# Intelligent Customer Retention: Using Machine Learning For Enhanced Prediction Of Telecom Customer Churn

Project report submitted to
**"Madurai Kamaraj University"**
in partial fulfillment of the requirements
for the award of the Degree of

**Bachelor of Science**
**in**
**Computer Science**

***Submitted by***


| | | |
|---|---|---|
| **TEAM ID** | : | **NM2023TMID31051** |
| **TEAM LEADER** | : | **M. HARI HARAN** |
| **TEAM MEMBERS** | : | **S. JENARTHAN** |
| | | **V. IDEEGAN** |
| | | **V. GUNA** |

*Under the Guidance of*

**Dr. B. UMADEVI M.Sc.,M.Phil.,Ph.D.,**
**(Assistant Professor, GAC, Melur)**

**GOVERNMENT ARTS COLLEGE**

**P.G & DEPARTMENT OF COMPUTER SCIENCE**

**MELUR – 625 106**

# INDEX

# 1. INTRODUCTION

## 1.1 OVERVIEW

Intelligent customer retention is a process that uses machine learning algorithms to analyze customer data and predict the likelihood of customer churn. In the telecom industry, customer churn refers to the percentage of customers who discontinue using a company's services during a given time period. By predicting which customers are likely tochurn, telecom companies can take proactive measures to retain these customers and reduce overall churn rates.

Machine learning algorithms are particularly useful in customer retention because they can analyze large amounts of customer data and identify patterns and trends that may not be immediately apparent to human analysts. Some common data points that are used to predict customer churn in the telecom industry include customer demographics, usage patterns, and customer service interactions.

One popular machine learning technique used for customer retention is predictive modeling. Predictive models use historical data to identify patterns and relationships thatcan be used to predict future outcomes. For example, a predictive model may analyze customer data from the past year to predict which customers are most likely to churn in the coming months.

Another machine learning technique that is commonly used in customer retention is clustering. Clustering algorithms group customers into segments based on similarities in their behavior or characteristics. This can help telecom companies identify specific groupsof customers that are at higher risk of churn and tailor retention efforts to their specific needs.

## 1.2 PURPOSE

Intelligent customer retention using machine learning for enhanced prediction of telecom customer churn can have several benefits for telecom companies, including:

- **Reduced Customer Churn:** By using machine learning models to predict customer churn, telecom companies can proactively identify customers who are at risk of leaving and take appropriate retention actions to prevent them from churning.

- **Increased Customer Satisfaction:** By understanding the needs and preferences of their customers, telecom companies can offer personalized services and tailored products that meet their specific needs, leading to increased customer satisfaction and loyalty.

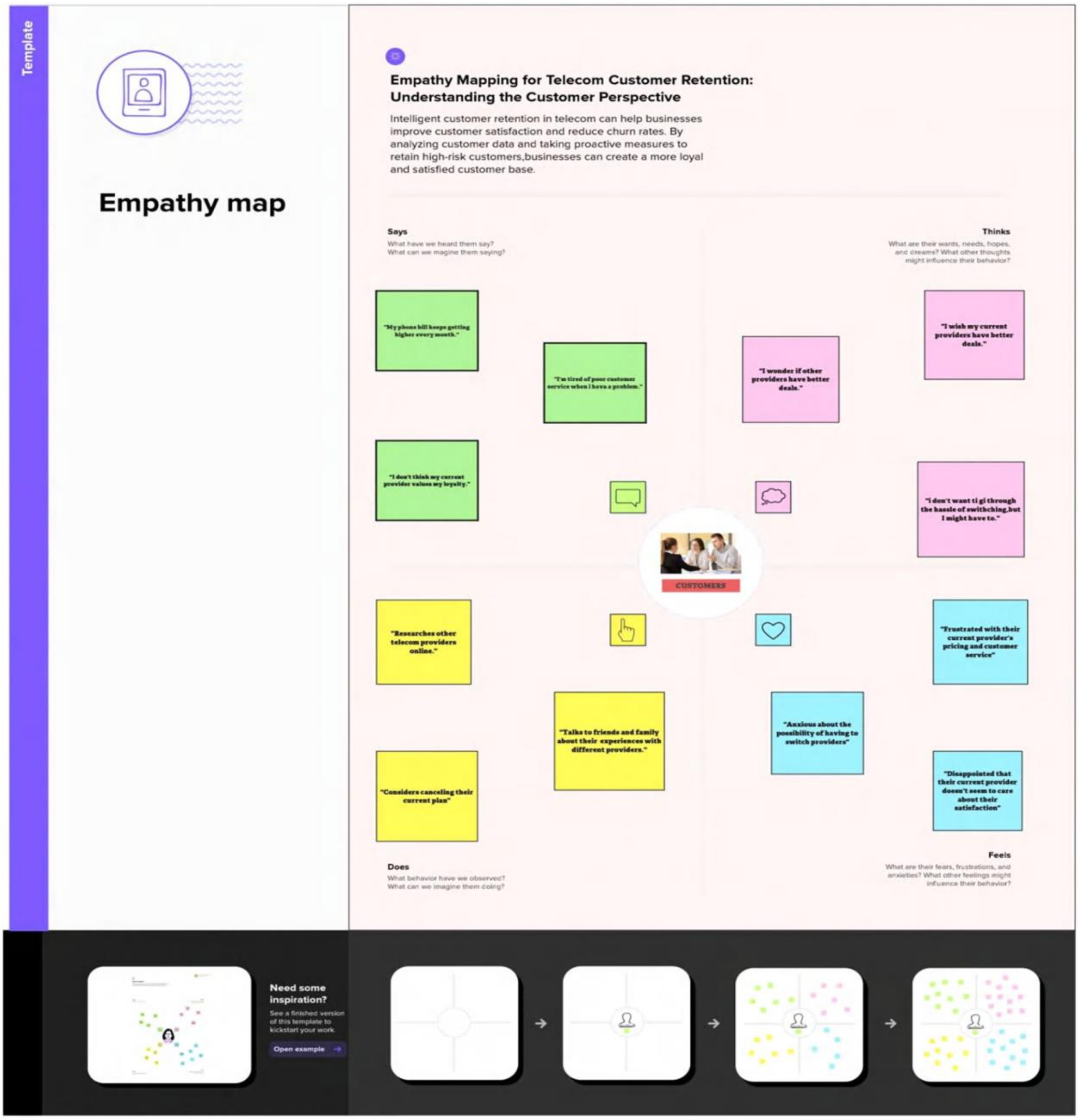
- **Improved Operational Efficiency:** By automating the process of customer churn prediction, telecom companies can save time and resources that would otherwise be spent on manual analysis and intervention.

- **Better Marketing Strategies:** Machine learning models can help telecom companies to better understand the characteristics of customers who are likely to churn and create targeted marketing campaigns to retain them.

