**BANK CUSTOMER CHURN PREDICTION WITH MACHINE LEARNING**

P.S.SAI VISHNU, KAUSHIK RAMABHOTLA, JUHI KUMARI

**Abstract**:

Customer churn is a major problem of customers leaving your products/subscription and moving to another service. Due to direct effect on profit margins, businesses now are looking to identify customers who are at the risk of churning and retaining them by personalized promotional offers. In order to retain them, they need to identify the customers as well as the reason of churning so that they can provide the customers with personalized offers and products. The aim of our project is to solve this problem for banking domain, by identifying which customers are at risk of churning and what are the reasons for churning with the help of machine learning algorithms. The main focus of the project is to predict how many customers are likely to churn from the bank with different machine learning algorithms.

1. **Introduction**

The term Customer Churn refers to the case when a customer or subscriber of a particular company terminates his or her relationship with the services provided by that company. The cost of customer churn in total includes the revenue, which is lost, and the costs associated with marketing to replace the old customers with new ones. Apart from that, the cost of initially acquiring a customer might not have been fully covered by that customer’s spending to date. Hence, minimizing customer churn is a main concern for most companies. Therefore, the capacity of predicting the customers who are on the verge of exiting the company’s services while there is still time left to take necessary actions to prevent it from happening appears to be a huge source of potential revenue for the companies.

Banking Industries which are mostly customer centric are as well not exempt from the repercussions of customer churning. They track interactions of the customer with the company to detect behaviors such as dormancy of account status reduce transactions etc in order to detect early signs of churn and prevent it. The top factors. Which lead customers to look for other alternatives seem to be bad quality of service. Exorbitant. Banking fees in today’s world each and every poor customer service. Seemingly unfair prices and lack of access to the. Services at all times and places trigger customer sentiments which lead to risk of churn and major revenue loss in turn.

Therefore, attaining the ability to predict churn which in turn enables the banks to prevent it by understating the customer’s needs. Choices and sentiments is the motive behind this project.

So, here We aim to accomplish the following for this study:

1. Identify and visualize which factors contribute to customer churn.
2. Build a prediction model that will perform the following.

* Classify if a customer is going to churn or not.
* Preferably and based on model performance, choose a model that will attach a probability to the churn to make it easier for customer service to target low hanging fruits in their efforts to prevent churn.

1. **PROCEDURE**

**I. Import the required libraries.**

Here we import the libraries required for the algorithm. The libraries required are pandas, numpy and to visualize the data we use the libraries such as matplotlib.pyplot and seaborn.

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

import seaborn as sns

**II. Import and read the required dataset**

So here the dataset we need is “Bank Customer Churn” dataset and if the dataset is in CSV format then we use read\_csv() function and if the dataset is in Excel format then we use read\_excel() function to import and read the dataset.

import io

df = pd.read\_excel('FFFFFFFFFFFFF.xlsx')

and to get information about the dataset we use the function info() and the code will be df.info() and the information we see is,

Data columns (total 23 columns):

# Column Non-Null Count Dtype

--- ------ -------------- -----

0 churn 23312 non-null int64

1 age 23312 non-null int64

2 rent\_or\_own 23312 non-null object

3 trivia\_played 23312 non-null int64

4 trivia\_shared\_results 23312 non-null int64

5 trivia\_view\_unlocked 23312 non-null int64

6 trivia\_view\_results 23312 non-null int64

7 cards\_share 23312 non-null int64

8 cards\_viewed 23312 non-null int64

9 cards\_helpful 23312 non-null int64

10 cards\_not\_helpful 23312 non-null int64

11 cards\_clicked 23312 non-null int64

12 has\_used\_mobile 23312 non-null int64

13 more\_than\_one\_mobile\_device 23312 non-null int64

14 payfreq 23312 non-null int64

15 loan\_pending 23312 non-null int64

16 withdrawn\_application 23312 non-null int64

17 paid\_off\_loan 23312 non-null int64

18 did\_not\_accept\_funding 23312 non-null int64

19 un\_linked\_account 23312 non-null int64

20 re\_linked\_account 23312 non-null int64

21 cash\_back\_engagement 23312 non-null float64

22 has\_referred 23312 non-null int64

There are total 23 columns and 48000 rows, here there are no missing values and here we have to differentiate the categorical variables and continuous variables. The following columns are the categorical variables:

1.churn

2.rent\_or\_own

3.has\_used\_mobile

4.more\_than\_one\_mobile

5.payfreq

6.loan\_pending

7.withdrawn\_application

8.paid\_off\_loan

9.did\_not\_accept\_funding

10.un\_linked\_account

11.re\_linked\_account

12.has\_referred

And the remaining columns are the continuous variables.

**III. Exploratory Data Analysis(EDA)**

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

Here we do the exploratory data analysis with different plots such as countplots and boxplots.

