Bank Marketing Dataset

CIND-820 Final Result and report

Ryerson University

Table of Contents

Abstract

What Is Marketing? Marketing refers to activities a company undertakes to promote the buying or selling of a product or service. Marketing includes advertising, selling, and delivering products to consumers or other businesses. Some marketing is done by affiliates on behalf of a company.

There are many ways of the marketing and one of them is the phone marketing.

Banks depends on the phone marketing to attract the existing customer for their new products or services.

The data is related with direct marketing campaigns (phone calls) of a Portuguese banking institution. The classification goal is to predict if the client will subscribe a term deposit (variable y).

Introduction

Bank, an institution that deals in money and its substitutes and provides other money-related services.

This financial institute depends on depositing money from customers with interest rate and lending the money to other customers with higher rate.

So, the bank to be able to complete this cycle and be able to lend money invest this money in other business and generate revenues and be profitable. they should be able to grab more customers with term deposit to be able do so.

Gaining more customers require many marketing campaigns to reach this target The marketing have many ways:

1. Ads, TV, sign boards, radio, etc.
2. Digital Marketing.
3. Telemarketing.

Many banks believe that the telemarketing beside to the Bank reputation have an major effect to grab new and existing customers to the services since the agent can on spot answer all the customers inquiries and can hear from the customers about their concerns and solve them immediately or raise it to the higher management which could increase the customer satisfaction and trust in the bank and have a long term deposit.

To reach this goal banks should have accurate data base about their customers through something call Know Your Customer (KYC) and asking to update it every two years and some banks asking to do it on yearly basis.

Even some banks they are going beyond that and will freeze the account if the customer didn't update his data.

Since the banks have the data, the banks will start to reach their customers and convince them to register in one of their term deposits.

The banks could reach to information can be useful to predict the customer behavior and try to contact the customer with higher percentage of acceptance to register in this program and improve the success rate and improve the operating cost.

Many banks now start using the machine learning and data science to have good results and reach to their target by creating a mathematical model and contact the customer with high probability to be part of their marketing campaign.

This model will tell the agent that this customer has a high possibility to join our program and the agent will sort them with high probability and contact them as high priority and keep the with low probability at the end of the list.

By this the bank will achieve many targets:

1. Increase the number of term of deposits.
2. Improve the revenue streams by secure the cash for long term investments.
3. Improve the agent's utilizations and target.
4. Reduce the cost in many aspects, HR cost, phone, costs, space, etc.

The Portuguese banking institution believes that the marketing campaign are giving the required results and attracting more customers in their term deposits so he wants to make a model for their customers so the successful phone rate increased by predicting if the customer will subscribe or not.

Here will be using the UCI machine learning data for Bank Marketing Analysis.

Data are available in:

http://archive.ics.uci.edu/ml/machine-learning-databases/00222/

In summary the data structure is:

Data Set Characteristics:	Multivariate	**Number of Instances:**	45211
Attribute Characteristics:	Real	**Number of Attributes:**	17
Associated Tasks:	Classification	**Missing Values?**	N/A

Data Analysis

Data exploration and importing.

As a first step, to be able to review and studying we need to review the data and use one of the main data tools which will be Python.
Will be reading the data from the csv file, below screen shot will show the importing steps and the results out of that:

```
In [1]: import pandas as pd
        import matplotlib.pyplot as plt
        import seaborn as sns
        %matplotlib inline
        from sklearn.model_selection import train_test_split
        import warnings
        warnings.filterwarnings("ignore")
```

```
In [3]: df = pd.read_csv("bank-full.csv",";")
        df_n = df.copy()
        df.head()
```

Out[3]:

	age	job	marital	education	default	balance	housing	loan	contact	day	month	duration	campaign	pdays	previous	poutcome	y
0	58	management	married	tertiary	no	2143	yes	no	unknown	5	may	261	1	-1	0	unknown	no
1	44	technician	single	secondary	no	29	yes	no	unknown	5	may	151	1	-1	0	unknown	no
2	33	entrepreneur	married	secondary	no	2	yes	yes	unknown	5	may	76	1	-1	0	unknown	no
3	47	blue-collar	married	unknown	no	1506	yes	no	unknown	5	may	92	1	-1	0	unknown	no
4	33	unknown	single	unknown	no	1	no	no	unknown	5	may	198	1	-1	0	unknown	no

After importing, the attributes should be checked and known:
The data input variables are from 0 – 16 with:

RangeIndex: 45211 entries, 0 to 45210
Data columns (total 17 columns):

#	Column	Count	Non-Null	Dtype
0	age	45211	non-null	int64
1	job	45211	non-null	object
2	marital	45211	non-null	object
3	education	45211	non-null	object
4	default	45211	non-null	object
5	balance	45211	non-null	int64
6	housing	45211	non-null	object
7	loan	45211	non-null	object
8	contact	45211	non-null	object
9	day	45211	non-null	int64
10	month	45211	non-null	object
11	duration	45211	non-null	int64
12	campaign	45211	non-null	int64

RangeIndex: 45211 entries, 0 to 45210 Data columns (total 17 columns):				
13	pdays	45211	non-null	int64
14	previous	45211	non-null	int64
15	poutcome	45211	non-null	object
16	y	45211	non-null	object

