
        tree.plot_tree(clf_entropy,
                       feature_names=features,
                       rounded=True, # Rounded node edges
                       filled=True, # Adds color according to class
                       proportion=True); # Displays the proportions of class samples instead of the whole number of sample

        # Saving figure by changing parameter values
        plt.savefig("static/tree.png", bbox_inches="tight",pad_inches=0.3, transpare=False)


        ## Apply the Random forest model and print the accuracy of Random forest Model
        #create a  Random forest model using Scikit-Learn.

        rfcl = RandomForestClassifier(n_estimators = 50)
        rfcl = rfcl.fit(x_train, y_train)
```



```
        #Predict the response for test dataset
        pred_RF = rfcl.predict(x_test)
        # Model Accuracy, how often is the classifier correct?
        acc_RF = accuracy_score(y_test, pred_RF)
        recall_RF = recall_score(y_test, pred_RF, average="binary", pos_label="yes")
        precision = precision_score(y_test, pred_RF,average="binary", pos_label="yes")
        f1 = f1_score(y_test, pred_RF,average="binary", pos_label="yes")

        #Store the accuracy results for each model in a dataframe for final comparison
        tempResultsDf = pd.DataFrame({'Method':['Random Forest'], 'accuracy': [acc_RF], 'recall': [recall_RF],'precision':[precision],'f1_score':[f1]})
        #The Pandas concat() function is used to concatenate (or join together) two or more Pandas objects such as dataframes or series.
        resultsDf1 = pd.concat([resultsDf0, tempResultsDf])
        resultsDf1

        # We use boosting for combining weak learners with high bias
        gbcl=GradientBoostingClassifier(n_estimators=200,learning_rate=0.1,random_state=22)
        ## Model training
        gbcl=gbcl.fit(x_train,y_train)
        #Predict the response for test dataset
        pred_BG=bgcl.predict(x_test)
        #Predict the response for test dataset
        pred_GB=gbcl.predict(x_test)
        # Model Accuracy, how often is the classifier correct?
        acc_GB=accuracy_score(y_test,pred_GB)
        recall_GB = recall_score(y_test, pred_GB, pos_label='yes')
        #Store the accuracy results for each model in a dataframe for final comparison
        precision_GB = precision_score(y_test, pred_GB, average="binary", pos_label="yes")
        f1score_GB=f1_score(y_test, pred_GB, average="binary", pos_label="yes")
        ##The concat() function is used to concatenate (or join together) two or more Pandas objects such as dataframes or series.
        resultsDf4=pd.DataFrame({"Method":["GradientBoostingClassifier"],"accuracy":[acc_GB],"recall":[recall_GB],'precision':[precision_GB],'f1_score':[f1score_GB] })
        resultsDf2=pd.concat([resultsDf1,resultsDf4])
        resultsDf2

        #---------------------
        self.result=resultsDf2
```

Reference:

- https://www.analyticsvidhya.com/blog/2018/06/comprehensive-guide-for-ensemble-models/
- An Introduction to Statistical Learning
  Gareth James • Daniela Witten •
  Trevor Hastie • Robert Tibshirani
- https://scikit-learn.org/stable/modules/ensemble.html
- https://www.javatpoint.com/confusion-matrix-in-machine-learning
- https://www.analyticsvidhya.com/blog/2020/04/confusion-matrix-machine-learning/
- https://www.tutorialspoint.com/scikit_learn/index.htm
- https://www.simplilearn.com/tutorials/machine-learning-tutorial/random-forest-algorith
- https://www.javatpoint.com/machine-learning-random-forest-algorithm