INTELLIGENT CUSTOMER RETENTION: USING MACHINE LEARNING FOR ENHANCED PREDICTION OF TELECOM CUSTOMER CHURN

SUBMITTED BY:

RAMANATHAN N (TEAM LEADER)

KUMARABUBRAMANI M.M

AKILAN G

JEYENDRAN D

1.INTRODUCTION

1.1 OVERVIEW

1.2 PURPOSE

1. INTRODUCTION:

1.1 OVERVIEW:

Intelligent customer retention is an important aspect of any business, and machine learning techniques can be used to enhance prediction of customer churn in the telecom industry. Telecom companies face high levels of customer churn, which can result in significant revenue losses. Predicting which customers are likely to churn can help companies take proactive measures to retain these customers and prevent revenue losses.

Machine learning techniques can be used to analyze customer data and identify patterns that indicate a customer is likely to churn. These patterns may include things like the number of calls made, the amount of data used, the length of time between recharges, and the type of plan the customer is on. By analyzing these patterns, machine learning algorithms can create predictive models that identify customers who are at risk of churning.

Once these at-risk customers have been identified, telecom companies can take proactive measures to retain them. This might include targeted marketing campaigns, special offers, or personalized customer service. By retaining these customers, telecom companies can reduce churn rates and increase revenue.

In addition to enhancing customer retention, machine learning techniques can also help telecom companies improve customer experience. By analyzing customer data, machine learning algorithms can identify areas where customers may be experiencing problems or frustrations with the service. This information can then be used to make improvements to the service, resulting in a better overall customer experience.

Overall, the use of machine learning techniques for intelligent customer retention in the telecom industry can lead to increased revenue, improved customer retention rates, and a better overall customer experience.

1.1.1 DATA COLLECTION:

ML depends heavily on data. It is the most crucial aspect that makes algorithm training possible. So, this section allows you to download the required dataset.

COLLECT THE DATASET

There are many popular open sources for collecting the data. E.g.: kaggle.com, UCI repository, etc.

In this project we have used .csv data. This data is downloaded from kaggle.com. Please refer to the link given below to download the dataset.

Link: https://www.kaggle.com/shrutimechlearn/churn-modelling

As the dataset is downloaded. Let us read and understand the data properly with the help of some visualization techniques and some analyzing techniques.

Note: There are a number of techniques for understanding the data. But here we have used some of it. In an additional way, you can use multiple techniques.

Importing The Libraries:

Import the necessary libraries as shown in the image.

```
#import necessary libraries
import pandas as pd
import numpy as np
import pickle
import matplotlib.pyplot as plt
%matplotlib inline
import seaborn as sns
import sklearn
from sklearn.preprocessing import LabelEncoder, OneHotEncoder
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import  RandomForestClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.svm import SVC
from sklearn.model_selection import RandomizedSearchCV
import imblearn
from imblearn.over_sampling import SMOTE
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix, f1_score
```

Read The Dataset:

Our dataset format might be in .csv, excel files, .txt, .json, etc. We can read the dataset with the help of pandas.

In pandas we have a function called read_csv() to read the dataset. As a parameter we have to give the directory of the csv file.

```python
#import dataset
data = pd.read_csv(r"C:\Users\Shivani_SB\OneDrive\Desktop\Telecom churn modelling-updated\data\DataSet.csv")
data
```

	customerID	gender	SeniorCitizen	Partner	Dependents	tenure	PhoneService	MultipleLines	InternetService	OnlineSecurity	...	DeviceProtection	TechSupport	StreamingTV
0	7590-VHVEG	Female	0	Yes	No	1	No	No phone service	DSL	No	...	No	No	No
1	5575-GNVDE	Male	0	No	No	34	Yes	No	DSL	Yes	...	Yes	No	No
2	3668-QPYBK	Male	0	No	No	2	Yes	No	DSL	Yes	...	No	No	No
3	7795-CFOCW	Male	0	No	No	45	No	No phone service	DSL	Yes	...	Yes	Yes	No
4	9237-HQITU	Female	0	No	No	2	Yes	No	Fiber optic	No	...	No	No	No
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
7038	6840-RESVB	Male	0	Yes	Yes	24	Yes	Yes	DSL	Yes	...	Yes	Yes	Yes
7039	2234-XADUH	Female	0	Yes	Yes	72	Yes	Yes	Fiber optic	No	...	Yes	No	Yes
7040	4801-JZAZL	Female	0	Yes	Yes	11	No	No phone service	DSL	Yes	...	No	No	No

Data Preparation:

As we have understood how the data is, let's pre-process the collected data.

The download data set is not suitable for training the machine learning model as it might have so much randomness so we need to clean the dataset properly in order to fetch good results. This activity includes the following steps.

Handling missing values

Handling categorical data

Handling Imbalance Data

Note: These are the general steps of pre-processing the data before using it for machine learning. Depending on the condition of your dataset, you may or may not have to go through all these steps.

Handling Missing Values:

Let’s find the shape of our dataset first. To find the shape of our data, the df.shape method is used. To find the data type, df.info() function is used.

```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 7043 entries, 0 to 7042
Data columns (total 20 columns):
 #   Column            Non-Null Count  Dtype  
---  ------            --------------  -----  
 0   gender            7043 non-null   object 
 1   SeniorCitizen     7043 non-null   int64  
 2   Partner           7043 non-null   object 
 3   Dependents        7043 non-null   object 
 4   tenure            7043 non-null   int64  
 5   PhoneService      7043 non-null   object 
 6   MultipleLines     7043 non-null   object 
 7   InternetService   7043 non-null   object 
 8   OnlineSecurity    7043 non-null   object 
 9   OnlineBackup      7043 non-null   object 
 10  DeviceProtection  7043 non-null   object 
 11  TechSupport       7043 non-null   object 
 12  StreamingTV       7043 non-null   object 
 13  StreamingMovies   7043 non-null   object 
 14  Contract          7043 non-null   object 
 15  PaperlessBilling  7043 non-null   object 
 16  PaymentMethod     7043 non-null   object 
 17  MonthlyCharges    7043 non-null   float64
 18  TotalCharges      7043 non-null   object 
 19  Churn             7043 non-null   object 
dtypes: float64(1), int64(2), object(17)
memory usage: 1.1+ MB
```

- For checking the null values, df.isnull() function is used. To sum those null values we use .sum() function. From the below image we found that there are no null values present in our dataset. So we can skip handling the missing values step.

```python
#checking for null values
data.TotalCharges = pd.to_numeric(data.TotalCharges, errors='coerce')
data.isnull().any()
```

```
gender              False
SeniorCitizen       False
Partner             False
Dependents          False
tenure              False
PhoneService        False
MultipleLines       False
InternetService     False
OnlineSecurity      False
OnlineBackup        False
DeviceProtection    False
TechSupport         False
StreamingTV         False
StreamingMovies     False
Contract            False
PaperlessBilling    False
PaymentMethod       False
MonthlyCharges      False
TotalCharges         True
Churn               False
dtype: bool
```

- From the above code of analysis, we can infer that column TotalCharges is having the missing values, we need to treat them in a required way.

```python
data["TotalCharges"].fillna(data["TotalCharges"].median() , inplace =True)
```

```python
data.isnull().sum()
```

```
gender              0
SeniorCitizen       0
Partner             0
Dependents          0
tenure              0
PhoneService        0
MultipleLines       0
InternetService     0
OnlineSecurity      0
OnlineBackup        0
DeviceProtection    0
TechSupport         0
StreamingTV         0
StreamingMovies     0
Contract            0
PaperlessBilling    0
PaymentMethod       0
MonthlyCharges      0
TotalCharges        0
Churn               0
dtype: int64
```

We will fill in the missing values in the Total Charges column by median as it's a numerical column and then again, we checked for null values to see if there is any null value left.

Handling Categorical Values:

As we can see our dataset has categorical data we must convert the categorical data to integer encoding or binary encoding.

To convert the categorical features into numerical features we use encoding techniques. There are several techniques but in our project we are using manual encoding with the help of list comprehension.

Label Encoding

Label Encoding refers to converting the labels into numeric form so as to convert it into the machine-readable form. Machine learning algorithms can then decide in a better way on how those labels must be operated. It is an important pre-processing step for the structured dataset in supervised learning.

```
from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()
data["gender"] = le.fit_transform(data["gender"])
data["Partner"] = le.fit_transform(data["Partner"])
data["Dependents"] = le.fit_transform(data["Dependents"])
data["PhoneService"] = le.fit_transform(data["PhoneService"])
data["MultipleLines"] = le.fit_transform(data["MultipleLines"])
data["InternetService"] = le.fit_transform(data["InternetService"])
data["OnlineSecurity"] = le.fit_transform(data["OnlineSecurity"])
data["OnlineBackup"] = le.fit_transform(data["OnlineBackup"])
data["DeviceProtection"] = le.fit_transform(data["DeviceProtection"])
data["TechSupport"] = le.fit_transform(data["TechSupport"])
data["StreamingTV"] = le.fit_transform(data["StreamingTV"])
data["StreamingMovies"] = le.fit_transform(data["StreamingMovies"])
data["Contract"] = le.fit_transform(data["Contract"])
data["PaperlessBilling"] = le.fit_transform(data["PaperlessBilling"])
data["PaymentMethod"] = le.fit_transform(data["PaymentMethod"])
data["Churn"] = le.fit_transform(data["Churn"])
```

Data after label encoding

```
data.head()
```

	gender	SeniorCitizen	Partner	Dependents	tenure	PhoneService	MultipleLines	InternetService	OnlineSecurity	OnlineBa
0	0	0	1	0	1	0	1	0	0	2
1	1	0	0	0	34	1	0	0	2	0
2	1	0	0	0	2	1	0	0	2	2
3	1	0	0	0	45	0	1	0	2	0
4	0	0	0	0	2	1	0	1	0	0

All the data is converted into numerical values.

Splitting the Dataset into Dependent and Independent variable

Let's split our dataset into independent and dependent variables.

- The independent variable in the dataset would be considered as 'x' and gender, Senior Citizen, Partner, Dependents, tenure, Phone Service, Multiple Lines, Internet Service, Online Security, Online Backup, Device Protection, Tec Support, Streaming TV, Streaming Movies, Contract, Paperless Billing, Payment Method, Monthly Charges, Total Charges columns would be considered as independent variable.
- 2. The dependent variable in the dataset would be considered as 'y' and the 'Churn' column is considered as dependent variable.

Now we will split the data of independent variables,

```
x= data.iloc[:,0:19].values
y= data.iloc[:,19:20].values
```

From the above code **":"** indicates that you are considering all the rows in the dataset and **"0:18"** indicates that you are considering columns 0 to 8 such as sex, job and purpose as input values and assigning them to variable x. In the same way in second line **“:”** indicates you are considering all the rows and **“18:19”** indicates that you are considering only last column as output value and assigning them to variable y.

After splitting we see the data as below

x

```
x

array([[0.0000e+00, 0.0000e+00, 1.0000e+00, ..., 2.0000e+00, 2.9850e+01,
         2.9850e+01],
        [1.0000e+00, 0.0000e+00, 0.0000e+00, ..., 3.0000e+00, 5.6950e+01,
         1.8895e+03],
        [1.0000e+00, 0.0000e+00, 0.0000e+00, ..., 3.0000e+00, 5.3850e+01,
         1.0815e+02],
        ...,
        [0.0000e+00, 0.0000e+00, 1.0000e+00, ..., 2.0000e+00, 2.9600e+01,
         3.4645e+02],
        [1.0000e+00, 1.0000e+00, 1.0000e+00, ..., 3.0000e+00, 7.4400e+01,
         3.0660e+02],
        [1.0000e+00, 0.0000e+00, 0.0000e+00, ..., 0.0000e+00, 1.0565e+02,
         6.8445e+03]])
```

Y

```
y

array([[0],
       [0],
       [1],
       ...,
       [0],
       [1],
       [0]], dtype=int64)
```

OneHot Encoding

Sometimes in datasets, we encounter columns that contain numbers of no specific order of preference. The data in the column usually denotes a category or value of the category and also when the data in the column is label encoded. This confuses the machine learning model, to avoid this, the data in the column should be One Hot encoded.

One Hot Encoding –

It refers to splitting the column which contains numerical categorical data to many columns depending on the number of categories present in that column. Each column contains “0” or “1” corresponding to which column it has been placed.

```
from sklearn.preprocessing import OneHotEncoder
one = OneHotEncoder()
a= one.fit_transform(x[:,6:7]).toarray()
b= one.fit_transform(x[:,7:8]).toarray()
c= one.fit_transform(x[:,8:9]).toarray()
d= one.fit_transform(x[:,9:10]).toarray()
e= one.fit_transform(x[:,10:11]).toarray()
f= one.fit_transform(x[:,11:12]).toarray()
g= one.fit_transform(x[:,12:13]).toarray()
h= one.fit_transform(x[:,13:14]).toarray()
i= one.fit_transform(x[:,14:15]).toarray()
j= one.fit_transform(x[:,16:17]).toarray()
x=np.delete(x,[6,7,8,9,10,11,12,13,14,16],axis=1)
x=np.concatenate((a,b,c,d,e,f,g,h,i,j,x),axis=1)
```

Handling Imbalance Data:

Data Balancing is one of the most important step, which need to be performed for classification models, because when we train our model on imbalanced dataset ,we will get biassed results, which means our model is able to predict only one class element.