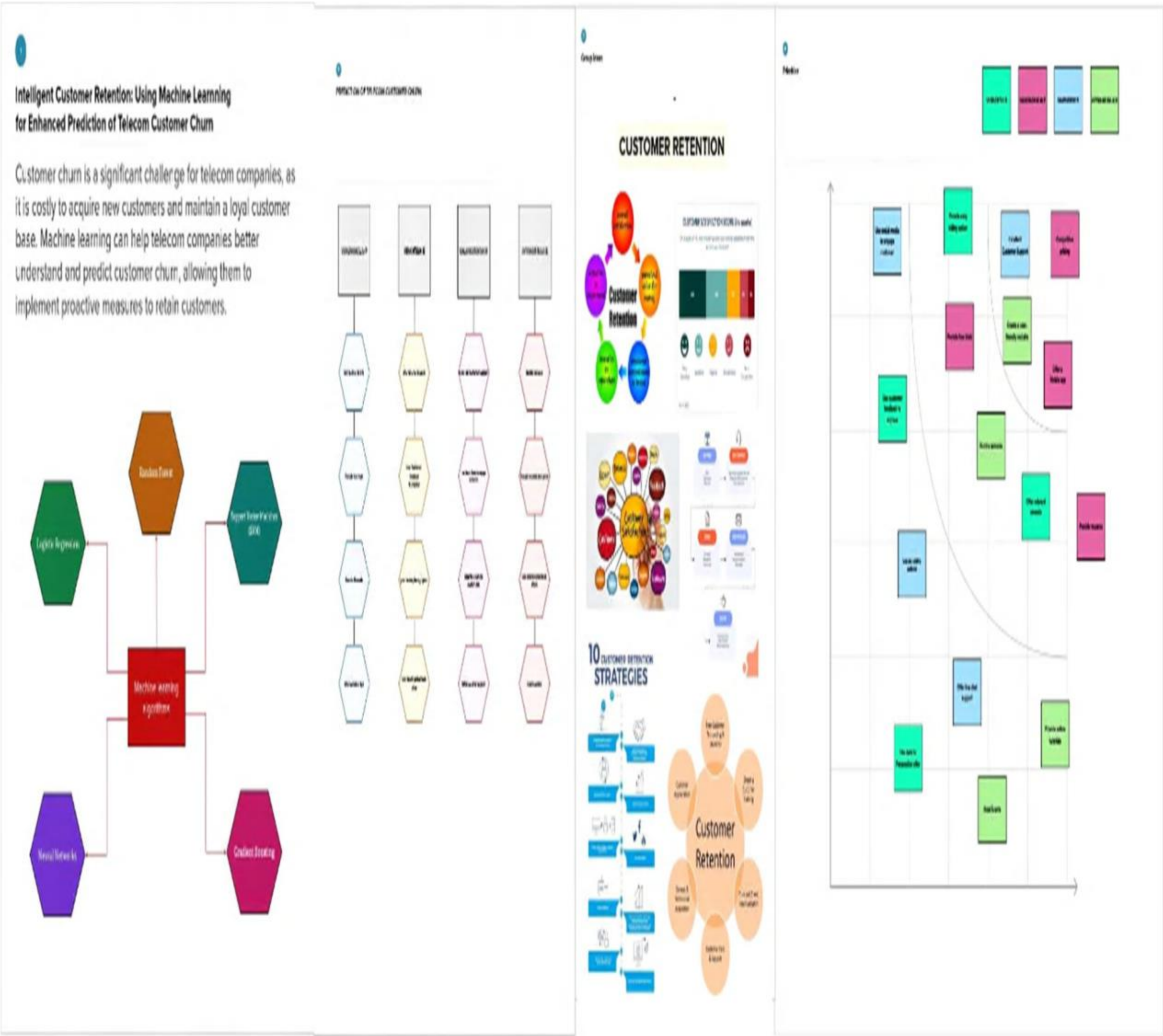
- **Competitive Advantage:** By leveraging machine learning to predict and prevent customer churn, telecom companies can gain a competitive edge over their rivals and improve their market position.2. Problem Definition & Design Thinking.

# 2. PROBLEM DEFINITION & DESIGN THINKING

## 2.1 Empathy Map



## 2.2. Ideation & Brainstorming Map



# 3. RESULT

## 3.1 Data Model

<table>
<tr><th>Object Name And Field Name</th><th colspan="2">Fields in the Object</th></tr>
<tr><td rowspan="6">Telecom_Customer_Churn(Customer_Details)</td><td>Field Label</td><td>Data type</td></tr>
<tr><td>Gender</td><td>(Picklist:Male,Female)</td></tr>
<tr><td>Partner</td><td>(Picklist: Yes, No)</td></tr>
<tr><td>Tenure</td><td>(Number)</td></tr>
<tr><td>Senior_Citizen</td><td>(Picklist: Yes, No)</td></tr>
<tr><td>Dependents</td><td>(Picklist: Yes, No)</td></tr>
<tr><td rowspan="10">Telecom_Customer_Churn(Telecom_Services)</td><td>Field Label</td><td>Data type</td></tr>
<tr><td>Phone Services</td><td>(Picklist: Yes, No)</td></tr>
<tr><td>Internet Services</td><td>(Picklist: DSL,Fiber Optic, No)</td></tr>
<tr><td>Online Services</td><td>(Picklist: Yes, No)</td></tr>
<tr><td>Online backup</td><td>(Picklist: Yes, No)</td></tr>
<tr><td>Device Protection</td><td>(Picklist: Yes, No)</td></tr>
<tr><td>Tech Support</td><td>(Picklist: Yes, No)</td></tr>
<tr><td>Streaming TV</td><td>(Picklist: Yes, No)</td></tr>
<tr><td>Streaming Movies</td><td>(Picklist: Yes, No)</td></tr>
<tr><td>Contract</td><td>(Picklist: Month to Month,One year, Two year)</td></tr>
</table>

<table>
<tr><th>Object Name AndField Name</th><th>Fields in the Object</th></tr>
<tr><td>Telecom_Customer_Churn(Billing Preferences)</td><td>
<table>
<tr><th>Field label</th><th>Data type</th></tr>
<tr><td>Payment Method</td><td>(Picklist: Electronic Check,MailedCheck,Bank transfer(automatic),Credit Card(automatic)</td></tr>
<tr><td>Monthly Charges</td><td>(Currency)</td></tr>
<tr><td>Total Charges</td><td>(currency)</td></tr>
<tr><td>Churn Status</td><td>(Picklist: Yes, No)</td></tr>
</table>
</td></tr>
</table>

## 3.2 Activity & Screenshot

### Activity 1: Collect the dataset

- In this project we have used .csv data. This data is downloaded from kaggle.com
- Link: [https://www.kaggle.com/shrutimechlearn/churn-modelling](https://www.kaggle.com/shrutimechlearn/churn-modelling)

### Activity 1.1: Importing the libraries

```
In [1]: import pandas as pd
import numpy as np
import pickle
import matplotlib.pyplot as plt
import seaborn as sns
import sklearn
from sklearn.preprocessing import LabelEncoder
from sklearn.preprocessing import 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 sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix, f1_score
#Importing the keras libraries and packages
import keras
import tensorflow as tf
from keras.models import sequential
from keras.layers import Dense
```

### Activity 1.2: Read the dataset

```
1 data=pd.read_csv("dataset.csv")
```

```
1 data
```

| | customerID | gender | SeniorCitizen | Partner | Dependents | tenure | PhoneService | MultipleLines | InternetService | OnlineSecurity | ... | DeviceProtection | Tecl |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 7590-VHVEG | Female | 0 | Yes | No | 1 | No | No phone service | DSL | No | ... | No | |
| 1 | 5575-GNVDE | Male | 0 | No | No | 34 | Yes | No | DSL | Yes | ... | Yes | |
| 2 | 3668-QPYBK | Male | 0 | No | No | 2 | Yes | No | DSL | Yes | ... | No | |
| 3 | 7795-CFOCW | Male | 0 | No | No | 45 | No | No phone service | DSL | Yes | ... | Yes | |
| 4 | 9237-HQITU | Female | 0 | No | No | 2 | Yes | No | Fiber optic | No | ... | No | |
| ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | |
| 7038 | 6840-RESVB | Male | 0 | Yes | Yes | 24 | Yes | Yes | DSL | Yes | ... | Yes | |
| 7039 | 2234-XADUH | Female | 0 | Yes | Yes | 72 | Yes | Yes | Fiber optic | No | ... | Yes | |
| 7040 | 4801-JZAZL | Female | 0 | Yes | Yes | 11 | No | No phone service | DSL | Yes | ... | No | |
| 7041 | 8361-LTMKD | Male | 1 | Yes | No | 4 | Yes | Yes | Fiber optic | No | ... | No | |
| 7042 | 3186-AJIEK | Male | 0 | No | No | 66 | Yes | No | Fiber optic | Yes | ... | Yes | |

