A Project Report on

TELEMARKETING CAMPAIGN PREDICTION

Submitted to

E&ICT IIT Roorkee

Submitted by

Gobind Kumar

CHAPTERS

Chapter 1- Introduction

To optimize and improve marketing strategies and their effectiveness bank industry needs to understand their customer's need. By understanding their customer's need bank industry can make smarter products and effective marketing plans. The main objective of this project is to predict customer's reaction to later marketing campaigns and improve the customer's reaction.

By exploring various features of customer banks can predict their customer's reaction. Bank can understand what particular type of customers are interested and what particular type of customers not interested. With this information banks can focus their efforts on particular type of customer and can plan new strategies to attract new customers.

Overall, this will lead to customer satisfaction as well as bank satisfaction.

Chapter 2- About the Resources

Data

The dataset is containing the details of customers who were contacted by direct phone call marketing campaigns. The data is of three months from May 2017 to July 2017. There are 16383 records.

- For each observation, the dataset has 16 attributes including both qualitative and quantitative attributes related to customer such as housing, age, salary, education, account balance etc.
- The dataset contains one binary output variable which has only two values "yes" or "no". This binary output variable reveling the outcome of the marketing campaigns. "yes" means customer has taken the car loan and "no" means customer did not taken the car loan.
- Dataset: Source(https://www.kaggle.com/gauravduttakiit/bank-telemarketing-campaign-case)

Software

For this project I have used Jupyter Notebook with python 3.

I have also used Pandas, Numpy, Matplotlib, Seaborn, Imbalanced-learn liberaries.

References

Asare-Frempong, Justice & Jayabalan, Manoj. (2017). Predicting Customer Response to Bank Direct Telemarketing Campaign. 10.1109/ICE2T.2017.8215961.

Chapter 3- Data Cleaning

Before proceeding to Exploratory data analysis, we required to clean the data. A number of changes were made to clean the dataset.

- Drop the unwanted attributes such as “customerid”.
- Drop outliers (such data points which are more than three SD away from the mean).
- Convert the “yes/no” value of response variable to 1/0.
- Find if any column containing null value and fill them with appropriate method.
- Make the dataset balanced before creating the model.

Chapter 4- Exploratory Data Analysis

Now we will explore the dataset by plotting the graphs and by finding distribution of key variables. In Exploratory Analysis we tried to find distributions of categorical as well as numerical features. Fig. 1 shows the Distribution of categorical features.

From fig. 1 we can understand the following points:

- Married customers are majority in the dataset.
- Maximum customers have secondary education.
- Most of the customers have taken housing loan.
- Contact details of most of the customers is unknown.
- Mostly customers are contacted in may month 2017.

Next, we try to find distribution of numerical features:

From fig. 2,3,4,5 we ca say that age is distributed better than days, balance and duration.

After that we try to find Relationship between categorical features and label

From fig.6,7,8 we can understand that

- Married customers have shown more interest.
- 'blue-collar' job type customers have shown maximum interest but majority of them said 'No'.

Fig. 9 shows the Relationship between Continuous Numerical Features and label.

Fig. 10 shows the correlation between numerical features. From figure we can understand that no feature is heavily correlated with other features.

Visualize the distribution of Categorical Feature:

