Predicting Potential Customers for Banks using Machine Learning Algorithms

BY

Abhinav Nair(19BPS1007)
Sparsh Raj (19BPS1028)
Hande Atharrva Devendra (19BPS1086)

A project report submitted to

Dr. Joshan Athanesious J

SCHOOL OF COMPUTER SCIENCE AND ENGINEERING

in partial fulfilment of the requirements for the course of

CSE4058 – Business Intelligence

VELLORE INSTITUTE OF TECHNOLOGY
Vandalur – Kelambakkam Road
CHENNAI - 600127
July 2022

Bonafide Certificate

This is to certify that the Project work titled “Smart Waste Management System” is being submitted by ***Abhinav Nair*** **(19BPS1007)**, ***Sparsh Raj*** **(19BPS1028),** and ***Hande Atharrva Devendra*** **(19BPS1086)** for the course **Business Intelligence**, is a record of bonafide work done under my guidance. The contents of this project work, in full or in parts, have neither been taken from any other source nor have been submitted to any other Institute or University.

Dr. Joshan Athanesious J

ACKNOWLEDGEMENT

We wish to express our sincere thanks and deep sense of gratitude to our project guide, **Dr. Joshan Athanesious J**, Senior Professor, School of Computer Science and Engineering, for his consistent encouragement and valuable guidance offered to us in a pleasant manner throughout the course of the project work.

We are extremely grateful to **Dr. Ganeshan R**, Dean of School of Computer Science and Engineering, VIT Chennai, for extending the facilities of the School towards our project and for his unstinting support.

We also take this opportunity to thank all the faculty of the School for their support and their wisdom imparted to us throughout the course.

We thank our parents, family, and friends for bearing with us throughout the course of our project and for the opportunity they provided us in undergoing this course in such a prestigious institution.

NAME WITH SIGNATURE **NAME WITH SIGNATURE**

Table of Contents

1. Abstract

This project dives into the banking system taking into consideration the problem of revenue decline in banks due to no long term investments by customers.
The main idea behind taking up the study is to understand their existing customers – their transaction patterns, product holdings, demographics, past trend, and other attributes and behaviour with the bank to devise an effective strategy.
It is to determine that if the bank can devise any strategy to tap potential around its existing customer base, and if it can scale up business quickly in a more cost-effective manner as there is no acquisition cost. These customers are already banking with them they need attention, service, and hand-holding.
The solution proposed is through binary classification of the available data which predicts whether the customer is a potential long term investor or not.
Retail banks deal with various problems during business expansion. Over the years, banks have been trying to expand their customer base without taking into full consideration the value each customer brings now and is able to bring in the future. Customers leave behind a large footprint in terms of the transactions they perform, which can be analysed to determine who the most valuable customers are and how to nurture and grow the business by leveraging the existing customer base.
The main idea behind taking up the study is to understand their existing customers – their transaction patterns, product holdings, demographics, past trend, and other attributes and behaviour with the bank to devise an effective strategy.
It is determined that if the bank can devise any strategy to tap potential around its existing customer base, then it can scale up business quickly in a more cost-effective manner as there is no acquisition cost. These customers are already banking with them; they need attention, service, and hand-holding.
So the bank wants to target its services to the selected groups of customer segments, created by differentiating between valuable and non-value-add customers. Currently, the bank works on gut feelings, without a strategy based on analytics, regarding which customer to target (whom to offer an additional service) and who is a potential risk.

2. Introduction

Customers are the most important factor for a business. Some customers can help the business to generate more profit compared to the others. A loyalty-prone customer intends to stay with the supplier who can provide the quality products. On the other hand, a deal-prone customer will always look for a better offer from a competitor. Customers can be classified as profitable and unprofitable. In this project, The Problem statement is that the customers of a bank are not investing enough for long term deposits.

We aim to perform Exploratory Data Analysis and Binary Classification on the Banking Dataset. The Scenario is that the customers of a bank are not investing enough for long term deposits. So, we will do EDA on the dataset and will identify the existing customers that have a higher chance to subscribe for long term deposits. So that the Banks can focus on them in particular.

Customers are the most important factor for a business. Some customers can help the business to generate more profit compared to the others. A loyalty-prone customer intends to stay with the supplier who can provide the quality products. On the other hand, a deal-prone customer will always look for a better offer from a competitor. Customers can be classified as profitable and unprofitable. Data accumulation has been highly increased in recent years which leads to a great interest in the field of machine learning (ML). The computer has proven its usefulness in predicting and pattern findings by learning the data. The capabilities of machine learning (ML) were extensively explored and significant successes were achieved in the field of language translation, self-driving, text recognition, image recognition and voice recognition.

3. Literature Review

A MACHINE LEARNING APPROACH TO IDENTIFY POTENTIAL CUSTOMER BASED ON PURCHASE BEHAVIOR:

(Doi: 10.1109/ICREST.2019.8644458.**)**

In order to break the revenue cap and stay ahead of the competition, it is important to understand customer purchasing behaviour. Different business industries proposed different policies in order to explore the potentiality of a customer based on statistical analysis. In this paper, the writers suggest a machine learning approach for identifying potential customers for a retail superstore. The paper suggests using customer behaviour data to help classify potential customers. We use this classification as a baseline, and then use machine learning algorithms to find a pattern that can be used to predict who is likely to be a customer. The accuracy is 98.7%. The research is the first to use machine learning to study customer behaviour in a retail superstore. The experiment of potential customer classification achieved a very high prediction accuracy of 99.4%. The research is based on data from Tara din, a grocery store, only. The findings of the study may not be applicable to other businesses, as the study was focused on identifying potential customers of Tara din Superstore. The findings of this research may not be applicable to all industries. However, this may be generalized enough to adhere to the problem definition for the whole grocery superstore system in Bangladesh. The data that is currently being used by Tara dins for their existing customers will be used in this research. Therefore, if there is an untapped customer segment not in this dataset, it will not be included in this study. In the future, machine learning will be able to help us understand customer behaviour, which will help us plan more appropriate marketing plans and efficient supply chain management.

A COMPARISON OF MACHINE LEARNING TECHNIQUES FOR CUSTOMER CHURN PREDICTION

(https://doi.org/10.1016/j.simpat.2015.03.003.)

In the telecommunications industry, to solve the problem of customer churn prediction, this paper uses various machine learning models. The data set used is a public domain dataset. First all the models are implemented and the cross-validation is used to evaluate the model. The models used are Support Vector Machines, Decision Tree Learning,

Artificial Neural Network, Naïve Bayes, Simple Regression and Logistic Regression. Then with the help of boosting, the performance was improved. The best classifier was found to be SVM with AdaBoost with accuracy of 97%.

A MACHINE LEARNING FRAMEWORK FOR PREDICTING PURCHASE BY ONLINE CUSTOMERS BASED ON DYNAMIC PRICING

(https://doi.org/10.1016/j.procs.2014.09.060)

In order to improve precise pricing purchases made by customers on an e-commerce system, the study applies sound machine learning algorithms to provide a basic outline and practical methodologies. The main aim is to predict the price of product based on the dynamic pricing of the product. Here, the consumer group is prioritized over the individual purchasers in order to better anticipate purchases. For this purpose, logistic regression analysis model is used to predict the purchase based on dynamic pricing.

POTENTIAL CUSTOMER DETECTION-BASED ON PURCHASE BEHAVIOUR:

(ISSN: 2393-9028 (PRINT) | ISSN: 2348-2281 (ONLINE))

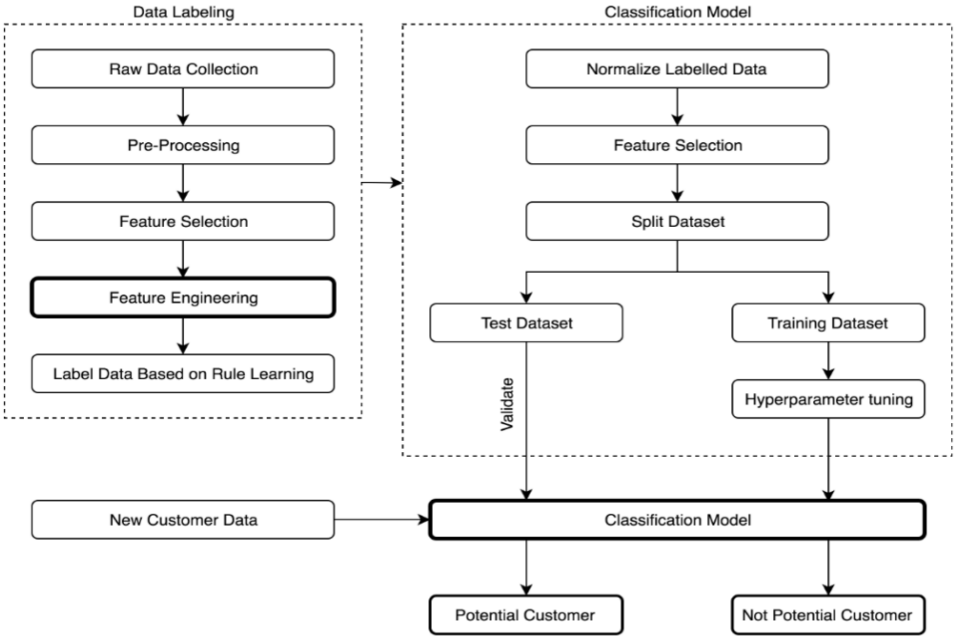
Future clients could provide benefits in the future. The Supervisor can make decisions and control relationships with clients explicitly when searching for these individuals. In this project, a novel machine-learning algorithm is used to find potential customers who visit websites. SVM can identify the best hyperplane by using a few vectors close to the limit. That said, if the assignment tests for the two classes are skewed, the PSVM generally fits the class with more examples better and makes more mistakes in the class with fewer examples. This project has developed a more accurate SVM algorithm, DFP-PSVM, that can be used to identify potential security risks. The changed calculation has a good ability to order examples of the two-class issues, even though the examples are uneven. We can identify which customers the product could attract new customers by advertising to them. Customers have been browsing the use of the product for example, we have given them samples and quotations etc. In addition, it can save product information that may be affected by the seller for their purchase history, which will help to reduce production losses and will serve as a good source of inspiration for all buyer's satisfaction, and they can also be classified using clustering methods as potential or not

4. Proposed Approach

RANDOM FOREST

Random forest is a Supervised Machine Learning Algorithm that is used widely in Classification and Regression problems. It builds decision trees on different samples and takes their majority vote for classification and average in case of regression.

One of the most important features of the Random Forest Algorithm is that it can handle the data set containing continuous variables as in the case of regression and categorical variables as in the case of classification. It performs better results for classification problems.



5. Implementation

About Dataset

There has been a revenue decline in a Bank and they would like to know what actions to take. After investigation, they found that the root cause was that their customers are not investing enough for long term deposits. So, the bank would like to identify existing customers that have higher chance to subscribe for a long-term deposit and focus marketing efforts on such customers.

Data Set Information

The data is related to direct marketing campaigns of a banking institution. The marketing campaigns were based on phone calls. Often, more than one contact to the same client was required, in order to access if the product (bank term deposit) would be subscribed ('yes') or not ('no') subscribed.
There are two datasets: train.csv with all examples (32950) and 21 inputs including the target feature, ordered by date, very close to the data analysed.
test.csv which is the test data that consists of 8238 observations and 20 features without the target feature

Goal: - The classification goal is to predict if the client will subscribe (yes/no) a term deposit (variable y).

The dataset contains train and test data. Features of train data are listed below and the test data have already been pre-processed.

Features

Feature	Feature Type	Description
age	numeric	age of a person
job	Categorical,nominal	type of job ('admin.','blue-collar','entrepreneur','housemaid','management','retired','self-employed','services','student','technician','unemployed','unknown')
marital	categorical,nominal	marital status ('divorced','married','single','unknown'; note: 'divorced' means divorced or widowed)
education	categorical,nominal	('basic.4y','basic.6y','basic.9y','high.school','illiterate','professional.course','university.degree','unknown')
default	categorical,nominal	has credit in default? ('no','yes','unknown')

housing	categorical,nominal	has housing loan? ('no','yes','unknown')
loan	categorical,nominal	has personal loan? ('no','yes','unknown')
contact	categorical,nominal	contact communication type ('cellular','telephone')
month	categorical,ordinal	last contact month of year ('jan', 'feb', 'mar', …, 'nov', 'dec')
day*of*week	categorical,ordinal	last contact day of the week ('mon','tue','wed','thu','fri')
duration	numeric	last contact duration, in seconds . Important note: this attribute highly affects the output target (e.g., if duration=0 then y='no')
campaign	numeric	number of contacts performed during this campaign and for this client (includes last contact)
pdays	numeric	number of days that passed by after the client was last contacted from a previous campaign (999 means client was not previously contacted)
previous	numeric	number of contacts performed before this campaign and for this client
poutcome	categorical,nominal	outcome of the previous marketing campaign ('failure','nonexistent','success')

Target variable (desired output):

Feature	Feature_Type	Description
y	binary	has the client subscribed a term deposit? ('yes','no')

6. Project Description

Our Project is based on Exploratory Data Analysis and Binary Classification on Banking Dataset. So, it consists of several modules as follows:-

- Importing banking dataset
- Exploratory Data analysis
- Pre-processing of the dataset
- Graphical Visualisation of the dataset
- Building multiple machine learning/deep learning models
- Implementing the machine/deep learning models
- Evaluating the built models
- Prediction of potential banking customers

7. Data Pre-Processing

- Data pre-processing is the process of transforming raw data into a useful, understandable format.
- Real-world or raw data usually has inconsistent formatting, human errors, and can also be incomplete.
- Data pre-processing resolves such issues and makes datasets completer and more efficient to perform data analysis.
- One of the main steps in data pre-processing is handling missing data. Missing data means absence of observations in columns that can be caused while procuring the data, lack of information, incomplete results etc.

```
[ ] import pandas as pd
    import numpy as np
    import matplotlib.pyplot as plt
    import seaborn as sns
    import keras
    from tensorflow.keras.models import Sequential
    from tensorflow.keras.layers import Dense
    %matplotlib inline

    import warnings
    warnings.filterwarnings("ignore")
```

```
    # import data modelling libraries
    from sklearn.preprocessing import LabelEncoder
    from sklearn.model_selection import train_test_split
    from imblearn.combine import SMOTETomek
    from sklearn.linear_model import LogisticRegression
    from sklearn import tree
    from sklearn.tree import DecisionTreeClassifier
    from sklearn.ensemble import RandomForestClassifier
    from sklearn.model_selection import GridSearchCV
    from sklearn.model_selection import RandomizedSearchCV
    from sklearn.model_selection import KFold
    from sklearn.model_selection import StratifiedKFold
    from sklearn.metrics import accuracy_score, confusion_matrix, classification_report
```

```
[ ] data= pd.read_csv("new_train.csv")

    # check shape of dataset
    print("shape of the data:", data.shape)
    data.head()
```

shape of the data: (32950, 16)

	age	job	marital	education	default	housing	loan	contact	month	day_of_week	duration	campaign	pdays	previous	poutcome	y
0	49	blue-collar	married	basic.9y	unknown	no	no	cellular	nov	wed	227	4	999	0	nonexistent	no
1	37	entrepreneur	married	university.degree	no	no	no	telephone	nov	wed	202	2	999	1	failure	no
2	78	retired	married	basic.4y	no	no	no	cellular	jul	mon	1148	1	999	0	nonexistent	yes
3	36	admin.	married	university.degree	no	yes	no	telephone	may	mon	120	2	999	0	nonexistent	no
4	59	retired	divorced	university.degree	no	no	no	cellular	jun	tue	368	2	999	0	nonexistent	no

```
[ ]  data.dtypes
```

```
age              int64
job             object
marital         object
education       object
default         object
housing         object
loan            object
contact         object
month           object
day_of_week     object
duration         int64
campaign         int64
pdays            int64
previous         int64
poutcome        object
y               object
dtype: object
```

Check Missing Data

One of the main steps in data pre-processing is handling missing data. Missing data means absence of observations in columns that can be caused while procuring the data, lack of information, incomplete results etc. Feeding missing data to your machine learning model could lead to wrong prediction or classification. Hence it is necessary to identify missing values and treat them.

```
[ ]  data.isnull().sum()
```

```
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
y               0
dtype: int64
```

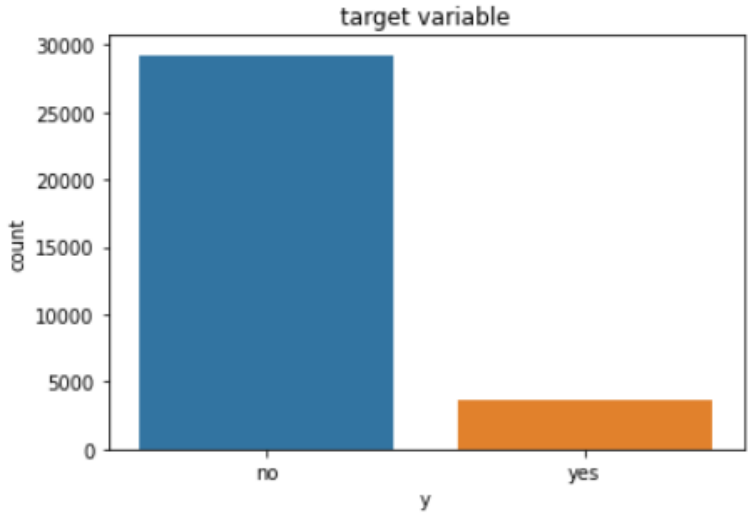
Check for Class Imbalance

```
[ ]  # target class count
     data["y"].value_counts()
```

```
no     29238
yes     3712
Name: y, dtype: int64
```

```
[ ]  sns.countplot(data["y"])
     plt.title("target variable")
```

```
Text(0.5, 1.0, 'target variable')
```




```
[ ]  # percentage of class present in target variable(y)
     print("percentage of NO and YES\n",data["y"].value_counts()/len(data)*100)
```

```
percentage of NO and YES
 no     88.734446
yes     11.265554
Name: y, dtype: float64
```

The class distribution in the target variable is ~89:11 indicating an imbalance dataset

8. Exploratory Data Analysis

Exploratory Data Analysis refers to the critical process of performing initial investigations on data so as to discover patterns, to spot anomalies, to test hypothesis and to check assumptions with the help of summary statistics and graphical representations.

- *Categorical data* is a type of data that is used to group information with similar characteristics.
- *Numerical data* is a type of data that expresses information in the form of numbers.
- *Univariate analysis* is the simplest form of analysing data. “Uni” means “one”, so in other words your data has only one variable. It doesn’t deal with causes or relationships (unlike regression) and its major purpose is to describe; It takes data, summarizes that data and finds patterns in the data.
- *Bivariate analysis* means the analysis of bivariate data. It is one of the simplest forms of statistical analysis, used to find out if there is a relationship between two sets of values. It usually involves the variables X and Y.

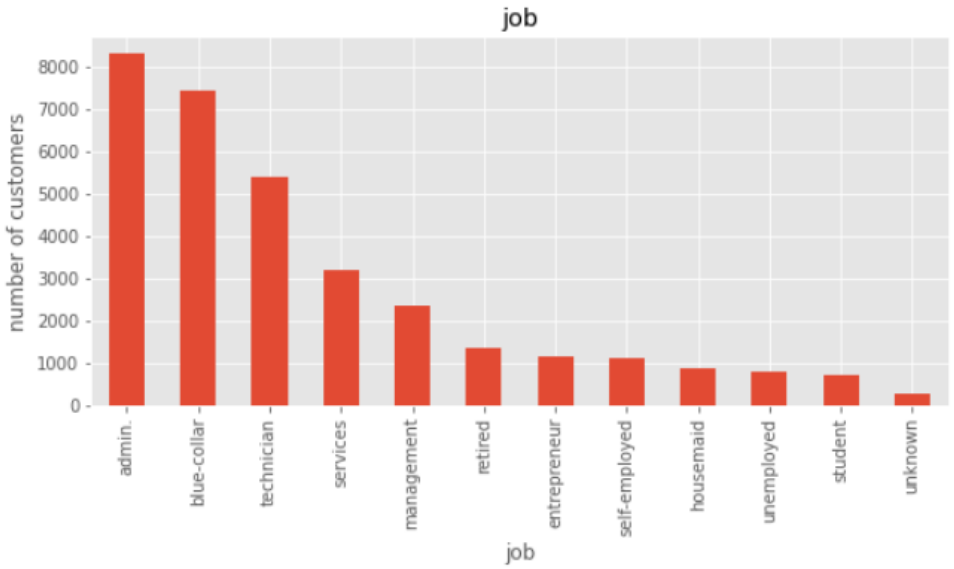
In Our Project, mostly we are going to do: -

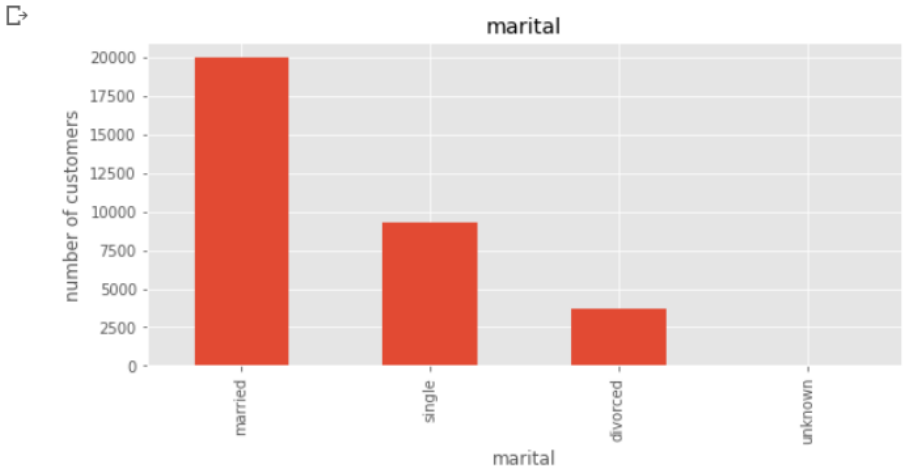
a. Univariate Analysis of Categorical Variables
b. Univariate Analysis of Numerical Columns
c. Bivariate Analysis of Categorical Columns

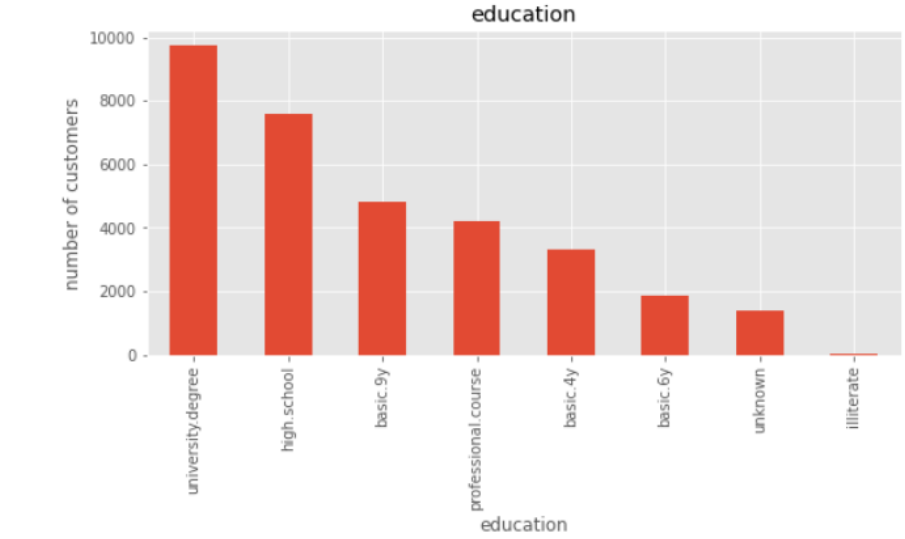
Univariate Analysis of Categorical Variables

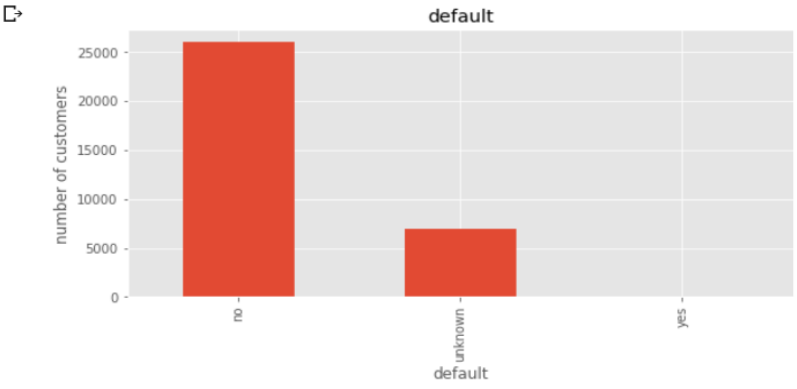
```
[ ]  # indentifying the categorical variables
     cat_var= data.select_dtypes(include= ["object"]).columns
     print(cat_var)

     # plotting bar chart for each categorical variable
     plt.style.use("ggplot")

     for column in cat_var:
         plt.figure(figsize=(20,4))
         plt.subplot(121)
         data[column].value_counts().plot(kind="bar")
         plt.xlabel(column)
         plt.ylabel("number of customers")
         plt.title(column)
```

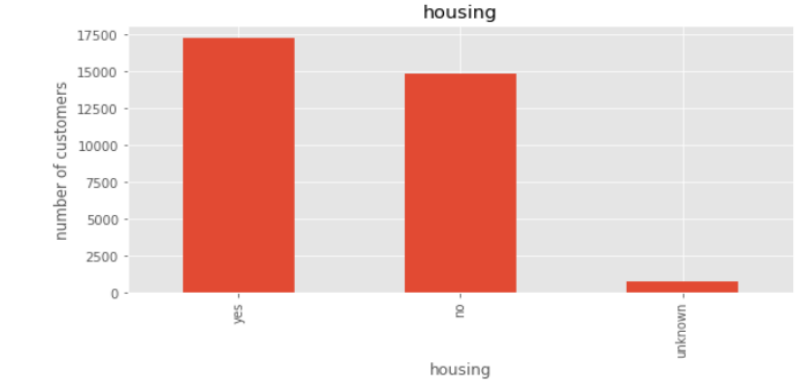
```
Index(['job', 'marital', 'education', 'default', 'housing', 'loan', 'contact',
       'month', 'day_of_week', 'poutcome', 'y'],
      dtype='object')
```

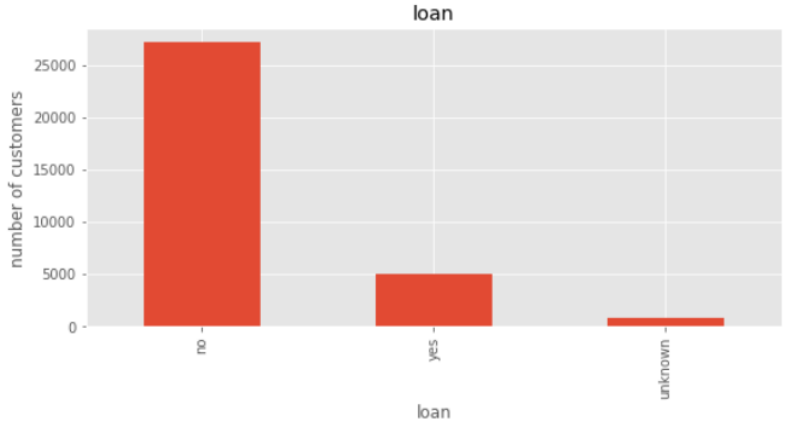


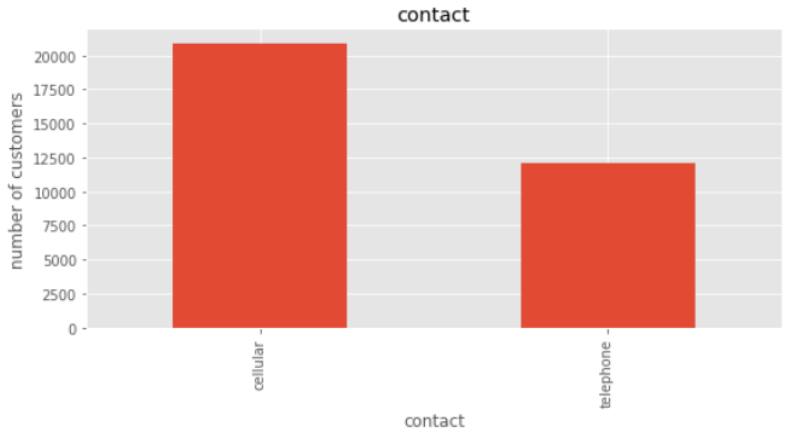

marital
number of customers
20000
17500
15000
12500
10000
7500
5000
2500
0
married
single
divorced
unknown
marital

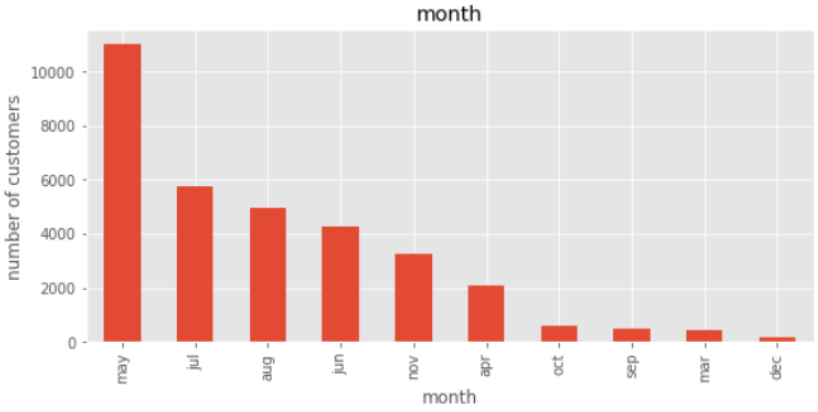

education
number of customers
10000
8000
6000
4000
2000
0
university.degree
high.school
basic.9y
professional.course
basic.4y
basic.6y
unknown
illiterate
education

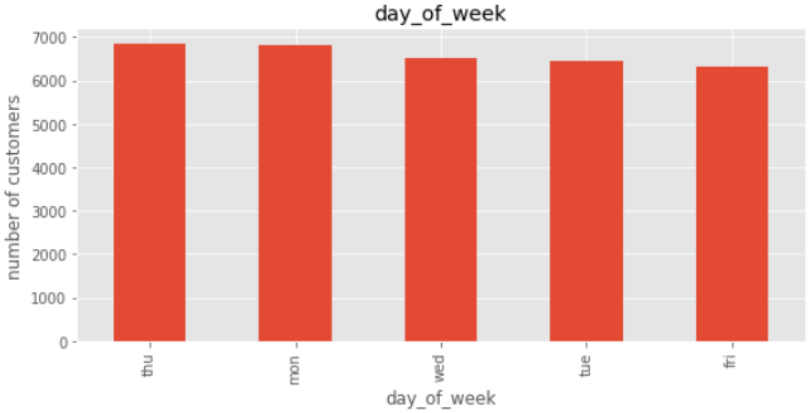

default
number of customers
25000
20000
15000
10000
5000
0
no
unknown
yes
default

