Marketing Campaign Prediction

A PROJECT REPORT

Submitted by

ABHAY SHAJI [Reg No: RA2112704010006]

Under the Guidance of

Dr. Kalpana A V

(Assistant Professor, Department of Data Science and Business Systems)

In partial fulfillment of the Requirements for the Degree of

MASTER OF TECHNOLOGY (INTEGRATED)

DEPARTMENT OF DATA SCIENCE AND BUSINESS SYSTEMS

FACULTY OF ENGINEERING AND TECHNOLOGY

SRM INSTITUTE OF SCIENCE AND TECHNOLOGY

NOVEMBER 2022

SRM INSTITUTE OF SCIENCE AND TECHNOLOGY

KATTANKULATHUR-603203

SRM INSTITUTE OF SCIENCE AND TECHNOLOGY

KATTANKULATHUR-603203

BONAFIDE CERTIFICATE

Certified that this project report titled "**MARKETING OUTCOME PREDICTION"** is the bonafide work of **"Abhay Shaji Reg No: RA2112704010006"** who carried out the project work under my supervision. Certified further, that to the best of my knowledge the work reported herein does not form part of any other thesis or dissertation on the basis of which a degree or award was conferred on an earlier occasion for this or any other candidate.

Dr. A.V.Kalpana
GUIDE
Assistant Professor
Dept. of DSBS

Dr.G Vadivu
PROGRAMME COORDINATOR
Dept. of DSBS

Dr. M.Lakshmi
HEAD OF DEPARTMENT
Dept. of DSBS

Signature of Internal Examiner

Signature of External Examiner

ABSTRACT

In banks, huge data records information about their customers. This data can be used to create and keep clear relationship and connection with the customers in order to target them individually for definite products or banking offers. Usually, the selected customers are contacted directly through: personal contact, telephone cellular, mail, and email or any other contacts to advertise the new product/service or give an offer, this kind of marketing is called direct marketing. In fact, direct marketing is in the main a strategy of many of the banks and insurance companies for interacting with their customers.

ACKNOWLEDGEMENTS

We express our humble gratitude to **Dr C. Muthamizhchelvan**, Vice-Chancellor, SRM Institute of Science and Technology, for the facilities extended for the project work and his continued support.

We extend our sincere thanks to Dean-CET, SRM Institute of Science and Technology, **Dr T.V.Gopal**, for his invaluable support.

We wish to thank **Dr Revathi Venkataraman**, Professor & Chairperson, School of Computing, SRM Institute of Science and Technology, for her support throughout the project work.

We are incredibly grateful to our Head of the Department**, Dr M. Lakshmi** Professor, Department of Data Science and Business Systems, SRM Institute of Science and Technology, for her suggestions and encouragement at all the stages of the project work.

We want to convey our thanks to our program coordinators **Dr.E.Sasikala,** Professor, Department of Data Science and Business Systems, SRM Institute of Science and Technology, for her inputs during the project reviews and support.

We register our immeasurable thanks to our Faculty Advisor**, Dr K Shantha Kumari, Ph.D.,** Assistant Professor, Department of Data Science and Business Systems, SRM Institute of Science and Technology, for leading and helping us to complete our course.

Our inexpressible respect and thanks to my guide, **Dr. Kalpana A V,** Assistant Professor, Department of Data Science and Business Systems, for providing me with an opportunity to pursue my project under his mentorship. She provided us with the freedom and support to explore the research topics of our interest.

We sincerely thank the Data Science and Business Systems staff and students, SRM Institute of Science and Technology, for their help during our project. Finally, we would like to thank parents, family members, and friends for their unconditional love, constant support, and encouragement.

ABHAY SHAJI

TABLE OF CONTENTS

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

1.1 General

Marketing is technique of exposing the target clients to a product via suitable systems and channels. It ultimately facilitates the way to buy the product or service and even helps in determining the need of the product and persuade customers to buy it. The overall aim is to increase sales of products and services for enterprise, business and financial institutions. It also helps to maintain the reputation of the company. Telemarketing is form of direct marketing in which salesperson approaches the customer either face to face or phone call and persuade him to buy the product. Telemarketing attains most popularity in 20th century and still gaining it. Nowadays, telephone (fixed-line or mobile) has been broadly used. It is cost effective and keeps the customers up to date. In Banking sector, marketing is the backbone to sell its product or service. Banking advertising and marketing is mostly based on an intensive knowledge of objective information about the market and the actual client needs for the bank profitable manner. Making right decisions in organizational operations are sometimes proved a great challenge where the quality of decision really matters. Decision Support Systems (DSS) are classified as a particular class of computerized facts and figures that helps the organization or administration into their decision making actions. The concept of DSS originates from a balance which lies between the data generated by computer and the judgment of human. According to Rupnik & Kukar (2007) the objective of decision support systems is to enhance the effectiveness of the decisions. This is a great tool which can analyze the sales data and provide further predictions. The purposes which can be established from the DSS are such as, analysis, optimization, forecasting and simulation. A study by Power (2008) found that research subjects who use DSS for the decision making, come-up with more effective decisions than those who did not use it. Nowadays, DSS is contributing a meaningful role in many fields such as for medical diagnosis, business and management, investment portfolios, command and control of military units, and statistics. DSS uses statistical data to overcome the deficiencies and helps the decision makers to take the right decision. Data mining (DM) plays vital role to support the Decision support systems which are based on the data obtained from the data mining models: rules, patterns and relationship. Data mining is the process of selecting, discovering, and modeling high volume of data to find and clarify unknown patterns. The objective of data mining in decision support systems is to suggest a tool which is easily accessible for the business users to analyze the data mining models. A specific technology used within the DSS is Machine learning (ML) that combines data and computer applications to accurately predicting the results. The fundamental principle of machine learning is to construct the algorithms that can obtain input data and then predict the results or outputs by using the statistical analysis within satisfactory interval. ML allows the DSS to obtain the new knowledge which helps it to make right decisions. Machine Learning can be mainly classified in 2 categories i.e. supervised learning and unsupervised learning. In supervised learning, the output of algorithm is already known and we use the input data to predict the output. The examples of supervised learning are regression and classification. In contrast, unsupervised learning we only have input data whereas no corresponding output variables are selected. The example of unsupervised learning is clustering.

1.2 MOTIVATION

The main idea behind developing the classification project is to build awareness regarding plant-watching, plant and their identification, especially plants found in India. It also caters to the need of simplifying the plant identification process and thus making plant-watching easier.

CHAPTER 2

LITERATURE REVIEW

2.1. Introduction:

This section explains the previous research work which have been already done in classification using ML techniques. The data which is used in this study work is the data of customers of a Portuguese banking institution. The similar data set has been used in Moro et al. (2011, 2014). In Moro et al. (2011), the aim of this study was to find the model that can increase the success rate of telemarketing for the bank. The statistical techniques of data mining which have been used in their research are Support Vector Machine (SVM), Decision Tree (DT) and Naive Bayes. The performance of these models was checked through the Receiver Operator Characteristics (ROC) curve (detail of ROC curve is given in section 5). Among all these statistical techniques, SVM comes up with the most efficient results. Regarding attributes, Call duration was the most relevant feature which states that longer calls tend increase the success rate. After that month of contact, number of contacts, days since last contact, last contact result and first contact duration attributes respectively. In Moro et al. (2014), objective of the study was to predict the success of bank telemarketing. Data set which they used in their research was consists of 150 attributes and is complete data 3 set of the period 2008 to 2013. They compare the 4 data mining models i.e. Logistic Regression (LR), Decision Tree, Support Vector Machine and Neural Network (NN). The best result was obtained by the neural network while decision trees discloses that probability of success in inbound calls are greater. Statistical learning algorithms have successfully been used in many research problems for classification. For example, Qi et al. (2018) conducted a research to find out the fault diagnosis system for reciprocating compressors. Reciprocating compressors are extensively used in petroleum industry. Data was taken from oil corporation (5 years operational data) and uses the Support Vector Machine to analyze it. They come up with the results that SVM accurately predicts the 80% right classification to find the potential faults in compressor. Similarly, Gil & Johnsson (2010) did a research in medical field for diagnosing the urological dysfunctions using SVM. In this research data was collected from the 381 patients who are suffering from a number of urological dysfunctions to check the overall worth of decision support system. The fivefold cross validation has been utilized for the robustness. The outputs of this study describe that for the purpose of classification SVM attained the accuracy of 84.25% . Nogami et al. (1996) utilized the machine learning in decision support system. In their research they introduce the air traffic management for the future which can manage the flight schedule and flow of air traffic professionally. Their system involves many decision makers and utilized it with the neural network. They require such system which can make the optimal decision in the critical situation. Their simulation studies prove that system which is based on neural network is performed more efficiently than the current air traffic system. Another research by Cramer et al. (2017) the machine learning methods are used in time series for rainfall prediction. Data was derived from the 42 cities including climatic features.They tried Support vector regression, NN, and k nearest neighbors. After performing these methods they come up with the results that machine learning methods have predictive accuracy. Wang & Summers (2012) used the machine learning in field of radiology. They used it for the neurological disease diagnosis images, medical image segmentation and MRI images. They come-up with the results that machine learning identifies the complex patterns. It also helps the radiologists to make right decisions. Furthermore, they suggest that development of technology in machine learning is an asset for long term in the field of radiology. Machine learning algorithms are also used in the field of applied mathematics. For instance, Barboza et al. (2017) did a research to predict the models for developing of bankruptcy by using the SVM and random forest methods. The data was taken from the Salomon Center database & Compustat about North American firms from period 1985 to 2013 with observations of more than 10,000. After applying SVM and RF techniques they compare the results with the ordinary used methods such as discriminant analysis and logistic regression. They concluded that ML techniques are come up with 10% averagely more accurate results than usual methods. To find the risk factors about failure of banks Le & Viviani (2017) conducted a research. In their study, a sample of 3000 US banks was analyzed by using 2 traditional statistical methods i.e. discriminant analysis and logistics regression. Then they compare these methods with the machine learning methods i.e. SVM, ANN and k-nearest neighbors. The results of this study illustrate that ANN and k-neighbors method gives the accurate predictions as compared to the 4 traditional methods.

CHAPTER 3

OBJECTIVES

The purpose of this study is to check the accuracy and performance of several models for classification. The full model which is used in this study consists of 16 independent variables. Feature selection approach has been used to select the best subsets of variables and then different type of classification algorithms have been utilized to check their accuracy and performance. The full model is then compared with the reduced model, obtained through feature selection, in terms of classificaion accuracy.

CHAPTER 4

WORKFLOW DIAGRAM

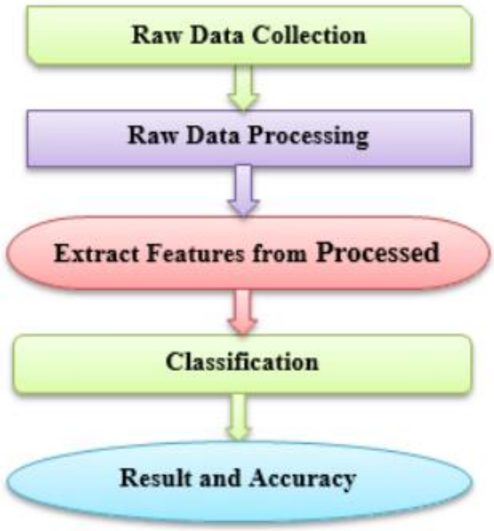


Figure 1: Workflow diagram

CHAPTER 5

METHODOLOGY

The main purpose of the code is to predict the outcome of the bank marketing campaign

The proposed approach is divided into four stages—retrieving data from datasets, pre-processing of data to improve the quality, feature extraction and using machine learning algorithms on extracted features for classifying the data patterns

A. BROWSE THROUGH DATA

The Aarhus University Signal Processing group, in collaboration with University of Southern Denmark, has recently released a dataset containing images of approximately 960 unique plants belonging to 12 species at several growth stages. It consists of 11,788 pictures and annotations such as 312 binary attributes, 15 component positions, 1 bounding box.

B. Pre-processing

Pre-processing developed a gray scale image dataset that is used to pixel-by-pixel image recognition and image size reductions. Then, these functions are aggregated and forwarded to the classifier. This increased processing time while retaining quality of the image.

The above fig. visualises target distribution of each plant category

C. Modelling

3 models are deployed- inception_v3, vgg_16, resnet_50. The models are compiled and fitted to the validated data. 10 Epochs are run. Loss and Accuracy are plotted on 2 separate graphs showing train and valid curves. A confusion matrix is displayed for each model.

D. Evaluation

A score sheet is generated on the basis of the models and will be created with the aid of the score sheet output. The Loss and Accuracy curves of each model is compared by plotting them on 2 separate graphs.

CHAPTER 6

PROJECT CODE

6.1 Code

```
import numpy as np # linear algebra

import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)

# Input data files are available in the read-only "../input/" directory

# For example, running this (by clicking run or pressing Shift+Enter) will list all files under the input directory

import os

for dirname, _, filenames in os.walk('/kaggle/input'):

    for filename in filenames:

        print(os.path.join(dirname, filename))



import numpy as np

import pandas as pd

import matplotlib.pyplot as plt
```

import seaborn as sns

```
import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)

# Input data files are available in the read-only "../input/" directory
# For example, running this (by clicking run or pressing Shift+Enter) will list all files under the input directory

import os
for dirname, _, filenames in os.walk('/kaggle/input'):
    for filename in filenames:
        print(os.path.join(dirname, filename))

outside of the current session
/kaggle/input/bank-marketing-campaigns-dataset/bank-additional-full.csv
```

In [2]:

```
#import required basic libraries
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

In [3]:

```
filename = '/kaggle/input/bank-marketing-campaigns-dataset/bank-additional-full.csv'
df = pd.read_csv(filename,sep=";")
df.head()
```

Out[3]:

	age	job	marital	education	default	housing	loan	contact	month	day_of_week	...	campaign	pdays	previous	poutcome	emp.var.rate	cons.price.idx	cons.conf.idx	euribor3m	nr.employed	y
0	56	housemaid	married	basic.4y	no	no	no	telephone	may	mon	...	1	999	0	nonexistent	1.1	93.994	-36.4	4.857	5191.0	no
1	57	services	married	high.school	unknown	no	no	telephone	may	mon	...	1	999	0	nonexistent	1.1	93.994	-36.4	4.857	5191.0	no

	age	job	marital	education	default	housing	loan	contact	month	day_of_week	...	campaign	pdays	previous	poutcome	emp.var.rate	cons.price.idx	cons.conf.idx	euribor3m	nr.employed	y
2	37	services	married	high.school	no	yes	no	telephone	may	mon	...	1	999	0	nonexistent	1.1	93.994	-36.4	4.857	5191.0	no
3	40	admin.	married	basic.6y	no	no	no	telephone	may	mon	...	1	999	0	nonexistent	1.1	93.994	-36.4	4.857	5191.0	no
4	56	services	married	high.school	no	no	yes	telephone	may	mon	...	1	999	0	nonexistent	1.1	93.994	-36.4	4.857	5191.0	no

5 rows × 21 columns

PRIMARY ANALYSIS OF CATEGORICAL FEATURES

In [4]:

```
df.info()
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 41188 entries, 0 to 41187
Data columns (total 21 columns):
 #   Column          Non-Null Count  Dtype  
---  ------          --------------  -----  
 0   age             41188 non-null  int64  
 1   job             41188 non-null  object 
 2   marital         41188 non-null  object 
 3   education       41188 non-null  object 
 4   default         41188 non-null  object 
 5   housing         41188 non-null  object 
 6   loan            41188 non-null  object 
 7   contact         41188 non-null  object 
 8   month           41188 non-null  object 
 9   day_of_week     41188 non-null  object 
 10  duration        41188 non-null  int64  
 11  campaign        41188 non-null  int64  
 12  pdays           41188 non-null  int64  
```

```
 13  previous        41188 non-null  int64
 14  poutcome        41188 non-null  object
 15  emp.var.rate    41188 non-null  float64
 16  cons.price.idx  41188 non-null  float64
 17  cons.conf.idx   41188 non-null  float64
 18  euribor3m       41188 non-null  float64
 19  nr.employed     41188 non-null  float64
 20  y               41188 non-null  object
dtypes: float64(5), int64(5), object(11)
memory usage: 6.6+ MB
```

In [5]:

```
#Numerical statistical summary
df.describe()
```

Out[5]:

	age	duration	campaign	pdays	previous	emp.var.rate	cons.price.idx	cons.conf.idx	euribor3m	nr.employed
count	41188.00000	41188.000000	41188.000000	41188.000000	41188.000000	41188.000000	41188.000000	41188.000000	41188.000000	41188.000000
mean	40.02406	258.285010	2.567593	962.475454	0.172963	0.081886	93.575664	-40.502600	3.621291	5167.035911
std	10.42125	259.279249	2.770014	186.910907	0.494901	1.570960	0.578840	4.628198	1.734447	72.251528
min	17.00000	0.000000	1.000000	0.000000	0.000000	-3.400000	92.201000	-50.800000	0.634000	4963.600000
25%	32.00000	102.000000	1.000000	999.000000	0.000000	-1.800000	93.075000	-42.700000	1.344000	5099.100000
50%	38.00000	180.000000	2.000000	999.000000	0.000000	1.100000	93.749000	-41.800000	4.857000	5191.000000
75%	47.00000	319.000000	3.000000	999.000000	0.000000	1.400000	93.994000	-36.400000	4.961000	5228.100000

	age	duration	campaign	pdays	previous	emp.var.rate	cons.price.idx	cons.conf.idx	euribor3m	nr.employed
max	98.00000	4918.000000	56.000000	999.000000	7.000000	1.400000	94.767000	-26.900000	5.045000	5228.100000

In [6]:

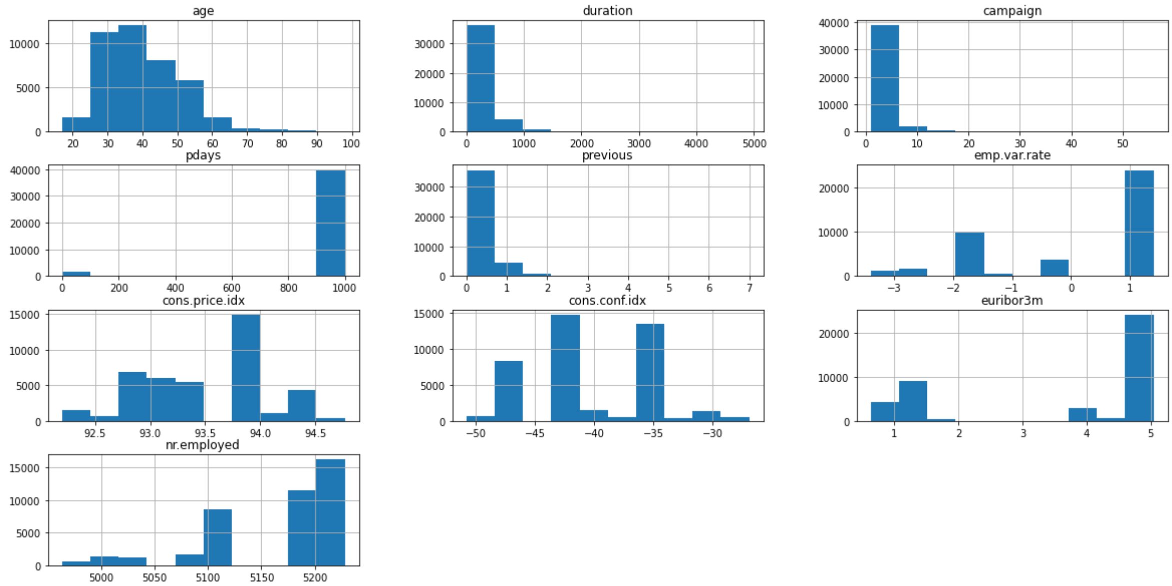
```
# Categorical statistical summary
df.describe(include=object)
```

Out[6]:

	job	marital	education	default	housing	loan	contact	month	day_of_week	poutcome	y
count	41188	41188	41188	41188	41188	41188	41188	41188	41188	41188	41188
unique	12	4	8	3	3	3	2	10	5	3	2
top	admin.	married	university.degree	no	yes	no	cellular	may	thu	nonexistent	no
freq	10422	24928	12168	32588	21576	33950	26144	13769	8623	35563	36548

In [7]:

```
# Creating histogram
df.hist(figsize= [20,10])
plt.show()
```



Above primary analysis shows that:

1) the data covers age groups from 98 to 17 years with mean of 40 years;30% of the clients are graduates; 52% of clients have taken housing loan and 82% have no personal loan.

2) Target variable y is categorical and has two categories- 'Yes' or 'No', hence this is a Classification Project.These two categories can be converted to binary and thus label encoding it. Maximum category found is 'No'.

3) Other independent categorical variables- job, marital, education, default, housing, loan, contact, month, day_of_week, poutcome- can be one hot encoded

4) Based on count, there are no missing values as such

MISSING VALUE ANALYSIS

In [8]:

```
df.isnull().sum()
```

Out[8]:

```
age               0
job               0
marital           0
education         0
default           0
housing           0
loan              0
contact           0
month             0
day_of_week       0
duration          0
campaign          0
pdays             0
previous          0
poutcome          0
emp.var.rate      0
cons.price.idx    0
cons.conf.idx     0
euribor3m         0
nr.employed       0
```

```
y                  0
dtype: int64
```

LABEL ENCODING

In [9]:

```
# Label encoding target variable
# converts label/words to numeric form without affecting dimensionality
# y- yes=1, no=0
df['y'] = df['y'].replace('yes', 1)
df['y'] = df['y'].replace('no', 0)
df.head()
```

Out[9]:

	age	job	marital	education	default	housing	loan	contact	month	day_of_week	...	campaign	pdays	previous	poutcome	emp.var.rate	cons.price.idx	cons.conf.idx	euribor3m	nr.employed	y
0	56	housemaid	married	basic.4y	no	no	no	telephone	may	mon	...	1	999	0	nonexistent	1.1	93.994	-36.4	4.857	5191.0	0
1	57	services	married	high.school	unknown	no	no	telephone	may	mon	...	1	999	0	nonexistent	1.1	93.994	-36.4	4.857	5191.0	0
2	37	services	married	high.school	no	yes	no	telephone	may	mon	...	1	999	0	nonexistent	1.1	93.994	-36.4	4.857	5191.0	0
3	40	admin.	married	basic.6y	no	no	no	telephone	may	mon	...	1	999	0	nonexistent	1.1	93.994	-36.4	4.857	5191.0	0
4	56	servi	marri	high.sc	no	no	yes	telep	may	mon	...	1	999	0	nonexis	1.1	93.994	-36.4	4.857	5191.0	0

	age	job	marital	education	default	housing	loan	contact	month	day_of_week	...	campaign	pdays	previous	poutcome	emp.var.rate	cons.price.idx	cons.conf.idx	euribor3m	nr.employed	y
		ces	ed	hol				hone							tent						

5 rows × 21 columns

In [10]:

```
# object datatypes are chosen as categorical datatypes
# one hot encoding represents the categorical variables as binary, increasing t
he dimensionality of the dataset
cat_col=[col for col in df.columns.values if df[col].dtype=='object']

# sepearting the numerical and categorical feature
df_cat=df[cat_col]
df_num= df.drop(cat_col,axis=1)
```

In [11]:

```
#dummy encoding the categorical features
df_cat_dum= pd.get_dummies(df_cat,drop_first=True)
```

In [12]:

```
df_features=pd.concat([df_num,df_cat_dum], axis=1)
```

In [13]:

```
df_features.head()
```

Out[13]:

	age	duration	campaign	pdays	previous	emp.var.rate	cons.price.idx	cons.conf.idx	euribor3m	nr.employed	...	month_may	month_nov	month_oct	month_sep	day_of_week_mon	day_of_week_thu	day_of_week_tue	day_of_week_wed	poutcome_nonexistent	poutcome_success
0	56	261	1	999	0	1.1	93.994	-36.4	4.857	5191.0	...	1	0	0	0	1	0	0	0	1	0
1	57	149	1	999	0	1.1	93.994	-36.4	4.857	5191.0	...	1	0	0	0	1	0	0	0	1	0

	age	duration	campaign	pdays	previous	emp.var.rate	cons.price.idx	cons.conf.idx	euribor3m	nr.employed	...	month_may	month_nov	month_oct	month_sep	day_of_week_mon	day_of_week_thu	day_of_week_tue	day_of_week_wed	poutcome_nonexistent	poutcome_success
2	37	226	1	999	0	1.1	93.994	-36.4	4.857	5191.0	...	1	0	0	0	1	0	0	0	1	0
3	40	151	1	999	0	1.1	93.994	-36.4	4.857	5191.0	...	1	0	0	0	1	0	0	0	1	0
4	56	307	1	999	0	1.1	93.994	-36.4	4.857	5191.0	...	1	0	0	0	1	0	0	0	1	0

5 rows × 54 columns

In [14]:

```
df_features.info()
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 41188 entries, 0 to 41187
Data columns (total 54 columns):
 #   Column                         Non-Null Count  Dtype
---  ------                         --------------  -----
 0   age                            41188 non-null  int64
 1   duration                       41188 non-null  int64
 2   campaign                       41188 non-null  int64
 3   pdays                          41188 non-null  int64
 4   previous                       41188 non-null  int64
 5   emp.var.rate                   41188 non-null  float64
 6   cons.price.idx                 41188 non-null  float64
 7   cons.conf.idx                  41188 non-null  float64
 8   euribor3m                      41188 non-null  float64
 9   nr.employed                    41188 non-null  float64
 10  y                              41188 non-null  int64
 11  job_blue-collar                41188 non-null  uint8
 12  job_entrepreneur               41188 non-null  uint8
 13  job_housemaid                  41188 non-null  uint8
 14  job_management                 41188 non-null  uint8
 15  job_retired                    41188 non-null  uint8
```

```
 16  job_self-employed              41188 non-null  uint8
 17  job_services                   41188 non-null  uint8
 18  job_student                    41188 non-null  uint8
 19  job_technician                 41188 non-null  uint8
 20  job_unemployed                 41188 non-null  uint8
 21  job_unknown                    41188 non-null  uint8
 22  marital_married                41188 non-null  uint8
 23  marital_single                 41188 non-null  uint8
 24  marital_unknown                41188 non-null  uint8
 25  education_basic.6y             41188 non-null  uint8
 26  education_basic.9y             41188 non-null  uint8
 27  education_high.school          41188 non-null  uint8
 28  education_illiterate           41188 non-null  uint8
 29  education_professional.course  41188 non-null  uint8
 30  education_university.degree    41188 non-null  uint8
 31  education_unknown              41188 non-null  uint8
 32  default_unknown                41188 non-null  uint8
 33  default_yes                    41188 non-null  uint8
 34  housing_unknown                41188 non-null  uint8
 35  housing_yes                    41188 non-null  uint8
 36  loan_unknown                   41188 non-null  uint8
 37  loan_yes                       41188 non-null  uint8
 38  contact_telephone              41188 non-null  uint8
 39  month_aug                      41188 non-null  uint8
 40  month_dec                      41188 non-null  uint8
 41  month_jul                      41188 non-null  uint8
 42  month_jun                      41188 non-null  uint8
 43  month_mar                      41188 non-null  uint8
 44  month_may                      41188 non-null  uint8
 45  month_nov                      41188 non-null  uint8
 46  month_oct                      41188 non-null  uint8
 47  month_sep                      41188 non-null  uint8
 48  day_of_week_mon                41188 non-null  uint8
 49  day_of_week_thu                41188 non-null  uint8
 50  day_of_week_tue                41188 non-null  uint8
 51  day_of_week_wed                41188 non-null  uint8
 52  poutcome_nonexistent           41188 non-null  uint8
 53  poutcome_success               41188 non-null  uint8
dtypes: float64(5), int64(6), uint8(43)
memory usage: 5.1 MB
```

FEATURE SELECTION USING RANDOM CLASSIFIER

In [15]:

```
# splitting the data into 70% training data and 30% test data

from sklearn.model_selection import train_test_split

X = df_features.drop(['y'], axis=1)
y = df_features['y']

trainx, testx, trainy, testy = train_test_split(X, y, test_size=0.3, random_sta
te=0)
```

In [16]:

```
# Use randomforest classifier

from sklearn.ensemble import RandomForestClassifier
```

```
rfc = RandomForestClassifier(n_estimators=10000, random_state=0, n_jobs=-1)

# Train the classifier
rfc.fit(trainx, trainy)
```

Out[16]:

```
RandomForestClassifier(n_estimators=10000, n_jobs=-1, random_state=0)
```

In [17]:

```
# Print the name and gini importance of each feature
feat_labels = X.columns.values

feature_importance = []
for feature in zip(feat_labels, rfc.feature_importances_):

    feature_importance.append(feature)
```

In [18]:

```
feature_importance
```

Out[18]:

```
[('age', 0.08618566386398764),
 ('duration', 0.29015602499077386),
 ('campaign', 0.04178199369579807),
 ('pdays', 0.03353857431900278),
 ('previous', 0.013738961767318078),
 ('emp.var.rate', 0.02341289011754231),
 ('cons.price.idx', 0.023455586654047598),
 ('cons.conf.idx', 0.026641943503223915),
 ('euribor3m', 0.10042013017159576),
 ('nr.employed', 0.052001436527475416),
 ('job_blue-collar', 0.00912955901482597),
 ('job_entrepreneur', 0.00408065698430441),
 ('job_housemaid', 0.0034507775674576468),
 ('job_management', 0.007619194267427778),
 ('job_retired', 0.006231741559246316),
 ('job_self-employed', 0.004297209231449068),
 ('job_services', 0.007059908897013911),
 ('job_student', 0.004545515313735655),
 ('job_technician', 0.011362997000859581),
 ('job_unemployed', 0.004234164752530876),
 ('job_unknown', 0.0017352275969970802),
 ('marital_married', 0.013703674878066122),
 ('marital_single', 0.011980118740034),
 ('marital_unknown', 0.0004681802865129859),
 ('education_basic.6y', 0.004533082261449234),
 ('education_basic.9y', 0.00841360183841855),
 ('education_high.school', 0.011911672920323208),
 ('education_illiterate', 0.00019547250437647l9),
 ('education_professional.course', 0.008862687930887311),
 ('education_university.degree', 0.013145967023621927),
 ('education_unknown', 0.005204166737204258),
 ('default_unknown', 0.008828277503150218),
 ('default_yes', 7.4083525258815036e-09),
 ('housing_unknown', 0.002209251379755515),
 ('housing_yes', 0.019947814943109477),
 ('loan_unknown', 0.0022258111044771538),
 ('loan_yes', 0.013294383844967334),
 ('contact_telephone', 0.010349589587515364),
 ('month_aug', 0.002542882713933205),
```

```
 ('month_dec', 0.00089080425655123),
 ('month_jul', 0.0026645019351060383),
 ('month_jun', 0.0029011522955605406),
 ('month_mar', 0.004887583078879729),
 ('month_may', 0.005116610052046957),
 ('month_nov', 0.0022956282126332097),
 ('month_oct', 0.005765904989179848),
 ('month_sep', 0.002176686695096346),
 ('day_of_week_mon', 0.01204060407787753),
 ('day_of_week_thu', 0.01225680296198673),
 ('day_of_week_tue', 0.01190380175245775),
 ('day_of_week_wed', 0.01185356865483465),
 ('poutcome_nonexistent', 0.008630332656024656),
 ('poutcome_success', 0.02371921697899626)]
```

In [19]:

```
# Create a selector object that will use the random forest classifier to identi
fy
# features that have an importance of more than 0.01
from sklearn.feature_selection import SelectFromModel

sfm = SelectFromModel(rfc, threshold=0.01)

# Train the selector
sfm.fit(trainx, trainy)
```

Out[19]:

```
SelectFromModel(estimator=RandomForestClassifier(n_estimators=10000, n_jobs=
-1,
                                                 random_state=0),
                threshold=0.01)
```

In [20]:

```
# Print the names of the most important features
selected_features = []
for feature_list_index in sfm.get_support(indices=True):
    selected_features.append(feat_labels[feature_list_index])
```

In [21]:

```
selected_features
```

Out[21]:

```
['age',
 'duration',
 'campaign',
 'pdays',
 'previous',
 'emp.var.rate',
 'cons.price.idx',
 'cons.conf.idx',
 'euribor3m',
 'nr.employed',
 'job_technician',
 'marital_married',
 'marital_single',
 'education_high.school',
 'education_university.degree',
 'housing_yes',
 'loan_yes',
 'contact_telephone',
 'day_of_week_mon',
```

```
 'day_of_week_thu',
 'day_of_week_tue',
 'day_of_week_wed',
 'poutcome_success']
```

23 features have been selected using RandomForestClassifier for further modelling

In [22]:

```
data_selected = df_features[selected_features]
data_selected.head()
```

Out[22]:

	age	duration	campaign	pdays	previous	emp.var.rate	cons.price.idx	cons.conf.idx	euribor3m	nr.employed	...	education_high.school	education_university.degree	housing_yes	loan_yes	contact_telephone	day_of_week_mon	day_of_week_thu	day_of_week_tue	day_of_week_wed	poutcome_success
0	56	261	1	999	0	1.1	93.994	-36.4	4.857	5191.0	...	0	0	0	0	1	1	0	0	0	0
1	57	149	1	999	0	1.1	93.994	-36.4	4.857	5191.0	...	1	0	0	0	1	1	0	0	0	0
2	37	226	1	999	0	1.1	93.994	-36.4	4.857	5191.0	...	1	0	1	0	1	1	0	0	0	0
3	40	151	1	999	0	1.1	93.994	-36.4	4.857	5191.0	...	0	0	0	0	1	1	0	0	0	0
4	56	307	1	999	0	1.1	93.994	-36.4	4.857	5191.0	...	1	0	0	1	1	1	0	0	0	0

5 rows × 23 columns

In [23]:

