"TO STUDY THE BANK MARKETING | WHO WILL SUBSCRIBE FOR DEPOSIT?"

-By Pratiksha Kore.

A

PROJECT REPORT ON

"TO STUDY THE BANK MARKETING | WHO WILL SUBSCRIBE FOR DEPOSIT?"

...SUBMITTED TO...

(DEPARTMENT OF STATISTICS)

PADMABHUSHAN DR.
VASANTRAODADA PATIL
MAHAVIDYALAYA, TASGAON.

SHIVAJI UNIVERSITY, KOLHAPUR

FOR THE PARTIAL FULFILMENT OF DEGREE

M.SC. STATISTICS (PART-II)

BY

MISS. KORE PRATIKSHA JAYANT

UNDER THE GUIDANCE OF

PROF. PATIL.D.D

FOR THE ACADEMIC YEAR 2022-2023

CERTIFICATE

This is to certify that the project entitled ***"To Study The Bank Marketing| Who Will Subscribe for Deposit?"*** being submitted by ***Miss. Pratiksha Jayant Kore.*** As a partial fulfilment for the award of degree of M.Sc. in Statistics (Part-II) of P.D.V.P.College, Tasgaon is a record of bonafide work carried out by them under my supervision and guidance.

To the best of my knowledge and belief the matter present in the project has not been submitted elsewhere for for any other purpose.

Date:

Place- Tasgaon

Teacher in-charge *Examiner in charge* *Head of Department*

Prof.Mr.Patil.D.D.

ACKNOWLEDGEMENT

We would like to express our sincere thanks of gratitude to our project mentor and guide Prof. Mr. Patil. D. D., Prof. Mr. Nalwade. D. J., Prof. Miss. Zambre. P. R. and Dept. of Statistics for giving us the opportunity to do this wonderful project. It is a great honor to get continuous support and guidance throughout the project from both of them. It was quite a testing time in the initial phase as we were challenged understanding the data set and to pick out the best machine learning algorithms. The meetings we had with Mr. Patil. D. D. Sir helped us visualize the problem more practically and gained numerous other ideas on handling a machine learning assignment. This made us more equipped and kept us encouraged for the task. Our, acknowledgment would be incomplete without the mention of the weekly laboratory sessions delivered by Mr. Patil. D. D. Sir which contributed hugely to our learning and compiling the entire project.

Finally, we would like to thank all non-teaching staff for their help in the completion of this project further we are thankful to all of our college Friends who gave us the necessary information to carry this project. We give sincerest thanks to all our family members for keeping our spirit up and their immense support during every stage of the project work.

ABSTRACT

In this project, we aim to increase campaign efficiency by identifying the main factors that affect the success of a campaign and predicting whether the campaign will be successful to a certain client, namely, whether the client will subscribe a term deposit. As our data are imbalanced, we use resampling methods before building models. After preprocessing the data, we build four models: Logistic Regression, Support Vector Machine, Random Forest, XG Boost. The optimal model we get is the one using neural network algorithm.

INDEX

INTRODUCTION

The core business of a financial institution can be broadly classified as lending and borrowing. Lending generates revenue to the bank in the form of interest from customers with some level of default risk involved. Borrowing, or rather attracting public's savings into the bank is another source of revenue generation, which can be less risky than the former. A bank usually invests the customer's long-term deposits into riskier financial assets which can earn the better return than what they pay to their customer.

There is a stiff competition among the financial institutions/banks in increasing the customer base in their retail banking segment. Along with offering innovative products to the public, a huge amount of money is spent on marketing their products. The term deposit is very important among the diverse range of products and services offered by banks in retail banking segment. With advancement in data science and machine learning and availability of data, most banks are adapting to a data-driven decision. The dataset here consists of direct marketing by contacting the clients and assessing the success rate of sales made.

There has been a revenue decline for the Portuguese bank, and they would like to know what actions to take. After investigation, we found out that the root cause is that their clients are not depositing as frequently as before. Knowing that term deposits allow banks to hold onto a deposit for a specific amount of time, so banks can invest in higher gain financial products to make a profit. In addition, banks also hold a better chance to persuade term deposit clients into buying other products such as funds or insurance to further increase their revenues. As a result, the Portuguese bank would like to identify existing clients that have a higher chance to subscribe for a term deposit and focus marketing effort on such clients.

In this project, we apply machine learning algorithms to build a predictive model of the data set to provide a necessary suggestion for marketing campaign team. The goal is to predict whether a client will subscribe a term deposit (variable y) with the help of a given set of dependent variables. This is a real dataset collected from a Portuguese bank that used its own contact-center to do direct marketing campaigns to motivate and attract the clients for their term deposit scheme to enhance the business.

OBJECTIVE

- Analysis of Bank Marketing data using Logistic Regression, Support Vector Machine, Random Forest, XG Boost for dataset using python or R studio.

❑ Research Questions:

- To get useful insights from the data and predict if a new customer will accept a deposit offer or not.
- To predict if the deposit can be given to that client on the basis of various features like age, job, marital status, education, default, lone, etc.

Methodology-

1) Exploratory data analysis (EDA)
2) Data visualization & data cleaning
3) Feature encoding & encoding category
4) Data preprocessing
5) Model training & model evaluation
6) Conclusion
7) Model Deployment

DATA DESCRIPTION

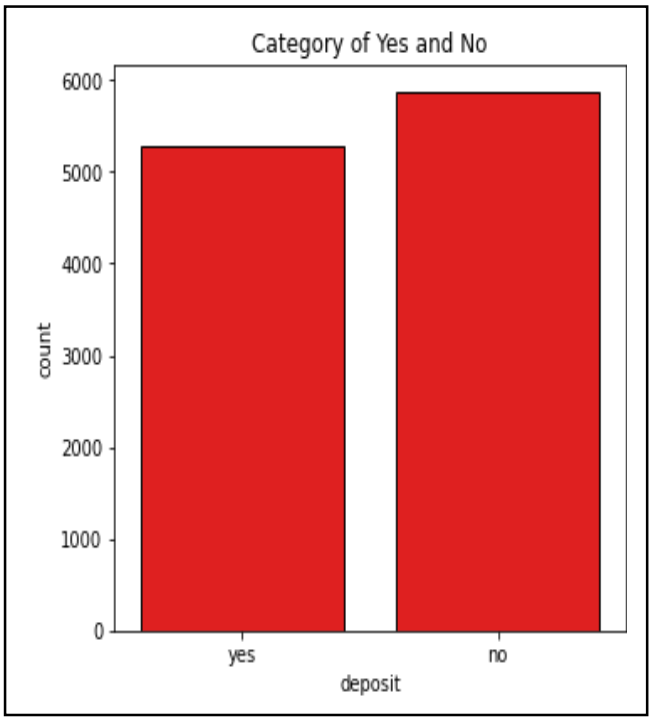
The dataset gives information about a marketing campaign of a financial institution and is taken from the website https://www.kaggle.com/datasets/janiobachmann/bank-marketing-dataset and is available in CSV format. The data set consists of 11163 rows and 17 columns/Features. There are attributes such as “age”, “job”, “marital”, “education”, etc.

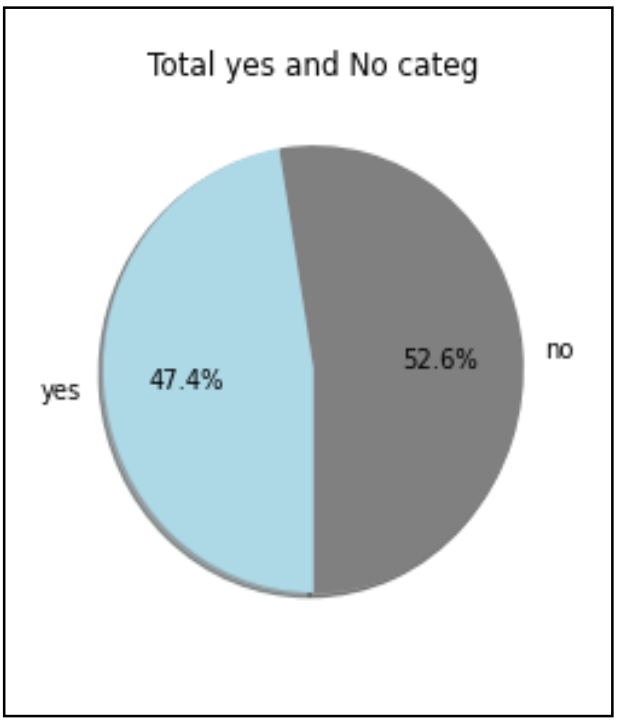
	age	job	marital	education	default	balance	housing	loan	contact	day	month	duration	campaign	pdays	previous	poutcome	deposit
0	59	admin.	married	secondary	no	2343	yes	no	unknown	5	may	1042	1	-1	0	unknown	yes
1	56	admin.	married	secondary	no	45	no	no	unknown	5	may	1467	1	-1	0	unknown	yes
….	….	……	…	…….	……	…..	……	….	….	….	….	….	….	….	….	……	….
11162	43	technician	married	secondary	no	0	no	yes	cellular	8	may	9	2	172	5	failure	no
11163	34	technician	married	secondary	no	0	no	no	cellular	9	jul	628	1	-1	0	unknown	no

The dataset consists of numerical as well as categorical values. So we need to convert categorical values into numerical form by coding.

EXPLORATORY DATA ANALYSIS (EDA)

- In the given data set there is no null value.
- We check the data is balanced or not i.e. total % of Yes or No Category

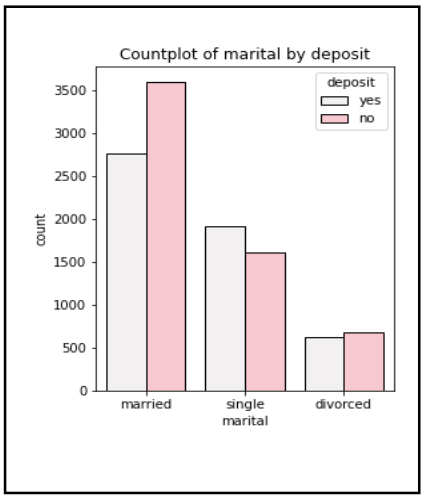




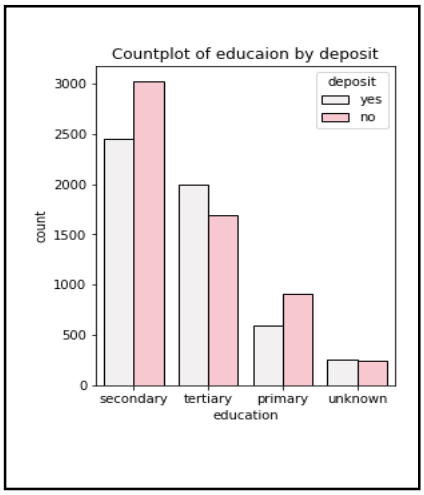
- The data is almost balanced of yes/no binary categories.

DATA VISUALIZATION

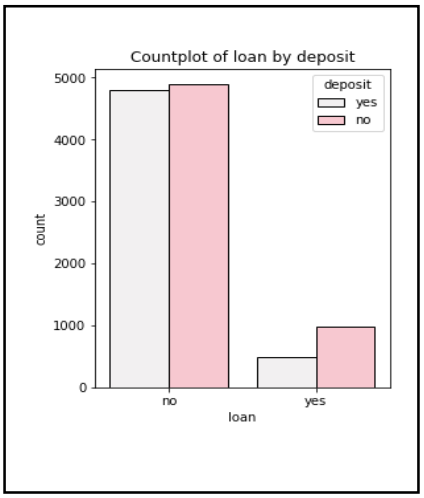
1) Univariate Data Analysis :



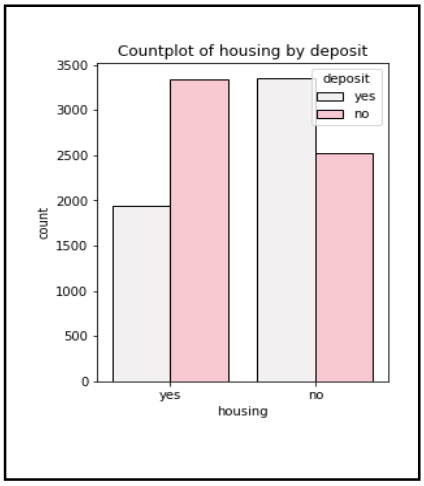
Single person is more likely to subscribe for deposit as compared to married person.



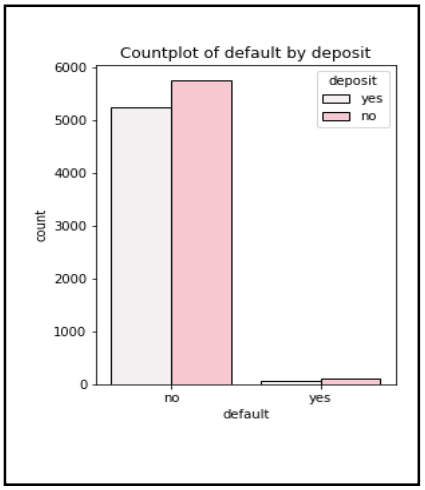
Tertiary person has more chances that will subscribe for the deposit as compared to unknow person.



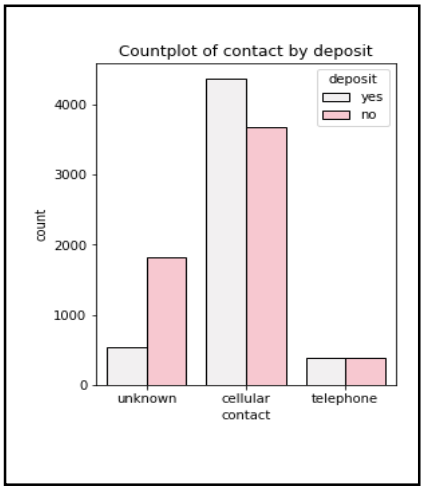
People with no/yes lone are not interested to subscribe the term deposit.



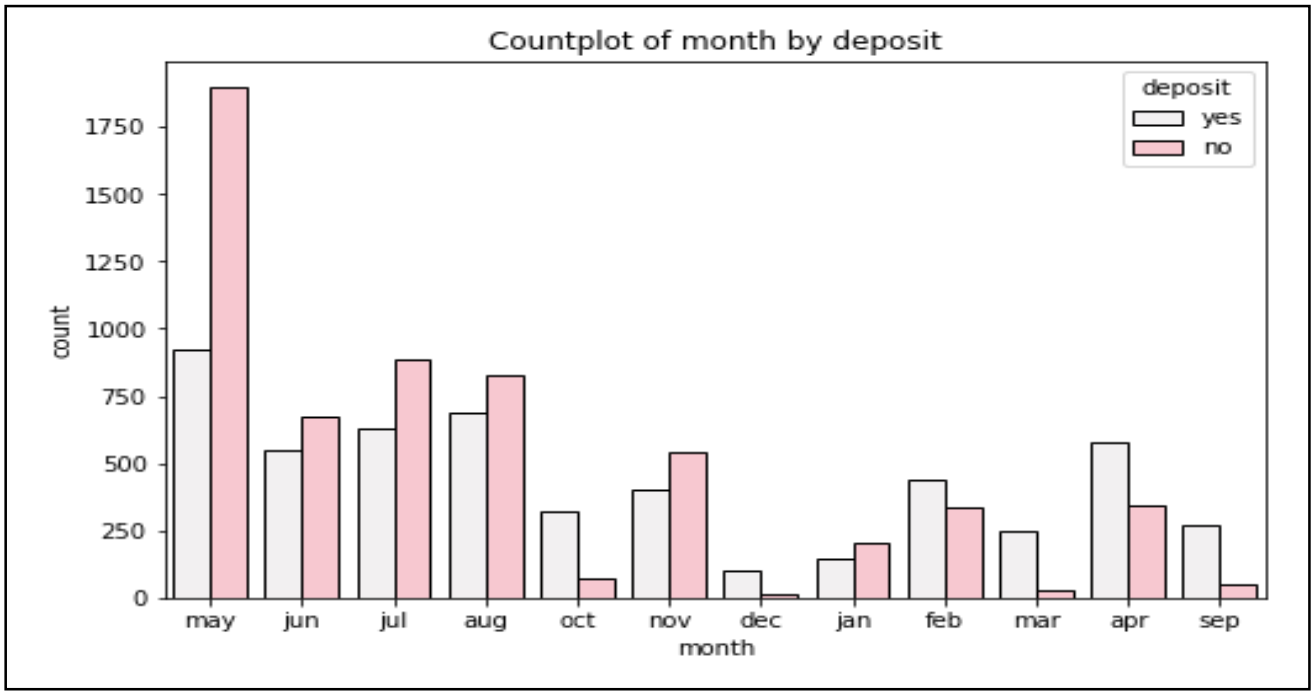
People who are not interested to take housing loans may be interested in subscribing for the term deposit.



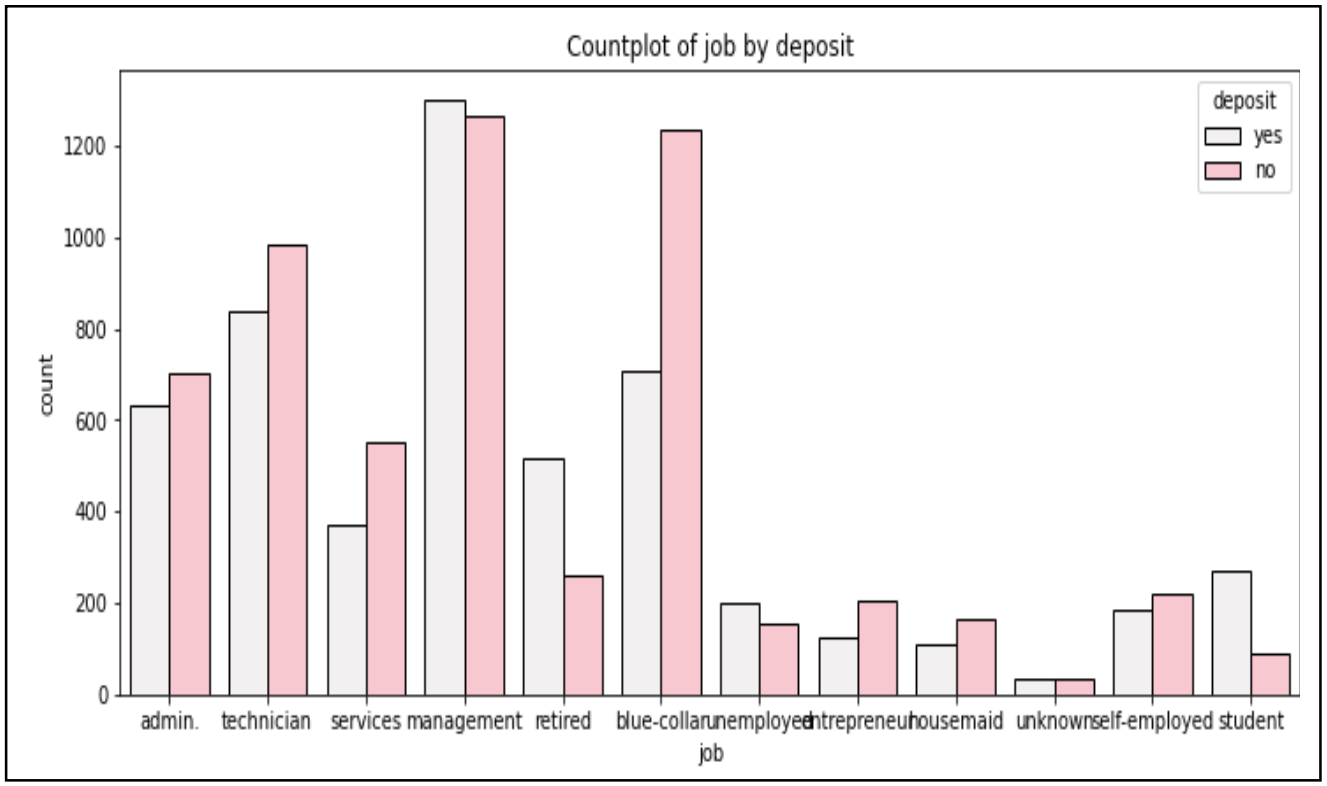
In the default feature there is person not interested for subscribed for deposit.



Cellular contact has more chances that person will subscribed the deposit while. unknown has very less.



In month of <u>may</u> the most of call was done (around 1000), while in <u>Dec</u> month this was very less (below 200).

