# CAPSTONE PROJECT

## Prediction to subscribe for a term deposit

## INTERIM REPORT

**Mentored by:**

Mr. Saurav Reddy

**Submitted by:** (Group – 12)

Ms. Diksha G Nayak

Ms. Gifty Maria Paul

Mr. Navneet Sankhla

Mr. Sai Nithil Gopu

Mr. Sameer Mannur

## Table of Contents:

# Overview:

## Industry Background and Key Objectives

Term deposits are a major source of income for a bank. A term deposit is a cash investment held at a financial institution. Your money is invested for an agreed rate of interest over a fixed amount of time, or term. The bank has various outreach plans to sell term deposits to their customers such as email marketing, advertisements, telephonic marketing, and digital marketing. Telephonic marketing campaigns still remain one of the most effective ways to reach out to people.

We are an in-house analytics team at this leading ecommerce player, tasked with better understanding order rejection by customers at various stages of delivery.

In particular, we have been assigned a dataset containing orders for women’s clothing, a certain portion of which have been cancelled due to various reasons.

## Business problem statement:

### 1) Business Problem Understanding

Term deposits are a major source of income for a bank. A term deposit is a cash investment held at a financial institution. Your money is invested for an agreed rate of interest over a fixed amount of time, or term. The bank has various outreach plans to sell term deposits to their customers such as email marketing, advertisements, telephonic marketing, and digital marketing. Telephonic marketing campaigns still remain one of the most effective ways to reach out to people. However, they require huge investment as large call centres are hired to actually execute these campaigns. Hence, it is crucial to identify the customers most likely to convert beforehand so that they can be specifically targeted via call. The data is related to direct marketing campaigns (phone calls) of a Portuguese banking institution. The classification goal is to predict if the client will subscribe to a term deposit (variable y).

## 2) Business Objective

Broadly speaking, recent developments in machine learning techniques and data mining has led to an interest of implementing these techniques in various fields.

We are trying to predict the customers for subscribing to the term deposit of our bank and we can do this by finding those factors which is affecting the customers to not take the Term deposit.

## Data Dictionary:

The bank client data**:** consists of below mentioned attributes:

- **age** (numeric)
- **job**: type of job (categorical: "admin.","unknown","unemployed","management","housemaid","entrepreneur","student","blue-collar", "self-employed", "retired", "technician", "services")
- **marital**: marital status (categorical: "married", "divorced", "single"; note: "divorced" means divorced or widowed
- **education**: (categorical: "unknown", "secondary", "primary", "tertiary")
- **default**: has credit in default? (Binary: "yes", "no")
- **balance**: average yearly balance, in euros (numeric)
- **housing**: has housing loan? (Binary: "yes", "no")
- **loan**: has personal loan? (Binary: "yes", "no")
  # related with the last contact of the current campaign:
- **contact**: contact communication type (categorical: "unknown", "telephone", "cellular")
- **day**: last contact day of the month (numeric)
- **month**: last contact month of year (categorical: "jan", "feb", "mar", …, "nov", "dec")
- **duration**: last contact duration, in seconds (numeric)
  # other attributes:
- **campaign**: number of contacts performed during this campaign and for this client (numeric, includes last contact)
- **pdays**: number of days that passed by after the client was last contacted from a previous campaign (numeric, -1 means client was not previously contacted)
- **previous**: number of contacts performed before this campaign and for this client (numeric)

- **poutcome**: outcome of the previous marketing campaign (categorical: "unknown", "other", "failure", "success")

**Output variable (desired target):**

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

## Data Pre-processing:

We begin by reading in the default csv file and getting a sense of the overall size and features we will be working with.

**Our default data file includes:**

Number of Columns:17
Total Number of Records: 45211

**Imputation of missing values:**

Our initial search helps us identify missing values. None of the columns have any missing values.

Hence, we won't be imputing null values.

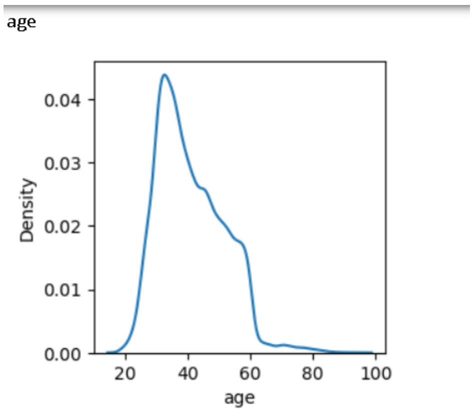
- In case there were null values in numerical variables we could have a look at the presence of outliers depending on the presence and absence we could impute them with either mean or median.
- In case there were null values in categorical variables we could impute them with mode.

```
age          0
job          0
marital      0
education    0
default      0
balance      0
housing      0
loan         0
contact      0
day          0
month        0
duration     0
campaign     0
pdays        0
previous     0
poutcome     0
y            0
dtype: int64
```

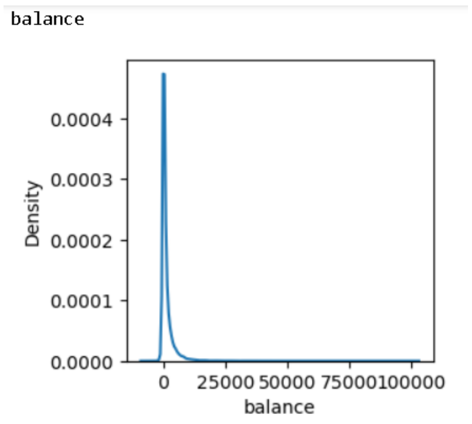
## Initial Data Exploration - Analysing Relationships between the Variables:

### Univariate analysis:

**Let us begin by performing univariate analysis on  numerical variables.**

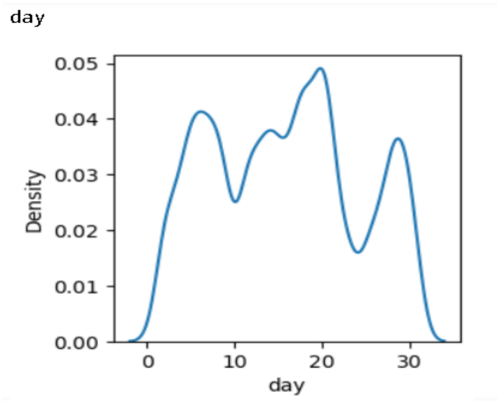


age Skewness: 0.6848179257252598, this indicates the data is right skewed.
age Kurtosis: 0.3195703759105042, it indicates that the data is leptokurtic.
Our data is unimodal and the value which has highest frequency of occurrences are 3 2.



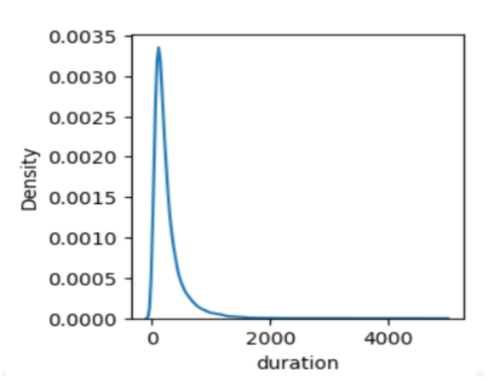
balance Skewness: 8.360308326166326, this indicates the data is right skewed.
balance Kurtosis: 140.75154662504158, it indicates that the data is leptokurtic.
Our data is unimodal and the value which has highest frequency of occurance is 0.

day



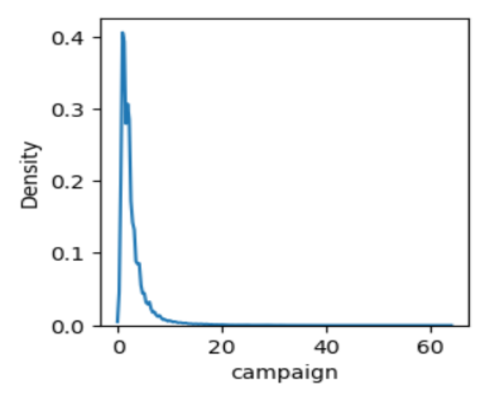
day Skewness: 0.09307901402122411, this indicates the data is right skewed.
day Kurtosis: -1.0598973728286003, it indicates that the data is leptokurtic.
Our data is multimodal and the value which has highest frequency of occurrence is 2 0.

duration



duration Skewness: 3.144318099423456, this indicates the data is right skewed.
duration Kurtosis: 18.153915269019706, it indicates that the data is leptokurtic.
Our data is unimodal and the value which has highest frequency of occurrences 124.

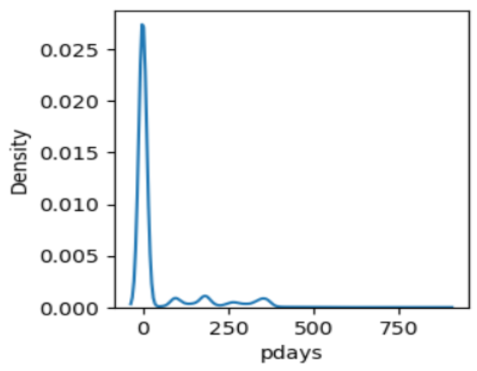
campaign



campaign Skewness: 4.898650166179674, this indicates the data is right skewed.
campaign Kurtosis: 39.2496508023021, it indicates that the data is leptokurtic.

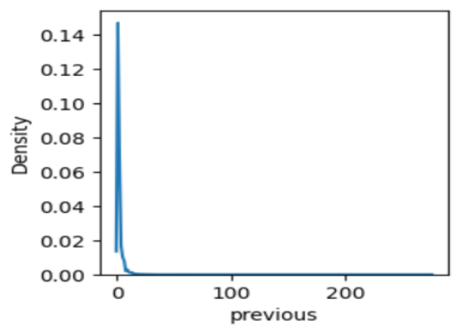
Our data is unimodal and the value which has highest frequency of occurance is 1.

pdays



pdays Skewness: 2.6157154736563477, this indicates the data is right skewed.
pdays Kurtosis: 6.935195210422799, it indicates that the data is leptokurtic.
Our data is unimodal and the value which has highest frequency of occurrence is -1.

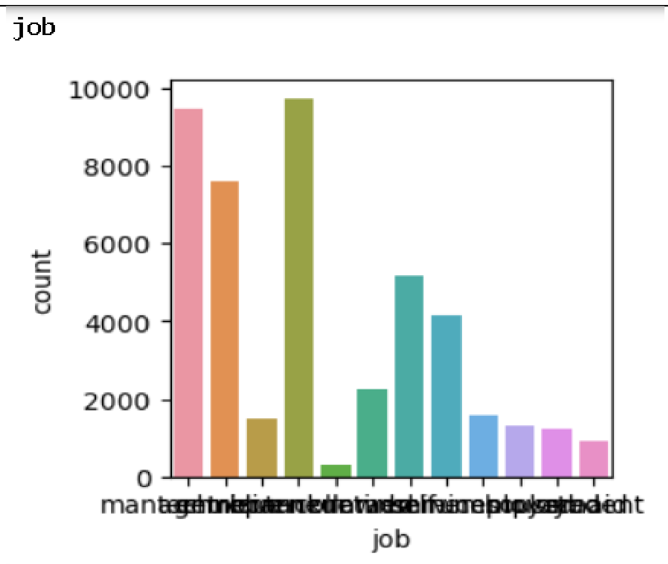
previous



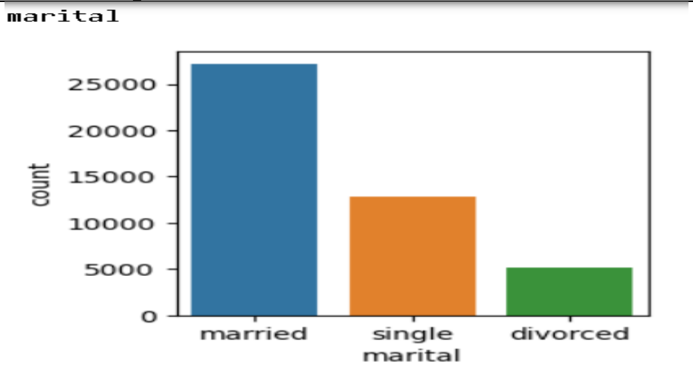
previous Skewness: 41.84645447266292, this indicates the data is right skewed.
previous Kurtosis: 4506.860660183261, it indicates that the data is leptokurtic.
Our data is unimodal and the value which has highest frequency of occurrence is 0.

**Let us begin by performing univariate analysis on categorical variables.**

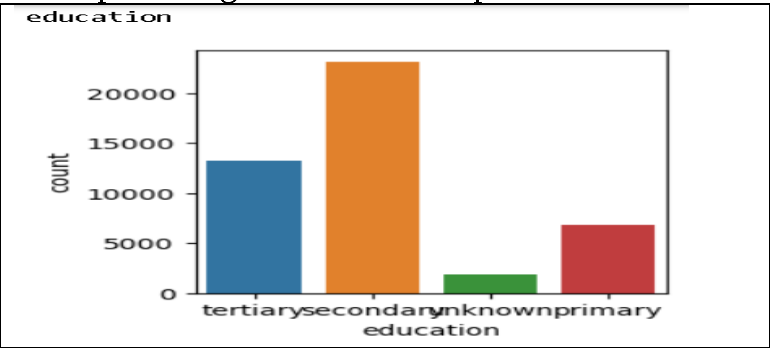
variable: job has these categories with count of:
Blue-collar employees and Management employees opt most for term deposit.

| | | |
|---|---|---|
| blue-collar | 9732 |  |
| management | 9458 | |
| technician | 7597 | |
| admin. | 5171 | |
| services | 4154 | |
| retired | 2264 | |
| self-employed | 1579 | |
| entrepreneur | 1487 | |
| unemployed | 1303 | |
| housemaid | 1240 | |
| student | 938 | |
| unknown | 288 | |

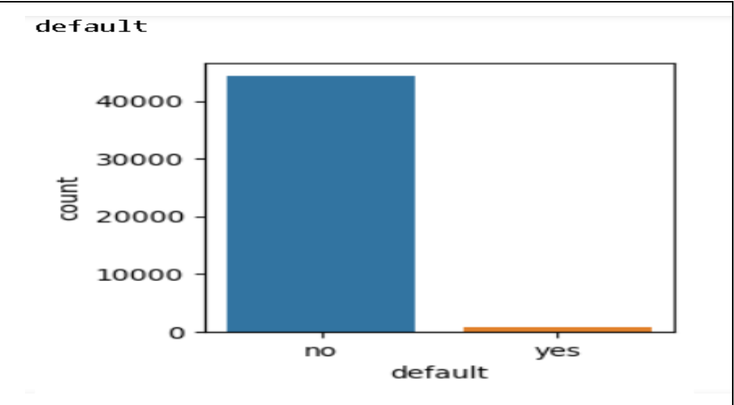
variable: marital has these categories with count of:
Married people opt the highest for term deposit.

| | | |
|---|---|---|
| married | 27214 |  |
| single | 12790 | |
| divorced | 5207 | |

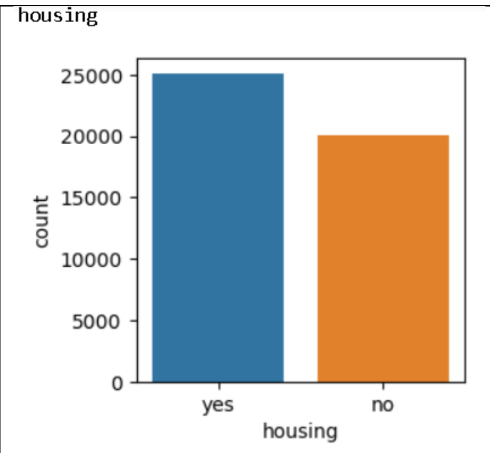
variable: education has these categories with count of:
People who have secondary education opt the highest for term deposit.

| | |
|---|---|
| secondary 23202<br>tertiary 13301<br>primary 6851<br>unknown 1857 |  |

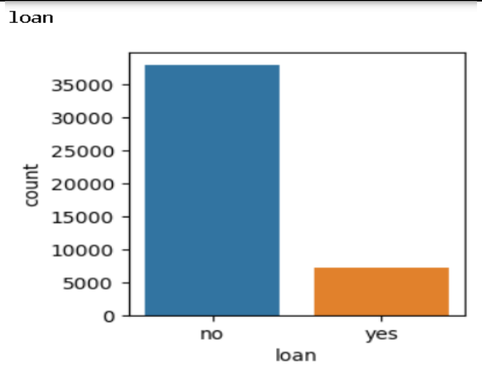
variable: default has these categories with count of:
A lot of People haven't opted term deposit by default.

| | |
|---|---|
| no 44396<br>yes 815 |  |

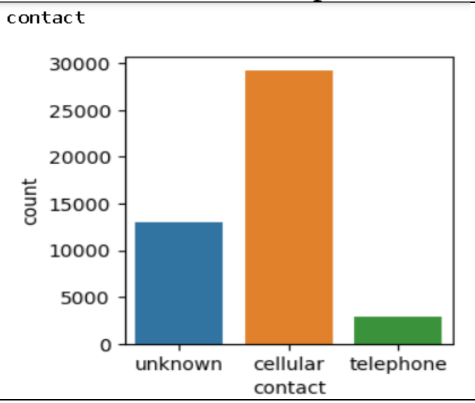
variable: housing has these categories with count of:
People who have their own house opt for term deposit.

| | |
|---|---|
| yes 25130<br>no 20081 |  |

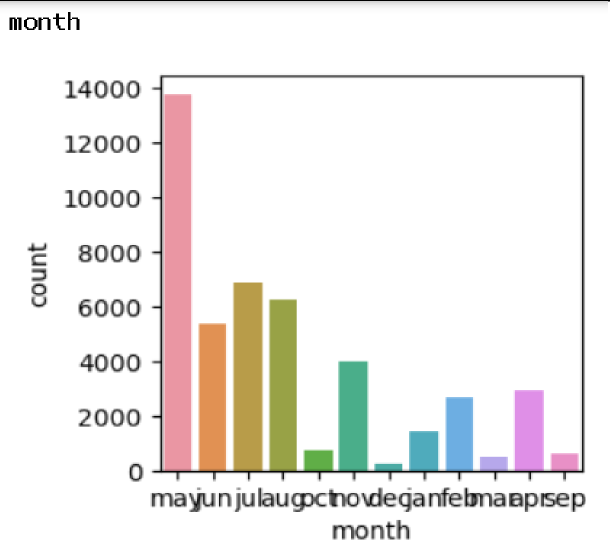
variable: loan has these categories with count of:
People who don't have any loan from bank are the highest to opt for term deposit.

| | |
|---|---|
| no 37967<br>yes 7244 |  |

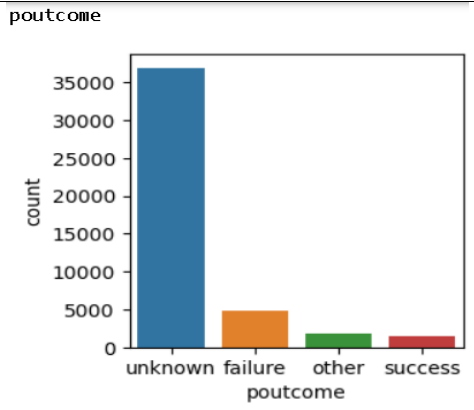
variable: contact has these categories with count of:
People who have cell phone are the highest to take these term deposit.

| | |
|---|---|
| cellular 29285<br>unknown 13020<br>telephone 2906 |  |

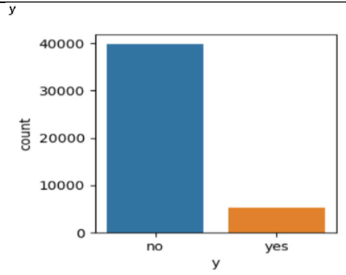
variable: month has these categories with count of:
People opt to take term deposit in highest numbers during the month of May.

| | |
|---|---|
| may 13766<br>jul 6895<br>aug 6247<br>jun 5341<br>nov 3970<br>apr 2932<br>feb 2649<br>jan 1403<br>oct 738<br>sep 579<br>mar 477<br>dec 214 |  |

variable: poutcome has these categories with count of:
Many people who are not classified as successful or failure in their career are the hig hest in numbers to take term deposit.

| | |
|---|---|
| unknown 36959<br>failure 4901<br>other 1840<br>success 1511 |  |

variable: y has these categories with count of:
Our target variable indicates that no of people who don't take term deposit are the hi ghest.

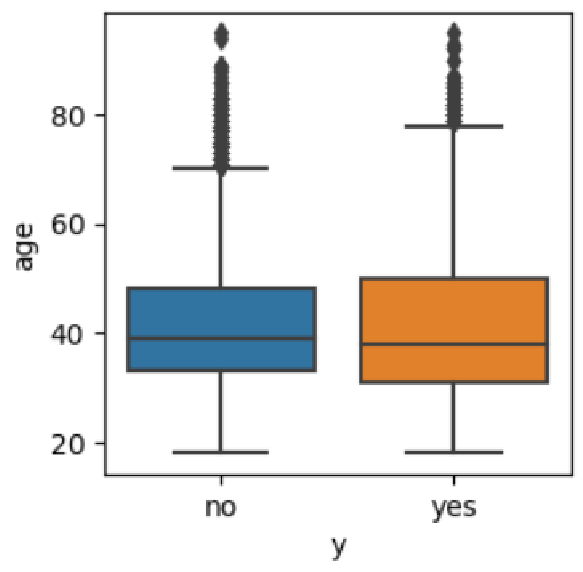
| | |
|---|---|
| no 39922<br>yes 5289 |  |

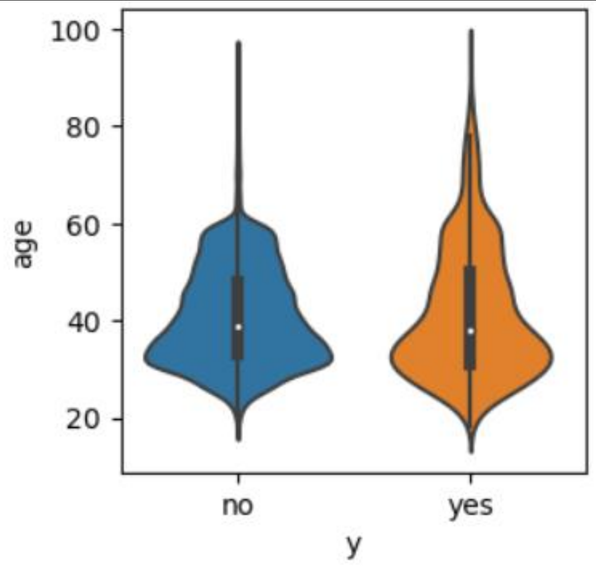
## Bivariate analysis:

Here we will attempt to understand how the available columns are impacting each other on a one-to-one basis.

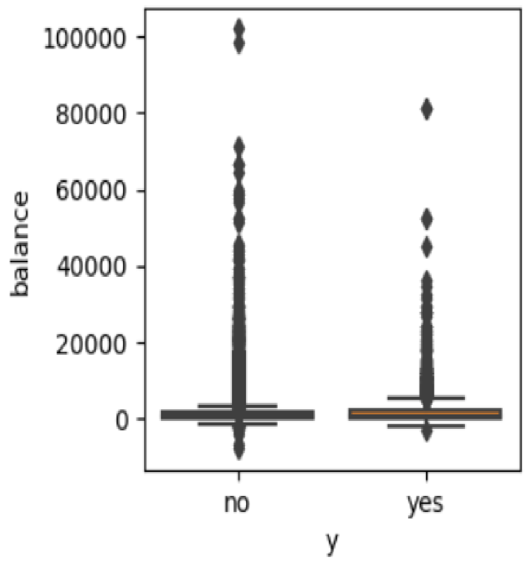
**First, we visualise the relationship between our target column (Status) and numerical features (Quantity, Amount)**

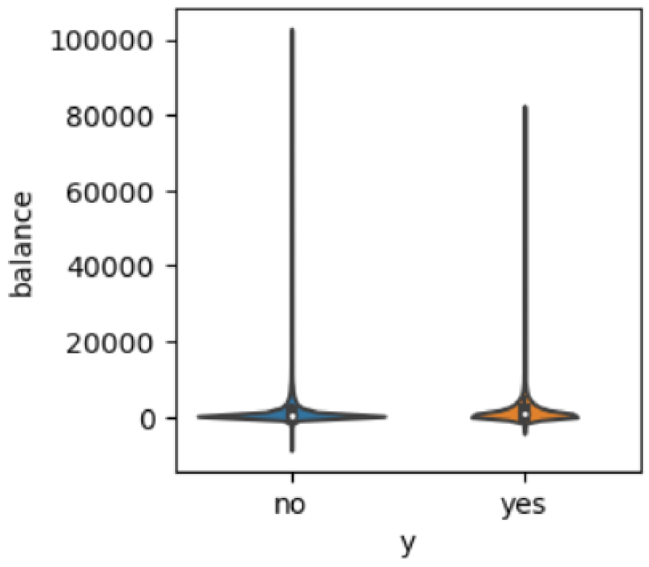
age



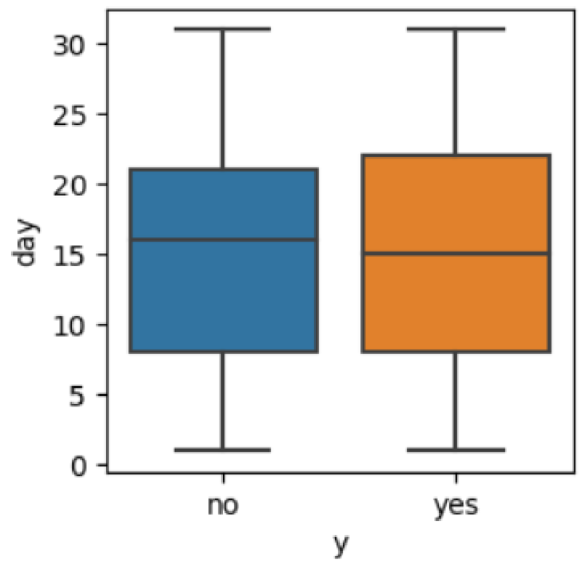


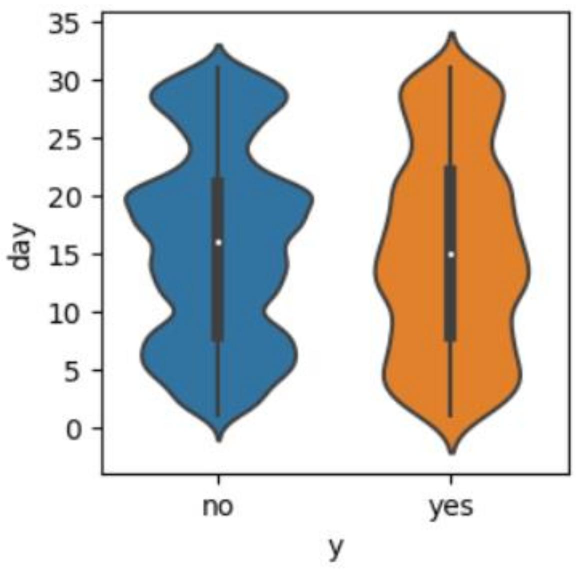
balance


100000
80000
60000
40000
20000
0
balance
no
yes
y

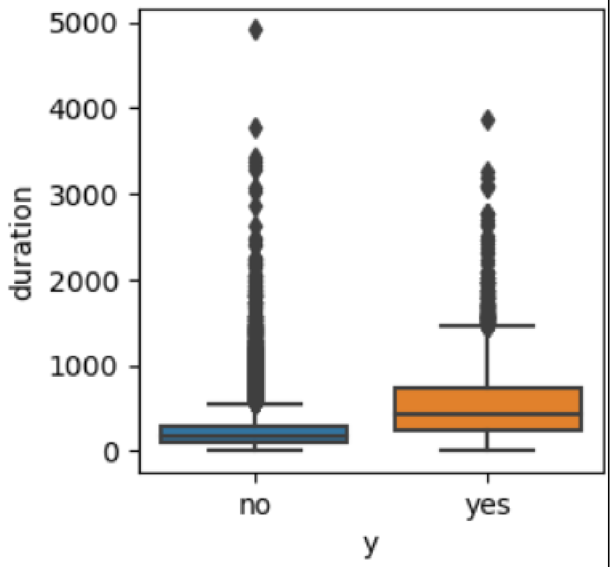

100000
80000
60000
40000
20000
0
balance
no
yes
y

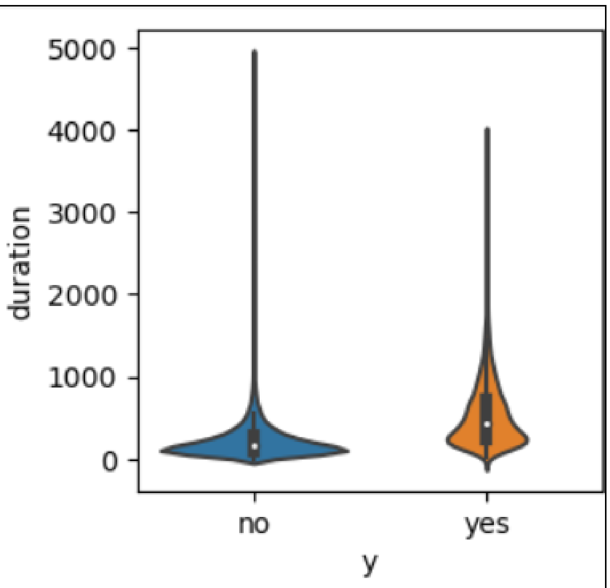
day


30
25
20
15
10
5
0
day
no
yes
y

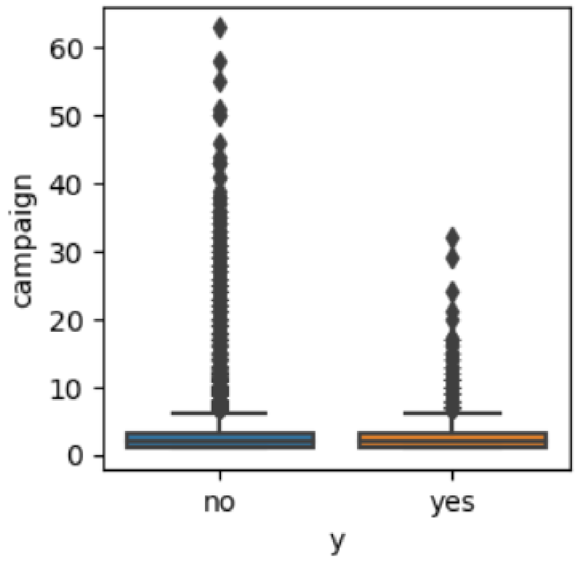

35
30
25
20
15
10
5
0
day
no
yes
y

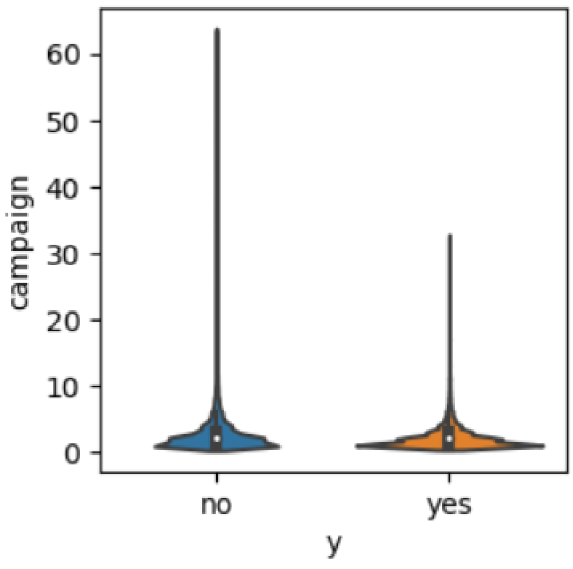
duration



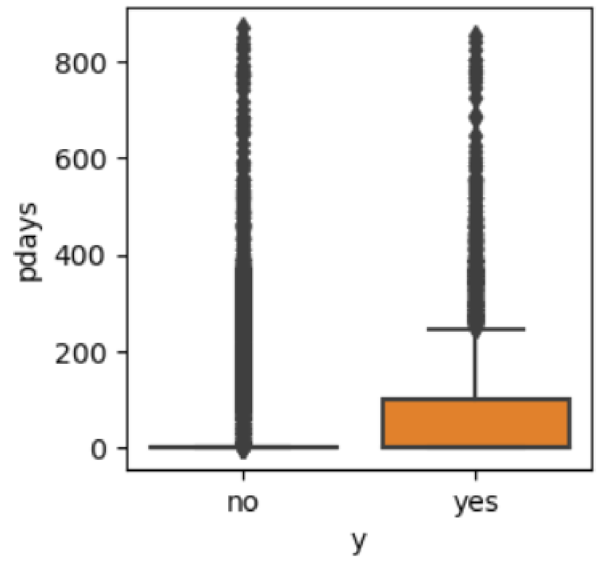


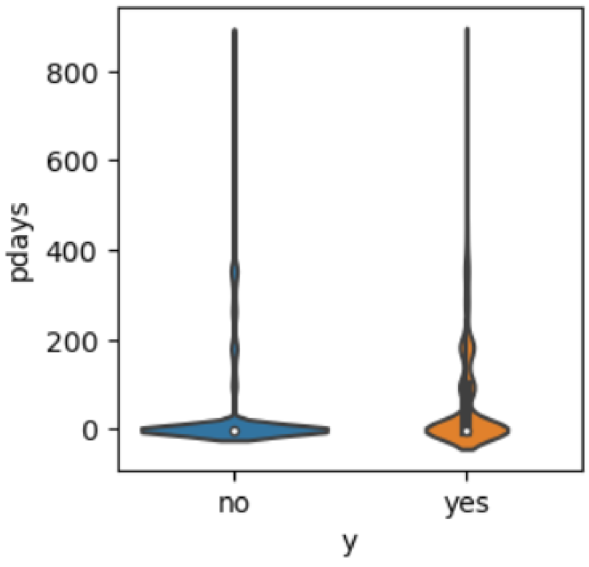
campaign



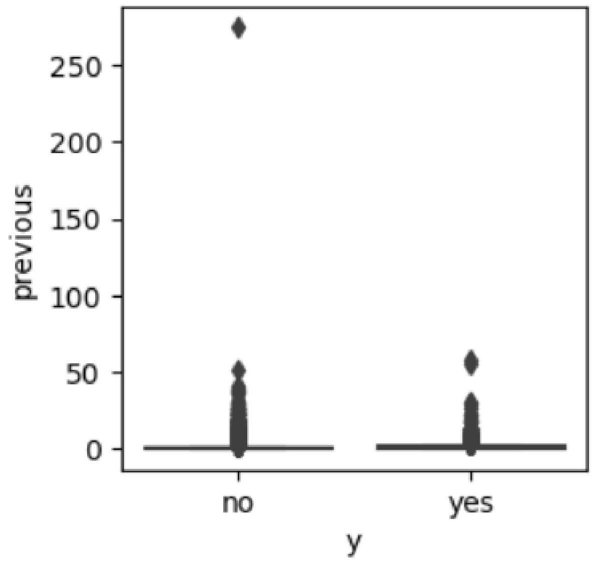


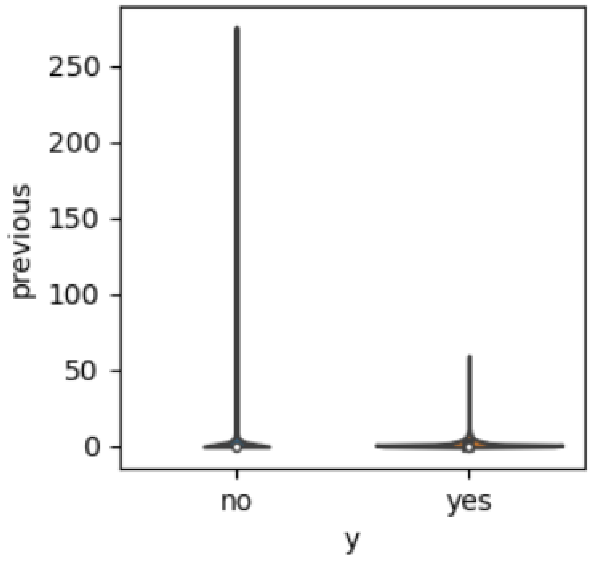
pdays





previous



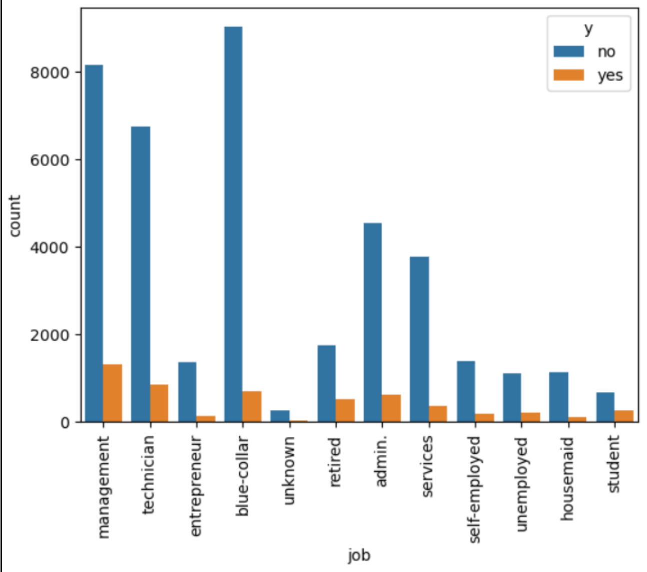


## Now, let us look at how categorical column data is distributed vis-à-vis 'y'

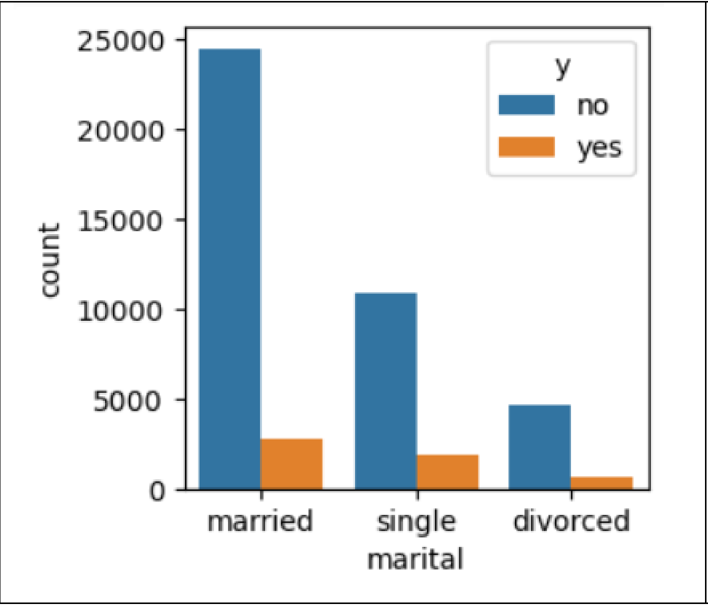
Broadly speaking, the status of Term deposit come under the 'No' -Term deposit segment, and that can be see across all visualisations.

- We have performed crosstab to get the values of every sub category under the category how it is varying with the target variable.
- To visually represent the same, we have used count plot and performed the analysis.

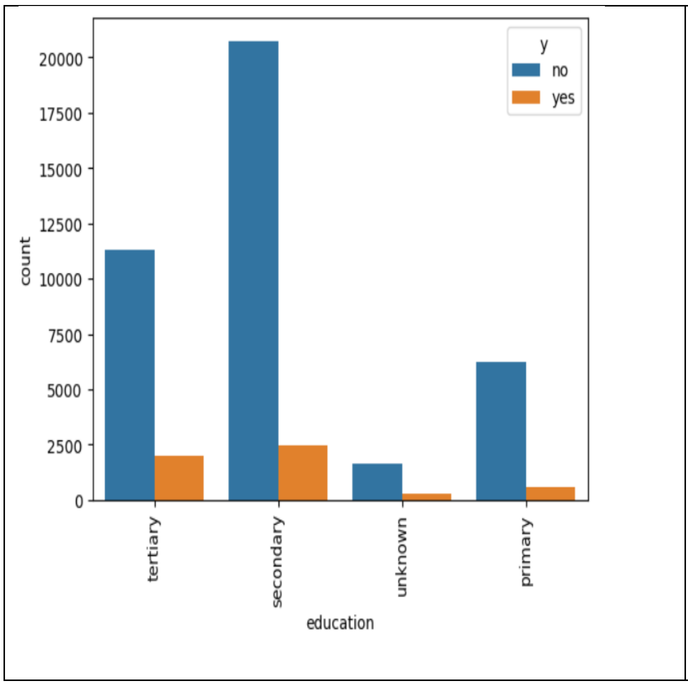
- We can infer from this data that people who have opted Term deposited are close to 10% of the people who have not opted Term Deposit.



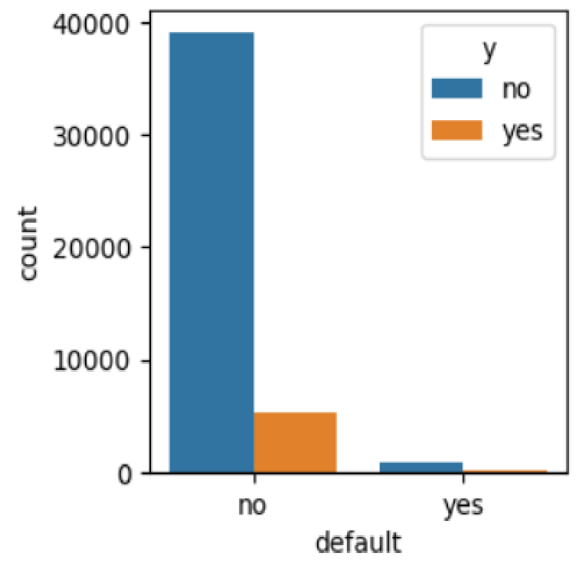
| y | no | yes |
|---|---|---|
| Admin | 4540 | 631 |
| Blue-collar | 9024 | 708 |
| entrepreneur | 1364 | 123 |
| Housemaid | 1131 | 109 |
| management | 8157 | 1301 |
| retired | 1748 | 516 |
| Self- employed | 1392 | 187 |
| services | 3785 | 369 |
| student | 669 | 269 |
| technician | 6757 | 840 |
| Unemployed | 1101 | 202 |
| unknown | 254 | 34 |



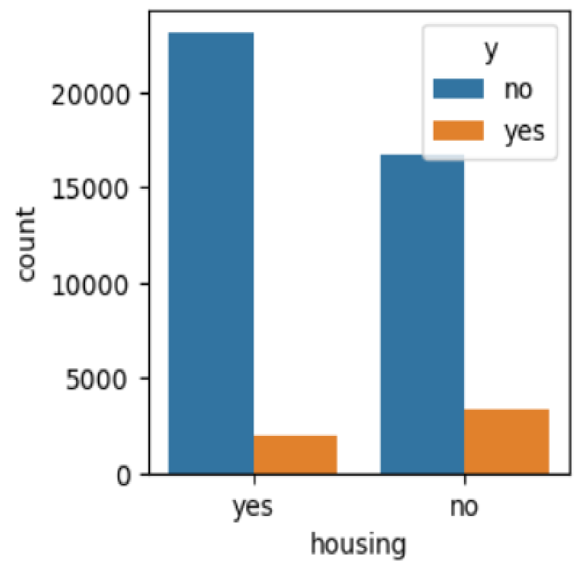
| y | no | Yes |
|---|---|---|
| Divorced | 4585 | 622 |
| Married | 24459 | 2755 |
| single | 10878 | 1912 |



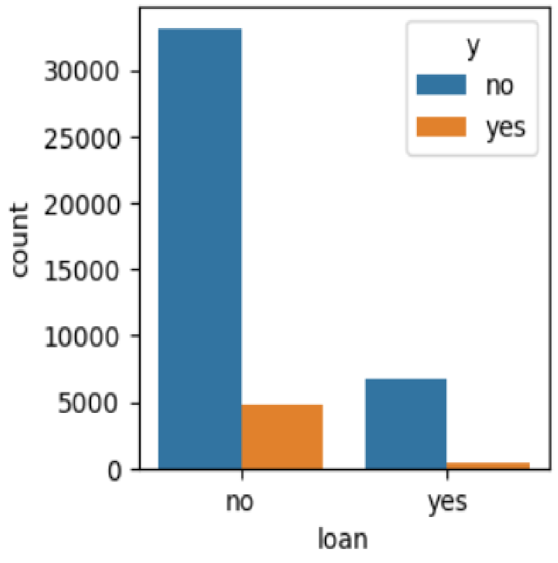
| y | No | yes |
|---|---|---|
| primary | 6260 | 591 |
| Secondary | 20752 | 2450 |
| Tertiary | 11305 | 1996 |
| unknown | 1605 | 252 |



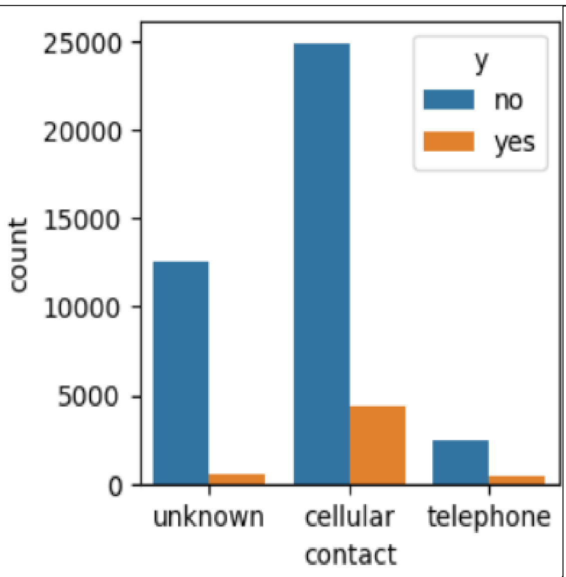
| Y | No | Yes |
|---|---|---|
| No | 39159 | 5237 |
| Yes | 763 | 52 |



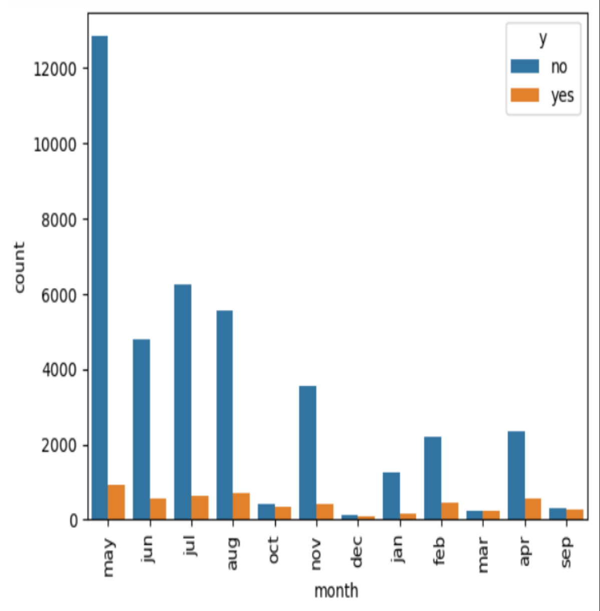
| y | no | yes |
|---|---|---|
| No | 16727 | 3354 |
| yes | 23195 | 1935 |