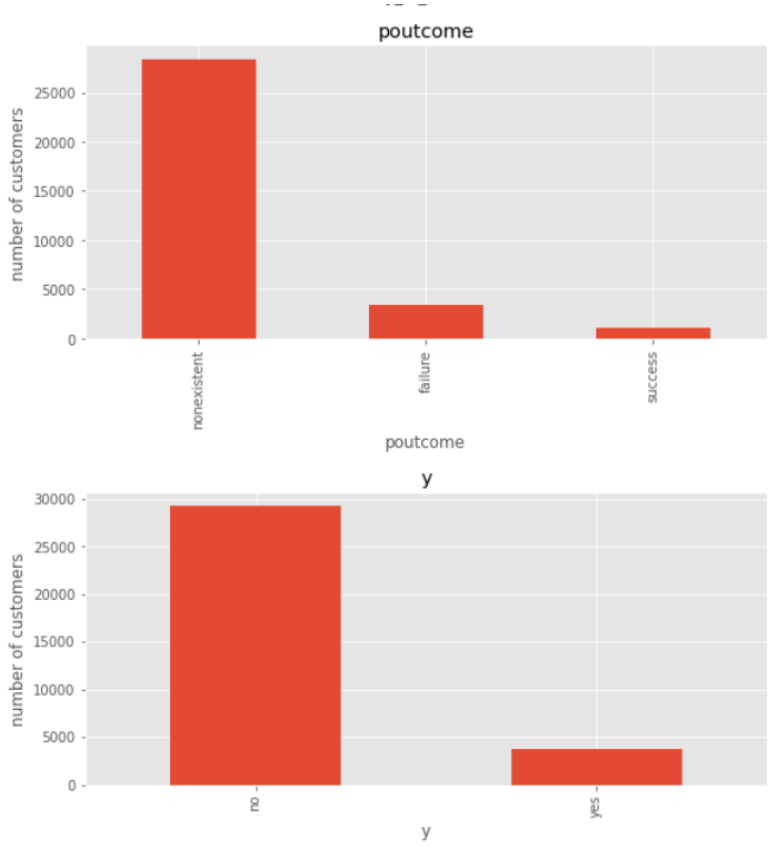

housing
number of customers
17500
15000
12500
10000
7500
5000
2500
0
yes
no
unknown
housing


loan
number of customers
25000
20000
15000
10000
5000
0
no
yes
unknown
loan


contact
number of customers
20000
17500
15000
12500
10000
7500
5000
2500
0
cellular
telephone
contact


month
number of customers
10000
8000
6000
4000
2000
0
may
jul
aug
jun
nov
apr
oct
sep
mar
dec
month


day_of_week
number of customers
7000
6000
5000
4000
3000
2000
1000
0
thu
mon
wed
tue
fri
day_of_week



Observations:

- The top three professions that our customers belong to are - administration, blue-collar jobs and technicians.
- A huge number of the customers are married.
- Majority of the customers do not have a credit in default
- Many of our past customers have applied for a housing loan but very few have applied for personal loans.
- Cell-phones seem to be the most favoured method of reaching out to customers.
- Many customers have been contacted in the month of May.
- The plot for the target variable shows heavy imbalance in the target variable.

The missing values in some columns have been represented as unknown. unknown represents missing data.

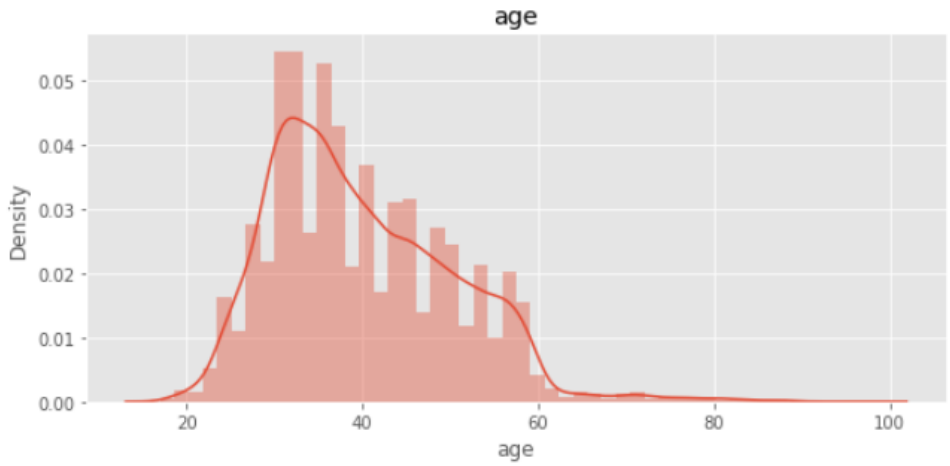
```
[ ]  # replacing "unknown" with the mode
     for column in cat_var:
         mode= data[column].mode()[0]
         data[column]= data[column].replace("unknown", mode)
```

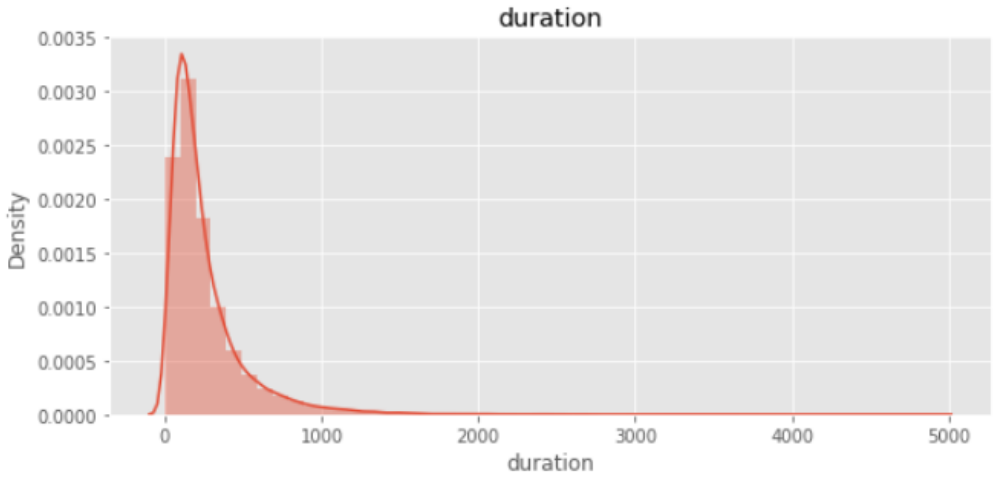
Univariate analysis of Numerical columns

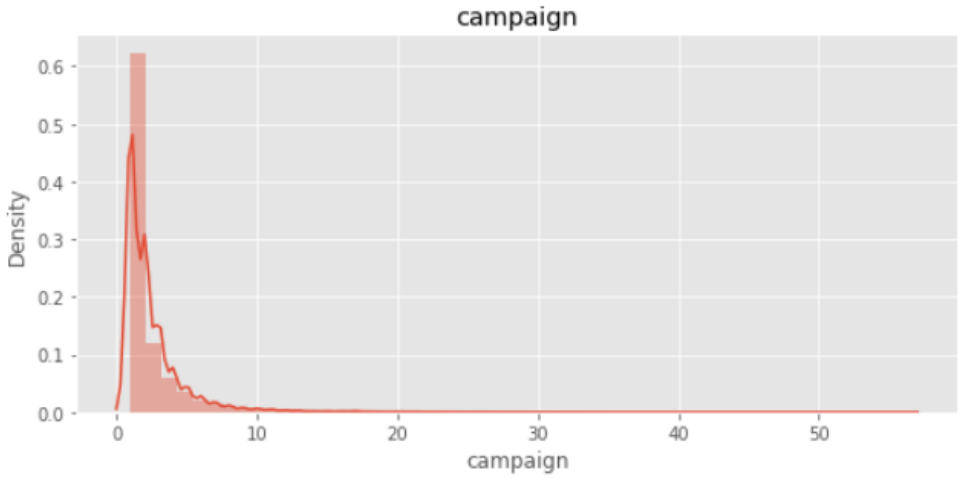
```
[ ]  # indentifying the numerical variables
     num_var= data.select_dtypes(include=np.number)
     num_var.head()
```

	age	duration	campaign	pdays	previous
0	49	227	4	999	0
1	37	202	2	999	1
2	78	1148	1	999	0
3	36	120	2	999	0
4	59	368	2	999	0

```
[ ]  # plotting histogram for each numerical variable
     plt.style.use("ggplot")
     for column in ["age", "duration", "campaign"]:
         plt.figure(figsize=(20,4))
         plt.subplot(121)
         sns.distplot(data[column], kde=True)
         plt.title(column)
```







Observation:

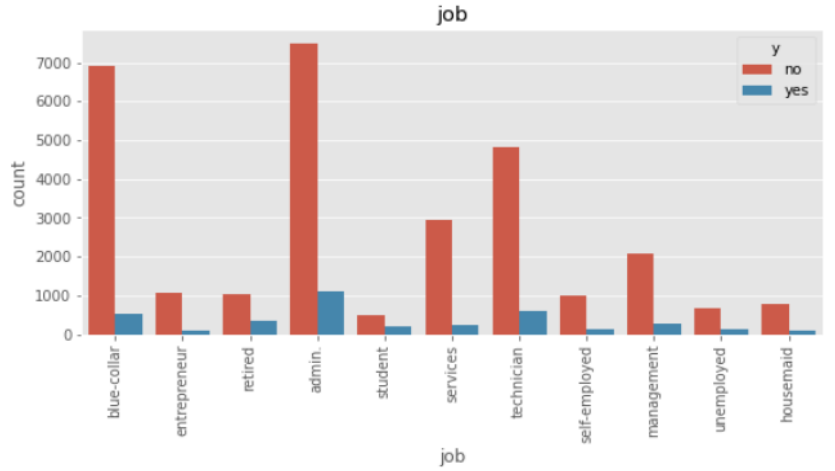
- As we can see from the histogram, the features age, duration and campaign are heavily skewed and this is due to the presence of outliers as seen in the boxplot for these features.
- Looking at the plot for pdays, we can infer that majority of the customers were being contacted for the first time because as per the feature description for pdays the value 999 indicates that the customer had not been contacted previously.