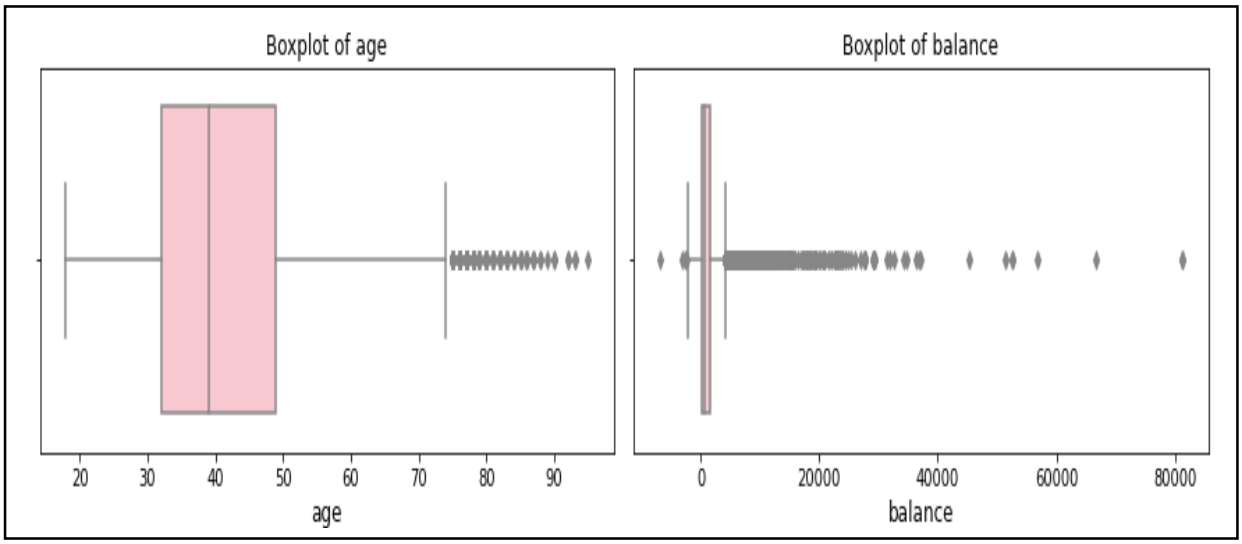


<u>Students and retired</u> people has more likely to subscribed for the term deposit.

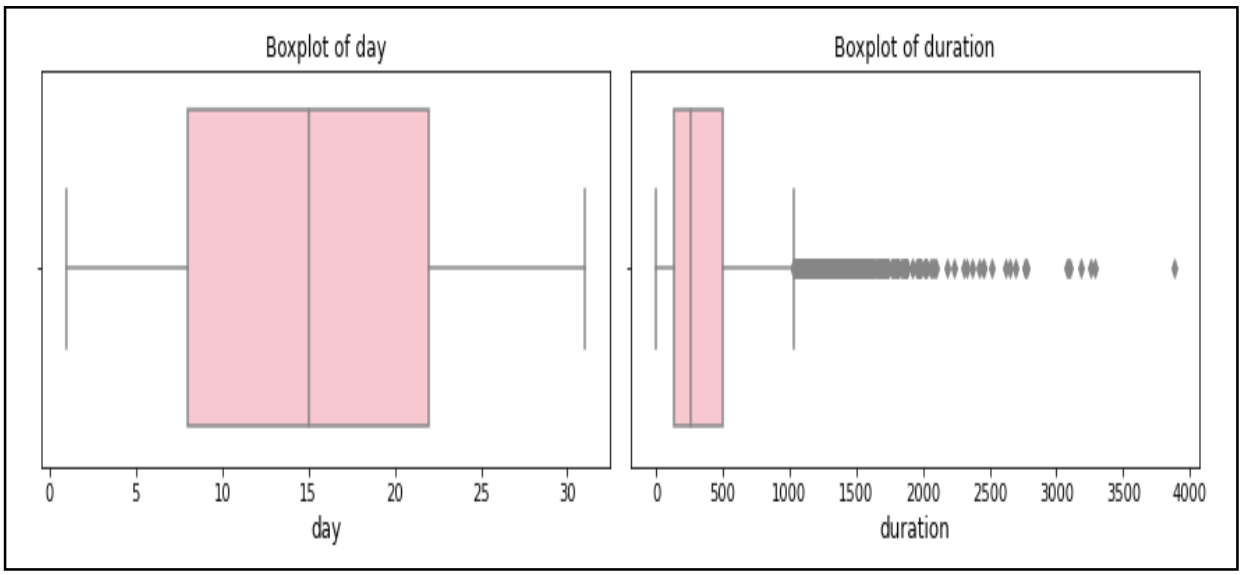
2) **Bivariate Data Analysis** :

If the data is small, we can detect the outlier by just looking at the data. But what if we have a huge data, how do we identify the outliers then? We need to use visualization and mathematical techniques.
The techniques of detecting outliers are:

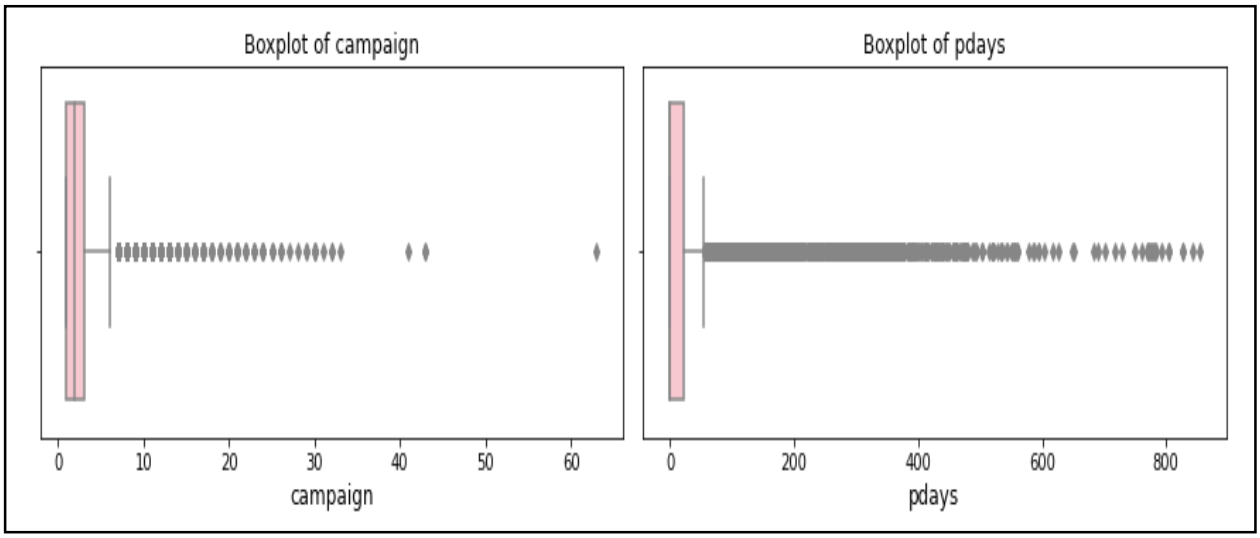
Boxplot:

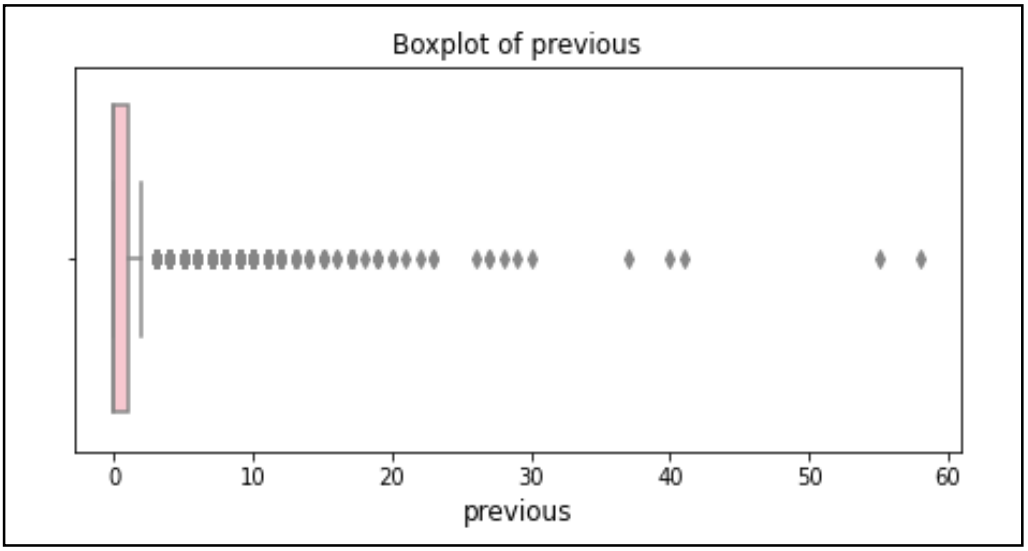


People those subscribed for deposit has higher bank balance.



Duration is matter most clearly seen from above plot higher the duration more the chances that person will subscribed for deposit.





Lower Campaign higher the chances that person will subscribed for deposit.

Handling Outliers:

1) Balance:
   We know Bank balance should not be negative there are some records with negative balance and very high number which acts as major outliers so we will drop this.

2) Duration:
   We drop the records with duration above 3000 which acts as major outliers.

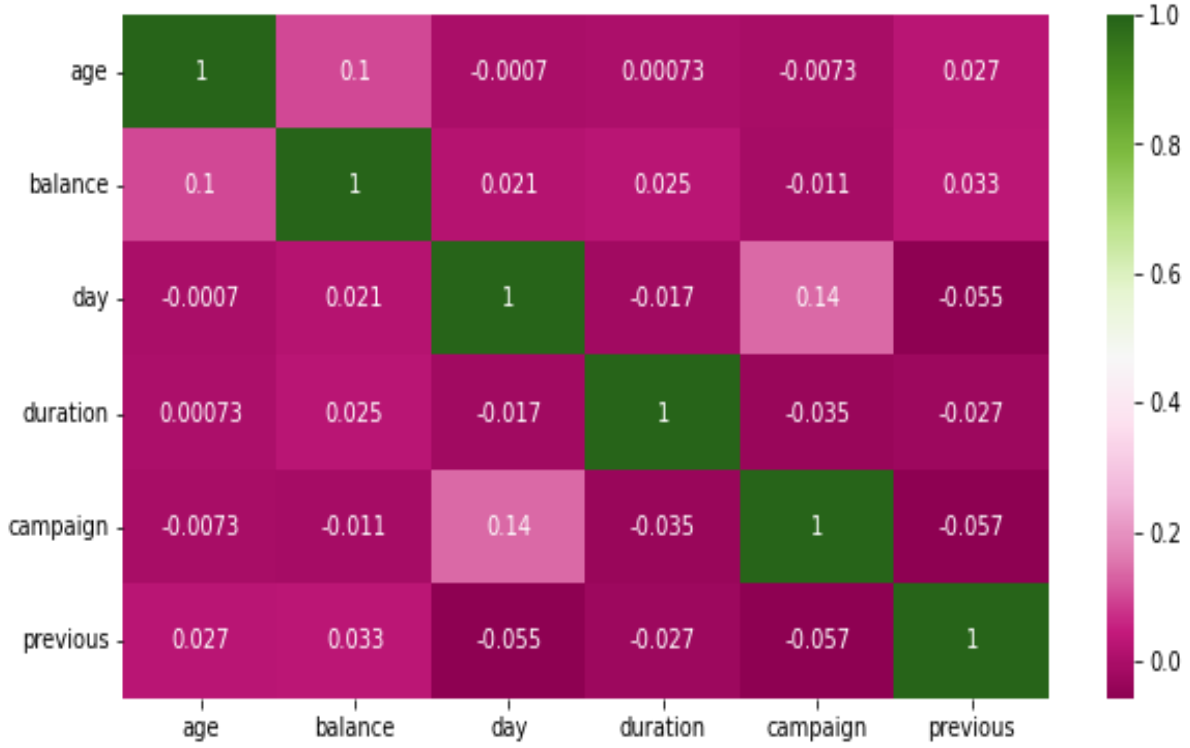
3) Campaign:
   We drop the records with campaign above 40 which acts as major outliers.