So we plot a graph for each and every column with the help of countplot for better understanding of the dataset. So the code for the countplot will be as follows:

and the countplots are as follows:

COUNT PLOT:

fig,ax =plt.subplots(1, 4, figsize=(25, 10))

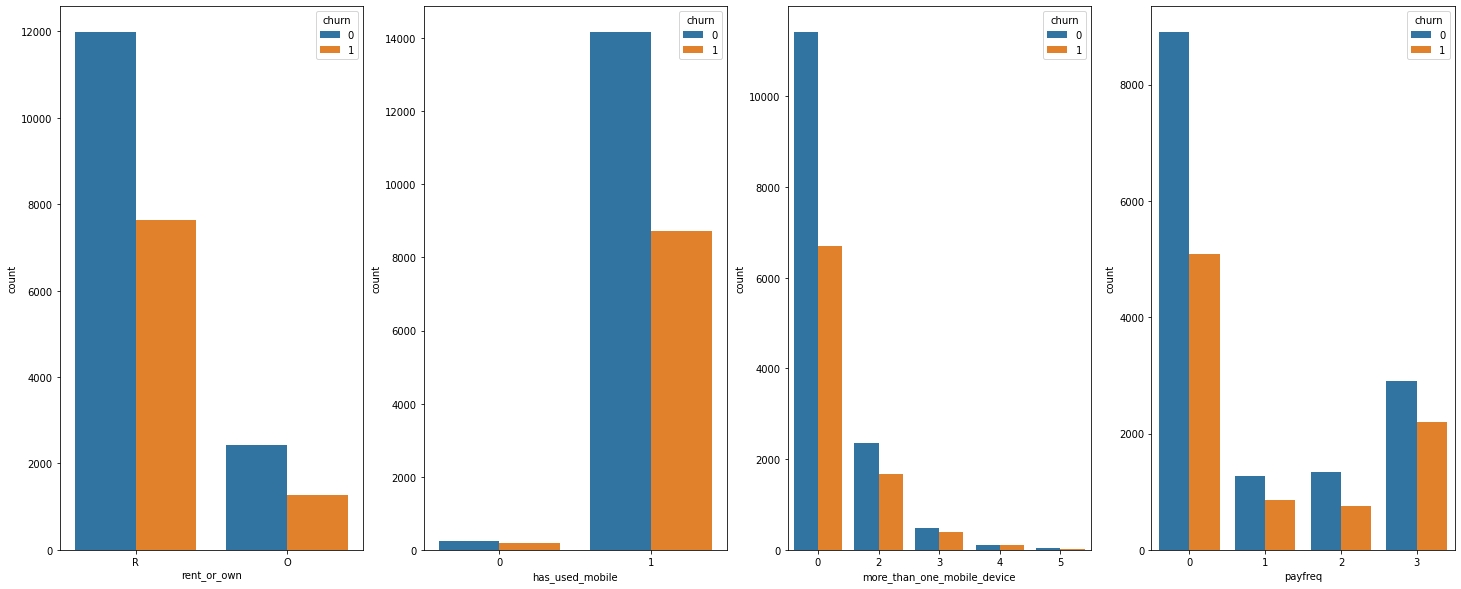
sns.countplot(x='rent\_or\_own',hue='churn',data=df,ax=ax[0])

sns.countplot(x='has\_used\_mobile',hue='churn',data=df,ax=ax[1])

sns.countplot(x='more\_than\_one\_mobile\_device',hue='churn',data=df,ax=ax[2])

sns.countplot(x='payfreq',hue='churn',data=df,ax=ax[3])

plt.show()



fig,ax =plt.subplots(1, 4, figsize=(25, 10))

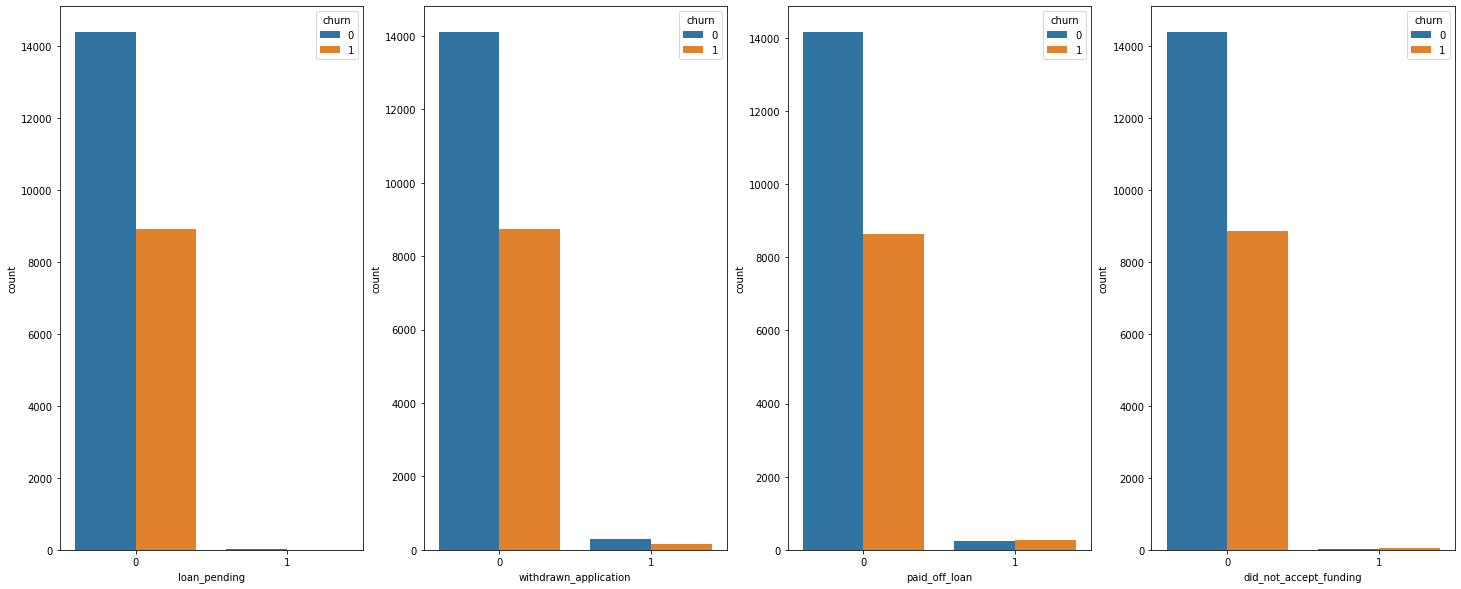
sns.countplot(x='loan\_pending',hue='churn',data=df,ax=ax[0])

sns.countplot(x='withdrawn\_application',hue='churn',data=df,ax=ax[1])

sns.countplot(x='paid\_off\_loan',hue='churn',data=df,ax=ax[2])

sns.countplot(x='did\_not\_accept\_funding',hue='churn',data=df,ax=ax[3])

plt.show()



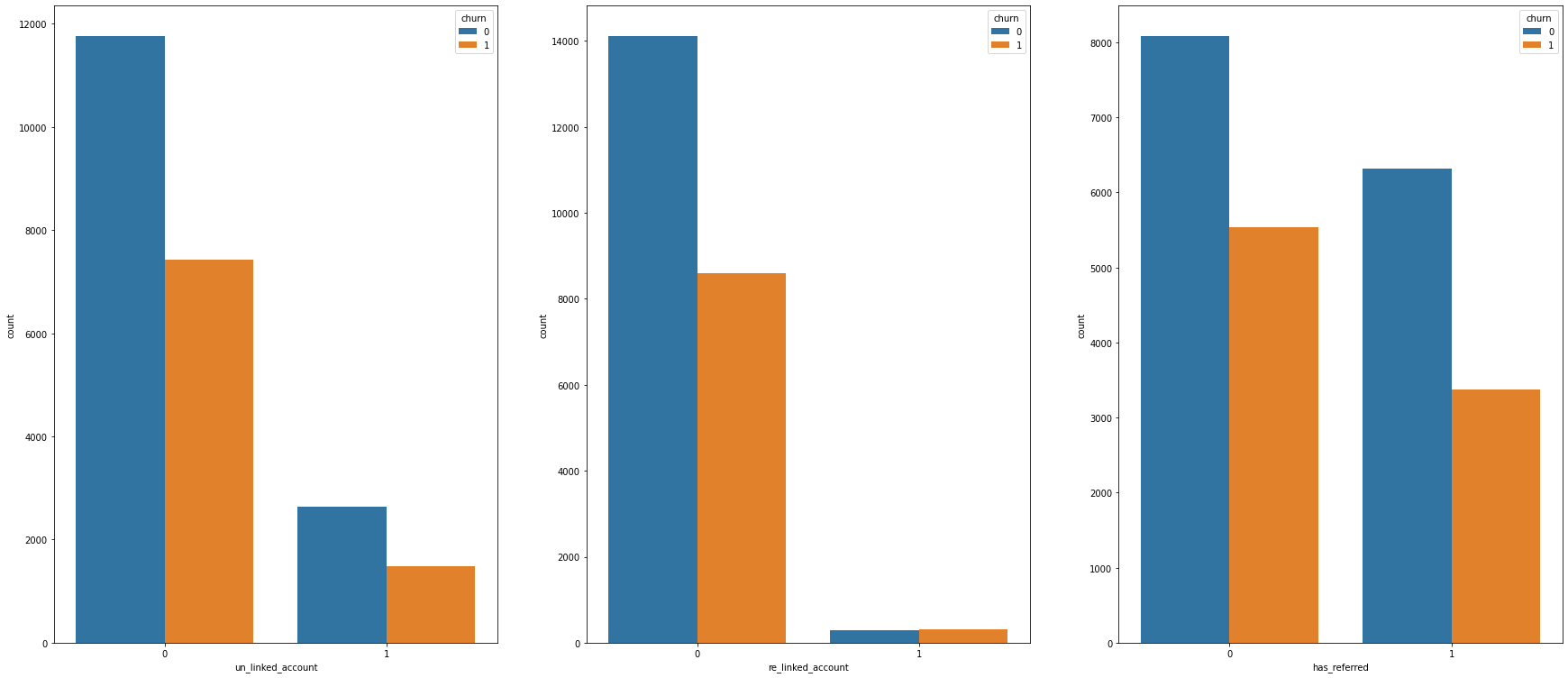
fig,ax =plt.subplots(1,3,figsize=(30, 13))

sns.countplot(x='un\_linked\_account',hue='churn',data=df,ax=ax[0])

sns.countplot(x='re\_linked\_account',hue='churn',data=df,ax=ax[1])

sns.countplot(x='has\_referred',hue='churn',data=df,ax=ax[2])

plt.show()



STATISTICAL PLOTS (BOX PLOTS) :

We will also plot the box plots to view the outliners and as there are 4 columns named after “TRIVIA”, we will start with them

A box plot (or box-and-whisker plot) shows the distribution of quantitative data in a way that facilitates comparisons between variables. The box shows the quartiles of the dataset while the whiskers extend to show the rest of the distribution.

And the code for boxplot is as follows:

fig,ax =plt.subplots(1,4,figsize=(20,5))

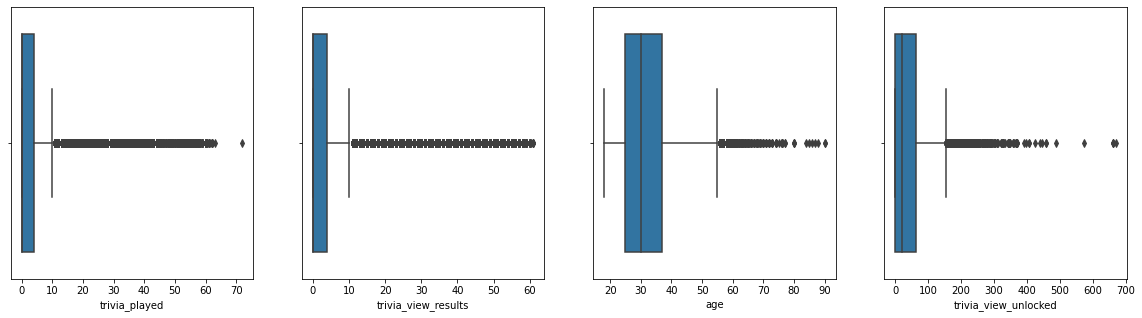
sns.boxplot(x=df['trivia\_played'],ax=ax[0])

sns.boxplot(x=df['trivia\_view\_results'],ax=ax[1])

sns.boxplot(x=df['age'],ax=ax[2])

sns.boxplot(x=df['trivia\_view\_unlocked'],ax=ax[3])

plt.show()

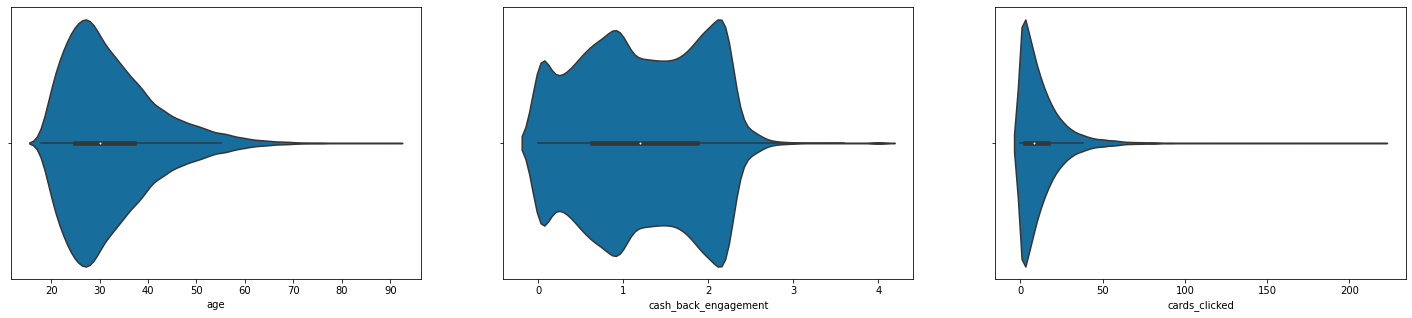


fig,ax =plt.subplots(1,3,figsize=(25,5))

sns.violinplot(x=df['age'],palette='colorblind',ax=ax[0])

sns.violinplot(x=df['cash\_back\_engagement'],ax=ax[1],palette='colorblind')

sns.violinplot(x=df['cards\_clicked'],ax=ax[2],palette='colorblind')



**IV. Dimensionality Reduction**

Dimensionality Reduction refers to techniques that reduce the number of input variables in a dataset. More input features often make a predictive modeling task more challenging to model, more generally referred to as curse of dimensionality.

When dealing with high dimensional data, it is often useful to reduce the dimensionality by projecting the data to a lower dimensional subspace which captures the “essence” of the data. This is called dimensionality reduction.

Dimensionality reduction yields a more compact, more easily interpretable representation of the target concept, focusing the user’s attention on the most relevant variables.

We can do **Dimensionality Reduction** using **Feature Selection**

In this there are categorical columns and numerical columns. So, the column which is not required will be removed in both categorical and numerical columns. We also use **Low Variance Filter**. In low variance filter, data columns with little changes in the data carry little information. Thus, all data columns with a variance lower than a given threshold can be removed. Notice that the variance depends on the column range, and therefore normalization is required before applying this technique.

So the steps for Dimensionality Reduction are as follows:

1) Separate the numerical column from the categorical column.

2) Convert the categorical column into numerical column using Label Encoder.

3) Then we do Data Scaling using Normalization method.

4) Then we apply Low Variance Filter.

**Step-1: Separate the numerical column from categorical column**

We can separate the numerical column from categorical column using drop() function.

numerical\_cols = df[['age','trivia\_played','trivia\_shared\_results','trivia\_view\_unlocked','trivia\_view\_results','cards\_share','cards\_viewed','cards\_helpful','cards\_clicked','cash\_back\_engagement']]

categorical\_cols = df.drop(numerical\_cols,axis=1)

then we can see both the categorical column and numerical column.

**Step-2: Convert the categorical column into numerical column using Label Encoder**.

Here the categorical column is rent\_or\_own.

from sklearn.preprocessing import LabelEncoder

le = LabelEncoder()

categorical\_cols['rent\_or\_own'] = le.fit\_transform(categorical\_cols['rent\_or\_own'])

So the values in rent\_or\_own column will become 0 or 1.

**Step-3: Data Scaling Using Normalization Method**

Data Scaling is the method used to standardize the range of features of data. Since, the range of values of data may vary widely, it becomes a necessary step in data preprocessing while using machine learning algorithms.

Normalization refers to rescaling real valued numeric attributes into the range 0 and 1. It is useful to scale the input attributes for a model that relies on the magnitude of values, such as distance measures used in k-nearest neighbors and in the preparation of coefficients in regression.

from sklearn.preprocessing import normalize

num\_norm = normalize(numerical\_cols)

data\_scaled = pd.DataFrame(num\_norm)

So now data scaling has been done. Here the shape of the dataset will be 23312 rows and 10 columns.

**Step-4: Low Variance Filter**

In low variance filter, data columns with little changes in the data carry little information. Thus, all data columns with a variance lower than a given threshold can be removed. Notice that the variance depends on the column range, and therefore normalization is required before applying this technique.

Consider a variable in our dataset where all the observations have the same value, say 1. If we use this variable, do you think it can improve the model we will build? The answer is no, because this variable will have zero variance.

So, we need to calculate the variance of each variable we are given. Then drop the variables having low variance as compared to other variables in our dataset. The reason for doing this is that variables with a low variance will not affect the target variable.

So First we find the variance of the numerical variables of the datascaled dataset using the function data\_scaled.var().

And output we see is

0 0.080750

1 0.003160

2 0.000101

3 0.041227

4 0.003014

5 0.000031

6 0.068551

7 0.000075

8 0.004942

9 0.000045

These all are the variances of the dataset data\_scaled. Then we will concatenate the column and variance using concat() function. Then we can see the columns with variance near to zero these columns can be deleted but before deleting we have to check their correlations.

**V. CORRELATIONS**

Variables within a dataset can be related for lots of reasons.

* Variables within a dataset can be related for lots of reasons.
* One variable could be lightly associated with another variable.
* Two variables could depend on a third unknown variable.

It can be useful in data analysis and modeling to better understand the relationships between variables. The statistical relationship between two variables is referred to as their correlation.

A correlation could be positive, meaning both variables move in the same direction, or negative, meaning that when one variable’s value increases, the other variables’ values decrease. Correlation can also be neutral or zero, meaning that the variables are unrelated.

* **Positive Correlation**: both variables change in the same direction.
* **Neutral Correlation**: No relationship in the change of the variables.
* **Negative Correlation**: variables change in opposite directions.

First separate the numerical column from categorical variable in the dataset data\_scaled.

So in the numerical set the shape is 23312 rows and 11 columns.

numerical\_cols = df[['age','churn','trivia\_played','trivia\_shared\_results','trivia\_view\_unlocked','trivia\_view\_results','cards\_share','cards\_viewed','cards\_helpful','cards\_clicked','cash\_back\_engagement']]

categorical\_cols = df.drop(numerical\_cols,axis=1)

We can also delete some columns by correlating the columns in the numerical set.

So we can compare the columns by heat map.

The Code for HeatMap is as follows:

plt.figure(figsize=(25,13))

sns.heatmap( numerical\_cols.corr(),annot=True)

plt.show()



1)We can see that there are two instances where the correlation is higher1. We need to delete one of them.

2)We can see that, trivia\_played and trivia\_view\_results have +1 correlation

3)We can see that, cards\_viewed and cash\_back\_engagement have +0.85 correlation.

4)We can see that, trivia\_played || churn has a value of -0.14

5)We can see that, trivia\_view\_results || churn has a value of -0.14

6)We can see that, cards\_viewed || churn has a value of -0.22

7)We can see that, cash\_back\_engagement || churn has a value of -0.19

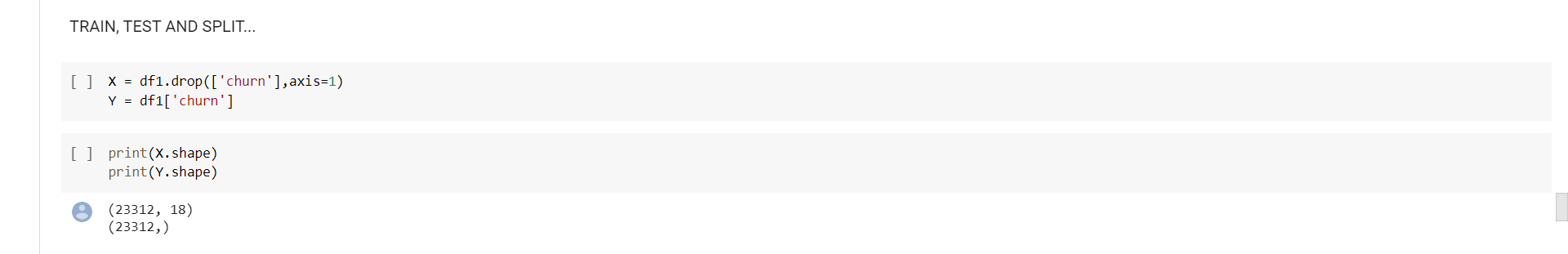
We opt to delete the columns (trivia\_view\_results, cash\_back\_engagement )

We can also see that the variance of these columns are also not that high, we can safely delete them. We can drop “Cash back engagement”, “Trivia view results” , “Cards helpful”, “Cards Share”.

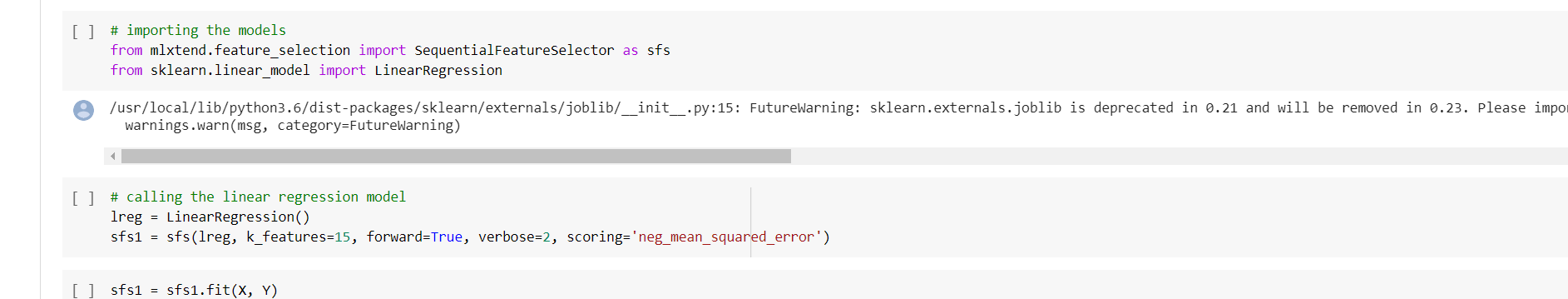
So we will have 23312 rows and 7 columns in numerical dataset and 23312 rows and 12 columns in categorical dataset. Then we concatenate the categorical and numerical columns and change the rent\_or\_own column.

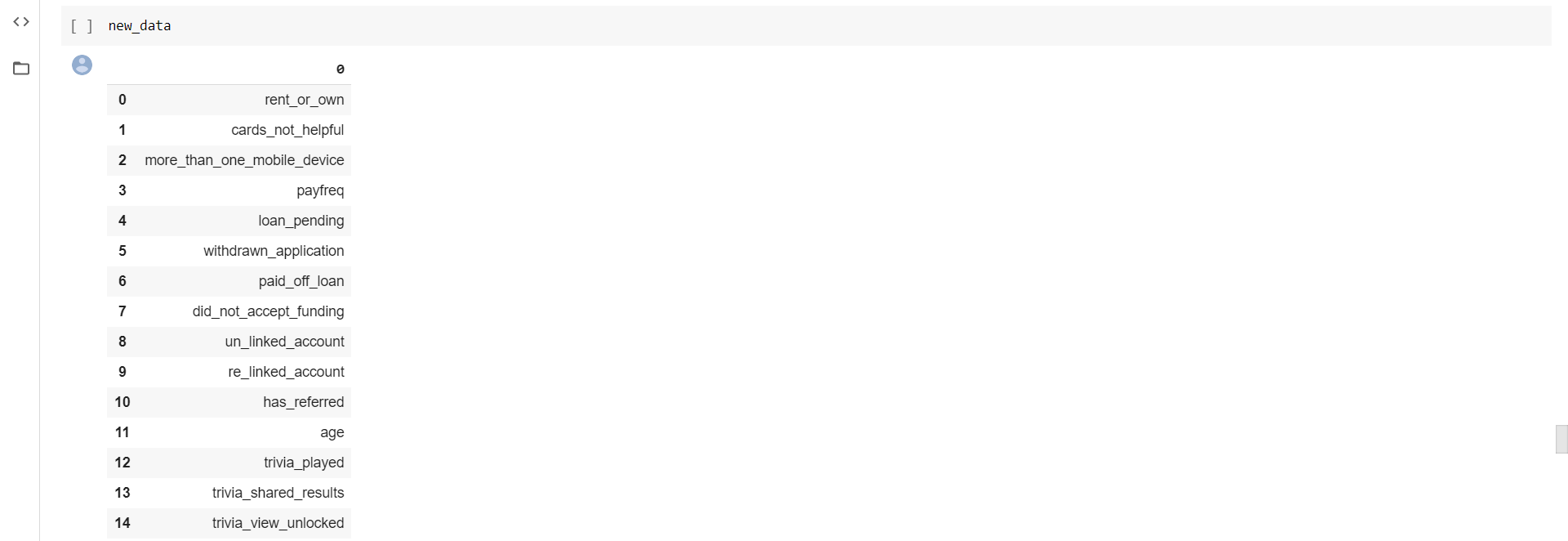
**VI. Train and Test Split using Feature scaling**

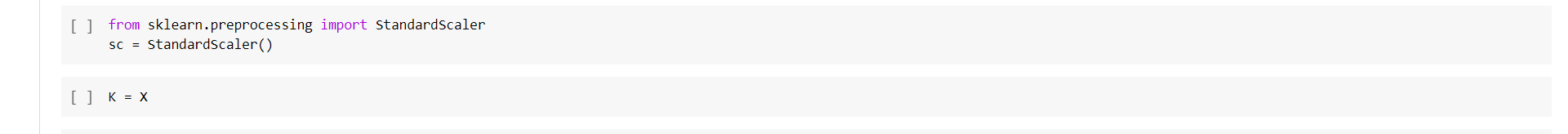
Split the dataset into train and test set using feature scaling and logical regression.



Import the Sequential Feature Selector, Linear regression model and Standard Scalar.