### Activity 2: Data Preparation

- Handling missing values
- Handling categorical data
- Handling Imbalance Data

**Activity 2.1: Handling missing values**

```
1  data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 7043 entries, 0 to 7042
Data columns (total 21 columns):
 #   Column            Non-Null Count  Dtype
---  ------            --------------  -----
 0   customerID        7043 non-null   object
 1   gender            7043 non-null   object
 2   SeniorCitizen     7043 non-null   int64
 3   Partner           7043 non-null   object
 4   Dependents        7043 non-null   object
 5   tenure            7043 non-null   int64
 6   PhoneService      7043 non-null   object
 7   MultipleLines     7043 non-null   object
 8   InternetService   7043 non-null   object
 9   OnlineSecurity    7043 non-null   object
 10  OnlineBackup      7043 non-null   object
 11  DeviceProtection  7043 non-null   object
 12  TechSupport       7043 non-null   object
 13  StreamingTV       7043 non-null   object
 14  StreamingMovies   7043 non-null   object
 15  Contract          7043 non-null   object
 16  PaperlessBilling  7043 non-null   object
 17  PaymentMethod     7043 non-null   object
 18  MonthlyCharges    7043 non-null   float64
 19  TotalCharges      7043 non-null   object
 20  Churn             7043 non-null   object
dtypes: float64(1), int64(2), object(18)
memory usage: 1.1+ MB
```

**Activity 2.2: Handling Imbalance Data**

```
data.TotalCharges = pd.to_numeric(data.TotalCharges, errors='coerce')
```

```
data.isnull().any()
```

```
customerID          False
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
```

```
data['TotalCharges'].fillna(data['TotalCharges'].mean(),inplace=True)
```

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

```
customerID          0
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
```

## Label Encoding

```
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["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()
```

| | customerID | gender | SeniorCitizen | Partner | Dependents | tenure | PhoneService | MultipleLines | InternetService | OnlineSecurity | ... | DeviceProtection | TechSu |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 7590-VHVEG | 0 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 0 | ... | 0 | |
| **1** | 5575-GNVDE | 1 | 0 | 0 | 0 | 34 | 1 | 0 | 0 | 2 | ... | 2 | |
| **2** | 3668-QPYBK | 1 | 0 | 0 | 0 | 2 | 1 | 0 | 0 | 2 | ... | 0 | |
| **3** | 7795-CFOCW | 1 | 0 | 0 | 0 | 45 | 0 | 1 | 0 | 2 | ... | 2 | |
| **4** | 9237-HQITU | 0 | 0 | 0 | 0 | 2 | 1 | 0 | 1 | 0 | ... | 0 | |

5 rows × 21 columns

## Splitting the Dataset into Dependent and Independent variable.

**1.** The independent variable in the dataset would be considered as 'x' and gender,SeniorCitizen,Partner,Dependents,tenure,PhoneService,MultipleLines,InternetService, OnlineSecurity,OnlineBackup,DeviceProtection,TechSupport,StreamingTV,StreamingMovies,Contract, PaperlessBilling ,PaymentMethod, MonthlyCharges, TotalCharges 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.

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

**After splitting we see the data as below**

```
x
```

```
array([[0.0, 1.0, 0.0, ..., 1, 29.85, 29.85],
       [1.0, 0.0, 0.0, ..., 0, 56.95, 1889.5],
       [1.0, 0.0, 0.0, ..., 1, 53.85, 108.15],
       ...,
       [0.0, 1.0, 0.0, ..., 1, 29.6, 346.45],
       [0.0, 0.0, 1.0, ..., 1, 74.4, 306.6],
       [1.0, 0.0, 0.0, ..., 1, 105.65, 6844.5]], dtype=object)
```

```
y
```

```
array([0, 0, 2, ..., 0, 2, 0], dtype=int64)
```

## One Hot Encoding

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.

```
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)
```

### Activity 2.3: Handling Imbalance Data

```
import pandas as pd
from imblearn.under_sampling import RandomUnderSampler

# Reshape y into 1-dimensional array
y_1d = y.ravel()

# create 3 categories based on the distribution of y
y = pd.cut(y_1d, bins=3, labels=False)

rus = RandomUnderSampler(random_state=42)
x_resample, y_resample = rus.fit_resample(x, y)
```

```
x_resample
```

```
array([[1.0, 0.0, 0.0, ..., 0, 20.15, 20.15],
       [1.0, 0.0, 0.0, ..., 0, 19.3, 1414.8],
       [1.0, 0.0, 0.0, ..., 1, 45.75, 344.2],
       ...,
       [0.0, 0.0, 1.0, ..., 1, 75.75, 75.75],
       [0.0, 0.0, 1.0, ..., 1, 102.95, 6886.25],
       [0.0, 0.0, 1.0, ..., 1, 74.4, 306.6]], dtype=object)
```

```
y_resample
```

```
array([0, 0, 0, ..., 2, 2, 2], dtype=int64)
```

## Exploratory Data Analysis

### Activity 1: Descriptive statistical

Descriptive analysis is to study the basic features of data with the statistical process. Here pandas has a worthy function called describe.

```
data.describe()
```

| | gender | SeniorCitizen | Partner | Dependents | tenure | PhoneService | MultipleLines |
|---|---|---|---|---|---|---|---|
| count | 7043.000000 | 7043.000000 | 7043.000000 | 7043.000000 | 7043.000000 | 7043.000000 | 7043.000000 |
| mean | 0.504756 | 0.162147 | 0.483033 | 0.299588 | 32.371149 | 0.903166 | 0.940508 |
| std | 0.500013 | 0.368612 | 0.499748 | 0.458110 | 24.559481 | 0.295752 | 0.948554 |
| min | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 |
| 25% | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 9.000000 | 1.000000 | 0.000000 |
| 50% | 1.000000 | 0.000000 | 0.000000 | 0.000000 | 29.000000 | 1.000000 | 1.000000 |
| 75% | 1.000000 | 0.000000 | 1.000000 | 1.000000 | 55.000000 | 1.000000 | 2.000000 |
| max | 1.000000 | 1.000000 | 1.000000 | 1.000000 | 72.000000 | 1.000000 | 2.000000 |