The attributes here are mixed between integers and objects.
Missing data is a fatal point in the predictions, so we must check if there is any missing info or not. The command is: df.isnull().sum().
The output of the command shows no missing data:

age 0	**day 0**
job 0	**month 0**
marital 0	**duration 0**
education 0	**campaign 0**
default 0	**pdays 0**
balance 0	**previous 0**
housing 0	**poutcome 0**
loan 0	**y 0**
contact 0	

dtype: int64

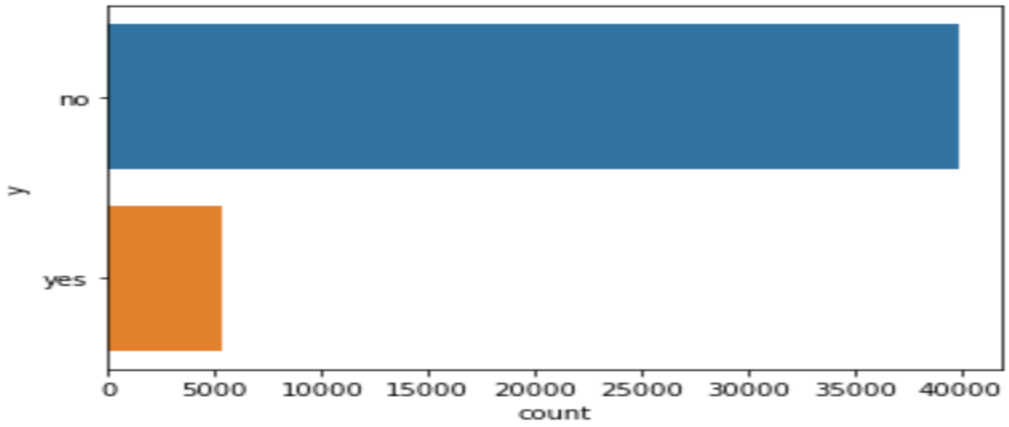
Also, the data can be described:

	age	balance	day	duration	campaign	pdays	previous
count	45211	45211	45211	45211	45211	45211	45211
mean	40.93621	1362.272	15.80642	258.1631	2.763841	40.19783	
std	10.61876	3044.766	8.322476	257.5278	3.098021	100.1287	
min	18	-8019	1	0	1	-1	
25%	33	72	8	103	1	-1	
50%	39	448	16	180	2	-1	
75%	48	1428	21	319	3	-1	
max	95	102127	31	4918	63	871	

Checking the Categorical Data.

First, we must check the target data output to see if the data are balanced or not:

no 0.88
yes 0.12
Name: y, dtype: float64



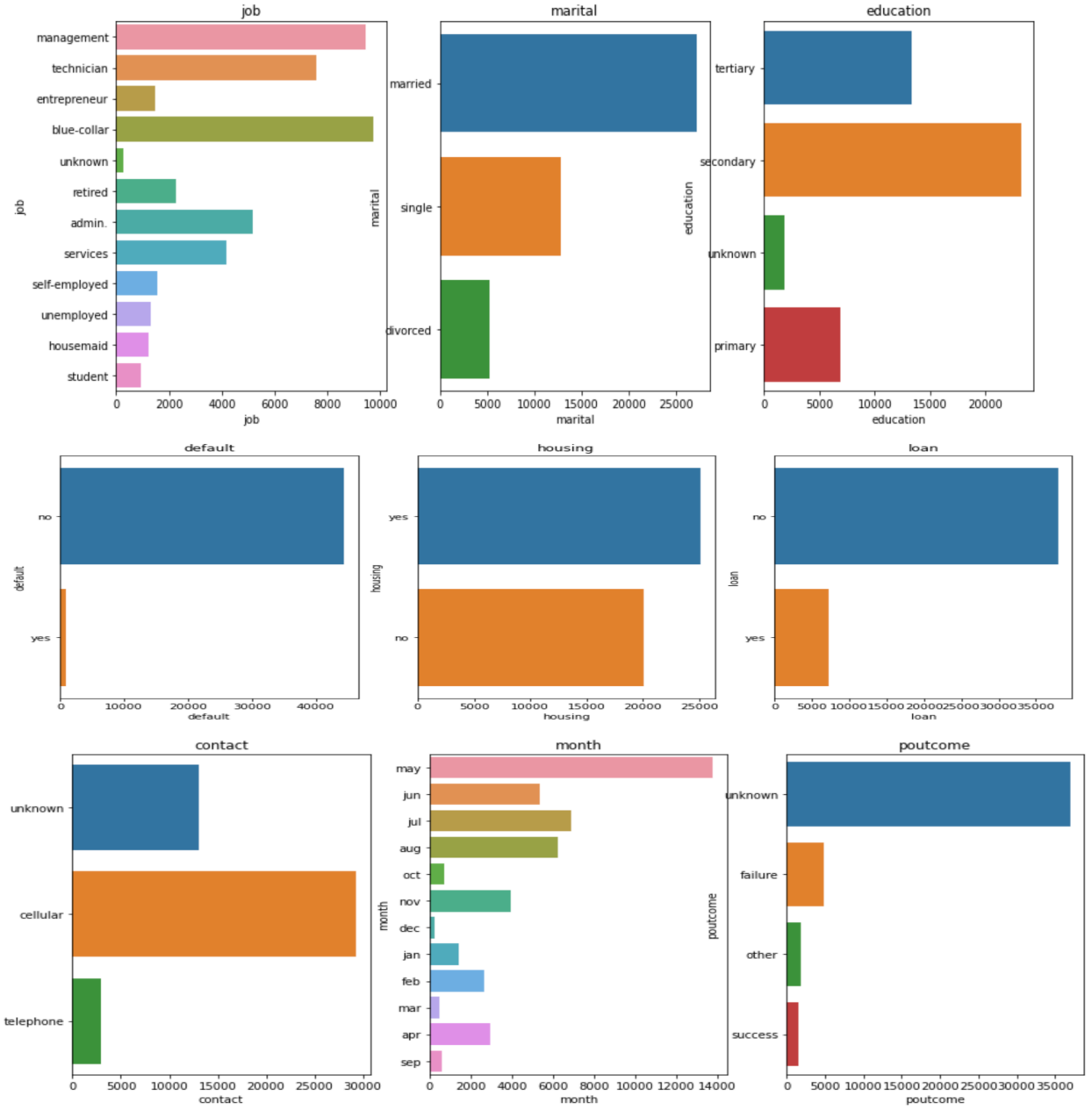
Our data showing that the target output is imbalanced.
This will make our prediction biased on **NO.**

We have 10 categorical data and they are:

- The feature is **job** and the number of categories are 12.
- The feature is **marital** and the number of categories are 3.
- The feature is **education** and the number of categories are 4.
- The feature is **default** and the number of categories are 2.
- The feature is **housing** and the number of categories are 2.
- The feature is **loan** and the number of categories are 2.
- The feature is **contact** and the number of categories are 3.
- The feature is **month** and the number of categories are 12.
- The feature is **poutcome** and the number of categories are 4.
- The feature is **y** and the number of categories are 2.