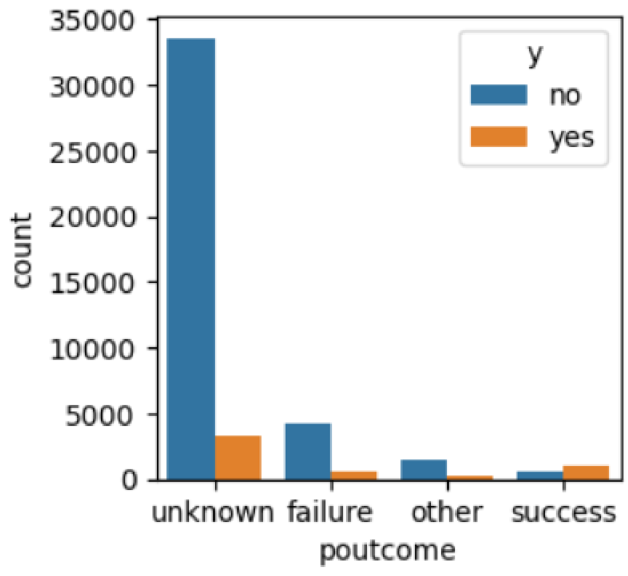
| Y | no | Yes |
|---|---|---|
| No | 33162 | 4805 |
| yes | 6760 | 482 |



| y | No | yes |
|---|---|---|
| Cellular | 24916 | 436<br>9 |
| Telephone | 2516 | 390 |
| unknown | 12490 | 530 |



| y | No | Yes |
|---|---|---|
| Apr | 2355 | 577 |
| Aug | 5559 | 688 |
| Dec | 114 | 100 |
| Feb | 2208 | 441 |
| Jan | 1261 | 142 |
| Jul | 6268 | 627 |
| Jun | 4795 | 546 |
| Mar | 229 | 248 |
| May | 12841 | 925 |
| Nov | 3567 | 403 |
| Oct | 415 | 323 |
| Sep | 310 | 269 |

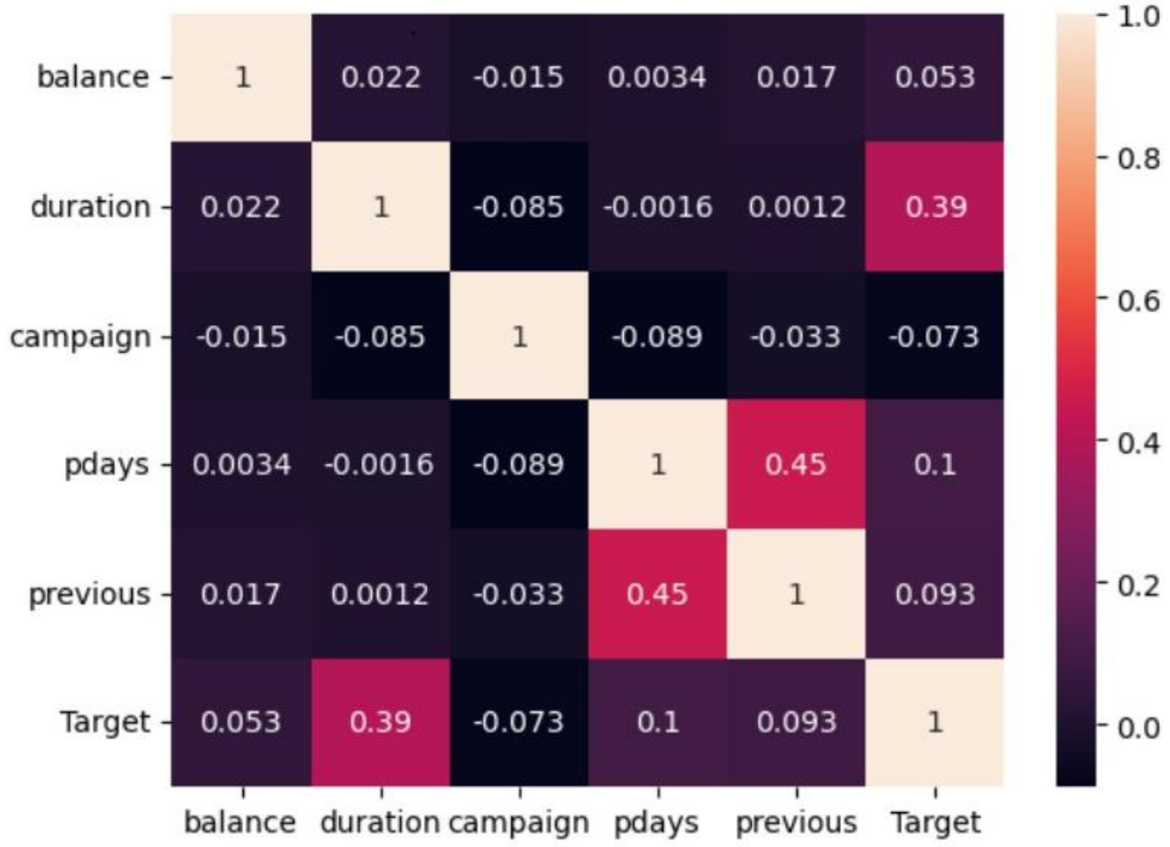


| y | no | yes |
|---|---|---|
| Failure | 4283 | 618 |
| Other | 1533 | 307 |
| Success | 533 | 978 |
| unknow n | 33573 | 338 6 |

## Multivariate Analysis:

In this section we aim to visualise the relationship between our target variable and features more broadly.

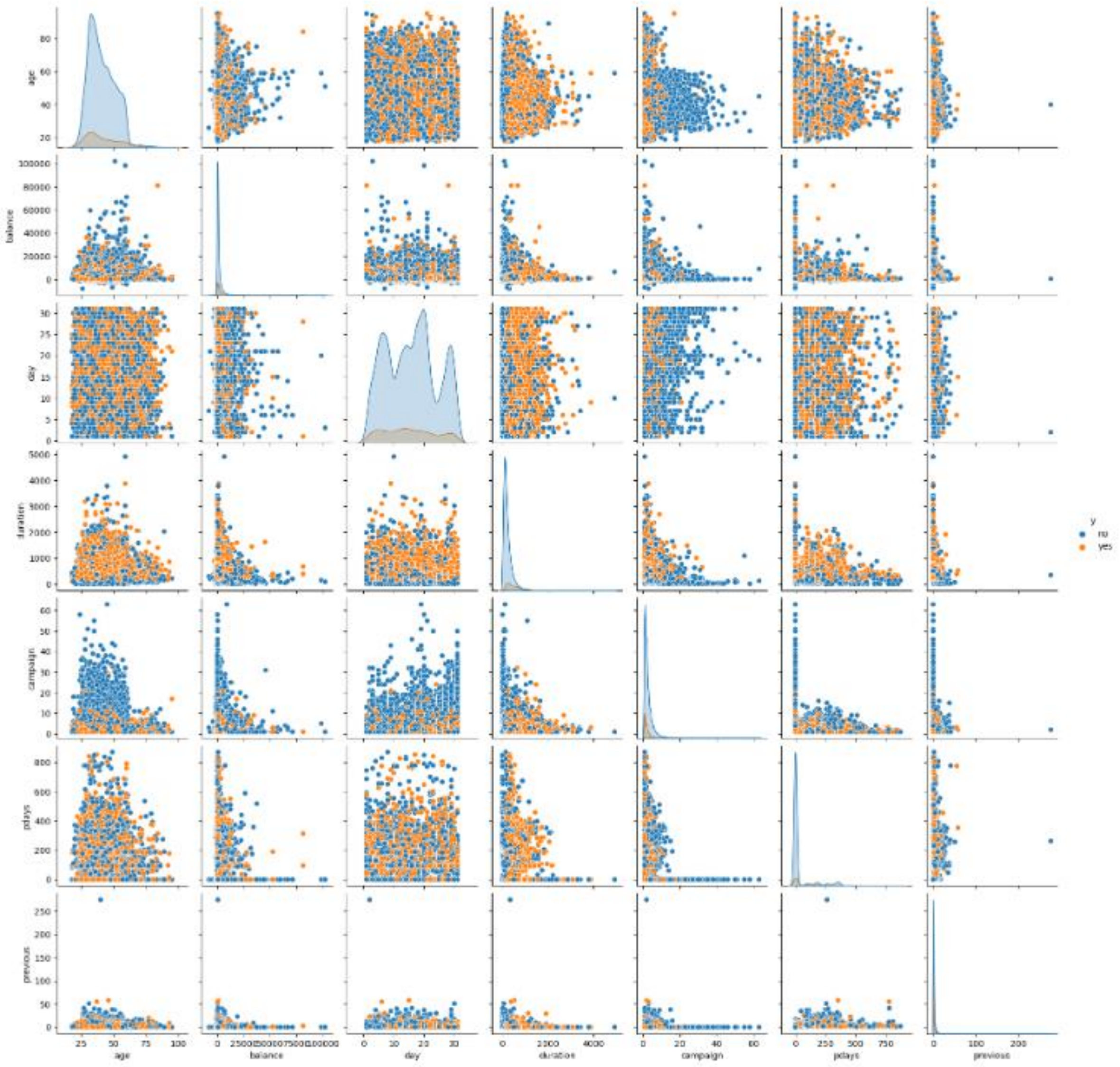
For instance, it might be useful to check how our term deposit is affected with different numerical and categorical variables. In this section we perform heatmap to check the correlation between the numerical variables with our Target variable.



- From our heatmap we can understand that there is high correlation between our numerical variables. We can see that duration is the only variable which is a little correlated with our Target variable.
- There exists very little multicollinearity between the variables pdays and previous, which is very negligible. Rest no other independent variables is exhibiting multicollinearity.

The Inference we get from pair plot is that:

- We can observe correlations between features and the target variable based on patterns and trends in the scatter plots.
- Outliers can be identified as data points deviating significantly from the overall pattern.



## Outlier Treatment:

While we have worked to treat missing values in our dataset, we have taken a different approach when it comes to outliers, given the type of data available and the scope of analysis that we aim to undertake.

We can see that after outlier treatment the number of rows is reduced from 45211 to 28193.

- We can infer that, 'pdays' and 'previous' variables are not showing any significant values.
- So, we cannot do outlier treatment.

**Treatment of Imbalanced Data**

Common consensus suggests that a proportion of at least 30:70 should be present in a binary target variable. However, in case of the ‘Status’ column, the share of values within ‘yes’ are of a lower proportion.

Similar to our decision on outliers, we have made certain considerations for our target variable, and not attempted to treat imbalanced immediately at the outset.

We have decided to use the data in its original proportion for our base model, and based on its performance will make decisions on whether and how to adjust the imbalance at a later stage.

## Model Building and Additional Data Treatments:

**Classification Focussed Predictive Modelling**

As originally mentioned in our objective, our aim with this analysis is to try and understand trends in order rejections. Accordingly, we will build a model that will help identify which of the features are most likely to affect rejections and consequently be useful in predicting rejections in the future.

As a start, we will build a logistic regression based classifier. We shall use popular metrics such as precision and f1-score among other, to interpret the performance of our base model.

For our logistic model, we shall start with reclassifying our target variable 'Status' with values 0 (for 'Not Rejected') and 1(for 'Rejected').

```
df['Status'] = df['Status'].apply(lambda x: 1 if x=='Rejected' else 0)
```

```
df1 = df.copy()
```

```
df1.set_index('Order ID', inplace=True)
```

```
df_target=df1['Status']
df_feature=df1.drop('Status',axis=1)
```

### Feature encoding and scaling:

We shall scale the numeric features of our dataset, and encode the categorical features. Numeric features are scaled to bring them into the same base level; and categorical features are encoded so that their numeric representations can be used to perform further analysis and be fed into out machine learning model.

- Scaling the numerical columns using the Standard Scaler function

```
X_norm = StandardScaler()
num_norm = X_norm.fit_transform(df_num)

df_num_scaled = pd.DataFrame(num_norm, columns = df_num.columns)

df_num_scaled = df_num_scaled.set_index(df['Order ID'])
```

- Encoding the categorical columns using Ordinal Encoding

```
df_cat1 = pd.get_dummies(df_cat[['Fulfilment','Sales Channel','Ship Service Level','Category','Courier Status','Region','B2B']]
```

```
df1['Size'].unique()
```

```
array(['S', '3XL', 'XL', 'L', 'XXL', 'XS', '6XL', 'M', '4XL', '5XL',
       'Free'], dtype=object)
```

```
oe = OrdinalEncoder(categories=[['XS','S','M','L','XL','XXL','3XL','4XL','5XL','6XL', 'Free']])
size = oe.fit_transform(df_cat[['Size']])
df_cat2 = pd.DataFrame(size)
```

```
df_cat2.rename(columns = {0:'Size'}, inplace=True)
```

- Concatenating the scaled numeric and encoded categorical columns to reform our dataset

```
df2 = pd.concat([df_num, df_cat1], axis=1)
```

```
df_cat2 = df_cat2.set_index(df['Order ID'])
```

```
final = pd.concat([df2, df_cat2,df_target], axis=1)
```

```
final.columns
```

```
Index(['Qty', 'Amount', 'Fulfilment_Merchant', 'Sales Channel_Non-Amazon',
       'Ship Service Level_Standard', 'Category_Bottom', 'Category_Dupatta',
       'Category_Ethnic Dress', 'Category_Saree', 'Category_Set',
       'Category_Top', 'Category_Western Dress', 'Category_kurta',
       'Courier Status_Shipped', 'Courier Status_Unshipped',
       'Region_North India', 'Region_Others', 'Region_South India',
       'Region_West India', 'B2B_Discount_Not_Eligible', 'Size', 'Status'],
      dtype='object')
```

- Separating our data into target and features sections and using the Train_Test_Split function to create appropriate data partitions to be fed into the model

```
X = final.drop('Status', axis=1)
y = final['Status']
```

```
Xc = sm.add_constant(X)
```

```
X_train, X_test, y_train, y_test = train_test_split(Xc, y, test_size=0.25, random_state=42)
```

- Feeding the data in to train our model

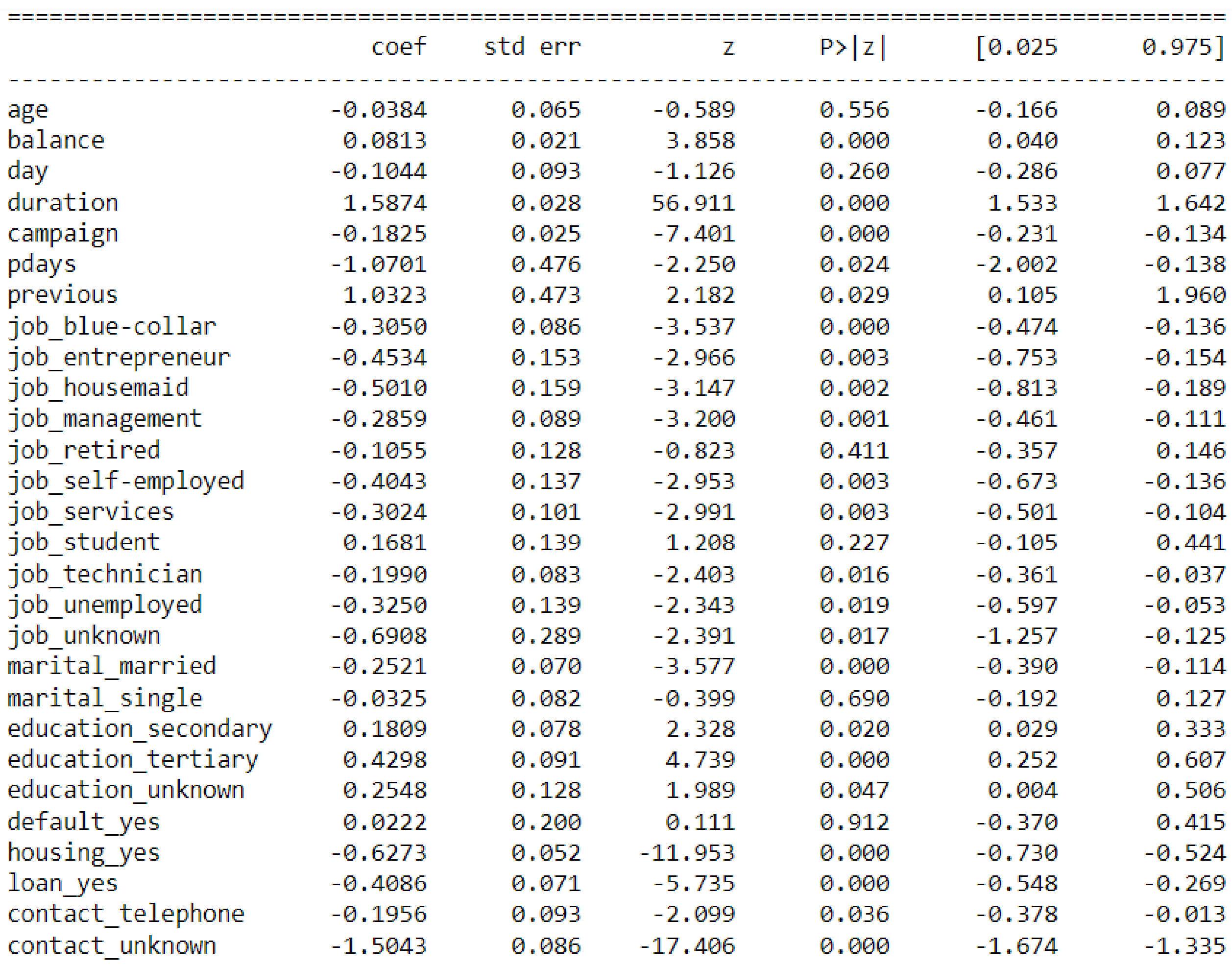
```
model = sm.Logit(y_train,X_train).fit()
```

