HAPPY LOAN BANK CREDIT CARD PROBLEM

IMPORTING ALL NECCESSARY LIBRARIES AND MODULES

In [36]:
```
import matplotlib.pyplot as plt
import pandas as pd
import numpy as np
import seaborn as sns

from sklearn.preprocessing import LabelEncoder
from sklearn.model_selection import train_test_split
from sklearn import metrics
from sklearn.linear_model import LinearRegression

import warnings
warnings.filterwarnings('always')
warnings.filterwarnings('ignore')
```

In [37]:
```
from sklearn.preprocessing import MinMaxScaler
from sklearn.linear_model import LogisticRegression

import matplotlib.pyplot as plt
import seaborn as sn

from sklearn.ensemble import GradientBoostingRegressor
from sklearn.metrics import roc_auc_score
```

READING THE DATASETS

In [38]:
```
train = pd.read_csv("train_s3TEQDk.csv")
test = pd.read_csv("test_mSzZ8RL.csv")
```

In [39]:
```
train.shape,test.shape
```

Out[39]: ((245725, 11), (105312, 10))

In [40]:
```
train.isnull().sum()
```

Out[40]:
```
ID                         0
Gender                     0
Age                        0
Region_Code                0
Occupation                 0
Channel_Code               0
Vintage                    0
Credit_Product         29325
Avg_Account_Balance        0
Is_Active                  0
Is_Lead                    0
dtype: int64
```

In [41]:
```
test.isnull().sum()
```

Out[41]:
```
ID                        0
Gender                    0
Age                       0
Region_Code               0
Occupation                0
Channel_Code              0
Vintage                   0
Credit_Product        12522
Avg_Account_Balance       0
Is_Active                 0
dtype: int64
```

TRAIN FILE HAS 29325 NULL VALUES IN CREDIT_PRODUCT

TEST FILE HAS 12522 NULL VALUES IN CREDIT_PRODUCT

In [42]:
```
print(train['Credit_Product'].value_counts())
print('****************************************')
print(test['Credit_Product'].value_counts())
```

```
No     144357
Yes     72043
Name: Credit_Product, dtype: int64
****************************************
No     61608
Yes    31182
Name: Credit_Product, dtype: int64
```

CREDIT_PRODUCT HAS AN CATEGORICAL TYPE OF YES/NO

FILLING MISSING VALUES

FILLING VALUES BY ANY METHOD LEADS TO DECREASE IN ACCURACY SO WE DECIDE TO DROP THE COLUMNS HAVING NULL VALUES

In [43]:
```
#Imputing with Mode
train['Credit_Product']= train['Credit_Product'].dropna()
test['Credit_Product']= test['Credit_Product'].dropna()
```

WE ALSO DROP THE UNNECCESSARY COLUMNS THAT HAVE ALMOST NO RELATIONSHIP WITH THE PREDICTION

In [44]:
```
train = train.drop(["ID","Gender","Region_Code","Occupation","Channel_Code"],axis=1)
test = test.drop(["ID","Gender","Region_Code","Occupation","Channel_Code"],axis=1)
```

In [45]:
```
train.columns
```

Out[45]:
```
Index(['Age', 'Vintage', 'Credit_Product', 'Avg_Account_Balance', 'Is_Active',
       'Is_Lead'],
      dtype='object')
```

In [46]:
```
test.columns
```

Out[46]:
```
Index(['Age', 'Vintage', 'Credit_Product', 'Avg_Account_Balance', 'Is_Active'], dtype='object')
```

ENCODE THE DATASETS

In [47]:
```
le = LabelEncoder()
var_mod = train.select_dtypes(include='object').columns
for i in var_mod:
    train[i] = le.fit_transform(train[i])

for i in var_mod:
    test[i] = le.fit_transform(test[i])
```

Seperate Features and Target

In [48]:
```
# Seperate Features and Target
X= train.drop(columns = ['Is_Lead'], axis=1)
y= train['Is_Lead']
```

SPLITTING INTO TRAIN AND VALIDATION SET FOR TRAINING THE MODEL

In [49]:
```
# 20% data as validation set
X_train,X_valid,y_train,y_valid = train_test_split(X,y,test_size=0.1,random_state=42
```