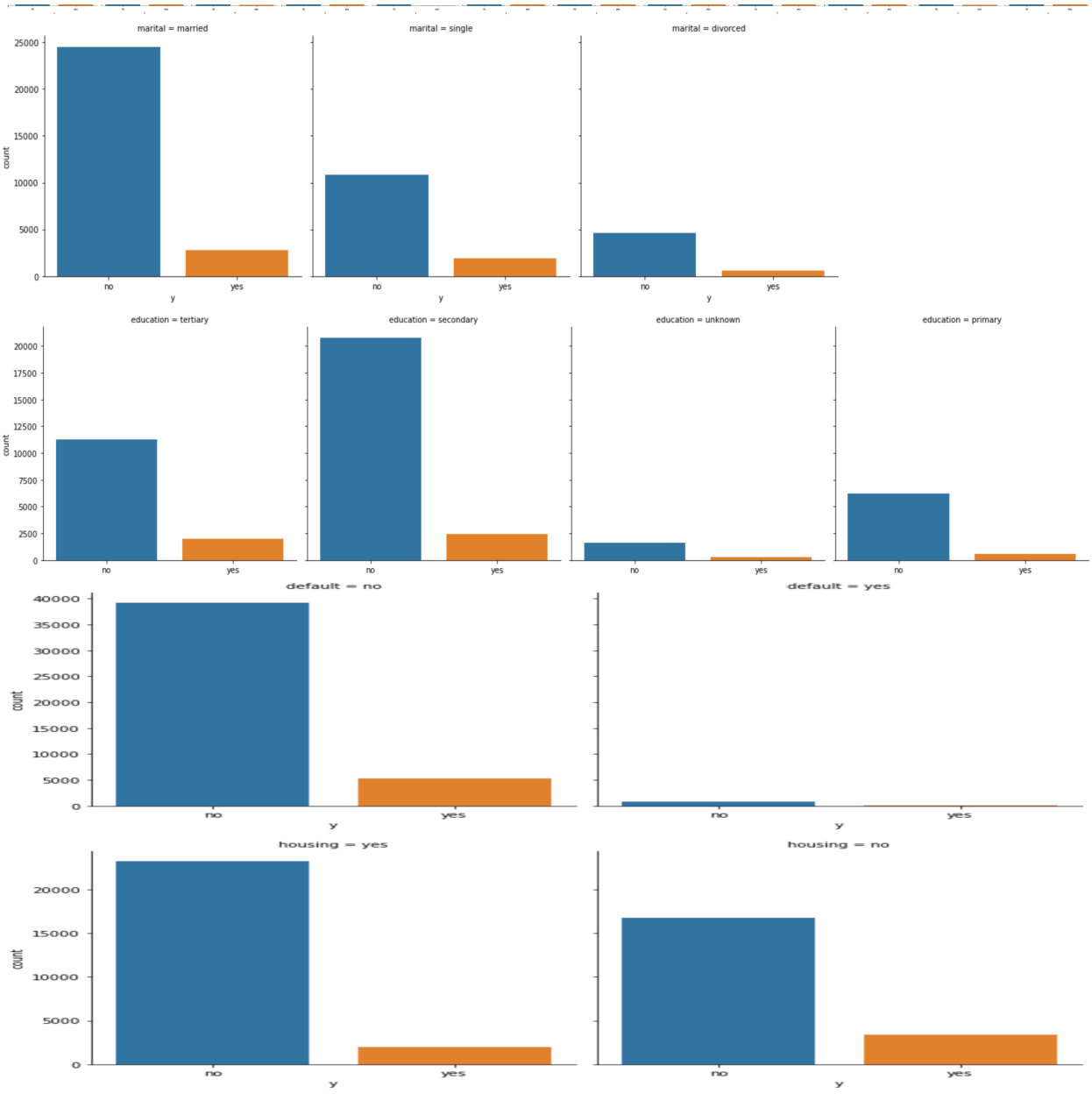
Now we must study the data for the categorical data with respect to the target data ‘y’.

And check the distribution for each feature.


job
management
technician
entrepreneur
blue-collar
unknown
retired
admin.
services
self-employed
unemployed
housemaid
student
0
2000
4000
6000
8000
10000
job
marital
married
single
divorced
0
5000
10000
15000
20000
25000
marital
education
tertiary
secondary
unknown
primary
0
5000
10000
15000
20000
education
default
no
yes
0
10000
20000
30000
40000
default
housing
yes
no
0
5000
10000
15000
20000
25000
housing
loan
no
yes
0
5000
10000
15000
20000
25000
30000
35000
loan
contact
unknown
cellular
telephone
0
5000
10000
15000
20000
25000
30000
contact
month
may
jun
jul
aug
oct
nov
dec
jan
feb
mar
apr
sep
0
2000
4000
6000
8000
10000
12000
14000
month
poutcome
unknown
failure
other
success
0
5000
10000
15000
20000
25000
30000
35000
poutcome

The relation between categorical feature and labels.

1. Check the target label split over the categorical features.
2. Find the relation with them.
    i. Even customers with No Default Majority of the time are refusing.
    ii. Candidate with Housing loan, most of the time are refusing.
    iii. Married, secondary, tertiary, and single are with High NO.



Checking the Numerical Data.

