Customer Churn Prediction

- **Submission by Aishwarya B(battulaaishwarya@gmail.com)**

- **Understanding Problem Statement**

Decreasing the Customer Churn is a key goal for any business. Predicting Customer Churn (also known as Customer Attrition) represents an additional potential revenue source for any business. Customer Churn impacts the cost to the business. Higher Customer Churn leads to loss in revenue and the additional marketing costs involved with replacing those customers with new ones.

In this challenge, as a data scientist of a bank, you are asked to analyze the past data and predict whether the customer will churn or not in the next 6 months. This would help the bank to have the right engagement with customers at the right time.

- **Objective**

Our objective is to build a machine learning model to predict whether the customer will churn or not in the next six months.

- **Training set**

train.csv contains the customer demographics and past activity with the bank. And also the target label representing whether the customer will churn or not.

Variable	**Description**
ID	Unique Identifier of a row
Age	Age of the customer
Gender	Gender of the customer (Male and Female)
Income	Yearly income of the customer
Balance	Average quarterly balance of the customer
Vintage	No. of years the customer is associated with bank
Transaction_Status	Whether the customer has done any transaction in the past 3 months or not
Product_Holdings	No. of product holdings with the bank

Credit_Card	Whether the customer has a credit card or not
Credit_Category	Category of a customer based on the credit score
Is_Churn	Whether the customer will churn in next 6 months or not

- **Test set**

test.csv contains the customer demographics and past activity with the bank. And you need to predict whether the customer will churn or not.

Variable	**Description**
ID	Unique Identifier of a row
Age	Age of the customer
Gender	Gender of the customer (Male and Female)
Income	Yearly income of the customer
Balance	Average quarterly balance of the customer
Vintage	No. of years the customer is associated with bank
Transaction_Status	Whether the customer has done any transaction in the past 3 months or not
Product_Holdings	No. of product holdings with the bank
Credit_Card	Whether the customer has a credit card or not
Credit_Category	Category of a customer based on the credit score

- **Data Preprocessing and Feature Engineering**

1) Make sure the training set does have any missing values

```
#Check whether there are missing values in the train data

train_df.isnull().sum()
```

```
ID                    0
Age                   0
Gender                0
Income                0
Balance               0
Vintage               0
Transaction_Status    0
Product_Holdings      0
Credit_Card           0
Credit_Category       0
Is_Churn              0
dtype: int64
```

2) Convert Categorical values to numeric values : such as ‘Gender’, ‘Income’, ‘Credit_Category’,’Product_Holdings’

```
# Converting the "Gender" categorical variable to numeric value
df1=pd.get_dummies(final_df['Gender'])
df1.head()
```

	Female	Male
0	1	0
1	1	0
2	1	0
3	1	0
4	1	0

```
final_df=pd.concat([df1,final_df],axis=1)
```

```
final_df.drop('Gender',axis=1,inplace=True)
```

```
final_df['Income'].unique()
```

```
array(['5L - 10L', 'Less than 5L', 'More than 15L', '10L - 15L'],
      dtype=object)
```

```
#Perform label encoding on the feature Income
final_df['Income']=final_df['Income'].map({'Less than 5L':0,'5L - 10L':1,'10L - 15L':2,'More than 15L':3})
final_df.head()
```

	Female	Male	ID	Age	Income	Balance	Vintage	Transaction_Status	Product_Holdings	Credit_Card	Credit_Category	Is_Churn
0	1	0	84e2fcc9	36	1	563266.44	4	0	1	0	Average	1.0
1	1	0	57fea15e	53	0	875572.11	2	1	1	1	Poor	0.0
2	1	0	8df34ef3	35	3	701607.06	2	1	2	0	Poor	0.0
3	1	0	c5c0788b	43	3	1393922.16	0	1	2	1	Poor	1.0
4	1	0	951d69c4	39	3	893146.23	1	1	1	1	Good	1.0

