Sri Lanka Institute of Information Technology

Machine Learning - Assignment

IT4060

Classification of Customer Churn Prediction (Telecom)

Group members

Student Name	ID	Email	Contact Number
Wijesinghe.K.H	IT20134358	it20134358@my.sliit.lk	071 8944639
Dilshan U.K. T	IT20124526	it20124526@my.sliit.lk	0702050322
Dilshan K. B. G. L	IT20151188	it20151188@my.sliit.lk	0773482114
Wickramasinghe G.A. N	IT20158972	it20158972@my.sliit.lk	0776475998

Table of Contents

Table of Figures (Figures)

Table of Figures (Tables)

Project links

GitHub Link: https://github.com/Hasaranga-sys/ML_Assignmenmt.git

YouTube Channel Link: https://youtube.com/@lahirukariyawasam9550

YouTube Video Link: https://youtu.be/W5IPmbLtOl8

Dataset Link: Telco Customer Churn | Kaggle

Classification of Customer Churn Prediction (Telecom)

1. Introduction

The percentage of consumers that ceased utilizing a company's product or service during a specific period is known as customer churn. Given that it is far less expensive to maintain current customers than it is to gain new ones, customer churn is one of the most crucial KPIs for a growing firm to monitor. Customers in the telecom sector have several service providers to pick from and can actively switch between them. In this fiercely competitive sector, the telecoms industry experiences an annual turnover rate of 15 to 25 percent.

Customer churn costs businesses a lot of money. According to one report, carriers lose $65 million every month due to churn based on a churn rate of slightly under two percent for top enterprises. Telecom businesses should identify the consumers who are most likely to leave to decrease customer churn.

Because most businesses have several clients and cannot afford to spend a lot of time with each one, individualized customer retention is challenging. The added money would not balance the expenditures, which would be too high. However, a business might focus its customer retention efforts exclusively on these "high-risk" consumers if it could foresee which clients are most likely to depart. A machine learning model may be used to identify consumers who are more likely to leave and how confident they are about doing so. We will employ a categorization strategy in this project.

determining whether a client will be lost through churn. Under the classification strategy, we employ 4 distinct algorithms. which include XGBoost, Logistic Regression, Decision Tree Classifier, and Random Forest Classifier. And we'll go with a more accurate model. The model team has chosen a publicly accessible dataset on Kaggle for training and testing. A web application with a user-friendly interface that sits on top of the machine learning model and enables interaction was also decided upon by the team.

2. Problem

Customer churn costs businesses a lot of money. According to one report, carriers lose $65 million every month due to churn based on a churn rate of slightly under two percent for top enterprises. When a consumer stop being a customer, that is, decides not to utilize your goods or services, this is known as customer attrition, also known as customer churn. The customer churn rate is the frequency with which customers stop utilizing your services. Every Telecom business encounter customer turnover: the goal is to determine the causes and reduce your churn rate.

This is a major issue because losing clients can reduce revenue and have a negative impact on company growth. The project's objective is to develop a predictive model using historical customer data that can identify patterns and elements that influence churn. A machine learning model that can accurately identify which consumers are most likely to stop using a company's services is being developed as part of the classification of customer churn prediction ML project. The project intends to give businesses advice on how to keep clients and enhance customer loyalty, ultimately raising revenue and increasing customer satisfaction.

3. Solution

The main goal is to determine whether a customer is likely to be churned based on data like phone service, internet service type, whether they have multiple services enabled, online security, device protection, tech support, and whether they stream movies and TV shows over the connection. In this study, the applicability of machine learning algorithms to identify high-probability clients who would leave was examined. Additionally offers consumers a web application they may use to quickly see a model's forecast of their likelihood to churn.

4. Data set

Dataset Name - Telco Customer Churn Prediction dataset

source – Kaggle

Link - Telco Customer Churn | Kaggle

Information on a fictitious telecom firm that offered home phone and Internet services to 7043 may be found in the telecom customer churn statistics. It shows which clients have canceled, stayed, or joined their service. Each client has several significant qualities, including Churn Score.