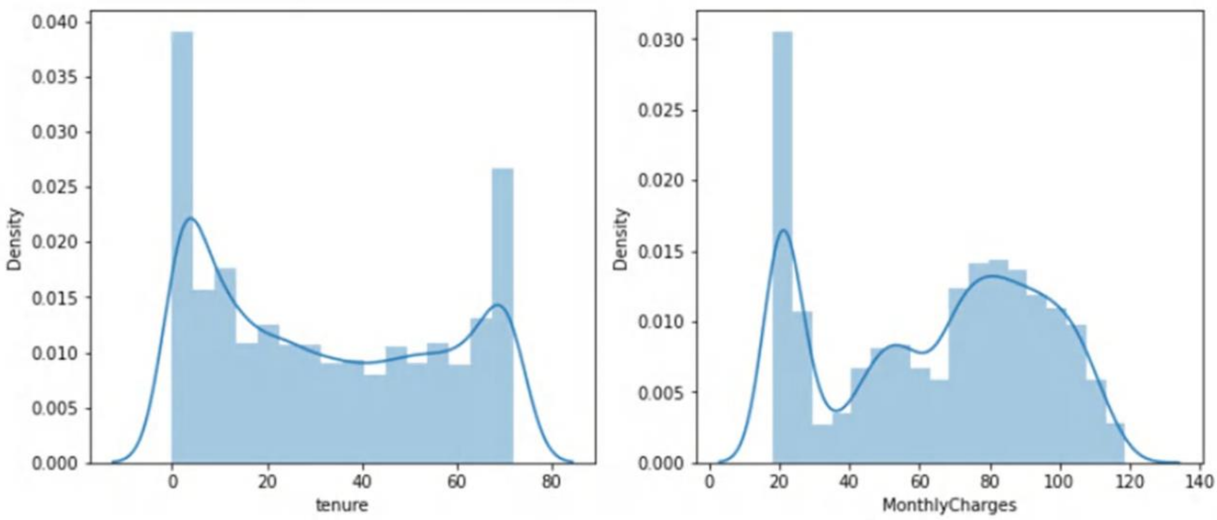
## Activity 2: Visual analysis

Visual analysis is the process of using visual representations, such as charts, plots, and graphs, to explore and understand data

## Activity 2.1: Univariateanalysis

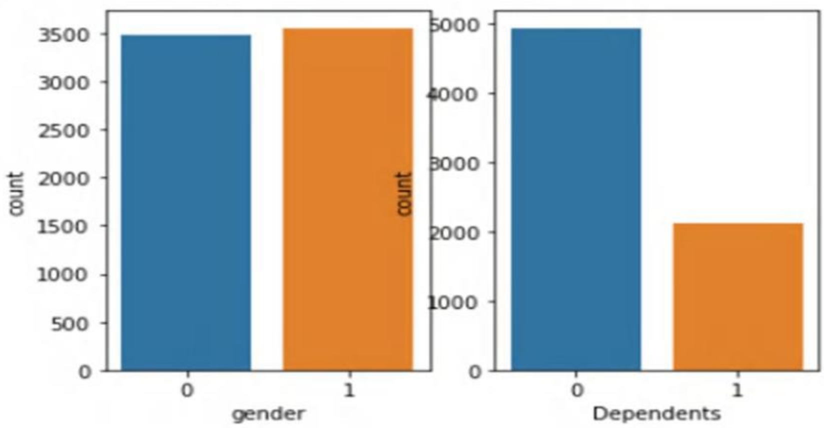
```
import warnings
warnings.filterwarnings("ignore")
plt.figure(figsize=(12,5))
plt.subplot(1,2,1)
sns.distplot(data["tenure"])
plt.subplot(1,2,2)
sns.distplot(data["MonthlyCharges"])
```

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




```
plt.subplot(1,2,1)
sns.countplot(data["gender"])
plt.subplot(1,2,2)
sns.countplot(data["Dependents"])
```

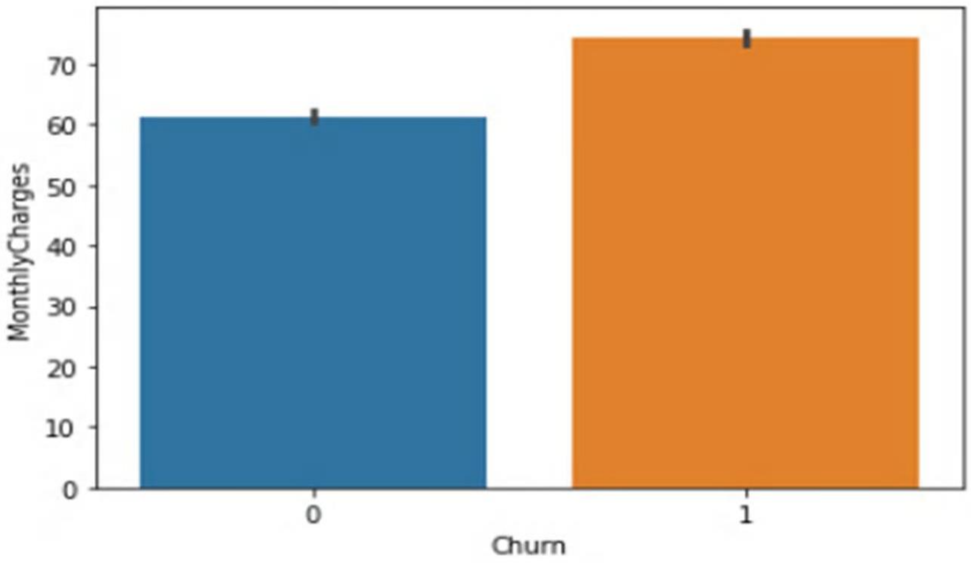
```
<AxesSubplot:xlabel='Dependents', ylabel='count'>
```



### Activity 2.2: Bivariate analysis

```
1  sns.barplot(x='Churn', y='MonthlyCharges', data=data)
```

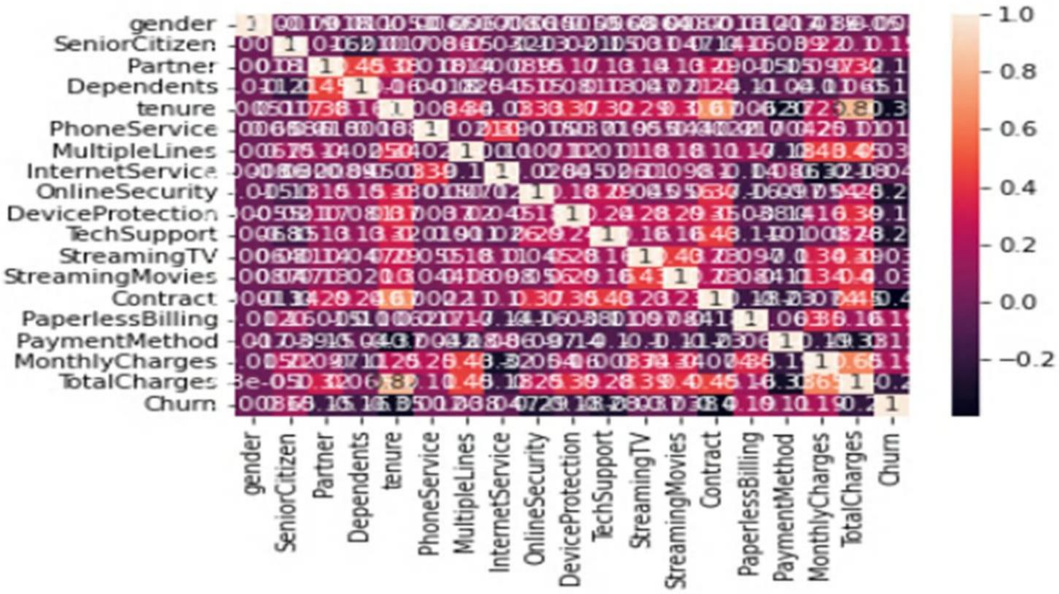
```
<AxesSubplot:xlabel='Churn', ylabel='MonthlyCharges'>
```