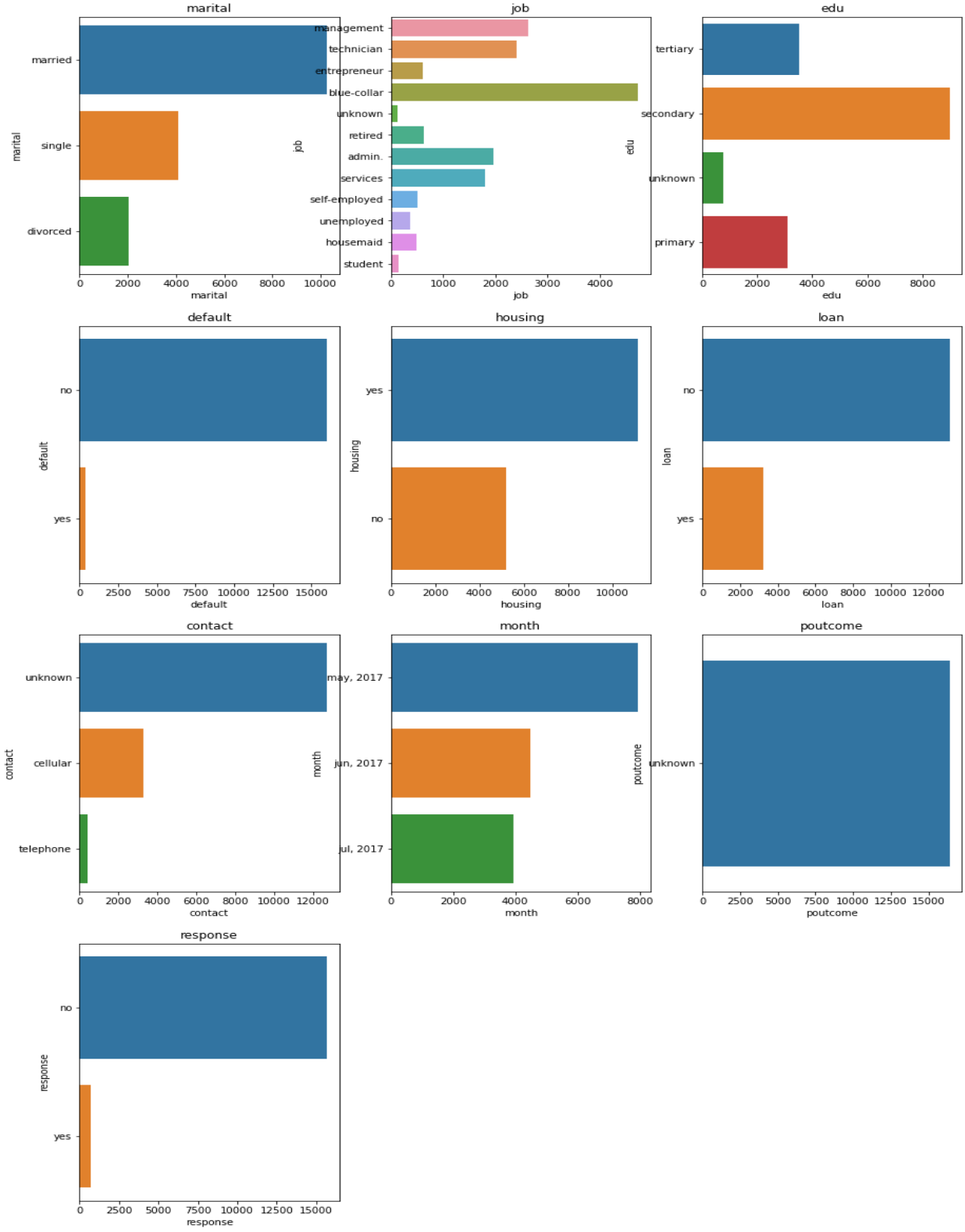


Fig. 1 Distribution of Categorical Feature

Visualize the distribution of Numerical Feature:

