Credit Card Behaviour Score Prediction Using Classification and Risk-Based Techniques

Eisa Hasan

OVERVIEW:

The project will focus on improving the credit risk management framework of a bank by developing a forward looking Behaviour Score, that is, a classification model that predicts whether a credit card customer will default in the following month. Based on the historical behavioral data of over 30000 customers a binary classification model has to be set up which helps to predict whether a customer will default in the next billing cycle. Such a model would allow the bank to flag potential defaulters in advance, allowing the bank to adjust the credit exposure, trigger early warnings systems and prioritize risk based actions. Once the data is cleaned and preprocessed then some new meaningful features are added via feature engineering. Then class imbalance is handled by down sampling and class weighting after which model is trained and evaluated by various metrics. The threshold is then tuned to enhance the performance of the model.

EDA AND FINANCIAL INSIGHTS

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

```
Customer_ID            0
marriage               0
sex                    0
education              0
LIMIT_BAL              0
age                  126
pay_0                  0
pay_2                  0
pay_3                  0
pay_4                  0
pay_5                  0
pay_6                  0
Bill_amt1              0
Bill_amt2              0
Bill_amt3              0
Bill_amt4              0
Bill_amt5              0
Bill_amt6              0
pay_amt1               0
pay_amt2               0
pay_amt3               0
pay_amt4               0
pay_amt5               0
pay_amt6               0
AVG_Bill_amt           0
PAY_TO_BILL_ratio      0
next_month_default     0
dtype: int64
```

We have some missing values in the age column.So we can replace them with the mean of the remaining values. Also we can change the datatype of age column to int64.

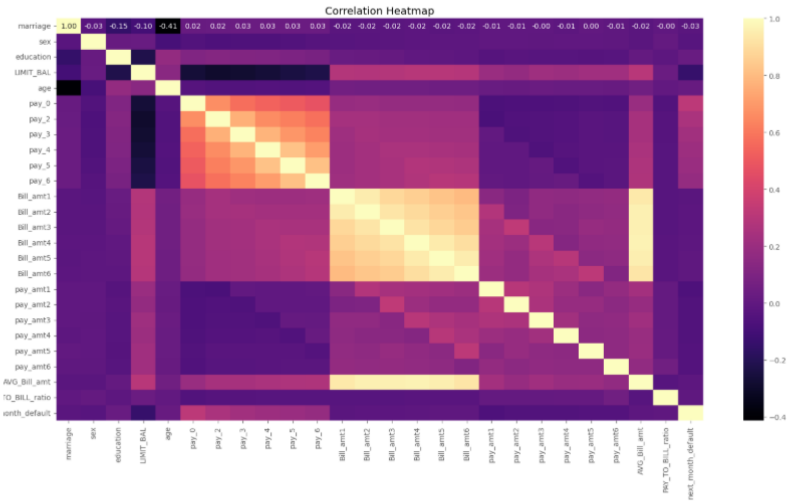
We have some missing values in our age column.So we replaced it with the mean of remaining values. Also id column was dropped since it is not a default predicting factor.

```
df['age'] = df['age'].fillna(value=df['age'].mean())
df['age'] = df['age'].astype('int64')
df[['marriage','sex','education']] = df[['marriage','sex','education']].astype('object')
df=df.drop(columns=['Customer_ID'])
df.info()
```

VISUALISATION

1. CORRELATION HEATMAP: using the matplotlib library we set the features and create the correlation matrix.

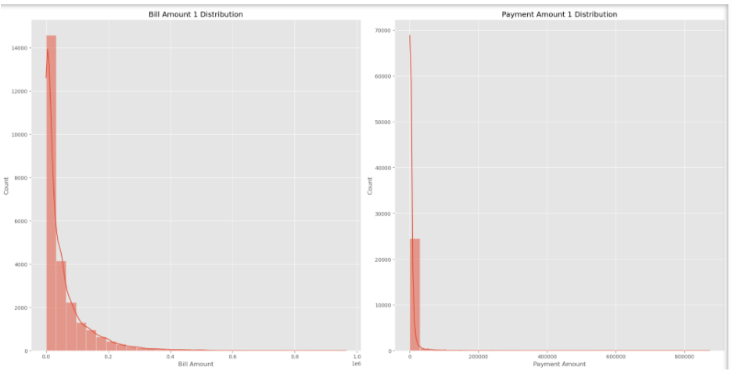
```
import matplotlib.pyplot as plt
import seaborn as sns

corr_matrix=df.corr()
plt.figure(figsize=(20,10))
sns.heatmap(corr_matrix,annot=True,fmt='.2f',cmap='magma')
plt.title('Correlation Heatmap')
plt.show()
```



The heatmap shows a strong correlation among the payment variables(pay_0,pay_2,pay_3,pay_4,pay_5,pay_6) and billing amount variables(bill_amt1,bill_amt2,bill_amt3,bill_amt4,bill_amt5,bill_amt6).the strong correlation among payment variables shows that the payments deliquency is persistent .So if a customer misses payment in the initial months then he is likely to miss payments in the subsequent months as well. Also the strong correlation among billing_amount variables suggests that customers maintain a consistent spending pattern over time.

2. HISTOGRAM :- we create histograms of bill_amount and payment amount of all the months.

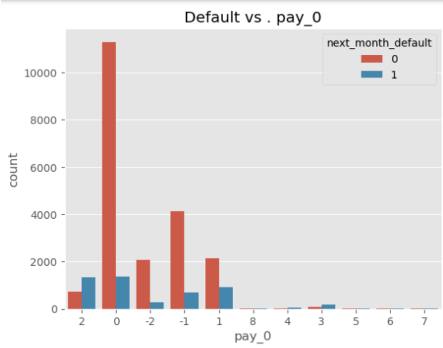
```
import warnings
warnings.filterwarnings("ignore")
for i in range(1, 7):
    plt.figure(figsize = (20, 10))
    plt.subplot(1, 2, 1)
    sns.histplot(df[f'Bill_amt{i}'], bins = 30, kde = True)
    plt.title(f'Bill Amount {i} Distribution')
    plt.xlabel('Bill Amount')
    plt.ylabel('Count')

    plt.subplot(1, 2, 2)
    sns.histplot(df[f'pay_amt{i}'], bins = 30, kde = True)
    plt.title(f'Payment Amount {i} Distribution')
    plt.xlabel('Payment Amount')
    plt.ylabel('Count')

    plt.tight_layout()
    plt.show()
```

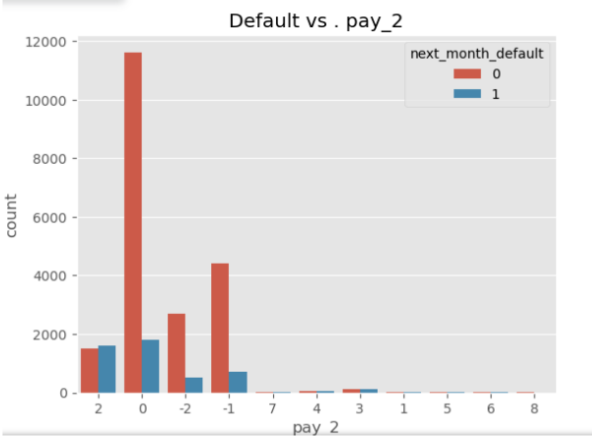


The billing_amount and payment_amount distributions are heavily right skewed suggesting that there is large chunk of customers having relatively small balances and payments .Also the spikes at zero in payment_amount across various months indicate that the customers miss or make partial payments for the previous month marking them as a red flag for default risk .The payment distribution is more concentrated near zero than bill distribution suggesting that the customers are accumulating debt faster than they are paying it increasing their chances of making default in the next billing cycle.

3) COUNTPLOT:- We create countplot of payment_amount of all months for both the classes.

```
payment_history_columns = ['pay_0', 'pay_2', 'pay_3', 'pay_4', 'pay_5', 'pay_6']

for col in payment_history_columns:
    sns.countplot(x=df[col].astype(str),hue=df['next_month_default'].astype(str), data = df)
    plt.title(f'Default vs . {col}')
    plt.show()
```



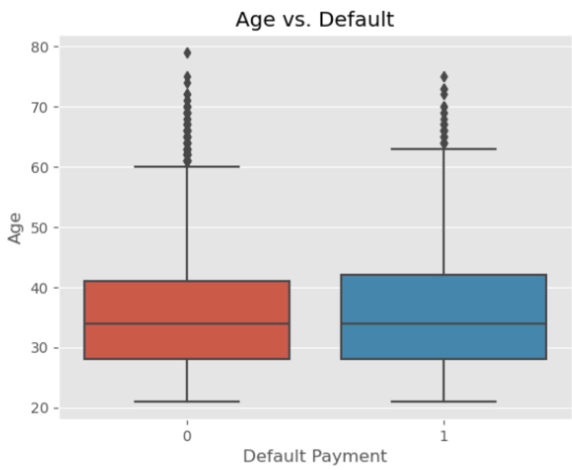


In the initial months (pay_0 , pay_2, pay_3), the sharp distinction between various categories like paid duly (-1), partial payment (0) suggests that any deviation from on time payment significantly increases default risk and such a customer is unable to recover from debt in the subsequent months as well. So repayment behaviour is a strong predictor of default risk. Customers who missed recent payments should be flagged for risk.

4) BOXPLOT:-

A) AGE VS DEFAULT

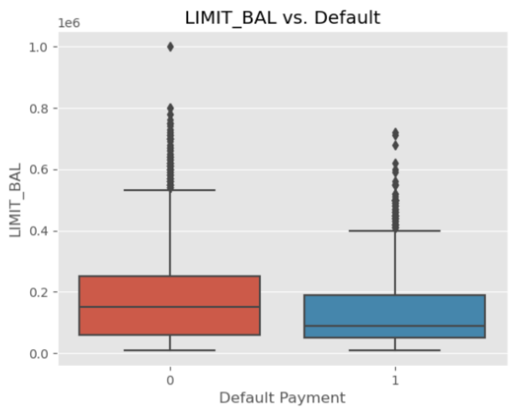
```
sns.boxplot(x = 'next_month_default', y = 'age', data = df)
plt.title('Age vs. Default')
plt.xlabel('Default Payment')
plt.ylabel('Age')
plt.show()
```



the box plot suggests that age is not a strong default predictor. The boxplots for the 2 classes overlap significantly suggesting that default risk exists over all demographics .So ,credit decisions should prioritize financial behaviour over demographic characteristics.

B) LIMIT_BAL VS DEFAULT

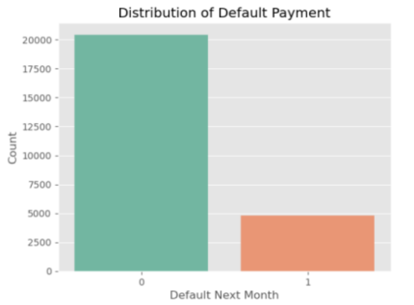
```
sns.boxplot(x = 'next_month_default', y = 'LIMIT_BAL', data = df)
plt.title('LIMIT_BAL vs. Default')
plt.xlabel('Default Payment')
plt.ylabel('LIMIT_BAL')
plt.show()
```



The median of limit_balance for non defaulters is almost double than that for defaulters(default_payment=1).This suggests that credit limit is higher for the creditworthy customers.Also this seperation between the 2 classes shows that credit_limit itself is a strong default risk predictor as it is assigned based on the credit score of the customers.

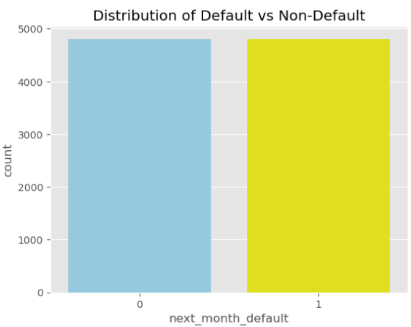
CLASS IMBALANCE

```
sns.countplot(x='next_month_default', data=df, palette='Set2')
plt.title('Distribution of Default Payment')
plt.xlabel('Default Next Month')
plt.ylabel('Count')
plt.show()
```



There is a significant class imbalance between the 2 classes which can significantly affect the performance of our model .So to handle it we have used downsampling technique.

```
from sklearn.utils import resample

# Separate majority (0) and minority (1) classes
df_majority = df[df['next_month_default'] == 0]
df_minority = df[df['next_month_default'] == 1]

# Downsample majority class to match minority
df_majority_downsampled = resample(df_majority,replace=False,n_samples=len(df_minority),random_state=42)

# Combine downsampled majority with minority
df_balanced = pd.concat([df_majority_downsampled, df_minority])

# Shuffle the dataset
df_balanced = df_balanced.sample(frac=1, random_state=42).reset_index(drop=True)

# Check new class distribution
print(df_balanced['next_month_default'].value_counts())
```



MODEL TRAINING

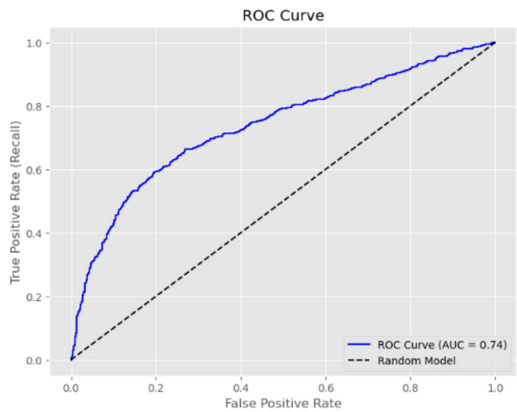
MODEL 1: LOGISTIC REGRESSION

```
# Separate features and target variable
X = df_balanced.drop('next_month_default', axis=1)
y = df_balanced['next_month_default']

# Train-test split
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Feature scaling
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)
```

```python
from sklearn.pipeline import Pipeline
from sklearn.impute import SimpleImputer
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix
from sklearn.metrics import fbeta_score

# Create pipeline
pipeline = Pipeline([
    ('imputer', SimpleImputer(strategy='mean')),
    ('scaler', StandardScaler()),
    ('model', LogisticRegression(max_iter=5000))  # optional class weighting
])

# Fit model
pipeline.fit(X_train_scaled, y_train)

y_prob = pipeline.predict_proba(X_test_scaled)[:, 1]
threshold=0.40
y_pred = (y_prob >= threshold).astype(int)

f2_score = fbeta_score(y_test, y_pred, beta=2)

# Step 7: Evaluate performance
print("Accuracy:", accuracy_score(y_test, y_pred))
print("Confusion Matrix:\n", confusion_matrix(y_test, y_pred))
print("Classification Report:\n", classification_report(y_test, y_pred))
print("f2 score:\n", f2_score)
```

```
Accuracy: 0.6843473738949558
Confusion Matrix:
 [[662 311]
 [296 654]]
Classification Report:
               precision    recall  f1-score   support

           0       0.69      0.68      0.69       973
           1       0.68      0.69      0.68       950

    accuracy                           0.68      1923
   macro avg       0.68      0.68      0.68      1923
weighted avg       0.68      0.68      0.68      1923

f2 score:
 0.6862539349422874
```

```python
from sklearn.metrics import roc_curve, roc_auc_score
import matplotlib.pyplot as plt

y_prob = pipeline.predict_proba(X_test_scaled)[:, 1]

#calculate false positive rate and true positive rate
fpr, tpr, thresholds = roc_curve(y_test, y_prob)

# Step 2: Calculate AUC score
auc_score = roc_auc_score(y_test, y_prob)

# Step 3: Plot ROC Curve
plt.figure(figsize=(8, 6))
plt.plot(fpr, tpr, label=f'ROC Curve (AUC = {auc_score:.2f})', color='blue')
plt.plot([0, 1], [0, 1], 'k--', label='Random Model')  # Diagonal line
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate (Recall)')
plt.title('ROC Curve')
plt.legend(loc='lower right')
plt.grid(True)
plt.show()
```



IMPORTS:

1.We import ‘Logistic Regression’ , from scikit-learn.

2.Pipeline and simple imputer.

3.fbeta_score,confusion_matrix,accuracy_score and classification_report metrics,

4 train_test_split-to split in training and testing data.

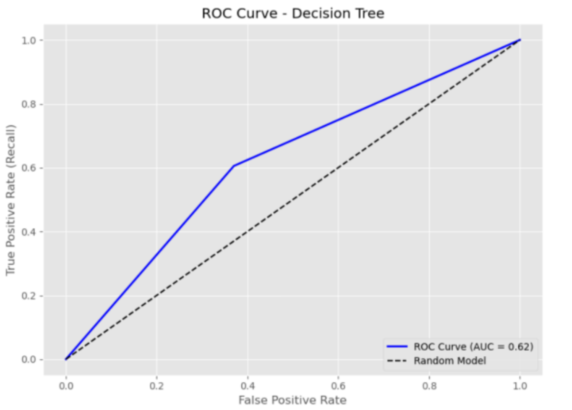
5.Standardscalar.

- X is variable defined by all features of the training df except label, while y=label.
- Now using the 'train_test_split' , 20% data is separated from the original for test purpose. On the basis of this X_train , y_train are passed from the model for the identification of patterns for the data.
- ' X_train_scaled = scaler.fit_transform(X_train)' standardizes the training features.
- ' X_test_scaled = scaler.transform(X_test)'. This applies the same transformation to the test features as learnt from the training data.
- Now we created a logistic regression pipeline and used Simple imputer to fill all the missing values in the balanced scaled data with the mean value.
- y_prob variable stores the probabilities that a customer will default next month.
- Now we have used a threshold value of 0.4 instead of default value of 0.5 and predicted y_pred.
- Now we make predictions on the X_test . how correctly our model is working is then conveyed to us through the f2_score,accuracy and classification report.
- Finally we imported roc_curve and roc_auc_score from sklearn.metrics to draw the ruc curve and get AUC.

2) DECISION TREE

```
from sklearn.tree import DecisionTreeClassifier

# Create pipeline
pipeline = Pipeline([
    ('imputer', SimpleImputer(strategy='mean')),
    ('scaler', StandardScaler()),
    ('model', DecisionTreeClassifier(random_state=42)) ])

# Fit model
pipeline.fit(X_train_scaled, y_train)

# Predict
y_prob = pipeline.predict_proba(X_test_scaled)[:, 1]
threshold=0.40
y_pred = (y_prob >= threshold).astype(int)

# Evaluate performance
f2_score = fbeta_score(y_test, y_pred, beta=2)

print("Accuracy:", accuracy_score(y_test, y_pred))
print("Confusion Matrix:\n", confusion_matrix(y_test, y_pred))
print("Classification Report:\n", classification_report(y_test, y_pred))
print("f2 score:\n", f2_score)
```

```
Accuracy: 0.6177847113884556
Confusion Matrix:
 [[613 360]
 [375 575]]
Classification Report:
              precision    recall  f1-score   support

           0       0.62      0.63      0.63       973
           1       0.61      0.61      0.61       950

    accuracy                           0.62      1923
   macro avg       0.62      0.62      0.62      1923
weighted avg       0.62      0.62      0.62      1923

f2 score:
 0.607180570221753
```

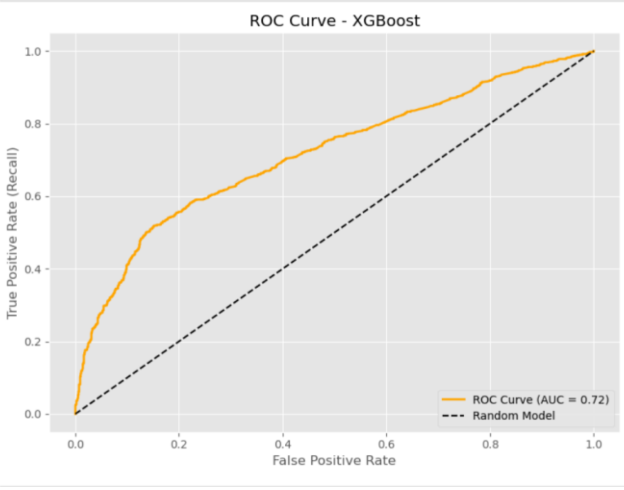


- We imported DecisionTreeClassifier from sklearn.tree and trained the model the same way as we trained our logistic regression model.
- Now we make predictions on the X_test . how correctly our model is working is then conveyed to us through the f2_score,accuracy and classification report.
- Finally we imported roc_curve and roc_auc_score from sklearn.metrics to draw the ruc curve and get AUC.

3) XGBOOST

```
#XG BOOST
from xgboost import XGBClassifier
# Create pipeline
pipeline = Pipeline([
    ('imputer', SimpleImputer(strategy='mean')),
    ('scaler', StandardScaler()),  # Still useful for some numeric stability
    ('model', XGBClassifier(use_label_encoder=False, eval_metric='logloss', random_state=42))
])

# Fit model
pipeline.fit(X_train_scaled, y_train)

# Predict
y_prob = pipeline.predict_proba(X_test_scaled)[:, 1]
threshold=0.40
y_pred = (y_prob >= threshold).astype(int)

# Evaluate performance
f2_score = fbeta_score(y_test, y_pred, beta=2)

print("Accuracy:", accuracy_score(y_test, y_pred))
print("Confusion Matrix:\n", confusion_matrix(y_test, y_pred))
print("Classification Report:\n", classification_report(y_test, y_pred))
print("f2 score:\n", f2_score)
```

```
Accuracy: 0.6708268330733229
Confusion Matrix:
 [[609 364]
 [269 681]]
Classification Report:
               precision    recall  f1-score   support

           0       0.69      0.63      0.66       973
           1       0.65      0.72      0.68       950

    accuracy                           0.67      1923
   macro avg       0.67      0.67      0.67      1923
weighted avg       0.67      0.67      0.67      1923

f2 score:
 0.7027863777089783
```

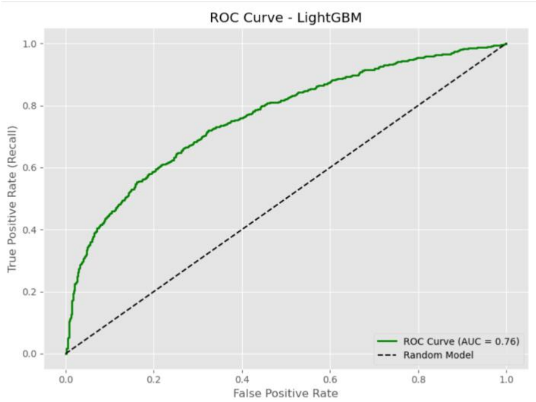


- We imported XGBClassifier from xgboost and trained the model the same way as we trained our logistic regression model.
- Now we make predictions on the X_test . how correctly our model is working is then conveyed to us through the f2_score,accuracy and classification report.
- Finally we imported roc_curve and roc_auc_score from sklearn.metrics to draw the ruc curve and get AUC.

4) LIGHT GBM

```
from lightgbm import LGBMClassifier

pipeline_lgbm = Pipeline([
    ('imputer', SimpleImputer(strategy='mean')),
    ('scaler', StandardScaler()),  # optional for tree models
    ('model', LGBMClassifier(random_state=42))
])

# Fit model
pipeline_lgbm.fit(X_train_scaled, y_train)

# Predict
y_prob = pipeline_lgbm.predict_proba(X_test_scaled)[:, 1]
threshold=0.40
y_pred = (y_prob >= threshold).astype(int)

# Evaluate performance
f2_score = fbeta_score(y_test, y_pred, beta=2)

print("Accuracy:", accuracy_score(y_test, y_pred))
print("Confusion Matrix:\n", confusion_matrix(y_test, y_pred))
print("Classification Report:\n", classification_report(y_test, y_pred))
print("f2 score:\n", f2_score)
```

```
Accuracy: 0.6853874154966199
Confusion Matrix:
 [[604 369]
 [236 714]]
Classification Report:
               precision    recall  f1-score   support

           0       0.72      0.62      0.67       973
           1       0.66      0.75      0.70       950

    accuracy                           0.69      1923
   macro avg       0.69      0.69      0.68      1923
weighted avg       0.69      0.69      0.68      1923

f2 score:
 0.7311079254556625
```



- We imported LGBMClassifier from lightgbm and trained the model the same way as we trained our logistic regression model.
- Now we make predictions on the X_test . how correctly our model is working is then conveyed to us through the f2_score,accuracy and classification report.
- Finally we imported roc_curve and roc_auc_score from sklearn.metrics to draw the ruc curve and get AUC.

EVALUATION AND MODEL SELECTION

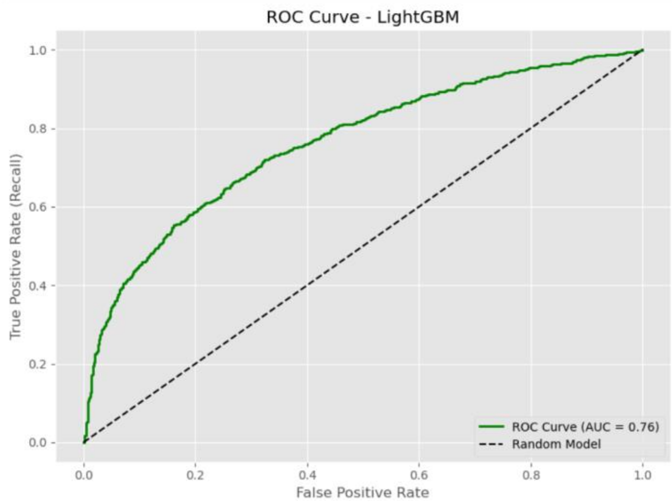
- The evaluation metrics considered while selecting the best performing model are as F1_SCORE,F2_SCORE,ACCURACY,AUC UNDER ROC.
- These factors have been prioritised as they best reflect the credit risk prioritites.
- Special focus was given to **maximise F2_SCORE** as this metric weights recall more than precision.So a high recall would mean that the model is catching most defaulters which helps in preventing financial risk.
- Now for a bank catching actual defaulters is more important than avoiding false alarms. So reducing false negative alarms(predicting customer will not default but ends up defaulting) is more important than reducing false positive alarms (predicting customer will default but he actually doesn't).
- AUC_ROC_CURVE is another important metric which quantifies the model's ability to identify high-risk customers especially when the data is highly unbalanced.It evaluates the model across all thresholds and can be used to tune threshold
- Now accuracy and F2_score reflect credit risk trade offs as increasing F2_score would mean reducing the threshold cutoff but this result in the increase in the number of false positive predictions(prediction customer will default but he

actually doesn't).As a result the accuracy of the model would drop with increasing F2_score.

- So based on these metrics and their importance in real world credit card default prediction **<u>I have chosen LIGHTGBM model with an F2_SCORE OF 0.7311079254556625</u>** to predict the output for the validation data set.

```
Accuracy: 0.6853874154966199
Confusion Matrix:
 [[604 369]
 [236 714]]
Classification Report:
              precision    recall  f1-score   support

           0       0.72      0.62      0.67       973
           1       0.66      0.75      0.70       950

    accuracy                           0.69      1923
   macro avg       0.69      0.69      0.68      1923
weighted avg       0.69      0.69      0.68      1923

f2 score:
 0.7311079254556625
```

(PERFORMANCE OF LIGHT GBM MODEL)



<u>CLASSIFICATION CUTOFF</u>

Now as more emphasis is on increasing F2_score as it prioritises recall, which basically means that the model is more focused on catching actual defaulters(reducing false negative predictions) even at the cost of misclassifying some non defaulters, so it can be achieved by reducing classification threshold from its default value of 0.5 to 0.4 . As a result of this drop in cutoff, more customers are flagged as defaulters reducing the false negative predictions and increasing recall. However we avoid choosing a smaller value than 0.4 as it would result in a poor accuracy as there is a tradeoff between F2_score and accuracy.

<u>BUSINESS IMPLICATION AND SUMMARY</u>

- We realised that false negative predictions are costlier than false positive predictions and as a result we gave more emphasis to recall.
- False positive predictions means that the model predicts that a customer will default but he actually doesn't. This could lead to loss of good customers as it would deny credit or give stricter limits to safe customers which would eventually affect the bank's image and trustworthiness.
- False negative predictions means that the model predicts that a customer will not default but he actually does. It could lead to direct financial loss for the bank.
- By prioritizing high **F2-score** (catching more actual defaulters), the bank minimizes the risk of loan defaults and associated revenue loss.
- The predictions help in distinguishing between low-risk and high-risk customers, guiding more targeted credit strategies.

LEARNINGS

- F2-score optimization gave better real-world utility than plain accuracy, especially for imbalanced data.
- AUC-ROC, F2-score, and Recall are more useful than Accuracy for model evaluation when the data is highly unbalanced.
- Tuning of threshold improved recall and F2_score ensuring fewer defaulters go undetected