3) **Label encoding and One-Hot encoding** are the Feature Engineering techniques used to convert the categorical values to numeric values
4) Normalized the values in the train dataset using sklearn StandardScaler before fitting the data to the classifiers.

```
#Normalizing the dataset
from sklearn import preprocessing
X = preprocessing.StandardScaler().fit(X).transform(X)
X[0:5]
```

```
array([[-0.52970736, -0.42556653, -0.46794917, -1.0903853 ,  1.0903853 ,
         1.19955573, -1.03209369, -0.97755572, -1.4069082 ,  0.29068519],
       [ 1.22558098, -1.3682772 ,  0.13762758, -1.0903853 ,  1.0903853 ,
        -0.17154183,  0.96890428, -0.97755572,  0.71077843, -0.96048408],
       [-0.63295962,  1.45985481, -0.1996996 , -1.0903853 ,  1.0903853 ,
        -0.17154183,  0.96890428,  0.77089655, -1.4069082 , -0.96048408],
       [ 0.19305843,  1.45985481,  1.14273497, -1.0903853 ,  1.0903853 ,
        -1.54263939,  0.96890428,  0.77089655,  0.71077843, -0.96048408],
       [-0.21995059,  1.45985481,  0.1717047 , -1.0903853 ,  1.0903853 ,
        -0.85709061,  0.96890428, -0.97755572,  0.71077843,  1.54185447]])
```

5) Also performed **auto-resampling** on the training data since the dataset is imbalanced i.e; the positive cases (1) are more than the negative sample cases(0).

```
#Performing auto-resampling on the train dataset since it is imbalanced i.e; the positive cases are higher than the negative case
from imblearn.over_sampling import SMOTE
method = SMOTE(sampling_strategy='auto')
```

```
X_resampled, y_resampled = method.fit_resample(X, y)
```

- **Training the model**

1) Cross Validation using train_test_split has been performed in order to select the best classifier based on the performance on the training data.

2) The models that have been tested are: Logistic Regression, SVM Classifier, KNN Classifier, Decision Tree, Random Forest Classifier.

```
#Using Logistic Regression classifier
from sklearn.linear_model import LogisticRegression
LR = LogisticRegression(C=0.01, solver='liblinear').fit(X_train,y_train)
LR
```

```
LogisticRegression(C=0.01, solver='liblinear')
```

```
#Using SVM classifier
from sklearn import svm
clf = svm.SVC(kernel='rbf')
clf.fit(X_train, y_train)
```

```
SVC()
```

```
#Using KNN classifier
from sklearn.neighbors import KNeighborsClassifier
k = 5
neigh = KNeighborsClassifier(n_neighbors = k).fit(X_train,y_train)
neigh
```

```
KNeighborsClassifier()
```

```
#Using Decision Tree classifier
from sklearn.tree import DecisionTreeClassifier
drugTree = DecisionTreeClassifier(criterion="entropy", max_depth = 20)
drugTree
```

```
DecisionTreeClassifier(criterion='entropy', max_depth=20)
```

```
#Using random Forest classifier
from sklearn.ensemble import RandomForestClassifier
classifier= RandomForestClassifier(n_estimators= 15, criterion="entropy")
classifier.fit(X_train, y_train)
```

```
RandomForestClassifier(criterion='entropy', n_estimators=15)
```

3) In case of each classifier the hyperparameter tuning has been performed and the model accuracy was calculated based on macro F1-score.

Model	Macro F1-score
Logistic Regression	0.483631171
SVM Classifier	0.438344595
KNN Classifier	0.516397261
Decision Tree Classifier	0.531599673
Random Forest Classifier	0.514616079

From the above table it is clear that Decision Tree is performing the best since Decision Tree tend to perform better than other models when it comes to imbalanced datasets.

- **Testing the model**

  1) Since it was found that Decision Tree is performing the best, the predictions on the test dataset were made using the Decision tree classifier.

```
#Making the predictions using Decision Tree classifier model
yhat = drugTree.predict(df)
```

  2) The Final output was stored in a new_df dataframe and written to a sample_output.csv file

```
#Writing the output to a .csv file
new_df.to_csv('sample_output.csv',index=False)
```