For Balancing the data we are using the SMOTE Method.

SMOTE: Synthetic minority over sampling technique, which will create new synthetic data points for under class as per the requirements given by us using KNN method.

```
from imblearn.over_sampling import SMOTE
```

```
smt = SMOTE()

x_resample, y_resample = smt.fit_resample(x,y)
```

```
x_resample
```

```
array([[0.00000000e+00, 0.00000000e+00, 1.00000000e+00, ...,
        2.00000000e+00, 2.98500000e+01, 2.98500000e+01],
       [1.00000000e+00, 0.00000000e+00, 0.00000000e+00, ...,
        3.00000000e+00, 5.69500000e+01, 1.88950000e+03],
       [1.00000000e+00, 0.00000000e+00, 0.00000000e+00, ...,
        3.00000000e+00, 5.38500000e+01, 1.08150000e+02],
       ...,
       [0.00000000e+00, 0.00000000e+00, 0.00000000e+00, ...,
        3.00000000e+00, 2.02307905e+01, 2.02307905e+01],
       [1.00000000e+00, 0.00000000e+00, 6.76069757e-01, ...,
        3.23930243e-01, 9.00059277e+01, 3.69766940e+03],
       [0.00000000e+00, 3.89455378e-01, 1.00000000e+00, ...,
        2.00000000e+00, 9.63258517e+01, 3.21144455e+03]])
```

```
y_resample
```

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

```
x.shape, x_resample.shape
```

```
((7043, 19), (10348, 19))
```

```
y.shape, y_resample.shape
```

```
((7043, 1), (10348,))
```

From the above picture, we can infer that, previously our dataset had 492 class 1, and 192 class items, after applying smote technique on the dataset the size has been changed for minority class.

1.1.2 EXPLORATORY DATA ANALYSIS:

In this milestone, we will see the exploratory data analysis.

Descriptive Statistical

Descriptive analysis is to study the basic features of data with the statistical process. Here pandas has a worthy function called describe. With this describe function we can understand the unique, top and frequent values of categorical features. And we can find mean, std, min, max and percentile values of continuous features.

```
data.describe()
```

	SeniorCitizen	tenure	MonthlyCharges
count	7043.000000	7043.000000	7043.000000
mean	0.162147	32.371149	64.761692
std	0.368612	24.559481	30.090047
min	0.000000	0.000000	18.250000
25%	0.000000	9.000000	35.500000
50%	0.000000	29.000000	70.350000
75%	0.000000	55.000000	89.850000
max	1.000000	72.000000	118.750000

Visual Analysis:

Visual analysis is the process of using visual representations, such as charts, plots, and graphs, to explore and understand data. It is a way to quickly identify patterns, trends, and outliers in the data, which can help to gain insights and make informed decisions.

Univariate Analysis:

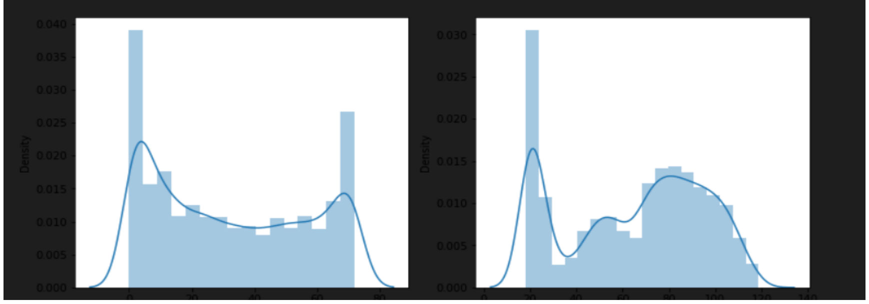
In simple words, univariate analysis is understanding the data with a single feature. Here we have displayed two different graphs such as distplot and countplot.

The Seaborn package provides a wonderful function distplot. With the help of distplot, we can find the distribution of the feature. To make multiple graphs in a single plot, we use subplot.

```
plt.figure(figsize=(12,5))
plt.subplot(1,2,1)
sns.distplot(data["tenure"])
plt.subplot(1,2,2)
sns.distplot(data["MonthlyCharges"])
```

```
C:\Users\Shivani_SB\anaconda3\lib\site-packages\seaborn\distributions.py:2619: FutureWarning: `distpl
your code to use either `displot` (a figure-level function with similar flexibility) or `histplot` (a
  warnings.warn(msg, FutureWarning)
C:\Users\Shivani_SB\anaconda3\lib\site-packages\seaborn\distributions.py:2619: FutureWarning: `distpl
your code to use either `displot` (a figure-level function with similar flexibility) or `histplot` (a
  warnings.warn(msg, FutureWarning)

<AxesSubplot:xlabel='MonthlyCharges', ylabel='Density'>
```



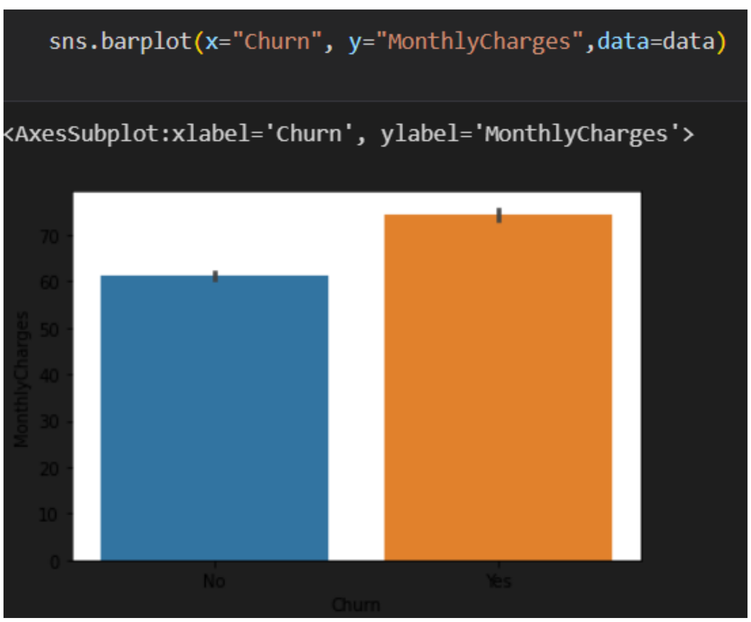
- In our dataset we have some categorical features. With the count plot function, we are going to count the unique category in those features. We have created a dummy data frame with categorical features. With for loop and subplot we have plotted this below graph.
- From the plot we came to know, Applicants income is skewed towards left side, where as credit history is categorical with 1.0 and 0.0

Countplot: **:-**

A count plot can be thought of as a histogram across a categorical, instead of quantitative, variable. The basic API and options are identical to those for barplot() , so you can compare counts across nested variables.

From the graph we can infer that , gender and education is a categorical variables with 2 categories , from gender column we can infer that 0-category is having more weightage than category-1,while education with 0,it means no education is a underclass when compared with category -1, which means educated .

Bivariate Analysis:

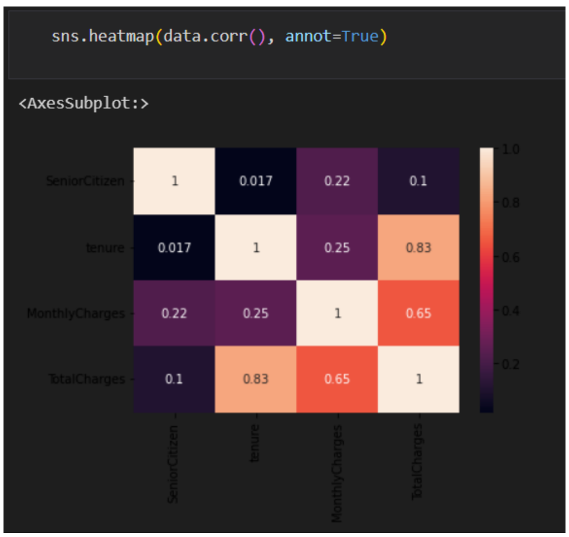


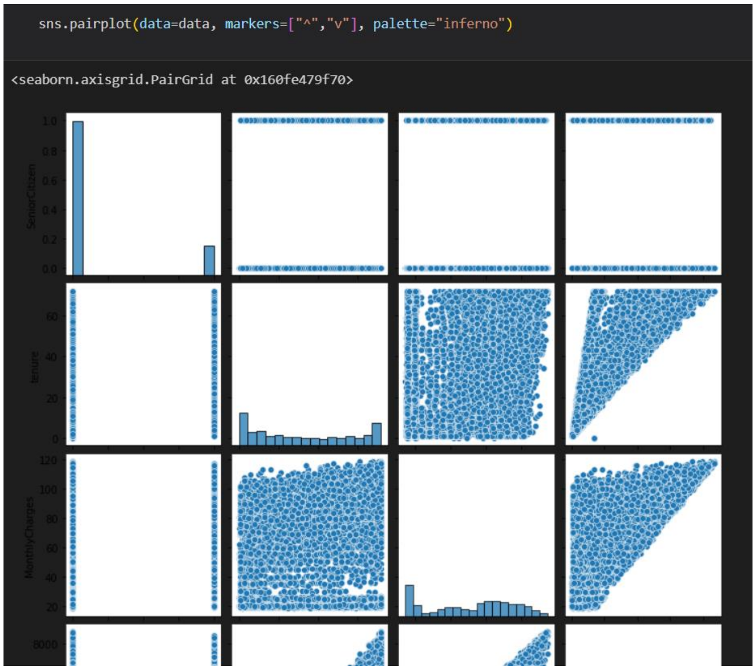
From the above graph we can infer the analysis such as

- Segmenting the gender column and married column based on bar graphs
- Segmenting the Education and Self-employed based on bar graphs ,for drawing insights such as educated people are employed.
- Loan amount term based on the property area of a person holding

Multivariate Analysis:

In simple words, multivariate analysis is to find the relation between multiple features. Here we have used a swarm plot from the seaborn package.





1.1.3 Model Building:

In this milestone, we will see the model building.

Training The Model In Multiple Algorithms:

Now our data is cleaned and it's time to build the model. We can train our data on different algorithms. For this project we are applying four classification algorithms. The best model is saved based on its performance.

Logistic Regression Model:

Logistic regression estimates the probability of an event occurring, such as voted or didn't vote, based on a given dataset of independent variables. Since the outcome is a probability, the dependent variable is bounded between 0 and 1. In logistic regression, a logit transformation is applied on the odds—that is, the probability of success divided by the probability of failure.

```
#importing and building the Decision tree model
def logreg(x_train,x_test,y_train,y_test):
    lr = LogisticRegression(random_state=0)
    lr.fit(x_train,y_train)
    y_lr_tr = lr.predict(x_train)
    print(accuracy_score(y_lr_tr,y_train))
    yPred_lr = lr.predict(x_test)
    print(accuracy_score(yPred_lr,y_test))
    print("***Logistic Regression***")
    print("Confusion_Matrix")
    print(confusion_matrix(y_test,yPred_lr))
    print("Classification Report")
    print(classification_report(y_test,yPred_lr))
```

```
#printing the train accuracy and test accuracy respectively
logreg(x_train,x_test,y_train,y_test)
```

```
0.7734960135298381
0.7734299516908213
***Logistic Regression***
Confusion_Matrix
[[754 279]
 [190 847]]
Classification Report
              precision    recall  f1-score   support

           0       0.80      0.73      0.76      1033
           1       0.75      0.82      0.78      1037

    accuracy                           0.77      2070
   macro avg       0.78      0.77      0.77      2070
weighted avg       0.78      0.77      0.77      2070
```

Decision Tree Model:

A function named decisionTree is created and train and test data are passed as the parameters. Inside the function, DecisionTreeClassifier algorithm is initialised and training data is passed to the model with the .fit() function. Test data is predicted with .predict() function and saved in a new variable. For evaluating the model, a confusion matrix and classification report is done.

```
#importing and building the Decision tree model
def decisionTree(x_train,x_test,y_train,y_test):
    dtc = DecisionTreeClassifier(criterion="entropy",random_state=0)
    dtc.fit(x_train,y_train)
    y_dt_tr = dtc.predict(x_train)
    print(accuracy_score(y_dt_tr,y_train))
    yPred_dt = dtc.predict(x_test)
    print(accuracy_score(yPred_dt,y_test))
    print("***Decision Tree***")
    print("Confusion_Matrix")
    print(confusion_matrix(y_test,yPred_dt))
    print("Classification Report")
    print(classification_report(y_test,yPred_dt))
```

```
#printing the train accuracy and test accuracy respectively
decisionTree(x_train,x_test,y_train,y_test)
```

```
0.9981879681082387
0.6067632850241546
***Decision Tree***
Confusion_Matrix
[[ 242  791]
 [  23 1014]]
Classification Report
              precision    recall  f1-score   support

           0       0.91      0.23      0.37      1033
           1       0.56      0.98      0.71      1037

    accuracy                           0.61      2070
   macro avg       0.74      0.61      0.54      2070
```