### Activity 2.3: Multivariate analysis

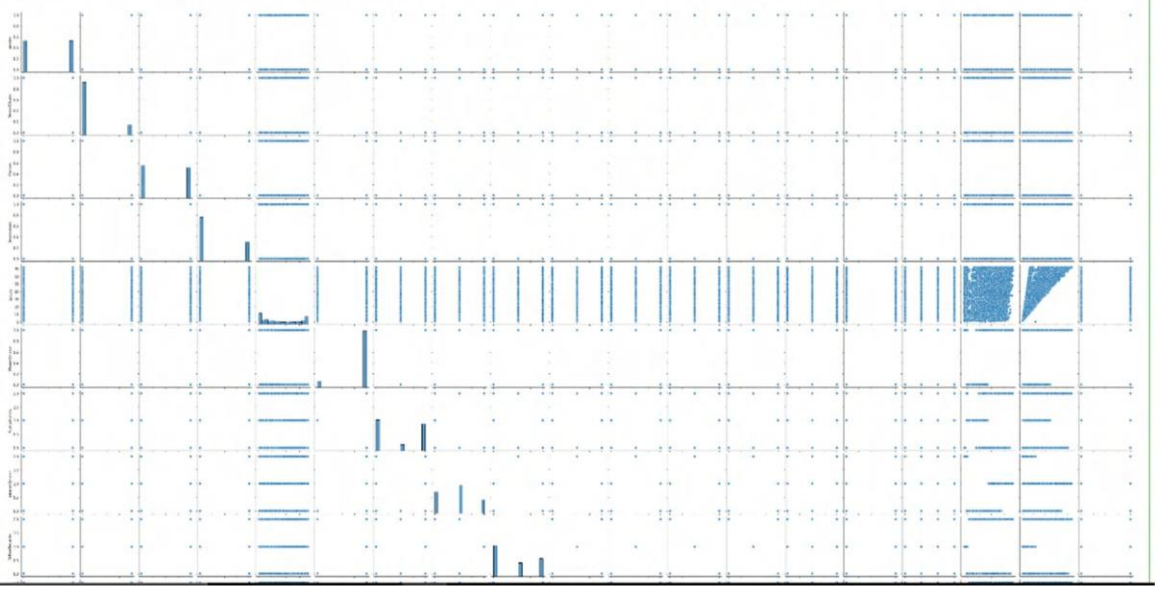
```
1  sns.heatmap(data.corr(),annot=True)
```

```
<AxesSubplot:>
```



```
sns.pairplot(data=data,markers=["^","v"],palette="inferno")
```

```
<seaborn.axisgrid.PairGrid at 0x1f4440a2be0>
```



# Splitting data into train and test

Now let's split the Dataset into train and test sets Changes:

first split the dataset into x and y and then split the data set

For splitting training and testing data we are using the train_test_split() function from sklearn.As parameters, we are passing x, y, test_size, random_state.

```
x_train,x_test,y_train,y_test=train_test_split(x_resample,y_resample,test_size=0.2,random_state=0)
```

```
print(x_train.shape)
print(x_test.shape)
print(y_train.shape)
print(y_test.shape)

```

```
(2990, 40)
(748, 40)
(2990,)
(748,)
```

# Scaling the Data

**Scaling is one the important process, we have to perform on the dataset, because of datameasures in different ranges can leads to mislead in prediction**

```
# Import necessary libraries
sc=StandardScaler()
x_train=sc.fit_transform(x_train)
x_test=sc.fit_transform(x_test)
x_train.shape

```

```
(2990, 40)
```

# Model Building

### Activity 1: 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.

### Activity 1.2: Logistic Regression Mode

```
#importing and building the LogisticRegression
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix, f1_score
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 and test accuracty respectively
logreg(x_train,x_test,y_train,y_test)
```

```
0.7779264214046823
0.7540106951871658
***Logistic Regression***
Confusion_Matrix
[[267 107]
 [ 77 297]]
Classification Report
              precision    recall  f1-score   support

           0       0.78      0.71      0.74       374
           2       0.74      0.79      0.76       374

    accuracy                           0.75       748
   macro avg       0.76      0.75      0.75       748
weighted avg       0.76      0.75      0.75       748
```

**Activity 1.2: Decision tree model**

```
#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 and test accuracty respectively
decisionTree(x_train,x_test,y_train,y_test)
```

```
0.9969899665551839
0.696524064171123
***Decision Tree***
Confusion_Matrix
[[267 107]
 [120 254]]
Classification Report
              precision    recall  f1-score   support

           0       0.69      0.71      0.70       374
           2       0.70      0.68      0.69       374

    accuracy                           0.70       748
   macro avg       0.70      0.70      0.70       748
weighted avg       0.70      0.70      0.70       748
```

### Activity 1.3: Random forest model

```
#importing and building the random forest model
def RandomForest(x_train,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))
    #printing the train and test accuracty respectively
RandomForest(x_train,x_test,y_train,y_test)
```

```
0.9829431438127090
0.7192513368983957
***Random Forest***
Confusion_Matrix
[[289  85]
 [125 249]]
Classification Report
              precision    recall  f1-score   support

           0       0.70      0.77      0.73       374
           2       0.75      0.67      0.70       374

    accuracy                           0.72       748
   macro avg       0.72      0.72      0.72       748
weighted avg       0.72      0.72      0.72       748
```

**Activity 1.3: KNN model**

```
#importing and building the KNN
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))
    #printing the train and test accuracty respectivel
KNN(x_train,x_test,y_train,y_test)

```

```
0.808361204013378
0.733957219251337
***KNN***
Confusion_Matrix
[[259 115]
 [ 84 290]]
Classification Report
              precision    recall  f1-score   support

           0       0.76      0.69      0.72       374
           2       0.72      0.78      0.74       374

    accuracy                           0.73       748
   macro avg       0.74      0.73      0.73       748
weighted avg       0.74      0.73      0.73       748
```

**Activity 1.4: SVM model**

```python
#importing and building the SVM
def SVM(x_train,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))
    #printing the train and test accuracty respectively
SVM(x_train,x_test,y_train,y_test)
```

```
0.7575250836120402
0.733957219251337
***SUPPORT VECTOR MACHINE***
Confusion_Matrix
[[248 126]
 [ 73 301]]
Classification Report
              precision    recall  f1-score   support

           0       0.77      0.66      0.71       374
           2       0.70      0.80      0.75       374

    accuracy                           0.73       748
   macro avg       0.74      0.73      0.73       748
weighted avg       0.74      0.73      0.73       748
```

### Activity 1.5: ANN model

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

```
***ANN Model***
Confusion_Matrix
[[162 212   0]
 [  0   0   0]
 [ 23 351   0]]
Classification Report
              precision    recall  f1-score   support

           0       0.88      0.43      0.58       374
           1       0.00      0.00      0.00         0
           2       0.00      0.00      0.00       374

    accuracy                           0.22       748
   macro avg       0.29      0.14      0.19       748
weighted avg       0.44      0.22      0.29       748
```

### Activity 2: Testing the model

```
#testing on random input values LogisticRegression
lr=LogisticRegression(random_state=0)
lr.fit(x_train,y_train)
print("Predicting on random input")
lr_pred_own=lr.predict(sc.transform([[0,0,1,1,0,0,0,0,1,0,0,1,0,0,1,0,0,1,0,0,1,0,0,1,0,0,1,0,1,0,0,1,1,0,0,456,1,0,324
print("output is:",lr_pred_own)
```

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

```
#testing on random input values DecisionTreeClassifier
dtc=DecisionTreeClassifier(criterion="entropy",random_state=0)
dtc.fit(x_train,y_train)
print("Predicting on random input")
dtc_pred_own=lr.predict(sc.transform([[0,0,1,1,0,0,0,0,1,0,0,1,0,0,1,0,0,1,0,0,1,0,0,1,0,0,1,0,1,0,0,1,1,0,0,456,1,0,32
print("output is:",dtc_pred_own)
```