4) pdays:
   As seen above pdays -1 means these peoples are new or involving 1st time in campaign and they have no previous contact so previous = 0 and poutcomes is unknown for all -1 values of pdays so we will drop pdays columns.

5) Previous:
   We drop the outliers in previous features.

3) **Correlation Heatmap**:

The following map shows the Pearson correlation between different variables which is constructed using the Python software.



Interpretation:

From the above heatmap we can now see which variables are poorly correlated and which ones are strongly correlated.

a) Self-relation i.e. of a feature to itself is equal to 1 as expected.
b) From the above correlation matrix we can observe that the correlation between “day” and “campaign” is 0.14, which indicates they are positively correlated and the applicant is good.
c) Here the correlation between “previous” and “campaign” is -0.057, which indicates they are negatively correlated.

DATA PREPROCESSING

Split Inputs and Output features

- X – Independent Variables (Inputs features)
- Y – Dependent Variables (Output Variables)

Category Ratio for Train & Test dataset

Ratio for train dataset

0 0.51504
1 0.48496

Ratio for test dataset

0 0.515152
1 0.484848

Features Scaling:

- Feature scaling is a technique to normalize the independent feature present in the dataset in a fix range.
- As there are some algorithm such as Logistic regression, SVM, Random Forest, XG Boost that requires scaling data to maximize accuracy

Why Features Scaling?

- Some machine learning algorithms are sensitive, they work on distance formula and use gradient descent as an optimizer. Having values on the same scales helps gradient descent to reach global minima smoothly. For example, logistic regression, svm, knn, random forest, xg boost.

STATISTICAL TOOLS

❖ Logistic Regression:

In our proposed model we had used Logistic Regression which is one of the popular Machine Learning algorithms that comes under the Supervised Learning technique. It is applicable for categorical variable.

Logistic regression is a statistical analysis method to predict a binary outcome, such as yes or no, based on prior observations of a data set.

A logistic regression model predicts a dependent data variable by analyzing the relationship between one or more existing independent variables.

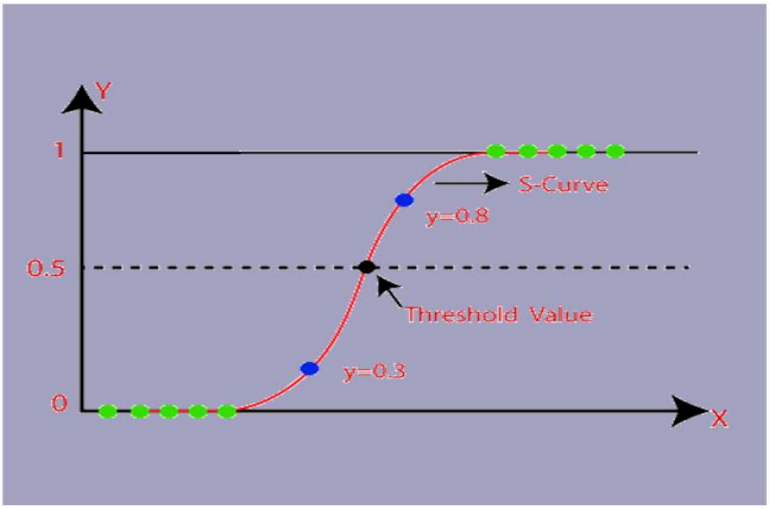
A logistic function or logistic curve is a common S-shaped curve (sigmoid curve) with equation, f(x) = $\frac{L}{1+e^{-k(x-x_0)}}$

Where, e = the natural logarithm base (also known as Euler's no.),

x_0= the x value of the sigmoid midpoint,

L = the curve's maximum value,

k = the logistic growth rate or steepness of the curve.

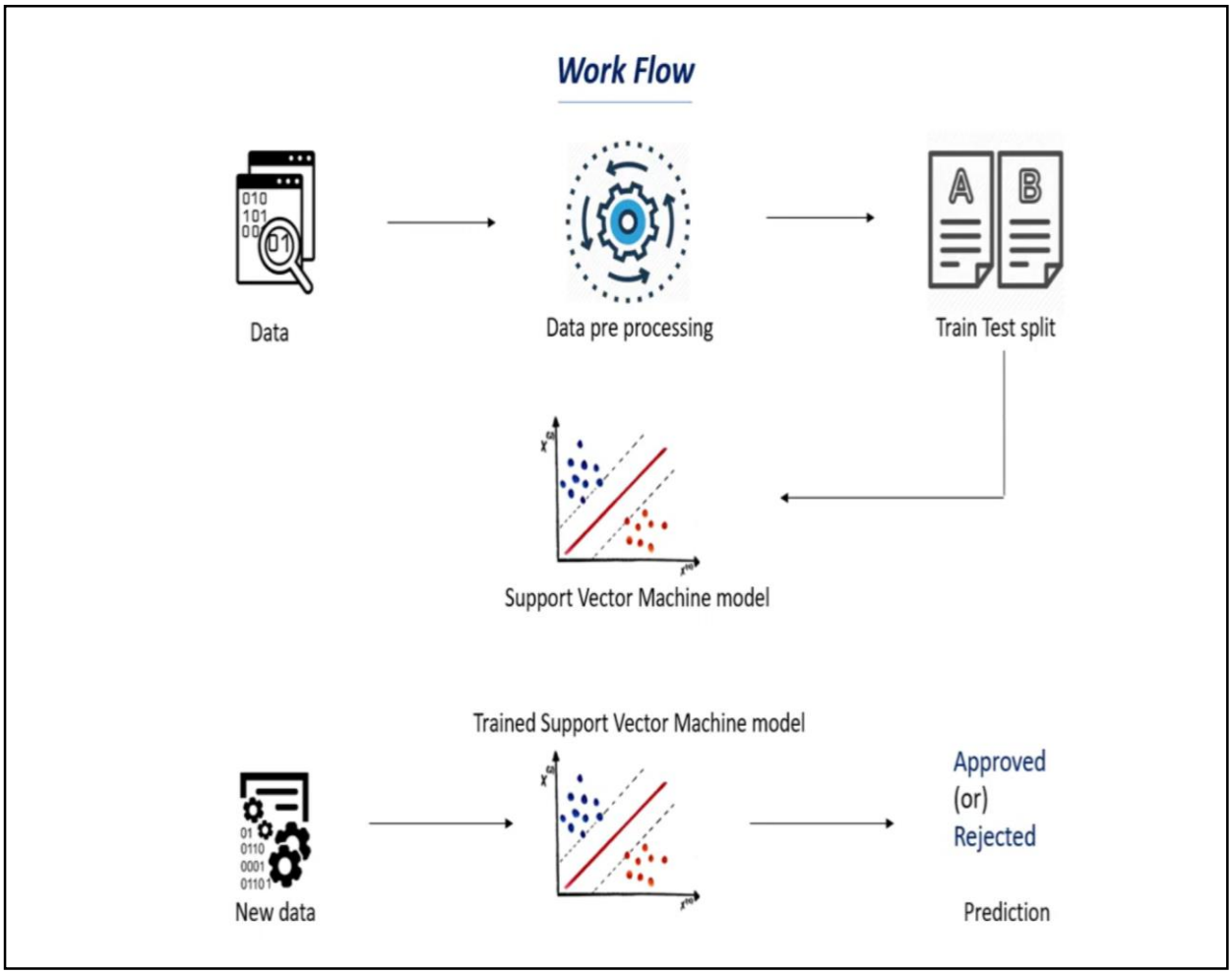


Standard logistic sigmoid function

i.e.