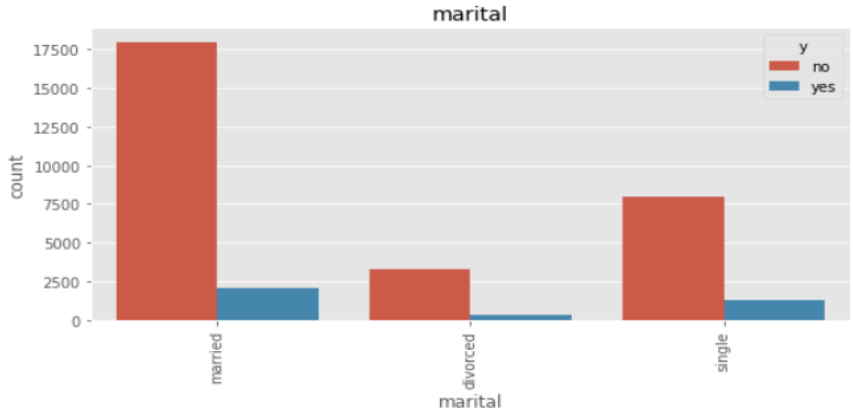
Since pdays and previous consist majorly only of a single value, their variance is quite less and hence we can drop them since technically will be of no help in prediction.

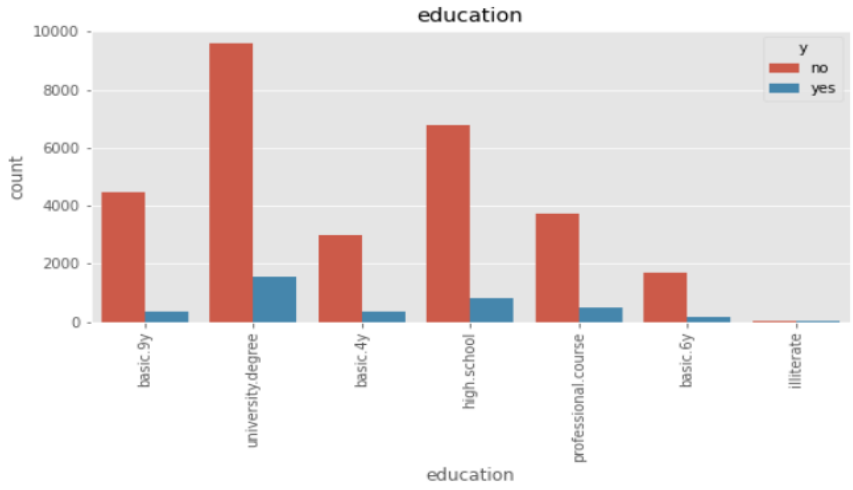
```
[ ]  data.drop(columns=["pdays", "previous"], axis=1, inplace=True)
```

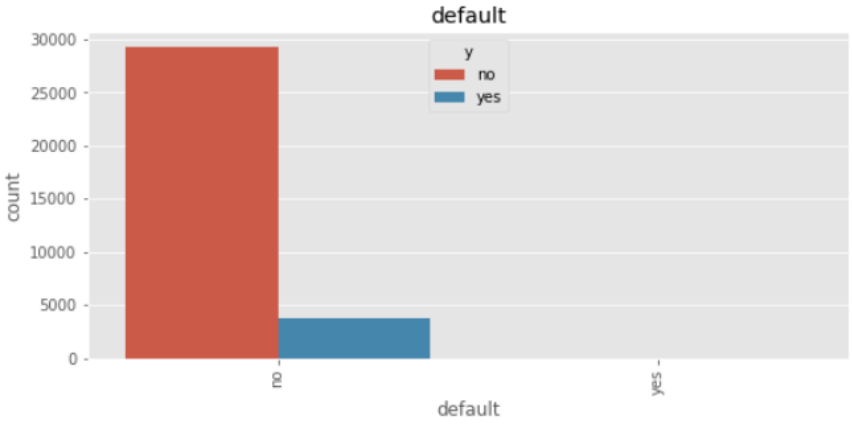
Bivariate Analysis of Categorical Columns

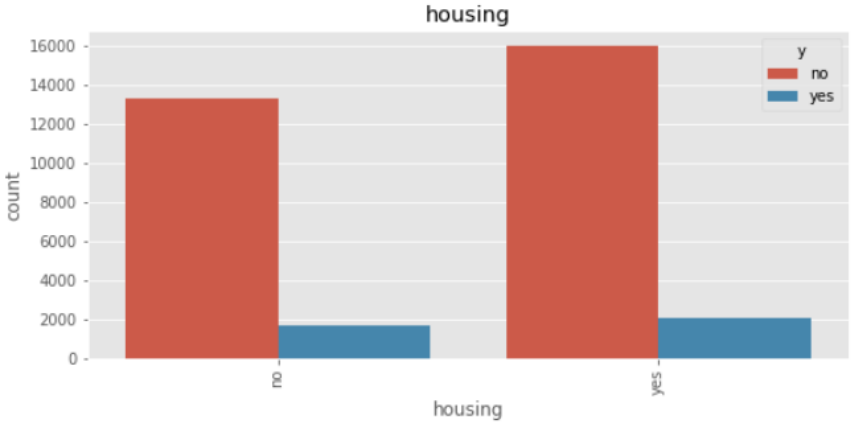
```
[ ]  plt.style.use("ggplot")
     for column in cat_var:
         plt.figure(figsize=(20,4))
         plt.subplot(121)
         sns.countplot(data[column], hue=data["y"])
         plt.title(column)
         plt.xticks(rotation=90)
```

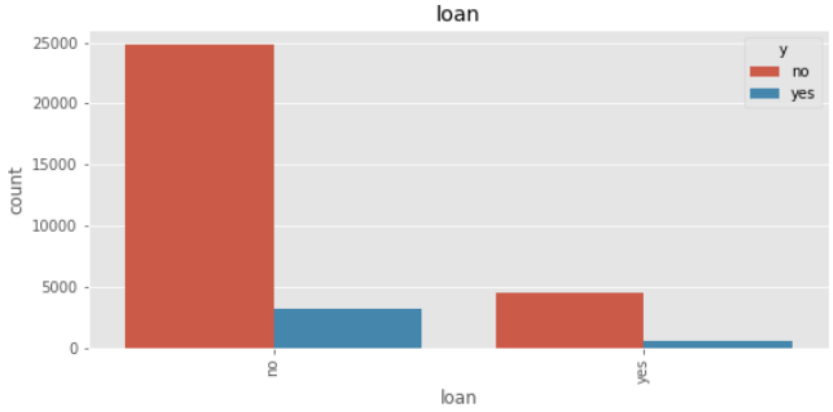


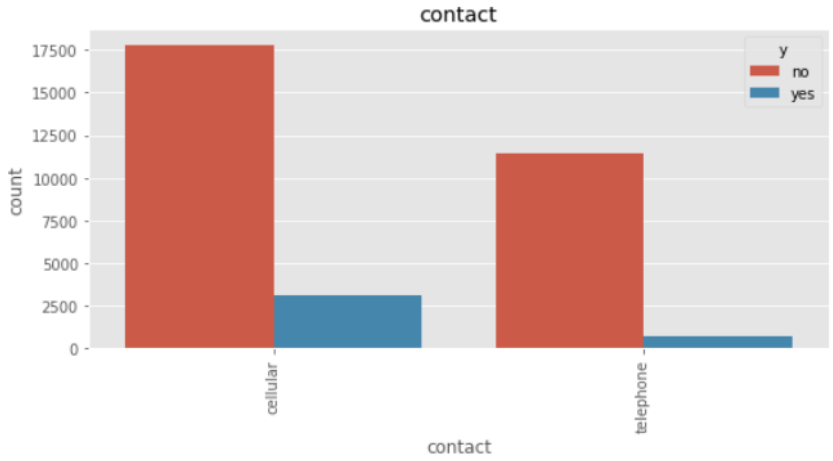


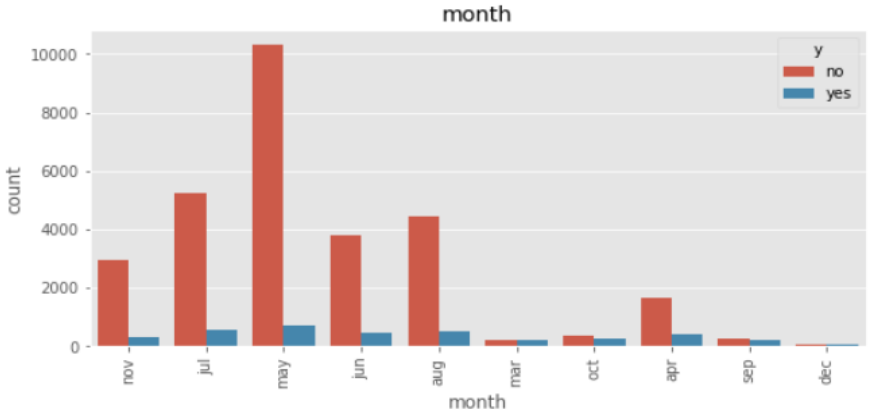


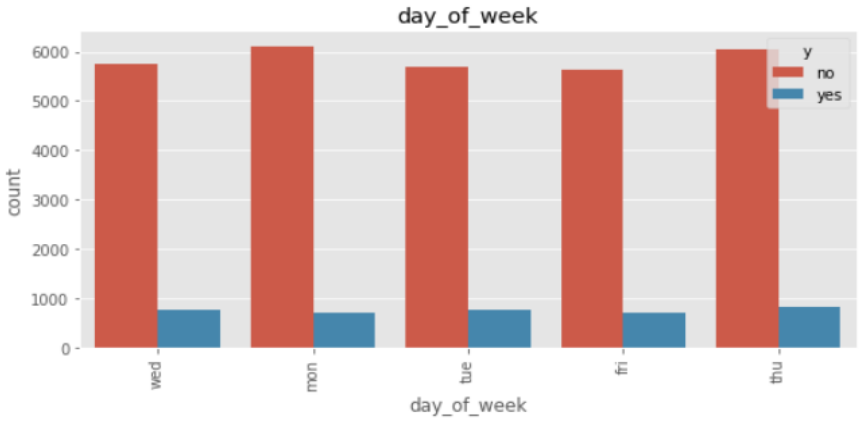

default
y
no
yes
count
30000
25000
20000
15000
10000
5000
0
no
yes
default

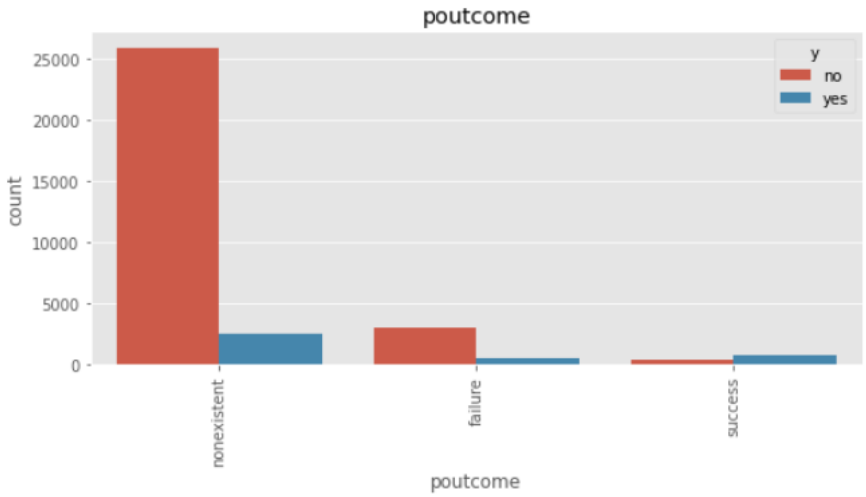

housing
y
no
yes
count
16000
14000
12000
10000
8000
6000
4000
2000
0
no
yes
housing