Random Forest Model:

A function named randomForest is created and train and test data are passed as the parameters. Inside the function, RandomForestClassifier algorithm is initialised and training data is passed to the model with .fit() function. Test data is predicted with .predict() function and saved in a new variable. For evaluating the model, a confusion matrix and classification report is done.

```python
#importing and building the random forest model
def RandomForest(x_tarin,x_test,y_train,y_test):
    rf = RandomForestClassifier(criterion="entropy",n_estimators=10,random_state=0)
    rf.fit(x_train,y_train)
    y_rf_tr = rf.predict(x_train)
    print(accuracy_score(y_rf_tr,y_train))
    yPred_rf = rf.predict(x_test)
    print(accuracy_score(yPred_rf,y_test))
    print("***Random Forest***")
    print("Confusion_Matrix")
    print(confusion_matrix(y_test,yPred_rf))
    print("Classification Report")
    print(classification_report(y_test,yPred_rf))
```

```python
#printing the train accuracy and test accuracy respectively
RandomForest(x_train,x_test,y_train,y_test)
```

```
0.9886446001449626
0.7536231884057971
***Random Forest***
Confusion_Matrix
[[563 470]
 [ 40 997]]
Classification Report
              precision    recall  f1-score   support

           0       0.93      0.55      0.69      1033
           1       0.68      0.96      0.80      1037

    accuracy                           0.75      2070
   macro avg       0.81      0.75      0.74      2070
weighted avg       0.81      0.75      0.74      2070
```

KNN Model:

A function named KNN is created and train and test data are passed as the parameters. Inside the function, KNeighborsClassifier algorithm is initialised and training data is passed to the model with .fit() function. Test data is predicted with .predict() function and saved in new variable. For evaluating the model, confusion matrix and classification report is done.

```python
#importing and building the KNN model
def KNN(x_train,x_test,y_train,y_test):
    knn = KNeighborsClassifier()
    knn.fit(x_train,y_train)
    y_knn_tr = knn.predict(x_train)
    print(accuracy_score(y_knn_tr,y_train))
    yPred_knn = knn.predict(x_test)
    print(accuracy_score(yPred_knn,y_test))
    print("***KNN***")
    print("Confusion_Matrix")
    print(confusion_matrix(y_test,yPred_knn))
    print("Classification Report")
    print(classification_report(y_test,yPred_knn))
```

```python
#printing the train accuracy and test accuracy respectively
KNN(x_train,x_test,y_train,y_test)
```

```
0.8570910848030925
0.7913043478260869
***KNN***
Confusion_Matrix
[[730 303]
 [129 908]]
Classification Report
              precision    recall  f1-score   support

           0       0.85      0.71      0.77      1033
           1       0.75      0.88      0.81      1037

    accuracy                           0.79      2070
   macro avg       0.80      0.79      0.79      2070
weighted avg       0.80      0.79      0.79      2070
```

SVM Model:

“Support Vector Machine” (SVM) is a supervised machine learning algorithm that can be used for both classification or regression challenges. However, it is mostly used in classification problems. In the SVM algorithm, we plot each data item as a point in n-dimensional space (where n is a number of features you have) with the value of each feature being the value of a particular coordinate.

```python
#importing and building the random forest model
def svm(x_tarin,x_test,y_train,y_test):
    svm = SVC(kernel = "linear")
    svm.fit(x_train,y_train)
    y_svm_tr = svm.predict(x_train)
    print(accuracy_score(y_svm_tr,y_train))
    yPred_svm = svm.predict(x_test)
    print(accuracy_score(yPred_svm,y_test))
    print("***Support Vector Machine***")
    print("Confusion_Matrix")
    print(confusion_matrix(y_test,yPred_svm))
    print("Classification Report")
    print(classification_report(y_test,yPred_svm))
```

```python
#printing the train accuracy and test accuracy respectively
svm(x_train,x_test,y_train,y_test)
```

```
0.7628654264315052
0.7555555555555555
***Support Vector Machine***
Confusion_Matrix
[[719 314]
 [192 845]]
Classification Report
              precision    recall  f1-score   support

           0       0.79      0.70      0.74      1033
           1       0.73      0.81      0.77      1037

    accuracy                           0.76      2070
   macro avg       0.76      0.76      0.75      2070
weighted avg       0.76      0.76      0.75      2070
```

ANN Model:

Building and training an Artificial Neural Network (ANN) using the Keras library with TensorFlow as the backend. The ANN is initialised as an instance of the Sequential class, which is a linear stack of layers. Then, the input layer and two hidden layers are added to the model using the Dense class, where the number of units and activation function are specified. The output layer is also added using the Dense class with a sigmoid activation function. The model is then compiled with the Adam optimizer, binary cross-entropy loss function, and accuracy metric. Finally, the model is fit to the training data with a batch size of 100, 20% validation split, and 100 epochs.

ANN Model

```
[ ] # Importing the Keras libraries and packages
    import keras
    from keras.models import Sequential
    from keras.layers import Dense
```

```
[ ] # Initialising the ANN
    classifier = Sequential()
```

```
[ ] # Adding the input layer and the first hidden layer
    classifier.add(Dense(units=30, activation='relu', input_dim=40))
```

```
[ ] # Adding the second hidden layer
    classifier.add(Dense(units=30, activation='relu'))
```

```
[ ] # Adding the output layer
    classifier.add(Dense(units=1, activation='sigmoid'))
```

```
[ ] # Compiling the ANN
    classifier.compile(optimizer='adam', loss='binary_crossentropy', metrics=['accuracy'])
```

```
# Fitting the ANN to the Training set
model_history = classifier.fit(x_train, y_train, batch_size=10, validation_split=0.33, epochs=200)
```

```
Epoch 1/200
555/555 [==============================] - 4s 3ms/step - loss: 0.5017 - accuracy: 0.7494 - val_loss: 0.4688 - val_accuracy: 0.7756
Epoch 2/200
555/555 [==============================] - 2s 3ms/step - loss: 0.4535 - accuracy: 0.7815 - val_loss: 0.4627 - val_accuracy: 0.7782
Epoch 3/200
555/555 [==============================] - 1s 3ms/step - loss: 0.4424 - accuracy: 0.7865 - val_loss: 0.4691 - val_accuracy: 0.7778
Epoch 4/200
555/555 [==============================] - 1s 2ms/step - loss: 0.4325 - accuracy: 0.7950 - val_loss: 0.4541 - val_accuracy: 0.7917
Epoch 5/200
555/555 [==============================] - 1s 2ms/step - loss: 0.4239 - accuracy: 0.8002 - val_loss: 0.4536 - val_accuracy: 0.7892
Epoch 6/200
555/555 [==============================] - 1s 3ms/step - loss: 0.4146 - accuracy: 0.8078 - val_loss: 0.4564 - val_accuracy: 0.7936
Epoch 7/200
555/555 [==============================] - 1s 2ms/step - loss: 0.4058 - accuracy: 0.8100 - val_loss: 0.4551 - val_accuracy: 0.7921
Epoch 8/200
555/555 [==============================] - 1s 2ms/step - loss: 0.3999 - accuracy: 0.8150 - val_loss: 0.4510 - val_accuracy: 0.7943
```

```
Epoch 195/200
555/555 [==============================] - 2s 3ms/step - loss: 0.1564 - accuracy: 0.9335 - val_loss: 0.7783 - val_accuracy: 0.8093
Epoch 196/200
555/555 [==============================] - 2s 3ms/step - loss: 0.1514 - accuracy: 0.9347 - val_loss: 0.7982 - val_accuracy: 0.7994
Epoch 197/200
555/555 [==============================] - 2s 3ms/step - loss: 0.1549 - accuracy: 0.9327 - val_loss: 0.8319 - val_accuracy: 0.7917
Epoch 198/200
555/555 [==============================] - 2s 3ms/step - loss: 0.1593 - accuracy: 0.9320 - val_loss: 0.7693 - val_accuracy: 0.8130
Epoch 199/200
555/555 [==============================] - 2s 3ms/step - loss: 0.1535 - accuracy: 0.9362 - val_loss: 0.7646 - val_accuracy: 0.8089
Epoch 200/200
555/555 [==============================] - 1s 3ms/step - loss: 0.1544 - accuracy: 0.9356 - val_loss: 0.7744 - val_accuracy: 0.8115
```

```
ann_pred = classifier.predict(x_test)
ann_pred = (ann_pred>0.5)
ann_pred
```

```
65/65 [==============================] - 0s 2ms/step
array([[False],
       [False],
       [ True],
       ...,
       [False],
       [False],
       [False]])
```

```
print(accuracy_score(ann_pred,y_test))
print("***ANN Model***")
print("Confusion_Matrix")
print(confusion_matrix(y_test,ann_pred))
print("Classification Report")
print(classification_report(y_test,ann_pred))
```

```
0.8067632850241546
***ANN Model***
Confusion_Matrix
[[840 193]
 [207 830]]
Classification Report
              precision    recall  f1-score   support

           0       0.80      0.81      0.81      1033
           1       0.81      0.80      0.81      1037

    accuracy                           0.81      2070
   macro avg       0.81      0.81      0.81      2070
```

Testing The Model:

```
#testing on random input values
lr = LogisticRegression(random_state=0)
lr.fit(x_train,y_train)
print("Predicting on random input")
lr_pred_own = lr.predict(sc.transform([[0,0,1,1,0,0,0,0,1,0,0,1,0,0,1,0,0,1,0,0,1,0,0,1,0,0,1,0,1,0,0,1,1,0,0,456,1,0,3245,4567]]))
print("output is: ",lr_pred_own)
```

```
Predicting on random input
output is:  [0]
```

```
#testing on random input values
dtc = DecisionTreeClassifier(criterion="entropy",random_state=0)
dtc.fit(x_train,y_train)
print("Predicting on random input")
dtc_pred_own = dtc.predict(sc.transform([[0,0,1,1,0,0,0,0,1,0,0,1,0,0,1,0,0,1,0,0,1,0,0,1,0,0,1,0,1,0,0,1,1,0,0,456,1,0,3245,4567]]))
print("output is: ",dtc_pred_own)
```

```
Predicting on random input
output is:  [0]
```

```
#testing on random input values
rf = RandomForestClassifier(criterion="entropy",n_estimators=10,random_state=0)
rf.fit(x_train,y_train)
print("Predicting on random input")
rf_pred_own = rf.predict(sc.transform([[0,0,1,1,0,0,0,0,1,0,0,1,0,0,1,0,0,1,0,0,1,0,0,1,0,0,1,0,1,0,0,1,1,0,0,456,1,0,3245,4567]]))
print("output is: ",rf_pred_own)
```

```
Predicting on random input
output is:  [0]
```

```
#testing on random input values
svc = SVC(kernel = "linear")
svc.fit(x_train,y_train)
print("Predicting on random input")
svm_pred_own = svc.predict(sc.transform([[0,0,1,1,0,0,0,0,1,0,0,1,0,0,1,0,0,1,0,0,1,0,0,1,0,0,1,0,1,0,0,1,1,0,0,456,1,0,3245,4567]]))
print("output is: ",svm_pred_own)
```

```
Predicting on random input
output is:  [0]
```

```
#testing on random input values
knn = KNeighborsClassifier()
knn.fit(x_train,y_train)
print("Predicting on random input")
knn_pred_own = knn.predict(sc.transform([[0,0,1,1,0,0,0,0,1,0,0,1,0,0,1,0,0,1,0,0,1,0,0,1,0,0,1,0,1,0,0,1,1,0,0,456,1,0,3245,4567]]))
print("output is: ",knn_pred_own)
```

```
Predicting on random input
output is:  [0]
```

For ANN

```
#testing on random input values
print("Predicting on random input")
ann_pred_own = classifier.predict(sc.transform([[0,0,1,1,0,0,0,0,1,0,0,1,0,0,1,0,0,1,0,0,1,0,0,1,0,0,1,0,1,0,0,1,1,0,0,456,1,0,3245,4567]]))
print(ann_pred_own)
ann_pred_own = (ann_pred_own>0.5)
print("output is: ",ann_pred_own)
```

```
Predicting on random input
1/1 [==============================] - 0s 24ms/step
[[1.]]
output is:  [[ True]]
```

1.1.4 Performance Testing & Hyperparameter Tuning:

In this milestone, we will see the performance testing and hyperparameter tuning.