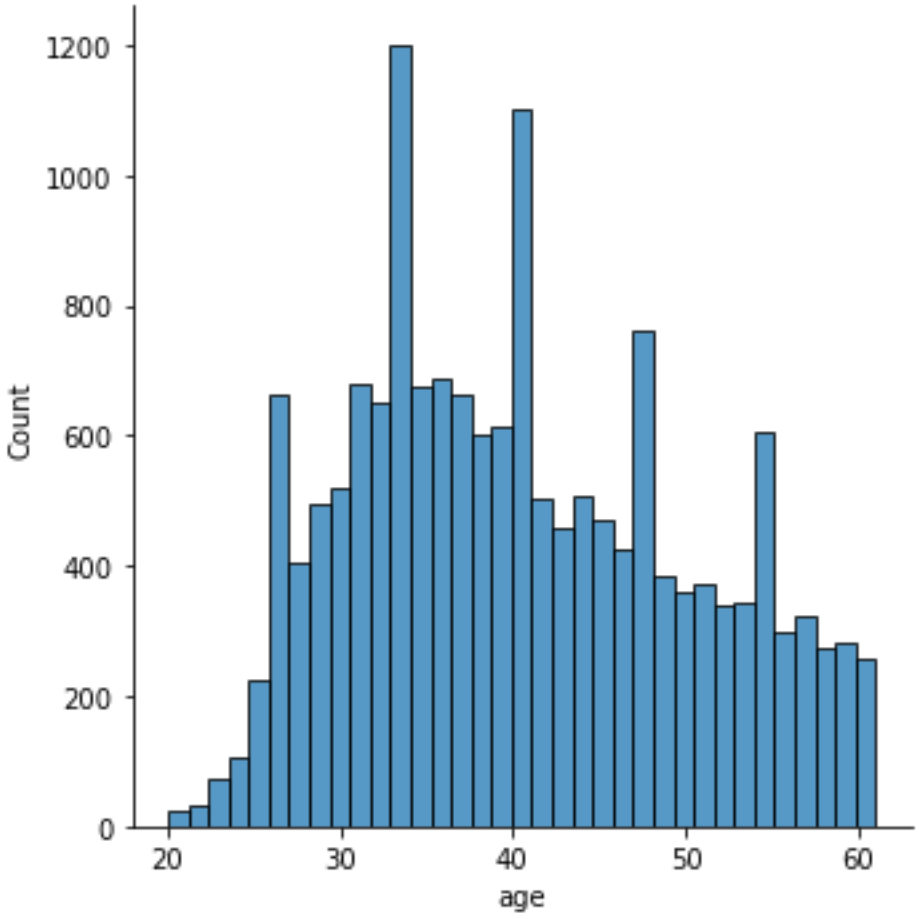


Fig. 2 Distribution of Age

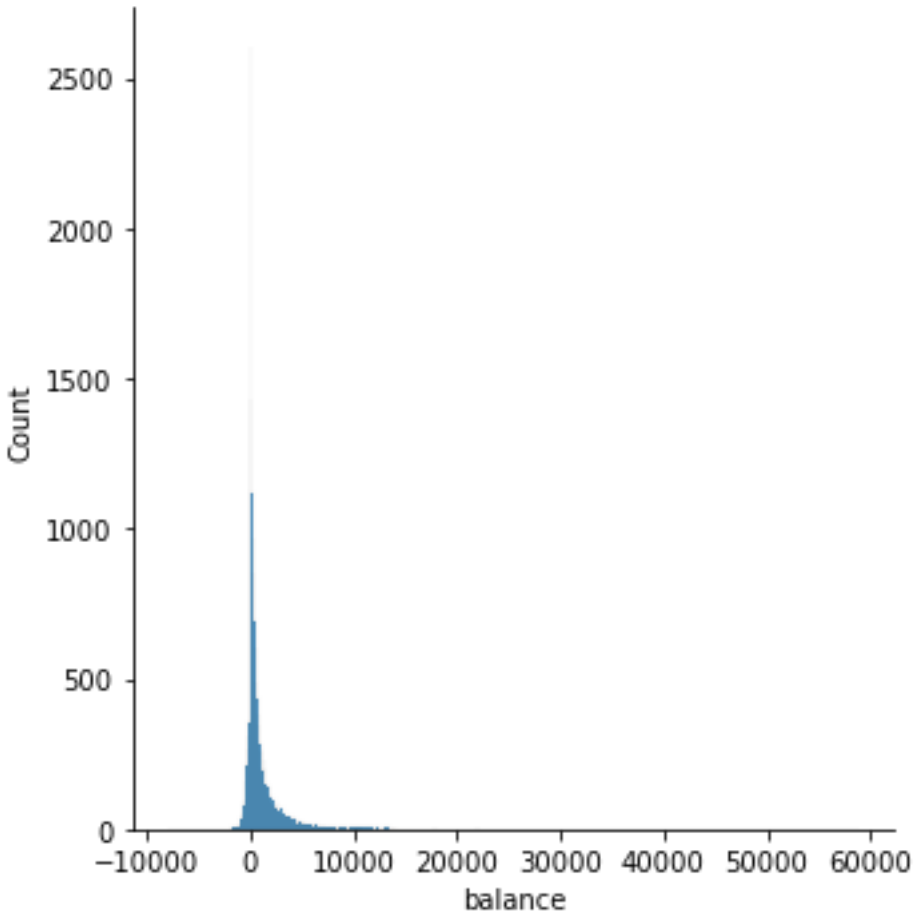


Fig. 3 Distribution of Balance

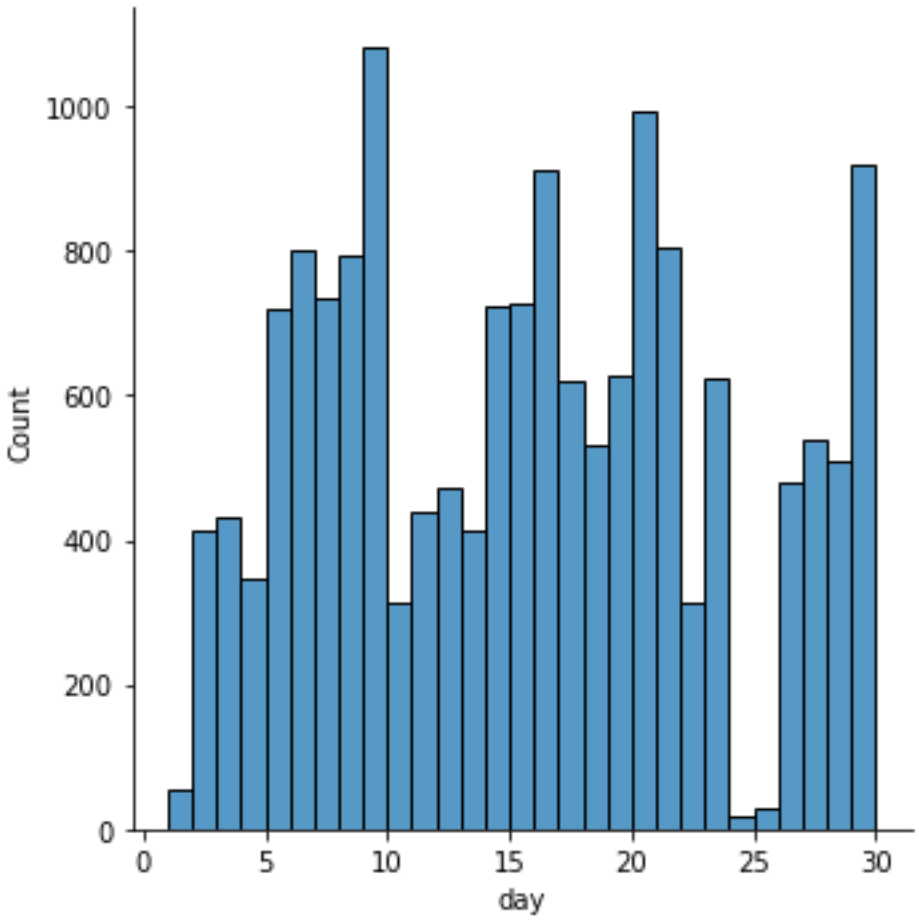


Fig. 4 Distribution of Day