L = 1, k = 1, $x_0 = 0$

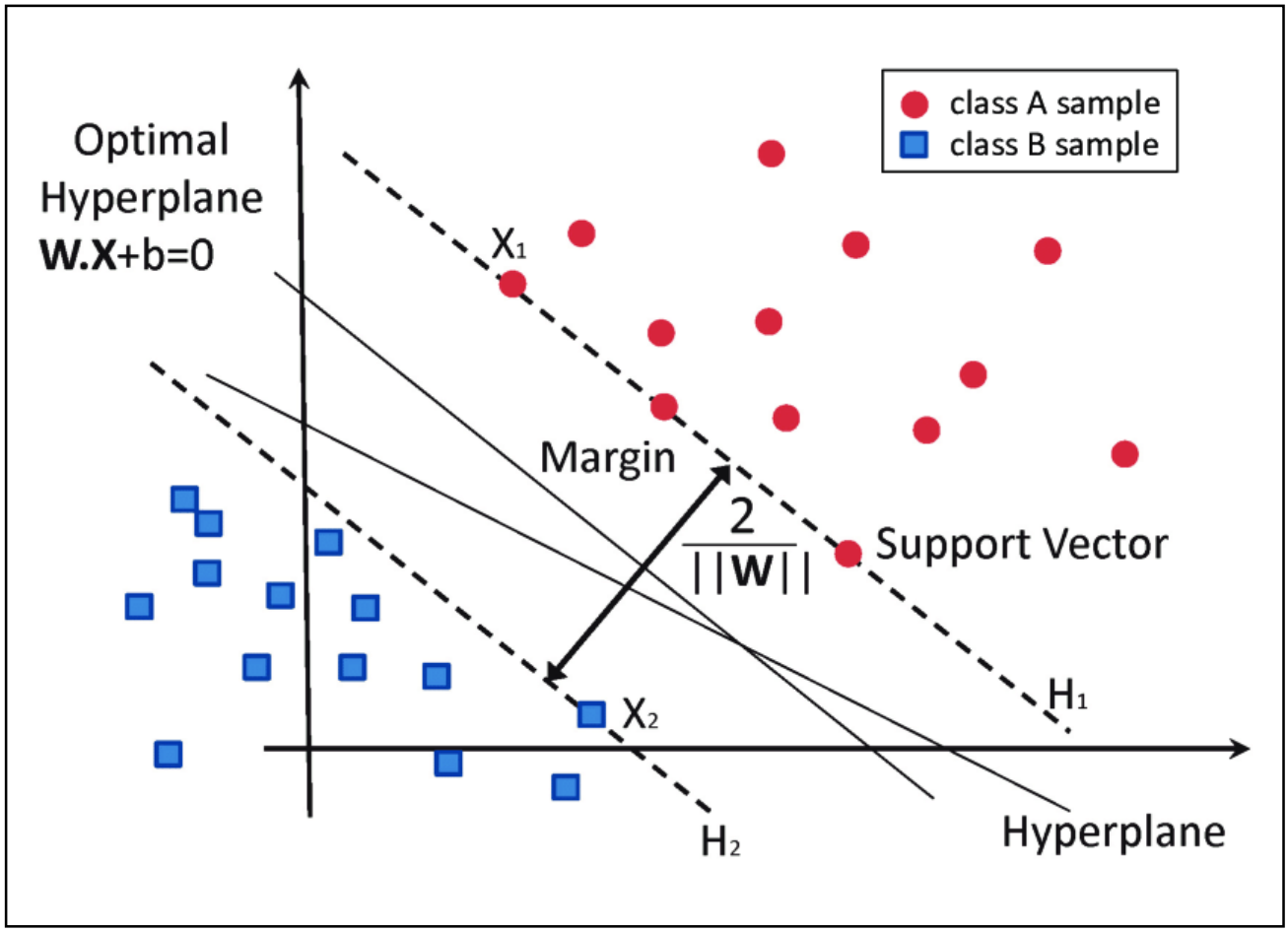
❖ Support Vector Machine:



Support Vector Machine or SVM is one of the most popular Supervised Learning algorithms, which is used for Classification as well as Regression problems. However, primarily, it is used for Classification problems in Machine Learning.

The goal of the SVM algorithm is to create the best line or decision boundary that can segregate n-dimensional space into classes so that we can easily put the new data point in the correct category in the future. This best decision boundary is called a hyperplane.

SVM chooses the extreme points/vectors that help in creating the hyperplane. These extreme cases are called as support vectors, and hence algorithm is termed as Support Vector Machine. Consider the below diagram in which there are two different categories that are classified using a decision boundary or hyperplan:



Why use SVM?

- It is effective in high dimensional spaces.
- It is effective in cases where number of dimensions is greater than the number of samples.
- It is a subset of training points in the decision function (called support vectors), so it is also memory efficient.

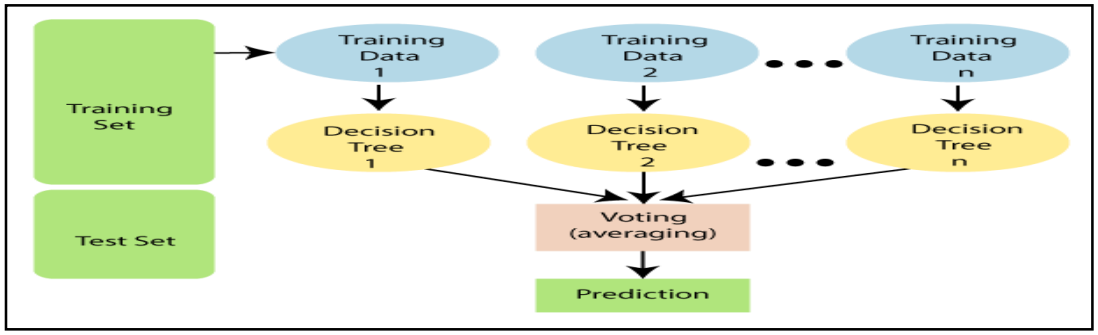
❖ Random Forest:

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. 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.

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.

The below diagram explains the working of the Random Forest algorithm

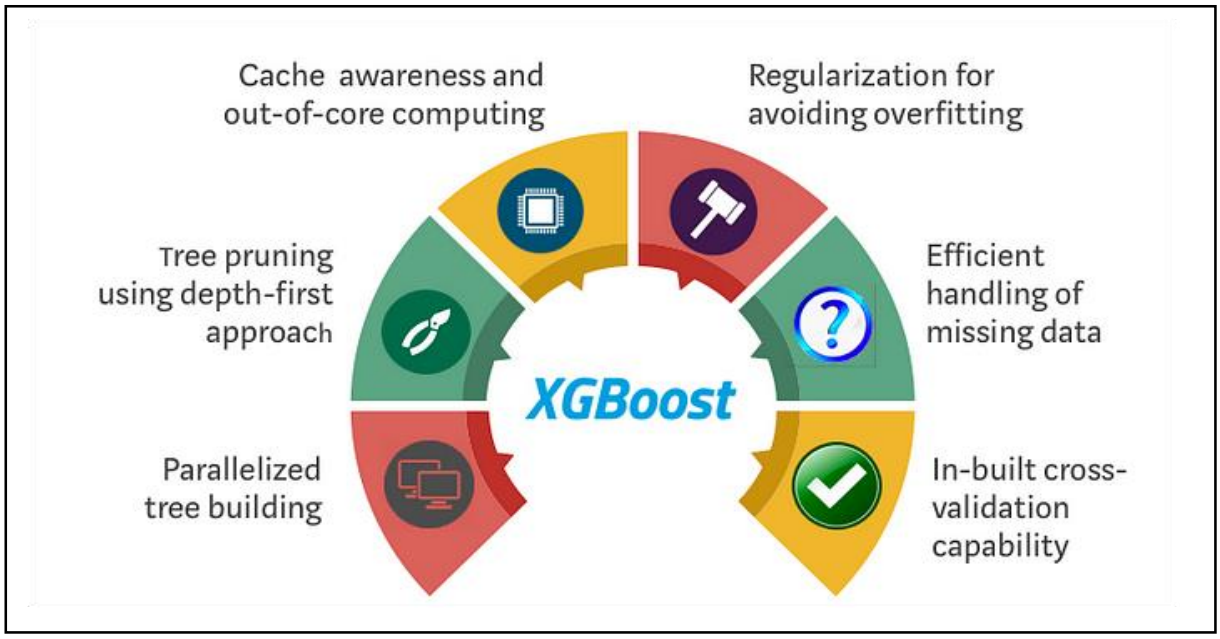


A random forest algorithm consists of many decision trees. The 'forest' generated by the random forest algorithm is trained through bagging or bootstrap aggregating. Bagging is an ensemble meta-algorithm that improves the accuracy of machine learning algorithms. The (random forest) algorithm establishes the outcome based on the predictions of the decision trees. It predicts by taking the average or mean of the output from various trees. Increasing the number of trees increases the precision of the outcome.

Why use 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.

❖ XG Boost:



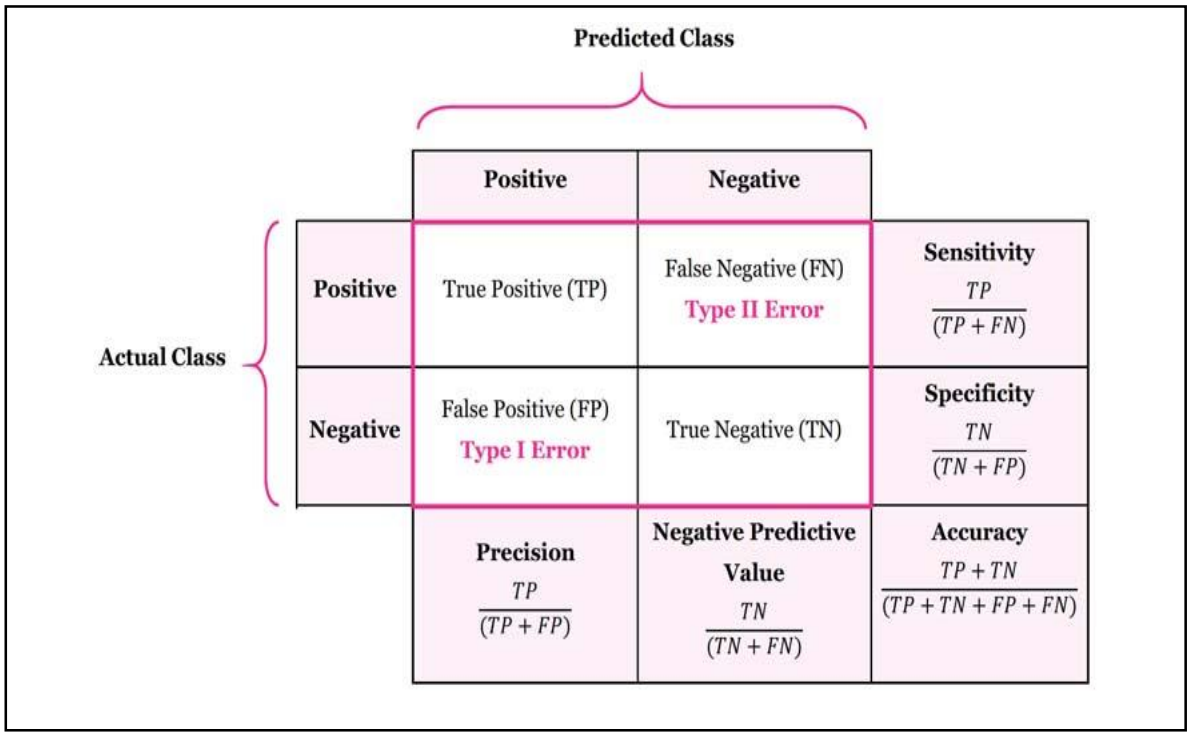
XG Boost(extreme Gardient Boosting) is a popular supervised learning algorithm used for regression and classification on large dataset. It uses sequentially – built shallow decision trees to provide accurate results and a Highly-Scalable training method that avoids over fitting.

XG Boost is used for supervised learning problems, where we use the training data(with multiple features)to predict a target variable.

Why use XG Boost?

- It take large number of observations in training data.
- It take Number features < number of observations in training data.
- It performs well when data has mixture numerical and categorical features or just numeric features.
- It take the model performance metrics are to be considered.