Testing Model With Multiple Evaluation Metrics:

Multiple evaluation metrics means evaluating the model's performance on a test set using different performance measures. This can provide a more comprehensive understanding of the model's strengths and weaknesses. We are using evaluation metrics for classification tasks including accuracy, precision, recall, support and F1-score.

Compare The Model:

For comparing the above four models, the compareModel function is defined.

```python
def compareModel(X_train,X_test,y_train,y_test):
    logreg(x_train,x_test,y_train,y_test)
    print('-'*100)
    decisionTree(X_train,X_test,y_train,y_test)
    print('-'*100)
    RandomForest(X_train,X_test,y_train,y_test)
    print('-'*100)
    svm(X_train,X_test,y_train,y_test)
    print('-'*100)
    KNN(X_train,X_test,y_train,y_test)
    print('-'*100)
```

```
compareModel(x_train,x_test,y_train,y_test)
```

```
0.7734960135298381
0.7734299516908213
***Logistic Regression***
Confusion_Matrix
[[754 279]
 [190 847]]
Classification Report
              precision    recall  f1-score   support

           0       0.80      0.73      0.76      1033
           1       0.75      0.82      0.78      1037

    accuracy                           0.77      2070
   macro avg       0.78      0.77      0.77      2070
weighted avg       0.78      0.77      0.77      2070
```

Comparing Model Accuracy Before & After Applying Hyperparameter Tuning:

Evaluating performance of the model From sklearn, cross_val_score is used to evaluate the score of the model. On the parameters, we have given rf (model name), x, y, cv (as 5 folds).

Note: To understand cross validation, refer to this link

```
y_rf = model.predict(x_train)
print(accuracy_score(y_rf,y_train))
yPred_rfcv = model.predict(x_test)
print(accuracy_score(yPred_rfcv,y_test))
print("***Random Forest after Hyperparameter tuning***")
print("Confusion_Matrix")
print(confusion_matrix(y_test,yPred_rfcv))
print("Classification Report")
print(classification_report(y_test,yPred_rfcv))
print("Predicting on random input")
rfcv_pred_own = model.predict(sc.transform([[0,0,1,1,0,0,0,0,1,0,0,1,0,0,1,0,0,1,0,0,1,0,0,1,0,0,1,0,1,0,0,1,1,0,0,456,1,0,3245,4567]]))
print("output is: ",rfcv_pred_own)
```

```
0.8859627929451558
0.7454106280193237
***Random Forest after Hyperparameter tuning***
Confusion_Matrix
[[553 480]
 [ 47 990]]
Classification Report
              precision    recall  f1-score   support

           0       0.92      0.54      0.68      1033
           1       0.67      0.95      0.79      1037

    accuracy                           0.75      2070
   macro avg       0.80      0.75      0.73      2070
weighted avg       0.80      0.75      0.73      2070
```

1.1.4 Model Deployment:

In this milestone, we will see the model deployment.

Save The Best Model:

Saving the best model after comparing its performance using different evaluation metrics means selecting the model with the highest performance and saving its weights and configuration. This can be useful in avoiding the need to retrain the model every time it is needed and also to be able to use it in the future.

```
classifier.save("telcom_churn.h5")
```

Integrate With Web Framework:

In this section, we will be building a web application that is integrated to the model we built. A UI is provided for the uses where he has to enter the values for predictions. The enter values are given to the saved model and prediction is showcased on the UI.

This section has the following tasks

- Building HTML Pages

- Building server side script
- Run the web application

Building Html Pages:

For this project create two HTML files namely

- base.html
- index.html
- predyes.html
- predno.html

and save them in the templates folder.

Build Python Code:

Import the libraries

```
from flask import Flask, render_template, request
import keras
from keras.models import  load_model
```

Load the saved model. Importing the flask module in the project is mandatory. An object of Flask class is our WSGI application. Flask constructor takes the name of the current module (__name__) as argument.

```
app = Flask(__name__)
model = load_model("telcom_churn.h5")
```

Render HTML page:

```python
@app.route('/') # rendering the html template
def home():
    return render_template('home.html')
```

Here we will be using a declared constructor to route to the HTML page which we have created earlier.

In the above example, ‘/’ URL is bound with the home.html function. Hence, when the home page of the web server is opened in the browser, the html page will be rendered. Whenever you enter the values from the html page the values can be retrieved using POST Method.

Retrieves the value from UI:

Here we are routing our app to predict() function. This function retrieves all the values from the HTML page using Post request. That is stored in an array. This array is passed to the model.predict() function. This function returns the prediction. And this prediction value will be rendered to the text that we have mentioned in the submit.html page earlier.

Main Function:

```python
@app.route('/')
def helloworld():
    return render_template("base.html")
@app.route('/assesment')
def prediction():
    return render_template("index.html")

@app.route('/predict', methods = ['POST'])
def admin():
    a= request.form["gender"]
    if (a == 'f'):
        a=0
    if (a == 'm'):
        a=1
    b= request.form["srcitizen"]
    if (b == 'n'):
        b=0
    if (b == 'y'):
        b=1
    c= request.form["partner"]
    if (c == 'n'):
        c=0
    if (c == 'y'):
        c=1
    d= request.form["dependents"]
    if (d == 'n'):
        d=0
    if (d == 'y'):
        d=1
    e= request.form["tenure"]
    f= request.form["phservices"]
    if (f == 'n'):
        f=0
    if (f == 'y'):
        f=1
    g= request.form["multi"]
    if (g == 'n'):
```

```
if (g == 'n'):
    g1,g2,g3=1,0,0
if (g == 'nps'):
    g1,g2,g3=0,1,0
if (g == 'y'):
    g1,g2,g3=0,0,1
h= request.form["is"]
if (h == 'dsl'):
    h1,h2,h3=1,0,0
if (h == 'fo'):
    h1,h2,h3=0,1,0
if (h == 'n'):
    h1,h2,h3=0,0,1
i= request.form["os"]
if (i == 'n'):
    i1,i2,i3=1,0,0
if (i == 'nis'):
    i1,i2,i3=0,1,0
if (i == 'y'):
    i1,i2,i3=0,0,1
j= request.form["ob"]
if (j == 'n'):
    j1,j2,j3=1,0,0
if (j == 'nis'):
    j1,j2,j3=0,1,0
if (j == 'y'):
    j1,j2,j3=0,0,1
k= request.form["dp"]
if (k == 'n'):
    k1,k2,k3=1,0,0
if (k == 'nis'):
    k1,k2,k3=0,1,0
if (k == 'y'):
    k1,k2,k3=0,0,1
l= request.form["ts"]
if (l == 'n'):
    l1,l2,l3=1,0,0
```

```
    l1,l2,l3=1,0,0
if (l == 'nis'):
    l1,l2,l3=0,1,0
if (l == 'y'):
    l1,l2,l3=0,0,1
m= request.form["stv"]
if (m == 'n'):
    m1,m2,m3=1,0,0
if (m == 'nis'):
    m1,m2,m3=0,1,0
if (m == 'y'):
    m1,m2,m3=0,0,1
n= request.form["smv"]
if (n == 'n'):
    n1,n2,n3=1,0,0
if (n == 'nis'):
    n1,n2,n3=0,1,0
if (n == 'y'):
    n1,n2,n3=0,0,1
o= request.form["contract"]
if (o == 'mtm'):
    o1,o2,o3=1,0,0
if (o == 'oyr'):
    o1,o2,o3=0,1,0
if (o == 'tyrs'):
    o1,o2,o3=0,0,1
p= request.form["pmt"]
if (p == 'ec'):
    p1,p2,p3,p4=1,0,0,0
if (p == 'mail'):
    p1,p2,p3,p4=0,1,0,0
if (p == 'bt'):
    p1,p2,p3,p4=0,0,1,0
if (p == 'cc'):
    p1,p2,p3,p4=0,0,0,1
q= request.form["plb"]
if (q == 'n'):
```

```
q= request.form["plb"]
if (q == 'n'):
    q=0
if (q == 'y'):
    q=1
r= request.form["mcharges"]
s= request.form["tcharges"]

t=[[int(g1),int(g2),int(g3),int(h1),int(h2),int(h3),int(i1),int(i2),int(i3),int(j1
print(t)
x = model.predict(t)
print(x[0])
if (x[[0]] <=0.5):
    y ="No"
    return render_template("predno.html", z = y)

if (x[[0]] >= 0.5):
    y ="Yes"
    return render_template("predyes.html", z = y)
```

Run The Web Application:

- Open anaconda prompt from the start menu
- Navigate to the folder where your python script is.
- Now type "python app.py" command
- Navigate to the localhost where you can view your web page.
- Click on the predict button from the top left corner, enter the inputs, click on the submit button, and see the result/prediction on the web.

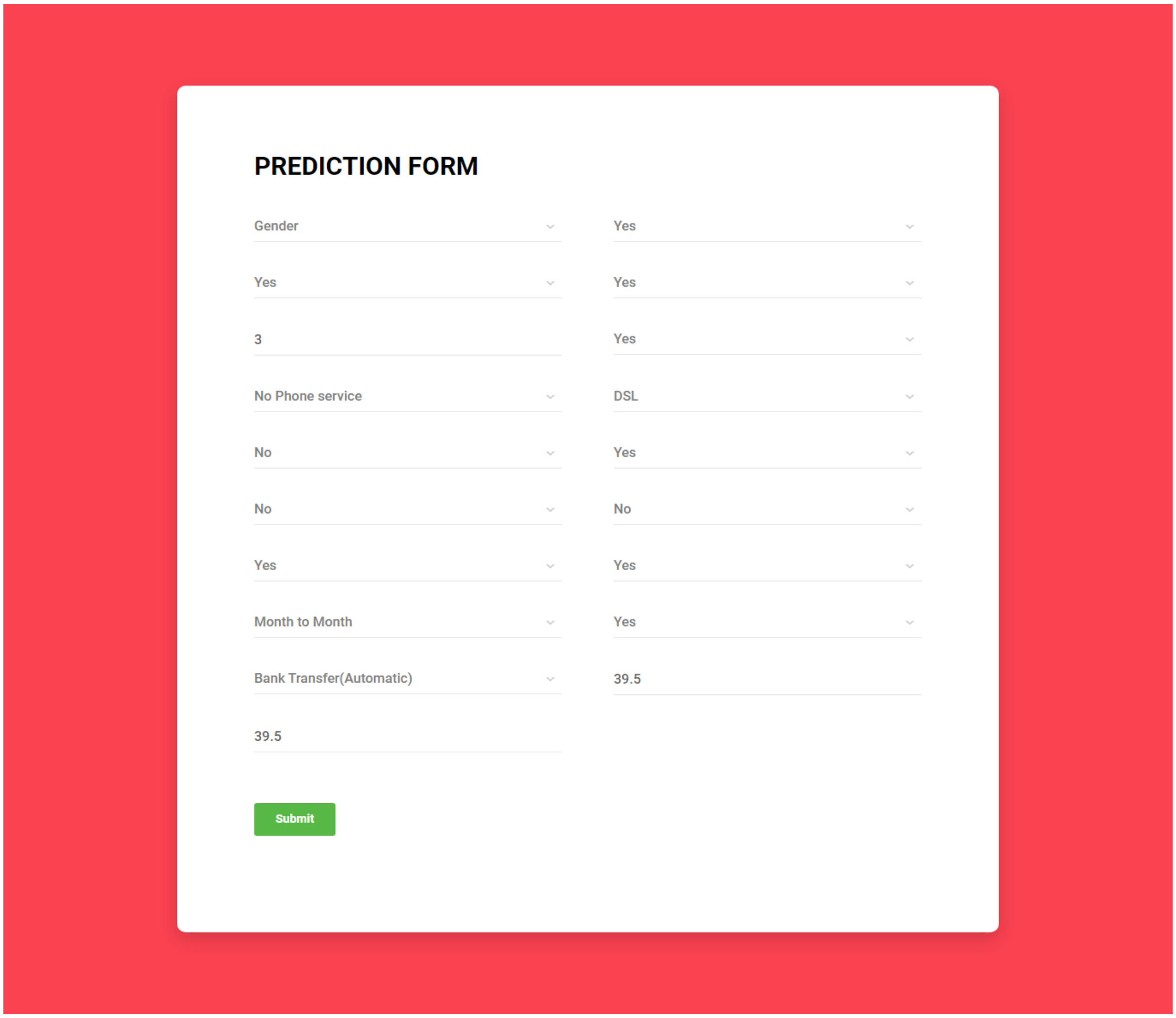
```
(base) C:\Users\Shivani_SB\OneDrive\Desktop\Telecom churn modelling-updated\flask app>python app.py
2023-01-26 00:46:27.532503: I tensorflow/core/platform/cpu_feature_guard.cc:193] This TensorFlow bir
h oneAPI Deep Neural Network Library (oneDNN) to use the following CPU instructions in performance-c
 AVX AVX2
To enable them in other operations, rebuild TensorFlow with the appropriate compiler flags.
 * Serving Flask app "app" (lazy loading)
 * Environment: production
   WARNING: This is a development server. Do not use it in a production deployment.
   Use a production WSGI server instead.
 * Debug mode: on
 * Restarting with watchdog (windowsapi)
2023-01-26 00:46:34.072445: I tensorflow/core/platform/cpu_feature_guard.cc:193] This TensorFlow bir
h oneAPI Deep Neural Network Library (oneDNN) to use the following CPU instructions in performance-c
 AVX AVX2
To enable them in other operations, rebuild TensorFlow with the appropriate compiler flags.
 * Debugger is active!
 * Debugger PIN: 109-979-709
 * Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)
```

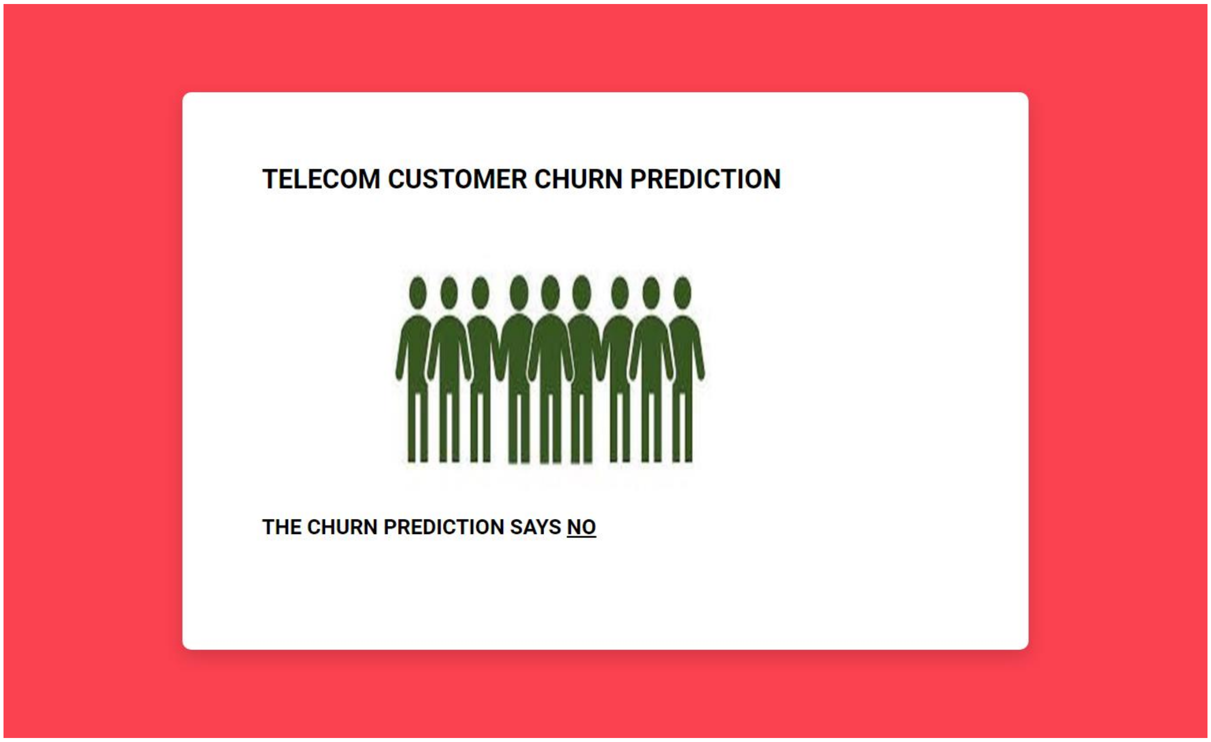
Now,Go the web browser and write the localhost url (http://127.0.0.1:5000) to get the below result

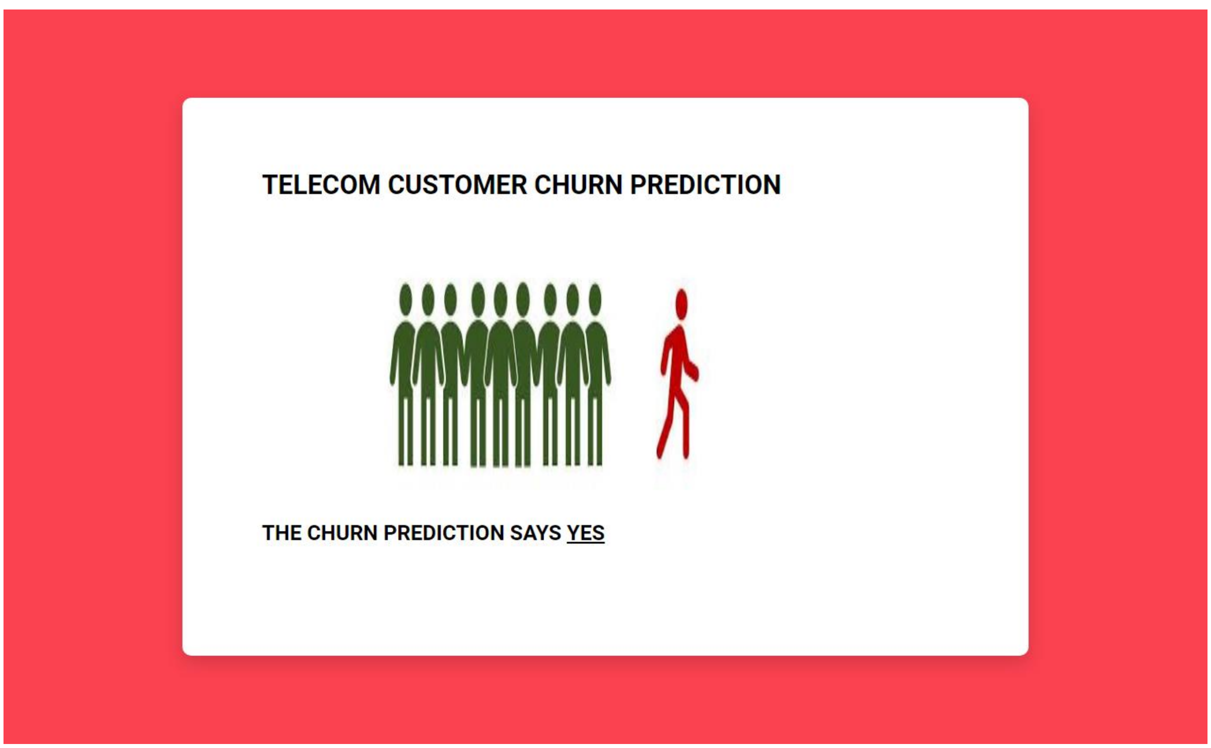
TELECOM CUSTOMER CHURN PREDICTION

Customer churn has become highly important for companies because of increasing competition among companies, increased importance of marketing strategies and conscious behaviour of customers in the recent years. Customers can easily trend toward alternative services. Companies must develop various strategies to prevent these possible trends, depending on the services they provide. During the estimation of possible churns, data from the previous churns might be used. An efficient churn predictive model benefits companies in many ways. Early identification of customers likely to leave may help to build cost effective ways in marketing strategies. Customer retention campaigns might be limited to selected customers but it should cover most of the customer. Incorrect predictions could result in a company losing profits because of the discounts offered to continuous subscribers.

Click me to continue with prediction


PREDICTION FORM
Gender
Yes
Yes
Yes
3
Yes
No Phone service
DSL
No
Yes
No
No
Yes
Yes
Month to Month
Yes
Bank Transfer(Automatic)
39.5
39.5
Submit


TELECOM CUSTOMER CHURN PREDICTION
THE CHURN PREDICTION SAYS NO


TELECOM CUSTOMER CHURN PREDICTION
THE CHURN PREDICTION SAYS YES

1.2 PURPOSE:

Customer churn is often referred to as customer attrition, or customer defection which is the rate at which the customers are lost. Customer churn is a major problem and one of the most important concerns for large companies. Due to the direct effect on the revenues of the companies, especially in the telecom field, companies are seeking to develop means to predict potential customer to churn. Looking at churn, different reasons trigger customers to terminate their contracts, for example better price offers, more interesting packages, bad service experiences or change of customers' personal situations.

Customer churn has become highly important for companies because of increasing competition among companies, increased importance of marketing strategies and conscious behavior of customers in the recent years. Customers can easily trend toward alternative services. Companies must develop various strategies to prevent these possible trends, depending on the services they provide. During the estimation of possible churns, data from the previous churns might be used. An efficient churn predictive model benefits companies in many ways. Early identification of customers likely to leave may help to build cost effective ways in marketing strategies. Customer retention campaigns might be limited to selected customers but it should cover most of the customer. Incorrect predictions could result in a company losing profits because of the discounts offered to continuous subscribers.

Telecommunication industry always suffers from a very high churn rates when one industry offers a better plan than the previous there is a high possibility of the customer churning from the present due to a better plan in such a scenario it is very difficult to avoid losses but through prediction, we can keep it to a minimal level.

Telecom companies often use customer churn as a key business metrics to predict the number of customers that will leave a telecom service provider. A machine learning model can be used to identity the probable churn customers and then makes the necessary business decisions.

2.DESIGN THINKING

2.1 TECHNICAL ARCHITECTURE

2.2 EMPATHY MAP

2.3BRAINSTROME

2. DESIGN THINKING:

2.1 TECHNICAL ARCHITECTURE:

Technical architecture refers to the overall design and structure of the hardware and software components of a system or application. It involves designing and implementing the underlying technical infrastructure that supports the application or system's functionality, including hardware, operating systems, network infrastructure, databases, and software components.

Technical architecture is concerned with defining the technical specifications and requirements of a system, as well as the interrelationships between the components that make up the system. This includes selecting appropriate technologies and platforms, designing system interfaces, and creating detailed technical specifications.

A well-designed technical architecture should be scalable, flexible, and maintainable. It should be able to handle the expected volume of traffic and data, as well as any potential growth in the future. It should also be adaptable to changing business needs and requirements, and easy to maintain and update over time.

Technical architecture is an important aspect of software development and is crucial for ensuring the successful delivery of complex systems and applications. It plays a key role in ensuring that the system or application is reliable, efficient, and secure, and meets the requirements of the end-users.

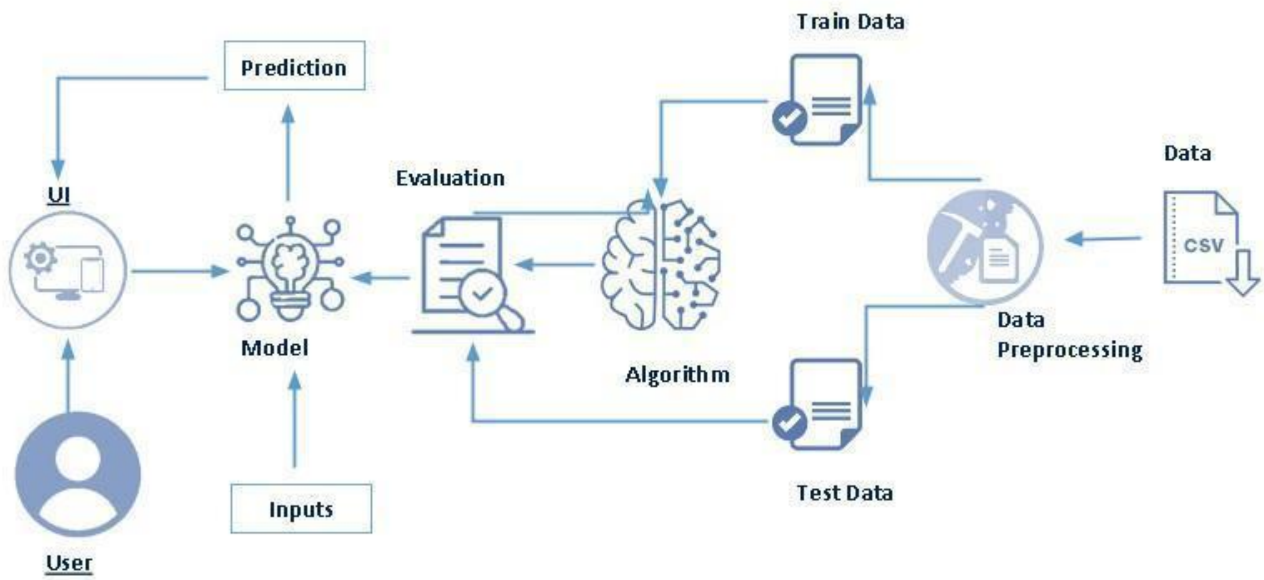


Fig: Technical Architecture

2.2 EMPATHY MAP:

An empathy map is a visual tool used in project management to gain a deeper understanding of the needs, behaviors, and motivations of stakeholders, such as customers or users. The empathy map is used to create a shared understanding among the project team and stakeholders about the people they are designing for.