Variable name	Description
Customer ID	a unique ID that makes each consumer identifiable.
gender	Whether a man or woman is the customer
SeniorCitizen	Whether or whether the client is a senior citizen (1, 0)
Partner	Whether or if the client has a partner (Yes, No)
Dependents	Whether or if the consumer is a dependant (Yes, No)
PhoneService	If the consumer has phone service, whether they do or not (Yes, No)
MultipleLines	If the client has more than one line (Yes, No, No phone service)
InternetService	Internet service provider for the customer (DSL, Fiber optic, or not)
OnlineSecurity	Whether the consumer has access to online security (internet service: Yes, No, No)
OnlineBackup	Whether the consumer has access to online backup (Internet service: Yes, No, No)
DeviceProtection	If the consumer has device protection (Yes, No, Internet service is not available),
TechSupport	Whether the consumer has access to tech help (Yes, No, Internet service not available)
StreamingTV	Whether the client has access to streaming TV (Internet service: Yes, No, or Not Available)
StreamingMovies	Whether the user has access to streaming movies (Internet service: Yes, No, No)
tenure	The length of time the consumer has been a customer of the business
Contract	The length of the customer's contract (month-to-month, a year, or two years)
PaperlessBilling	If the consumer receives paperless billing (Yes, No)
PaymentMethod	The customer's chosen payment method (credit card, automated bank transfer, electronic check, paper check)
MonthlyCharges:	The monthly fee assessed to the client
TotalCharges	The overall cost incurred by the client
Churn	Target, Whether the client has departed within the past month (Yes/No)

Table 1: Attributes of the dataset

5. Methodology

5.1 Data Preparation

Then as we checked data types for all the columns, we observed TotalCharges is in object datatypes we just converted into float datatype.

```
# convert to numeric
df.TotalCharges=pd.to_numeric(df.TotalCharges,errors='coerce')
```

```
df.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       11
Churn               0
dtype: int64
```

Figure 1: Checked data types

Handle null values within the dataset and replace them with mean. It was observed that there are 11 null values available in this TotalCharges data set and replace them with the mean, outliers checking before filling up null values. Remove customer ID column is a unique number assigned to every customer. Therefore, ID has no prediction power.

```
#fill null values
mean=df['TotalCharges'].mean()
```

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

```
df.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
```

```
#remove columns
```

```
df.drop(columns= ['customerID'], axis=1, inplace=True)
```

Figure 2:Handle null values

Checking correlation (To Identify the important columns for our prediction)

```
for i in (df):

    plt.figure(i)
    sys.catplot(data=df,x=i,hue='Churn',kind='count',palette="ch:.35")
# plotting correlation heatmap
dataplot = sys.heatmap(df.corr(), cmap="YlGnBu", annot=True)
plt.show()
```

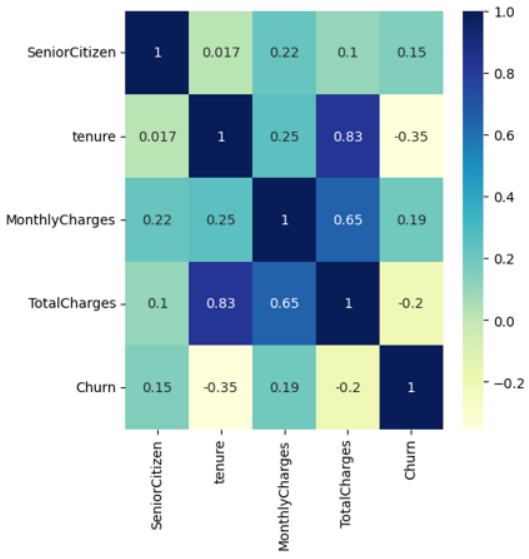


Figure 3:Checking correlation

Map churn column value Yes, no into 1,0

```
import numpy  as np
df['Churn'] = np.where(df.Churn == 'Yes',1,0)
df
```

```
df['Churn']
```

```
0       0
1       0
2       1
3       0
4       1
       ..
7038    0
7039    0
7040    0
7041    1
7042    0
Name: Churn, Length: 7043, dtype: int32
```

Figure 4:Map churn column value

Apply One-hot encoding to replace categorical data columns with numerical ones because it creates new (binary) columns, indicating the presence of each possible value from the original data. graphical representation the correlation of all the variables with churn column after hot one hot encoding.

```
#dummy the dataframe to convert categorical variables into numerical variables for further analysis
df_dummy = pd.get_dummies(df)
```

```
# correlation with churn column plotting for each and every variable in dummy df
plt.figure(figsize=(20,10))
df_dummy.corr()['Churn'].sort_values(ascending = False).plot(kind='bar')
```

```
<AxesSubplot:>
```