❖ Confusion Matrix:



A confusion matrix is a technique for summarizing the performance of a classification algorithm.

Calculating a confusion matrix can give you a better idea of what your classification model is getting right and what types of errors it is making.

Accuracy - Accuracy is the measure of total performance of classification model and it is simply a ratio of correctly predicted observation to the total observations.

Precision - Precision is the ratio of correctly predicted positive observations to the total predicted positive observations.

Recall (Sensitivity) - Recall is the ratio of correctly predicted positive observations to the all observations in actual class - yes.

Recall = TP/TP+FN

F1 score - F1 Score is the weighted average of Precision and Recall. Therefore, this score takes both false positives and false negatives into account.

F1 Score = 2(Recall Precision) / (Recall + Precision)

Why Recall is IMP?

Recall -- How much percentages from total actual positive class (yes) that are predicted correctly by machine

- Person who is actually going to subscribe for terms deposit should be always predicted correctly in order to have more profit
- Means **Recall score should be good**.
- FN should be less as possible
- So that bank will target mainly on those types of customers and increase their profit income
- There is no matter what machine predict for those types of customers which are not actually willing to subscribe for terms deposit because they are not target customers for company.
- But by improving servises quality company can focus those customer in features.

Recall Score :

- Here we will mainly focus on Recall Score
- Recall score of XG boost is very good i.e. **87%**

CODING

##EDA and Visualization

##Import Library

- import numpy as np
- import matplotlib.pyplot as plt
- import pandas as pd
- import seaborn as sns
- import warnings
- warnings.filterwarnings('ignore')

##Import Dataset

- df=pd.read_csv("/content/Bank Mrketing data.csv")
- df

##To check null value

- df.isnull().sum()

##To check duplicate value

- df.duplicated().sum()

##To check Data types

- df.dtypes

##Data balance

- plt.figure(figsize=(12,5))
- plt.subplot(1,2,1)
- sns.countplot(x='deposit',data=df,color='red',edgecolor="black")
- plt.title("Category of Yes and No")

- from matplotlib import colors
- ## add colors
- colors = ['gray','lightblue']
- labels =df['deposit'].value_counts(sort = True).index
- sizes = df['deposit'].value_counts(sort = True)
- plt.pie(sizes, labels=labels, colors=colors, autopct='%1.1f%%', shadow=True, startangle=270,)
- plt.title('Total yes and No categ',size = 12,color='black')
- plt.show()

Visualization of categorical features

Count plot

```
plt.figure(figsize=[4,5])
sns.countplot(x='marital', hue='deposit',color='pink',edgecolor="black",data=df)
plt.title("Countplot of marital by deposit")
plt.show()

plt.figure(figsize=[4,5])
sns.countplot(x='education', hue='deposit',color='pink',edgecolor="black",data=df)
plt.title("Countplot of educaion by deposit")
plt.show()

plt.figure(figsize=[4,5])
sns.countplot(x='default', hue='deposit',color='pink',edgecolor="black",data=df)
plt.title("Countplot of default by deposit")
plt.show()

plt.figure(figsize=[4,5])
sns.countplot(x='housing', hue='deposit',color='pink',edgecolor="black",data=df)
plt.title("Countplot of housing by deposit")
plt.show()

plt.figure(figsize=[4,5])
sns.countplot(x='loan', hue='deposit',color='pink',edgecolor="black",data=df)
plt.title("Countplot of loan by deposit")
plt.show()

plt.figure(figsize=[4,5])
sns.countplot(x='contact', hue='deposit',color='pink',edgecolor="black",data=df)
plt.title("Countplot of contact by deposit")
plt.show()

plt.figure(figsize=[8,5])
sns.countplot(x='month', hue='deposit',color='pink',edgecolor="black",data=df)
plt.title("Countplot of month by deposit")
plt.show()
```

- plt.figure(figsize=[4,5])
- sns.countplot(x='poutcome', hue='deposit',color='pink',edgecolor="black" ,data=df)
- plt.title("Countplot of poutcom by deposit")
- plt.show()
- plt.figure(figsize=[12,5])
- sns.countplot(x='job', hue='deposit',color='pink',edgecolor="black",data= df)
- plt.title("Countplot of job by deposit")
- plt.show()

##Visualization of Numerical features

##Box plot

- df_num = df[['age', 'balance']]
- col = ['age', 'balance']
- plt.figure(figsize=(12,12))
- for i,v in enumerate(col):
- print(i,v)
- plt.subplot(4,2,i+1)
- sns.boxplot(x=v,data=df_num,color='pink')
- plt.title("Boxplot of {}".format(v),size=12,color="Black")
- plt.xlabel("{}".format(v),size=12)
- plt.tight_layout()
- plt.show()

- df_num = df[['day', 'duration']]
- col = ['day', 'duration']
- plt.figure(figsize=(12,12))
- for i,v in enumerate(col):
- print(i,v)
- plt.subplot(4,2,i+1)
- sns.boxplot(x=v,data=df_num,color='pink')
- plt.title("Boxplot of {}".format(v),size=12,color="Black")
- plt.xlabel("{}".format(v),size=12)
- plt.tight_layout()
- plt.show()

- df_num = df[['campaign', 'pdays']]
- col = ['campaign', 'pdays']
- plt.figure(figsize=(12,12))
- for i,v in enumerate(col):
- print(i,v)

- plt.subplot(4,2,i+1)
- sns.boxplot(x=v,data=df_num,color='pink')
- plt.title("Boxplot of {}".format(v),size=12,color="Black")
- plt.xlabel("{}".format(v),size=12)
- plt.tight_layout()
- plt.show()

- df_num = df[['previous']]
- col = ['previous']
- plt.figure(figsize=(12,12))
- for i,v in enumerate(col):
- print(i,v)
- plt.subplot(4,2,i+1)
- sns.boxplot(x=v,data=df_num,color='pink')
- plt.title("Boxplot of {}".format(v),size=12,color="Black")
- plt.xlabel("{}".format(v),size=12)
- plt.tight_layout()
- plt.show()

##Handling Outliers

##1. Balance

- len(df[df['balance']<0])/len(df)
- df[(df['balance']>10000)|(df['balance']<0)]
  ##drops negative values of balance
- df.drop(df[(df['balance']>40000)|(df['balance']<0)].index,inplace=True,axis=0)

##2. Duration

- df[df['duration']>3000]
  ##Drop records with duration above 3000 which acts as major outliers
- df.drop(df[df['duration']>3000].index,inplace=True,axis=0)

##3. Campaign

- df[df['campaign']>40]
  ##Drop major outliers of campaign column
- df.drop(df[df['campaign']>30].index,axis=0,inplace=True)

##4. Pdays

- df[df['pdays']==-1]
- df[df['poutcome']=='unknown']
- df[df['previous']==0]
  ##Drop pdays columns from above conclusion
- df.drop("pdays",inplace=True,axis=1)

##5.previous

- df[df['previous']>30]
  ##Drop outliers in previous featutrs
- df.drop(df[df['previous']>30].index,axis=0,inplace=True)
- #df1 = df[df['balance']>0]
- plt.hist(x="campaign",data=df,color='red')
- plt.show()

##Correlation Heatmap

- #pearson's Correlation,which measures the strength of a linear relationship
- df_num = df[['age', 'balance', 'day', 'duration', 'campaign', 'previous']]
- plt.figure(figsize=(10,5))
- sns.heatmap(data=df_num.corr(), cmap="PiYG" ,annot=True)
- plt.show()

##Encoding Category

- #create dict for binary encoding
- dic = {"yes":1,"no":0}
- from sklearn.preprocessing import LabelEncoder
- dic = {"yes":1,"no":0}
- lst = ["deposit","loan","default","housing"]
- for i in lst:
- df[i] = df[i].map(dic)
- # Ordinal Encoding
- l=['month',"contact","poutcome"]
- for i in l:
- le=LabelEncoder()
- df[i]=le.fit_transform(df[i].values)
- df = pd.get_dummies(df, columns = ['job','marital','education'])
- df=df.reset_index()
- df.drop('index',axis=1,inplace=True)
- df

##Data Preprocessing

##Import Library

- from sklearn.pipeline import make_pipeline
- from sklearn.model_selection import StratifiedShuffleSplit, StratifiedKFo
  ld
- from sklearn.model_selection import cross_val_score
- from sklearn.model_selection import GridSearchCV

##Split inputs and output features

- X = df.drop('deposit',axis=1)
- Y = df['deposit']

- Y

- sss = StratifiedShuffleSplit(n_splits=1,test_size=0.3,random_state=1)
- for train_index,test_index in sss.split(X,Y):
- train_df = df.loc[train_index]
- test_df = df.loc[test_index]

##Category Ratio for Train & Test dataset

- print("Ratio for train dataset")
- print(train_df['deposit'].value_counts()/train_df.shape[0])
- print()
- print("ratio for test dataset")
- print(test_df['deposit'].value_counts()/test_df.shape[0])

##Train & Test dataset

- X_train = train_df.drop("deposit",axis=1)
- Y_train = train_df['deposit']

- X_test = test_df.drop("deposit",axis=1)
- Y_test = test_df['deposit']

- X_train
- X_test

- from sklearn.preprocessing import StandardScaler
- ss = StandardScaler()
- X_train_s = ss.fit_transform(X_train)
- X_test_s = ss.transform(X_test)

##Model Training and Model Evaluation

##Import all library need for model training

- #importing all the required ML packages
- #logistic regression
- from sklearn.linear_model import LogisticRegression
- #support vector Machine
- from sklearn import svm
- #Random Forest
- from sklearn.neighbors import KNeighborsClassifier
- #training and testing data split
- from sklearn.model_selection import train_test_split
- #accuracy measure
- from sklearn import metrics
- #for confusion matrix

- from sklearn.metrics import confusion_matrix

- from sklearn.metrics import classification_report
- from sklearn.preprocessing import StandardScaler

##Logistic Regression

- lr= LogisticRegression()
- lr.fit(X_train_s,Y_train)
- Y_pred_lr = lr.predict(X_test_s)

- print("Testing Accuracy of LogisticRegression : ",metrics.accuracy_score (Y_test,Y_pred_lr))
- #print("Accuracy of LogisticRegression",pipe_lr.score(X_test,Y_test))

- print("Training Accuracy of LogisticRegression : ",lr.score(X_train_s,Y_t rain))

Testing Accuracy of Logistic Regression : 0.7894
Training Accuracy of Logistic Regression : 0.7950

##SVM (Support Vector Machine)

- svm=svm.SVC(kernel='linear')
- svm.fit(X_train_s,Y_train)
- Y_pred_svm = svm.predict(X_test_s)

