Project Report

Bank Marketing Prediction

Submitted towards the partial fulfillment of the criteria for award of PGA by Imarticus

Submitted by:

M. Inbarasan

PGA - 20: Feb 2022

Abstract

In our day-to-day life we mostly deal with money, to hold our money in safe place, everyone is rushing towards bank. Bank plays a key role in controling money transactions and providing credits to its customer. As for the future concerns many people like to hold some money for their certain urgent needs and for short-term maturities. So the customers make their financial investment in many ways, but majority of investment is relies on Banking or Financial Institutions.

For Example: Term Deposit

A **Term Deposit** is a fixed-term investment that includes the deposit of money into an account at a financial institution.

Term deposit investments usually carry short-term maturities ranging from one month to a few years and will have varying levels of required minimum deposits. By this term deposits, bank will also earn customers by lending a credit to customer, and customer can have a short term maturities over a period of time.

In this project, I created a model that can find the customer who subscribed for term deposits, and who are not. By that model we can create a new attractive plans with high maturities can earn more customers and also increase the revenue of the Bank/Finanicial Institutions.

Acknowledgements

We are using this opportunity to express my gratitude to everyone who supported us throughout the course of this group project. We are thankful for their aspiring guidance, invaluably constructive criticism and friendly advice during the project work. I am sincerely grateful to them for sharing their truthful and illuminating views on a number of issues related to the project.

Further, we were fortunate to have Mr.Akilan as our mentor. He has readily shared his immense knowledge in data analytics and guide us in a manner that the outcome resulted in enhancing our data skills.

We wish to thank, all the faculties, as this project utilized knowledge gained from every course that formed the PGA program.

We certify that the work done by us for conceptualizing and completing this project is original and authentic.

Date**:* ***M.Inbarasan

***Place**: Chennai*

Certificate of Completion

I hereby certify that the project titled “Bank Marketing Prediction Model and Deployment” was undertaken and completed under supervision by Joyson.J from the batch of PGA - 20

Mentor: Mr.

Date:

Place: Chennai.

Table of Contents

Chapter 5 - Hyper Tuning

Hypertuning the best model

Finding the best Parameters

Metrics Analysis

Chapter 6 - Conclusion

Suggestions

Chapter 7 - Model Deployment

Importing Libraries

Input Parsing

Prediction

Conclusion

Reference

Chapter - 1
Introduction

Title & Objective of the Study:

This Dataset titled "Bank Marketing Prediction" is to predict the customer has subscribed to Term Deposit or Not. Here, by analysing the different set of attributes, we can understand the pattern of Customer who is having active Term Deposit. With the help of various Functionalities we can understand the pattern by Exploratory Data Analysis, Analysing Features and building a best model with higher accuracy and reliability can helps to predict the Future Customers.

Need of the Study:

Saving money is vital. It provides financial security and freedom and secures us in a financial emergency. By saving money, we can avoid debt, which relieves stress. However, despite knowing the importance of savings, we often lose sight of it and spend more of our money in the present.

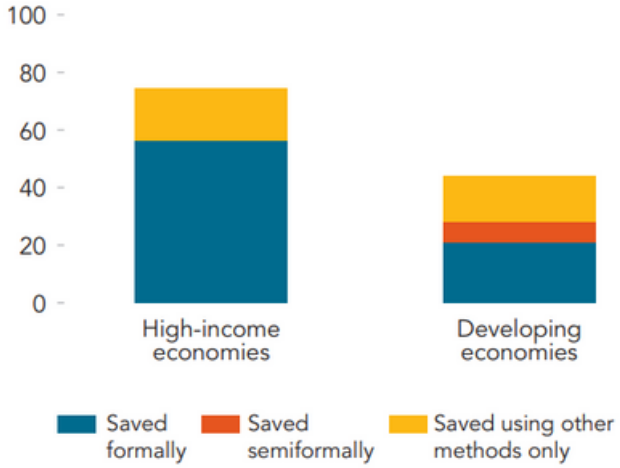
Do not save what is left after spending, but spend what is left after saving.
- ***Warren Buffett***

The price of anything is the amount of life you exchange for it.
– ***Henry David Thoreau***

In 2017, 48 percent of adults around the world reported having saved or set aside money in the past 12 months. In high-income economies 71 percent of adults reported having saved, while in developing economies 43 percent did so.

Globally, more than half of adults who save choose to do so at a financial institution

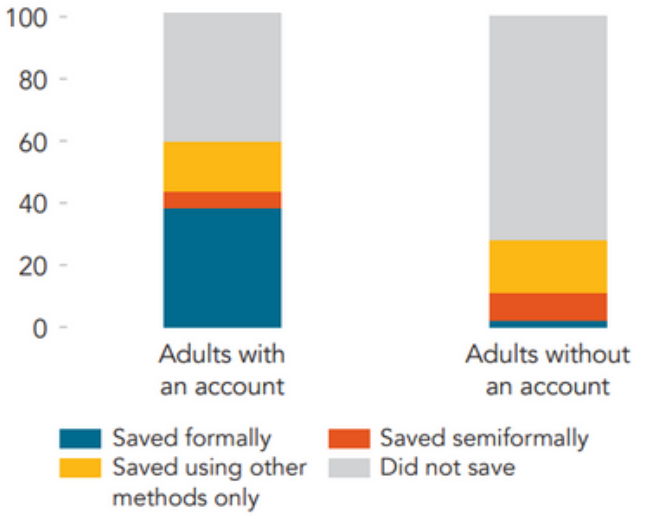
Adults saving any money in the past year (%), 2017



Source: Global Findex database.
Note: People may save in multiple ways, but categories are constructed to be mutually exclusive. *Saved formally* includes all adults who saved any money formally. *Saved semiformally* includes all adults who saved any money semiformally but not formally. Data on semiformal saving are not collected in most high-income economies.

Almost a third of unbanked adults save

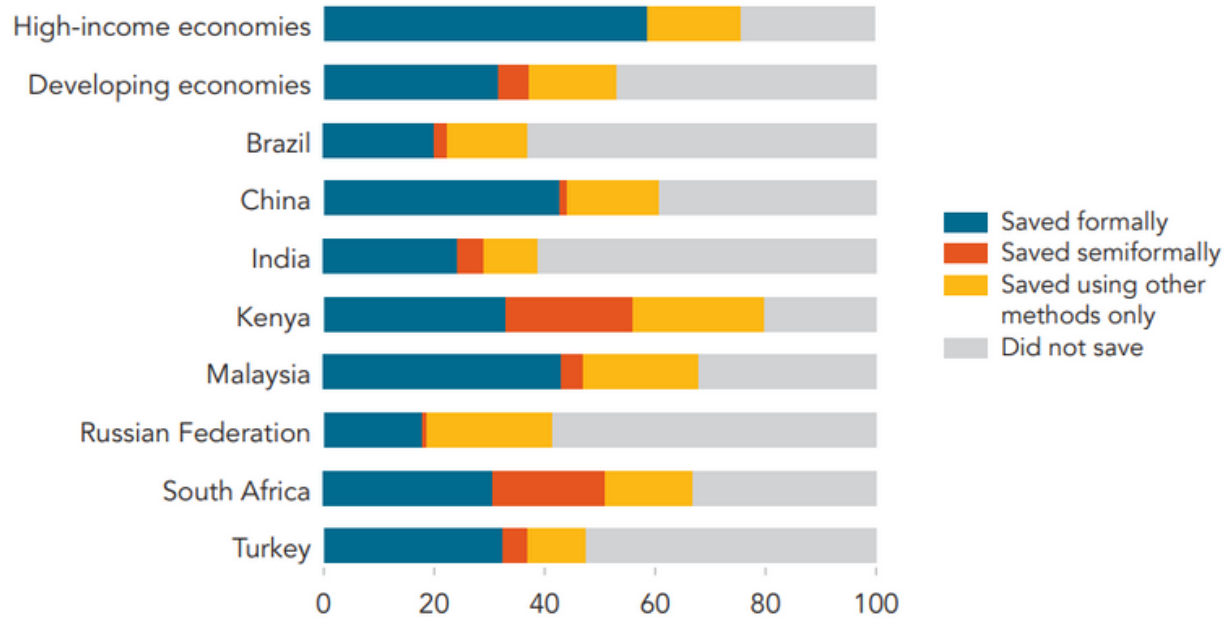
Adults by account ownership and savings behavior in the past year (%), 2017



Source: Global Findex database.
Note: People may save in multiple ways, but categories are constructed to be mutually exclusive. *Saved formally* includes all adults who saved any money formally. *Saved semiformally* includes all adults who saved any money semiformally but not formally.

Account owners do not necessarily use their account to save—or even save at all

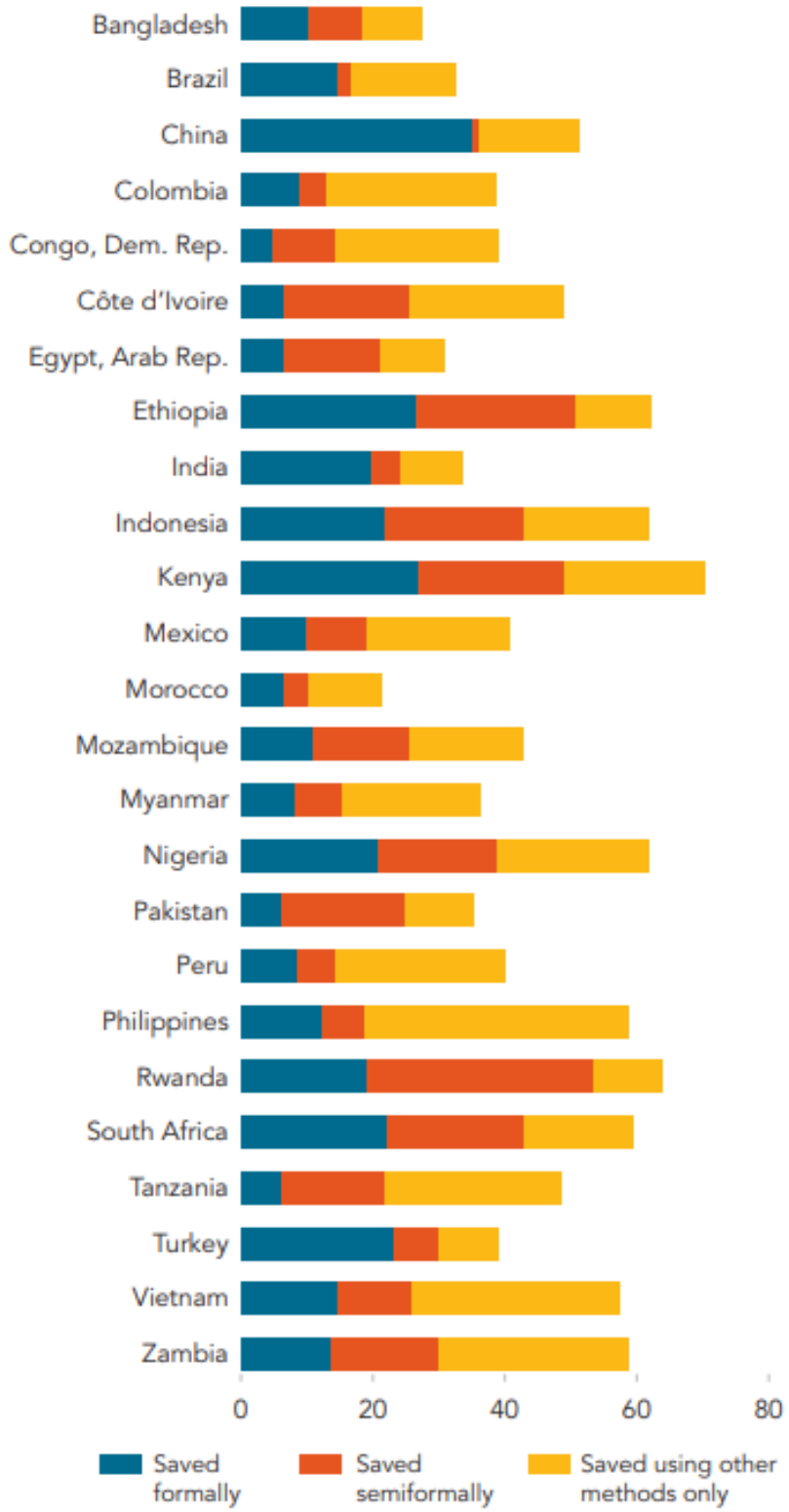
Adults with an account by savings behavior in the past year (%), 2017



Source: Global Findex database.
Note: Account owners may save in multiple ways, but categories are constructed to be mutually exclusive. *Saved formally* includes all account owners who saved any money formally. *Saved semiformally* includes all account owners who saved any money semiformally but not formally. Data on semiformal saving are not collected in most high-income economies. In all individual economies shown, about 70 percent or more of adults have an account.