The empathy map typically consists of four quadrants, which represent different aspects of the stakeholder's experience:

Think and Feel: This quadrant captures the stakeholder's emotions, attitudes, and beliefs related to the project or product.

See: This quadrant captures the stakeholder's physical environment, including what they see, hear, and touch.

Hear: This quadrant captures the stakeholder's perceptions of what other people say, such as feedback or comments from other stakeholders.

Do: This quadrant captures the stakeholder's actions, behaviors, and interactions related to the project or product.

The empathy map helps the project team to put themselves in the stakeholder's shoes and gain a deeper understanding of their needs, pain points, and desires. By creating a shared understanding of the stakeholder's experience, the project team can develop more effective strategies for addressing their needs and creating a better user experience.

The empathy map is a useful tool in agile and design thinking methodologies, as it helps to facilitate collaboration and communication among team members and stakeholders. It can be used at various stages of the project, from the initial planning and discovery phase to the testing and evaluation phase.

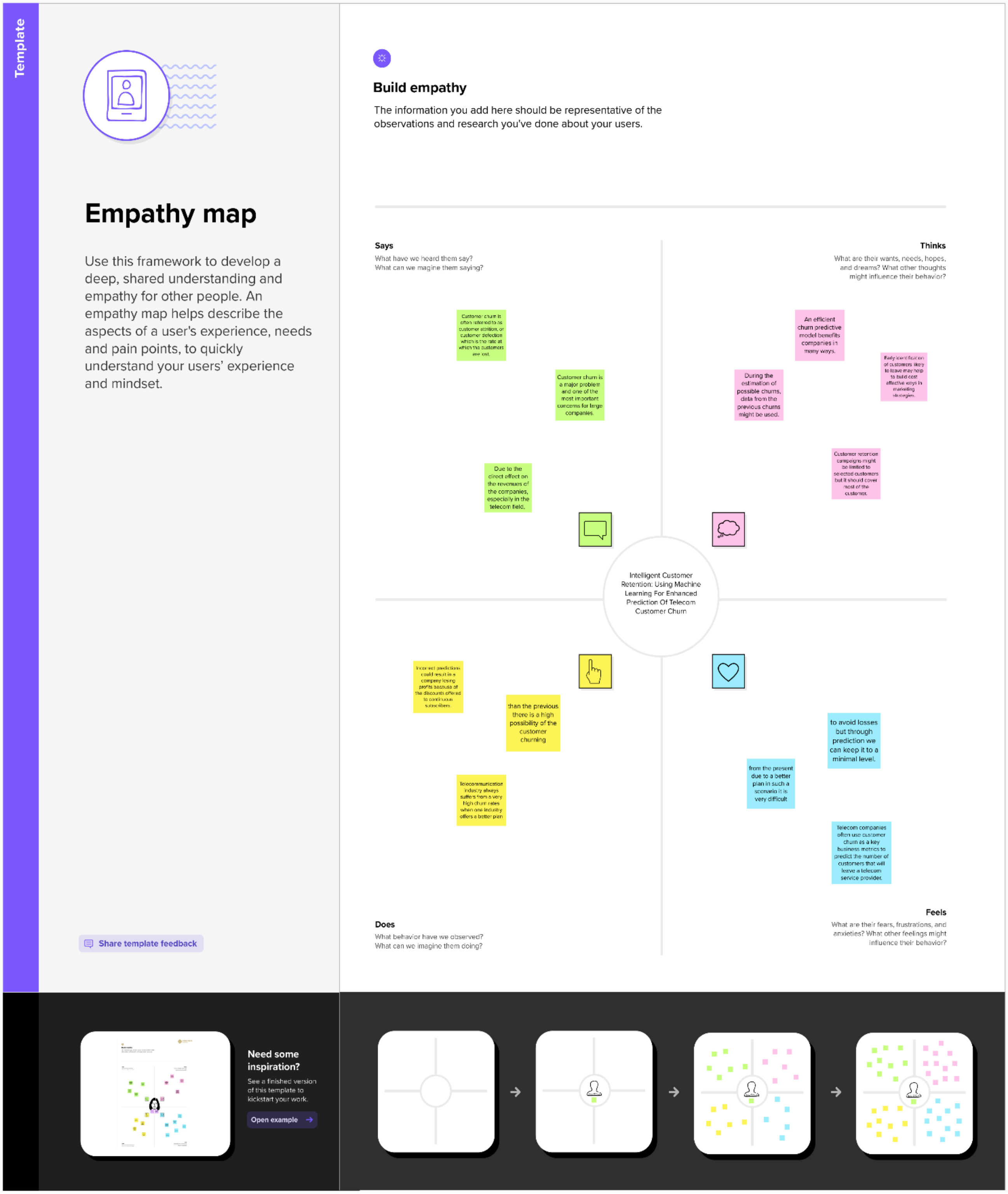


Fig: Empathy Map

2.3 BRAINSTORME:

Brainstorming is a group creativity technique used in project management to generate a large number of ideas or solutions to a problem. It involves bringing together a diverse group of people to freely and openly share their ideas and suggestions, without judgment or criticism. The goal of brainstorming is to encourage creative thinking, spark new ideas, and generate a broad range of potential solutions to a problem.

Brainstorming typically follows these steps:

1. Define the problem or challenge that needs to be addressed.
2. Form a diverse group of individuals with different backgrounds, experiences, and perspectives.
3. Set a clear goal or objective for the brainstorming session.
4. Encourage participants to freely and openly share their ideas and suggestions.
5. Record all ideas and suggestions, without criticism or judgment.
6. Review and refine the ideas generated during the brainstorming session.
7. Select the most promising ideas and develop them further.
8. Implement the selected ideas and monitor their effectiveness.

Brainstorming is a useful technique for generating new ideas, improving project outcomes, and fostering collaboration and teamwork. It is often used in agile and design thinking methodologies to promote innovation and creativity. By creating a supportive environment that encourages participation and idea sharing, brainstorming can help project teams to develop more effective solutions and improve project outcomes.

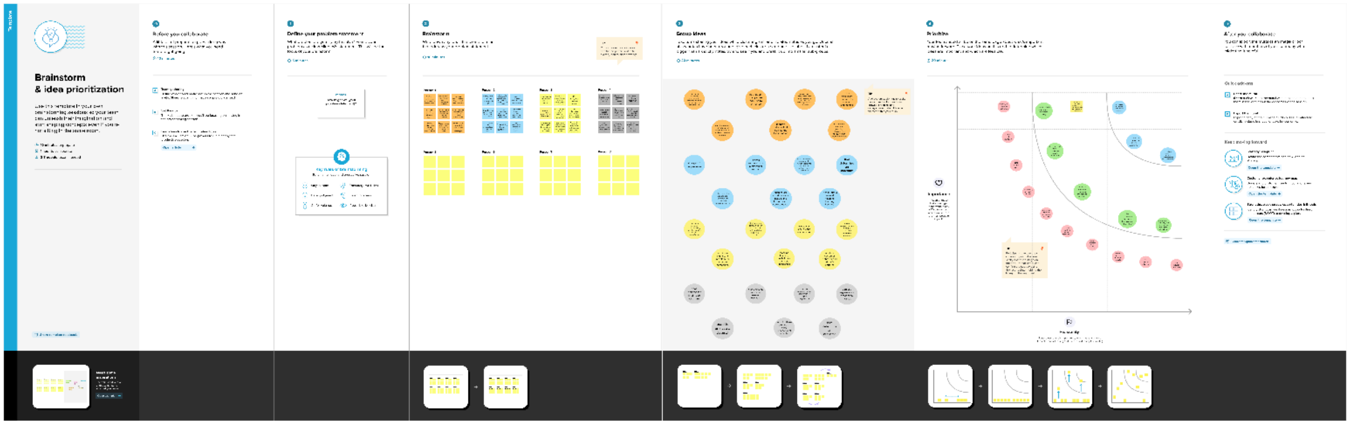
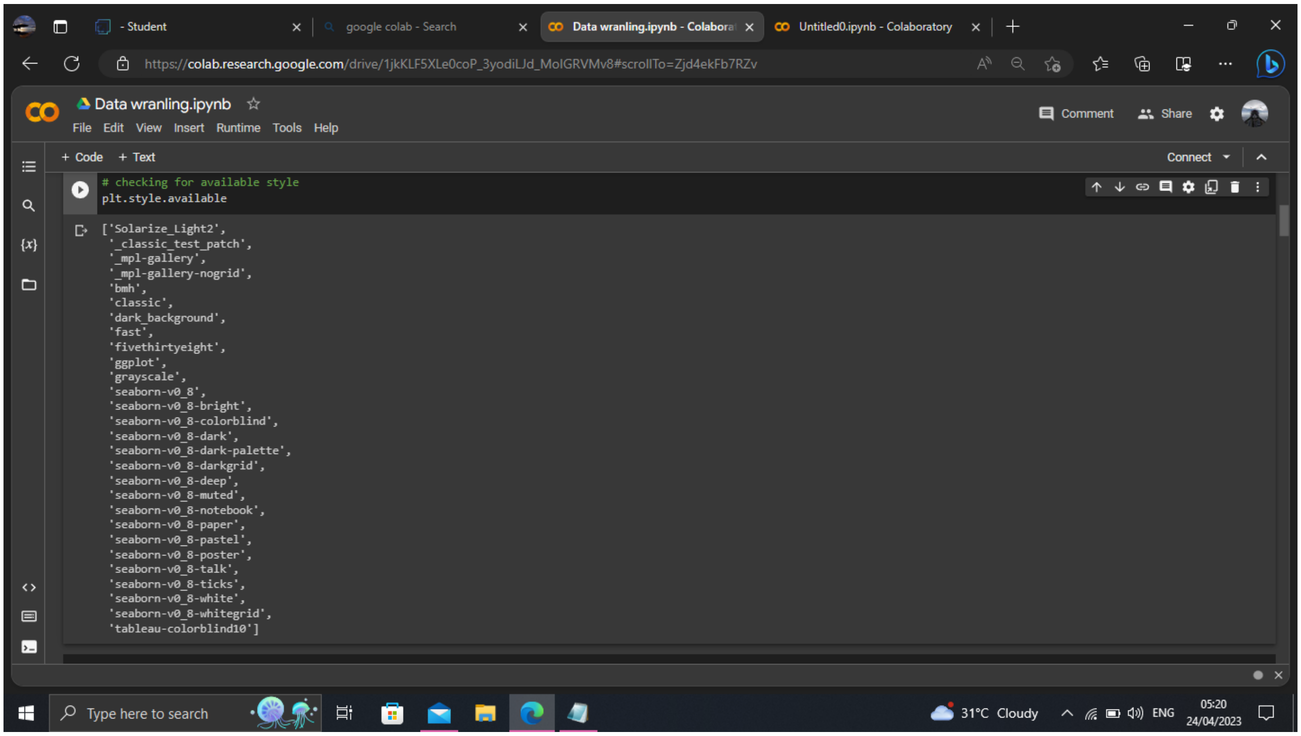


Fig: Brainstorm

3. RESULT



Data wranling.ipynb
File Edit View Insert Runtime Tools Help
+ Code + Text
# checking for available style
plt.style.available
['Solarize_Light2',
'_classic_test_patch',
'_mpl-gallery',
'_mpl-gallery-nogrid',
'bmh',
'classic',
'dark_background',
'fast',
'fivethirtyeight',
'ggplot',
'grayscale',
'seaborn-v0_8',
'seaborn-v0_8-bright',
'seaborn-v0_8-colorblind',
'seaborn-v0_8-dark',
'seaborn-v0_8-dark-palette',
'seaborn-v0_8-darkgrid',
'seaborn-v0_8-deep',
'seaborn-v0_8-muted',
'seaborn-v0_8-notebook',
'seaborn-v0_8-paper',
'seaborn-v0_8-pastel',
'seaborn-v0_8-poster',
'seaborn-v0_8-talk',
'seaborn-v0_8-ticks',
'seaborn-v0_8-white',
'seaborn-v0_8-whitegrid',
'tableau-colorblind10']

```python
df.shape
```

```
(10000, 14)
```

```python
df.columns
```

```
Index(['RowNumber', 'CustomerId', 'Surname', 'CreditScore', 'Geography',
       'Gender', 'Age', 'Tenure', 'Balance', 'NumOfProducts', 'HasCrCard',
       'IsActiveMember', 'EstimatedSalary', 'Exited'],
      dtype='object')
```

```python
df.dtypes
```

```
RowNumber            int64
CustomerId           int64
Surname             object
CreditScore          int64
Geography           object
Gender              object
Age                  int64
Tenure               int64
Balance            float64
NumOfProducts        int64
HasCrCard            int64
IsActiveMember       int64
EstimatedSalary    float64
Exited               int64
dtype: object
```

```python
# Printing Unique Values of the categorical variables
print(df['Geography'].unique())
print(df['Gender'].unique())
print(df['NumOfProducts'].unique())
print(df['HasCrCard'].unique())
print(df['IsActiveMember'].unique())
```

```
['France' 'Spain' 'Germany']
['Female' 'Male']
[1 3 2 4]
[1 0]
[1 0]
```

```python
# Checking if there are null values or not
df.isnull().sum()
```

```
RowNumber          0
CustomerId         0
Surname            0
CreditScore        0
Geography          0
Gender             0
Age                0
Tenure             0
Balance            0
NumOfProducts      0
HasCrCard          0
IsActiveMember     0
EstimatedSalary    0
Exited             0
dtype: int64
```

```
[ ] df.describe()
```