## Analysis and Scoring:

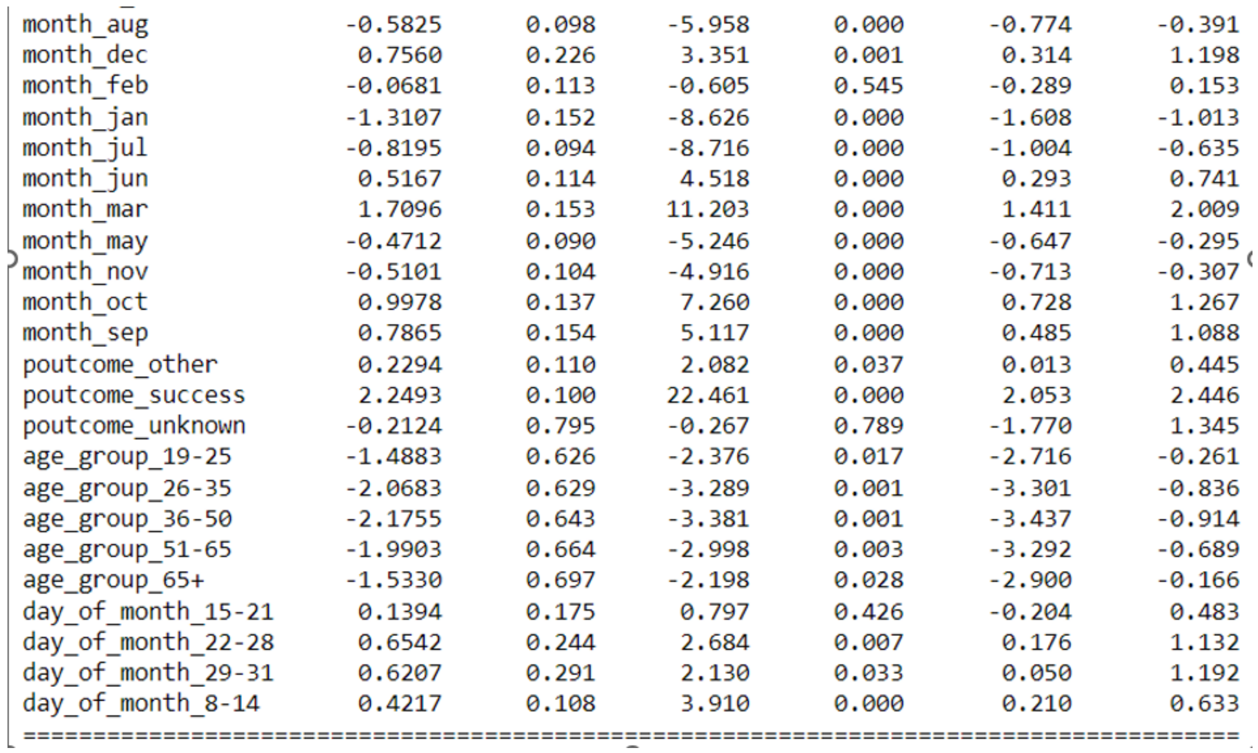
Now that we have split our model in training and testing segments, and trained our model using the training segment, we shall use various methods to check performance.

First, we use the model.summary() function to output a table of various results from our Logistic Regression Classifier.

```
Optimization terminated successfully.
         Current function value: 0.227387
         Iterations 8
                           Logit Regression Results
==============================================================================
Dep. Variable:                 Target   No. Observations:                31647
Model:                          Logit   Df Residuals:                    31596
Method:                           MLE   Df Model:                           50
Date:                Wed, 21 Jun 2023   Pseudo R-squ.:                  0.3731
Time:                        10:34:36   Log-Likelihood:                -7196.1
converged:                       True   LL-Null:                       -11479.
Covariance Type:            nonrobust   LLR p-value:                     0.000
================================================================================
```



```
==================================================================================
                         coef    std err          z      P>|z|      [0.025      0.975]
----------------------------------------------------------------------------------
age                   -0.0384      0.065     -0.589      0.556      -0.166       0.089
balance                0.0813      0.021      3.858      0.000       0.040       0.123
day                   -0.1044      0.093     -1.126      0.260      -0.286       0.077
duration               1.5874      0.028     56.911      0.000       1.533       1.642
campaign              -0.1825      0.025     -7.401      0.000      -0.231      -0.134
pdays                 -1.0701      0.476     -2.250      0.024      -2.002      -0.138
previous               1.0323      0.473      2.182      0.029       0.105       1.960
job_blue-collar       -0.3050      0.086     -3.537      0.000      -0.474      -0.136
job_entrepreneur      -0.4534      0.153     -2.966      0.003      -0.753      -0.154
job_housemaid         -0.5010      0.159     -3.147      0.002      -0.813      -0.189
job_management        -0.2859      0.089     -3.200      0.001      -0.461      -0.111
job_retired           -0.1055      0.128     -0.823      0.411      -0.357       0.146
job_self-employed     -0.4043      0.137     -2.953      0.003      -0.673      -0.136
job_services          -0.3024      0.101     -2.991      0.003      -0.501      -0.104
job_student            0.1681      0.139      1.208      0.227      -0.105       0.441
job_technician        -0.1990      0.083     -2.403      0.016      -0.361      -0.037
job_unemployed        -0.3250      0.139     -2.343      0.019      -0.597      -0.053
job_unknown           -0.6908      0.289     -2.391      0.017      -1.257      -0.125
marital_married       -0.2521      0.070     -3.577      0.000      -0.390      -0.114
marital_single        -0.0325      0.082     -0.399      0.690      -0.192       0.127
education_secondary    0.1809      0.078      2.328      0.020       0.029       0.333
education_tertiary     0.4298      0.091      4.739      0.000       0.252       0.607
education_unknown      0.2548      0.128      1.989      0.047       0.004       0.506
default_yes            0.0222      0.200      0.111      0.912      -0.370       0.415
housing_yes           -0.6273      0.052    -11.953      0.000      -0.730      -0.524
loan_yes              -0.4086      0.071     -5.735      0.000      -0.548      -0.269
contact_telephone     -0.1956      0.093     -2.099      0.036      -0.378      -0.013
contact_unknown       -1.5043      0.086    -17.406      0.000      -1.674      -1.335
```



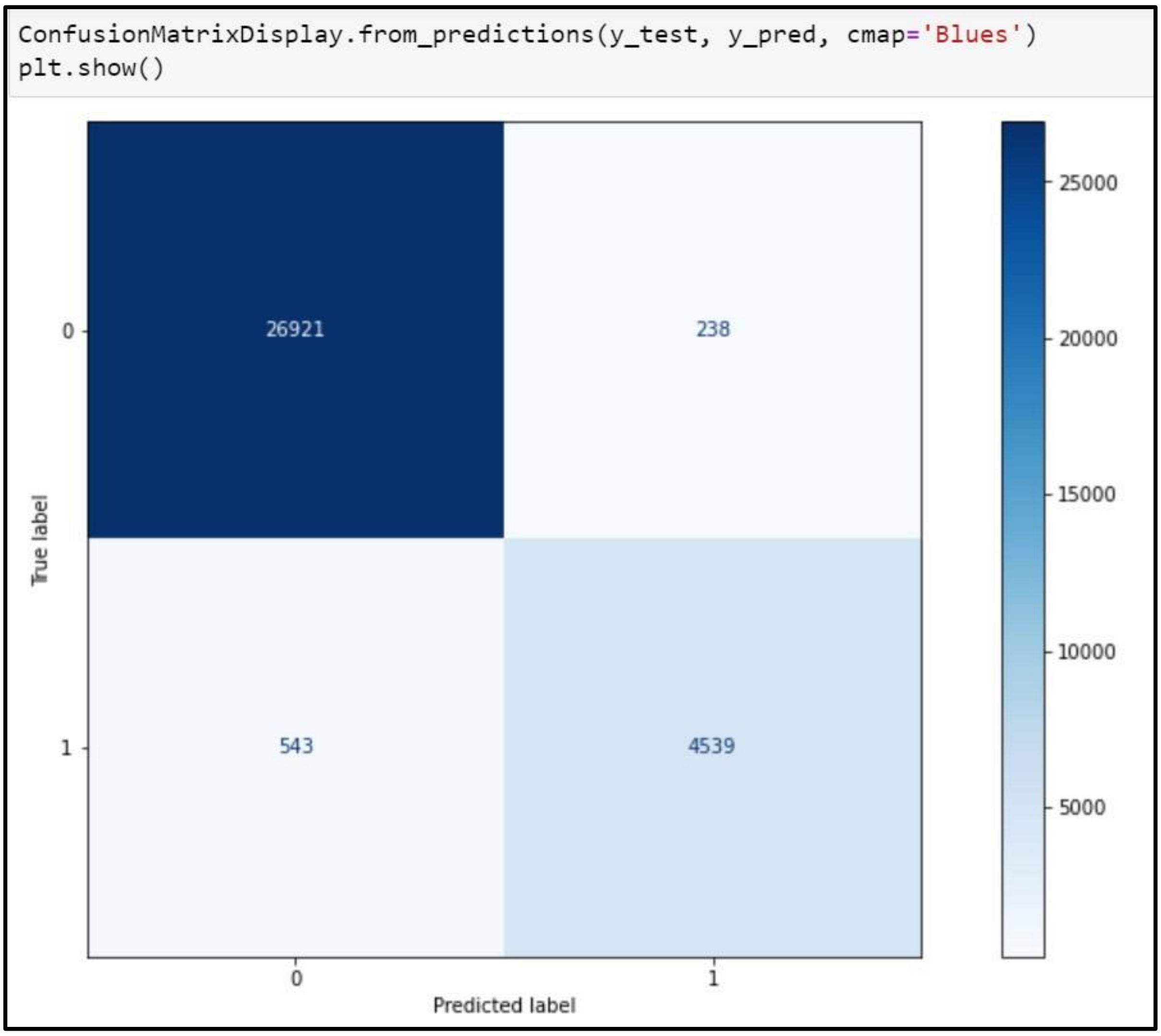
```
month_aug             -0.5825      0.098     -5.958      0.000      -0.774      -0.391
month_dec              0.7560      0.226      3.351      0.001       0.314       1.198
month_feb             -0.0681      0.113     -0.605      0.545      -0.289       0.153
month_jan             -1.3107      0.152     -8.626      0.000      -1.608      -1.013
month_jul             -0.8195      0.094     -8.716      0.000      -1.004      -0.635
month_jun              0.5167      0.114      4.518      0.000       0.293       0.741
month_mar              1.7096      0.153     11.203      0.000       1.411       2.009
month_may             -0.4712      0.090     -5.246      0.000      -0.647      -0.295
month_nov             -0.5101      0.104     -4.916      0.000      -0.713      -0.307
month_oct              0.9978      0.137      7.260      0.000       0.728       1.267
month_sep              0.7865      0.154      5.117      0.000       0.485       1.088
poutcome_other         0.2294      0.110      2.082      0.037       0.013       0.445
poutcome_success       2.2493      0.100     22.461      0.000       2.053       2.446
poutcome_unknown      -0.2124      0.795     -0.267      0.789      -1.770       1.345
age_group_19-25       -1.4883      0.626     -2.376      0.017      -2.716      -0.261
age_group_26-35       -2.0683      0.629     -3.289      0.001      -3.301      -0.836
age_group_36-50       -2.1755      0.643     -3.381      0.001      -3.437      -0.914
age_group_51-65       -1.9903      0.664     -2.998      0.003      -3.292      -0.689
age_group_65+         -1.5330      0.697     -2.198      0.028      -2.900      -0.166
day_of_month_15-21     0.1394      0.175      0.797      0.426      -0.204       0.483
day_of_month_22-28     0.6542      0.244      2.684      0.007       0.176       1.132
day_of_month_29-31     0.6207      0.291      2.130      0.033       0.050       1.192
day_of_month_8-14      0.4217      0.108      3.910      0.000       0.210       0.633
==================================================================================
```

Next, given the classification nature of this model, we shall use a standard cut-off of 0.5 to put predictions into the ‘Rejected’/’Not Rejected’ buckets

```
y_pred_prob = model.predict(X_test)
y_pred = [ 0 if x < 0.5 else 1 for x in y_pred_prob]
```

```
y_pred_prob_train = model.predict(X_train)
y_pred_train = [ 0 if x < 0.5 else 1 for x in y_pred_prob_train]
```

As part of performance scoring, let us look at the confusion matrix, classification report and roc_auc score.



```
ConfusionMatrixDisplay.from_predictions(y_test, y_pred, cmap='Blues')
plt.show()
```

Below we can see that class wise and overall accuracy, precision and f1-score of the model on the training and testing set is broadly similar. However the model's ability to accurately classify class 1 ('Rejected') values seems weaker.

Our base model's relative weakness in accurately predicting when order rejections might occur are a key reason for considering improvement techniques and alternative classifiers.

```
class_report = classification_report(y_test, y_pred)
print(class_report)
```

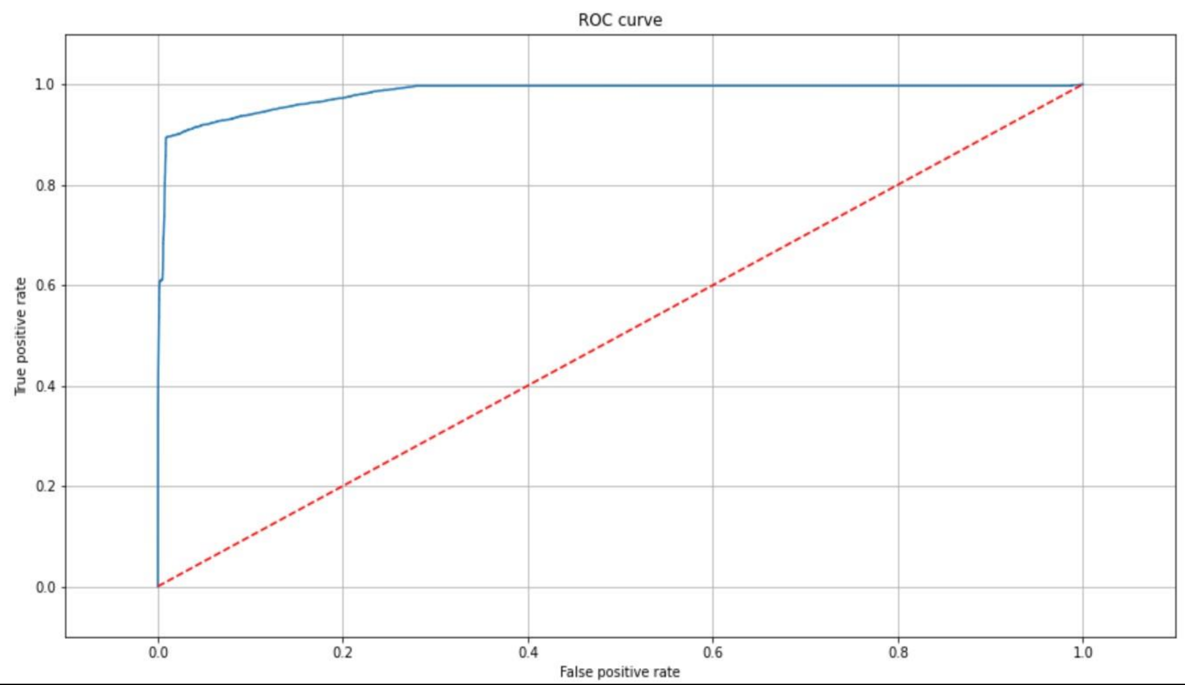
```
              precision    recall  f1-score   support

           0       0.98      0.99      0.99     27159
           1       0.95      0.89      0.92      5082

    accuracy                           0.98     32241
   macro avg       0.97      0.94      0.95     32241
weighted avg       0.98      0.98      0.98     32241
```

```
class_report_train = classification_report(y_train, y_pred_train)
print(class_report_train)
```

```
              precision    recall  f1-score   support

           0       0.98      0.99      0.99     81361
           1       0.95      0.90      0.92     15359

    accuracy                           0.98     96720
   macro avg       0.96      0.94      0.95     96720
weighted avg       0.98      0.98      0.98     96720
```

Finally, our model receives an AUC score of .9820. Given the range of AUC from 0 to 1, on this metric, we can conclude that our model has good performance.