****The New dataset will be as follows:



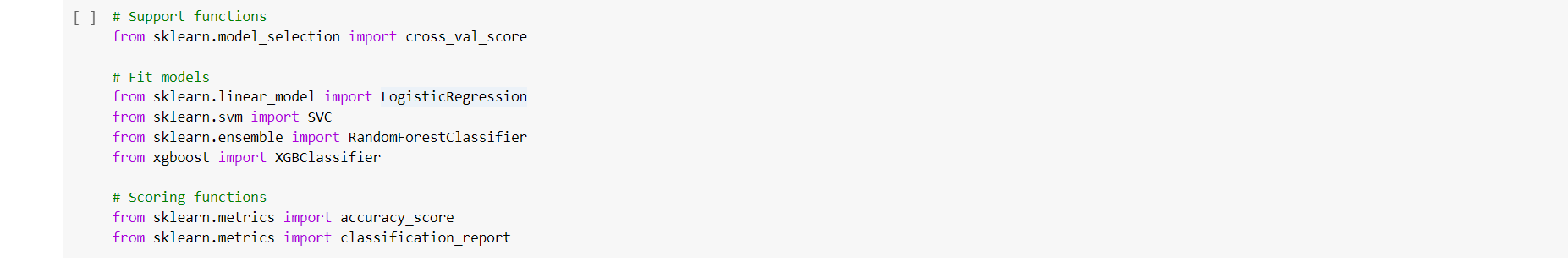
Now split the dataset into train and test dataset

from sklearn.model\_selection import  train\_test\_split

X\_train, X\_test, Y\_train, Y\_test = train\_test\_split(K, Y, test\_size=0.2,random\_state=25)

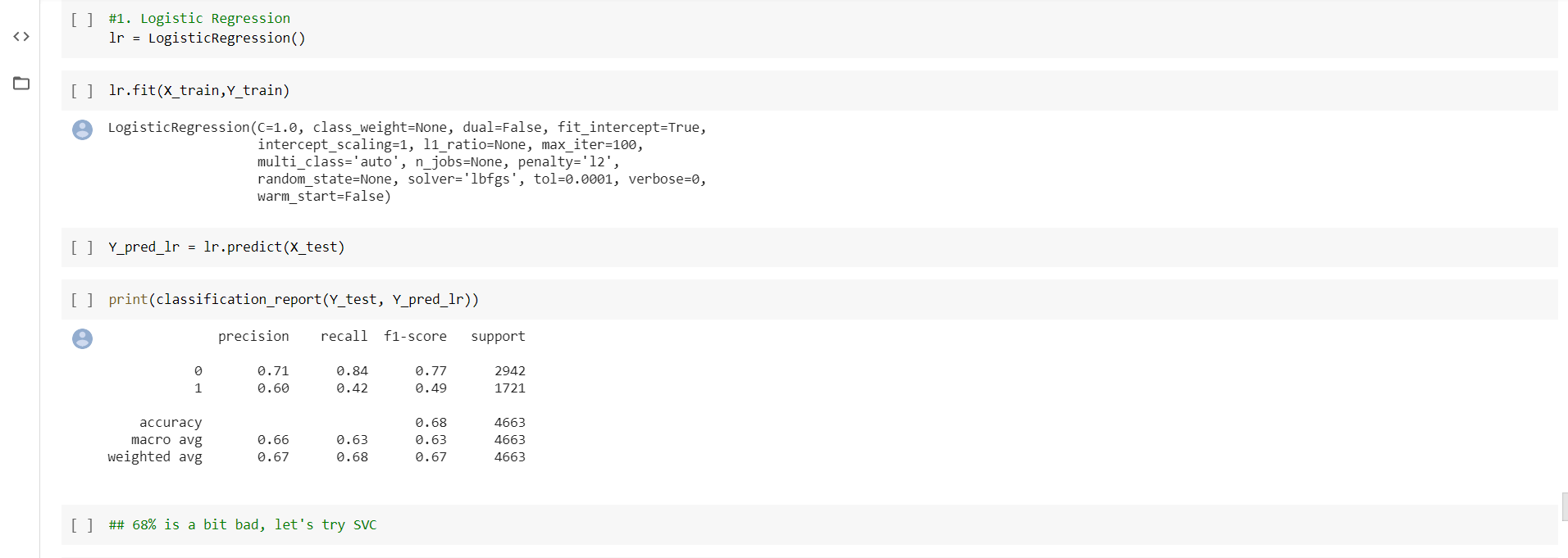
**VI. Model Training and Prediction:**

Import the required Regression models, Functions.