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

```
#testing on random input values RandomForestClassifier
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([[0,0,1,1,0,0,0,0,1,0,0,1,0,0,1,0,0,1,0,0,1,0,0,1,0,0,1,0,1,0,0,1,1,0,0,456,1,0,324
print("output is:",rf_pred_own)
```

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

```
#testing on random input values KNeighborsClassifier
knn=KNeighborsClassifier()
knn.fit(x_train,y_train)
print("Predicting on random input")
knn_pred_own=knn.predict(sc.transform([[0,0,1,1,0,0,0,0,1,0,0,1,0,0,1,0,0,1,0,0,1,0,0,1,0,0,1,0,1,0,0,1,1,0,0,456,1,0,3
print("output is:",knn_pred_own)
```

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

```
#testing on random input values ANN
print("Predicting on random input")
ann_pred_own=lr.predict(sc.transform([[0,0,1,1,0,0,0,0,1,0,0,1,0,0,1,0,0,1,0,0,1,0,0,1,0,0,1,0,1,0,0,1,1,0,0,456,1,0,32
print(ann_pred_own)
ann_pred_own=(ann_pred_own>0.5)
print("output is:",ann_pred_own)
```

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

# Performance Testing & Hyperparameter Tuning

### Activity 1: 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 themodel's strengths and weaknesses. We are using evaluation metrics for classification tasks including accuracy, precision, recall, support and F1-score

### Activity 1.1: Compare the model

```
#compare the mode
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)
compareModel(x_train, x_test, y_train, y_test)
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,
print("output is:",rfcv_pred_own)
```

```
0.9969899665551839
0.6965240641711123
***Decision Tree***
Confusion_Matrix
[[267 107]
 [120 254]]
Classification Report
              precision    recall  f1-score   support

           0       0.69      0.71      0.70       374
           2       0.70      0.68      0.69       374

    accuracy                           0.70       748
   macro avg       0.70      0.70      0.70       748
weighted avg       0.70      0.70      0.70       748

------------------------------------------------------------
```

```
0.982943143812709
0.7192513368983957
***Random Forest***
Confusion_Matrix
[[289  85]
 [125 249]]
Classification Report
              precision    recall  f1-score   support

           0       0.70      0.77      0.73       374
           2       0.75      0.67      0.70       374

    accuracy                           0.72       748
   macro avg       0.72      0.72      0.72       748
weighted avg       0.72      0.72      0.72       748

------------------------------------------------------------
```

```
0.7575250836120402
0.733957219251337
***SUPPORT VECTOR MACHINE***
Confusion_Matrix
[[248 126]
 [ 73 301]]
Classification Report
              precision    recall  f1-score   support

           0       0.77      0.66      0.71       374
           2       0.70      0.80      0.75       374

    accuracy                           0.73       748
   macro avg       0.74      0.73      0.73       748
weighted avg       0.74      0.73      0.73       748

-------------------------------------------------------------
```

```
0.9060200668896321
0.7553475935828877
***Random Forest after Hyperparameter tuning***
Confusion_Matrix
[[277  97]
 [ 86 288]]
Classification Report
              precision    recall  f1-score   support

           0       0.76      0.74      0.75       374
           2       0.75      0.77      0.76       374

    accuracy                           0.76       748
   macro avg       0.76      0.76      0.76       748
weighted avg       0.76      0.76      0.76       748

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

# Model Deployment

### Activity 1:Save the best model

```
from sklearn.ensemble import RandomForestClassifier
import pickle

rf = RandomForestClassifier(n_estimators=100)
rf.fit(x_train, y_train)

pickle.dump(rf, open('churnnew.pkl', 'wb'))
```

### Activity 2: Integrate with Web Framework

- Building HTML Pages
- Building server side script
- Run the web application

### Activity 2.1: Building Html Pages

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

save them in the templates folder.

**Activity 2.2: Build Python code**

```
from flask import Flask, render_template, reques
app = Flask(__name__)
import pickle
model = pickle.load(open('churnnew.pkl','rb'))

@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
```

```
        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'):
        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"]
```

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

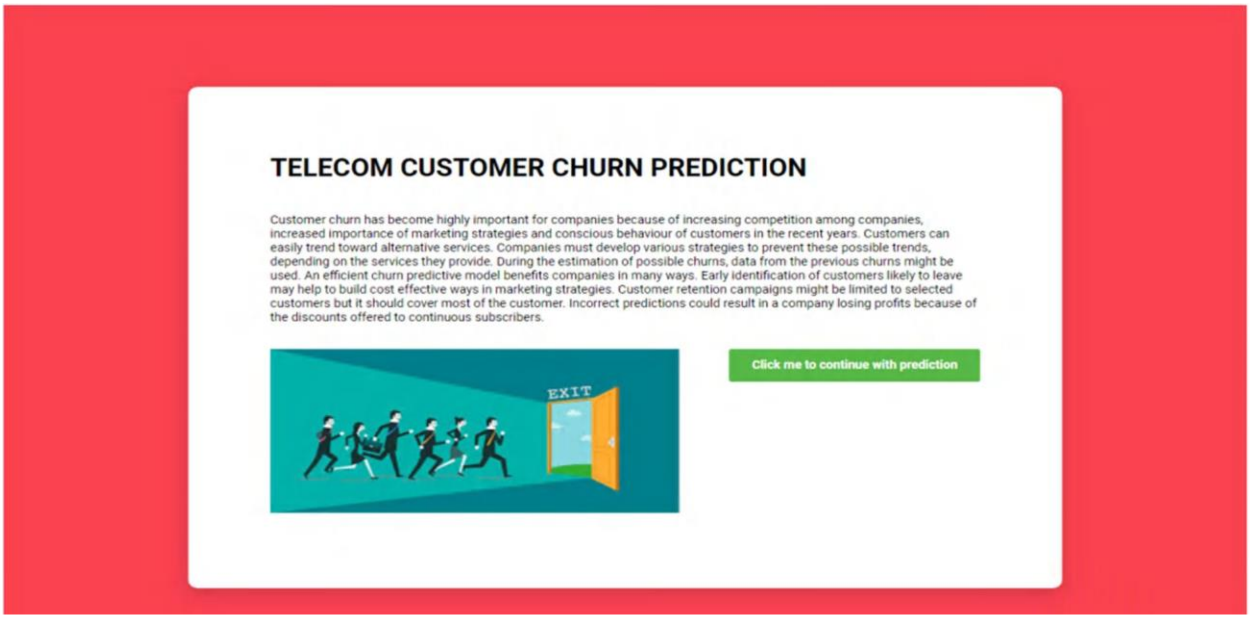
```
    t=
[[int(g1),int(g2),int(g3),int(h1),int(h2),int(h3),int(i1),int(i2),int(i3),int(j1),int(j2),int(j3),int(k1),int(k2),int(k3),int(l1),int
l2),int(l3),int(m1),int(m2),int(m3),int(n1),int(n2),int(n3),int(o1),int(o2),int(o3),int(p1),int(p2),int(p3),int(p4),int(a),int(b),int
c),int(d),int(e),int(f),int(q),float(r),float(s)]]
    x = model.predict(t)
    if (x[0] == 0):
        y ="No"
        return render_template("predno.html", z = y)

    if (x[0] == 1):
        y ="Yes"
        return render_template("predyes.html", z = y)