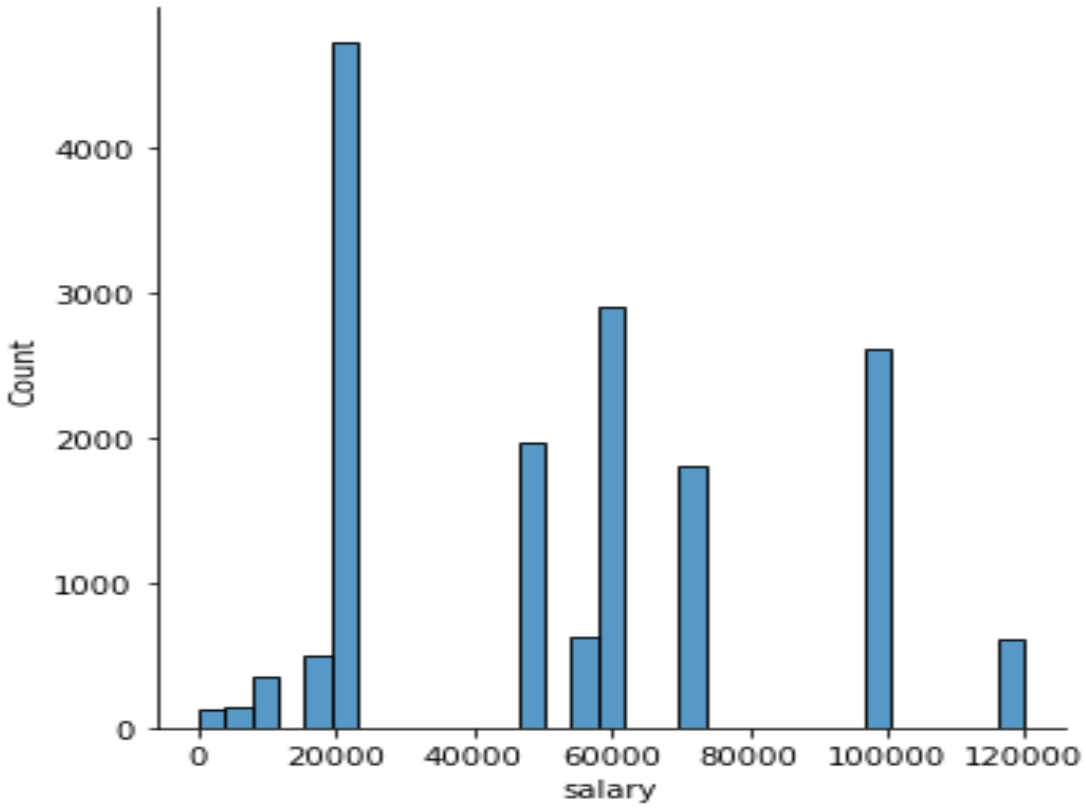


Fig. 5 Distribution of Salary

Finding Relationship between Categorical features and Output Variable:

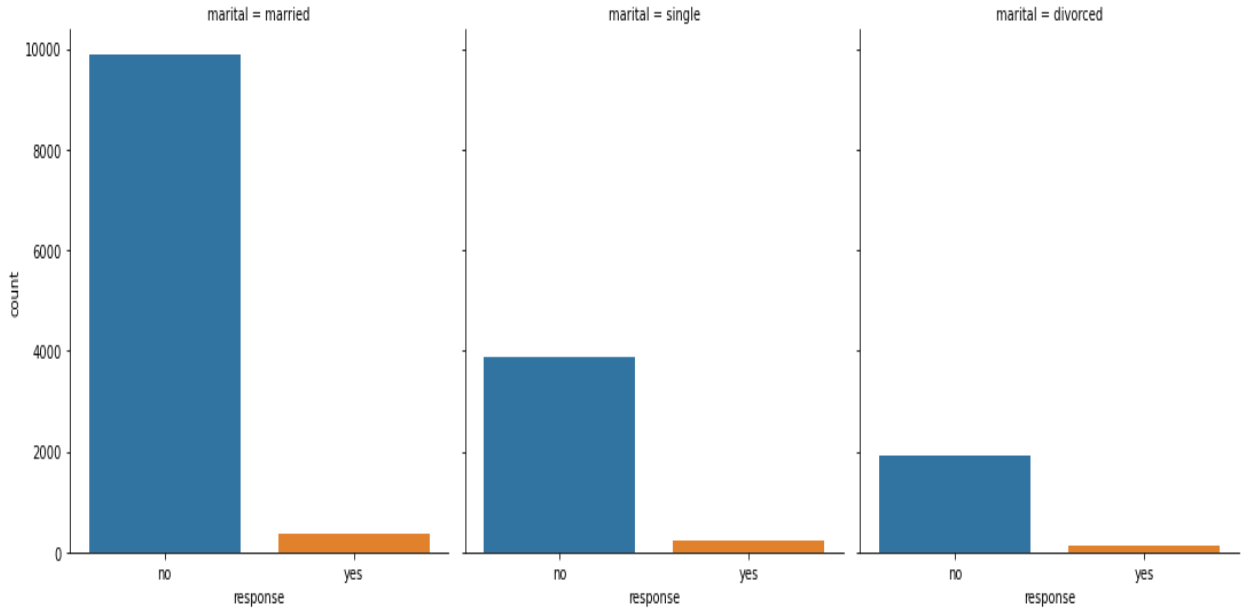


Fig. 6 Relationship between ‘marital’ and ‘response’

