In [1]:
```
#importing necessary libraries
import pandas as pd
import numpy as np
from sklearn import ensemble
from sklearn.metrics import f1_score
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import train_test_split
from sklearn.model_selection import GridSearchCV
from sklearn.metrics import confusion_matrix, classification_report ,accuracy_score
from imblearn.combine import SMOTETomek
```

In [2]:
```
#import train and test data
train= pd.read_csv("train.csv")
test =pd.read_csv('test.csv')
```

In [3]:
```
#shape of train and test data
train.shape , test.shape
```

Out[3]:
```
((6650, 11), (2851, 10))
```

In [4]:
```
#concatenate train and test data to build a model
data = pd.concat([train,test], axis=0).reset_index(drop=True)
# shape of data
data.shape
```

Out[4]:
```
(9501, 11)
```

In [5]:
```
#ischurn became float type
data.head()
```

Out[5]:

	ID	Age	Gender	Income	Balance	Vintage	Transaction_Status	Product_Holdings	Credit_Card	C
0	84e2fcc9	36	Female	5L - 10L	563266.44	4	0	1	0	
1	57fea15e	53	Female	Less than 5L	875572.11	2	1	1	1	
2	8df34ef3	35	Female	More than 15L	701607.06	2	1	2	0	
3	c5c0788b	43	Female	More than 15L	1393922.16	0	1	2	1	
4	951d69c4	39	Female	More than 15L	893146.23	1	1	1	1	

In [6]:
```
label = {'Less than 5L':1, '5L - 10L': 2, '10L - 15L':3, 'More than 15L':4 }

data['Income']= data.Income.map(label)
```

In [7]:
```
data.head()
```

Out[7]:

	ID	Age	Gender	Income	Balance	Vintage	Transaction_Status	Product_Holdings	Credit_Card	C
0	84e2fcc9	36	Female	2	563266.44	4	0	1	0	
1	57fea15e	53	Female	1	875572.11	2	1	1	1	
2	8df34ef3	35	Female	4	701607.06	2	1	2	0	
3	c5c0788b	43	Female	4	1393922.16	0	1	2	1	
4	951d69c4	39	Female	4	893146.23	1	1	1	1	

In [8]:
```
# one hot encoding to convert cateorical data into numeric
data=pd.get_dummies(data ,columns = ['Gender','Product_Holdings','Credit_Category'],drop_firs
data.head()
```

Out[8]:

	ID	Age	Income	Balance	Vintage	Transaction_Status	Credit_Card	Is_Churn	Gender_Female	G
0	84e2fcc9	36	2	563266.44	4	0	0	1.0	1	
1	57fea15e	53	1	875572.11	2	1	1	0.0	1	
2	8df34ef3	35	4	701607.06	2	1	0	0.0	1	
3	c5c0788b	43	4	1393922.16	0	1	1	1.0	1	
4	951d69c4	39	4	893146.23	1	1	1	1.0	1	

In [9]:
```
data.head(2)
```

Out[9]:

	ID	Age	Income	Balance	Vintage	Transaction_Status	Credit_Card	Is_Churn	Gender_Female	Ger
0	84e2fcc9	36	2	563266.44	4	0	0	1.0	1	
1	57fea15e	53	1	875572.11	2	1	1	0.0	1	

In [10]:
```
#split back train and test data
train_proc , test_proc = data[:train.shape[0]] , data[train.shape[0]:].reset_index(drop=True)

features = [c for c in train_proc.columns if c not in ['ID','Is_Churn']]
```

In [11]:
```
train_proc.head(2)
```

Out[11]:

	ID	Age	Income	Balance	Vintage	Transaction_Status	Credit_Card	Is_Churn	Gender_Female	Ger
0	84e2fcc9	36	2	563266.44	4	0	0	1.0	1	
1	57fea15e	53	1	875572.11	2	1	1	0.0	1	

In [12]:
```
test_proc.head(2)
```

Out[12]:

	ID	Age	Income	Balance	Vintage	Transaction_Status	Credit_Card	Is_Churn	Gender_Female	G
0	55480787	50	4	1008636.39	2	1	1	NaN	1	
1	9aededf2	36	2	341460.72	2	0	1	NaN	0	

In [13]:
```
#imbalanced data
data['Is_Churn'].value_counts()
```

Out[13]:
```
0.0    5113
1.0    1537
Name: Is_Churn, dtype: int64
```

In [14]:
```
# split the train dataset into train and validation sets (80% belongs to train and 20% test)

trn , val = train_test_split(train_proc, test_size=0.2 , random_state=1)

#splitting the trn data to train and validate model

x_trn, x_val = trn[features],val[features]

y_trn, y_val = trn['Is_Churn'],val['Is_Churn']

#for testing our data from test.csv
X_test = test_proc[features]
```