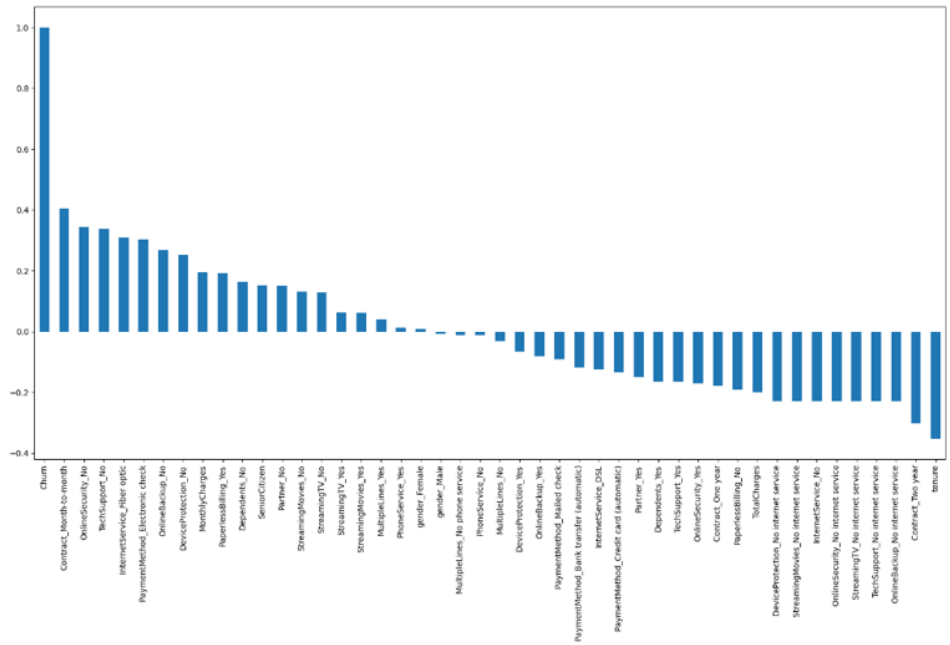
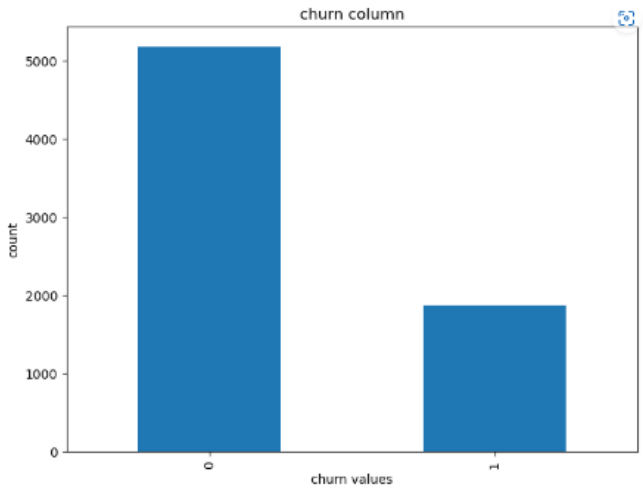


Figure 5:graphical representation the correlation of all the variables

We can utilize methods like SMOTEENN, to handle class imbalance in a dataset. The dataset can be divided into independent and dependent variables, and then we can rebalance using SMOTEENN to generate a new, more evenly distributed training set. To confirm the model's efficacy across the full dataset, it is crucial to assess its performance across both classes.

```
# checking how many churn and non churned customes are there
y_value.value_counts().plot(kind='bar',figsize=(8,6))
plt.xlabel("churn values")
plt.ylabel("count")
plt.title("churn column")
```

```
Text(0.5, 1.0, 'churn column')
```




```
#checking balance of data set
100* y_value.value_counts()/len(y_value)
```

```
0    73.463013
1    26.536987
Name: Churn, dtype: float64
```

```
sm = SMOTEENN()
#from imblearn.over_sampling import SMOTE
#sm=SMOTE()
X_resampled, y_resampled = sm.fit_resample(x_values,y_value)
```

```
#checking rebalanced data percentage of churn class
100* y_resampled.value_counts()/len(y_resampled)
```

```
1    55.092357
0    44.907643
Name: Churn, dtype: float64
```

Figure 6:rebalance using SMOTEENN

5.2 Model Selection

Split data set into training and testing.

1. Training Set – 80% of the records
2. Testing Set – 20% of the record

After splitting the dataset, use 4 types of Classification algorithms to train the dataset.

5.3 Data Visualization

To comprehend the relationships between features and look for possible patterns, we utilized data visualization in this project. We did this by creating a heatmap from the dataset's correlation matrix. The seaborn python library which is used for Matplotlib based data visualization is used in visualizing the heatmap and different ranges of colors were used to represent the data values. To handle outliers, we needed a clear grasp of upper and lower bounds, therefore we used a boxplot to represent the individual attributes.