- print("Testing Accuracy of SVM : ",metrics.accuracy_score(Y_test,Y_pr ed_svm))
- #print("Accuracy of SVM",pipe_svm.score(X_test,Y_test))

- print("Training Accuracy of SVM : ",svm.score(X_train_s,Y_train))

Testing Accuracy of SVM : 0.79330
Training Accuracy of SVM : 0.7955

##Random Forest

- rf = RandomForestClassifier(n_estimators=200)
- rf.fit(X_train,Y_train)
- Y_pred_rf = rf.predict(X_test)
- print("Testing Accuracy of RF : ",metrics.accuracy_score(Y_test,Y_pred _rf))
- #print("Accuracy of of RF",pipe_rf.score(X_test,Y_test))

- print("Testing Accuracy of RF : ",rf.score(X_train,Y_train))

Testing Accuracy of RF : 0.8392
Testing Accuracy of RF : 1.0

XG Boost

- import xgboost
- xgb = xgboost.XGBClassifier(n_estimators=80, learning_rate=0.1, gamma=0, subsample=0.75,
  - colsample_bytree=1, max_depth=5)
- xgb.fit(X_train_s,Y_train.squeeze().values)

- #calculate and print scores for the model
- y_train_preds = xgb.predict(X_train_s)
- y_test_preds = xgb.predict(X_test_s)
- print(xgb.score(X_test_s,Y_test))
- print(xgb.score(X_train_s,Y_train))
- print('The Test accuracy of the XGB is',metrics.accuracy_score(Y_test,y_test_preds))
- print('The Train accuracy of the XGB is',metrics.accuracy_score(y_train_preds,Y_train))

The Test accuracy of the XG Boost is 0.8510
The Train accuracy of the XG Boost is 0.8834

- print(classification_report(Y_test,y_test_preds))
- print(confusion_matrix(Y_test,y_test_preds))

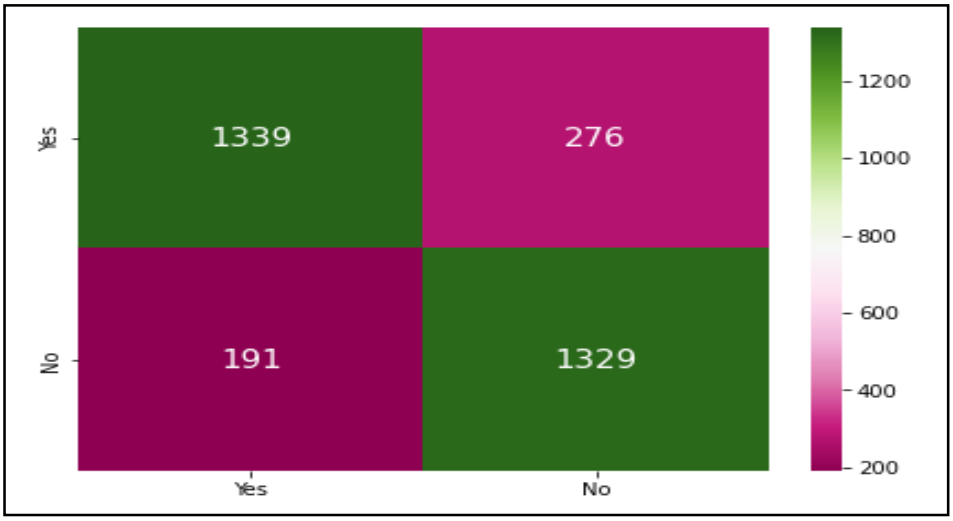
	precision	recall	f1-score	support
0	0.88	0.83	0.85	1615
1	0.83	0.87	0.85	1520
accuracy			0.85	3135
macro avg	0.85	0.85	0.85	3135
weighted avg	0.85	0.85	0.85	3135

[[1339 276]
[191 1329]]

Interpretation- Accuracy of XG Boost is very good without overfit.

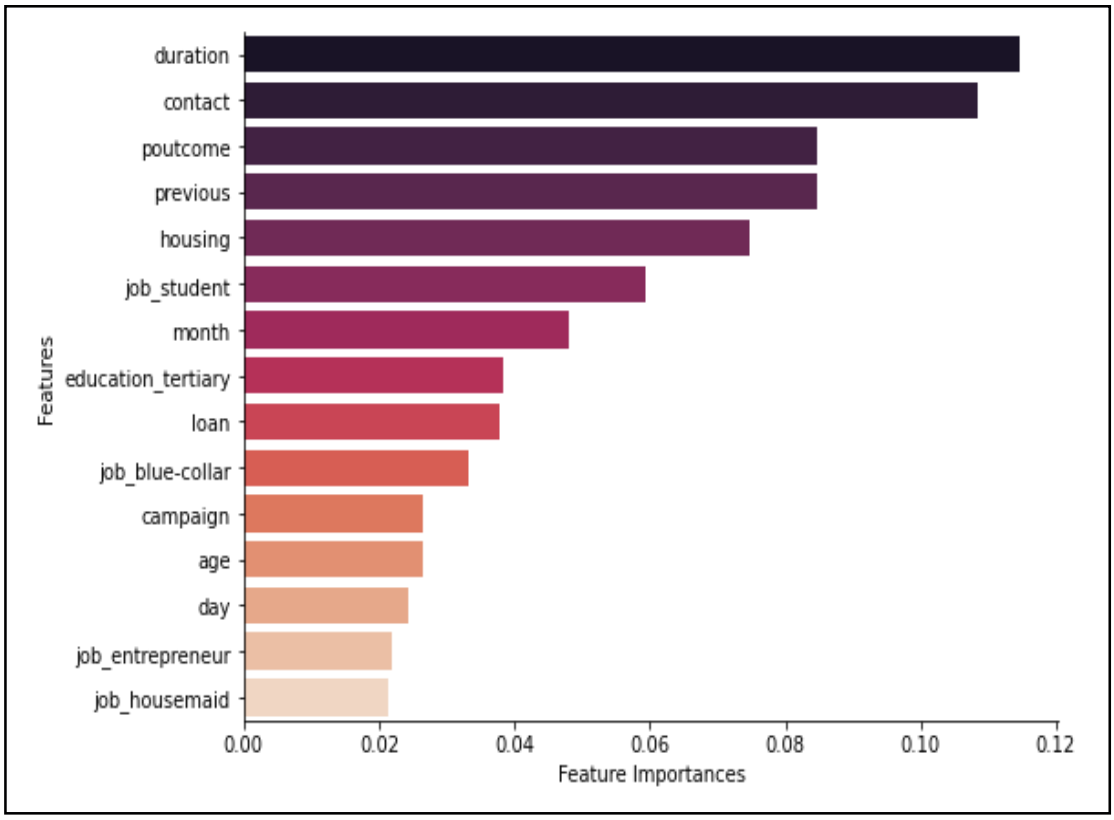
##Confusion matrix of XG Boost

- plt.figure(figsize=(7,5))
- sns.heatmap(confusion_matrix(Y_test,y_test_preds),annot=True,cmap="PiYG",
    - fmt="d",cbar=True,xticklabels=['Yes','No'],yticklabels=['Yes','No'],
    - annot_kws={"fontsize":15})
- plt.show()



##Feature importance by XG Boost

- # using random forest here to get feature importances
- plt.figure(figsize=(8,6))
- importances= xgb.feature_importances_
- feature_importances= pd.Series(importances, index=X_train.columns).sort_values(ascending=False)
- sns.barplot(x=feature_importances[:15], y=feature_importances.index[:15], palette="rocket")
- sns.despine()
- plt.xlabel("Feature Importances")
- plt.ylabel("Features")
- plt.show()



##Random Forest

- rf1 =RandomForestClassifier(random_state=0,n_estimators=200,max_features=25,max_depth=10,min_samples_leaf=50)
- rf1.fit(X_train_s,Y_train.squeeze().values)

- #calculate and print scores for the model for top 15 features
- y_train_preds = rf1.predict(X_train_s)
- y_test_preds = rf1.predict(X_test_s)
- print(rf1.score(X_test_s,Y_test))
- print(rf1.score(X_train_s,Y_train)

The Test accuracy of the Random Forest is 0.8213
The Train accuracy of the Random Forest is 0.8311

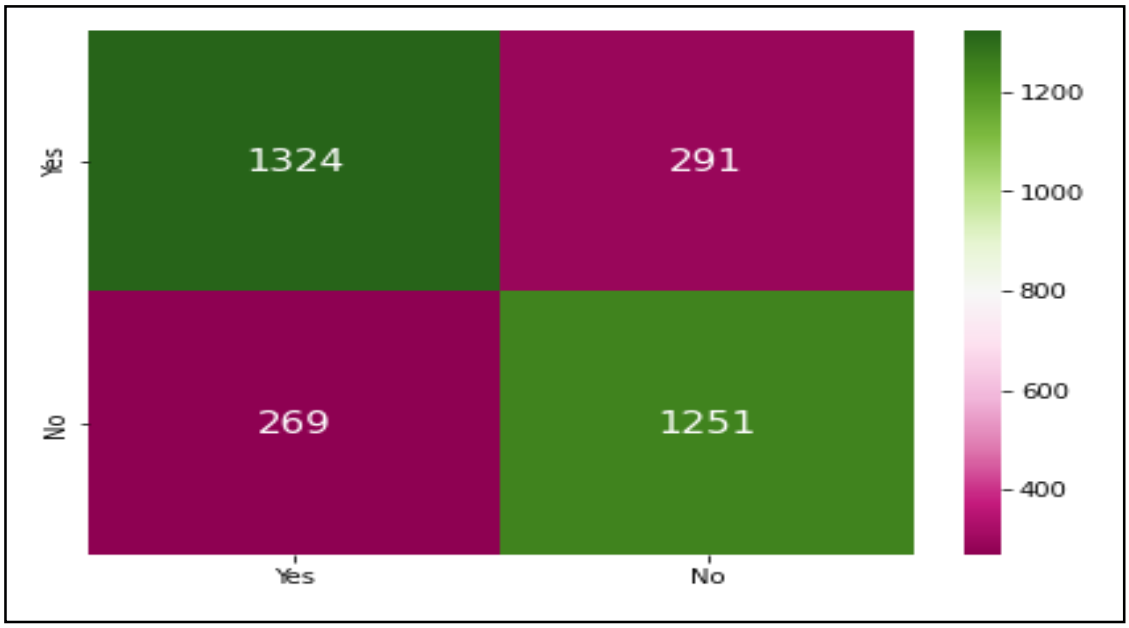
- print(classification_report(Y_test,y_test_preds))
- print(confusion_matrix(Y_test,y_test_preds))

	precision	recall	f1-score	support
0	0.83	0.82	0.83	1615
1	0.81	0.82	0.82	1520
accuracy			0.82	3135
macro avg	0.82	0.82	0.82	3135
weighted avg	0.82	0.82	0.82	3135