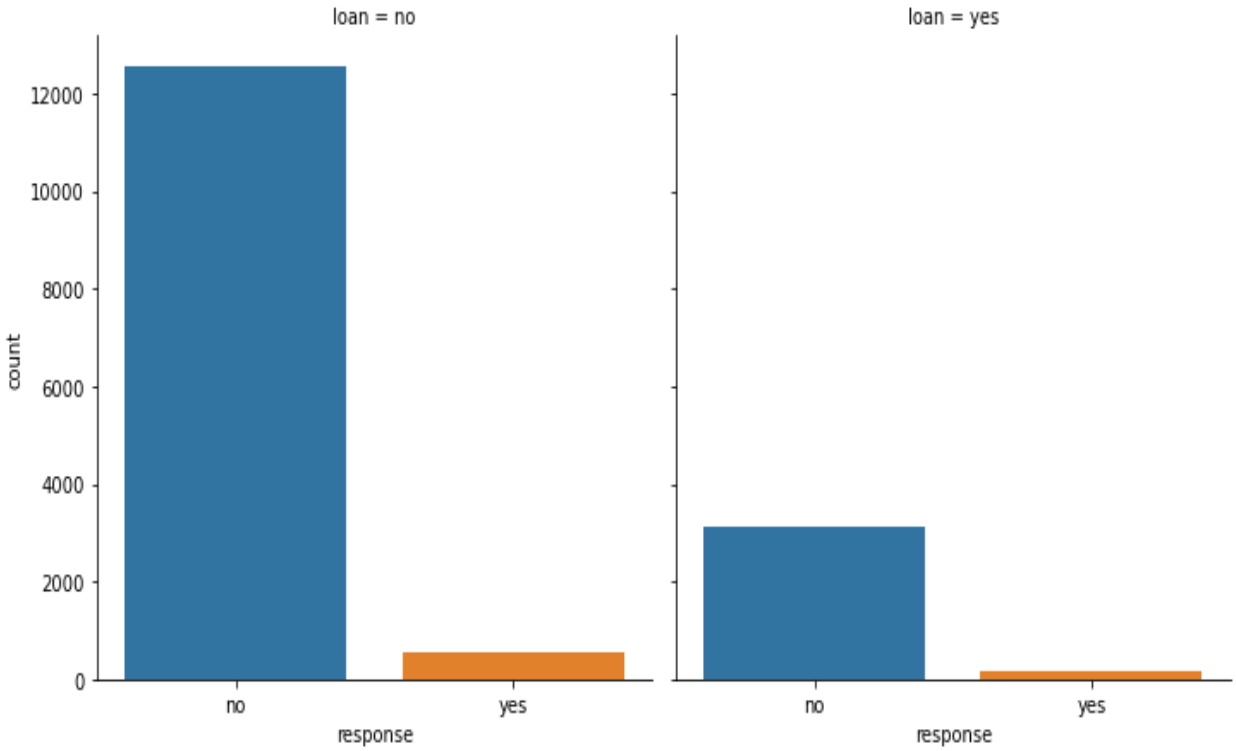


Fig. 7 Relationship between ‘loan’ and ‘response’