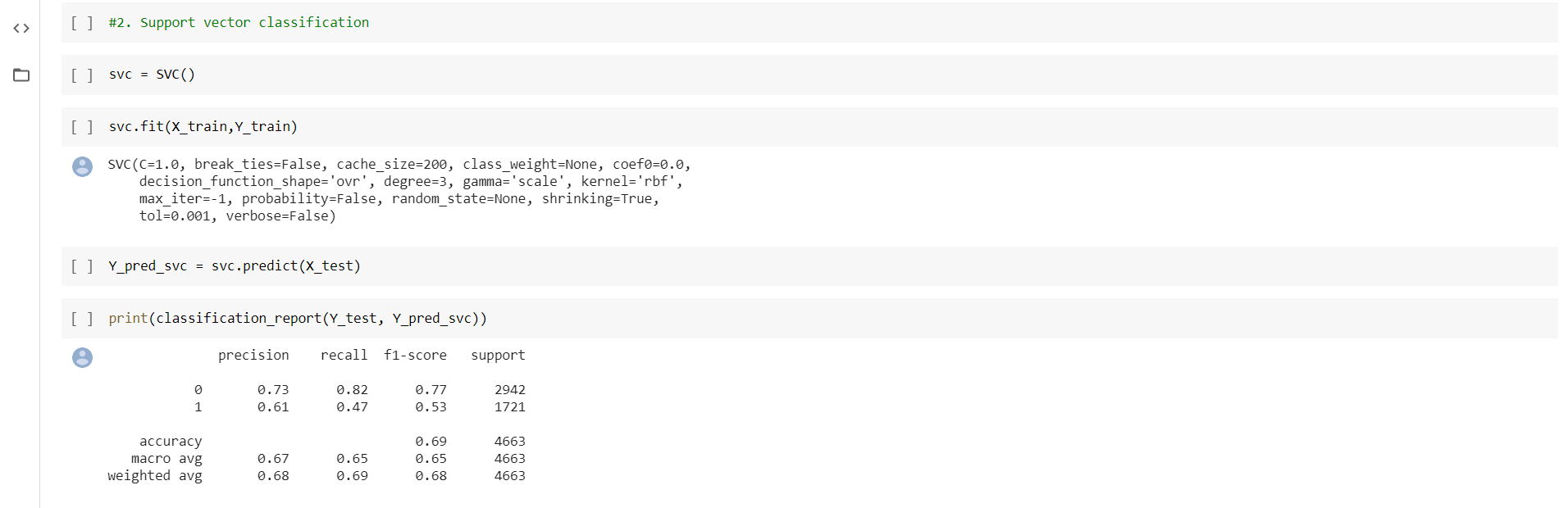


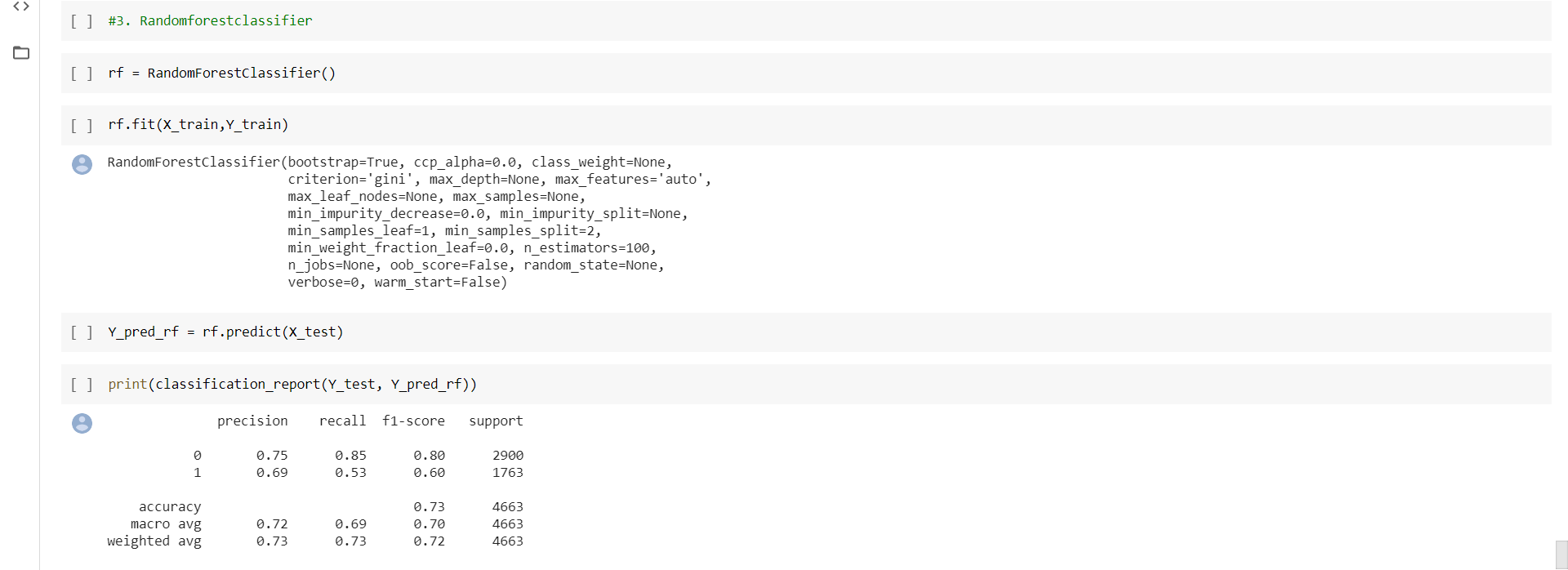
Train the Models, The models are Logistic Regression, Support Vector Classification, Random Forest and XG Boost.

1)Logistic Regression



2) Support Vector Classification(SVC)



3) Random Forest Classification

4) XG Boost

**Comparison of all the 4 models:**

|  |  |
| --- | --- |
| Model Name | Accuracy(%) |
| 1. Logistic Regression | 67 |
| 1. Support Vector Classifier | 69 |
| 1. Random Forest | 73 |
| 1. XgBoost | 74.7 |

**Conclusion:**

We would like to conclude our work here because, we think that the accuracy range of this dataset lies in between 70-80 percent only. We have not done hyper parameter tuning . If we do that, maybe we can achieve a little push in the accuracy. We believe that the obtained accuracy Is decent because there were many missing values that too in the categorical variables.

**Future Work:**

Try other models as MLP classifier

1. Try other packages as Keras and Pytorch to implement MLP classifier
2. Do some more feature transformations (feature engineering)
3. Try Hyper Parameter Tuning on the models