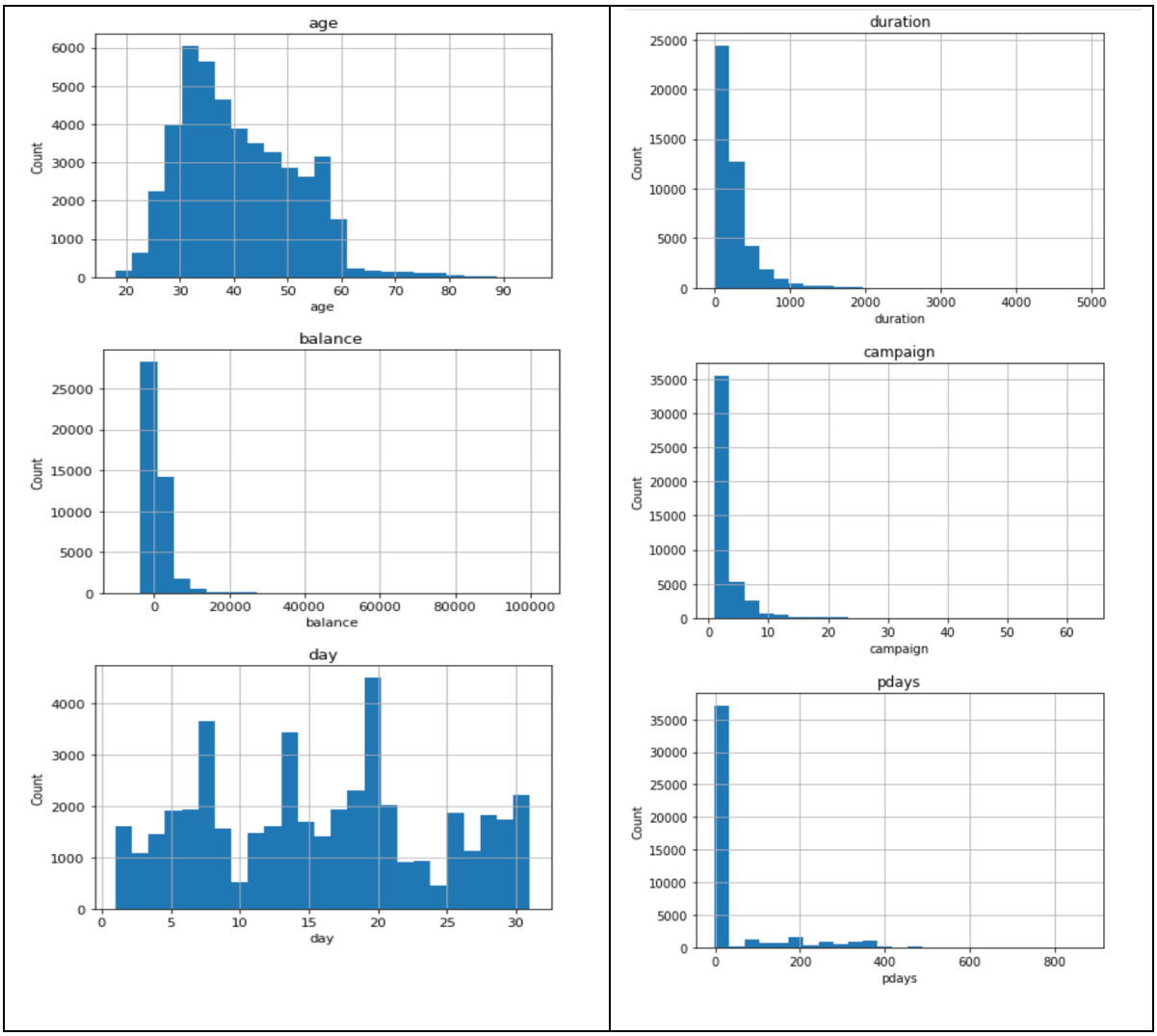
After searching and checking the Numerical attributes, we found 7 attributes are available and they are:

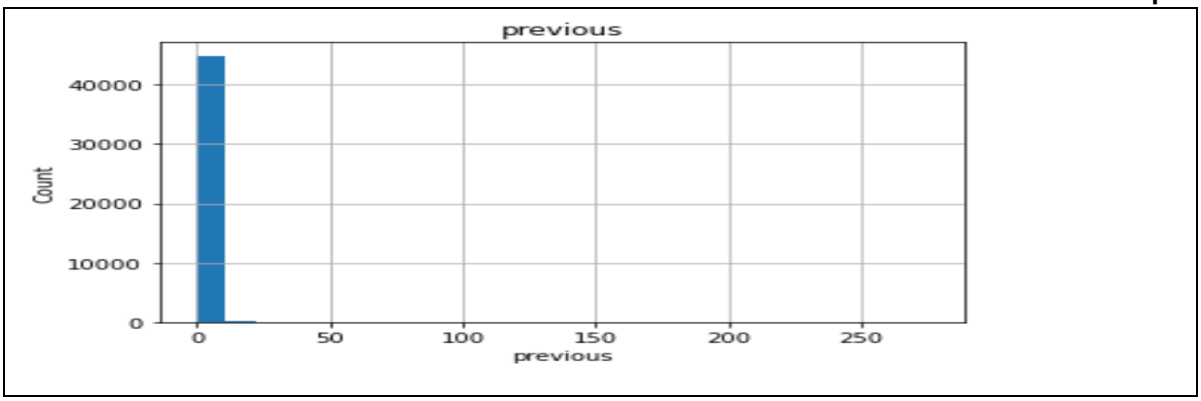
age	balance	day	duration	campaign	pdays	previous

Also,

- They are not discrete.
- They are continuous data.

Now we need to check the distribution for them:

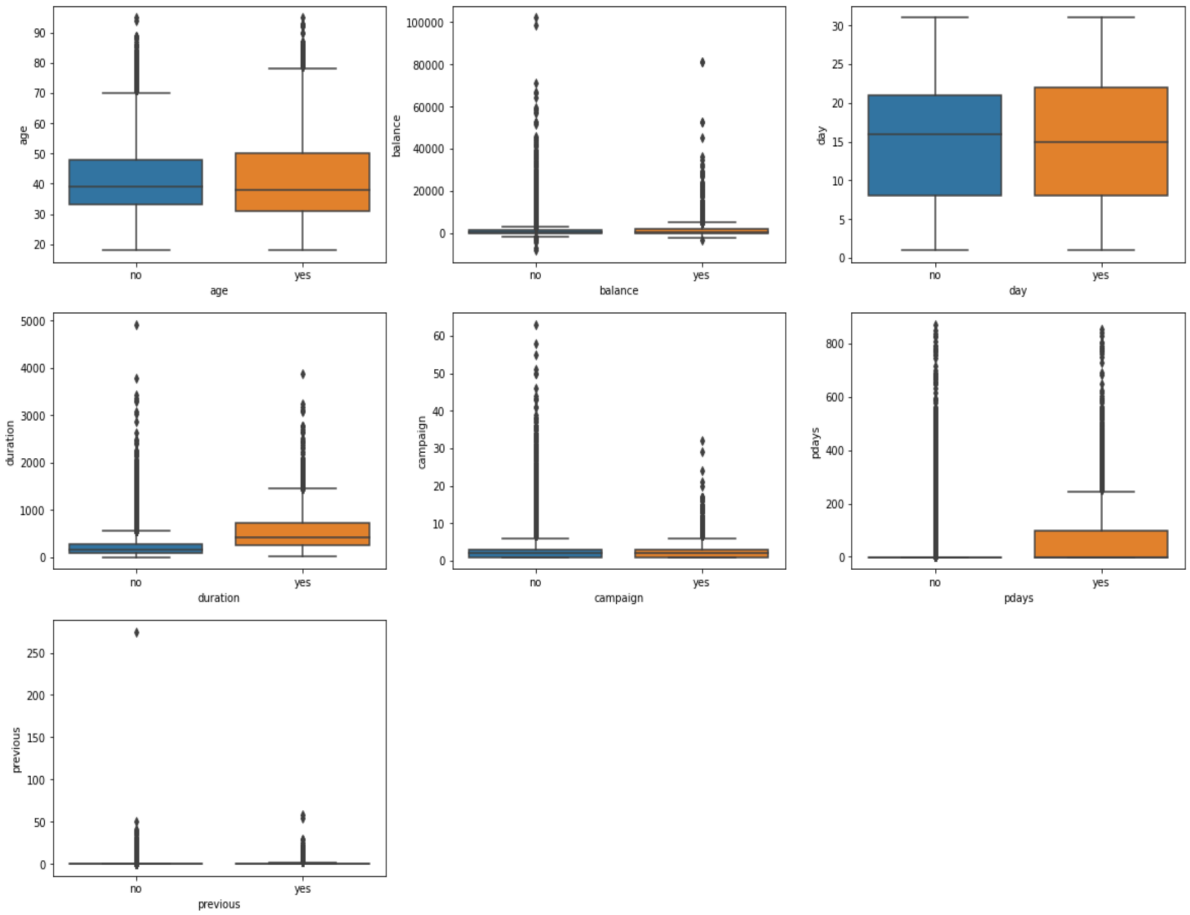




We note that the majority are left skew. From the above paragraph, we found some outliers.

Relation between continuous feature and target data.

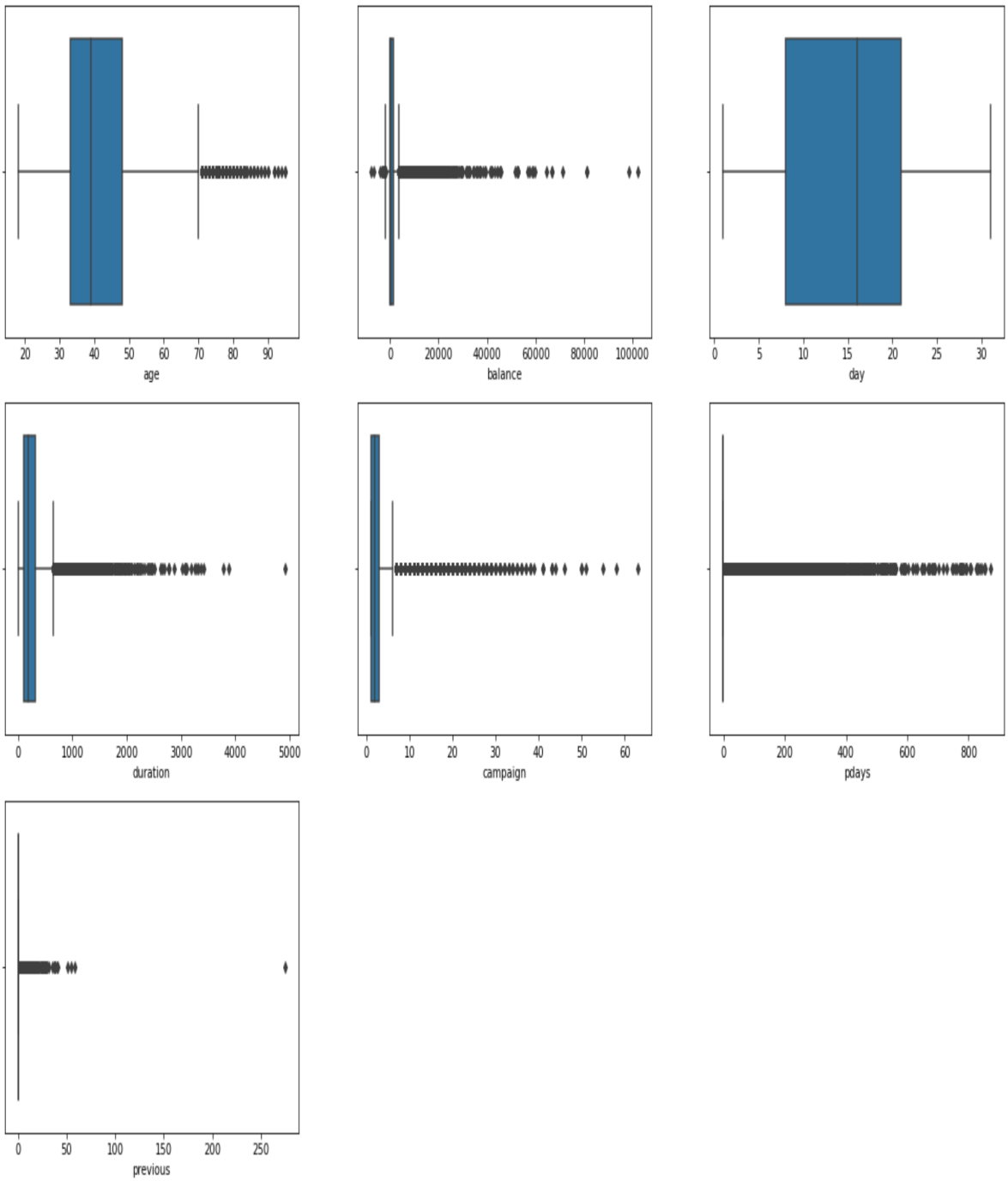
From previous chart we can notice some outliers so do check that we plot the boxplot to see the outliers.