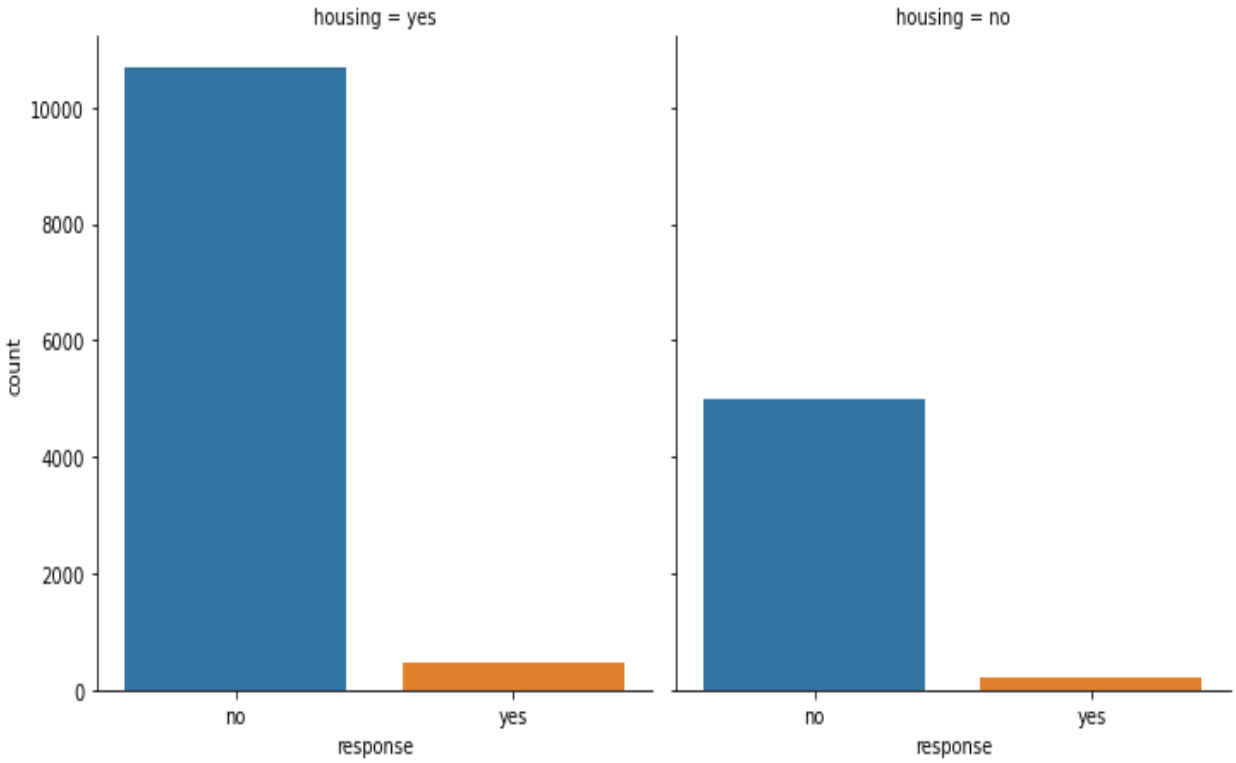


Fig. 8 Relationship between ‘housing’ and ‘response’

Finding Relationship between Numerical features and Output Variable:

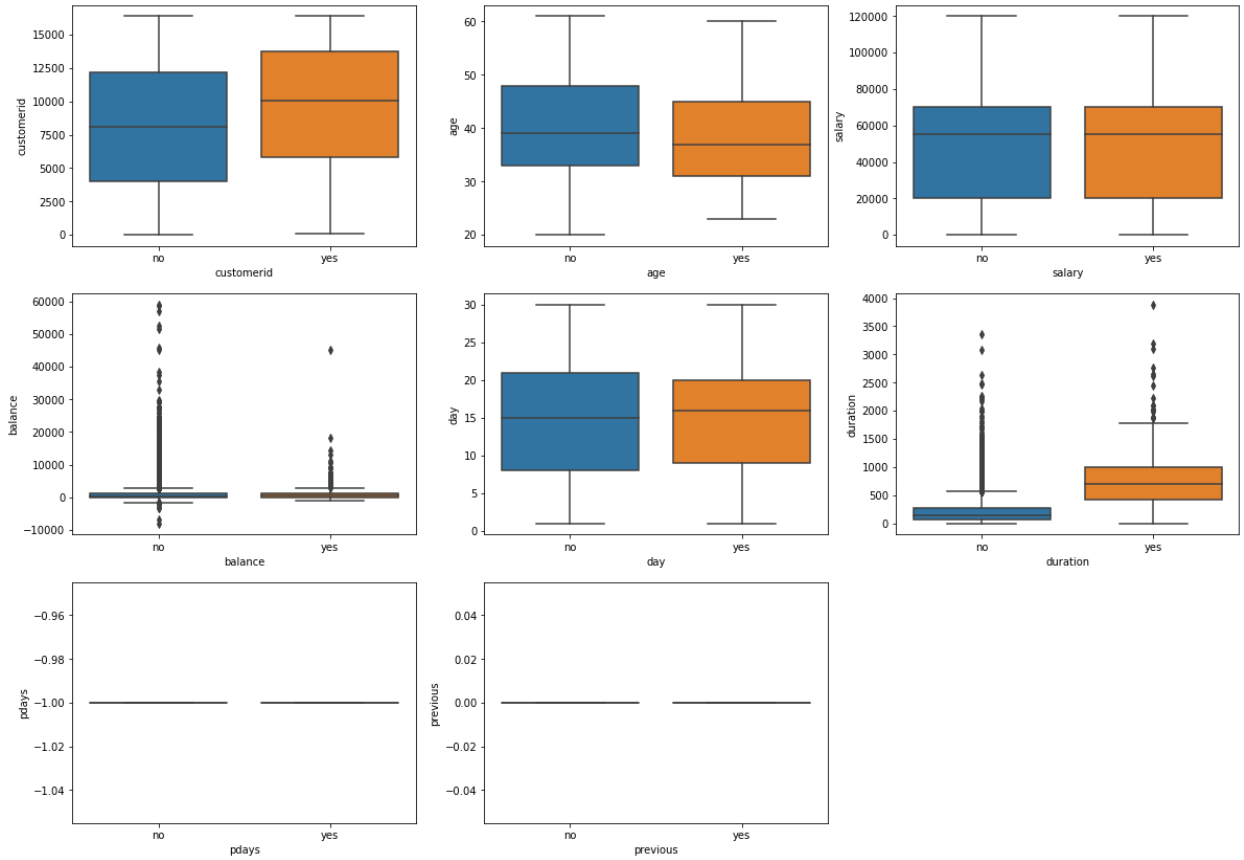


Fig. 9 Relationship between Numerical features and Output Variable

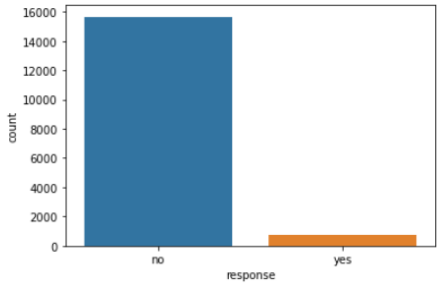
Finding Correlation between numerical features:

Fig. 10 Correlation map of numerical features

Chapter 5- Machine Learning: Classification

Balancing the data before model training.

```
In [24]: # Checking by countplot
         sns.countplot(x='response',data=df)
         plt.show()
```




```
In [25]: # Checking by numbers.
         df['response'].groupby(df['response']).count()

Out[25]: response
         no     15665
         yes      708
         Name: response, dtype: int64
```

- The Dataset is biased toward 'no'. So it is a unbalansed dataset. In feature enginering i will try to baance the dataset.