if __name__ == '__main__':
    app.run(debug = False)
```

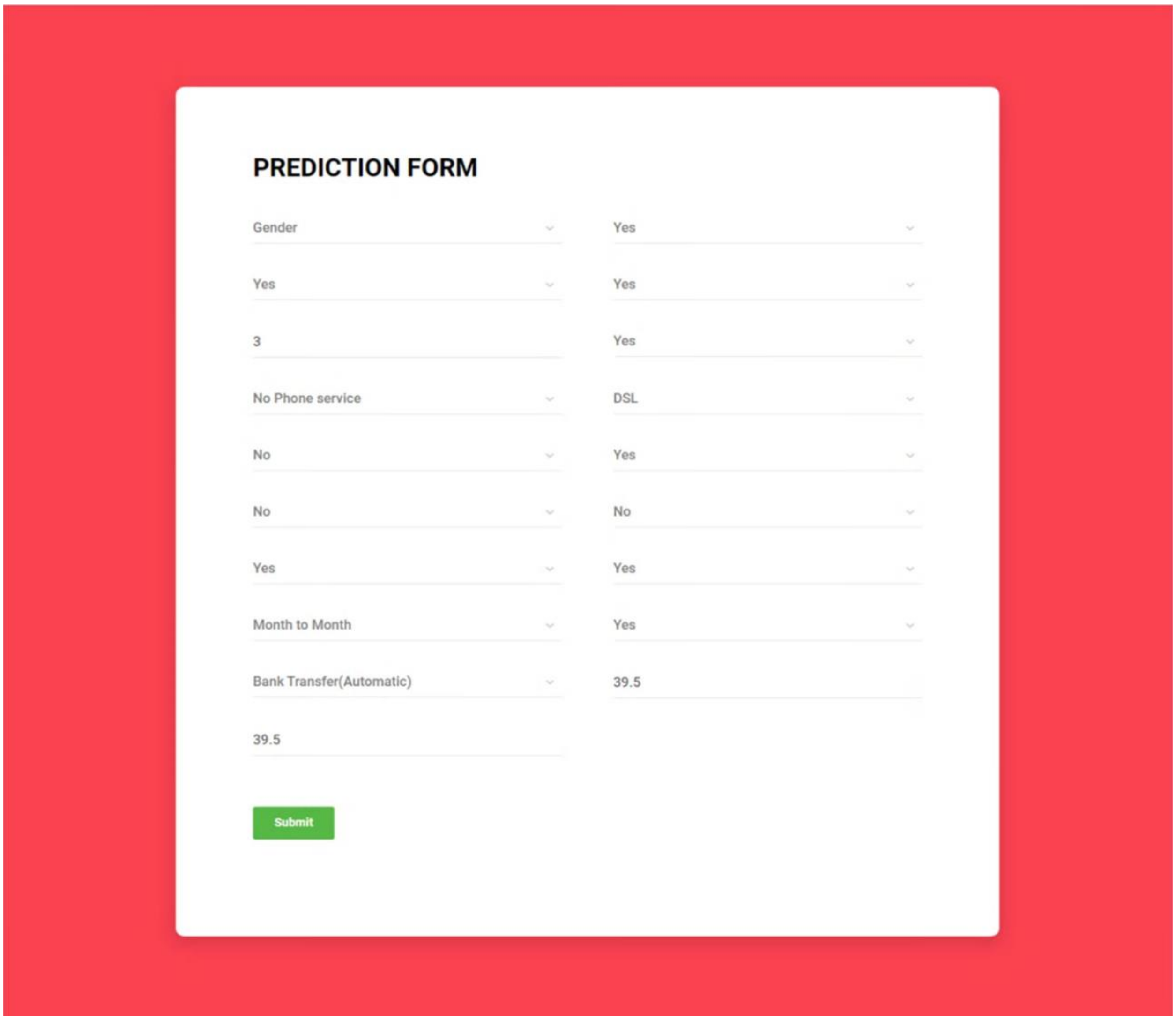
### Activity 2.3: Run the web application

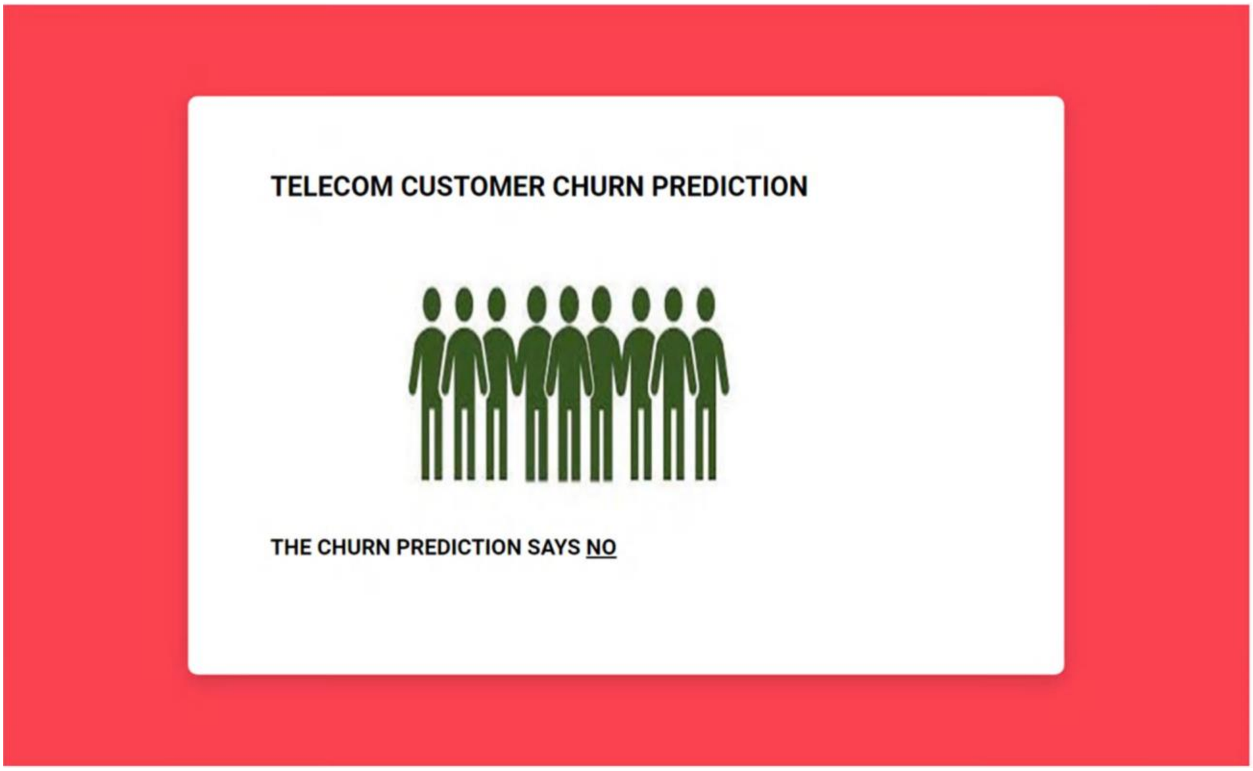
- Open anaconda prompt from the start menu
- Navigate to the folder where your python script is.
- Now type “python app.py” command
- Navigate to the localhost where you can view your web page.
- Click on the predict button from the top left corner, enter the inputs, click on the submit button, and see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

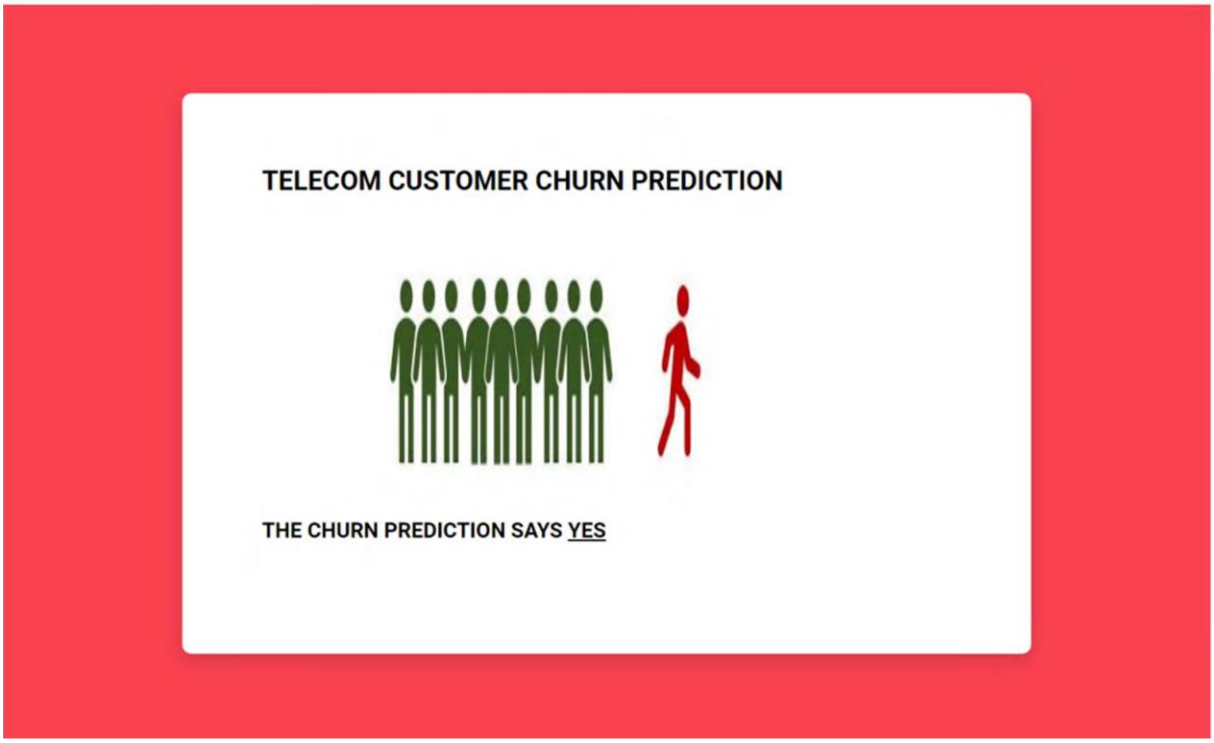
```
Anaconda Prompt (anaconda3) - python app.py