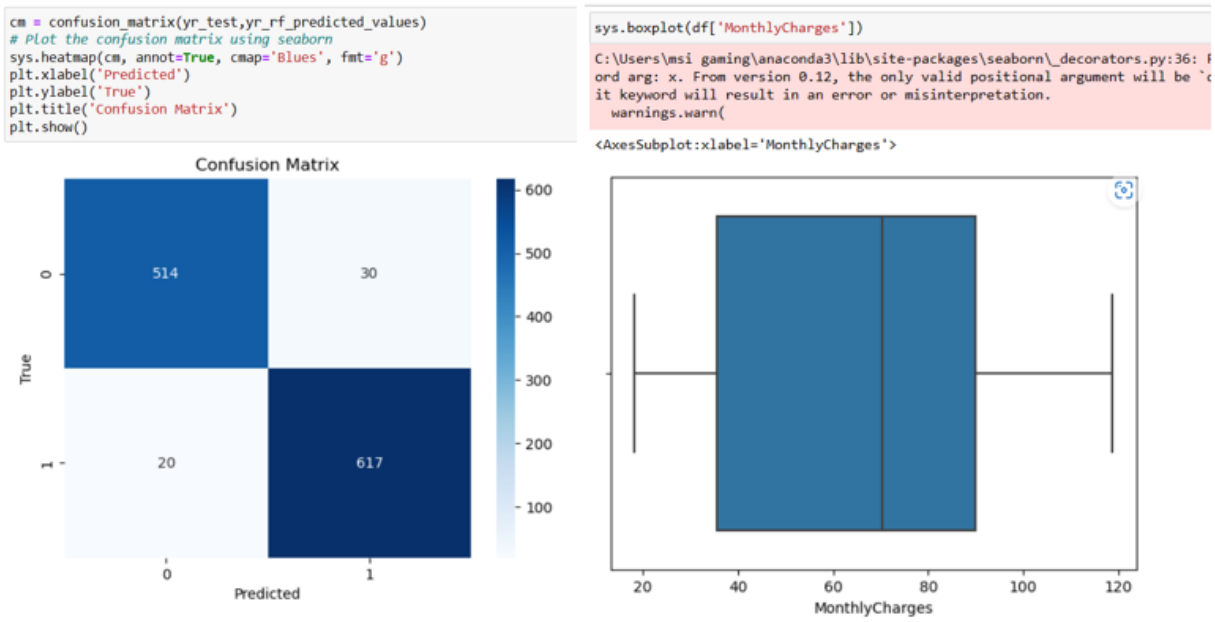


Figure 7:Data Visualization

It is unusual to discover the data points separated from other values during the Data Analysis process. As a result, we deal with and eliminate the outliers because it was crucial. We next plotted a scattered matrix and evaluated the data distribution in the graph since we needed to have a clear knowledge of how each variable related to one another to decide on the method to use.

6. Limitations

We must consider a few circumstances while discussing restrictions, and because we have previously said that these requirements are covered in detail in the preceding parts, we will utilize a table to do so in this section.

Condition	Reason
Having Null values	There were 11 rows with null values
Having large data set	There were only 7000 instances in the data set
Parameter selecting for the analysis	We renamed a parameter that was previously nameless and contained unneeded integer data.
the dataset contains duplicates or outliers	There were few outliers but no duplicates. Through histograms, we find upper limits and lower limits and eliminate them.
Not dividing the data between 2 categories as balance	The dataset had to be resampled because the data was not divided between 2 categories at a level of 50%.

Table 2:Limitations

7. Future work

Using a training set for neural networks:

- Using neural networks to train the dataset will increase prediction accuracy since they employ hidden layers to create predictions whereas linear regression techniques simply use input and output nodes.

8. Results and discussion

Our customer churn prediction ML project's results show that the four algorithms we employed—logistic regression, XGBoost, decision tree, and random forest—performed well and had high accuracy ratings. XGBoost scored 94.9%, decision tree scored 93.6%, logistic regression scored 94.5%, and random forest scored 95.7%, the highest accuracy score.

These findings have led us to select the random forest algorithm as the final model for predicting loss of clients. It outperformed the other algorithms and received the greatest accuracy score, which explains why. Its improved performance may have been influenced by the fact that random forests can handle non-linear correlations and interactions between variables.

As a result of our customer churn prediction ML project, businesses now have a useful tool for identifying clients who may cancel their services in the future. The model created with the help of the random forest algorithm was highly accurate and has the potential to boost

customer retention and profitability for businesses. To ensure the model's continuous efficacy, it is crucial to keep an eye on its performance and update it as needed.

9. Appendix

Source code

```
# -*- coding: utf-8 -*-
"""ML_Assignment 01.ipynb

Automatically generated by Colaboratory.

Original file is located at
    https://colab.research.google.com/drive/1-
eOe4vCZ3XTeD1VnLuacig4SkHlbaE34
"""

# import data seta
file="/content/Telco-Customer-Churn.csv"
import pandas as pd
import matplotlib.pyplot as plt

missing_values = ["n/a", "na", "--"," "]
#df=pd.read_csv(file,na_values=missing_values)
df=pd.read_csv(file)

#top few rows
df.head()

# display data as metrix
df.shape

df.describe()

df.dtypes

# convert to numeric
df.TotalCharges=pd.to_numeric(df.TotalCharges,errors='coerce')

df.isnull().sum()
```

```
df.dtypes

#location of null values
df.loc[df["TotalCharges"].isnull()==True]

#outliers checking before fill up null values
import seaborn as sys
sys.boxplot(df['TotalCharges'])

sys.boxplot(df['MonthlyCharges'])

#fill null values
mean=df['TotalCharges'].mean()

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

df.isnull().sum()

#remove columns
df.drop(columns= ['customerID'], axis=1, inplace=True)

#visulaize count of churn and not churn for each independent variable
#please wait for few mins until it loads all graphs and charts
import seaborn as sys

# for i in (df):

#     plt.figure(i)
#     sys.catplot(data=df,x=i,hue='Churn',kind='count',palette="ch:.35")
# # plotting correlation heatmap
# dataplot = sys.heatmap(df.corr(), cmap="YlGnBu", annot=True)
# plt.show()

df

import numpy  as np
df['Churn'] = np.where(df.Churn == 'Yes',1,0)
df

df['Churn']

#dummy the dataframe to convert categorical variables into numerical
variables for further analysis
```

```
df_dummy = pd.get_dummies(df)

df_dummy

#dataplot=sys.heatmap(df_dummy.corr())
#plt.show()

# correlation with churn column plotting for each and every variable in
dummy df
plt.figure(figsize=(20,10))
df_dummy.corr()['Churn'].sort_values(ascending = False).plot(kind='bar')

#numerically represents correlation with churn class attribute
df_dummy.corr()['Churn'].sort_values(ascending = False)

# modified data frame import as a csv
df_dummy.to_csv('new_churn.csv')

import pandas as pd
from sklearn import metrics
from sklearn.model_selection import train_test_split
from sklearn.metrics import recall_score
from sklearn.metrics import classification_report
from sklearn.metrics import confusion_matrix
from sklearn.tree import DecisionTreeClassifier
from imblearn.combine import SMOTEENN

#open newly created csv
df_model=pd.read_csv("/content/new_churn.csv")
df_model



#drop the nunamed column
df_model=df_model.drop("Unnamed: 0",axis=1)

x_values=df_model.drop("Churn",axis=1)

y_value=df_model["Churn"]
y_value

# breaking x,y into test and train (test- 20% train-80%)
```

```
x_train,x_test,y_train,y_test=train_test_split(x_values,y_value,test_size=
0.2)

# checking how many churn and non churned customes are there
y_value.value_counts().plot(kind='bar',figsize=(8,6))
plt.xlabel("churn values")
plt.ylabel("count")
plt.title("churn column")

#checking balance of data set
100* y_value.value_counts()/len(y_value)

sm = SMOTEENN()
X_resampled, y_resampled = sm.fit_resample(x_values,y_value)

#checking rebalanced data percentage of churn class
100* y_resampled.value_counts()/len(y_resampled)



"""# **Model Build**

##**Decision tree**
"""

model_dt=DecisionTreeClassifier(criterion = "gini",random_state=0  )

# random_state = 100,max_depth=6, min_samples_leaf=8

model_dt.fit(x_train,y_train)

y_pred=model_dt.predict(x_test)
y_pred

model_dt.score(x_test,y_test)

print(classification_report(y_test, y_pred))

xr_train,xr_test,yr_train,yr_test=train_test_split(X_resampled,