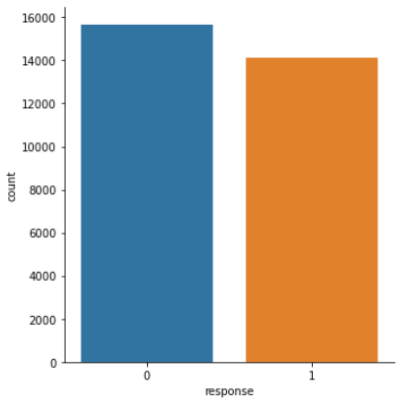
Fig. 11 Checking ‘yes’ vs ‘no’ in the dataset.

After balancing the data:

```
In [43]: !pip install imbalanced-learn

         Requirement already satisfied: imbalanced-learn in c:\users\govin\anaconda3\lib\site-packages (0.8.1)
         Requirement already satisfied: numpy>=1.13.3 in c:\users\govin\anaconda3\lib\site-packages (from imbalanced-learn) (1.20.1)
         Requirement already satisfied: scikit-learn>=0.24 in c:\users\govin\anaconda3\lib\site-packages (from imbalanced-learn) (0.24.
         1)
         Requirement already satisfied: scipy>=0.19.1 in c:\users\govin\anaconda3\lib\site-packages (from imbalanced-learn) (1.6.2)
         Requirement already satisfied: joblib>=0.11 in c:\users\govin\anaconda3\lib\site-packages (from imbalanced-learn) (1.0.1)
         Requirement already satisfied: threadpoolctl>=2.0.0 in c:\users\govin\anaconda3\lib\site-packages (from scikit-learn>=0.24->imb
         alanced-learn) (2.1.0)

In [44]: from imblearn.over_sampling import RandomOverSampler
         oversample=RandomOverSampler(0.90)

         C:\Users\govin\anaconda3\lib\site-packages\imblearn\utils\_validation.py:587: FutureWarning: Pass sampling_strategy=0.9 as keyw
         ord args. From version 0.9 passing these as positional arguments will result in an error
           warnings.warn(

In [45]: X_rs,y_rs=oversample.fit_resample(X.to_numpy(),y.to_numpy())

In [46]: X.shape,y.shape,X_rs.shape,y_rs.shape

Out[46]: ((16383, 32), (16383,), (29782, 32), (29782,))
```

Fig. 12 Stabilizing the dataset using RandomOverSampler from imblearn library

```
In [50]: y_rsdf=pd.DataFrame(y_rs,columns=['response'])

In [51]: sns.catplot(x='response',kind='count',data=y_rsdf)

Out[51]: <seaborn.axisgrid.FacetGrid at 0x21dc641eb50>
```




```
In [72]: y_rsdf['response'].groupby(y_rsdf['response']).count()

Out[72]: response
         0    15675
         1    14107
         Name: response, dtype: int64
```

Fig. 13 Catplot After Stabilizing the dataset.

Applying Linear Regression:

We took 70% data for training and 30% for testing

```
In [52]: from sklearn.model_selection import train_test_split
         Xtrain,Xtest,ytrain,ytest=train_test_split(X_rsdf,y_rsdf,train_size=0.7)

In [53]: Xtrain.shape,Xtest.shape,ytrain.shape,ytest.shape

Out[53]: ((20847, 32), (8935, 32), (20847, 1), (8935, 1))

In [54]: from sklearn.linear_model import LogisticRegression

In [55]: logR=LogisticRegression(solver='lbfgs', max_iter=1000).fit(Xtrain,np.ravel(ytrain,order="c"))
         logR

Out[55]: LogisticRegression(max_iter=1000)

In [56]: logR.predict_proba(Xtest)

Out[56]: array([[4.89152153e-01, 5.10847847e-01],
                [9.08553479e-01, 9.14465206e-02],
                [3.19682765e-01, 6.80317235e-01],
                ...,
                [8.86004355e-01, 1.13995645e-01],
                [9.58350284e-01, 4.16497162e-02],
                [9.80913116e-04, 9.99019087e-01]])

In [57]: print(logR.intercept_)

         [-0.07329451]

In [58]: print(logR.coef_)

         [[-4.86473929e-02 -3.55273521e-06 -3.11678577e-05  2.96105854e-02
            6.71709886e-03  2.01189229e-01 -1.94444009e-01 -8.00389438e-02
           -1.49339547e-02 -5.62935807e-02  2.30311694e-02  8.75836772e-03
            2.71701230e-02  1.65276315e-02 -2.01476540e-02 -4.45225982e-02
            1.94578394e-03 -5.79029639e-03  9.67358283e-04 -1.00060738e-02
           -3.21559232e-02 -7.82750786e-02  2.48718828e-02  1.22653952e-02
            8.79469818e-01  4.05549653e-02 -9.93318507e-01  8.96212878e-01
           -1.63755142e-01 -8.05751460e-01 -1.97432947e-01  5.01901818e-02]]
```

Fig. 14 Importing Linear Regression and applying to our training dataset.