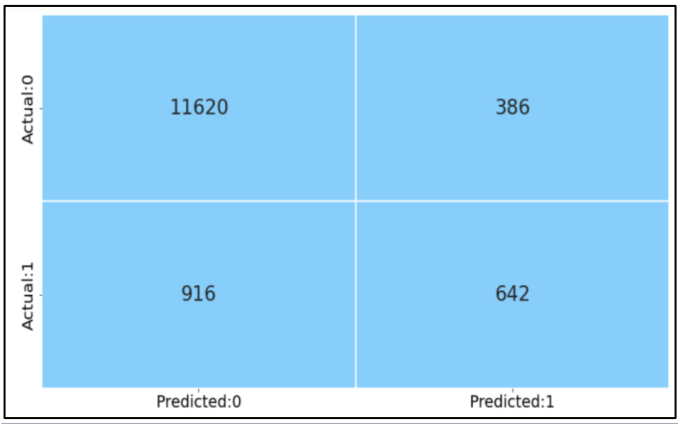
```
AUC: 0.9820507540976898
```



## Random Forest

Random forest is a commonly-used machine learning algorithm, which combines the output of multiple decision trees to reach a single result. Its ease of use and flexibility have fuelled its adoption, as it handles both classification and regression problems.

## Confusion Matrix



As part of performance scoring, let us look at the confusion matrix, classification report and roc_auc score.

**We create generalised function to calculate the metrics of the train and test set.**

## Training data

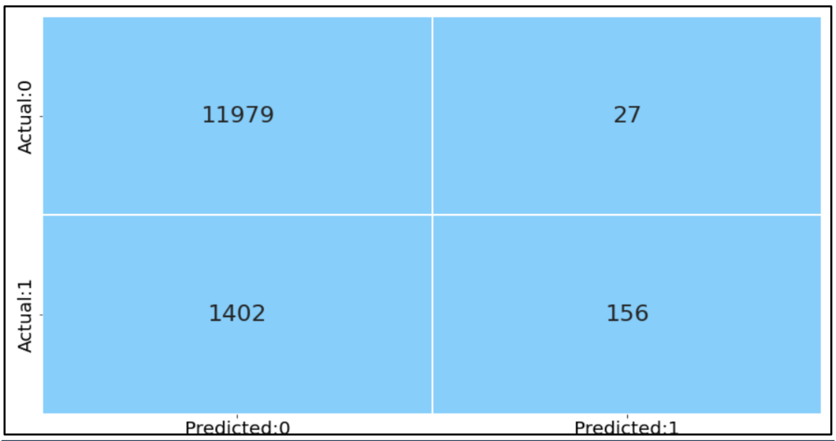
```
              precision    recall  f1-score   support

           0       1.00      1.00      1.00     27916
           1       1.00      0.98      0.99      3731

    accuracy                           1.00     31647
   macro avg       1.00      0.99      0.99     31647
weighted avg       1.00      1.00      1.00     31647
```

## Test data

```
              precision    recall  f1-score   support

           0       0.93      0.97      0.95     12006
           1       0.62      0.41      0.50      1558

    accuracy                           0.90     13564
   macro avg       0.78      0.69      0.72     13564
weighted avg       0.89      0.90      0.90     13564
```

We can see that, the accuracy of train and test data are different, hence we need to tune the model and then see the accuracy.

**Tune the Hyperparameter using Grid Search CV [Random Forest]**



As part of performance scoring, let us look at the confusion matrix, classification report and roc_auc score.

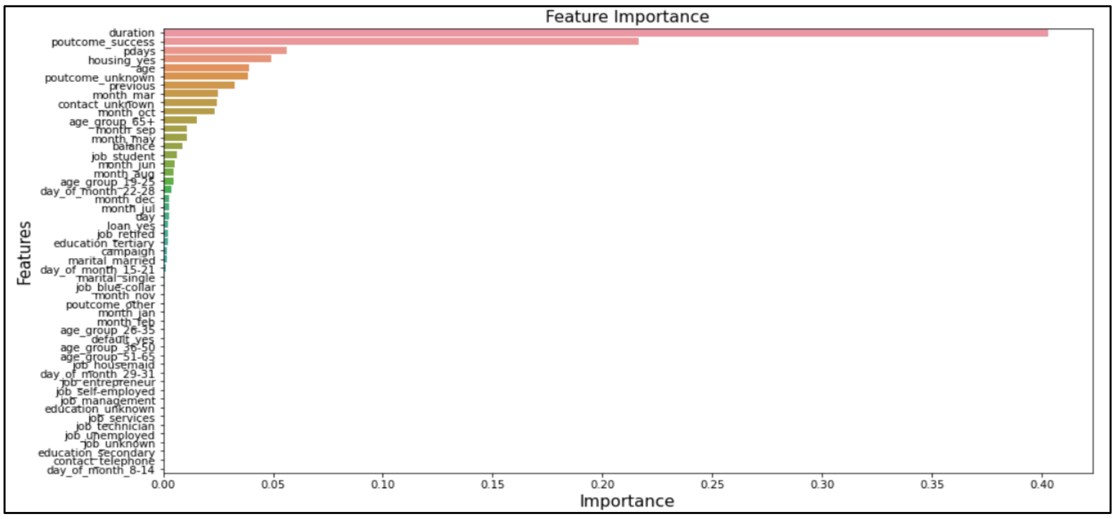
```
rf_model.score(x_train,y_train)
```

```
0.8910165260530224
```

```
rf_model.score(x_test,y_test)
```

```
0.8946475965791801
```

We can see that after performing Tuning accuracy of both train and test dataset are almost equal, hence we can proceed with the model for prediction.
Feature Importance graph represents those features on which the dependent variable is dependent the highest.

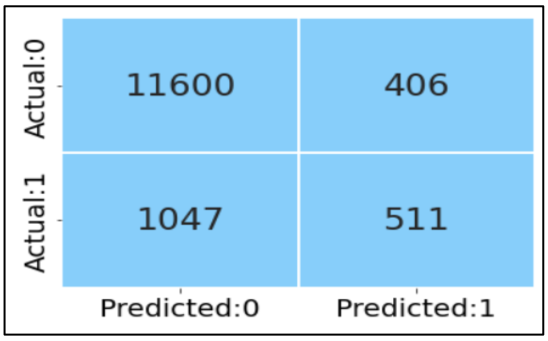


We can see from our outcome that for this prediction model Duration, Poutcome_success, Pdays are the most important features, on which the dependent variable is relying.

## KNN

The k-nearest neighbours algorithm, also known as KNN or k-NN, is a non-parametric, supervised learning classifier, which uses proximity to make classifications or predictions about the grouping of an individual data point.

## Confusion Matrix

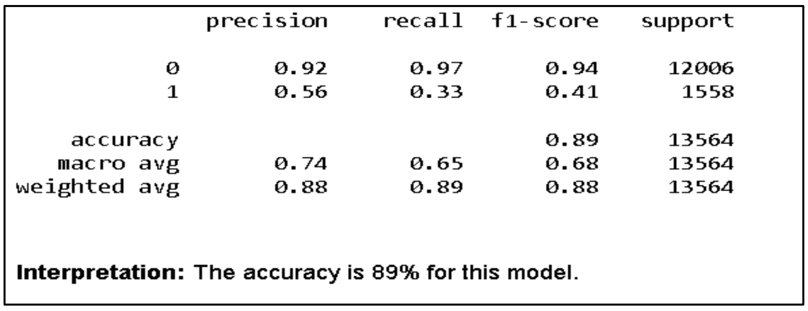


| | Predicted:0 | Predicted:1 |
|---|---|---|
| Actual:0 | 11600 | 406 |
| Actual:1 | 1047 | 511 |

As part of performance scoring, let us look at the confusion matrix, classification report and roc_auc score.

**We create generalised function to calculate the metrics of the train and test set.**

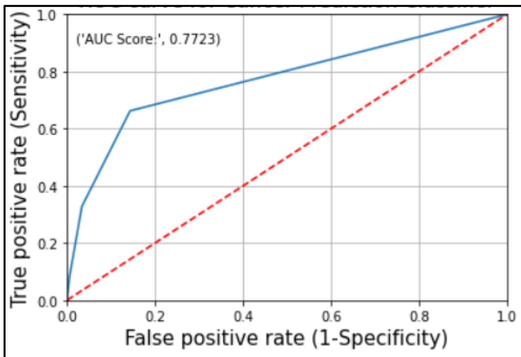
**Train data**



```
              precision    recall  f1-score   support

           0       0.92      0.97      0.94     12006
           1       0.56      0.33      0.41      1558

    accuracy                           0.89     13564
   macro avg       0.74      0.65      0.68     13564
weighted avg       0.88      0.89      0.88     13564
```

**Interpretation:** The accuracy is 89% for this model.

**Test data**

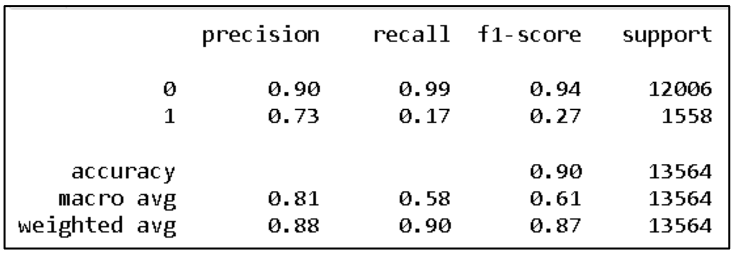


**Interpretation:** The red dotted line represents the ROC curve of a purely random classifier; a good classifier stays as far away from that line as possible (toward the top-left corner).

We can see that, the accuracy of train and test data are different, hence we need to tune the model and then see the accuracy.

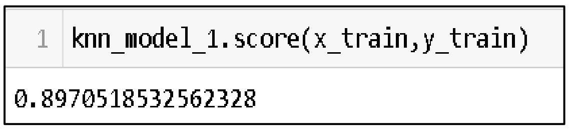
## Optimum value of K using Grid Search

## Train data



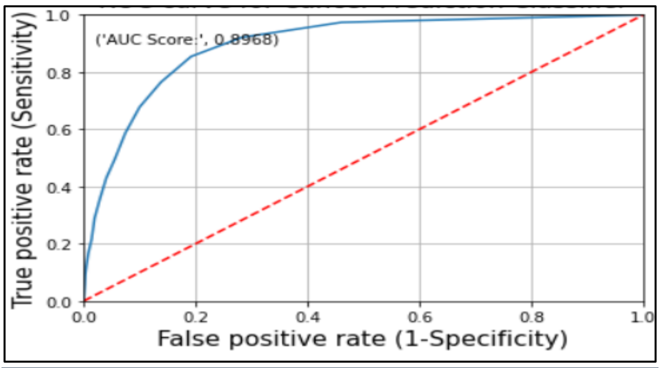
```
              precision    recall  f1-score   support

           0       0.90      0.99      0.94     12006
           1       0.73      0.17      0.27      1558

    accuracy                           0.90     13564
   macro avg       0.81      0.58      0.61     13564
weighted avg       0.88      0.90      0.87     13564
```

We can see that accuracy of train set is changed to 0.8975.



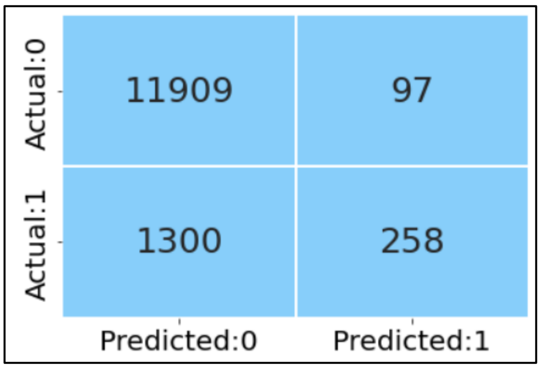
```
knn_model_1.score(x_train,y_train)
```

```
0.8970518532562328
```



## Test data

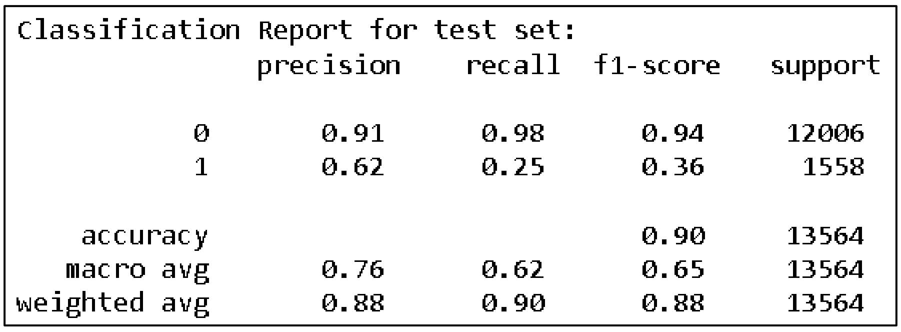
## Confusion Matrix



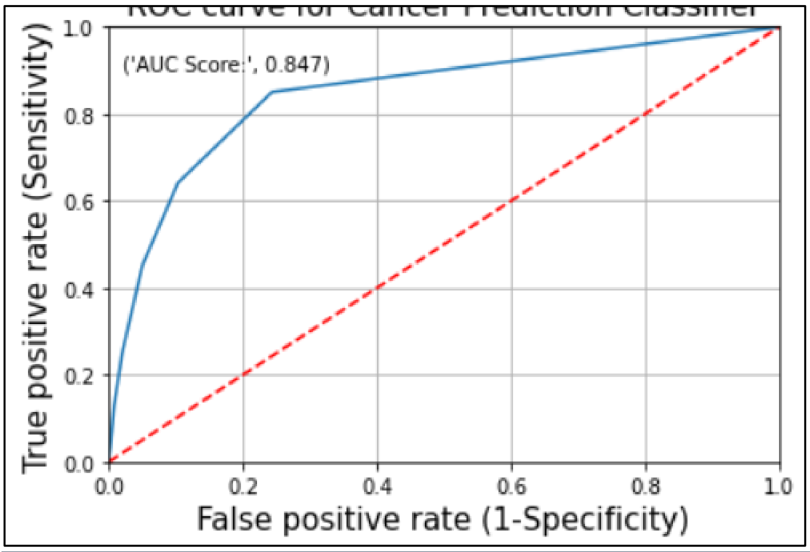
| | Predicted:0 | Predicted:1 |
|---|---|---|
| Actual:0 | 11909 | 97 |
| Actual:1 | 1300 | 258 |

As part of performance scoring, let us look at the confusion matrix, classification report and roc_auc score.

## Tune the Hyperparameter using Grid Search CV [Random Forest]



```
Classification Report for test set:
              precision    recall  f1-score   support

           0       0.91      0.98      0.94     12006
           1       0.62      0.25      0.36      1558

    accuracy                           0.90     13564
   macro avg       0.76      0.62      0.65     13564
weighted avg       0.88      0.90      0.88     13564
```

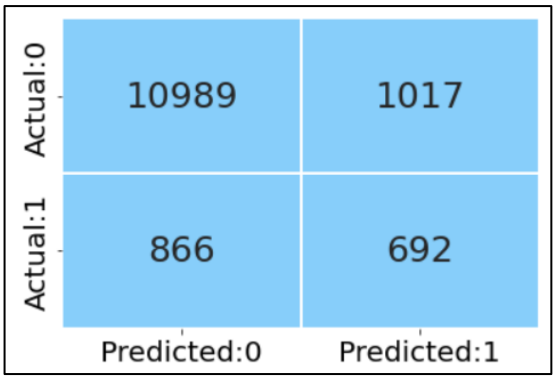


We can see that after performing Tuning accuracy of both train and test dataset are almost equal, hence we can proceed with the model for prediction.