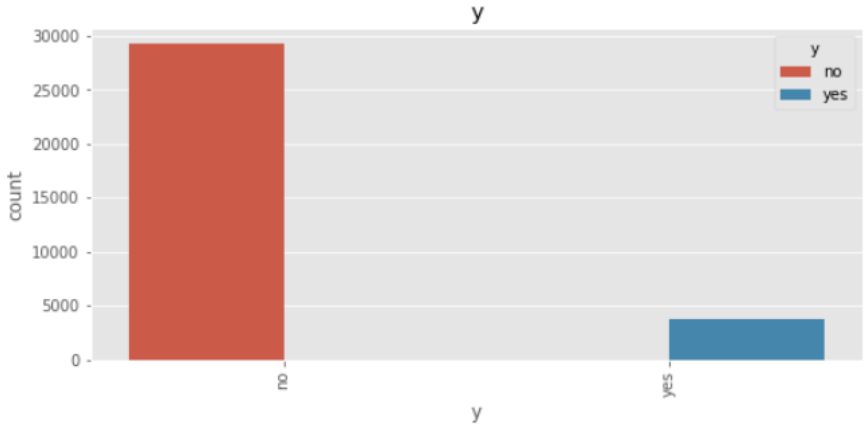

loan
y
no
yes
count
25000
20000
15000
10000
5000
0
no
yes
loan


contact
y
no
yes
count
17500
15000
12500
10000
7500
5000
2500
0
cellular
telephone
contact


month
y
no
yes
count
10000
8000
6000
4000
2000
0
nov
jul
may
jun
aug
mar
oct
apr
sep
dec
month


day_of_week
y
no
yes
count
6000
5000
4000
3000
2000
1000
0
wed
mon
tue
fri
thu
day_of_week


poutcome
y
no
yes
count
25000
20000
15000
10000
5000
0
nonexistent
failure
success
poutcome


y
y
no
yes
count
30000
25000
20000
15000
10000
5000
0
no
yes
y

Observations:

- Customers having administrative jobs form the majority amongst those who have subscribed to the term deposit.
- They are married
- They hold a university degree
- They do not hold a credit in default
- Housing loan doesn't seem a priority to check for since an equal number of customers who have and have not subscribed to it seem to have subscribed to the term deposit.
- Cell-phones should be the preferred mode of contact for contacting customers.

Handling Outliers

- **Outliers are stragglers: -**

Extremely high or extremely low values — in a data set that can throw off your stats.

Outliers cause significant impact on the Mean and Variance. It becomes necessary to treat the outliers.

```
[ ]  data.describe()
```

	age	duration	campaign
count	32950.000000	32950.000000	32950.000000
mean	40.014112	258.127466	2.560607
std	10.403636	258.975917	2.752326
min	17.000000	0.000000	1.000000
25%	32.000000	103.000000	1.000000
50%	38.000000	180.000000	2.000000
75%	47.000000	319.000000	3.000000
max	98.000000	4918.000000	56.000000

Age duration and campaign are skewed towards right, we will compute the IQR and replace the outliers with the lower and upper boundaries

```
[ ]  # compute interquantile range to calculate the boundaries
     lower_boundries= []
     upper_boundries= []
     for i in ["age", "duration", "campaign"]:
         IQR= data[i].quantile(0.75) - data[i].quantile(0.25)
         lower_bound= data[i].quantile(0.25) - (1.5*IQR)
         upper_bound= data[i].quantile(0.75) + (1.5*IQR)

         print(i, ":", lower_bound, ",",  upper_bound)

         lower_boundries.append(lower_bound)
         upper_boundries.append(upper_bound)
```

```
age : 9.5 , 69.5
duration : -221.0 , 643.0
campaign : -2.0 , 6.0
```

```
[ ]  lower_boundries
```

```
[9.5, -221.0, -2.0]
```

```
[ ]  upper_boundries
```

```
[69.5, 643.0, 6.0]
```

```
[ ]  # replace the all the outliers which is greater then upper boundary by upper boundary
     j = 0
     for i in ["age", "duration", "campaign"]:
         data.loc[data[i] > upper_boundries[j], i] = int(upper_boundries[j])
         j = j + 1
```

Since,

- for age the lower boundary (9.5) < minimum value (17)
- for duration and campaign, the lower boundaries are negative (-221.0), (-2.0) resp.

replacing outliers with the lower boundary is not required

```
[ ]  # without outliers
     data.describe()
```

	age	duration	campaign
count	32950.000000	32950.000000	32950.000000
mean	39.929894	234.923915	2.271077
std	10.118566	176.854558	1.546302
min	17.000000	0.000000	1.000000
25%	32.000000	103.000000	1.000000
50%	38.000000	180.000000	2.000000
75%	47.000000	319.000000	3.000000
max	69.000000	643.000000	6.000000

After replacing the outliers with the upper boundary, the maximum values have been changed without impacting any other parameters like mean, standard deviation and quartiles.

9. Creating Machine Learning Model

Encoding Categorical Features

Machine learning algorithm can only read numerical values. It is therefore essential to encode categorical features into numerical values

```
[ ] #categorical features
    cat_var
```

```
Index(['job', 'marital', 'education', 'default', 'housing', 'loan', 'contact',
       'month', 'day_of_week', 'poutcome', 'y'],
      dtype='object')
```

```
[ ] # check categorical class
    for i in cat_var:
        print(i, ":", data[i].unique())
```

```
job : ['blue-collar' 'entrepreneur' 'retired' 'admin.' 'student' 'services'
 'technician' 'self-employed' 'management' 'unemployed' 'housemaid']
marital : ['married' 'divorced' 'single']
education : ['basic.9y' 'university.degree' 'basic.4y' 'high.school'
 'professional.course' 'basic.6y' 'illiterate']
default : ['no' 'yes']
housing : ['no' 'yes']
loan : ['no' 'yes']
contact : ['cellular' 'telephone']
month : ['nov' 'jul' 'may' 'jun' 'aug' 'mar' 'oct' 'apr' 'sep' 'dec']
day_of_week : ['wed' 'mon' 'tue' 'fri' 'thu']
poutcome : ['nonexistent' 'failure' 'success']
y : ['no' 'yes']
```

Features like job education month day_of_week has so many categories, we will Label Encode them as One Hot Encoding would create so many columns

```
[ ] # initializing label encoder
    le= LabelEncoder()

    # iterating through each categorical feature and label encoding them
    for feature in cat_var:
        data[feature]= le.fit_transform(data[feature])
```

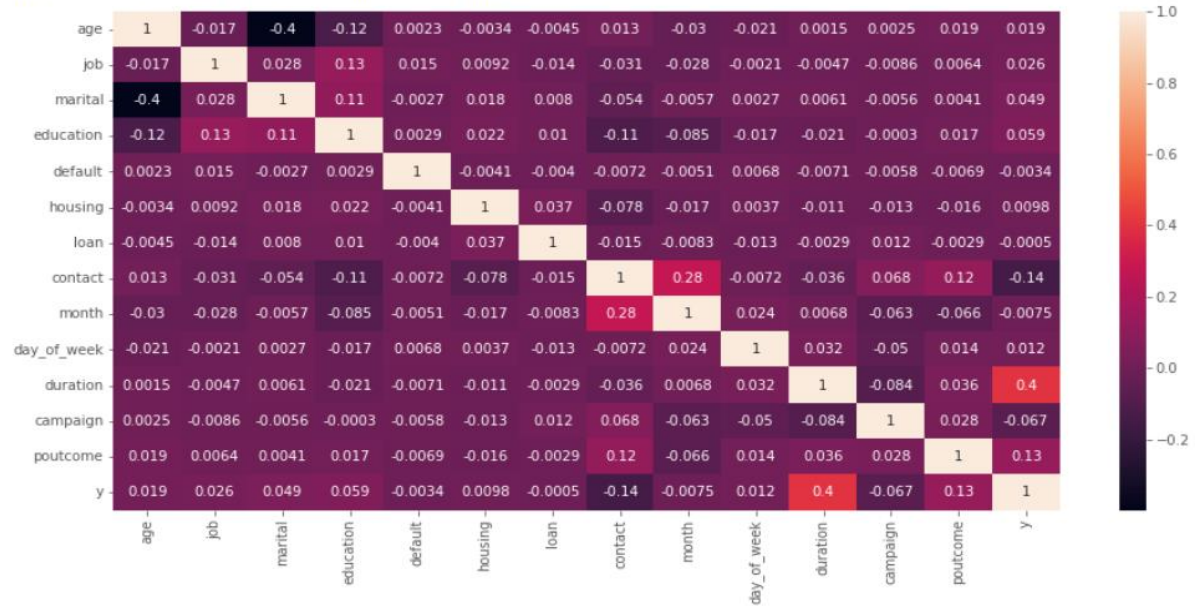
```
[ ] # label encoded dataset
    data.head()
```

	age	job	marital	education	default	housing	loan	contact	month	day_of_week	duration	campaign	poutcome	y
0	49	1	1	2	0	0	0	0	7	4	227	4	1	0
1	37	2	1	6	0	0	0	1	7	4	202	2	0	0
2	69	5	1	0	0	0	0	0	3	1	643	1	1	1
3	36	0	1	6	0	1	0	1	6	1	120	2	1	0
4	59	5	0	6	0	0	0	0	4	3	368	2	1	0

```
[ ]  # feature variables
     x= data.iloc[:, :-1]

     # target variable
     y= data.iloc[:, -1]
```

```
[ ]  plt.figure(figsize=(15,7))
     sns.heatmap(data.corr(), annot=True)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f4a37526c10>
```



There are no features that are highly correlated and inversely correlated. If we had, we could have written the condition that if the correlation is higher than 0.8 (or can be any threshold value depending on the domain knowledge) and less than -0.8, we could have dropped those features. Because those correlated features would have been doing the same job.