```
data_selected.info()
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 41188 entries, 0 to 41187
Data columns (total 23 columns):
 #   Column                         Non-Null Count  Dtype  
---  ------                         --------------  -----  
 0   age                            41188 non-null  int64  
 1   duration                       41188 non-null  int64  
 2   campaign                       41188 non-null  int64  
 3   pdays                          41188 non-null  int64  
 4   previous                       41188 non-null  int64  
 5   emp.var.rate                   41188 non-null  float64
 6   cons.price.idx                 41188 non-null  float64
 7   cons.conf.idx                  41188 non-null  float64
 8   euribor3m                      41188 non-null  float64
 9   nr.employed                    41188 non-null  float64
 10  job_technician                 41188 non-null  uint8  
 11  marital_married                41188 non-null  uint8  
 12  marital_single                 41188 non-null  uint8  
 13  education_high.school          41188 non-null  uint8  
 14  education_university.degree    41188 non-null  uint8  
 15  housing_yes                    41188 non-null  uint8  
 16  loan_yes                       41188 non-null  uint8  
 17  contact_telephone              41188 non-null  uint8  
 18  day_of_week_mon                41188 non-null  uint8  
 19  day_of_week_thu                41188 non-null  uint8  
 20  day_of_week_tue                41188 non-null  uint8  
 21  day_of_week_wed                41188 non-null  uint8  
 22  poutcome_success               41188 non-null  uint8  
dtypes: float64(5), int64(5), uint8(13)
memory usage: 3.7 MB
```

STANDARDIZING THE DATA USING MINMAXSCALER

Since the features have different ranges, it needs to be scaled for building a better model. MinMaxScaler of skcit library scales and translates each feature individually such that it is in the given range on the training set, e.g. between zero and one. Default range is zero and one,which is being used below.

In [24]:

```
from sklearn.preprocessing import MinMaxScaler
scaler = MinMaxScaler()
scaler.fit(data_selected)
data_standardised = scaler.fit_transform(data_selected)
```

In [25]:

```
data_std= pd.DataFrame(data_standardised)
data_std.head()
```

Out[25]:

	0	1	2	3	4	5	6	7	8	9	...	13	14	15	16	17	18	19	20	21	22
0	0.481481	0.053070	0.0	1.0	0.0	0.9375	0.698753	0.60251	0.957379	0.859735	...	0.0	0.0	0.0	0.0	1.0	1.0	0.0	0.0	0.0	0.0
1	0.493827	0.030297	0.0	1.0	0.0	0.9375	0.698753	0.60251	0.957379	0.859735	...	1.0	0.0	0.0	0.0	1.0	1.0	0.0	0.0	0.0	0.0
2	0.246914	0.045954	0.0	1.0	0.0	0.9375	0.698753	0.60251	0.957379	0.859735	...	1.0	0.0	1.0	0.0	1.0	1.0	0.0	0.0	0.0	0.0
3	0.283951	0.030704	0.0	1.0	0.0	0.9375	0.698753	0.60251	0.957379	0.859735	...	0.0	0.0	0.0	0.0	1.0	1.0	0.0	0.0	0.0	0.0
4	0.481481	0.062424	0.0	1.0	0.0	0.9375	0.698753	0.60251	0.957379	0.859735	...	1.0	0.0	0.0	1.0	1.0	1.0	0.0	0.0	0.0	0.0

5 rows × 23 columns

In [26]:

```
data_std.info()
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 41188 entries, 0 to 41187
Data columns (total 23 columns):
 #   Column  Non-Null Count  Dtype  
---  ------  --------------  -----  
 0   0       41188 non-null  float64
 1   1       41188 non-null  float64
 2   2       41188 non-null  float64
 3   3       41188 non-null  float64
 4   4       41188 non-null  float64
 5   5       41188 non-null  float64
 6   6       41188 non-null  float64
 7   7       41188 non-null  float64
 8   8       41188 non-null  float64
 9   9       41188 non-null  float64
 10  10      41188 non-null  float64
 11  11      41188 non-null  float64
 12  12      41188 non-null  float64
 13  13      41188 non-null  float64
```

```
 14  14      41188 non-null  float64
 15  15      41188 non-null  float64
 16  16      41188 non-null  float64
 17  17      41188 non-null  float64
 18  18      41188 non-null  float64
 19  19      41188 non-null  float64
 20  20      41188 non-null  float64
 21  21      41188 non-null  float64
 22  22      41188 non-null  float64
dtypes: float64(23)
memory usage: 7.2 MB
```

BUILDING SUPERVISED MODELS

In [27]:

```
X= data_std
```

In [28]:

```
X.shape
```

Out[28]:

```
(41188, 23)
```

In [29]:

```
y= pd.DataFrame(df_features['y'])
```

In [30]:

```
y.shape
```

Out[30]:

```
(41188, 1)
```

In [31]:

```
y.info()
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 41188 entries, 0 to 41187
Data columns (total 1 columns):
 #   Column  Non-Null Count  Dtype
---  ------  --------------  -----
 0   y       41188 non-null  int64
dtypes: int64(1)
memory usage: 321.9 KB
```

In [32]:

```
# spilting the dataset into training and testing
trainx, testx, trainy, testy = train_test_split(X, y, test_size = 0.30, random_
state=10)

print("trainx ",trainx.shape)

print("testx ",testx.shape)

print("trainy ",trainy.shape)

print("testy ",testy.shape)
trainx  (28831, 23)
testx  (12357, 23)
trainy  (28831, 1)
testy  (12357, 1)
```

SIMPLE LOGISTIC REGRESSION

In [33]:

```
# logistic regression performs binary classifications, and gives probability ou
tput
```

```
import statsmodels.api as sm
from sklearn.linear_model import LogisticRegression
log_reg_model = sm.Logit(trainy, trainx).fit()
print(log_reg_model.summary())
Optimization terminated successfully.
         Current function value: 0.212454
         Iterations 8
                           Logit Regression Results
==============================================================================
==
Dep. Variable:                      y   No. Observations:                  288
31
Model:                          Logit   Df Residuals:                      288
08
Method:                           MLE   Df Model:
22
Date:                Sun, 20 Nov 2022   Pseudo R-squ.:                    0.39
28
Time:                        12:37:05   Log-Likelihood:                  -6125
.3
converged:                       True   LL-Null:                         -1008
7.
Covariance Type:            nonrobust   LLR p-value:                       0.0
00
==============================================================================
==
                 coef    std err          z      P>|z|      [0.025      0.97
5]
------------------------------------------------------------------------------
--
0              0.3918      0.182      2.153      0.031       0.035       0.7
48
1             22.5210      0.429     52.490      0.000      21.680      23.3
62
2             -2.0690      0.761     -2.717      0.007      -3.561      -0.5
77
3             -1.8481      0.195     -9.464      0.000      -2.231      -1.4
65
4             -2.3960      0.311     -7.708      0.000      -3.005      -1.7
87
5             -4.1217      0.395    -10.431      0.000      -4.896      -3.3
47
6              2.1751      0.288      7.562      0.000       1.611       2.7
39
7              0.6590      0.141      4.676      0.000       0.383       0.9
35
8              1.8401      0.369      4.986      0.000       1.117       2.5
63
9             -2.5501      0.365     -6.990      0.000      -3.265      -1.8
35
10             0.1221      0.067      1.811      0.070      -0.010       0.2
54
11            -0.1101      0.077     -1.422      0.155      -0.262       0.0
42
12             0.0806      0.087      0.923      0.356      -0.091       0.2
52
```

```
13              -0.0009      0.064     -0.014      0.989      -0.126       0.1
24
14               0.2482      0.056      4.405      0.000       0.138       0.3
59
15              -0.0016      0.048     -0.034      0.973      -0.096       0.0
93
16              -0.0456      0.068     -0.676      0.499      -0.178       0.0
87
17              -0.8051      0.074    -10.894      0.000      -0.950      -0.6
60
18              -0.0851      0.077     -1.101      0.271      -0.236       0.0
66
19               0.0647      0.075      0.859      0.390      -0.083       0.2
12
20               0.0908      0.077      1.177      0.239      -0.060       0.2
42
21               0.0932      0.077      1.207      0.228      -0.058       0.2
45
22               0.1414      0.201      0.702      0.483      -0.253       0.5
36
==============================================================================
==
```

It is noticed that 8 iterations were performed. p value near zero or low indicates the model is statistically alright. Moving forward with prediicting y and then evaluation matrix

In [34]:

```
# let 'y_pred_prob' be the predicted values of y
y_pred_prob = log_reg_model.predict(testx)

# print the y_pred_prob
y_pred_prob.head()
```

Out[34]:

```
29773    0.062592
14070    0.068129
39364    0.450767
29279    0.061301
11888    0.003388
dtype: float64
```

In [35]:

```
# convert probabilities to 0 and 1 using 'if_else'
predy = ['0' if x < 0.5 else '1' for x in y_pred_prob]
# convert the predicted values to type 'float32'
predy = np.array(predy, dtype=np.float32)

# print the first five predictions
predy[0:5]
```

Out[35]:

```
array([0., 0., 0., 0., 0.], dtype=float32)
```

The above process of considering probability is done because confusion matrix doesnot provide result when continuos and binary data is fed into it.

In [36]:

```
# Evaluation Metrics 1- Confusion matrics
from sklearn import metrics
from sklearn.metrics import confusion_matrix
cf1=pd.DataFrame(confusion_matrix(testy,predy),columns=['Predicted 0','Predicte
d 1'], index =['Actual 0','Actual 1'])
```

```
cf1
```

Out[36]:

	Predicted 0	Predicted 1
Actual 0	10657	278
Actual 1	859	563

In [37]:

```
# Evaluation Metrics 2- Accuracy using Classification Report
from sklearn.metrics import classification_report
test_report1 = classification_report(testy,predy)
print(test_report1)
              precision    recall  f1-score   support

           0       0.93      0.97      0.95     10935
           1       0.67      0.40      0.50      1422

    accuracy                           0.91     12357
   macro avg       0.80      0.69      0.72     12357
weighted avg       0.90      0.91      0.90     12357
```

In [38]:

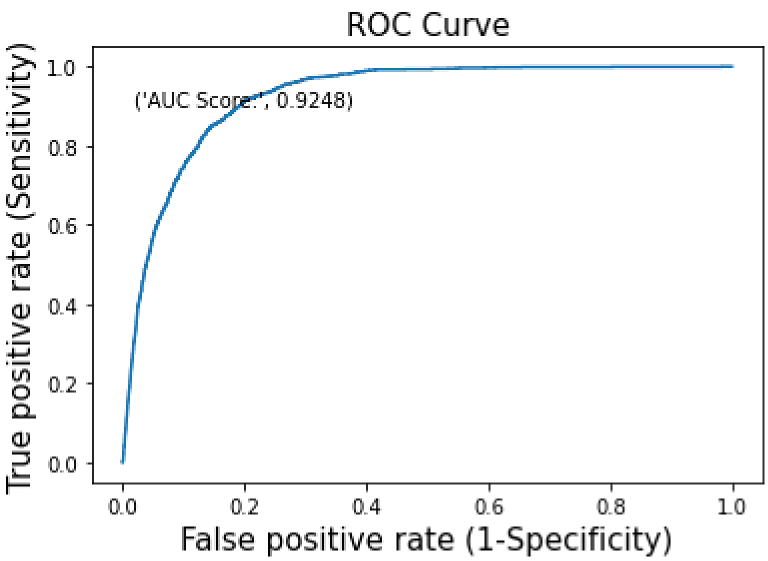
```
# Evaluation Metrics 3- Cohen value using Kappa score
from sklearn.metrics import cohen_kappa_score
kappa_value1 = cohen_kappa_score(testy, predy)
print(kappa_value1)
0.45057636476835083
```

In [39]:

```
# Evaluation metrics 4- Plot the ROC curve to get AUC score

from sklearn.metrics import roc_auc_score
from sklearn.metrics import roc_curve

fpr, tpr, thresholds = roc_curve(testy,y_pred_prob)
plt.plot(fpr, tpr)

plt.title('ROC Curve', fontsize = 15)
plt.xlabel('False positive rate (1-Specificity)', fontsize = 15)
plt.ylabel('True positive rate (Sensitivity)', fontsize = 15)

plt.text(x = 0.02, y = 0.9, s = ('AUC Score:',round(roc_auc_score(testy, y_pred
_prob),4)))
```

Out[39]:

```
Text(0.02, 0.9, "('AUC Score:', 0.9248)")
```



In [40]:

```
# tabulate the results
score_card = pd.DataFrame(columns=['Model', 'AUC Score', 'Precision Score', 'Re
call Score', 'Accuracy Score',
                                   'Kappa Score', 'f1-score'])
score_card = score_card.append({'Model': 'Logistic Regresion',
                                    'AUC Score' : roc_auc_score(testy, y_pred_p
rob),
                                    'Precision Score': metrics.precision_score(
testy, predy),
                                    'Recall Score': metrics.recall_score(testy,
predy),
                                    'Accuracy Score': metrics.accuracy_score(te
sty, predy),
                                    'Kappa Score': cohen_kappa_score(testy, pre
dy),
                                    'f1-score': metrics.f1_score(testy, predy)}
,
                                    ignore_index = True)
score_card
```

Out[40]:

	Model	AUC Score	Precision Score	Recall Score	Accuracy Score	Kappa Score	f1-score
0	Logistic Regresion	0.924839	0.669441	0.395921	0.907987	0.450576	0.49757

ADABOOST

In [41]:

```
from sklearn.ensemble import AdaBoostClassifier

# build the model
adaboost = AdaBoostClassifier(random_state=10)
# fit the model
adaboost.fit(trainx, trainy)
```

```
/opt/conda/lib/python3.7/site-packages/sklearn/utils/validation.py:993: Data
ConversionWarning: A column-vector y was passed when a 1d array was expected
. Please change the shape of y to (n_samples, ), for example using ravel().
  y = column_or_1d(y, warn=True)
```

Out[41]:

```
AdaBoostClassifier(random_state=10)
```

In [42]:

```
y_pred_adaboost = adaboost.predict(testx)
```

In [43]:

```
# Evaluation Metrics 1- Confusion matrics
cf2=pd.DataFrame(confusion_matrix(testy,y_pred_adaboost),columns=['Predicted 0'
,'Predicted 1'], index =['Actual 0','Actual 1'])
cf2
```

Out[43]:

	Predicted 0	Predicted 1
Actual 0	10688	247
Actual 1	915	507

In [44]:

```
# Evaluation Metrics 2- Accuracy using Classification Report
test_report2 = classification_report(testy,y_pred_adaboost)
print(test_report2)
              precision    recall  f1-score   support

           0       0.92      0.98      0.95     10935
           1       0.67      0.36      0.47      1422

    accuracy                           0.91     12357
   macro avg       0.80      0.67      0.71     12357
weighted avg       0.89      0.91      0.89     12357
```

In [45]:

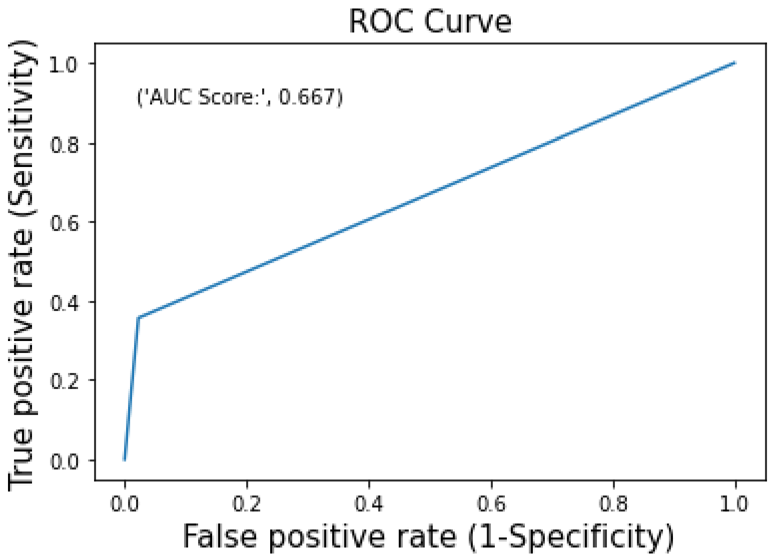
```
# Evaluation Metrics 3- Cohen value using Kappa score
kappa_value2 = cohen_kappa_score(testy, y_pred_adaboost)
print(kappa_value2)
0.4197151070930797
```

In [46]:

```
# Evaluation metrics 4- Plot the ROC curve to get AUC score
fpr, tpr, thresholds = roc_curve(testy, y_pred_adaboost)
plt.plot(fpr, tpr)

plt.title('ROC Curve', fontsize = 15)
plt.xlabel('False positive rate (1-Specificity)', fontsize = 15)
plt.ylabel('True positive rate (Sensitivity)', fontsize = 15)

plt.text(x = 0.02, y = 0.9, s = ('AUC Score:',round(roc_auc_score(testy, y_pred
_adaboost),4)))
```

Out[46]:

```
Text(0.02, 0.9, "('AUC Score:', 0.667)")
```



In [47]:

```
adaboost_metrics = pd.Series({'Model': "AdaBoost",
                     'AUC Score' : metrics.roc_auc_score(testy, y_pred_adaboost
),
                 'Precision Score': metrics.precision_score(testy, y_pred_adabo
ost),
                 'Recall Score': metrics.recall_score(testy, y_pred_adaboost),
                 'Accuracy Score': metrics.accuracy_score(testy, y_pred_adaboos
t),
                 'Kappa Score': cohen_kappa_score(testy, y_pred_adaboost),
                  'f1-score':metrics.f1_score(testy, y_pred_adaboost)})
score_card = score_card.append(adaboost_metrics , ignore_index = True)
score_card
```

Out[47]:

	Model	AUC Score	Precision Score	Recall Score	Accuracy Score	Kappa Score	f1-score
0	Logistic Regresion	0.924839	0.669441	0.395921	0.907987	0.450576	0.497570
1	AdaBoost	0.666976	0.672414	0.356540	0.905964	0.419715	0.465993

K-NEAREST NEIGHBOUR(KNN)

In [48]:

```
# KNN is classification algorithm that provides class output, default value of
n-neighbours = 5
from sklearn.neighbors import KNeighborsClassifier
knn_model=KNeighborsClassifier(n_neighbors=5).fit(trainx,trainy.values.ravel())
```

In [49]:

```
predy=knn_model.predict(testx)
```

In [50]:

```
# Evaluation Metrics 1- Confusion matrics
```

```
cf3=pd.DataFrame(confusion_matrix(testy,predy),columns=['Predicted 0','Predicte
d 1'], index =['Actual 0','Actual 1'])
cf3
```

Out[50]:

	Predicted 0	Predicted 1
Actual 0	10631	304
Actual 1	1050	372

In [51]:

```
# Evaluation Metrics 2- Accuracy using Classification Report
test_report3 = classification_report(testy,predy)
print(test_report3)
              precision    recall  f1-score   support

           0       0.91      0.97      0.94     10935
           1       0.55      0.26      0.35      1422

    accuracy                           0.89     12357
   macro avg       0.73      0.62      0.65     12357
weighted avg       0.87      0.89      0.87     12357
```

In [52]:

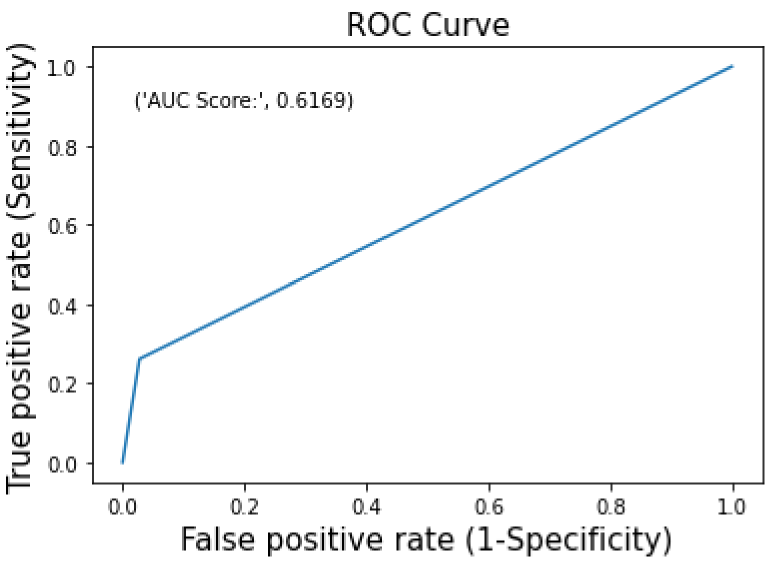
```
# Evaluation Metrics 3- Cohen value using Kappa score
kappa_value3 = cohen_kappa_score(testy, predy)
print(kappa_value3)
0.3029301768545051
```

In [53]:

```
# Evaluation metrics 4- Plot the ROC curve to get AUC score
fpr, tpr, thresholds = roc_curve(testy, predy)
plt.plot(fpr, tpr)

plt.title('ROC Curve', fontsize = 15)
plt.xlabel('False positive rate (1-Specificity)', fontsize = 15)
plt.ylabel('True positive rate (Sensitivity)', fontsize = 15)

plt.text(x = 0.02, y = 0.9, s = ('AUC Score:',round(roc_auc_score(testy, predy)
,4)))
```

Out[53]:

```
Text(0.02, 0.9, "('AUC Score:', 0.6169)")
```



In [54]:

```
knn_metrics = pd.Series({'Model': "KNN",
                     'AUC Score' : metrics.roc_auc_score(testy, predy),
                 'Precision Score': metrics.precision_score(testy, predy),
                 'Recall Score': metrics.recall_score(testy, predy),
                 'Accuracy Score': metrics.accuracy_score(testy,predy),
                 'Kappa Score': cohen_kappa_score(testy, predy),
                  'f1-score':metrics.f1_score(testy, predy)})
score_card = score_card.append(knn_metrics , ignore_index = True)
score_card
```

Out[54]:

	Model	AUC Score	Precision Score	Recall Score	Accuracy Score	Kappa Score	f1-score
0	Logistic Regresion	0.924839	0.669441	0.395921	0.907987	0.450576	0.497570
1	AdaBoost	0.666976	0.672414	0.356540	0.905964	0.419715	0.465993
2	KNN	0.616901	0.550296	0.261603	0.890426	0.302930	0.354623

SUPPORT VECTOR MACHINE(SVM)

In [55]:

```
# provides class output
from sklearn.svm import SVC
from sklearn import linear_model
svm_lin_model= SVC(kernel='linear').fit(trainx,trainy.values.ravel())
```

In [56]:

```
predy=svm_lin_model.predict(testx)
```

In [57]:

```
# Evaluation Metrics 1- Confusion matrics
```

```
cf4=pd.DataFrame(confusion_matrix(testy,predy),columns=['Predicted 0','Predicte
d 1'], index =['Actual 0','Actual 1'])
cf4
```

Out[57]:

	Predicted 0	Predicted 1
Actual 0	10750	185
Actual 1	1091	331

In [58]:

```
# Evaluation Metrics 2- Accuracy using Classification Report
test_report4 = classification_report(testy,predy)
print(test_report4)
              precision    recall  f1-score   support

           0       0.91      0.98      0.94     10935
           1       0.64      0.23      0.34      1422

    accuracy                           0.90     12357
   macro avg       0.77      0.61      0.64     12357
weighted avg       0.88      0.90      0.87     12357
```

In [59]:

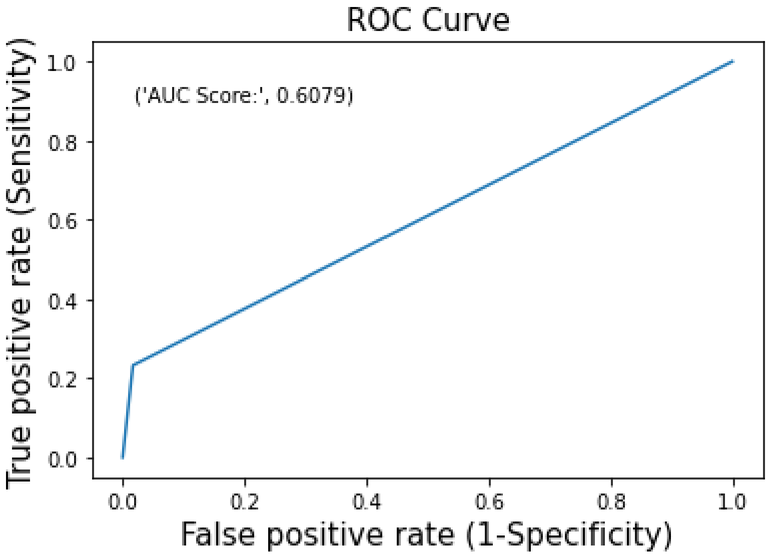
```
# Evaluation Metrics 3- Cohen value using Kappa score
kappa_value4 = cohen_kappa_score(testy, predy)
print(kappa_value4)
0.29860862560843104
```

In [60]:

```
# Evaluation metrics 4- Plot the ROC curve to get AUC score
fpr, tpr, thresholds = roc_curve(testy, predy)
plt.plot(fpr, tpr)

plt.title('ROC Curve', fontsize = 15)
plt.xlabel('False positive rate (1-Specificity)', fontsize = 15)
plt.ylabel('True positive rate (Sensitivity)', fontsize = 15)

plt.text(x = 0.02, y = 0.9, s = ('AUC Score:',round(roc_auc_score(testy, predy)
,4)))
```

Out[60]:

```
Text(0.02, 0.9, "('AUC Score:', 0.6079)")
```



In [61]:

```
svm_metrics = pd.Series({'Model': "SVM",
                     'AUC Score' : metrics.roc_auc_score(testy, predy),
                 'Precision Score': metrics.precision_score(testy, predy),
                 'Recall Score': metrics.recall_score(testy, predy),
                 'Accuracy Score': metrics.accuracy_score(testy,predy),
                 'Kappa Score': cohen_kappa_score(testy, predy),
                  'f1-score':metrics.f1_score(testy, predy)})
score_card = score_card.append(svm_metrics , ignore_index = True)
score_card
```

Out[61]:

	Model	AUC Score	Precision Score	Recall Score	Accuracy Score	Kappa Score	f1-score
0	Logistic Regresion	0.924839	0.669441	0.395921	0.907987	0.450576	0.497570
1	AdaBoost	0.666976	0.672414	0.356540	0.905964	0.419715	0.465993
2	KNN	0.616901	0.550296	0.261603	0.890426	0.302930	0.354623
3	SVM	0.607926	0.641473	0.232771	0.896739	0.298609	0.341589

AUC score of Logistic Regression is high which makes it a good model for prediction. But higher precision, recall, accuracy, kappa and f1 score makes Adaboost the better model in comparision with all the others.

CHAPTER 7

REQUIREMENTS

7.1 Hardware requirements

Processor : Intel Multicore Processor (i3 or i5 or i7)

RAM : 4GB or Above

Hard Disk : 100GB or Above

7.2 Software Requirements

Programming Language : Python

Operating System : Windows or Linux

Tools : Anaconda Navigator,Tensorflow,Keras

TENSORFLOW:

The standard name for Machine Learning in the Data Science industry is TensorFlow. It facilitates building of both statistical Machine Learning solutions as well as deep learning through its extensive interface of CUDA GPUs. The most basic data type of TensorFlow is a tensor which is a multi-dimensional array.

It is an open-source toolkit that can be used for build machine learning pipelines so that you can build scalable systems to process data. It provides support and functions for various applications of ML such as Computer Vision, NLP and Reinforcement Learning. TensorFlow is one of the must-know tools of Machine Learning for beginners.

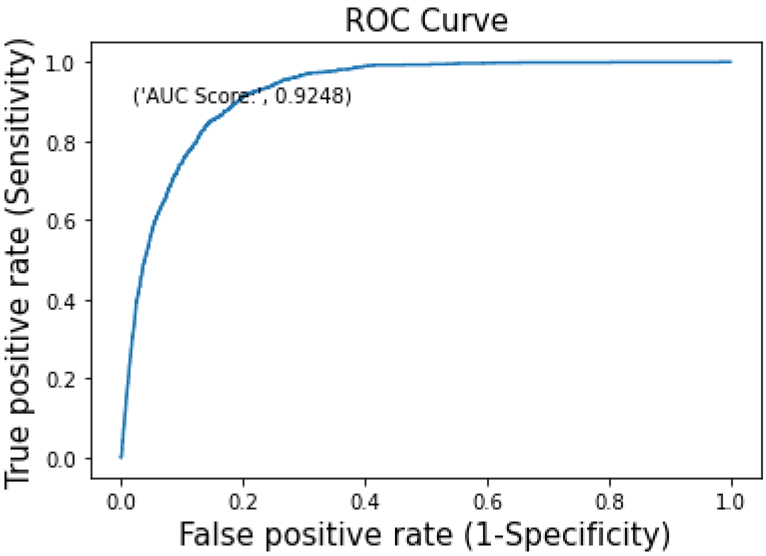
KERAS:

Keras is an open-source neural network library that provides support for Python. It is popular for its modularity, speed, and ease of use. Therefore, it can be used for fast experimentation as well as rapid prototyping. It provides support for the implementation of convolutional neural networks, Recurrent Neural Networks as well as both. It is capable of running seamlessly on the CPU and GPU. Compared to more widely popular libraries like

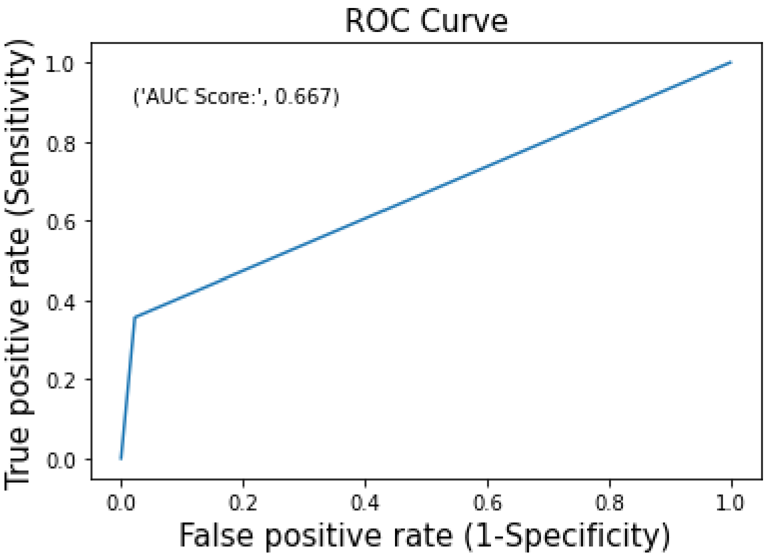
TensorFlow and Pytorch, Keras provides user-friendliness that allows the users to readily implement neural networks without dwelling over the technical jargon.

CHAPTER 8

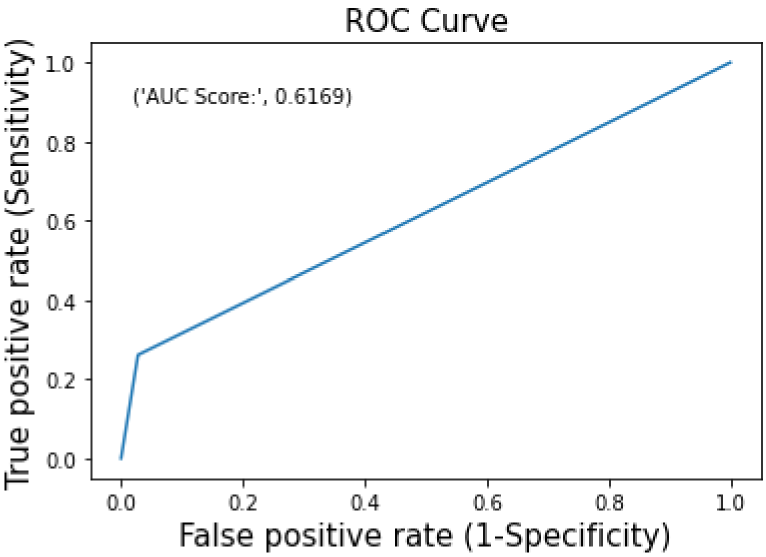
PROJECT FINDINGS



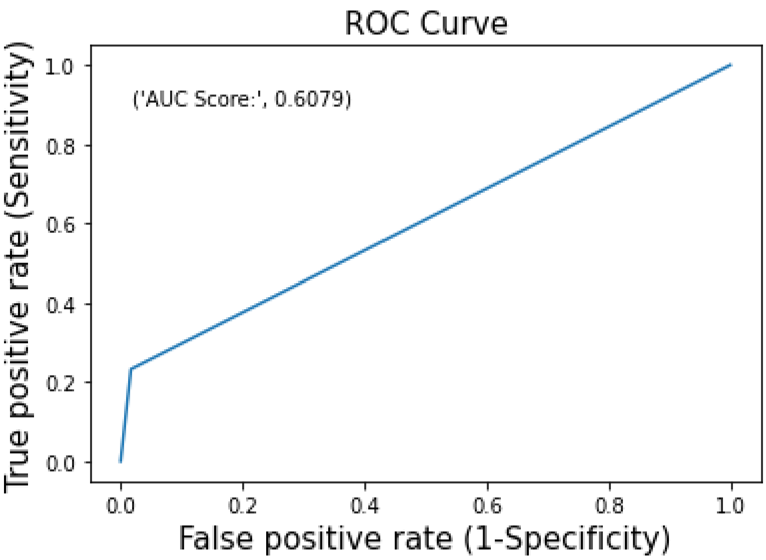
ROC CURVE OF – LOGISTIC REGRISSON



ROC CURVE OF – ADABOOST



ROC CURVE OF – KNN



ROC CURVE OF – SVM

	Model	AUC Score	Precision Score	Recall Score	Accuracy Score	Kappa Score	f1-score
0	Logistic Regresion	0.924839	0.669441	0.395921	0.907987	0.450576	0.497570
1	AdaBoost	0.666976	0.672414	0.356540	0.905964	0.419715	0.465993
2	KNN	0.616901	0.550296	0.261603	0.890426	0.302930	0.354623
3	SVM	0.607926	0.641473	0.232771	0.896739	0.298609	0.341589

AUC score of Logistic Regression is high which makes it a good model for prediction. But higher precision, recall, accuracy, kappa and f1 score makes Adaboost the better model in comparision with all the others.

THE GIVEN TABLE GIVE THE ACCURACY OUTCOME OF ALL THE MODLES USED.

CHAPTER 9

CONCLUSION

Predition of outcome of bank marketing outcome is obtained using different models and the accuracy and outcome of different models is obtained.

CHAPTER 10

FUTURE ENHANCEMENTS

1) Create an android/iOS app which will be more convenient to user.
2) System can be implemented using cloud which can store large amount of data for comparison and provide high computing power for processing).