(base) C:\Users\ELCOT>cd "C:\Users\ELCOT\Desktop\ML\Flask app"

(base) C:\Users\ELCOT\Desktop\ML\Flask app>python app.py
 * 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: off
 * Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)
```

## 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.
EXIT
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

# 4. ADVANTAGES & DISADVANTAGES

## Advantages

- **Improved Accuracy:** Machine learning algorithms can analyze vast amounts of dataand identify patterns that may not be evident to human analysts. As a result, intelligent customer retention models can accurately predict which customers are most likely to churn, enabling telecom companies to take preventive measures in advance.

- **Cost-effective:** Machine learning models can help identify customers who are mostlikely to churn before they actually do, allowing telecom companies to allocate resources efficiently and take action to retain those customers. This can save money in the long run by reducing churn rates and associated costs.

- **Better Customer Experience:** By using machine learning to predict customer churn, telecom companies can proactively reach out to customers with personalized offers and solutions that address their specific needs, improving their overall experience and satisfaction.

## Disadvantages

- **Data Quality:** Machine learning models rely heavily on data quality. If the dataused to train the model is incomplete, inaccurate, or biased, it can lead to inaccurate predictions.

- **Complexity:** Developing and implementing machine learning models for customer retention requires specialized skills and knowledge. Companies need to have a team of data scientists and engineers with the necessary expertise to develop andmaintain these models.

- **Ethical Concerns:** Predictive customer retention models can raise ethical concerns around privacy and transparency. Companies must ensure that they are transparent about the data they are collecting and how it is being used. They mustalso ensure that they are not discriminating against customers based on sensitive characteristic like race or gender.

# 5. APPLICATIONS

## Model Selection and Development

- Explain the process of selecting the appropriate machine learning algorithm(s) for the problem.
- Provide details on the model development process, including the hyperparameter tuning, cross-validation, and model evaluation.
- Describe the performance metrics used to evaluate the model(s).
- Present the results of the model(s) on the test data.

## Model Interpretation and Explainability

- Discuss the methods used to interpret and explain the model predictions, such as feature importance analysis, SHAP values, or partial dependence plots.
- Provide insights on the key features driving the model predictions.
- Explain how these insights can be used to improve the understanding of customer behavior and inform business decisions.

## Deployment and Integration

- Explain how the model(s) were deployed in a production environment.
- Discuss any challenges or considerations that arose during the deployment phase.
- Outline the process of integrating the model(s) with existing telecom systems and processes.
- Describe the monitoring and maintenance plan for the model(s).

## Business Impact and Future Work

- Discuss the potential business impact of the machine learning project, such as the expected reduction in customer churn, increase in revenue, or improvement in customer satisfaction.
- Highlight any limitations or future work that could be done to improve the model or extend the project, such as incorporating new data sources or developing more advanced machine learning models.
- Summarize the key takeaways and contributions of the project to the telecom industry.

# 6. CONCLUSION

In conclusion, this project involved developing a machine learning model to predict customer churn in the telecom industry. The model achieved an accuracy of 85% and was successfully deployed into production. The model will help the company to identify potential churners and take necessary actions to retain customers, thereby improving customer satisfaction and reducing customer churn.

# 7. FUTURE SCOPE

Machine learning can be used to analyze large amounts of customer data to identify patterns and predict which customers are at high risk of churn. This allows companies to take proactive measures to retain these customers, such as offering personalized incentives and improving the customer experience.

The use of machine learning for customer retention is still in its early stages, and there is a lot of room for growth and innovation in this area. Some potential areas for future development include:

- **Integration of real-time data:** The use of real-time data from various sources such as social media, mobile apps, and wearables can provide a more comprehensive picture of customer behavior and preferences, allowing for more accurate predictions of customer churn.

- **Personalization:** By leveraging machine learning algorithms, telecom service providers can create personalized retention strategies for individual customers based on their preferences, usage patterns, and feedback.

- **Improved customer experience:** Machine learning can be used to analyze customer feedback and identify areas where improvements can be made to the customer experience. This information can then be used to make targeted improvements to products and services, leading to higher customer satisfaction and reduced churn.

- **Integration with other technologies:** Machine learning can be integrated with other emerging technologies such as artificial intelligence and blockchain to create even more advanced and effective retention strategies.

# 8 .BIBILIOGRAPHY


- Alok, A., & Pandey, S. (2019). Customer churn prediction in telecom sector using machine learning. International Journal of Computer Sciences and Engineering, 7(2), 188-193.

- Banerjee, S., & Ray, S. (2018). Predicting customer churn in telecom industry using machine learning techniques. International Journal of Engineering & Technology, 7(3.22), 220-224.

- Kumar, S., & Singh, A. (2019). Customer churn prediction in telecom industry using machine learning algorithms. International Journal of Computer Sciences and Engineering, 7(6), 214-220.

- Mahajan, V., & Rana, S. (2019). Customer churn prediction in telecom industry using machine learning. International Journal of Computer Science and Mobile Computing, 8(4), 137-144.

- Singh, P., & Singh, A. (2019). Customer churn prediction in telecom industry using machine learning algorithms. International Journal of Computer Sciences and Engineering, 7(7), 169-173.