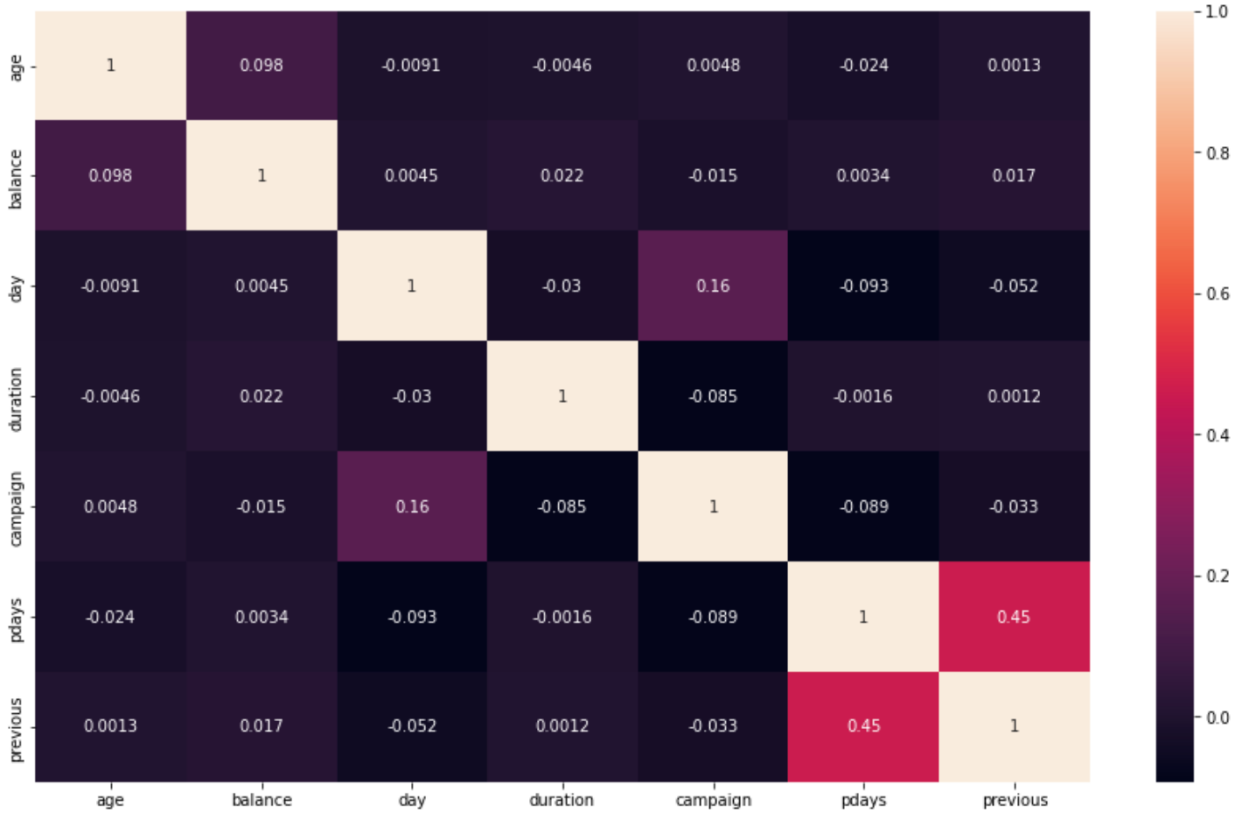
Find the outliers in numerical features.

Many points are outliers which may affect the model if we compare them with the target data ‘y’.

In the next figure we can find the outliers for most of the data, except day.



Explore the correlation between these numerical features.



From the heat map we can note that almost zero correlation magnitude for the numerical fact except for the Previous and pdays, they have low positive correlation 0.45.

Feature dropping.

Studying the features based on the above charts we can conclude:

- Drop the default feature since **they are highly imbalance.**

```
df.groupby(['y','default']).size()
```

```
y    default
no   no         39159
     yes          763
yes  no          5237
     yes           52
dtype: int64
```

- Age, Balance, duration, campaign, pdays and day will be kept.

Scale Numerical Value.

Numerical data should be scaled.

```
from sklearn.preprocessing import StandardScaler
df = dfo.copy()
scaler = StandardScaler()
num_col = ['age', 'day', 'campaign', 'pdays','previous']
df[num_col] = scaler.fit_transform(df[num_col])
df.head()
```

	age	job	marital	education	default	balance	housing	loan	contact	day	month	duration	campaign	pdays	previous	poutcome	y
0	1.606965	management	married	tertiary	no	2143	yes	no	unknown	-1.298476	may	261	-0.569351	-0.411453	-0.25194	unknown	no
1	0.288529	technician	single	secondary	no	29	yes	no	unknown	-1.298476	may	151	-0.569351	-0.411453	-0.25194	unknown	no
2	-0.747384	entrepreneur	married	secondary	no	2	yes	yes	unknown	-1.298476	may	76	-0.569351	-0.411453	-0.25194	unknown	no
3	0.571051	blue-collar	married	unknown	no	1506	yes	no	unknown	-1.298476	may	92	-0.569351	-0.411453	-0.25194	unknown	no
4	-0.747384	unknown	single	unknown	no	1	no	no	unknown	-1.298476	may	198	-0.569351	-0.411453	-0.25194	unknown	no

Encode the categorical Values.