In [15]:
```
x_trn.shape , y_trn.shape
```

Out[15]:
```
((5320, 14), (5320,))
```

In [16]:
```
y_trn.value_counts()
# unbalaced data
```

Out[16]:
```
0.0    4083
1.0    1237
Name: Is_Churn, dtype: int64
```

In [17]:
```
# since the data is imbalance we will balance the data
state = np.random.RandomState(42)
X_outliers = state.uniform(low=0, high =1, size=(x_trn.shape[0], x_trn.shape[1]))
```

In [18]:
```
#implement over_samplng for handling Imbalanced

smk = SMOTETomek(random_state=42)
X_res, Y_res= smk.fit_resample(x_trn, y_trn)

print(X_res.shape)
print(Y_res.shape)
```

```
(6072, 14)
(6072,)
```

In [19]:
```
Y_res.value_counts()
```

Out[19]:
```
0.0    3036
1.0    3036
Name: Is_Churn, dtype: int64
```

In [20]:
```
Y_res.head()
```

Out[20]:
```
0    1.0
1    0.0
2    0.0
3    1.0
4    0.0
Name: Is_Churn, dtype: float64
```

In [21]:
```
X_res.head()
```

Out[21]:

	Age	Income	Balance	Vintage	Transaction_Status	Credit_Card	Gender_Female	Gender_Male	Product_
0	37	4	36045.00	4	0	0	0	1	
1	30	2	2151102.15	1	1	0	1	0	
2	29	1	210585.06	0	1	0	0	1	
3	43	1	446160.06	1	0	1	1	0	
4	31	1	99873.00	1	1	1	0	1	

In [22]:
```
#using Random forest classifier, fit the model and get the accuracy on training data
rfc = RandomForestClassifier()
```

In [23]:
```
#hyperparameter tuning using gridsearchcv
forest_params = {'max_depth':range(1,5), 'max_features': range(5,10)}

clf = GridSearchCV(rfc, forest_params, cv = 10, scoring='accuracy')
```

In [24]:
```
clf.fit(X_res,Y_res)
```

Out[24]:
```
GridSearchCV(cv=10, estimator=RandomForestClassifier(),
             param_grid={'max_depth': range(1, 5),
                         'max_features': range(5, 10)},
             scoring='accuracy')
```

In [25]:
```
print(clf.best_params_)
```

```
{'max_depth': 4, 'max_features': 5}
```

In [26]:
```
print(clf.best_score_)
```

```
0.7699950684990896
```

In [27]:
```
#predcton on valdation data
y_pred = clf.predict(x_val)
```

In [28]:
```
y_pred
```

Out[28]:
```
array([0., 0., 0., ..., 0., 1., 0.])
```

In [29]:
```
#check validation score less compared to test data
clf.score(x_val,y_val)
```

Out[29]:
```
0.7255639097744361
```

In [40]:
```
#pring the classification report and test data
print(classification_report(y_val,y_pred))
```

```
              precision    recall  f1-score   support

         0.0       0.80      0.86      0.83      1030
         1.0       0.35      0.26      0.30       300

    accuracy                           0.73      1330
   macro avg       0.58      0.56      0.57      1330
weighted avg       0.70      0.73      0.71      1330
```

In [31]:
```
# evalution metric - f1 score with average macro
print(f1_score(y_val, y_pred, average='macro'))
```

```
0.565651531649862
```

In [32]:
```
#predict target values for X_test
#X_test = test_proc[features]
```

```
Y_test = clf.predict(X_test)
Y_test
```

Out[32]:
```
array([0., 0., 0., ..., 1., 1., 0.])
```

In [33]:
```
#predctng the probability for class  zero and one respectively
y_pr =  clf.predict_proba(X_test)
```

In [34]:
```
#the probality to churn to predct the analsis in near future
test['Probability for churn'] = y_pr[:,1]
```

In [35]:
```
test.tail()
```

Out[35]:

	ID	Age	Gender	Income	Balance	Vintage	Transaction_Status	Product_Holdings	Credit_Car
2846	19e40adf	40	Female	10L - 15L	1338458.22	0	0	1	
2847	52d5bc8d	48	Female	More than 15L	1448280.27	0	1	2	
2848	f708121b	59	Male	More than 15L	1100555.64	3	0	1	
2849	f008715d	34	Female	5L - 10L	1502818.92	2	0	1	
2850	36b81f59	61	Female	10L - 15L	913787.73	0	1	2	

In [36]:
```
df = pd.DataFrame({'ID':test_proc['ID'] , 'Is_Churn' :Y_test })
df.head()
```

Out[36]:

	ID	Is_Churn
0	55480787	0.0
1	9aededf2	0.0
2	a5034a09	0.0
3	b3256702	0.0
4	dc28adb5	0.0

In [42]:
```
#convert to csv file for submission
df.to_csv('sample_sub_final.csv',index=False)
```

In [ ]:
```
```