Handling imbalanced dataset

Since the class distribution in the target variable is ~89:11 indicating an imbalance dataset, we need to resample it.

```
[ ]  #initialising oversampling
     smote= SMOTETomek(0.75)

     #implementing oversampling to training data
     x_sm, y_sm= smote.fit_resample(x,y)

     # x_sm and y_sm are the resampled data

     # target class count of resampled dataset
     y_sm.value_counts()
```

```
0    28941
1    21631
Name: y, dtype: int64
```

Splitting resampled data in train and test data

```
[ ]  x_train, x_test, y_train, y_test= train_test_split(x_sm, y_sm, test_size=0.2, random_state=42)
```

Grid search and hyperparameter tuning

- Hyperparameter tuning is one of the most important parts of a machine learning pipeline.
- There are several ways to perform hyperparameter tuning. Two of them are grid search and random search
- Grid search is the simplest algorithm for hyperparameter tuning. Basically, we divide the domain of the hyperparameters into a discrete grid. Then, we try every combination of values of this grid, calculating some performance metrics using cross-validation. The point of the grid that maximizes the average value in cross-validation, is the optimal combination of values for the hyperparameters.
- Random search is similar to grid search, but instead of using all the points in the grid, it tests only a randomly selected subset of these points. The smaller this subset, the faster but less accurate the optimization. The

larger this dataset, the more accurate the optimization but the closer to a grid search.

Logistic Regression

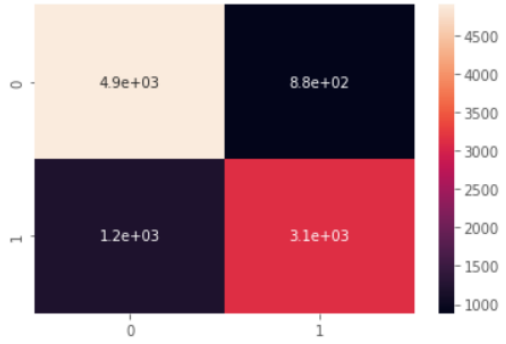
Logistic regression is the appropriate regression analysis to conduct when the dependent variable is dichotomous (binary). Like all regression analyses, the logistic regression is a predictive analysis. Logistic regression is used to describe data and to explain the relationship between one dependent binary variable and one or more nominal, ordinal, interval or ratio-level independent variables.

```
[ ] # selecting the classifier
    log_reg= LogisticRegression()

    # selecting hyperparameter tuning
    log_param= {"C": 10.0**np.arange(-2,3), "penalty": ["l1", "l2"]}

    # defining stratified Kfold cross validation
    cv_log= StratifiedKFold(n_splits=5)

    # using gridsearch for respective parameters
    gridsearch_log= GridSearchCV(log_reg, log_param, cv=cv_log, scoring= "f1_macro", n_jobs=-1, verbose=2)

    # fitting the model on resampled data
    gridsearch_log.fit(x_train, y_train)

    # printing best score and best parameters
    print("best score is:" ,gridsearch_log.best_score_)
    print("best parameters are:" ,gridsearch_log.best_params_)
```

```
Fitting 5 folds for each of 10 candidates, totalling 50 fits
best score is: 0.7940588731717027
best parameters are: {'C': 100.0, 'penalty': 'l2'}
```

```
# checking model performance
y_predicted= gridsearch_log.predict(x_test)

cm= confusion_matrix(y_test, y_predicted)
print(cm)
sns.heatmap(cm, annot=True)
print(accuracy_score(y_test, y_predicted))
print(classification_report(y_test, y_predicted))
```

```
[[4897  884]
 [1201 3131]]
0.7938297241174725
              precision    recall  f1-score   support

           0       0.80      0.85      0.82      5781
           1       0.78      0.72      0.75      4332

    accuracy                           0.79     10113
   macro avg       0.79      0.78      0.79     10113
weighted avg       0.79      0.79      0.79     10113
```



Random Forest

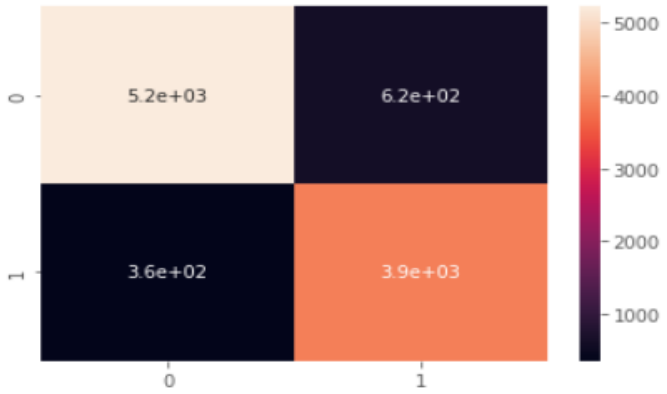
A random forest algorithm consists of many decision trees. The (random forest) algorithm establishes the outcome based on the predictions of the decision trees. It predicts by taking the average or mean of the output from various trees. Increasing the number of trees increases the precision of the outcome.

```
[ ] # random forest
    rf= RandomForestClassifier()

    rf_param= {
              "n_estimators": [int(x) for x in np.linspace(start=100, stop=1000, num=10)],
              "max_features": ["auto", "sqrt", "log2"],
    #           "max_depth": [4,5,6,7,8],
              "max_depth": [int(x) for x in np.linspace(start=5, stop=30, num=6)],
              "min_samples_split": [5,10,15,100],
              "min_samples_leaf": [1,2,5,10],
              "criterion":['gini', 'entropy']
             }

    cv_rf= StratifiedKFold(n_splits=5)

    randomsearch_rf= RandomizedSearchCV(rf, rf_param, cv=cv_rf, scoring= "f1_macro", n_jobs=-1, verbose=2, n_iter=10)

    randomsearch_rf.fit(x_train, y_train)

    print("best score is:", randomsearch_rf.best_score_)
    print("best parameters are:", randomsearch_rf.best_params_)
```

```
Fitting 5 folds for each of 10 candidates, totalling 50 fits
best score is: 0.9013138973860535
best parameters are: {'n_estimators': 300, 'min_samples_split': 10, 'min_samples_leaf': 2, 'max_features': 'sqrt', 'max_depth': 30, 'criterion': 'entropy'}
```

```
[ ]  # checking model performance
     y_predicted_rf= randomsearch_rf.predict(x_test)

     print(confusion_matrix(y_test, y_predicted_rf))
     sns.heatmap(confusion_matrix(y_test, y_predicted_rf), annot=True)
     print(accuracy_score(y_test, y_predicted_rf))
     print(classification_report(y_test, y_predicted_rf))
```

```
[[5218  618]
 [ 363 3916]]
0.9030153237765695
              precision    recall  f1-score   support

           0       0.93      0.89      0.91      5836
           1       0.86      0.92      0.89      4279

    accuracy                           0.90     10115
   macro avg       0.90      0.90      0.90     10115
weighted avg       0.90      0.90      0.90     10115
```



Artificial Neural Network:

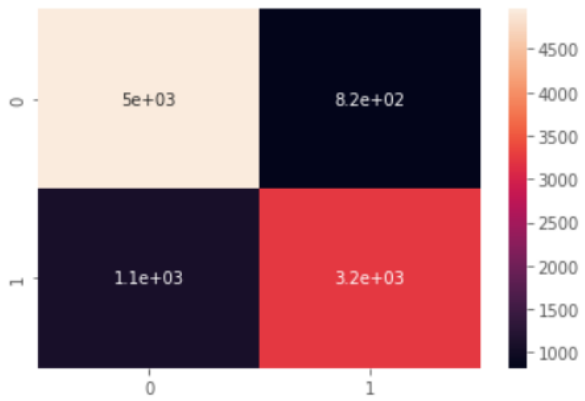
Computing systems inspired by the biological neural networks that make up animal brains are known as artificial neural networks (ANNs), or more simply, neural nets.

```
[ ]  ANN_Classifier=Sequential()
     ANN_Classifier.add(Dense(units = 6, kernel_initializer = 'uniform', activation = 'relu', input_dim = 13))
     ANN_Classifier.add(Dense(units = 6, kernel_initializer = 'uniform', activation = 'relu'))
     ANN_Classifier.add(Dense(units = 1, kernel_initializer = 'uniform', activation = 'sigmoid'))
     ANN_Classifier.compile(optimizer = 'adam', loss = 'binary_crossentropy', metrics = ['accuracy'])
     ANN_Classifier.fit(x_train, y_train, batch_size = 10, epochs = 20)
     ANN_Classifier.predict(x_test)
```

```
Epoch 1/20
4046/4046 [==============================] - 8s 2ms/step - loss: 0.4759 - accuracy: 0.7664
Epoch 2/20
4046/4046 [==============================] - 7s 2ms/step - loss: 0.4224 - accuracy: 0.8020
Epoch 3/20
4046/4046 [==============================] - 7s 2ms/step - loss: 0.4186 - accuracy: 0.8035
Epoch 4/20
4046/4046 [==============================] - 7s 2ms/step - loss: 0.4162 - accuracy: 0.8035
Epoch 5/20
4046/4046 [==============================] - 8s 2ms/step - loss: 0.4141 - accuracy: 0.8043
Epoch 6/20
4046/4046 [==============================] - 15s 4ms/step - loss: 0.4127 - accuracy: 0.8060
Epoch 7/20
4046/4046 [==============================] - 7s 2ms/step - loss: 0.4124 - accuracy: 0.8056
Epoch 8/20
4046/4046 [==============================] - 7s 2ms/step - loss: 0.4114 - accuracy: 0.8068
Epoch 9/20
4046/4046 [==============================] - 7s 2ms/step - loss: 0.4107 - accuracy: 0.8070
Epoch 10/20
4046/4046 [==============================] - 7s 2ms/step - loss: 0.4095 - accuracy: 0.8093
Epoch 11/20
4046/4046 [==============================] - 9s 2ms/step - loss: 0.4068 - accuracy: 0.8118
Epoch 12/20
4046/4046 [==============================] - 7s 2ms/step - loss: 0.4053 - accuracy: 0.8128
Epoch 13/20
4046/4046 [==============================] - 7s 2ms/step - loss: 0.4046 - accuracy: 0.8149
Epoch 14/20
4046/4046 [==============================] - 7s 2ms/step - loss: 0.4018 - accuracy: 0.8140
Epoch 15/20
4046/4046 [==============================] - 6s 2ms/step - loss: 0.4017 - accuracy: 0.8160
Epoch 16/20
4046/4046 [==============================] - 6s 2ms/step - loss: 0.4004 - accuracy: 0.8146
Epoch 17/20
4046/4046 [==============================] - 7s 2ms/step - loss: 0.3976 - accuracy: 0.8184
Epoch 18/20
4046/4046 [==============================] - 7s 2ms/step - loss: 0.3973 - accuracy: 0.8168
Epoch 19/20
4046/4046 [==============================] - 6s 2ms/step - loss: 0.3951 - accuracy: 0.8186
Epoch 20/20
4046/4046 [==============================] - 6s 2ms/step - loss: 0.3955 - accuracy: 0.8183
array([[0.63662255],
       [0.01173812],
       [0.09545735],
       ...,
       [0.2811761 ],
       [0.05038118],
       [0.01400724]], dtype=float32)
```

```
[ ] y_predicted_rf= ANN_Classifier.predict(x_test)
    y_predicted_rf = (y_predicted_rf > 0.5)
    confusion_matrix(y_test, y_predicted_rf)
    sns.heatmap(confusion_matrix(y_test, y_predicted_rf), annot=True)
    print(accuracy_score(y_test, y_predicted_rf))
    print(classification_report(y_test, y_predicted_rf))
```

```
0.8102442400870167
              precision    recall  f1-score   support

           0       0.82      0.86      0.84      5781
           1       0.80      0.75      0.77      4332

    accuracy                           0.81     10113
   macro avg       0.81      0.80      0.80     10113
weighted avg       0.81      0.81      0.81     10113
```