Using the hot coding in python to transfer the categorical data to numbers to be able used in the model for predictions and predict the target data.

	job_admin.	job_blue-collar	job_entrepreneur	job_housemaid	job_management	job_retired	job_self-employed	job_services	job_student	job_technician	...	poutcome_success	poutcome_unknown	age	balance	day	duration	campaign	pdays	previous	y
0	0	0	0	0	1	0	0	0	0	0	...	0	1	58	2143	5	261	1	-1	0	0
1	0	0	0	0	0	0	0	0	0	1	...	0	1	44	29	5	151	1	-1	0	0
2	0	0	1	0	0	0	0	0	0	0	...	0	1	33	2	5	76	1	-1	0	0
3	0	1	0	0	0	0	0	0	0	0	...	0	1	47	1506	5	92	1	-1	0	0
4	0	0	0	0	0	0	0	0	0	0	...	0	1	33	1	5	198	1	-1	0	0

Modeling.

- First, we must split the data into training and testing.
  The Ratio it will be 80% Training 20% Testing.
- In the prediction modeling will be using 4 models:
  - Random Forest.
  - Logistic Regression.
  - K-Nearest Neighbors (KNN).
  - Decision Tree.

The score for each model is:

	Random	Logistic Reg	KNN	Decision
Score	90.82	89.91	89.62	88.13

Random forest output.

	precision	recall	f1-score	support
0	0.93	0.97	0.95	7993
1	0.66	0.43	0.52	1050
accuracy			0.91	9043
Macro Avg	0.8	0.7	0.73	9043
Weighted Avg	0.9	0.91	0.9	9043

Confusion Matrix for Random Forest is:	7764	229
	601	449

Logistic Regression output.

	precision	recall	f1-score	support
0	0.91	0.98	0.94	7993
1	0.64	0.3	0.41	1050
accuracy			0.90	9043
Macro Avg	0.78	0.64	0.68	9043
Weighted Avg	0.88	0.90	0.88	9043

Confusion Matrix for Logistic Regresion is:	7820	173
	739	311

KNN output.

	precision	recall	f1-score	support
0	0.91	0.98	0.94	7993
1	0.63	0.25	0.36	1050
accuracy			0.90	9043
Macro Avg	0.77	0.62	0.65	9043
Weighted Avg	0.88	0.90	0.88	9043

Confusion Matrix for KNN is:	7838	155
	784	266

Decision Tree output.

	precision	recall	f1-score	support
0	0.94	0.93	0.93	7993
1	0.49	0.52	0.50	1050
accuracy			0.88	9043
Macro Avg	0.71	0.72	0.72	9043
Weighted Avg	0.88	0.88	0.88	9043

Confusion Matrix for Decision Tree is:	7431	562
	511	539

Results

The random forest is the best score, and it has 90.82% compared to the others but KNN has the highest Positive prediction in the confusion matrix with 7838 correct predictions compared to 7764 for the RF.

Based on that we can consider our KNN model the best model to be used.

KNN output:

	KNN
Score	89.62

	precision	recall	f1-score	support
0	0.91	0.98	0.94	7993
1	0.63	0.25	0.36	1050
accuracy			0.90	9043
Macro Avg	0.77	0.62	0.65	9043
Weighted Avg	0.88	0.90	0.88	9043

Confusion Matrix for KNN is:	7838	155
	784	266

Annexes

References.

- Moro, S., Cortez, P., & Rita, P. (2014). A Data-Driven Approach to Predict the Success of Bank Telemarketing. Decision Support Systems, 62, 22-31. https://doi.org/10.1016/j.dss.2014.03.001

- Bank Marketing Data Classification Using Machine Learning Dr. Smitha Shekar B1 , Pooja A2 Associate Professor, Department of CS&E, Dr.A.I.T, Bengaluru, India1 M.Tech Student, Department of CS&E, Dr.A.I.T, Bengaluru, India2

- Tuba Parlar1 , Songul Kakilli Acaravci2 * 1 Vocational School of Antakya, Department of Computer Technology, Mustafa Kemal University, Antakya, Hatay, Turkey, 2 Faculty of Economics and Administrative Sciences, Department of Finance and Accounting, Mustafa Kemal University, Hatay, Turkey.

- https://github.com/np788/bankmarketing/blob/master/Bank_Marketing_Project.ipynb.

- https://medium.com/swlh/using-machine-learning-to-predict-subscription-to-bank-term-deposits-for-clients-with-python-aec8a4690807.

- https://www.youtube.com/watch?v=z8VE71abh_k.

- https://rstudio-pubs-static.s3.amazonaws.com/793938_04e78c2ce4d04878b722f5e001f9ff90.html.

- https://www.kaggle.com/datasets/janiobachmann/bank-marketing-dataset.