In [65]:
```
ct=pd.crosstab(df_res['response'],df_res['predicted'])
ct
```

Out[65]:

response \ predicted	0	1
0	3857	862
1	792	3424

In [66]:
```
from sklearn.metrics import accuracy_score,f1_score,precision_score,recall_score,roc_auc_score
from sklearn import metrics as sm
```

In [67]:
```
accuracy=sm.accuracy_score(ytest2,pred2)
f1=f1_score(ytest2,pred2)
precision=precision_score(ytest2,pred2)
recall=recall_score(ytest2,pred2)
roc_auc=roc_auc_score(ytest2,pred2)
```

In [68]:
```
print('Accuracy is: ' ,accuracy)
print('F1 Score is: ' ,f1)
print('Precision is: ',precision)
print('Recall is: '   ,recall)
print('Roc Auc is: '  ,roc_auc)
```

```
Accuracy is:  0.8148852825965305
F1 Score is:  0.8054575394024935
Precision is:  0.7988800746616892
Recall is:  0.8121442125237192
Roc Auc is:  0.8147391967471318
```

Fig. 15 Results of LR model

Applying KNN classification:

```
In [69]: from sklearn.neighbors import KNeighborsClassifier
         from sklearn import metrics

         k_range = range(1,40)
         scores={}
         scores_list=[]
         for k in k_range:
             knn=KNeighborsClassifier(n_neighbors=k)
             knn.fit(Xtrain,np.ravel(ytrain,order="c"))
             _pred=knn.predict(Xtest)
             scores[k]=metrics.accuracy_score(ytest,_pred)
             scores_list.append(metrics.accuracy_score(ytest,_pred))

In [70]: scores

Out[70]: {1: 0.9823167319529938,
          2: 0.9823167319529938,
          3: 0.966200335758254,
          4: 0.966200335758254,
          5: 0.9540011191941802,
          6: 0.9540011191941802,
          7: 0.9419138220481253,
          8: 0.9419138220481253,
          9: 0.9326245103525461,
          10: 0.9326245103525461,
          11: 0.9252378287632904,
          12: 0.9253497481813094,
          13: 0.9174034695019586,
          14: 0.9186345831001679,
          15: 0.9131505316172356,
          16: 0.91035254616676,
          17: 0.9037493005036373,
          18: 0.9016228315612759,
          19: 0.8944599888080582,
          20: 0.8943480693900392,
          21: 0.8885282596530498,
          22: 0.8872971460548406,
          23: 0.8828203693340795,
          24: 0.8805819809736989,
          25: 0.8762171236709569,
          26: 0.8731952993844432,
          27: 0.8719641857862339,
          28: 0.872188024622272,
          29: 0.8708449916060437,
          30: 0.8710688304420817,
          31: 0.8679350867375489,
          32: 0.8695019585898154,
          33: 0.8688304420817011,
```

Fig. 16 Importing KNN from sklearn and applying to our training data.

```
In [71]: plt.plot(k_range,scores_list)
         plt.xlabel('Value of K for KNN')
         plt.ylabel('Testing Accuracy')R

Out[71]: Text(0, 0.5, 'Testing Accuracy')
```

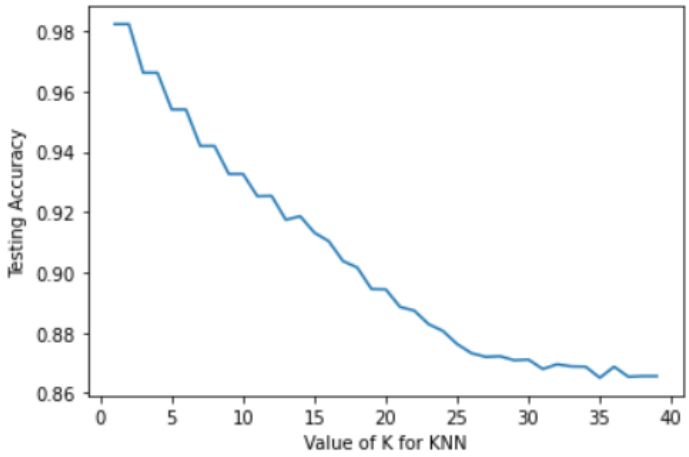


Fig. 17 Results of KNN model.

Chapter 6- Conclusion:

With Linear Regression we get the accuracy of 81.4 percent and F1 score is 0.80.

Precision with Linear Regression is 0.79. The value of ROC AUC is 0.81.

On the other hand, with KNN we tried with rang of (1,40) of K. Initially the accuracy of KNN model decreases drastically but as the value K increases to 26-27 the accuracy start being stable.

Accuracy of KNN model is more than Logistic Regression.

Contact Details:

Gobind Kumar

email: govind166kumar@gmail.com

M. Number: 7015753821