## Naives Bayes

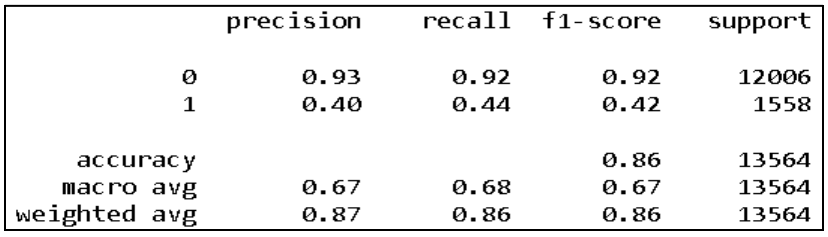
The Naive Bayes Algorithm is one of the crucial algorithms in machine learning that helps with classification problems. It is derived from Bayes' probability theory and is used for text classification, where you train high-dimensional datasets.

## Confusion Matrix



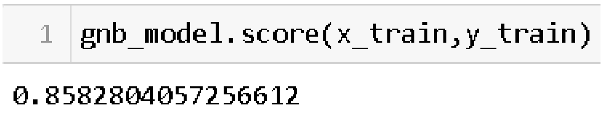
As part of performance scoring, let us look at the confusion matrix, classification report and roc_auc score.

## We create generalised function to calculate the metrics of the train and test set.



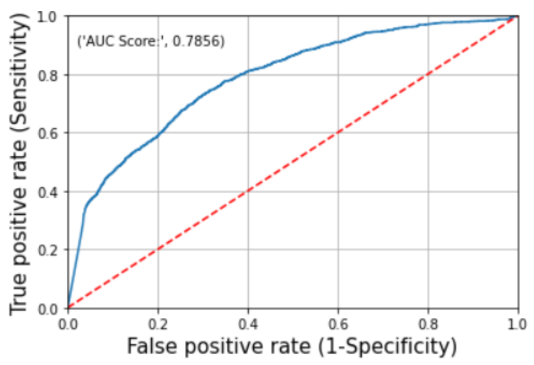
```
              precision    recall  f1-score   support

           0       0.93      0.92      0.92     12006
           1       0.40      0.44      0.42      1558

    accuracy                           0.86     13564
   macro avg       0.67      0.68      0.67     13564
weighted avg       0.87      0.86      0.86     13564
```

We can see that, the accuracy of train and test data are different, hence we need to tune the model and then see the accuracy.



```
gnb_model.score(x_train,y_train)
```

```
0.8582804057256612
```



**Interpretation:** From the above plot, we can see that our classifier (gnb_model) is away from the red dotted line; with the AUC score 0.581.

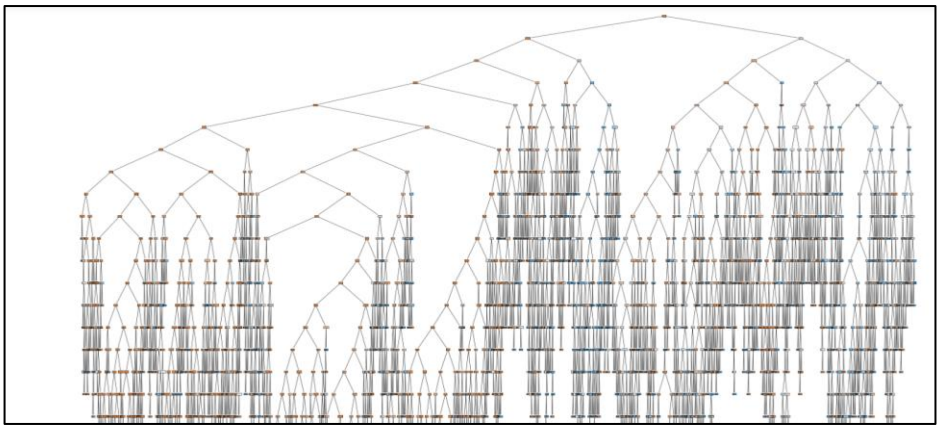
**Note:** Algorithms like Naive Bayes and tree based algorithms do not need feature scaling or normalization. Performing a features scaling in these algorithms may not have much effect.

We can see that after performing Tuning accuracy of both train and test dataset are almost equal, hence we can proceed with the model for prediction.

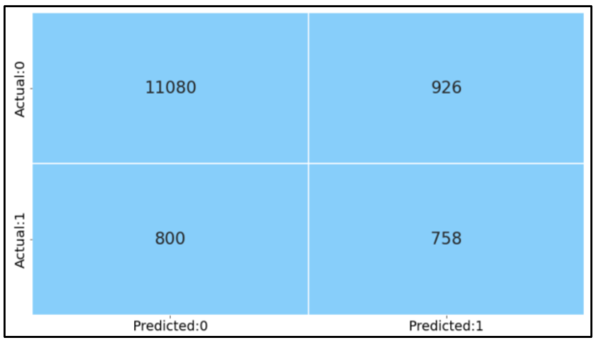
## Decision Tree

A decision tree is a non-parametric supervised learning algorithm, which is utilized for both classification and regression tasks. It has a hierarchical, tree structure, which consists of a root node, branches, internal nodes and leaf nodes.

## Decision Tree model depicted in the below image.



## Confusion Matrix



As part of performance scoring, let us look at the confusion matrix, classification report and roc_auc score.

**We create generalised function to calculate the metrics of the train and test set.**

## Overfitting in Decision Tree

### Train data

```
              precision    recall  f1-score   support

           0       1.00      1.00      1.00     27916
           1       1.00      1.00      1.00      3731

    accuracy                           1.00     31647
   macro avg       1.00      1.00      1.00     31647
weighted avg       1.00      1.00      1.00     31647
```

### Test data

```
              precision    recall  f1-score   support

           0       0.93      0.92      0.93     12006
           1       0.45      0.49      0.47      1558

    accuracy                           0.87     13564
   macro avg       0.69      0.70      0.70     13564
weighted avg       0.88      0.87      0.87     13564
```

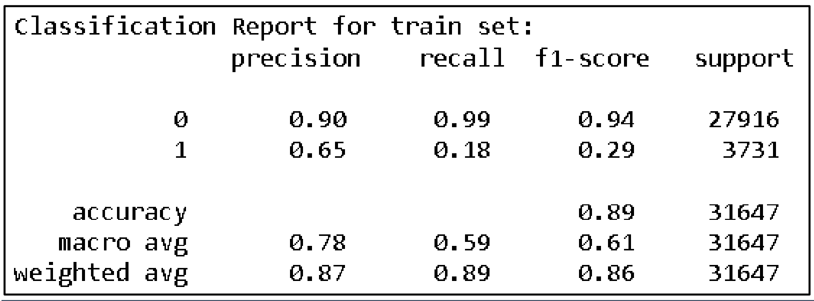
We can see that, the accuracy of train and test data are different, hence we need to tune the model and then see the accuracy.

## Tune parameters Gridsearch CV

```
CPU times: total: 2min 3s
Wall time: 2min 6s
```

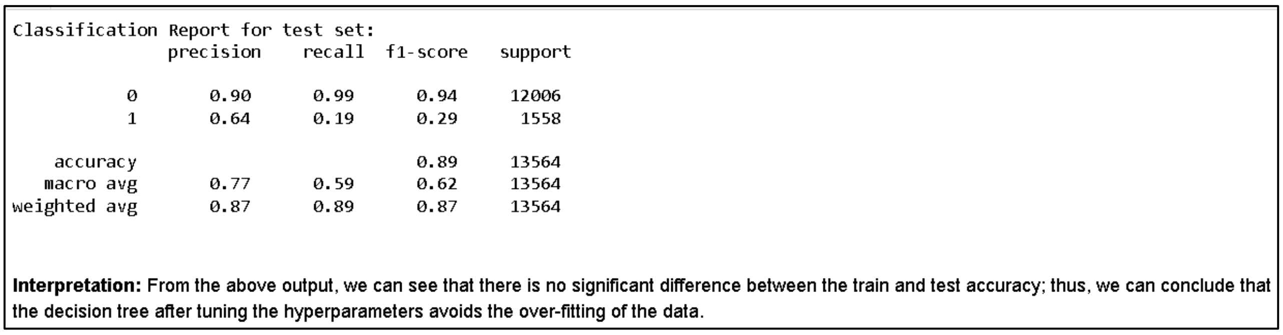
## Train data

## Tune the Hyperparameter using Grid Search CV [Decision Tree]



```
Classification Report for train set:
              precision    recall  f1-score   support

           0       0.90      0.99      0.94     27916
           1       0.65      0.18      0.29      3731

    accuracy                           0.89     31647
   macro avg       0.78      0.59      0.61     31647
weighted avg       0.87      0.89      0.86     31647
```

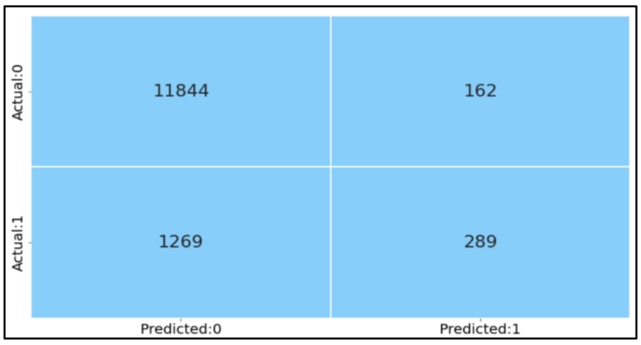
## Test data



```
Classification Report for test set:
              precision    recall  f1-score   support

           0       0.90      0.99      0.94     12006
           1       0.64      0.19      0.29      1558

    accuracy                           0.89     13564
   macro avg       0.77      0.59      0.62     13564
weighted avg       0.87      0.89      0.87     13564
```

**Interpretation:** From the above output, we can see that there is no significant difference between the train and test accuracy; thus, we can conclude that the decision tree after tuning the hyperparameters avoids the over-fitting of the data.

## Confusion Matrix



As part of performance scoring, let us look at the confusion matrix, classification report and roc_auc score.

```
dt_model.score(x_test,y_test)
```

```
0.8945001474491301
```

We can see that after performing Tuning accuracy of both train and test dataset are almost equal, hence we can proceed with the model for prediction.

## Boosting

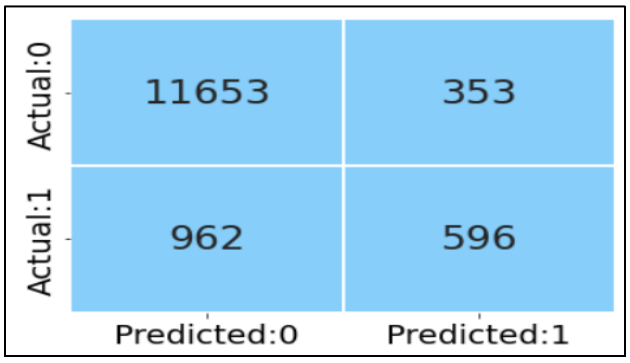
The Ensemble technique considers multiple models for predicting the results. Bagging and Boosting are two of the types of ensembles. The bagging methods construct the multiple models in parallel; whereas, the boosting methods construct the models sequentially.

Earlier, we have studied one of the bagging (bootstrap aggregating) technique i.e. Random Forest. The boosting method fits multiple weak classifiers to create a strong classifier. In this method, the model tries to correct the errors in the previous model. In this section, we learn some of the boosting methods such as AdaBoost, Gradient Boosting and XGBoost.

## ADABOOST

Let us build the AdaBoost classifier with decision trees. The model creates several stumps (decision tree with only a single decision node and two leaf nodes) on the train set and predicts the class based on these weak learners (stumps). For the first model, it assigns equal weights to each sample. It assigns the higher weight for the wrongly predicted samples and lower weight for the correctly predicted samples. This method continues till all the observations are correctly classified or the predefined number of stumps is created.

## Confusion Matrix

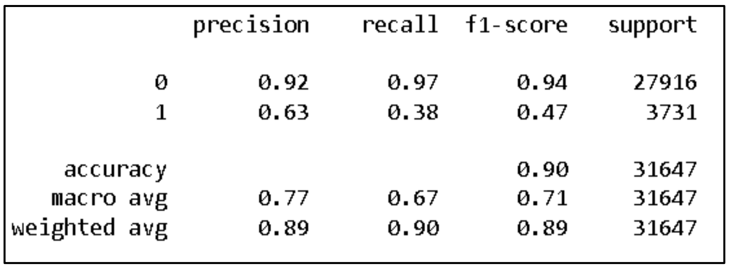


| | Predicted:0 | Predicted:1 |
|---|---|---|
| Actual:0 | 11653 | 353 |
| Actual:1 | 962 | 596 |

As part of performance scoring, let us look at the confusion matrix, classification report and roc_auc score.

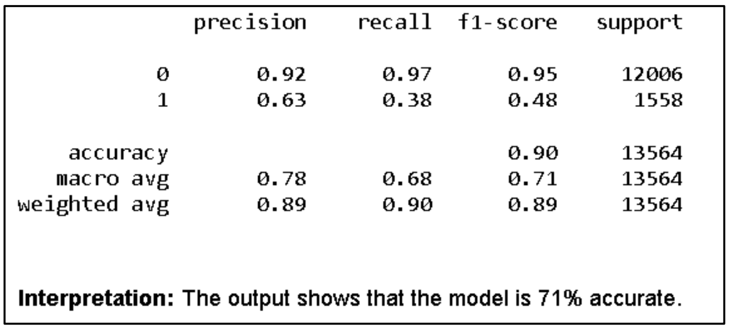
## We create generalised function to calculate the metrics of the train and test set.

### Train data



```
              precision    recall  f1-score   support

           0       0.92      0.97      0.94     27916
           1       0.63      0.38      0.47      3731

    accuracy                           0.90     31647
   macro avg       0.77      0.67      0.71     31647
weighted avg       0.89      0.90      0.89     31647
```

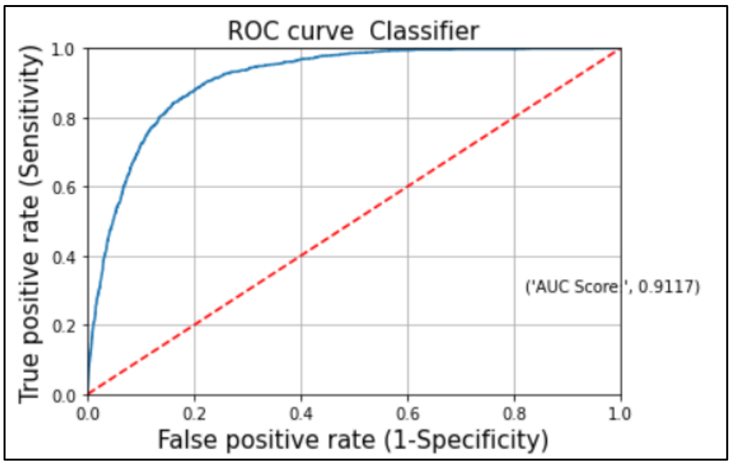
### Test data



```
              precision    recall  f1-score   support

           0       0.92      0.97      0.95     12006
           1       0.63      0.38      0.48      1558

    accuracy                           0.90     13564
   macro avg       0.78      0.68      0.71     13564
weighted avg       0.89      0.90      0.89     13564
```

**Interpretation:** The output shows that the model is 71% accurate.

We can see that, the accuracy of train and test data are different, hence we need to tune the model and then see the accuracy.



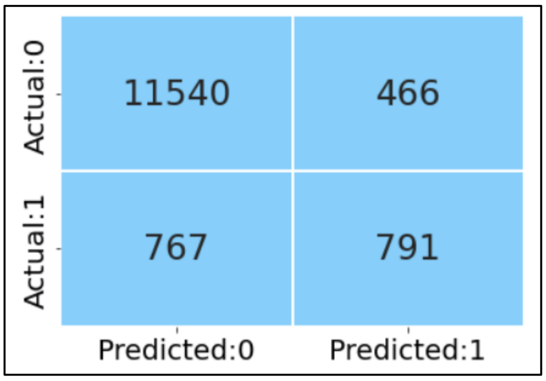
**Interpretation:** The red dotted line represents the ROC curve of a purely random classifier; a good classifier stays as far away from that line as possible (toward the top-left corner).

From the above plot, we can see that the AdaBoost model is away from the dotted line; with the AUC score 0.6532.

## Gradient Boosting

This method optimizes the differentiable loss function by building the number of weak learners (decision trees) sequentially. It considers the residuals from the previous model and fits the next model to the residuals. The algorithm uses a gradient descent method to minimize the error.

## Confusion Matrix

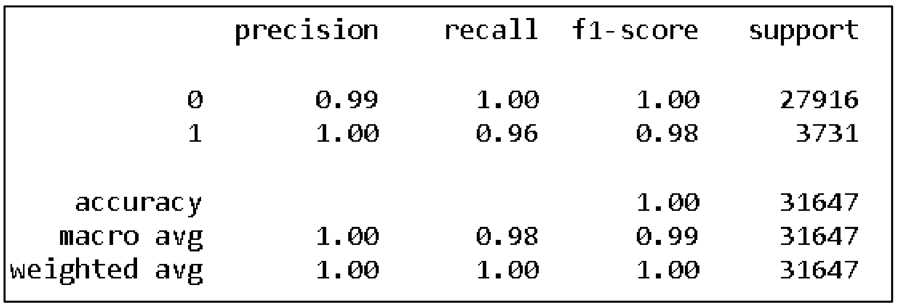


|  | Predicted:0 | Predicted:1 |
|---|---|---|
| Actual:0 | 11540 | 466 |
| Actual:1 | 767 | 791 |

As part of performance scoring, let us look at the confusion matrix, classification report and roc_auc score.

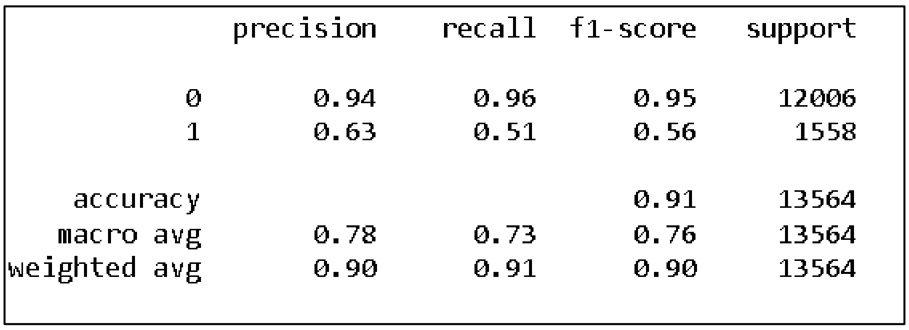
**We create generalised function to calculate the metrics of the train and test set.**

## Train data



```
              precision    recall  f1-score   support

           0       0.99      1.00      1.00     27916
           1       1.00      0.96      0.98      3731

    accuracy                           1.00     31647
   macro avg       1.00      0.98      0.99     31647
weighted avg       1.00      1.00      1.00     31647
```

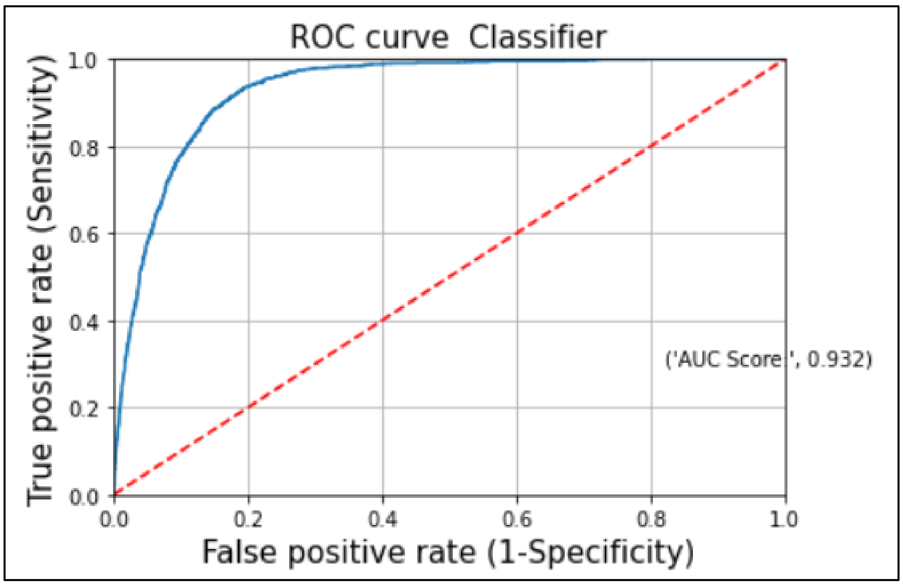
## Test data



```
              precision    recall  f1-score   support

           0       0.94      0.96      0.95     12006
           1       0.63      0.51      0.56      1558

    accuracy                           0.91     13564
   macro avg       0.78      0.73      0.76     13564
weighted avg       0.90      0.91      0.90     13564
```

**Interpretation:** The classification report shows that the model is 71% accurate. Also, the sensitivity and specificity are equal.

We can see that, the accuracy of train and test data are different, hence we need to tune the model and then see the accuracy.



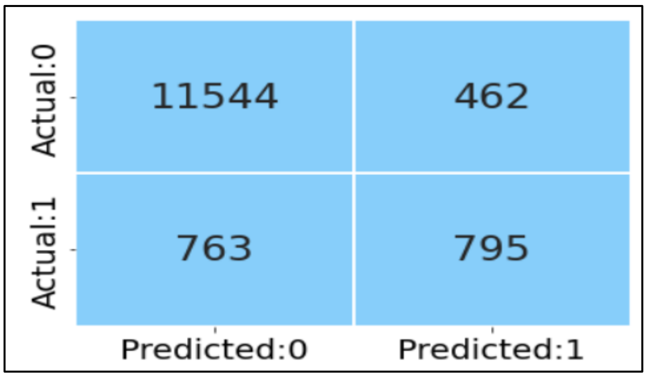
**Interpretation:** The red dotted line represents the ROC curve of a purely random classifier; a good classifier stays as far away from that line as possible (toward the top-left corner).

From the above plot, we can see that the gradient boosting model is away from the dotted line; with the AUC score 0.6554.

## XG boost

XGBoost (extreme gradient boost) is an alternative form of gradient boosting method. This method generally considers the initial prediction as 0.5 and build the decision tree to predict the residuals. It considers the regularization parameter to avoid overfitting.

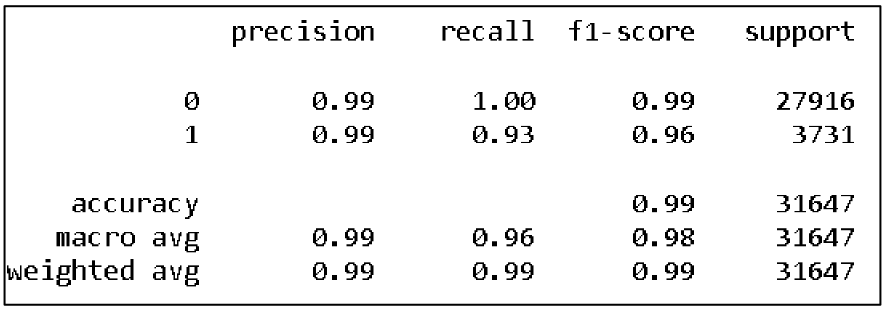
## Confusion Matrix



As part of performance scoring, let us look at the confusion matrix, classification report and roc_auc score.

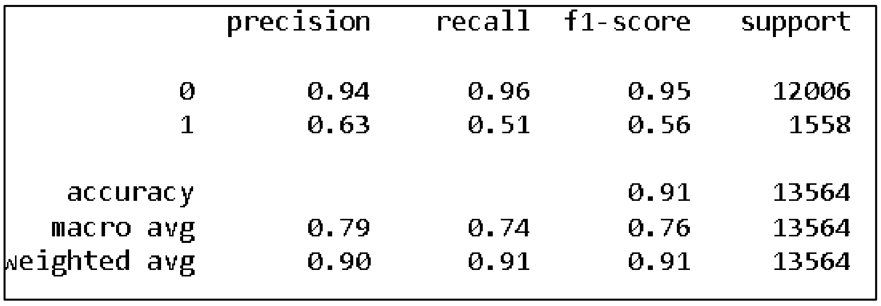
**We create generalised function to calculate the metrics of the train and test set.**

### Train data



```
              precision    recall  f1-score   support

           0       0.99      1.00      0.99     27916
           1       0.99      0.93      0.96      3731

    accuracy                           0.99     31647
   macro avg       0.99      0.96      0.98     31647
weighted avg       0.99      0.99      0.99     31647
```

### Test data



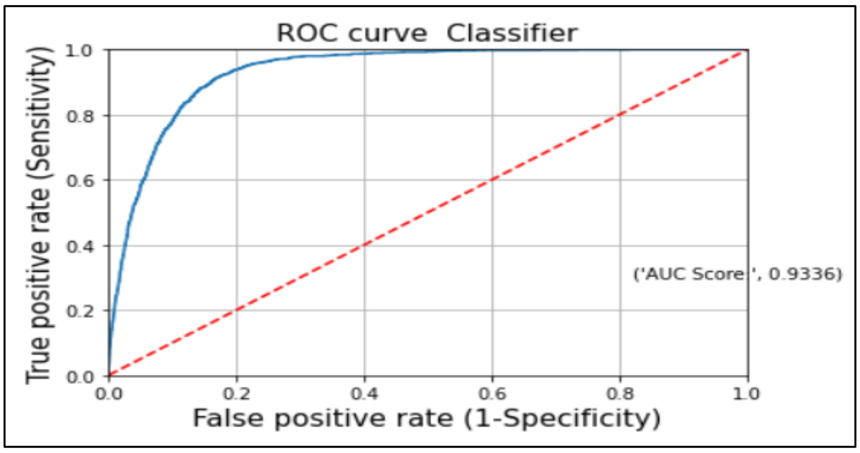
```
              precision    recall  f1-score   support

           0       0.94      0.96      0.95     12006
           1       0.63      0.51      0.56      1558

    accuracy                           0.91     13564
   macro avg       0.79      0.74      0.76     13564
weighted avg       0.90      0.91      0.91     13564
```

We can see that, the accuracy of train and test data are different, hence we need to tune the model and then see the accuracy.

```
xgb_model.score(X_train,y_train)
```

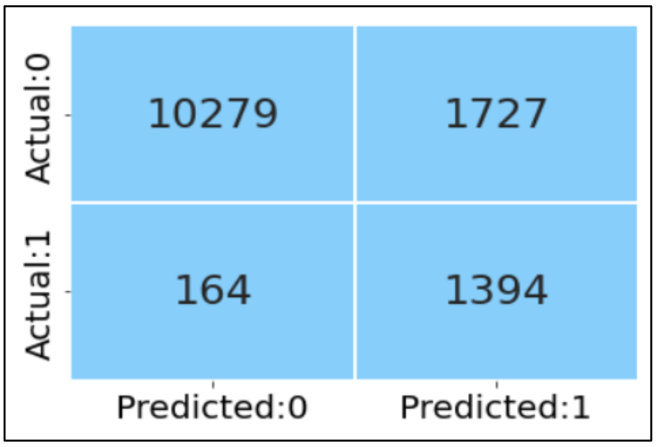
```
0.9908048156223339
```

We can see that after performing Tuning accuracy of both train and test dataset are almost equal, hence we can proceed with the model for prediction.



**Interpretation:** The red dotted line represents the ROC curve of a purely random classifier; a good classifier stays as far away from that line as possible (toward the top-left corner).
From the above plot, we can see that the XGBoost model is away from the dotted line; with the AUC score 0.6461.

**<u>Light Gradient Boost</u>**
**<u>Confusion Matrix</u>**



| | Predicted:0 | Predicted:1 |
|---|---|---|
| Actual:0 | 10279 | 1727 |
| Actual:1 | 164 | 1394 |

As part of performance scoring, let us look at the confusion matrix, classification report and roc_auc score.

```
lgbm_model.score(X_train,y_train)
```

```
0.8712674187126742
```

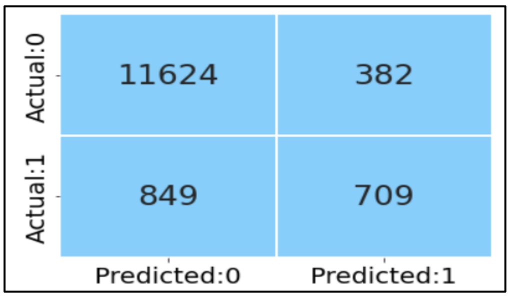
```
lgbm_model.score(X_test,y_test)
```

```
0.8605868475375995
```

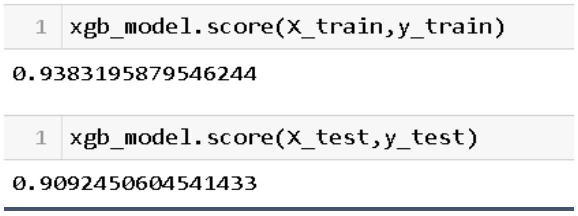
## Tune the Hyperparameter using Grid Search CV [Boost]

```
Classification Report for test set:
              precision    recall  f1-score   support

           0       0.93      0.97      0.95     12006
           1       0.65      0.46      0.54      1558

    accuracy                           0.91     13564
   macro avg       0.79      0.71      0.74     13564
weighted avg       0.90      0.91      0.90     13564
```

## Confusion Matrix



As part of performance scoring, let us look at the confusion matrix, classification report and roc_auc score.



```
xgb_model.score(X_train,y_train)
```

```
0.9383195879546244
```

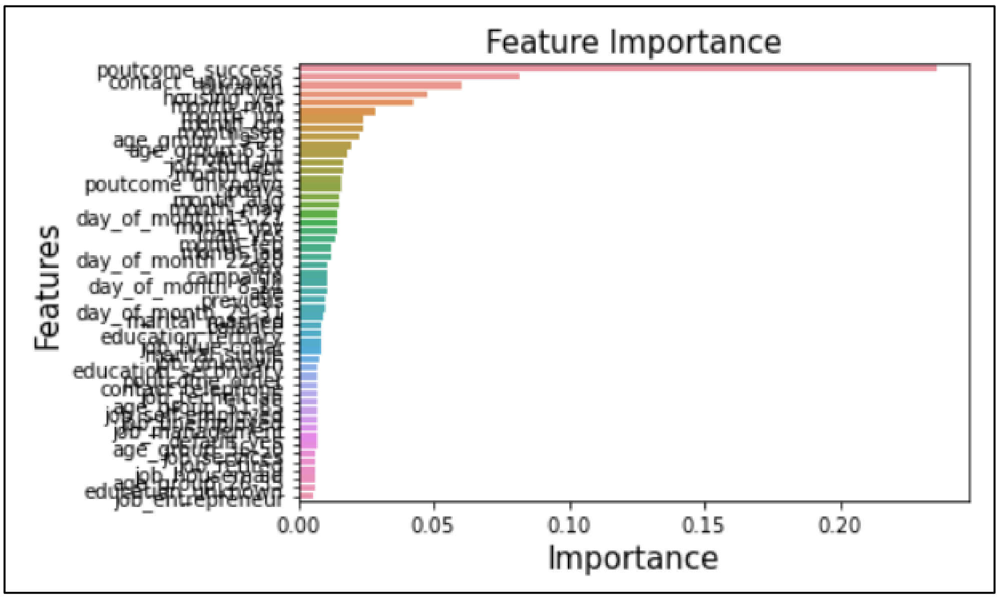
```
xgb_model.score(X_test,y_test)
```

```
0.9092450604541433
```

We can see that after performing Tuning accuracy of both train and test dataset are almost equal, hence we can proceed with the model for prediction.

Feature Importance graph represents those features on which the dependent variable is dependent the highest.

**Identify the important features with the help of XG Boost**



We can see from our outcome that for this prediction model Poutcome_success, contact, Housing are the most important features, on which the dependent variable is dependent.

**Future Actions:**

Through the outcome of the models, we understand that the time taken by XG Boost is less and this model is giving highest accuracy. Hence, we proceed to use XG Boost model.

Further, we would like to extend the project by retaining the customers for subscribing to the term deposit. By identifying the features which contribute the most and how we can improve any of the factor with respect to business perspective.