	RowNumber	CustomerId	CreditScore	Age	Tenure	Balance	NumOfProducts	HasCrCard	IsActiveMember	EstimatedSalary	Exited
count	10000.00000	1.000000e+04	10000.000000	10000.000000	10000.000000	10000.000000	10000.000000	10000.00000	10000.000000	10000.000000	10000.000000
mean	5000.50000	1.569094e+07	650.528800	38.921800	5.012800	76485.889288	1.530200	0.70550	0.515100	100090.239881	0.203700
std	2886.89568	7.193619e+04	96.653299	10.487806	2.892174	62397.405202	0.581654	0.45584	0.499797	57510.492818	0.402769
min	1.00000	1.556570e+07	350.000000	18.000000	0.000000	0.000000	1.000000	0.00000	0.000000	11.580000	0.000000
25%	2500.75000	1.562853e+07	584.000000	32.000000	3.000000	0.000000	1.000000	0.00000	0.000000	51002.110000	0.000000
50%	5000.50000	1.569074e+07	652.000000	37.000000	5.000000	97198.540000	1.000000	1.00000	1.000000	100193.915000	0.000000
75%	7500.25000	1.575323e+07	718.000000	44.000000	7.000000	127644.240000	2.000000	1.00000	1.000000	149388.247500	0.000000
max	10000.00000	1.581569e+07	850.000000	92.000000	10.000000	250898.090000	4.000000	1.00000	1.000000	199992.480000	1.000000

```
[ ] df.head()
```

	RowNumber	CustomerId	Surname	CreditScore	Geography	Gender	Age	Tenure	Balance	NumOfProducts	HasCrCard	IsActiveMember	EstimatedSalary	Exited
0	1	15634602	Hargrave	619	France	Female	42	2	0.00	1	1	1	101348.88	1
1	2	15647311	Hill	608	Spain	Female	41	1	83807.86	1	0	1	112542.58	0
2	3	15619304	Onio	502	France	Female	42	8	159660.80	3	1	0	113931.57	1
3	4	15701354	Boni	699	France	Female	39	1	0.00	2	0	0	93826.63	0
4	5	15737888	Mitchell	850	Spain	Female	43	2	125510.82	1	1	1	79084.10	0

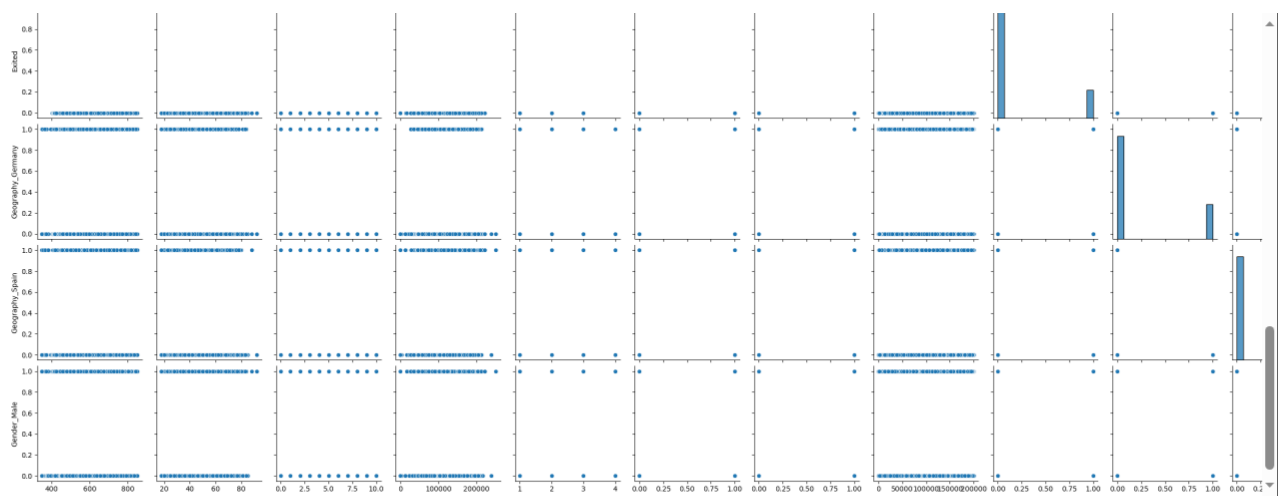
```
[ ] final_dataset.head()
```

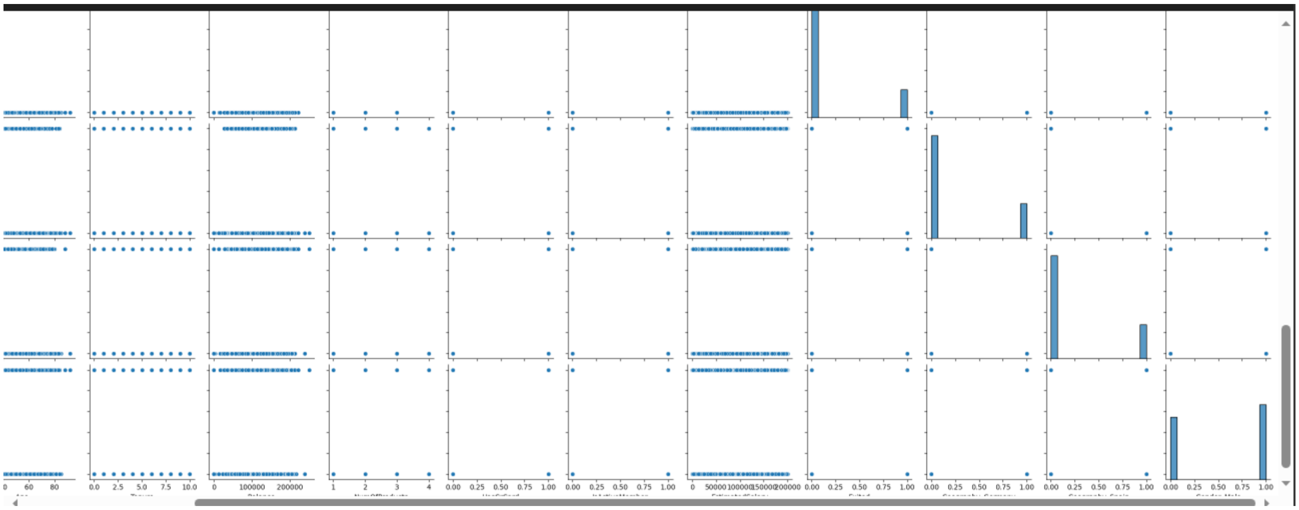
	CreditScore	Geography	Gender	Age	Tenure	Balance	NumOfProducts	HasCrCard	IsActiveMember	EstimatedSalary	Exited
0	619	France	Female	42	2	0.00	1	1	1	101348.88	1
1	608	Spain	Female	41	1	83807.86	1	0	1	112542.58	0
2	502	France	Female	42	8	159660.80	3	1	0	113931.57	1
3	699	France	Female	39	1	0.00	2	0	0	93826.63	0
4	850	Spain	Female	43	2	125510.82	1	1	1	79084.10	0

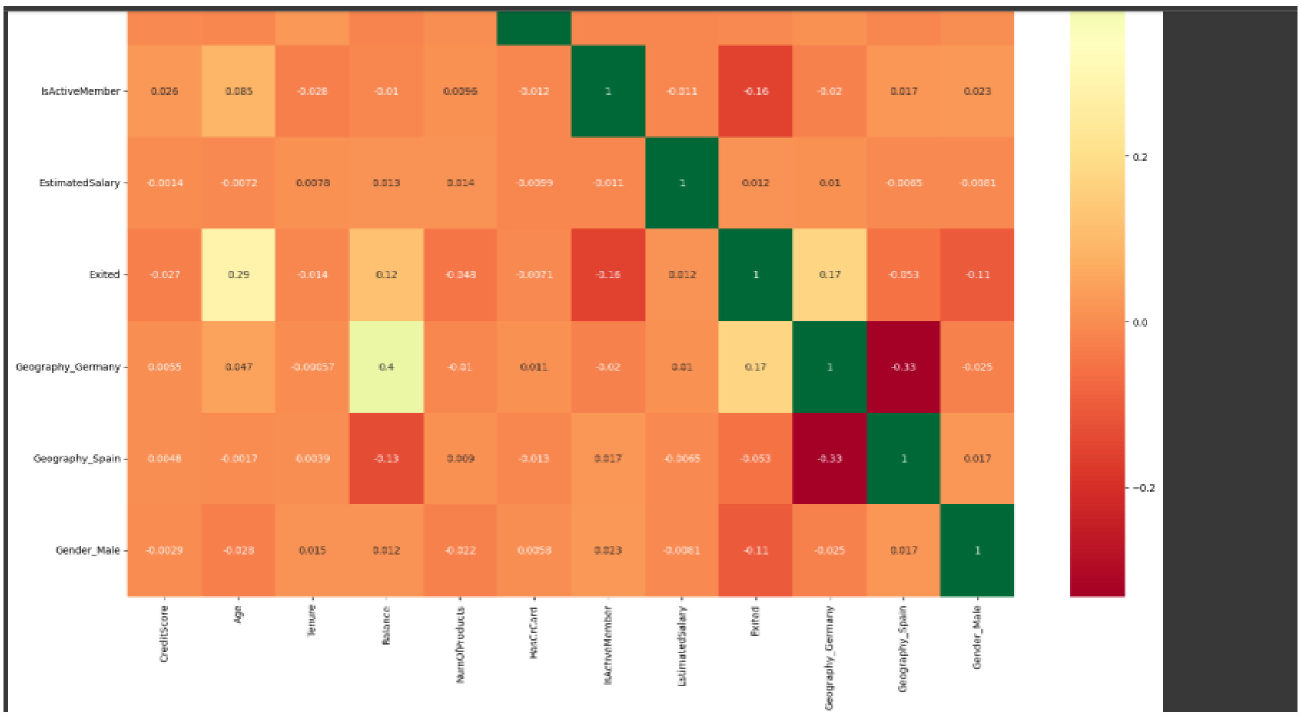
```
# Converting the categorical variables into numerical and avoiding Dummy Varibale Trap
final_dataset = pd.get_dummies(final_dataset, drop_first=True)
```

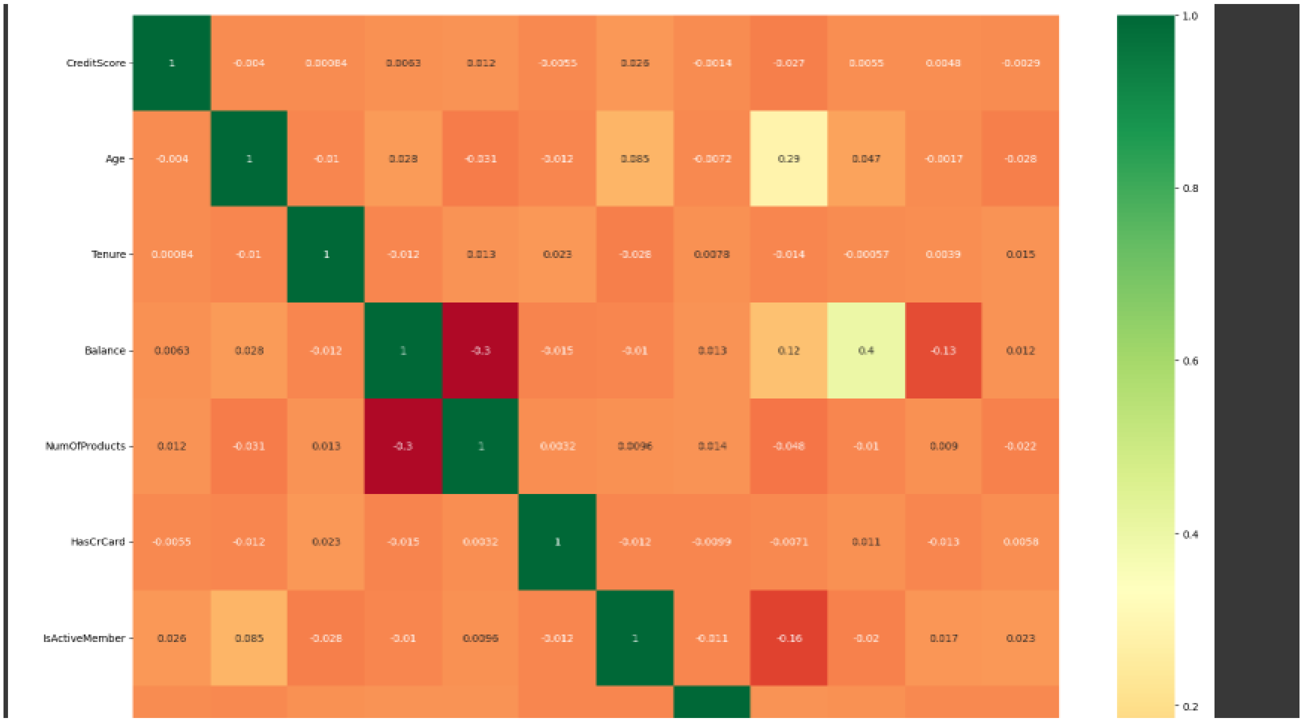
```
[ ] final_dataset.head()
```

	CreditScore	Age	Tenure	Balance	NumOfProducts	HasCrCard	IsActiveMember	EstimatedSalary	Exited	Geography_Germany	Geography_Spain	Gender_Male
0	619	42	2	0.00	1	1	1	101348.88	1	0	0	0
1	608	41	1	83807.86	1	0	1	112542.58	0	0	1	0
2	502	42	8	159660.80	3	1	0	113931.57	1	0	0	0
3	699	39	1	0.00	2	0	0	93826.63	0	0	0	0
4	850	43	2	125510.82	1	1	1	79084.10	0	0	1	0