LSTM:

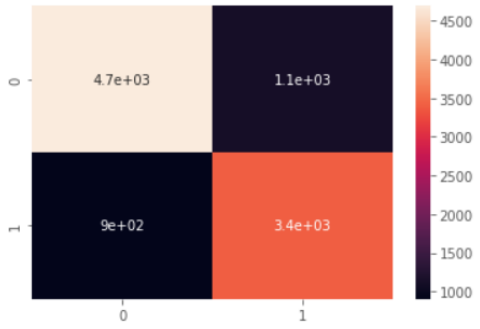
LSTM Model is also one of the candidates for our test as it is a type of neural network capable of learning dependent data that is going to be present in the long-term plan of our project and this is made possible by the model's unique combination of layers that interact with each other.

```
[ ]  #LSTM Model
     model = Sequential()
     model.add(LSTM(4,input_dim=13))
     model.add(Dropout(0.1))
     model.add(Dense(1))
     model.add(Activation('sigmoid'))
```

```
model.compile(loss='binary_crossentropy',optimizer='adam',metrics=['accuracy'])
checkpointer = callbacks.ModelCheckpoint(filepath="checkpoint-{epoch:02d}.hdf5", verbose=1, save_best_only=True, monitor='val_acc',mode='max')
model.fit(x_train, y_train, batch_size=batch_size, epochs=10, validation_data=(x_test, y_test))
```

```
loss, accuracy = model.evaluate(x_test, y_test)
print("Accuracy of LSTM Model: %.2f%%" % ( accuracy*100))
y_pred = model.predict(x_test)
y_pred = (y_pred > 0.5)
confusion_matrix(y_test, y_pred)
sns.heatmap(confusion_matrix(y_test, y_pred), annot=True)
print(classification_report(y_test, y_pred))
```

```
317/317 [==============================] - 1s 3ms/step - loss: 0.4300 - accuracy: 0.7998
Accuracy of LSTM Model: 79.98%
              precision    recall  f1-score   support

           0       0.84      0.81      0.82      5813
           1       0.75      0.79      0.77      4310

    accuracy                           0.80     10123
   macro avg       0.80      0.80      0.80     10123
weighted avg       0.80      0.80      0.80     10123
```



SVM:

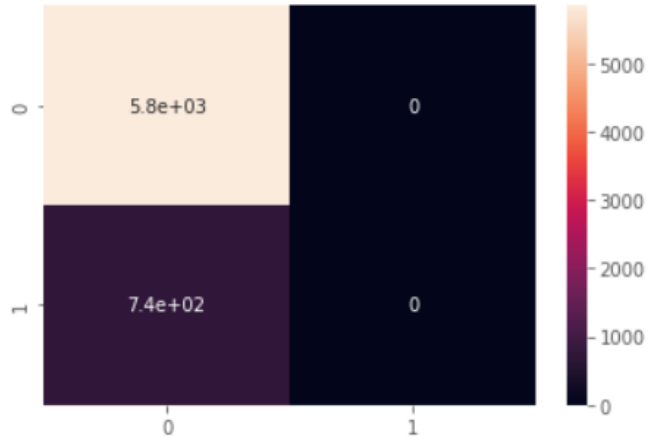
SVM is a model used to find a plane of N dimension space that classifies all the data points on it and it is used for both classifications as well for the purposes of regression. The Regression Plane size is dependent on the dimensionality of the data.

```
#SVM Model
classifier = SVC(kernel='rbf', random_state = 1)
classifier.fit(x_train,y_train)
```

```
y_pred = classifier.predict(x_test)
```

```
[ ] test_set["Predictions"] = y_pred
```

```
Accuracy Of SVM For The Given Dataset :  0.887556904400607
              precision    recall  f1-score   support

           0       0.89      1.00      0.94      5849
           1       0.00      0.00      0.00       741

    accuracy                           0.89      6590
   macro avg       0.44      0.50      0.47      6590
weighted avg       0.79      0.89      0.83      6590
```



Result and Discussion:

Prediction on the Test dataset

We have to perform the same preprocessing operations on the test data that we have performed on the train data. But here we already have pre-processed data which is present in the csv file new_test.csv

```
[ ]  test_data= pd.read_csv("new_test.csv")
     test_data.head()
```

	age	job	marital	education	default	housing	loan	contact	month	day_of_week	duration	campaign	poutcome
0	32	4	0	6	0	0	0	0	3	3	131	5	1
1	37	10	3	6	0	0	0	0	4	3	100	1	1
2	55	5	0	5	1	2	0	0	3	2	131	2	1
3	44	2	1	0	1	0	0	1	4	3	48	2	1
4	28	0	2	3	0	0	0	0	5	0	144	2	1

- Depending on the metric score obtained from both the ML Algorithms, comparing them and whose metric score will be better.
- We will Predict the value of our target variable using that algorithm.
- Random Forest classifier has given the best metric score on the validation data.

```
[ ]  # predicting the test data
     y_predicted= randomsearch_rf.predict(test_data)
     y_predicted
```

```
array([0, 0, 0, ..., 1, 0, 0])
```

```
# dataset of predicted values for target variable y
prediction= pd.DataFrame(y_predicted, columns=["y_predicted"])
prediction_dataset= pd.concat([test_data, prediction], axis=1)
prediction_dataset['y_predicted'] = prediction_dataset['y_predicted'].map({1: 'yes', 0: 'no'})
prediction_dataset.to_csv('Output.csv')
prediction_dataset
```

	age	job	marital	education	default	housing	loan	contact	month	day_of_week	duration	campaign	poutcome	y_predicted
0	32	4	0	6	0	0	0	0	3	3	131	5	1	no
1	37	10	3	6	0	0	0	0	4	3	100	1	1	no
2	55	5	0	5	1	2	0	0	3	2	131	2	1	no
3	44	2	1	0	1	0	0	1	4	3	48	2	1	no
4	28	0	2	3	0	0	0	0	5	0	144	2	1	no
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
8233	48	4	1	2	0	2	0	0	6	3	554	1	1	yes
8234	30	7	2	3	0	2	0	0	6	0	159	1	1	no
8235	33	7	1	3	0	0	0	0	4	1	472	1	0	yes
8236	44	1	1	1	0	2	2	1	6	1	554	5	1	no
8237	42	1	1	2	1	2	0	0	6	3	83	5	1	no

8238 rows × 14 columns

10. Conclusion

Hence in our project, we have done a lot of things to get the performance we wanted: -

The purpose of our project was to predict whether a customer would register a long-time deposit or not provide customer data.
There were a wide range of categories and other numerical variables that capture a variety of customer information and bank-customer relationships.
We first did an EDA and found that there are no empty data values, and the data is not equal, where "no" is a majority class. After a consistent analysis we found that the day_of_the week is not very helpful when it comes to predicting target fluctuations. But on the other hand, some numerical features often predict the best-intentioned variable.
After performing the data visualization with T-SNE, it became clear that the data did not have very good differences between classes, as the two classes were significantly different even after trying different parameters.
After pre-processing the basic data, we encoded the categorical data with Label Encoding Method.
After that we trained our data using five models: Logistic Regression, Random Forest, ANN, LSTM and SVM.
After using all the models, we realized that the model that provided the best performance was the Random Forest and the AUC test was 0.904 which was somewhat similar or better to the results the researchers found in the related research papers.
The reason could be attested to the fact the dataset required heavy classification along with regression techniques which are provided by random forest and SVM better and their accuracies do support these facts.

11. References

- https://www.researchgate.net/publication/227441142_Logistic_regression_in_data_analysis_An_overview
- https://towardsdatascience.com/understanding-random-forest-58381e0602d2
- https://www.researchgate.net/publication/331563946_A_Machine_Learning_Approach_to_Identify_Potential_Customer_Based_on_Purchase_Behavior
- https://towardsdatascience.com/hyper-parameter-tuning-and-model-selection-like-a-movie-star-a884b8ee8d68
- https://www.sciencedirect.com/science/article/pii/S187705091401309X
- http://nebula.wsimg.com/afc36d99e2de075a106c66797c86161d?AccessKeyId=DFB1BA3CED7E7997D5B1&disposition=0&alloworigin=1
- https://ieeexplore.ieee.org/document/8644458