The Code.

```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
get_ipython().run_line_magic('matplotlib', 'inline')
from sklearn.model_selection import train_test_split
import warnings
warnings.filterwarnings("ignore")


# In[ ]:


dfo = pd.read_csv("bank-full.csv",";")
df = dfo.copy()
df.head()


# In[ ]:


df.isnull().sum()


# In[ ]:


df.describe()


# In[ ]:


df.columns


# In[ ]:


df.shape


# In[ ]:


df.info()
```

```
# In[ ]:


print(round(df.y.value_counts(normalize=True),2))
sns.countplot(y = 'y',data = df)


# In[ ]:


for col in df.select_dtypes(include = 'object').columns:
    print(col)
    print(df[col].unique())


# In[ ]:


cat_feat = [feature for feature in df.columns if df[feature].dtypes =='O']
for feature in cat_feat:
    print("The feature is {} and the number of categories are {}".format(feature,
len(df[feature].unique())))


# In[ ]:


plt.figure(figsize=(15,88),facecolor='white')
plotnumber = 1

for categorical_feature in cat_feat:
    ax = plt.subplot(12,3,plotnumber)
    sns.countplot(y = categorical_feature,data = df)
    plt.xlabel(categorical_feature)
    plt.title(categorical_feature)
    plotnumber+=1
plt.show()


# In[ ]:


for categorical_feature in cat_feat:
    sns.catplot(x = 'y',col = categorical_feature,kind = 'count', data = df)
plt.show()


# In[ ]:
```

```
num_feat = [feature for feature in df.columns if df[feature].dtypes !='O']
print("Number of Numerical Variable is: ",len(num_feat) )
df[num_feat].head()


# In[ ]:


disc_feat = [feature for feature in num_feat if len(df[feature].unique()) <25]
print("Number of Discrete Variable is:", len(disc_feat))


# In[ ]:


cont_feat = [feature for feature in num_feat if feature not in disc_feat+['y']]
print("Number of Continuous Variable is:", len(cont_feat))


# In[ ]:


for features in num_feat:
    df2 = df.copy()
    df2[features].hist(bins=25)
    plt.xlabel(features)
    plt.ylabel('Count')
    plt.title(features)
    plt.show()


# In[ ]:


plt.figure(figsize=(20,60),facecolor='white')
plotnumber = 1
for cont_f in cont_feat:
    ax = plt.subplot(12,3,plotnumber)
    sns.boxplot(x='y',y=df[cont_f],data = df)
    plt.xlabel(cont_f)
    plotnumber+=1
plt.show()


# In[ ]:
```

```
plt.figure(figsize=(20,60),facecolor='white')
plotnumber = 1
for num_f in num_feat:
    ax = plt.subplot(12,3,plotnumber)
    sns.boxplot(df[num_f])
    plt.xlabel(num_f)
    plotnumber+=1
plt.show()


# In[ ]:


cor_mat = df.corr()
fig=plt.figure(figsize=(15,9))
sns.heatmap(cor_mat,annot = True)


# In[ ]:


df.groupby(['y','default']).size()


# In[ ]:


df.drop(['default'],axis = 1, inplace = True)


# In[ ]:


df.groupby(['y','pdays']).size()


# In[ ]:


df.drop(['pdays'],axis = 1,inplace = True)


# In[ ]:


df.groupby(['age'],sort=True)['age'].count()
```

```
# In[ ]:


df.groupby(['y','balance'],sort=True)['balance'].count()


# In[ ]:


df.groupby(['y','duration'],sort=True)['duration'].count()
df1 = df[df['duration'] < 5/60]
df1.groupby(['y','duration'],sort=True)['duration'].count()


# In[ ]:


df['duration'] = df['duration'].apply(lambda n:n/60).round(2)


# In[ ]:


df.groupby(['y','campaign'],sort=True)['campaign'].count()


# In[ ]:


df1 = df[df['campaign']<33]


# In[ ]:


df1.groupby(['y','campaign'],sort=True)['campaign'].count()


# In[ ]:


df1 = df[df['previous']<31]


# In[ ]:


from sklearn.preprocessing import StandardScaler
df = dfo.copy()
```

```
scaler = StandardScaler()
num_col = ['age', 'day', 'campaign', 'pdays','previous']
df[num_col] = scaler.fit_transform(df[num_col])
df.head()


# In[ ]:


from sklearn.preprocessing import OneHotEncoder
encoder = OneHotEncoder(sparse = False)
cat_col  =  ['job',  'marital',  'education',  'default',  'housing','loan',  'contact',
'month', 'poutcome']
df = dfo.copy()
df_encoded = pd.DataFrame(encoder.fit_transform(df[cat_col]))
df_encoded.columns = encoder.get_feature_names(cat_col)
df = df.drop(cat_col,axis =1)
df = pd.concat([df_encoded,df],axis =1)
df['y'] = df['y'].apply(lambda x: 1 if x=='yes' else 0)
print("Shape of Data frame is:", df.shape)
df.head()


# In[ ]:


X = df.drop(['y'],axis =1)
y = df['y']
from sklearn.model_selection import train_test_split
X_train,X_test,y_train,y_test                                                    =
train_test_split(X,y,shuffle=True,test_size=.2,random_state=1)


# In[ ]:


cols = X_train.columns
from sklearn.preprocessing import RobustScaler
scaler = RobustScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.fit_transform(X_test)
X_train = pd.DataFrame(X_train, columns = [cols])
X_test = pd.DataFrame(X_test, columns = [cols])


# In[ ]:
```

```
from sklearn.ensemble import RandomForestClassifier
rfc = RandomForestClassifier(n_estimators=100,random_state=0)
rfc.fit(X_train, y_train)
R_pred = rfc.predict(X_test)


# In[ ]:


from sklearn.metrics import accuracy_score, classification_report,confusion_matrix
R_score = accuracy_score(y_test,R_pred)
R_cm = confusion_matrix(y_test,R_pred)
print("Random Model Score is:",(R_score.round(4))*100)
print(classification_report(y_test,R_pred))
print("Confusion Matrix for Random Forest is:",'\n', R_cm)


# In[ ]:


from sklearn.linear_model import LogisticRegression
LG_model = LogisticRegression()
Fun_LG = LG_model.fit(X_train,y_train)
L_pred = LG_model.predict(X_test)


# In[ ]:


L_score = accuracy_score(y_test,L_pred)
L_cm = confusion_matrix(y_test,L_pred)
print("Logistic Model Score is:",(L_score.round(4))*100)
print(classification_report(y_test,L_pred))
print("Confusion Matrix for Logistic Regression is:",'\n', L_cm)


# In[ ]:


from sklearn.neighbors import KNeighborsClassifier
knn = KNeighborsClassifier(n_neighbors = 20)
K_model= knn.fit(X_train, y_train)
K_pred = K_model.predict(X_test)


# In[ ]:
```

```
K_score = accuracy_score(y_test,K_pred)
K_cm = confusion_matrix(y_test,K_pred)
print("KNN Score is:",(K_score.round(4))*100)
print(classification_report(y_test,K_pred))
print("Confusion Matrix for KNN is:",'\n', K_cm)


# In[ ]:


from sklearn.tree import DecisionTreeClassifier
clf = DecisionTreeClassifier()
DTree = clf.fit(X_train,y_train)
D_pred = DTree.predict(X_test)


# In[ ]:


D_score = accuracy_score(y_test,D_pred)
D_cm = confusion_matrix(y_test,D_pred)
print("Decision Tree Score is:",(D_score.round(4))*100)
print(classification_report(y_test,D_pred))
print("Confusion Matrix for Decision Tree is:",'\n', D_cm)
```

The Output file.