IsActiveMember
EstimatedSalary
Exited
Geography_Germany
Geography_Spain
Gender_Male
CreditScore
Age
Tenure
Balance
NumOfProducts
HasCrCard
IsActiveMember
EstimatedSalary
Exited
Geography_Germany
Geography_Spain
Gender_Male




```
# Splitting the dataset into Training and Testing Data
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X,y,test_size=0.2, random_state = 42)
```

```
# Standardizing the Dataset
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
X_train = sc.fit_transform(X_train)
X_test = sc.transform(X_test)
```

```
print(X_train)
```

```
[[ 0.35649971 -0.6557859   0.34567966 ... -0.57946723 -0.57638802
   0.91324755]
 [-0.20389777  0.29493847 -0.3483691  ...  1.72572313 -0.57638802
   0.91324755]
 [-0.96147213 -1.41636539 -0.69539349 ... -0.57946723  1.73494238
   0.91324755]
 ...
 [ 0.86500853 -0.08535128 -1.38944225 ... -0.57946723 -0.57638802
  -1.09499335]
 [ 0.15932282  0.3900109   1.03972843 ... -0.57946723 -0.57638802
   0.91324755]
 [ 0.47065475  1.15059039 -1.38944225 ...  1.72572313 -0.57638802
   0.91324755]]
```

```
[0.12076443 0.2359372  0.10377652 0.12889567 0.14172989 0.03096833
 0.04739773 0.11802909 0.0286561  0.02159771 0.02224733]
```

```
from sklearn.ensemble import RandomForestClassifier
rf = RandomForestClassifier()
rf.fit(X_train,y_train)
```

```
▸ RandomForestClassifier
RandomForestClassifier()
```

```
y_pred = rf.predict(X_test)
```

```
from sklearn.metrics import accuracy_score, confusion_matrix
cm = confusion_matrix(y_test,y_pred)
print(cm)
print(accuracy_score(y_test,y_pred))
```

```
[[1545   62]
 [ 207  186]]
0.8655
```

```
# pickling the Model
import pickle
file = open('Customer_Churn_Prediction.pkl', 'wb')
pickle.dump(rf, file)
```

4. ADVANTAGE & DISADVANTAGE

ADVANTAGES:

Intelligent customer retention using machine learning for enhanced prediction of telecom customer churn offers several advantages, including:

Improved customer retention: Machine learning algorithms can analyze customer data to identify patterns and trends that indicate the likelihood of customer churn. By predicting which customers are at risk of leaving, telecom companies can proactively engage with them to address their concerns and improve their satisfaction, ultimately reducing churn rates.

Personalized customer experiences: Machine learning algorithms can analyze customer data to create personalized experiences and offers that meet their specific needs and preferences. This can help to improve customer loyalty and satisfaction, as well as drive revenue growth.

Cost savings: Acquiring new customers can be expensive, so reducing churn rates can be a cost-effective strategy for telecom companies. By identifying at-risk customers and implementing targeted retention strategies, companies can reduce the need to acquire new customers and save on marketing and advertising expenses.

Competitive advantage: Telecom companies that use machine learning to predict and reduce customer churn can gain a competitive advantage by providing better customer experiences and reducing customer turnover. This can help to differentiate their brand and increase customer loyalty and market share.

Data-driven insights: Machine learning algorithms can analyze large amounts of customer data to identify patterns and trends that may not be visible to the human eye. By using data-driven insights, telecom companies can make more informed business decisions and improve their overall performance.

Overall, intelligent customer retention using machine learning offers numerous benefits for telecom companies, including improved customer retention, personalized experiences, cost savings, competitive advantage, and data-driven insights.

DISADVANTAGES:

While there are several advantages of using intelligent customer retention using machine learning for enhanced prediction of telecom customer churn, there are also some potential disadvantages to consider:

Data privacy concerns: The use of machine learning algorithms requires large amounts of customer data, including personal and sensitive information. This can raise privacy concerns among customers, especially if they feel that their data is being used without their consent or for purposes they do not approve of.

Over-reliance on technology: While machine learning algorithms can provide valuable insights and predictions, they should not be the only factor considered when making retention decisions. Over-reliance on technology can lead to overlooking other important factors that may impact customer churn, such as customer satisfaction and loyalty.

Bias and accuracy issues: Machine learning algorithms may not always be accurate in predicting customer churn, and may also be biased based on the data used to train them. This can result in inaccurate predictions and potentially harm the customer relationship if the company takes action based on incorrect predictions.

Implementation and maintenance costs: Implementing machine learning algorithms can be costly, and ongoing maintenance and updates are necessary to ensure accurate predictions. This may require additional resources and investment from the company.

Limited scope: Machine learning algorithms can only analyze data that is available to them, which may not capture all factors that may influence customer churn. This may limit the effectiveness of the retention strategies developed based on the predictions.

In summary, while there are benefits to using machine learning for enhanced prediction of telecom customer churn, there are also potential disadvantages that must be considered, including data privacy concerns, over-reliance on technology, bias and accuracy issues, implementation and maintenance costs, and limited scope. Companies must carefully consider these factors when deciding whether to adopt machine learning algorithms for customer retention.

5. APPLICATIONS

APPLICATIONS:

Intelligent customer retention using machine learning for enhanced prediction of telecom customer churn has a wide range of potential applications, including:

Predicting customer churn: Machine learning algorithms can be used to analyze customer data to predict which customers are at risk of leaving. This can help telecom companies to take proactive steps to retain these customers, such as offering personalized incentives or improving their customer experience.

Improving customer satisfaction: By analyzing customer data, machine learning algorithms can identify areas where customer satisfaction is low and suggest strategies for improving it. This can help telecom companies to better meet the needs and preferences of their customers, leading to higher satisfaction levels and reduced churn rates.

Personalizing marketing and promotions: Machine learning algorithms can analyze customer data to create personalized marketing messages and promotions. This can help to increase engagement and loyalty among customers, as well as drive revenue growth.

Optimizing customer service: Machine learning algorithms can be used to analyze customer service interactions to identify areas where improvements can be made. This can help to improve the overall customer experience and reduce churn rates.

Analyzing customer feedback: Machine learning algorithms can be used to analyze customer feedback, such as reviews and social media posts, to identify trends and insights that can inform retention strategies.

Developing customer retention strategies: By analyzing customer data and predicting churn, machine learning algorithms can help telecom companies to develop targeted retention strategies that are tailored to the specific needs and preferences of their customers.

Overall, intelligent customer retention using machine learning has a wide range of potential applications that can help telecom companies to reduce churn rates, improve customer satisfaction, and drive revenue growth.

6. CONCLUSION

CONCLUSION:

The importance of churn prediction will help many companies, mainly in telecom industries, to have a profitable income and achieve good revenue. Customer churn prediction is the major issue in the Telecom Industry, and due to this, companies are trying to keep the existing ones from leaving rather than acquiring a new customer. Three tree-based algorithms were chosen because of their applicability and diversity in this type of application. By using Random Forest, XGBoost, and Logistic regression, we will get more accuracy comparing other algorithms. Here we are using the dataset of some customers about their service plan and checking the values of them and have a precise prediction, which will help to identify the customers who are going to migrate to other company services. By this, the Telecom Company can have a clear view and can provide them some exiting offers to stay in that service. The obtained results show that our proposed churn model produced better results and performed better by using machine learning techniques.Random Forest produced better accuracy among the various methods.

7. FUTURE SCOPE

FUTURE SCOPE:

The future scope of intelligent customer retention using machine learning for enhanced prediction of telecom customer churn is promising, with several potential developments on the horizon, including:

Improved accuracy: As machine learning algorithms continue to develop, their accuracy in predicting customer churn is likely to improve. This will enable telecom companies to more effectively target at-risk customers and develop retention strategies that are tailored to their specific needs.

Greater personalization: Machine learning algorithms are becoming increasingly sophisticated in their ability to analyze customer data and create personalized experiences. In the future, telecom companies may be able to use machine learning to offer even more personalized services and promotions, leading to higher customer satisfaction and loyalty.

Integration with other technologies: Machine learning algorithms may be integrated with other emerging technologies, such as 5G and the Internet of Things (IoT), to create even more personalized and engaging customer experiences.

Greater focus on customer experience: As customer experience continues to become a key differentiator for telecom companies, the use of machine learning algorithms for customer retention is likely to become even more prevalent. Companies that prioritize customer experience and use machine learning to improve it are likely to see higher customer satisfaction and loyalty.

Expansion to other industries: While intelligent customer retention using machine learning has primarily been used in the telecom industry, it has potential applications in other industries as well. Companies in industries such as retail, healthcare, and finance may be able to use machine learning to reduce customer churn and improve customer satisfaction.

8. APPENDIX (SOURCE CODE)

```
# Importing the essential Libraries
import pandas as pd
import numpy as np

# Reading the Dataset
df = pd.read_csv('Churn_Modelling.csv')
df.head()

df.shape

df.columns

df.dtypes

# Printing Unique Values of the categorical variables
print(df['Geography'].unique())
print(df['Gender'].unique())
print(df['NumOfProducts'].unique())
print(df['HasCrCard'].unique())
print(df['IsActiveMember'].unique())
```

```
# Checking if there are null values or not
df.isnull().sum()

df.describe()

df.head()

#Including only Potential Predictors as independent varibles
final_dataset = df[['CreditScore', 'Geography', 'Gender', 'Age', 'Tenure', 'Balance',
'NumOfProducts', 'HasCrCard', 'IsActiveMember', 'EstimatedSalary', 'Exit']]

final_dataset.head()

# Converting the categorical variables into numerical and avoiding Dummy Varibale
Trap
final_dataset = pd.get_dummies(final_dataset, drop_first=True)

final_dataset.head()

import seaborn as sns
```

```
sns.pairplot(final_dataset)


import matplotlib.pyplot as plt
%matplotlib inline

# Plotting The Correlations between all the features
corrmat = final_dataset.corr()
top_corr_features = corrmat.index
plt.figure(figsize=(20,20))
sns.heatmap(final_dataset[top_corr_features].corr(), annot=True, cmap='RdYlGn')

final_dataset.head()

# Splitting the Dataset into Dependent and Independent Variables
X = final_dataset.iloc[:, [0,1,2,3,4,5,6,7,9,10,11]]
y = final_dataset.iloc[:, 8].values

X.head()
```

```
y

# Splitting the dataset into Training and Testing Data
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X,y,test_size=0.2, random_state =
42)

# Standardizing the Dataset
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
X_train = sc.fit_transform(X_train)
X_test = sc.transform(X_test)

print(X_train)

## Feature Importance
from sklearn.ensemble import ExtraTreesRegressor
model = ExtraTreesRegressor()
model.fit(X,y)
```

```
print(model.feature_importances_)


from sklearn.ensemble import RandomForestClassifier

rf = RandomForestClassifier()

rf.fit(X_train,y_train)


y_pred = rf.predict(X_test)


from sklearn.metrics import accuracy_score, confusion_matrix

cm = confusion_matrix(y_test,y_pred)

print(cm)

print(accuracy_score(y_test,y_pred))


# pickling the Model

import pickle

file = open('Customer_Churn_Prediction.pkl', 'wb')

pickle.dump(rf, file)
```