y_resampled,test_size=0.2)

model_resampled_dt=DecisionTreeClassifier(criterion =
"gini",random_state=0  )
```

```
model_resampled_dt.fit(xr_train,yr_train)

y_prediction_resampled_dt=model_resampled_dt.predict(xr_test)

model_resampled_dt.score(xr_test,yr_test)

print(classification_report(yr_test,y_prediction_resampled_dt))

confusion_matrix(yr_test,y_prediction_resampled_dt)

"""###**RandomForest classify**"""

from sklearn.ensemble import RandomForestClassifier

random_forest_model=RandomForestClassifier()

random_forest_model.fit(x_train,y_train)
y_predict_value_random_forest=random_forest_model.predict(x_test)
random_forest_model.score(x_test,y_test)

print(classification_report(y_test,y_predict_value_random_forest))

"""**here even though we had 81% accuracy still our minority class f1
score recall precision still very low because of imbalanced data set.now
let's rebalance the data set using upsampling + down sampling
technique(SMOTEENN)**

**here below we don't need to re implement SMOTEEN() because we already
implemented SMOTEEN() and rebalanced  dataset in data analysis.so we can
use that dataset here.**
"""


100* y_resampled.value_counts()/len(y_resampled)

xr_train,xr_test,yr_train,yr_test=train_test_split(X_resampled,
y_resampled,test_size=0.2)

model_rf_resampled=RandomForestClassifier()
model_rf_resampled.fit(xr_train,yr_train)
yr_rf_predicted_values=model_rf_resampled.predict(xr_test)
model_rf_resampled.score(xr_test,yr_test)
```

```
print(classification_report(yr_test,yr_rf_predicted_values))

confusion_matrix(yr_test,yr_rf_predicted_values)

cm = confusion_matrix(yr_test,yr_rf_predicted_values)
# Plot the confusion matrix using seaborn
sys.heatmap(cm, annot=True, cmap='Blues', fmt='g')
plt.xlabel('Predicted')
plt.ylabel('True')
plt.title('Confusion Matrix')
plt.show()



"""##**Logistic Regression**"""

from sklearn.linear_model import LogisticRegression

model_lr=LogisticRegression(solver='lbfgs', max_iter=1000)
model_lr.fit(x_train,y_train)

y_lr_predicted_value=model_lr.predict(x_test)

# getting accuracy
model_lr.score(x_test,y_test)

y_lr_predicted=model_lr.predict(x_test)
print(classification_report(y_test,y_lr_predicted))

"""**here even though we had 81% accuracy still our minority class f1
score recall precision still very low because of imbalanced data set.now
let's rebalance the data set using upsampling + down sampling
technique(SMOTEENN)**

**here below we don't need to re implement SMOTEEN() because we already
implemented SMOTEEN() and realanced teh dataset in decision tree
classifier.so we can use that dataset here.**
"""

#xr_lr_train,xr_lr_test,yr_lr_train,yr_lr_test=train_test_split(xr_lr_valu
es,yr_lr_value)
model_lr_resampled=LogisticRegression(solver='lbfgs',max_iter=10000)
```

```
model_lr_resampled.fit(xr_train,yr_train)
model_lr_resampled.score(xr_test,yr_test)

yr_lr_predicted_value=model_lr_resampled.predict(xr_test)
print(classification_report(yr_test,yr_lr_predicted_value))

confusion_matrix(yr_test,yr_lr_predicted_value)

##**XGBooost (gradient boost)**"""

from xgboost import XGBClassifier

model_xgb=XGBClassifier(use_label_encoder=False,eval_metrics='mlogloss')
model_xgb.fit(x_values,y_value)
model_xgb.score(x_test,y_test)

y_xgb_predicted_value=model_xgb.predict(x_test)

print(classification_report(y_test,y_xgb_predicted_value))

"""**even though accuracy is 83% ,customer churn class(1) has low f1 score
,precision,accuracy.so we use SMOTEEN technique to rebalance the data
set**

**here below we don't need to re implement SMOTEEN() because we already
implemented SMOTEEN() and realanced teh dataset in decision tree
classifier.so we can use that dataset here.**
"""

model_xgb_resampled=XGBClassifier(use_label_encoder=False,eval_metrics='ml
ogloss')

model_xgb_resampled.fit(xr_train,yr_train)
yr_xgb_predicted_val=model_xgb_resampled.predict(xr_test)
model_xgb_resampled.score(xr_test,yr_test)

print(classification_report(yr_test,yr_xgb_predicted_val))

metrics.accuracy_score(yr_test,yr_xgb_predicted_val)

confusion_matrix(yr_test,yr_xgb_predicted_val)
```