[[1324 291]
[269 1251]]

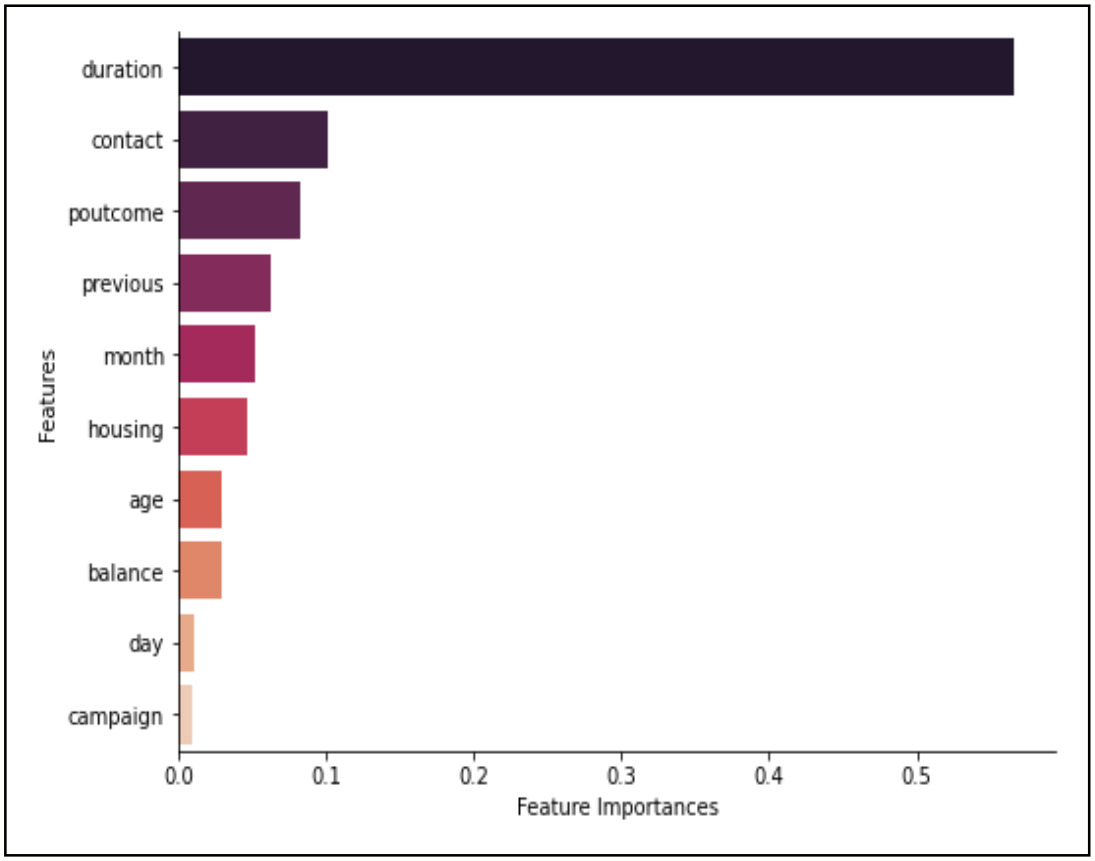
##Confusion matrix by Random Forest

- plt.figure(figsize=(7,5))
- sns.heatmap(confusion_matrix(Y_test,y_test_preds),annot=True,cmap="PiYG",
  - fmt="d",cbar=True,xticklabels=['Yes','No'],yticklabels=['Yes','No'],
  - annot_kws={"fontsize":15})
- plt.show()



##Feature Importances by Random Forest

- plt.figure(figsize=(8,6))
- importances= rf1.feature_importances_
- feature_importances= pd.Series(importances, index=X_train.columns).sort_values(ascending=False)
- sns.barplot(x=feature_importances[0:10], y=feature_importances.index[0:10], palette="rocket")
- sns.despine()
- plt.xlabel("Feature Importances")
- plt.ylabel("Features")
- plt.show()



Interpretation-

Common IMP features which are consider by XG Boost and Random Forest which are: Duration, contact, Poutcomes

##Cross Validation Accuracy

- ## XG Boost
- gs_svm_scores = cross_val_score(xgb, X=X_train_s, y=Y_train, cv=5,scoring='accuracy', n_jobs=-1)
- print('CV Mean Accuracy: {0:.1f}%'.format(np.mean(gs_svm_scores)*100))

CV Mean Accuracy: 84.5%

- ## Random Forest
- gs_svm_scores = cross_val_score(rf1, X=X_train_s, y=Y_train, cv=5,scoring='accuracy', n_jobs=-1)
- print('CV Mean Accuracy: {0:.1f}%'.format(np.mean(gs_svm_scores)*100))

CV Mean Accuracy: 81.7%

Interpretation-

- Cross validation accuracy of XG Boost: CV Mean Accuracy: 84.5%
- Cross validation accuracy of Random Forest: CV Mean Accuracy: 81.7%.
- Cross validation accuracy of XG boost is greater than random Forest.

##Model Deployment

- #Import Library
- from sklearn.base import BaseEstimator, TransformerMixin
- from sklearn.preprocessing import OneHotEncoder, MinMaxScaler,LabelEncoder,OrdinalEncoder
- from sklearn.impute import SimpleImputer
- from sklearn.pipeline import Pipeline
- from sklearn.compose import ColumnTransformer
- import xgboost
- #Dataframe
- df1
- #Correct Index
- df1=df1.reset_index()
- df1.drop('index',axis=1,inplace=True)
- #dataframe
- df1
- #Binary Encoding
- dic = {"yes":1,"no":0}
- lst = ["loan","default","housing"]
- for i in lst:
- df1[i] = df1[i].map(dic)
- #Encode of target variable(deposit)
- dic = {"yes":1,"no":0}
- df1["deposit"] = df1["deposit"].map(dic)
- df1
- #Ordinal Encoding
- contact_list = df1['contact'].unique().tolist()

```
poutcome_list = ['success','unknown', 'other','failure']
month_list = df1['month'].unique().tolist()
month_list = list(reversed(month_list))
print(contact_list)
print(poutcome_list)
print(month_list)
contact_label=list(range(0,3,1))
poutcome_label=list(range(0,4,1))
month_label=list(range(0,12,1))
dic_contact=dict(zip(contact_list,contact_label))
dic_poutcome=dict(zip(poutcome_list,poutcome_label))
dic_month=dict(zip(month_list,month_label))
print(dic_contact)
print(dic_poutcome)
print(dic_month)
#mapping dict with ordinal categ features
df1["contact"] = df1["contact"].map(dic_contact)
df1["poutcome"] = df1["poutcome"].map(dic_poutcome)
df1["month"] = df1["month"].map(dic_month)
#X and Y split
X = df1.drop('deposit',axis=1)
Y = df1['deposit']
#StratifiedShuffleSplit
sss = StratifiedShuffleSplit(n_splits=1,test_size=0.3,random_state=1)
for train_index,test_index in sss.split(X,Y):
train_df = df1.loc[train_index]
test_df = df1.loc[test_index]
#Train and Test dataset
X_train = train_df.drop("deposit",axis=1)
Y_train = train_df['deposit']

X_test = test_df.drop("deposit",axis=1)
Y_test = test_df['deposit']
X_train
## Tranformer
trf1 = ColumnTransformer([
('ohe',OneHotEncoder(sparse=False,handle_unknown='ignore'),[1,2,3])
],remainder='passthrough')
trf2 = ColumnTransformer(
transformers=[('scaler', StandardScaler(), [0,-1])],
remainder='passthrough'
trf3 = xgboost.XGBClassifier(n_estimators=80, learning_rate=0.1, gamm
a=0, subsample=0.75,
```

```
        colsample_bytree=1, max_depth=5)
pipe = Pipeline([
('trf1',trf1),
('trf2',trf2),
('trf3',trf3)
])
#stpes of pipes
pipe.steps
#Train model with 70% data set
pipe.fit(X_train,Y_train)
#Predict test data
y_pred = pipe.predict(X_test)
y_pred
#classification report
print(classification_report(Y_test,y_pred))
#accuacry score by model
from sklearn.metrics import accuracy_score
accuracy_score(Y_test,y_pred)
#cross validation using cross_val_score with CV =10
from sklearn.model_selection import cross_val_score
score=cross_val_score(pipe, X_train, Y_train, cv=10, scoring='accuracy')
print("mean accuarcy of XGB",score.mean())
#Predict Complete New Test data
test_input2 = np.array([25,'blue-collar','married','primary',0, 5000,0,1,1,9,11,921,10,2,2],dtype=object).reshape(1,15)
test_input2
df4 = pd.DataFrame(test_input2,columns=X_train.columns)
df4
pipe.predict(df4)
pipe.predict(test_input2)
#Save Model By dump function
#import pickle
#pickle.dump(rf,open('model.pkl','wb'))
```

CONCLUSION

From the study conducted, the results are impressive and convincing in terms of using a machine learning algorithm to decide on the marketing campaign of the bank. Almost all (age, balance, day, duration, campaign, previous) of the attributes contribute significantly to the building of a predictive model. Among the four classification approach used to model the data, the XG Boost model yielded the 88% best accuracy. This model is simple and easy to implement.

The bank marketing manager can identify the potential client by using the model if the client's information like education, housing loan, Personal loan, duration of the call, number of contacts performed during this campaign, previous outcomes, etc. is available. This will help in minimizing the cost to the bank by avoiding to call customers who are unlikely to subscribe the term deposit. They can run a more successful telemarketing campaign using this model.

REFERENCE

Websites –

https://www.kaggle.com/ (For data collection)

Medium blog

Research Paper –

1) http://www.columbia.edu/~jc4133/ADA-Project.pdf
2) Deposit subscribe prediction using data mining techniques based real marketing dataset

YouTube Videos –

1) https://youtu.be/LanIDRcm5x8
2) https://youtu.be/cGE5dyditUY
3) https://youtu.be/3pfq9Tn-YdA

Reference books –

1) http://myweb.sabanciuniv.edu/rdehkharghani/files/2016/02/The-Morgan-Kaufmann-Series-in-Data-Management-Systems-Jiawei-Han-Micheline-Kamber-Jian-Pei-Data-Mining.-Concepts-and-Techniques-3rd-Edition-Morgan-Kaufmann-2011.pdf

Software used –

1) Python
2) MS-Excel