Savings behavior varies widely across developing economies

Adults saving any money in the past year (%), 2017



Source: Global Findex database.
Note: People may save in multiple ways, but categories are constructed to be mutually exclusive. *Saved formally* includes all adults who saved any money formally. *Saved semiformally* includes all adults who saved any money semiformally but not formally.

Source: World : Global Findex Report - 2017
https://globalfindex.worldbank.org/

Saving money is one of the essential aspects of building wealth and having a secure financial future. Saving money gives you a way out from uncertainties of life and provides you with an opportunity to enjoy a quality life. Putting aside a sum of money in a systematic manner can help you steer out of many hurdles and obstacles in life. It can support you in your hour of need and ensure that your family has something to fall back on in case of an unfortunate event. There are many reasons to save and several ways to save with ease. Savings is crucial for everyone, regardless of their earnings, spending and life stage.

- It offers peace of mind: Knowing that you have a certain amount accumulated for times of your need, gives you the peace of mind. You can lead a stress-free life with the knowledge that you will not have to struggle if things take an unexpected route
- It gives you a better future: Your savings can be the answer to a number of your goals. You can buy a house, accumulate funds for your retirement, or purchase a vehicle. You can secure your future, indulge in the best of things that life has to offer and live a very fulfilling life
- It provides for your children’s education: With a considerable amount of savings, you can fuel your children’s dreams and pay for the best schools and colleges across the world
- You can plan your short term goals: Savings are not just aimed at the long term. You can also benefit from savings in the short term. A lot of people save for a few months and then travel
- It gives your family security in case of an unfortunate event: By saving in a disciplined manner, you can make sure that your family is well-provided for. In unfortunate times, your savings can act as a cushion for your loved ones and help them overcome any financial difficulty

We have various way to save our money, like Bonds, Share Market, Equity, Immovable Assets. The most important way to save and provide a economical stability for country and citizens is Term Deposit. Institutional Bank's main objective is to Accept Deposit and Lend Credit to the customers whom depends on the Bank.

So to acheive economic stability and money circulation, Bank has to attract more customers with active term deposit, because the term deposit of individual customer is thrown into the money pool. By lending credit from the money pool of the bank, other person will pay the debt for the sum of money what he got from the bank as credit, then the credit is shared over the people who had deposited a sum of amount in bank for a period of time. Hence over a period of time the amount is matured and repayed to the customer with attractive benefits.

By Segmenting the customers with consideration of trend what we so far analysed, can get a potential customer. So, the Banking Institution can provide attractive plans and benefits to the customer to have more active Term Deposits.

Data Description:

Input variables:

- bank client data:

1 - age (numeric)
2 -job : type of job (categorical: "admin.","unknown","unemployed","management","housemaid","entrepreneur", "student","blue-collar","self-employed","retired","technician","services")
3 - marital : marital status (categorical: "married","divorced","single"; note: "divorced" means divorced or widowed)
4 - education (categorical: "unknown","secondary","primary","tertiary")
5 - default: has credit in default? (binary: "yes","no")
6 - balance: average yearly balance, in euros (numeric)
7 - housing: has housing loan? (binary: "yes","no")
8 - loan: has personal loan? (binary: "yes","no")

- related with the last contact of the current campaign:

9 - contact: contact communication type (categorical: "unknown","telephone","cellular")
10 - day: last contact day of the month (numeric)
11 - month: last contact month of year (categorical: "jan", "feb", "mar", ..., "nov", "dec")
12 - duration: last contact duration, in seconds (numeric)

- other attributes:

13 - campaign: number of contacts performed during this campaign and for this client (numeric, includes last contact)
14 - pdays: number of days that passed by after the client was last contacted from a previous campaign (numeric, -1 means client was not previously contacted)
15 - previous: number of contacts performed before this campaign and for this client (numeric)
16 - poutcome: outcome of the previous marketing campaign (categorical: "unknown","other","failure","success")

- output variable (desired target):

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

The concerned Dataset is having 17 distinct features with 8 Numerical and 9 Categorical Columns. So lots of Study and exploration of data is needed to get a perfect model to predict Potential Customers. This Data have Outliers, but here the Outlier tends to be hold most important customers, so those aren't treated.

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 45211 entries, 0 to 45210
Data columns (total 17 columns):
 #   Column                Non-Null Count  Dtype
---  ------                --------------  -----
 0   Age                   45211 non-null  int64
 1   Job                   45211 non-null  object
 2   Marital Status        45211 non-null  object
 3   Education             45211 non-null  object
 4   Credit                45211 non-null  object
 5   Balance (euros)       45211 non-null  int64
 6   Housing Loan          45211 non-null  object
 7   Personal Loan         45211 non-null  object
 8   Contact               45211 non-null  object
 9   Last Contact Day      45211 non-null  int64
 10  Last Contact Month    45211 non-null  object
 11  Last Contact Duration 45211 non-null  int64
 12  Campaign              45211 non-null  int64
 13  Pdays                 45211 non-null  int64
 14  Previous              45211 non-null  int64
 15  Poutcome              45211 non-null  object
 16  Subscription          45211 non-null  int64
dtypes: int64(8), object(9)
memory usage: 5.9+ MB
Shape of The dataset:  (45211, 17)
Total Size of the dataset:  768587
Dimensions of dataset:  2
```

Data Source:

https://www.kaggle.com/aakashverma8900/portuguese-bankmarketing
(Source: Kaggle.com)

Tools & Technique:

Tools: Jupyter Notebook, VSCode.

Technique: All Machine Learning Models, Imbalancing Technique (SMOTE).

Chapter - 2
Data Exploration and Analysing

Data View:

The process includes importing data with required Libraries and understanding the basic properties of data like data shape, data types and Data description.

```
# Primary Libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import warnings

# Preprocessing Libraries
from sklearn.preprocessing import LabelEncoder
from sklearn.preprocessing import MinMaxScaler
from sklearn.model_selection import cross_val_score, train_test_split
from imblearn.over_sampling import SMOTE

# Machine Learning Libraries
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.naive_bayes import GaussianNB
from sklearn.svm import SVC
from sklearn.ensemble import AdaBoostClassifier
from sklearn.ensemble import GradientBoostingClassifier
from sklearn.neighbors import KNeighborsClassifier

# Evaluation Metrics
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix, cohen_kappa_score, recall_score
from sklearn.metrics import auc, roc_curve

warnings.filterwarnings ('ignore')
```

```
data = pd.read_csv('Bank Marketing.csv')
data.head()
```

Important Libraries:

Pandas - Data Manipulation Libraries
Numpy - Numerical Computation Libraries
Matplotlib & Seaborn - Data Visualization Libraries
Sklearn - Machine Learning Libraries

Data Dimension & Feature Analysis:

```
Shape of The dataset:  (45211, 17)
Total Size of the dataset:  768587
Dimensions of dataset:  2

Exporting
Descriptive Statistical Report for Numerical Columns

Numerical Columns:  ['Age' 'Balance (euros)' 'Last Contact Day' 'Last Contact Duratio
 'Campaign' 'Pdays' 'Previous' 'Subscription']
Number of Numerical Columns in the dataset:  8

Categorical Columns:  ['Job' 'Marital Status' 'Education' 'Credit' 'Housing Loan'
 'Personal Loan' 'Contact' 'Last Contact Month' 'Poutcome']
Number of Categorical Columns in the dataset:  9
```

The Dataset is having 17 distinct feature attributes and contains of 45211 observation which is from real world analysis.

From the Data's feature analysis, we have two different segements like Numerical and Categorical column. In this dataset, Categorical column is more dominating than Numerical column.

Data Analysis:

The process includes analysing all the types of variables and missing values, Cardinality of Categorical Values, Analysing various attributes, and determine how those attributes are having impact to produce conserned output. Even Outliers are found for all variables and noted.

Analysing Missing Values:

```
Percentage of Null Values in the dataset:
                    Percentage
Age                        0.0
Job                        0.0
Marital Status             0.0
Education                  0.0
Credit                     0.0
Balance (euros)            0.0
Housing Loan               0.0
Personal Loan              0.0
Contact                    0.0
Last Contact Day           0.0
Last Contact Month         0.0
Last Contact Duration      0.0
Campaign                   0.0
Pdays                      0.0
Previous                   0.0
Poutcome                   0.0
Subscription               0.0
```

The process involves in calculation of Percentage of Missing Values in the given dataset. By analysing the Output, In the dataset there is no Missing Values to treat. Because banking Institution has to maintain keen resources to perform complex functionalities and investing in good technology to perform vast transactions in a period of time.

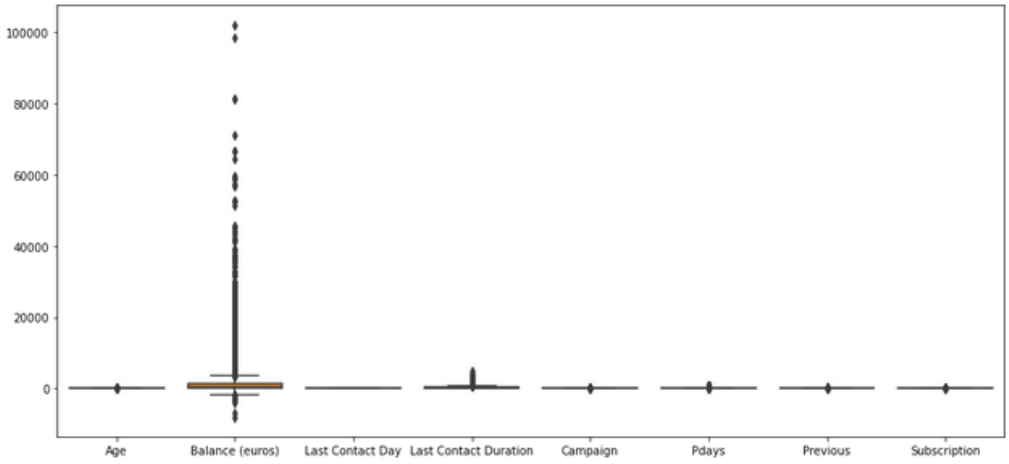
Cardinality Check:

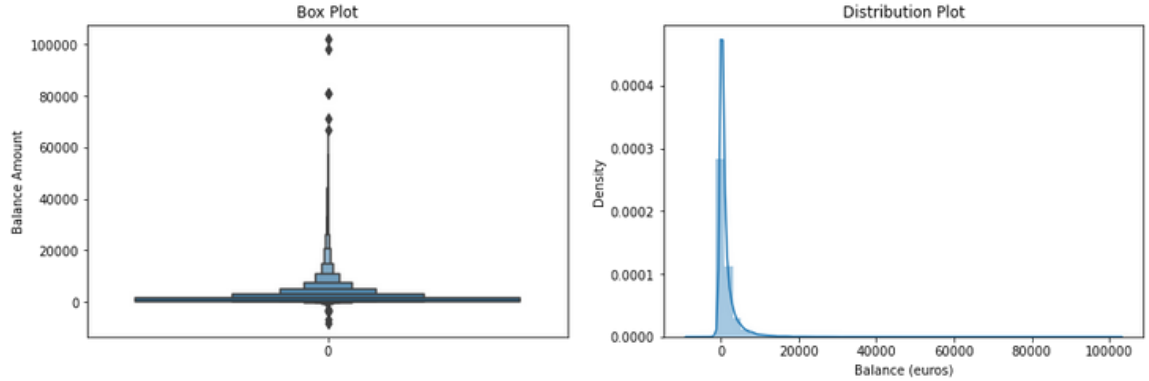
Cardinality:

	Counts
Job	12
Marital Status	3
Education	4
Credit	2
Housing Loan	2
Personal Loan	2
Contact	3
Last Contact Month	12
Poutcome	4

"Cardinality" refers to the number of possible values that a feature can assume. Dealing with high cardinality turned out to be one of the most interesting parts of the challenge. We can encounter Cardinality with approaches like Label Encoding, One Hot Encoding, Binary Encoding, Manual Encoding.

Outlier Analysis:





In this Dataset, there is Outlier in Balance(Euro's). But here for this dataset, those outlier is directly pointing towards the potential customers. Majority of outliers were lies above the upper boundary which means the customers with large sum of amount as balance in respective bank. So, the outliers also plays an important role in here.

Analysing Features:

Analysing individual features with the dependent features will reproduce the importance of the particular feature variable to predict the exact output. By condisering various constrains like how various Sector people having balance in their account, jobs which influence subscription, number of male and female who have subscribed to Term Deposit, etc.,

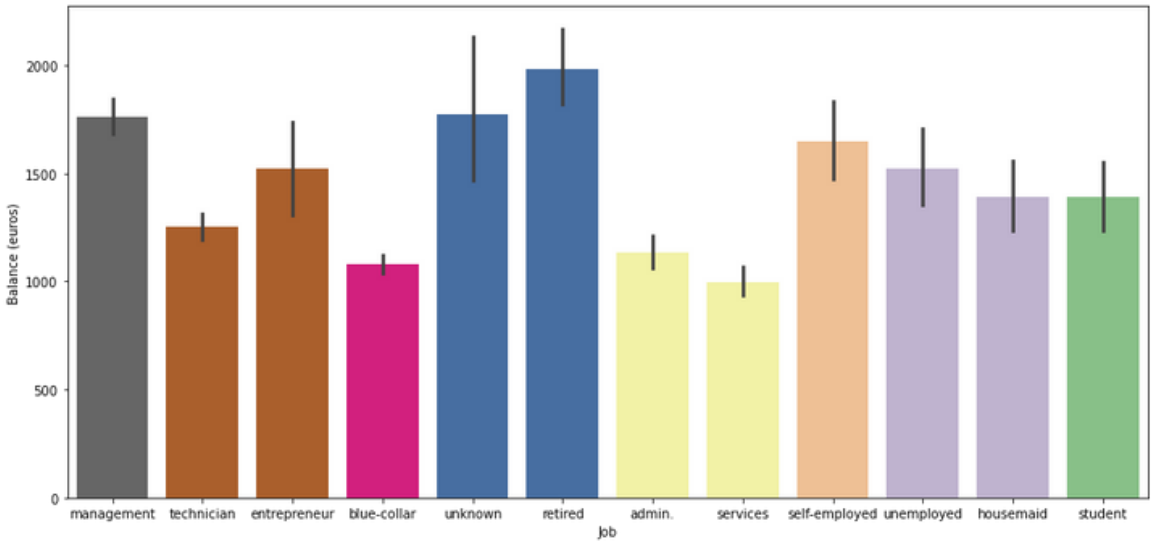


Fig 1: Analysis of Money Pooling according to Job

Retired People is holding average balance of 1800+ euros, because they may have Special Benefit amounts, Savings. After them majority who holds large amount of money as their bank balance is Management and Unknown Category.

Person who working in Administration Sector, Serive and Blue - Collar Jobs, is having holding very less amount of Balance in Bank.

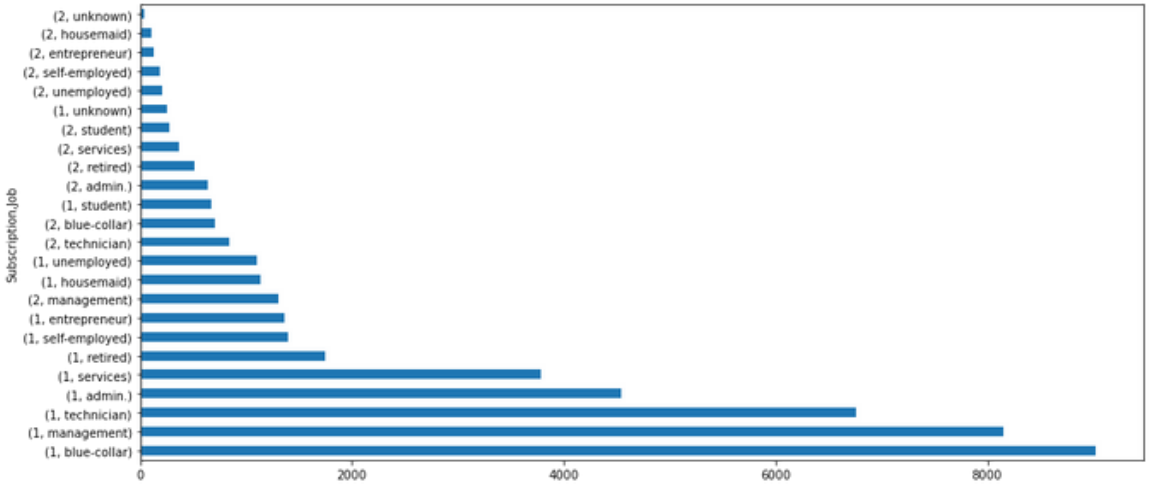


Fig 2: Trend of Sectors and Subscription

The Trend of Sector and People in particular sector is shown and how much they influence the subscription for Term Deposit is also shown above. Here, the people who is working in Blue Collar Jobs hasn't subscribed to Term Deposit. And the trend shows clear picture of person who earn less is not having Term Deposit

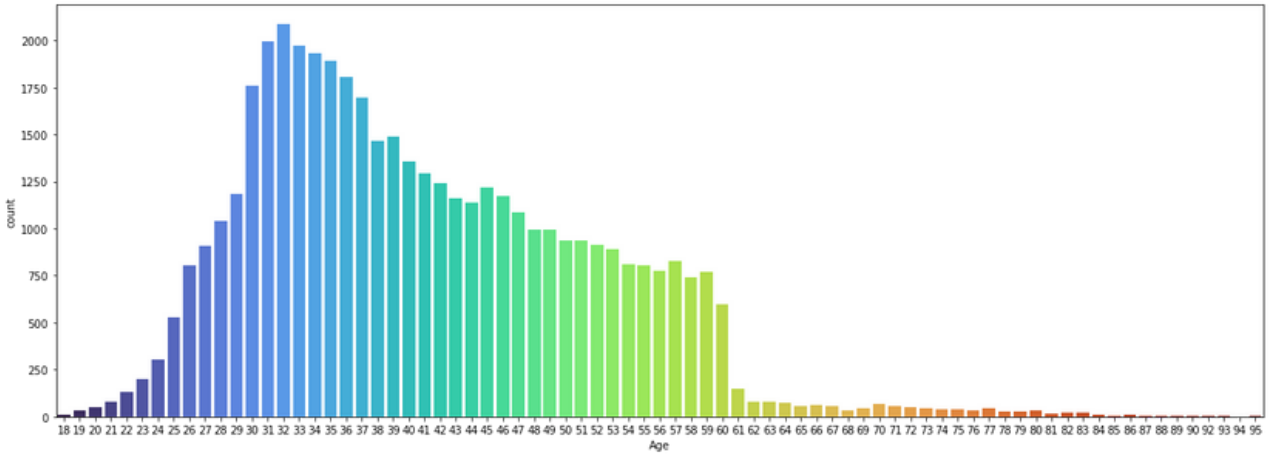


Fig 3: Age Distribution

We can see the trend that people who is active with bank is working in certain sectors, because they have to rely on bank for their monthly credit, and the trend goes on downphase when the person age is above 60.

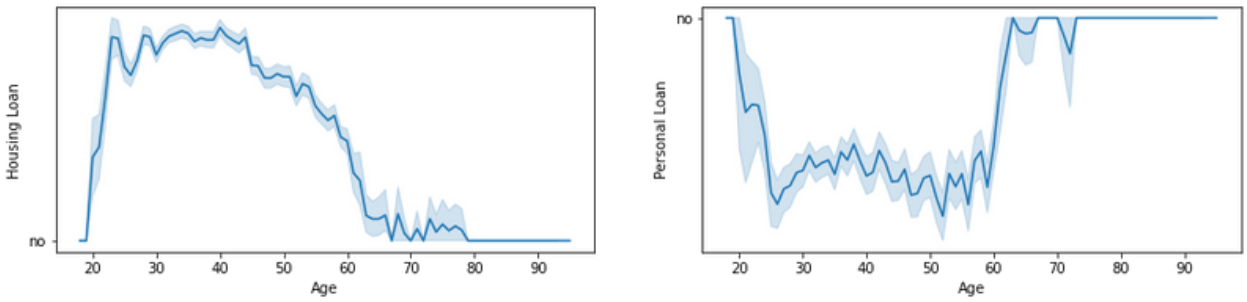


Fig 4: Trend of Housing Loan and Personal Loan according to Age

From the Graphs, we can analyse the age wise distribution according to Housing and Personal Loan.

Barely we can analyse that people who aged above 60 are less probability to have personal and Housing loan. But the trend is peaked over age of 20 - 55, why because majority of people is lies in working age, so they need to have better economic background and they also tends to spend more in Housing, Vechicles, etc.,

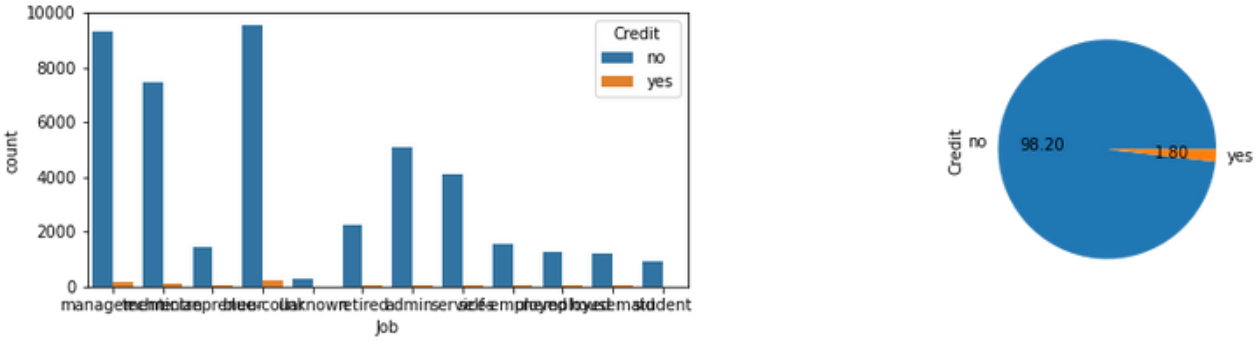


Fig 5: Trend of Job and Credit

From the Graph, Very less number of people having credit, By analysing the Number of Customer having frequent Credit is very less.

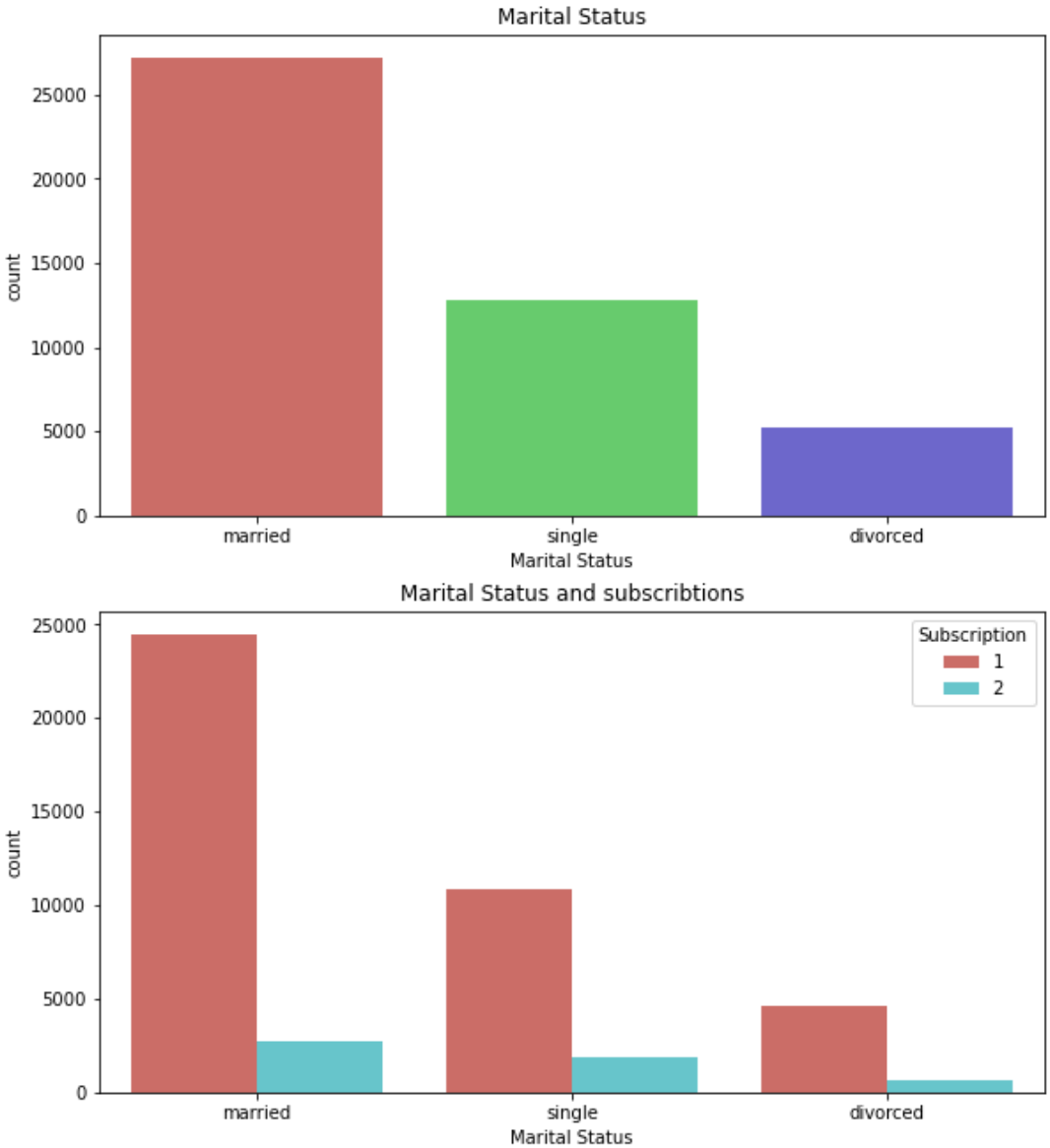


Fig 6: Marital Status and Subscription

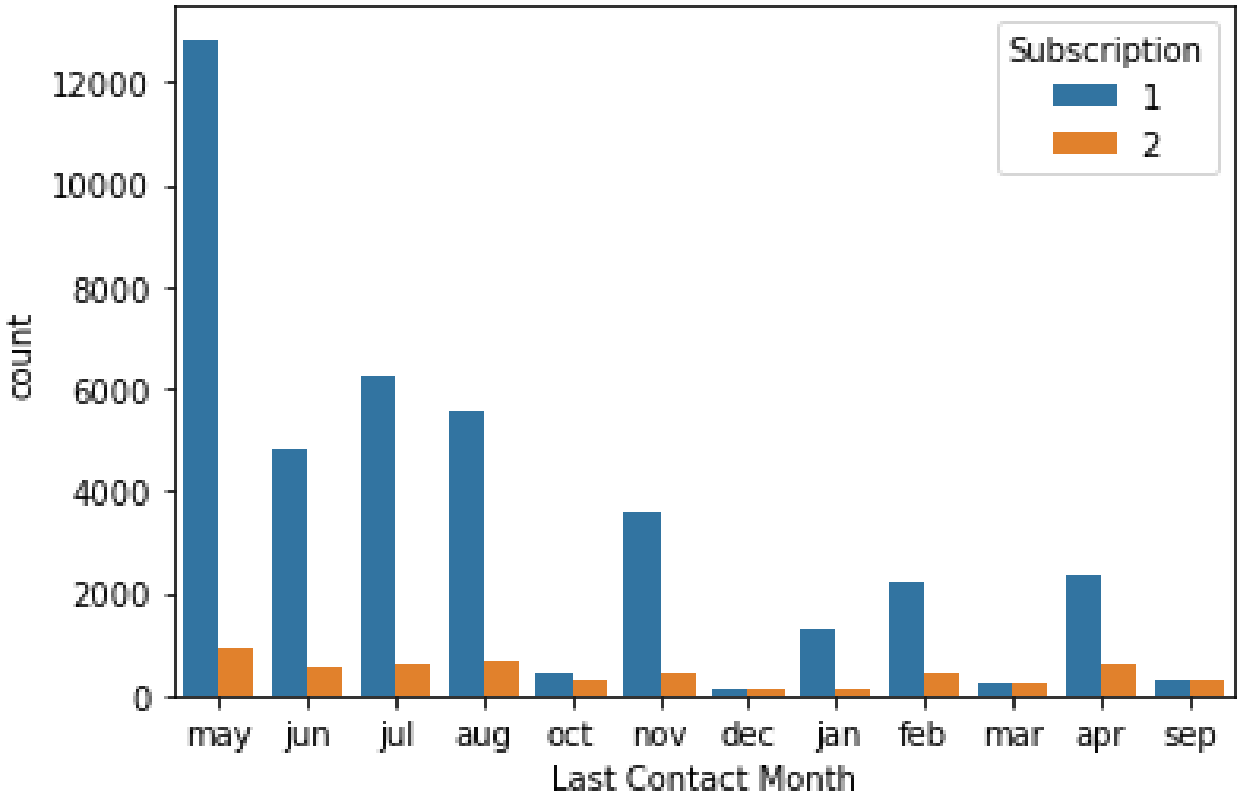


Fig 7: Customer Subscription and approach of Banking Personnel

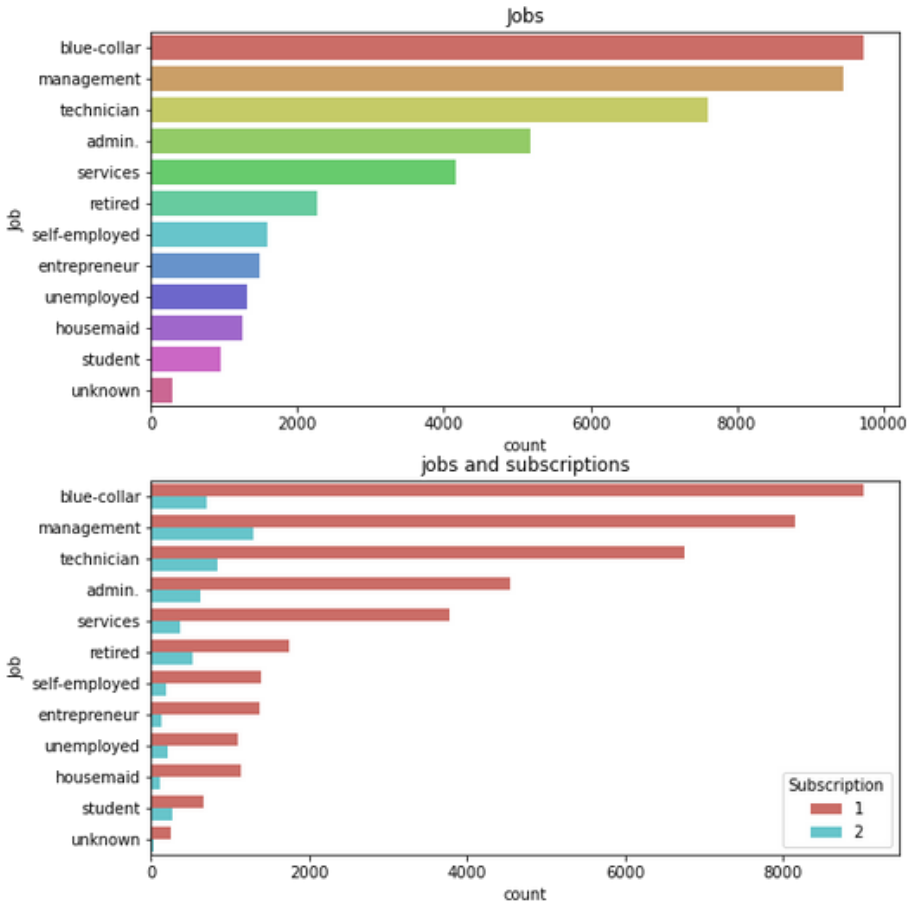


Fig 8: Trends of Job and Subscription

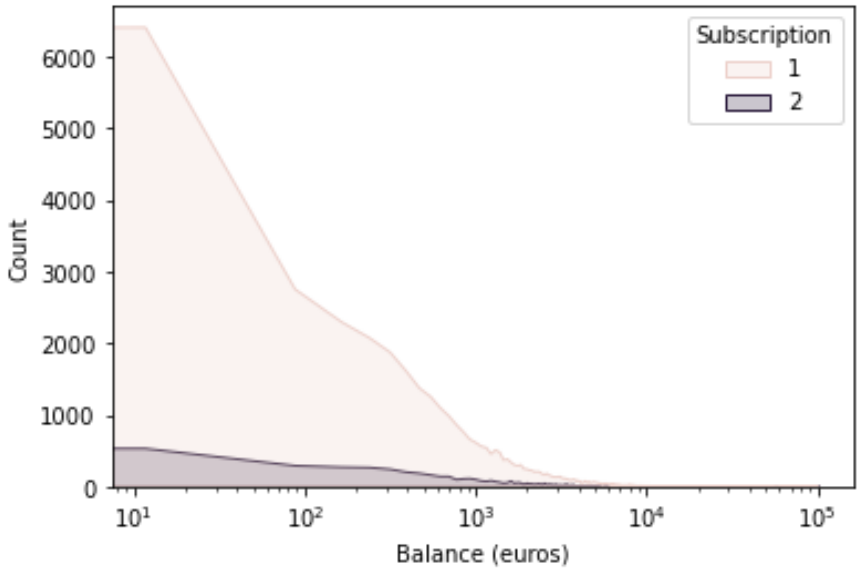


Fig 9: Balance and Subscription

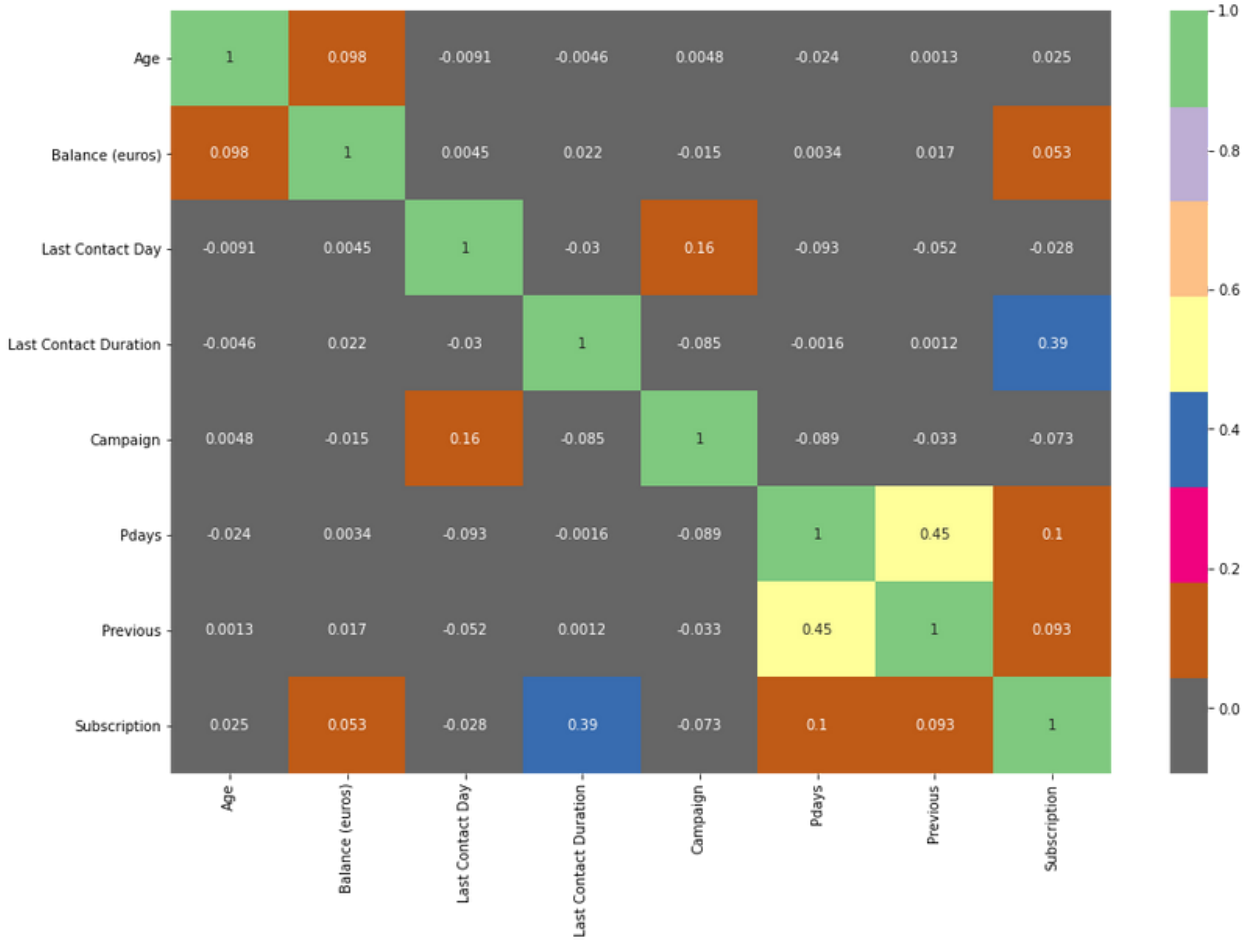


Fig 10: Correlation Map using Heat Map

Feature Engineering:

The Feature Engineering is the process of catching what avaliable data is appropriate to use for Machine Learning algorithm, these engineered features are also responsible for testing the accuracy of models and further improving it.

Feature Encoding:

In this procedure the Categorical values are treated with appropriate number mapping. Machine learning algorithms can then decide in a better way how those labels must be operated. It is an important pre-processing step for the structured dataset in supervised learning.

Feature Encoding

```
cat = data.select_dtypes(['object'])
for i in cat.columns:
    print(f'\n{i} contains of unique values: {cat[i].unique()}')
```

✓ 0.1s

```
Output exceeds the size limit. Open the full output data in a text editor

Job contains of unique values: ['management' 'technician' 'entrepreneur' 'blue-collar' 'unknown'
 'retired' 'admin.' 'services' 'self-employed' 'unemployed' 'housemaid'
 'student']

Marital Status contains of unique values: ['married' 'single' 'divorced']

Education contains of unique values: ['tertiary' 'secondary' 'unknown' 'primary']

Credit contains of unique values: ['no' 'yes']

Housing Loan contains of unique values: ['yes' 'no']

Personal Loan contains of unique values: ['no' 'yes']
...
Contact contains of unique values: ['unknown' 'cellular' 'telephone']

Last Contact Month contains of unique values: ['may' 'jun' 'jul' 'aug' 'oct' 'nov' 'dec' 'jan' 'feb' 'mar' 'apr' 'sep']

Poutcome contains of unique values: ['unknown' 'failure' 'other' 'success']
```

```
cat.columns
```

✓ 0.5s Python

```
Index(['Job', 'Marital Status', 'Education', 'Credit', 'Housing Loan',
       'Personal Loan', 'Contact', 'Last Contact Month', 'Poutcome'],
      dtype='object')
```

```
lb = LabelEncoder()
for i in cat.columns:
    data[i] = lb.fit_transform(data[i])
```

✓ 0.1s Python

```
data.head()
```

✓ 0.1s Python

	Age	Job	Marital Status	Education	Credit	Balance (euros)	Housing Loan	Personal Loan	Contact	Last Contact Day	Last Contact Month	Last Contact Duration	Campaign	Pdays	Previous	Poutcome	Subscription
0	58	4	1	2	0	2143	1	0	2	5	8	261	1	-1	0	3	1
1	44	9	2	1	0	29	1	0	2	5	8	151	1	-1	0	3	1
2	33	2	1	1	0	2	1	1	2	5	8	76	1	-1	0	3	1
3	47	1	1	3	0	1506	1	0	2	5	8	92	1	-1	0	3	1
4	33	11	2	3	0	1	0	0	2	5	8	198	1	-1	0	3	1

Label Encoding refers to converting the labels into a numeric form so as to convert them into the machine-readable form.

Feature Scaling:

Feature Scaling is a technique to standardize the independent features present in the data in fixed range. There are two main approaches like Standardization and Normalization. Here, the dataset is targeting categorical features, hence I used MinMaxScaler to perform Feature Scaling. Before performing the Feature Scaling, we have to segregate the data as Train and Test.

Seperating Dependent and Independent variable

```
x = data.drop(['Subscription'], axis = 1)
y = data.Subscription
```

✓ 0.9s

Train Test Split

```
x_train, x_test, y_train, y_test = train_test_split(x,y, test_size = 0.3, stratify = y)
```

✓ 0.1s

Feature Scaling

```
scale = MinMaxScaler()
x_train = scale.fit_transform(x_train)
x_test = scale.transform(x_test)
```

✓ 0.8s

Transform features by scaling each feature to a given range.

This estimator scales and translates each feature individually such that it is in the given range on the training set, e.g. between zero and one.

The transformation is given by:
X_std = (X - X.min(axis=0)) / (X.max(axis=0) - X.min(axis=0))
X_scaled = X_std * (max - min) + min

This transformation is often used as an alternative to zero mean, unit variance scaling.

Prevalance Rate:

Prevalance Rate of the Dependent variable helps to understand the distribution of the output labels in the dataset.

Prevalance Rate of Dependent Variable

```
data['Subscription'].value_counts(normalize= True)
```

✓ 0.5s

```
1    0.883015
2    0.116985
Name: Subscription, dtype: float64
```

Chapter - 3
Model Building

Data Preprocessing is finished and the data is ready for building a model and can predict the values by analysing the underlying pattern what it learn from the dataset. To understand the Underlying pattern, we have lots of modelling approaches to recognize the pattern to earn efficient results.

- Logistic Regression
- Decision Tree
- Random Forest
- Support Vector Machine
- Naive Bayes
- Gradient Boosting
- Ada Boosting
- K-Nearest Neighbors

Logistic Regression: Logistic regression is a supervised learning classification algorithm used to predict the probability of a target variable. The nature of target or dependent variable is dichotomous, which means there would be only two possible classes.

Logistic Regression

```
lr = LogisticRegression()
lr.fit(x_train, y_train)

y_pred = lr.predict(x_train)
y_pred1 = lr.predict(x_test)

print('Train Metrics')
print(metrics(y_train, y_pred))
print('\nTest Metrics')
print(metrics(y_test, y_pred1))

acc_lr_train = accuracy_score(y_train, y_pred)
acc_lr_test = accuracy_score(y_test, y_pred1)

cv_lr = cross_val_score(lr, x_train, y_train, cv = 5)
print(f'Mean Score of Logistic Model: {cv_lr.mean()}')

report = report.append({'Model': 'Logistic Regression',
                        'Train Accuracy': acc_lr_train,
                        'Test Accuracy': acc_lr_test,
                        'Cohen-Kappa Score': cohen_kappa_score(y_test, y_pred1),
                        'Recall': recall_score(y_test, y_pred1, average = 'weighted'),
                        'Mean Score': cv_lr.mean()}, ignore_index= True)
```

Decision Tree: Decision Tree is a Supervised learning technique that can be used for both classification and Regression problems, but mostly it is preferred for solving Classification problems. It is a tree-structured classifier, where internal nodes represent the features of a dataset, branches represent the decision rules and each leaf node represents the outcome.

Decision Tree Classifier

```
dt = DecisionTreeClassifier()
dt.fit(x_train, y_train)

y_pred = dt.predict(x_train)
y_pred1 = dt.predict(x_test)

print('Train Metrics')
print(metrics(y_train, y_pred))
print('\nTest Metrics')
print(metrics(y_test, y_pred1))

acc_dt_train = accuracy_score(y_train, y_pred)
acc_dt_test = accuracy_score(y_test, y_pred1)

cv_dt = cross_val_score(dt, x_train, y_train, cv = 5)
print(f'Mean Score of Logistic Model: {cv_lr.mean()}')

report = report.append({'Model': 'Decision Tree',
                        'Train Accuracy': acc_dt_train,
                        'Test Accuracy': acc_dt_test,
                        'Cohen-Kappa Score': cohen_kappa_score(y_test, y_pred1),
                        'Recall': recall_score(y_test, y_pred1, average = 'weighted'),
                        'Mean Score': cv_dt.mean()}, ignore_index= True)
```

✓ 0.8s

Random Forest: "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."

Random Forest Classifier

```
rf = RandomForestClassifier()
rf.fit(x_train, y_train)

y_pred = rf.predict(x_train)
y_pred1 = rf.predict(x_test)

print('Train Metrics')
print(metrics(y_train, y_pred))
print('\nTest Metrics')
print(metrics(y_test, y_pred1))

cv_rf = cross_val_score(rf, x_train, y_train, cv = 5)


report = report.append({'Model': 'Random Forest',
                        'Train Accuracy': accuracy_score(y_train, y_pred),
                        'Test Accuracy': accuracy_score(y_test, y_pred1),
                        'Cohen-Kappa Score': cohen_kappa_score(y_test, y_pred1),
                        'Recall': recall_score(y_test, y_pred1, average = 'weighted'),
                        'Mean Score': cv_rf.mean()}, ignore_index= True)
```

✓ 14.2s

Support Vectot Machine: 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.

Support Vector Machine

```
svm = SVC()
svm.fit(x_train, y_train)

y_pred = svm.predict(x_train)
y_pred1 = svm.predict(x_test)

print('Train Metrics')
print(metrics(y_train, y_pred))
print('\nTest Metrics')
print(metrics(y_test, y_pred1))

cv_svm = cross_val_score(svm, x_train, y_train, cv = 5)


report = report.append({'Model': 'Support Vector Machine',
                        'Train Accuracy': accuracy_score(y_train, y_pred),
                        'Test Accuracy': accuracy_score(y_test, y_pred1),
                        'Cohen-Kappa Score': cohen_kappa_score(y_test, y_pred1),
                        'Recall': recall_score(y_test, y_pred1, average = 'weighted'),
                        'Mean Score': cv_svm.mean()}, ignore_index= True)
```

✓ 1m 15.2s

Naive Bayes: Naive Bayes classifiers are a collection of classification algorithms based on Bayes' Theorem. It is not a single algorithm but a family of algorithms where all of them share a common principle, i.e. every pair of features being classified is independent of each other.

Naive Bayes

```
nb = GaussianNB()
nb.fit(x_train, y_train)

y_pred = nb.predict(x_train)
y_pred1 = nb.predict(x_test)

print('Train Metrics')
print(metrics(y_train, y_pred))
print('\nTest Metrics')
print(metrics(y_test, y_pred1))

cv_nb = cross_val_score(nb, x_train, y_train, cv = 5)


report = report.append({'Model': 'Naive Bayes',
                        'Train Accuracy': accuracy_score(y_train, y_pred),
                        'Test Accuracy': accuracy_score(y_test, y_pred1),
                        'Cohen-Kappa Score': cohen_kappa_score(y_test, y_pred1),
                        'Recall': recall_score(y_test, y_pred1, average = 'weighted'),
                        'Mean Score': cv_nb.mean()}, ignore_index= True)
```

✓ 0.3s

Gradient Boosting: The main idea behind this algorithm is to build models sequentially and these subsequent models try to reduce the errors of the previous model.

Gradient Boosting Classifier

```
gb = GradientBoostingClassifier()
gb.fit(x_train, y_train)

y_pred = gb.predict(x_train)
y_pred1 = gb.predict(x_test)


print('Train Metrics')
print(metrics(y_train, y_pred))
print('\nTest Metrics')
print(metrics(y_test, y_pred1))

cv_gb = cross_val_score(gb, x_train, y_train, cv = 5)


report = report.append({'Model': 'Gradient Boosting',
                        'Train Accuracy': accuracy_score(y_train, y_pred),
                        'Test Accuracy': accuracy_score(y_test, y_pred1),
                        'Cohen-Kappa Score': cohen_kappa_score(y_test, y_pred1),
                        'Recall': recall_score(y_test, y_pred1, average = 'weighted'),
                        'Mean Score': cv_gb.mean()}, ignore_index= True)
```

✓ 22.4s

Ada Boosting: AdaBoost also called Adaptive Boosting is a technique in Machine Learning used as an Ensemble Method. The most common algorithm used with AdaBoost is decision trees with one level that means with Decision trees with only 1 split. These trees are also called Decision Stumps.

AdaBoosting Classifier

```
abc = AdaBoostClassifier()
abc.fit(x_train, y_train)

y_pred = abc.predict(x_train)
y_pred1 = abc.predict(x_test)

print('Train Metrics')
print(metrics(y_train, y_pred))
print('\nTest Metrics')
print(metrics(y_test, y_pred1))

cv_abc = cross_val_score(abc, x_train, y_train, cv = 5)


report = report.append({'Model': 'Ada Boost Classifier',
                        'Train Accuracy': accuracy_score(y_train, y_pred),
                        'Test Accuracy': accuracy_score(y_test, y_pred1),
                        'Cohen-Kappa Score': cohen_kappa_score(y_test, y_pred1),
                        'Recall': recall_score(y_test, y_pred1, average = 'weighted'),
                        'Mean Score': cv_abc.mean()}, ignore_index= True)
```

✓ 4.6s

K Nearest Neighbors: K Nearest Neighbor algorithm falls under the Supervised Learning category and is used for classification (most commonly) and regression. It is a versatile algorithm also used for imputing missing values and resampling datasets.

KNN classifier

```
from tqdm import tqdm
knn_report = pd.DataFrame(columns = ['K', 'Train Accuracy', 'Test Accuracy'])
for i in tqdm(range(1, 5)):
    knn = KNeighborsClassifier(n_neighbors= i)
    knn.fit(x_train, y_train)

    pre_train = knn.predict(x_train)
    pre_test = knn.predict(x_test)

    knn_report = knn_report.append({'K': i,
                                    'Train Accuracy': accuracy_score(y_train, pre_train),
                                    'Test Accuracy': accuracy_score(y_test, pre_test)}, ignore_index = True)
    knn_report
```

```
knn = KNeighborsClassifier(n_neighbors= 3)
knn.fit(x_train, y_train)

pre_train = knn.predict(x_train)
pre_test = knn.predict(x_test)

print('Train Metrics')
print(metrics(y_train, y_pred))
print('\nTest Metrics')
print(metrics(y_test, y_pred1))

cv_knn = cross_val_score(knn, x_train, y_train, cv = 5)


report = report.append({'Model': 'KNN Classifier',
                        'Train Accuracy': accuracy_score(y_train, y_pred),
                        'Test Accuracy': accuracy_score(y_test, y_pred1),
                        'Cohen-Kappa Score': cohen_kappa_score(y_test, y_pred1),
                        'Recall': recall_score(y_test, y_pred1, average = 'weighted'),
                        'Mean Score': cv_knn.mean()}, ignore_index= True)
```

✓ 36.8s

Report Analysis:

The report contains of Train data's Accuracy, Test Data's Accuracy, Recall, Cohen Kappa Score, Mean Score, these metrics parameter helps to understand the model's performance so that we can pick the best model from the analysis of Report.

Report Analysis

```
report
```

✓ 0.8s

	Model	Train Accuracy	Test Accuracy	Cohen-Kappa Score	Recall	Mean Score
0	Logistic Regression	0.890195	0.891330	0.253807	0.891330	0.890100
1	Decision Tree	1.000000	0.866337	0.364060	0.866337	0.872911
2	Random Forest	1.000000	0.903273	0.441868	0.903273	0.907385
3	Support Vector Machine	0.892628	0.892141	0.226063	0.892141	0.891238
4	Naive Bayes	0.839353	0.836331	0.314658	0.836331	0.840301
5	Gradient Boosting	0.910797	0.904379	0.431388	0.904379	0.904288
6	Ada Boost Classifier	0.897399	0.898408	0.388692	0.898408	0.897305
7	KNN Classifier	0.897399	0.898408	0.388692	0.898408	0.884886

Metrics:

Train Accuracy & Test Accuracy:

Accuracy is calculated with the help of accuracy metrics in Sklearn, by comparing the **True Value** and **Predicted value**. This give the result how good our model is performed over the Train and Test Results over the Predicted Values.

Cohen Kappa Score:

Cohen Kappa Score which helps to test **Interrater Reliability**.

Range: -1 to +1, it also helps to answer the question of Consistency

- 0 - represents yhe amount of agreement that can be expected from random chance,
- 1 - perfect agreement between the raters.
- <=0 - No agreement
- 0.01 - 0.20 - none to slight
- 0.21 - 0.40 -- Fair Agreement
- 0.41 - 0.60 -- Moderate
- 0.61 - 0.80 -- Substancial
- 0.81 - 1.00 -- Perfect Agreement

Recall:

It quantifies the **number of Positive Class Prediction** made out of all Positive examples in the given Dataset.

Mean Score:

This can be calculated by the help of **Cross Validation method,** which gives the results to check for **Model Stabillity**. Here for consideration I took **K folds as 5** .For each Fold it took all the samples and consider it for validation and produces the result of Stability.

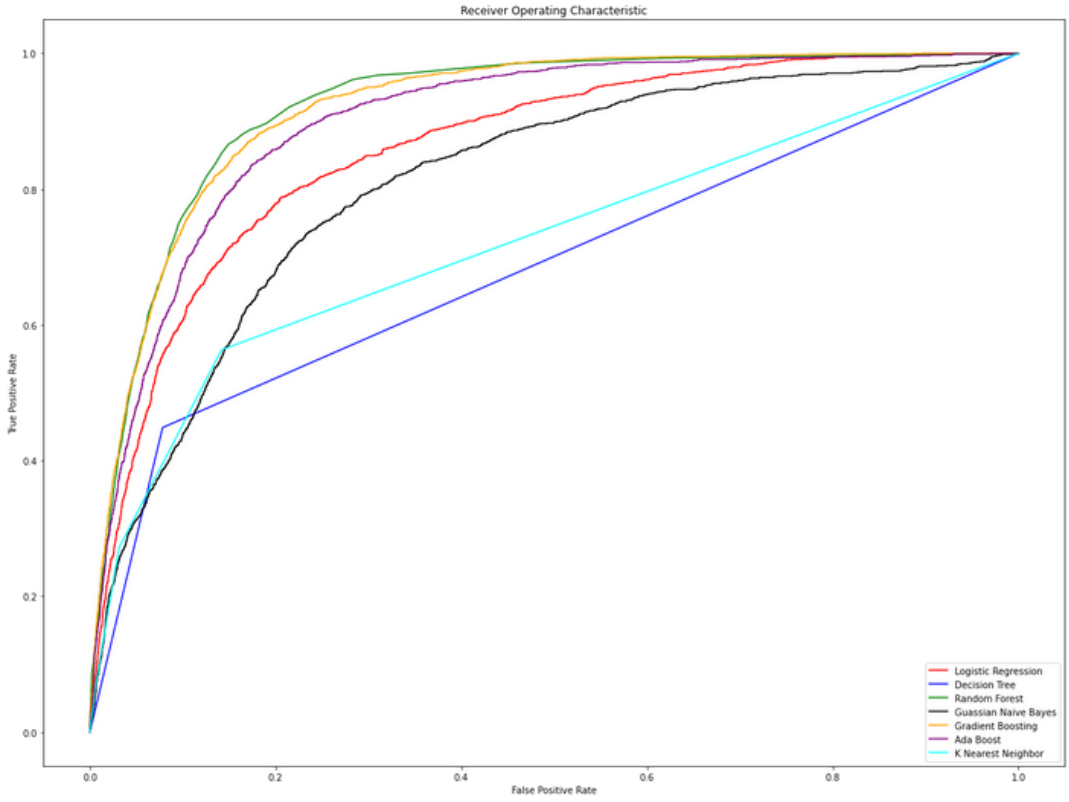
Mean Score Analysis:

	Logistic Regression	Decision Tree	Random Forest	Support Vector Machine	Naive Bayes	Gradient Boosting	KNN	Ada Boost Classfier
0	0.891469	0.868088	0.905371	0.890679	0.840442	0.903949	0.882306	0.897788
1	0.886730	0.870142	0.908373	0.889889	0.833333	0.903791	0.885624	0.895735
2	0.893032	0.874546	0.910886	0.891926	0.839943	0.909622	0.884342	0.899194
3	0.888292	0.872334	0.908516	0.890662	0.838995	0.900616	0.885132	0.896666
4	0.890978	0.879444	0.903776	0.893032	0.848791	0.903460	0.887028	0.897140

ROC & AUC:

The Receiver Operator Characteristic (ROC) curve is an evaluation metric for binary classification problems. It is a probability curve that plots the TPR against FPR at various threshold values and essentially separates the 'signal' from the 'noise'. The Area Under the Curve (AUC) is the measure of the ability of a classifier to distinguish between classes and is used as a summary of the ROC curve.

The higher the AUC, the better the performance of the model at distinguishing between the positive and negative classes.



Conclusion:

From the result we can see that Random Forest Classifer does a really great job. It has a good accuracy and also a good stability.

Even though all the models are performing well in Train data, when i consider the recall score it shows very less score, so we can clearly say our model is biased over one particular class of 1.

To solve this problem we have to balance the data by either Oversampling or Undersampling. This can be achieved by various dedicated methods like SMOTE, Near Miss.

Chapter - 4
Balancing Dataset

Balancing the Dataset:

Class imbalance is a scenario that arises when we have unequal distribution of class in a dataset i.e. the no. of data points in the negative class (majority class) very large compared to that of the positive class (minority class).

Generally, the imbalance is a problem of class imbalance, mostly we are interested in the positive class/minority class. If the imbalance data is not treated properly, the model may not be able to learn the data well, it may bias over one majority class. This results in high bias problem in the model.
Hence we need to bias the model.

Problem:

In the given problem, we have two class that are subscribed and not subscribed, where So a binary classifier model need not be a complex model to predict all outcomes as 1 meaning not - subscribed and achieve a great accuracy of 99%. Clearly, in such cases where class distribution is skewed, the accuracy metric is biased and not preferable.

Solution:

To overcome this overfitting problem, we have technique called **SMOTE**.

SMOTE - Synthetic Minority Over-Sampling Technique

SMOTE is an oversampling technique where the synthetic samples are generated for the minority class. This algorithm helps to *overcome the overfitting problem* posed by random oversampling. It focuses on the feature space to generate new instances with the help of interpolation between the positive instances that lie together.

Balancing Dataset

```
sm = SMOTE()
x_train, y_train = sm.fit_sample(x_train, y_train)
x_train.shape, y_train.shape
```

✓ 0.3s

```
((55866, 16), (55866,))
```

```
y_train.value_counts(normalize = True)
```

✓ 0.6s

```
1    0.5
2    0.5
Name: Subscription, dtype: float64
```

Report Analysis:

Here the Model is build on the balanced data, and the balancing is achevied with the help of approach called "SMOTE". The report contains of Train data's Accuracy, Test Data's Accuracy, Recall, Cohen Kappa Score, Mean Score, these metrics parameter helps to understand the model's performance so that we can pick the best model from the analysis of Report.

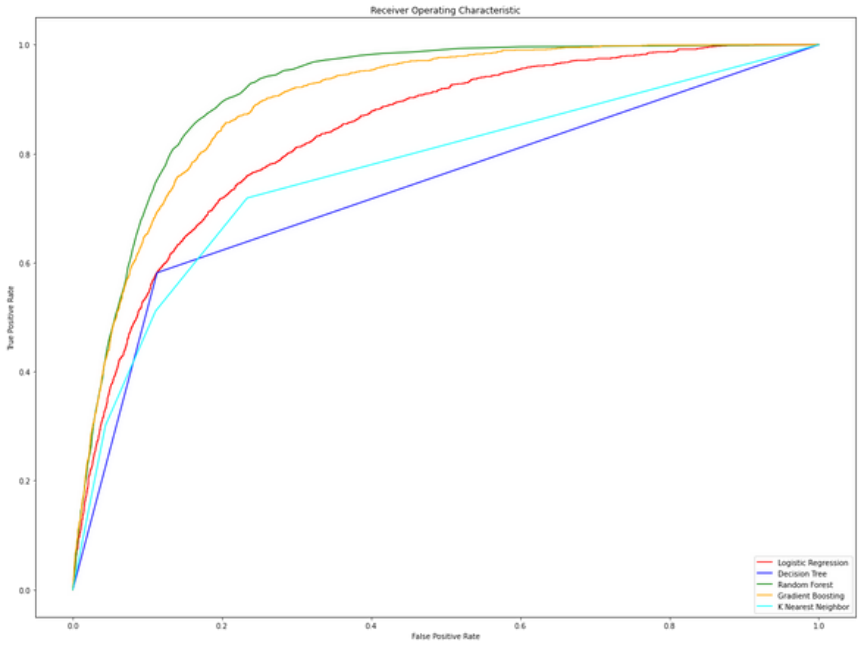
```
report
```
✓ 0.8s

	Model	Train Accuracy	Test Accuracy	Cohen-Kappa Score	Recall	Mean Score
0	Logistic Regression	0.851502	0.812592	0.354903	0.812592	0.848424
1	Decision Tree	1.000000	0.851592	0.393279	0.851592	0.890400
2	Random Forest	1.000000	0.884031	0.500708	0.884031	0.924643
3	Support Vector Machine	0.880661	0.840018	0.395609	0.840018	0.871014
4	Gradient Boosting	0.896109	0.854984	0.458746	0.854984	0.887858
5	KNN Classifier	0.896109	0.854984	0.458746	0.854984	0.876492

Mean Score Analysis:

	Logistic Regression	Decision Tree	Random Forest	Support Vector Machine	Gradient Boosting	KNN
0	0.772239	0.812153	0.844818	0.780383	0.808305	0.788258
1	0.868791	0.908082	0.942809	0.892598	0.908887	0.896357
2	0.866195	0.910588	0.945494	0.894030	0.906918	0.902085
3	0.870849	0.912378	0.945852	0.896626	0.911215	0.898326
4	0.864047	0.908798	0.944241	0.891435	0.903965	0.897431

ROC & AUC:



Conclusion:

The best pick model is **RandomForestClassifier** for this problem, even though it yields better result. The model is overfitted, so to solve those overfitting in traning data, we can adjust the parameter to get better results.

For adjusting the parameter we can access over certain parameters like,

- To Increase model's reliabiity:

1) Max depth
2) N_Estimators
3) Min_Sample_split
4) Min_Sample_leaf

- To increase the speed of calculation:

1) oob_score
2) n_jobs

Chapter - 5
Hyper Tuning

The best way to think about hyperparameters is like the settings of an algorithm that can be adjusted to optimize performance. In the case of a random forest, hyperparameters include the number of decision trees in the forest and the number of features considered by each tree when splitting a node. (The parameters of a random forest are the variables and thresholds used to split each node learned during training). The best hyperparameters are usually impossible to determine ahead of time, and tuning a model is where machine learning turns from a science into trial-and-error based engineering.

List of Few Hyperparameters to tune the model to get efficient models.

- n_estimators = number of trees in the foreset
- max_features = max number of features considered for splitting a node
- max_depth = max number of levels in each decision tree
- min_samples_split = min number of data points placed in a node before the node is split
- min_samples_leaf = min number of data points allowed in a leaf node
- bootstrap = method for sampling data points (with or without replacement)

Random Forest Hyper Parameter Tuning - 1

```
rf1 = RandomForestClassifier(n_estimators= 50, max_depth = 5, n_jobs= 4)
rf1.fit(x_train, y_train)


y_pred = rf1.predict(x_train)
y_pred1 = rf1.predict(x_test)

print('Train Metrics')
print(metrics(y_train, y_pred))
print('\nTest Metrics')
print(metrics(y_test, y_pred1))

hyper_tuned_rf_report = hyper_tuned_rf_report.append({'Model': 'RF Hypertuned 1',
                        'Train Accuracy': accuracy_score(y_train, y_pred),
                        'Test Accuracy': accuracy_score(y_test, y_pred1),
                        'Cohen-Kappa Score': cohen_kappa_score(y_test, y_pred1),
                        'Recall': recall_score(y_test, y_pred1, average = 'weighted')}, ignore_index= True)
```

Random Forest Hyper Parameter Tuning - 2

```
rf2 = RandomForestClassifier(n_estimators= 75, max_depth = 100, max_features='auto', n_jobs= 4)
rf2.fit(x_train, y_train)


y_pred = rf2.predict(x_train)
y_pred1 = rf2.predict(x_test)

print('Train Metrics')
print(metrics(y_train, y_pred))
print('\nTest Metrics')
print(metrics(y_test, y_pred1))

hyper_tuned_rf_report = hyper_tuned_rf_report.append({'Model': 'RF Hypertuned 2',
                        'Train Accuracy': accuracy_score(y_train, y_pred),
                        'Test Accuracy': accuracy_score(y_test, y_pred1),
                        'Cohen-Kappa Score': cohen_kappa_score(y_test, y_pred1),
                        'Recall': recall_score(y_test, y_pred1, average = 'weighted')}, ignore_index= True)
```

✓ 1.8s

Grid Search CV:

The idea that hyperparameter tuning using scikit-learn's GridSearchCV was the greatest invention of all time. It runs through all the different parameters that is fed into the parameter grid and produces the best combination of parameters, based on a scoring metric of your choice (accuracy, f1, etc). Obviously, nothing is perfect and GridSearchCV is no exception:

- "best parameters" results are limited
- process is time-consuming

The "best" parameters that GridSearchCV identifies are technically the best that could be produced, but only by the parameters that you included in your parameter grid.

```
params = {'n_estimators': [75, 80, 85, 100],
          'n_jobs': [4],
          'max_features': ['auto', 10],
          'max_depth': [50, 60, 70]}

rf = RandomForestClassifier()
gscv = GridSearchCV(rf, params)
gscv.fit(x_train, y_train)

print('Best Score: ', gscv.best_score_)
print('Best Parameter: ', gscv.best_params_)
```

✓ 6m 18.2s

```
Best Score:  0.9242256042972248
Best Parameter:  {'max_depth': 60, 'max_features': 'auto', 'n_estimators': 75, 'n_jobs': 4}
```

Report Analysis

```
hyper_tuned_rf_report
```

✓ 0.2s

	Model	Train Accuracy	Test Accuracy	Cohen-Kappa Score	Recall
0	RF Hypertuned 1	0.870582	0.817311	0.395257	0.817311
1	RF Hypertuned 2	0.999946	0.885063	0.505483	0.885063

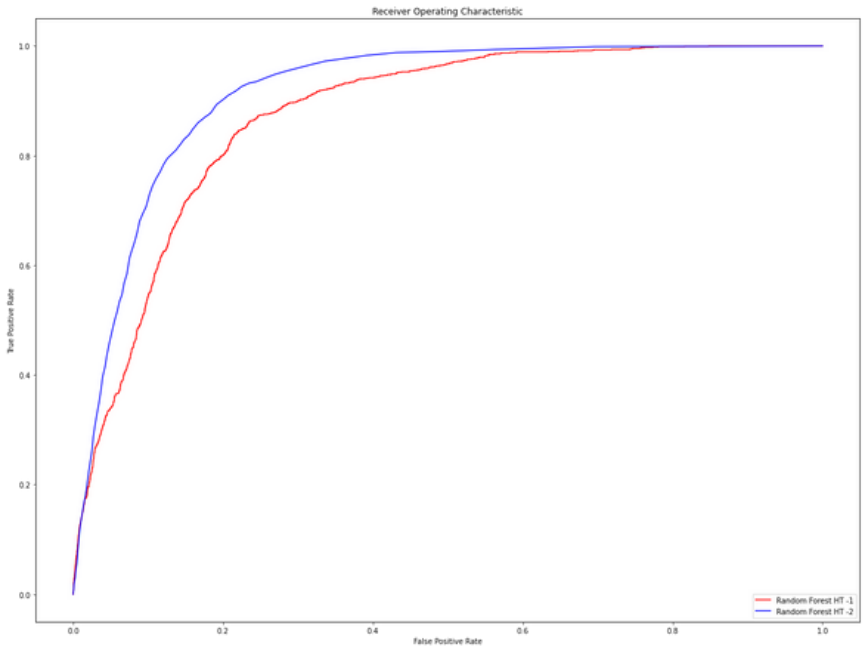
ROC & AUC:

```
# Hyper Parameter Tuning 1
probs = rf1.predict_proba(x_test)
preds = probs[:, 1]
fprrf1, tprrf1, threshrf1 = roc_curve(y_test, preds, pos_label=2)
roc_aucrf = auc(fprrf1, tprrf1)

# Hyper Parameter Tuning 2
probs = rf2.predict_proba(x_test)
preds = probs[:, 1]
fprrf2, tprrf2, threshrf2 = roc_curve(y_test, preds, pos_label=2)
roc_aucrf = auc(fprrf2, tprrf2)
```

✓ 0.2s

```
plt.figure(figsize=(20,15))
plt.plot(fprrf1, tprrf1, label = "Random Forest HT -1", color = 'red')
plt.plot(fprrf2, tprrf2, label = "Random Forest HT -2", color = 'blue')
plt.title('Receiver Operating Characteristic')
plt.ylabel('True Positive Rate')
plt.xlabel('False Positive Rate')
plt.legend(loc = 'lower right')
```

✓ 0.5s



From the Analysis of report and understand the various evaluation metric parameters, we can conclude the Random Forest is the best model. So after analysing the result produced by Random Forest is reliable. By tuning, the various parameters in Random Forest model we can get reliable results. Hence the HyperParameter Tuning is taken place it can be acheiv ed better with the help of GridSearchCV which is dedicated to build numerous models based on the parameter we entered to the baseline model.

Best Reliable Model:

```
rf = RandomForestClassifier(max_depth= 60, max_features='auto', n_estimators= 1000, n_jobs=4 , max_leaf_nodes=1000)
rf.fit(x_train, y_train)
y_pred = rf.predict(x_train)
y_pred1 = rf.predict(x_test)

print('Train Metrics')
print(metrics(y_train, y_pred))
print('\nTest Metrics')
print(metrics(y_test, y_pred1))

cv_rf = cross_val_score(rf, x_train, y_train, cv = 5)
cv_test_rf = cross_val_score(rf, x_test, y_test, cv = 5)
```

✓ 2m 13.1s

The model is build and tuned with the best estimator that was returned from the GridSearchCV results.

```
Train Metrics
Confusion Matrix:
 [[25918  2007]
 [ 1079 26846]]
Classification Report:
              precision    recall  f1-score   support

           1       0.96      0.93      0.94     27925
           2       0.93      0.96      0.95     27925

    accuracy                           0.94     55850
   macro avg       0.95      0.94      0.94     55850
weighted avg       0.95      0.94      0.94     55850

Accuracy Score: 0.9447448522829006
None

Test Metrics
Confusion Matrix:
 [[10749  1248]
 [  451  1116]]
Classification Report:
              precision    recall  f1-score   support

           1       0.96      0.90      0.93     11997
           2       0.47      0.71      0.57      1567

    accuracy                           0.87     13564
   macro avg       0.72      0.80      0.75     13564
weighted avg       0.90      0.87      0.89     13564

Accuracy Score: 0.8747419640224122
None
```

Metrics Analysis:

The Problem of Overfitting is reduced, we can clearly see that from the results of Train and Test accuracy scores.

Mean Score Analysis:

This can be calculated by the help of Cross Validation method, which gives the results to check for Model Stabillity. Here for consideration I took K folds as 5 .For each Fold it took all the samples and consider it for validation and produces the result of Stability.

```
print('Train Cross Validation Report ')
print("Mean Score of Model: ", cv_rf.mean())
print('Cross Validation Score: ', cv_rf)

print('\nTest Cross Validation Report')
print('Mean Score: ', cv_test_rf.mean())
print('Cross Validation Score: ', cv_test_rf)
```

✓ 0.1s

```
Train Cross Validation Report
Mean Score of Model:  0.9077350044762758
Cross Validation Score:  [0.83213966 0.92309758 0.93025962 0.92900627 0.92417189]

Test Cross Validation Report
Mean Score:  0.9048215899193982
Cross Validation Score:  [0.90195356 0.91043126 0.90047917 0.9063767  0.90486726]
```

The result produces was so reliable and score of both train and test are acheived very close to each other.

Saving the Model:

The trained models in a file and restore them in order to reuse it to compare the model with other models, to test the model on a new data. The saving of data is called Serialization, while restoring the data is called Deserialization. This can be donw with the help of Pickling

Pickling:

The pickle module implements binary protocols for serializing and de-serializing a Python object structure. “Pickling” is the process whereby a Python object hierarchy is converted into a byte stream, and “unpickling” is the inverse operation, whereby a byte stream (from a binary file or bytes-like object) is converted back into an object hierarchy. Pickling (and unpickling) is alternatively known as “serialization”, “marshalling,” 1 or “flattening”; however, to avoid confusion, the terms used here are “pickling” and “unpickling”.

In real world sceanario, the use pickling and unpickling are widespread as they allow us to easily transfer data from one server/system to another and then store it in a file or database.

```
import os
import pickle

path = os.getcwd()
filename = 'randomforest.pkl'

# Pickling the model in a Current Working Directory
pickle.dump(rf, open(path + '\\'+ filename, 'wb'))
```

✓ 0.6s

Steps:

- The given dataset is unbalanced and biased over unsubscribed customers, by analysing the data and even build the model along with unbalancing problem, provide better results but the results what we earned not so confident. Because the model is learned and biased over majority in the dataset.

- After this, I tried to solve the problem of unbalancing by using **SMOTE**, which is the best method of oversampling to solve this unbalancing problem. By fitting the training samples into the model, we can generate new samples from the minority class.

- After completing, I tried to build the model with the help of **HyperTuning**, which is the best method to improve the model's reliability.

Chapter - 6
Conclusion

Conclusion:

Without sampling the dataset, the **Logistic Regression** did a great job than any other models and the results are also more reliable.

After sampling the dataset by using **SMOTE**, **Random Forest** provide a better result, so I took **Random Forest** as a base model and did hypertuning to reduce the overfitting by controlling important parameters.

So in final, the results are based on 95% confidence interval, where the **Mean Score** of train accuracy is approxiamated over *89%* and test accuracy is approximated over *91%.*

Chapter - 7
Model Deployment

Model Deployment:

Deploying a machine learning model, known as model deployment, simply means to integrate a machine learning model and integrate it into an existing production environment where it can take in an input and return an output. The purpose of deploying your model is so that you can make the predictions from a trained ML model available to others, whether that be users, management, or other systems. Model deployment is closely related to ML systems architecture, which refers to the arrangement and interactions of software components within a system to achieve a predefined goal.

Before you deploy a model, there are a couple of criteria that your machine learning model needs to achieve before it's ready for deployment:

- **Portability**: this refers to the ability of your software to be transferred from one machine or system to another. A portable model is one with a relatively low response time and one that can be rewritten with minimal effort.
- **Scalability**: this refers to how large your model can scale. A scalable model is one that doesn't need to be redesigned to maintain its performance.

Different Methods to Deploy Your Model:

There are three general ways to deploy your ML model: one-off, batch, and real-time.

- **One-off**

It's not always that you need to continuously train a machine learning model in order for it to be deployed. Sometimes a model is only needed once or periodically. In this case, the model can simply be trained ad-hoc when it's needed and pushed to production until it deteriorates enough to need some fixing.

- **Batch**

Batch training allows you to constantly have an up-to-date version of your model. It is a scalable method that takes a subsample of data at a time, eliminating the need to use the full data set for each update. This is good if you use the model on a consistent basis, but don't necessarily require the predictions in real-time.

- **Real-time**

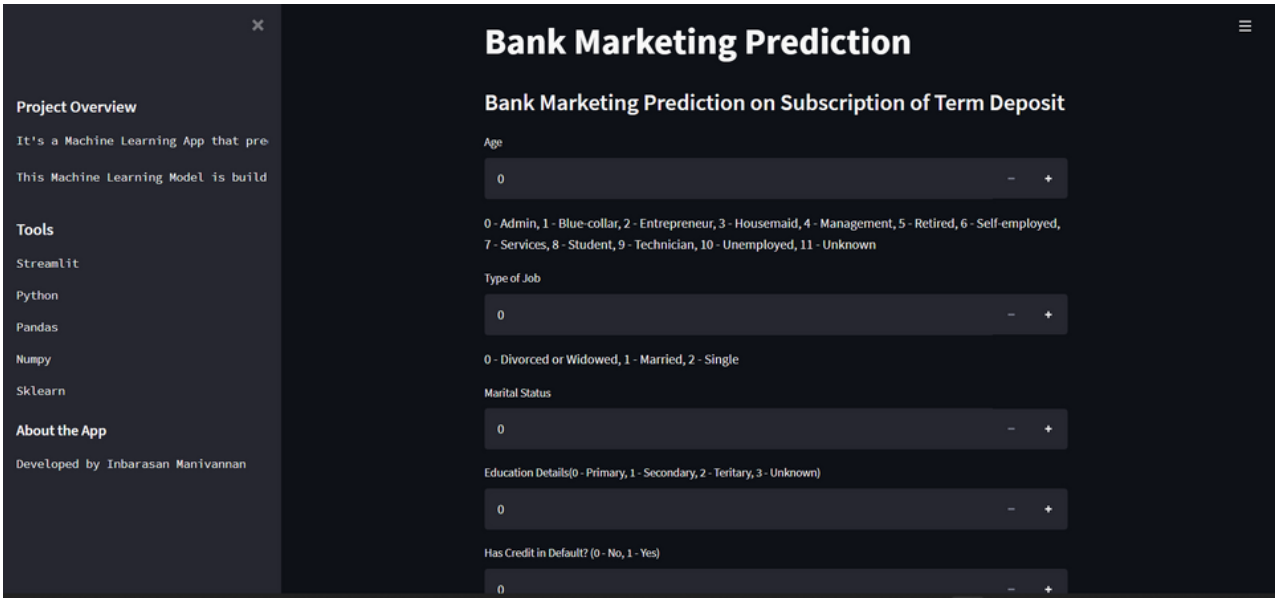
In some cases you'll want a prediction in real-time, for example, to determine if a transaction is fraudulent or not. This is possible by using online machine learning models, like linear regression using stochastic gradient descent.

Code:

```
# Importing Libraries
import streamlit as st
import pickle
import numpy as np
import string
st.set_option('deprecation.showfileUploaderEncoding',False)

# Loading a best model
model = pickle.load(open('randomforest.pkl', 'rb'))

# Input Boxes:
# Client Information
age = st.number_input("Age", 0, 100)
st.write('0 - Admin, 1 - Blue-collar, 2 - Entrepreneur, 3 - Housemaid, 4 - Management, 5 - Retired, 6 - Self-employed, 7 - Services, 8 - Student, 9 - Technician, 10 -
job = st.number_input('Type of Job', 0, 11)
st.write('0 - Divorced or Widowed, 1 - Married, 2 - Single')
marital = st.number_input('Marital Status', 0, 2)
education = st.number_input('Education Details(0 - Primary, 1 - Secondary, 2 - Teritary, 3 - Unknown)', 0, 3)
default = st.number_input('Has Credit in Default? (0 - No, 1 - Yes)', 0,1)
balance = st.number_input("Average Yearly balance in Euro's", 0, 1000000)
housing = st.number_input('Has Housing Loan (0 - No, 1 - Yes)', 0,1)
loan = st.number_input('Has Personal Loan (0 - No, 1 - Yes)', 0,1)

# Last Contact of Current Campaign Information
contact = st.number_input('Contact Communication Type (0 - Cellular, 1 - Telephone, 2 - Unknown)', 0,2)
day = st.number_input('Last Contact day of the month', 0,31)
month = st.number_input('Last Contact month of the Year (January - 0...Dec - 12)', 0,12)
duration = st.number_input('Last Contact Duration in Seconds', 0,28800)

# Campaigning Information
campaign = st.number_input('Number of contacts performed during this campaign and for this client', 0, 63)
pdays = st.number_input('Number of days that passed by after the client was last contacted from a previous campaign (-1 - Not Contacted)', -1, 1)
previous = st.number_input('Number of contacts performed before this campaign and for this client', 0, 275)
poutcome = st.number_input('Outcome of the previous marketing campaign (0 - Failure, 1 - Other, 2 - Success, 3 - Unknown)', 0,3)


#Input List of values
input = [[age, job, marital, education, default, balance, housing, loan, contact, day, month, duration, campaign, pdays, previous, poutcome]]
```

```
# Output Show
potential="""
  <div style="background-color:#F4D03F;padding:10px >
   <h2 style="color:white;text-align:center;"> Potential Customer - will Subscribe</h2>
   </div>
"""
not_potential="""
  <div style="background-color:#F08080;padding:10px >
   <h2 style="color:black ;text-align:center;"> will not Subscribe</h2>
   </div>
"""


if st.button("Predict"):
    output= model.predict(input)
    updated_res = output.flatten().astype(float)

    if output > 0.75:
        st.markdown(not_potential,unsafe_allow_html=True)
        st.write('Suggestion: ')
        st.write('We can suggest the Customer with some attractive benefits we can make them subscribe to TERM DEPOSIT. The Benefits like Tax Benefits, Easy Loan Acee

    else:
        st.markdown(potential,unsafe_allow_html=True)
        st.write('Suggestion: ')
        st.write('By giving the above parameters, the probability of subscribing for TERM DEPOSIT is high. So the Customer will subscribe for TERM DEPOSIT')

if __name__ == '__main__':
    main()
```

Output:



The model is Successfully deployed using platform called Streamlit. By giving the input parameters in real time we can earn result and suggestion based on the result what we earned.

Refrences

- https://pypi.org/
- https://pandas.pydata.org/docs/
- https://numpy.org/doc/stable/
- https://scikit-learn.org/stable/
- https://streamlit.io/
- https://docs.streamlit.io/