SCALING THE TRAIN TEST AND VALIDATION SETS

In [50]:
```
scaler = MinMaxScaler(feature_range=(0,1))
```

In [51]:
```
scaledtrainX = scaler.fit_transform(X_train)
scaledtestX = scaler.fit_transform(X_valid)
scaledx= scaler.fit_transform(X)

scaledtest= scaler.fit_transform(test)
```

IMPORT MODEL ALGORITHMS AND CHECK WHICH IS GIVING HIGHEST ROC_AUC_SCORE

In [52]:
```
from sklearn.linear_model import Ridge, Lasso, LinearRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.neighbors import KNeighborsRegressor
from sklearn.naive_bayes import GaussianNB
from sklearn.ensemble import RandomForestClassifier,RandomForestRegressor
from xgboost import XGBClassifier
```

In [53]:
```
algos = [
          LogisticRegression(),
         DecisionTreeClassifier(max_depth=3),
         XGBClassifier(learning_rate=0.01, n_estimators=25, max_depth=15,gamma=0.6,
                    reg_lambda=2, booster='dart', colsample_bylevel=0.6, colsample_b
         RandomForestClassifier(max_depth=2, random_state=0),
         GaussianNB()
        ]

algos2 = [GradientBoostingRegressor(n_estimators=100, learning_rate=1.0,
                                    max_depth=1, random_state=0),
           KNeighborsRegressor(),
          Lasso(), Ridge(), LinearRegression()
         ]
```

```
names = [
         'K Neighbors Regressor', 'Decision Tree Regressor', 'XGB Classifier','Rando
         'GaussianNB()',
        ]
names2 = ['GBR','KNN','LASSO','RIDGE','Linear REgression']
roc_auc_score_list = []
```

In [54]:

```
for name in algos:
    model = name.fit(scaledtrainX,y_train)
    roc= roc_auc_score(y_valid, model.predict_proba(scaledtestX)[:, 1])
    print(name,roc)
```

```
LogisticRegression() 0.8500987006763038
DecisionTreeClassifier(max_depth=3) 0.8441537453733152
[15:34:30] WARNING: C:/Users/Administrator/workspace/xgboost-win64_release_1.4.0/sr
c/learner.cc:1095: Starting in XGBoost 1.3.0, the default evaluation metric used wit
h the objective 'binary:logistic' was changed from 'error' to 'logloss'. Explicitly
set eval_metric if you'd like to restore the old behavior.
XGBClassifier(base_score=0.5, booster='dart', colsample_bylevel=0.6,
              colsample_bynode=0.5, colsample_bytree=0.6, gamma=0.6, gpu_id=-1,
              importance_type='gain', interaction_constraints='',
              learning_rate=0.01, max_delta_step=0, max_depth=15,
              min_child_weight=1, missing=nan, monotone_constraints='()',
              n_estimators=25, n_jobs=4, num_parallel_tree=1, random_state=27,
              reg_alpha=0, reg_lambda=2, scale_pos_weight=1, seed=27,
              subsample=0.52, tree_method='exact', validate_parameters=1,
              verbosity=None) 0.8571990086219796
RandomForestClassifier(max_depth=2, random_state=0) 0.8393360307875475
GaussianNB() 0.8460025556657739
```

In [55]:

```
clf3 = GradientBoostingRegressor(n_estimators=105, learning_rate=0.9,
                                 max_depth=1, random_state=22).fit(scaledtrainX,y_tr
roc_auc_score(y_valid, clf3.predict(scaledtestX),multi_class='ovo')
```

Out[55]: 0.8572723438830566

GBR HAS HIGHEST ROC SCORE

BUILD THE MODEL USING GBR AND SAVE IT TO CSV FILE

In [42]:

```
submission01 = pd.read_csv('sample_submission_eyYijxG.csv')
model = GradientBoostingRegressor(n_estimators=105, learning_rate=0.9,
                                  max_depth=1, random_state=22)
model.fit(scaledx, y)
final_predictions = model.predict(scaledtest)
submission01['Is_Lead'] = final_predictions
#only positive predictions for the target variable
submission01['Is_Lead'] = submission01['Is_Lead'].apply(lambda x: 0 if x<0 else x)
submission01.to_csv('my_submission000F.csv', index=False)
```

In [ ]: