PROJECT TITLE: CREDIT CARD DEFAULT PREDICTION

NAME – PRAJWAL SINGH

ENROLLMENT NO.- 22117105

EMAIL- prajwal_s@me.iitr.ac.in

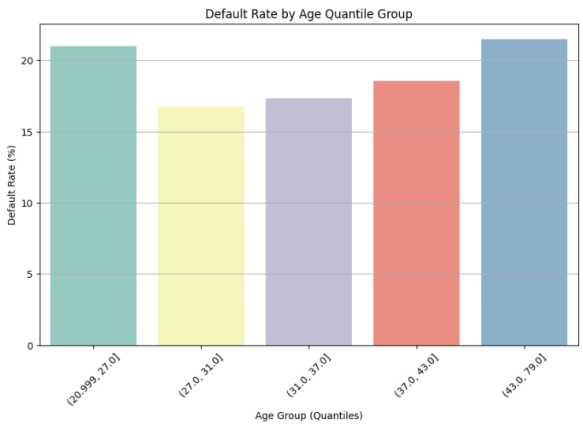
PROJECT OVERVIEW:

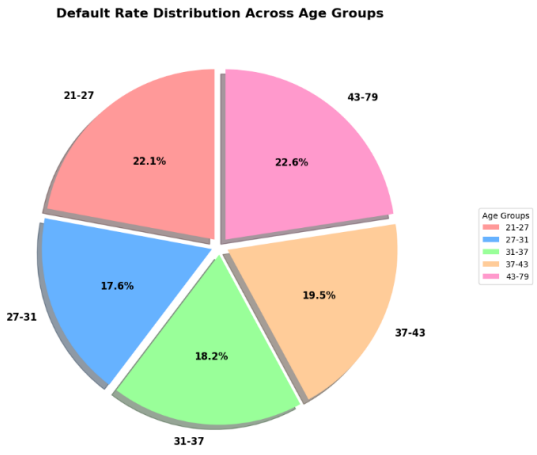
This project was undertaken as part of the Finclub Summer Project 2025 with the objective of supporting Bank A in improving its credit risk management. The bank provided a dataset containing anonymized records of over 30,000 credit card customers. The goal was to predict whether a customer would default
on their credit card payment in the upcoming month. Accurate prediction enables the bank to preemptively reduce risk exposure and enhance operational strategies.

The problem was framed as a binary classification task, with the target variable `next_month_default` taking a value of 1 if the customer defaults and 0 otherwise. In addition to predictive modeling, a significant focus was placed on exploratory data analysis (EDA) to extract financially meaningful insights and to construct new meaningful features.

EXPLORATORY DATA ANALYSIS:

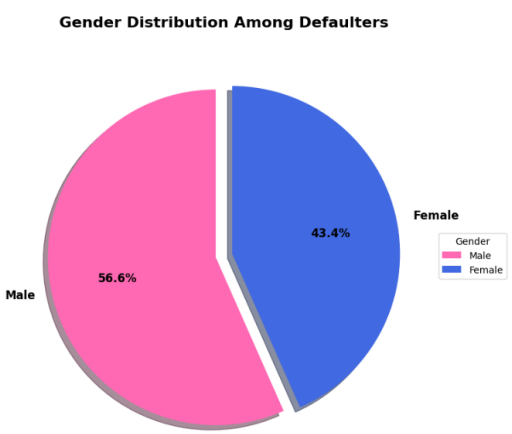
GENDER-AGE-MARRIAGE ANALYSIS:

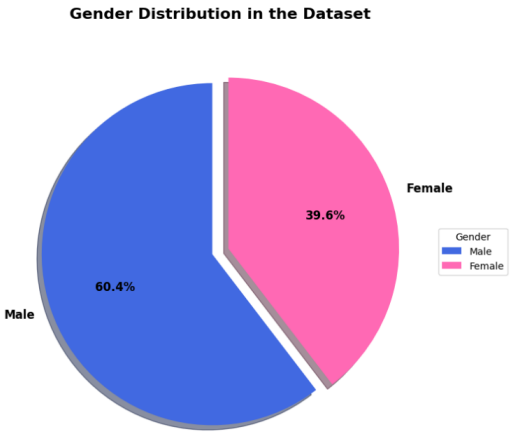


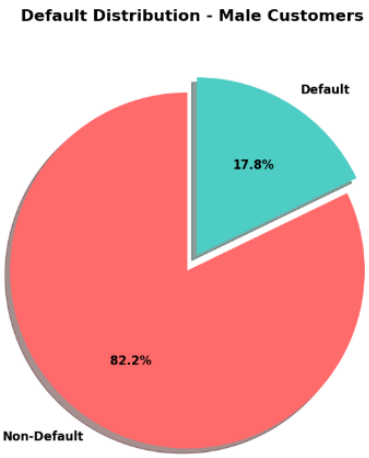


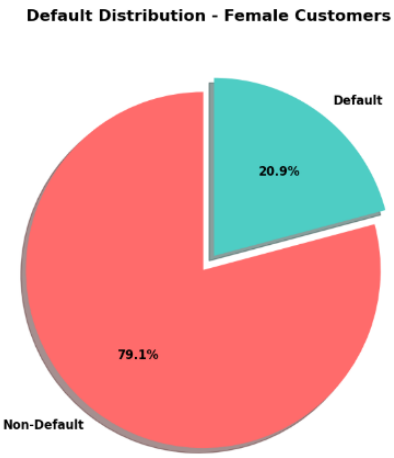
AGE-

1. First of all, there were some missing values in the age columns (126 values), which was taken care of in pre-processing.
2. Different age groups are made such that it contains almost the same number of customers.
3. Age groups 21-27 and 43-79 (higher credit limits) are the most susceptible to default.



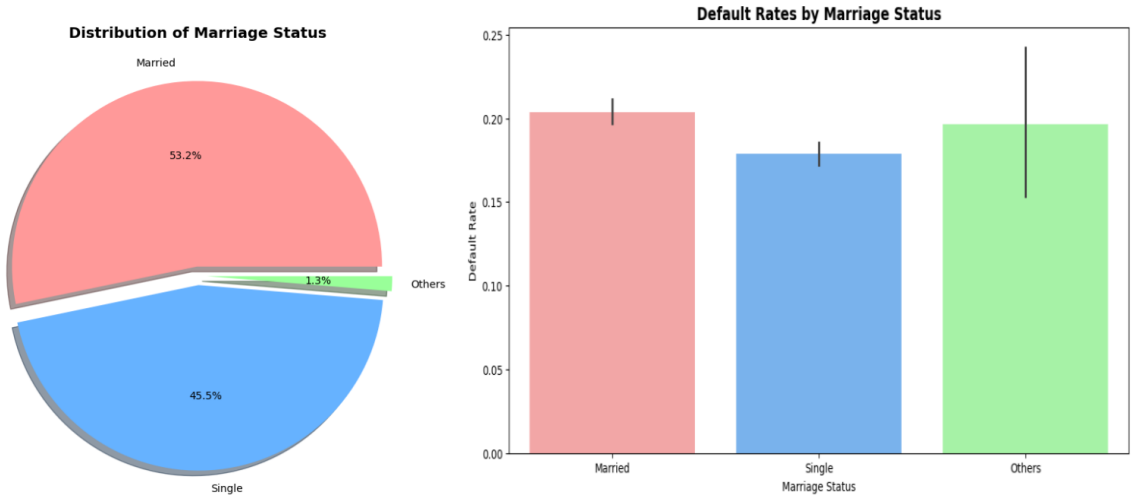


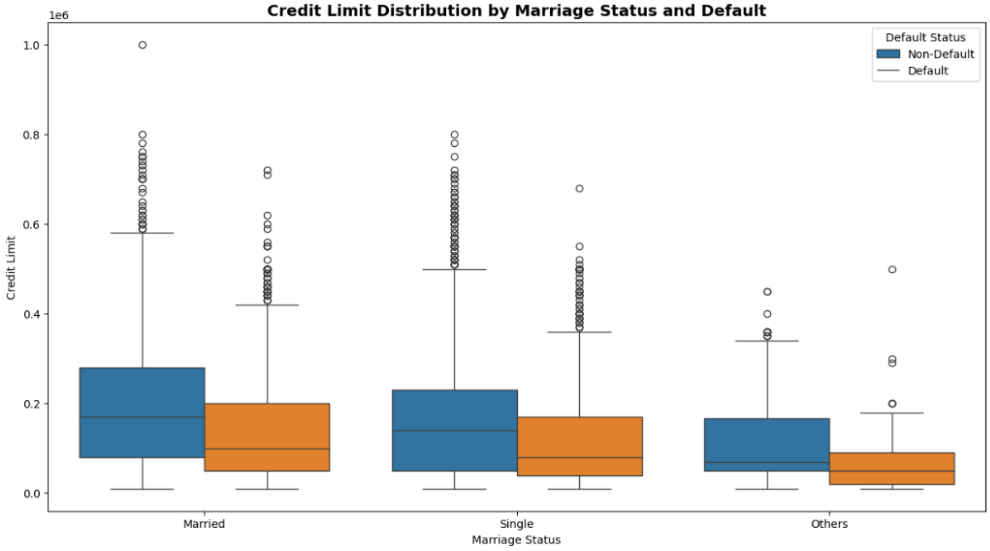




GENDER-

1. Females have higher default rates under gender segment.
2. Males have higher absolute number of defaulters are there are more males in the population.
3. Females seem to be more consistent with the credit utilization.





MARRIAGE-

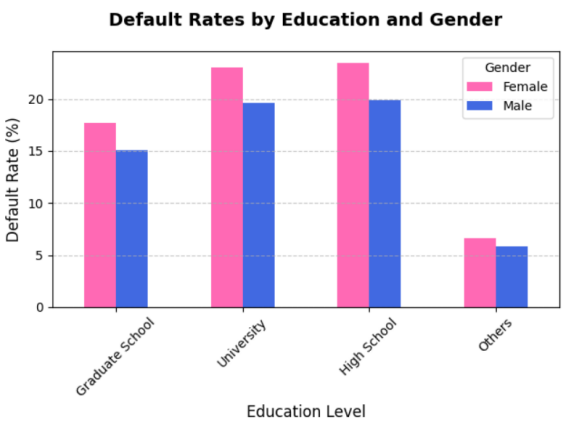
1. Married individuals form highest segment.
2. Higher credit limits reflect established financial stability.
3. Higher default rates in married despite financial stability is potentially due to larger financial responsibilities and family obligations.

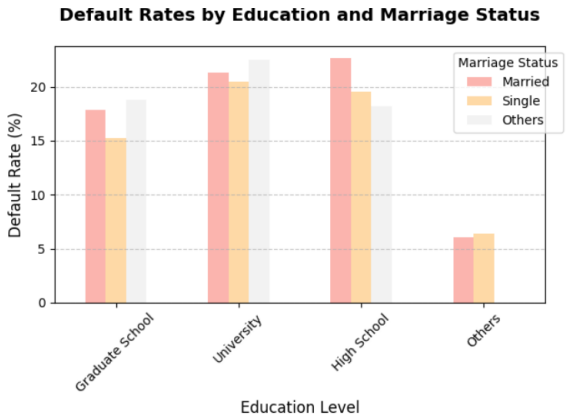
COMBINED INSIGHTS-

1. Females consistently show higher default rate across all age groups and marital status.
2. Married females have higher credit limit probably due to dual income and combined household creditworthiness.
3. Single females have lower credit than males possibly due to income gap.

4. Married younger segment (21-27 years) is the most risky.

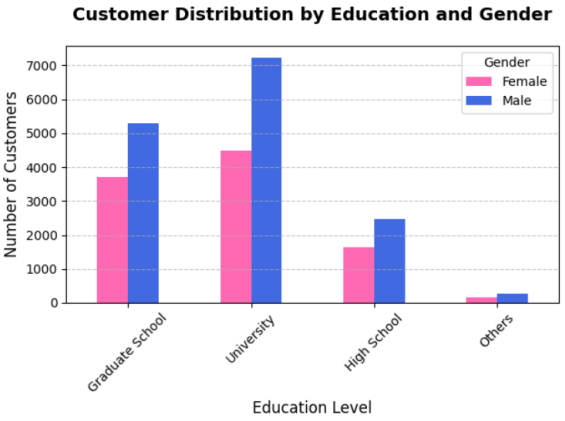
EDUCATION ANALYSIS:

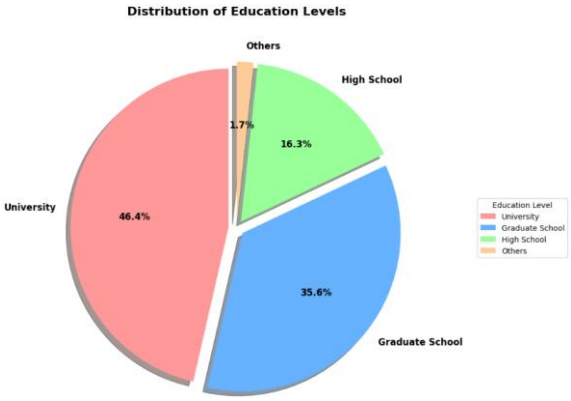




Females tend to have **higher default rates** across all education groups.

Married individuals tend to have higher default rates (disregarding the 'others' because the population is very small to draw any conclusion).

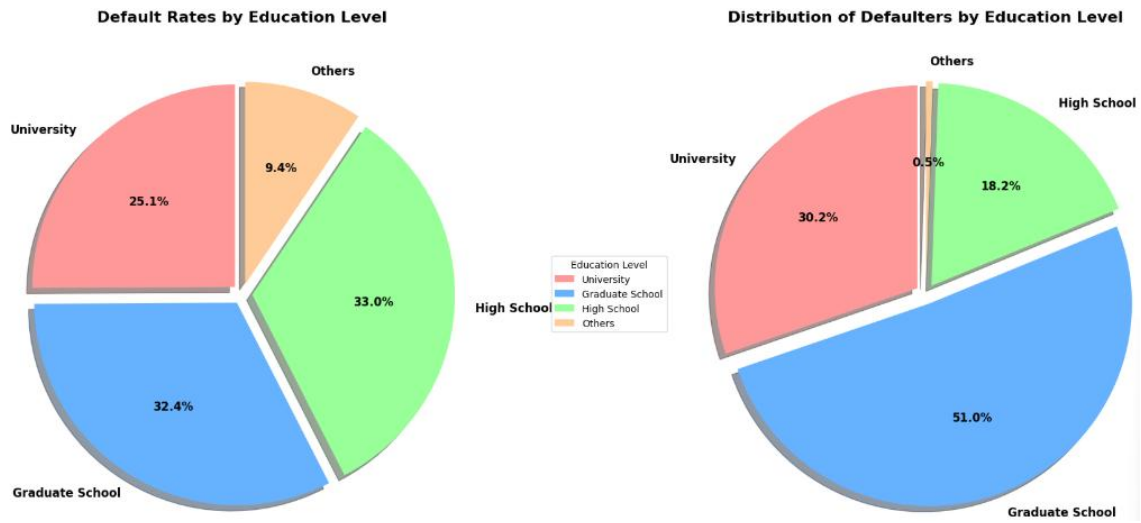




University Males make up the highest chunk of population and in general University group has the highest education, but this is also the class which shows highest disparity in males and females.

Graduate school individuals make up the highest portion of defaulters (around 51%, which is **more than half!!!**)

Graduate school and High school groups show the highest default rates among all groups.



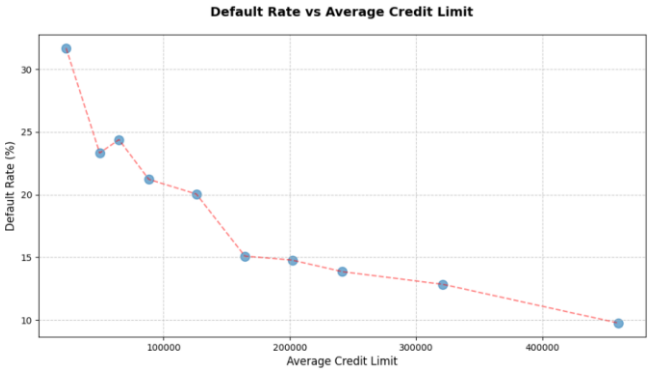
Conclusions:-

1. Female with High School education, married, and aged 21–27 exhibit the highest default rate.
2. Among males, those with **High School education, aged 37–43, and single** show the highest default rate.
3. In numbers University educated, males and females, aged 21-27 and single make up for the highest number of defaulters.

Identifying these groups was very important in management based decision making but they're not a very good predictor for default status, which is why we'll move towards financial analysis.

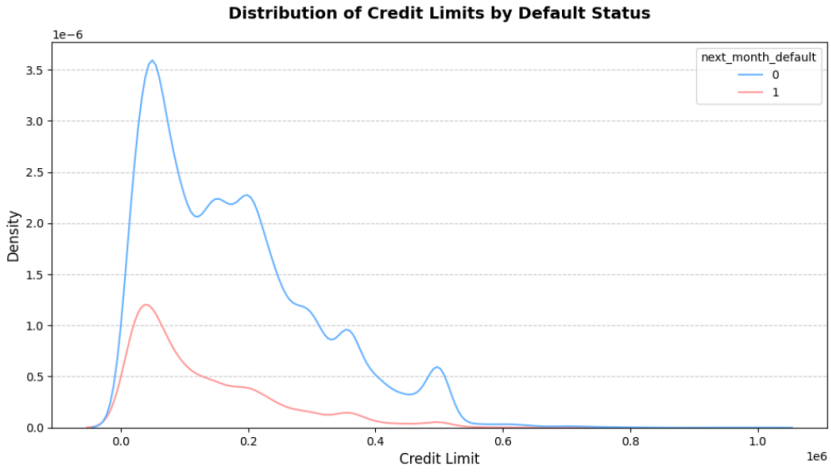
FINANCIAL ANALYSIS:

1) LIMIT_BAL (Credit Limit)-



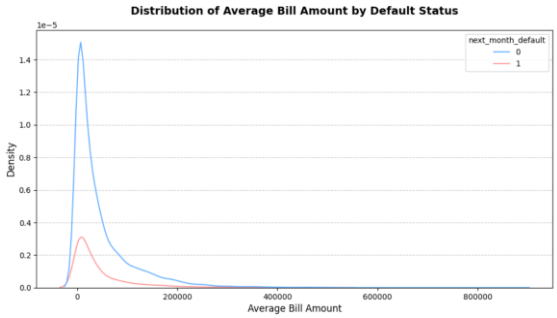
- Individuals with lower seem to show very high default rates, which is expected taking into account their financial situation.

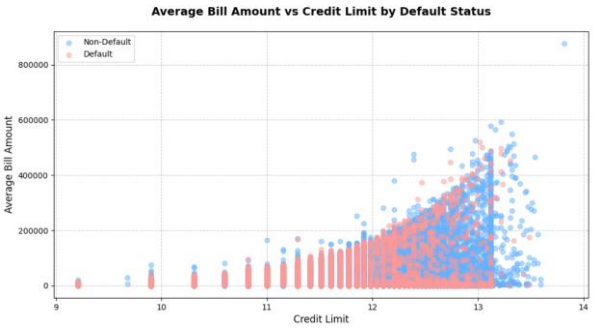
- Segment below 180,000 of credit limit contains most of the defaulters. With high concentration of defaulters with credit limit < 100,000. credit limit > 270,000 can be considered a more relaxed area.



- Though credit limit density show a similar pattern for both, which is why credit limit (LIMIT_BAL) alone might not be a very influential factor.

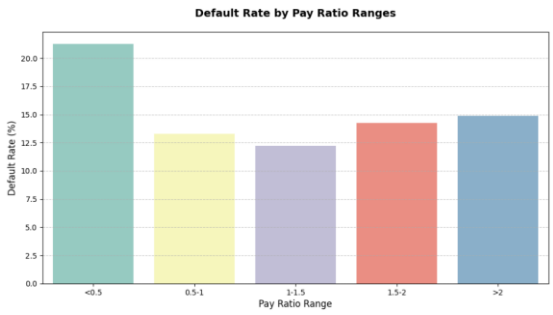
2) Average Bill Amount:

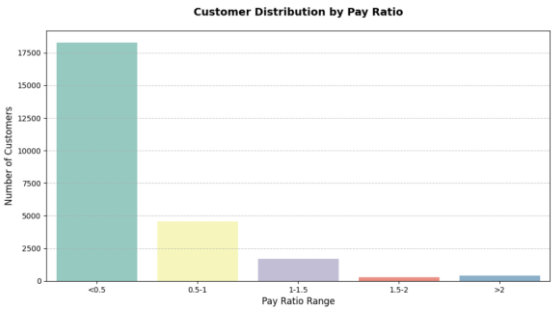




- Average bill amount doesn't seem to tell much as both defaulters and non defaulters have similar density graphs, only thing it tell is that non-defaulters have higher bill amount (which is kind of obvious).

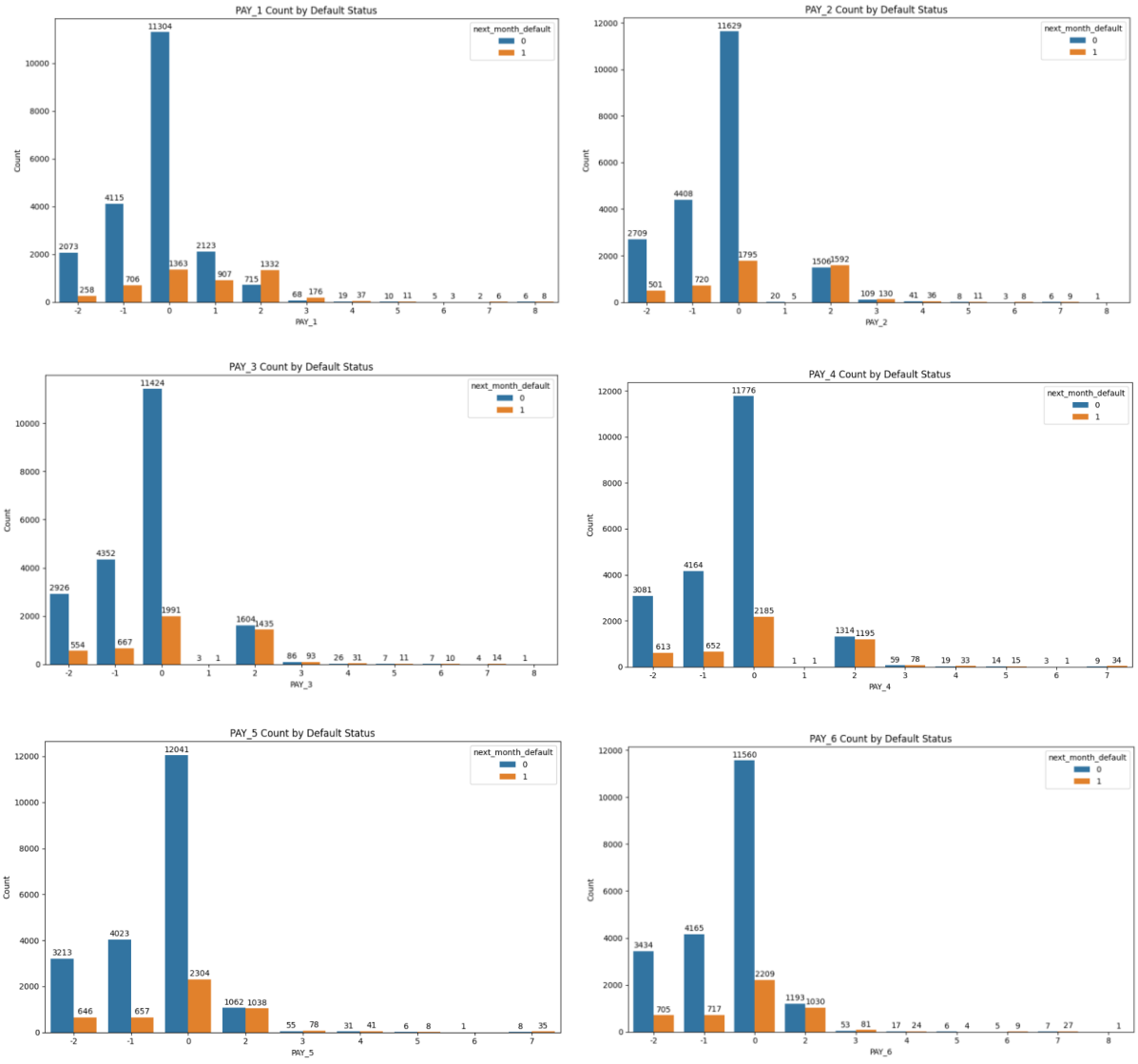
3) PAY_TO_BILL_RATIO:





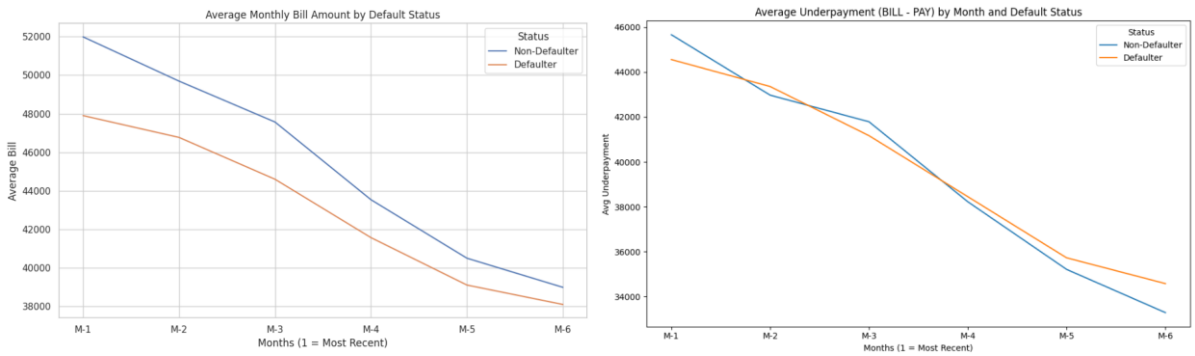
- Higher PAY_TO_BILL_RATIO show lower risk of default. Ratio of >1.5 can be considered a pretty good spot. (ratio of >1.5 show higher rates because they contain very small population).

4) pay_m:



- Recent Month Dynamics (pay_1)-
  Clear visibility of one-month delays, Active tracking of short-term delinquency, Immediate payment behaviour indicator.
- Historical Pattern (pay_2 to pay_6)-
  Direct Progression to longer delays, Minimal single-month delinquencies, Distinct payment behaviour evolution.

5) pay_amt and bill_amt analysis:



- bill_amt is not a good predictor as it doesn't have much abnormalities and just says that bill_amt for non-defaulters is higher on average. So instead we'll use average bill amt.
- Now for second graph we can see difference of bill-amt and pay-amt, Recent payment behaviour (m= 1 and 2) : non-defaulters show higher underpayment rates indicates healthy revolving credit usage, demonstrates strategic minimum payments and normal credit management.
  Historical patterns (m= 3-6) :Defaulters exhibit higher underpayment frequency, clear early warning signals present, consistent pattern of payment stress and strong predictor of future defaults.
- EDA ends here and on the basis of this analysis we will design features in a later stage.

DATA PRE-PROCESSING AND FEATURE ENGINEERING:

Data Pre-processing-

- Taking care of missing age values by using k-nearest neighbours to substitute it with average of the k=8 nearest neighbours.
- Dropping redundant data line customer ID and NaN values, fortunately there were no NaN values.
- In the plot of LIMIT_BAL we saw that it has a rightward skewed curve, so it should be replaced with a logarithmic curve.

- Prepare X and y, and do test train split with test set as 20%, and stratify the data such that it contains the equal percent of y in both test and train sets.

```
#as we saw from above EDA, limit bal shows right skewed distribution, so we will transform it to log(1+limit_bal)

df['LIMIT_BAL'] = np.log(1 + df['LIMIT_BAL'])
✓ 0.0s
```

- We'll be using SMOTE to address the class imbalance and then in model training we'll use class weights.

Feature Engineering:

```
#feature Engineering
# 1. Credit Usage Metrics
df['CREDIT_UTILIZATION'] = df['Bill_amt1'] / df['LIMIT_BAL']
df['AVG_UTILIZATION'] = df[['Bill_amt1', 'Bill_amt2', 'Bill_amt3',
                            'Bill_amt4', 'Bill_amt5', 'Bill_amt6']].mean(axis=1) / df['LIMIT_BAL']
df['UTILIZATION_CHANGE'] = (df['Bill_amt1'] - df['Bill_amt2']) / df['LIMIT_BAL']

# 2. Payment Behavior
# Payment Streak
df['PAYMENT_STREAK'] = 0
for i in range(1, 7):
    df.loc[df[f'pay_{i}'] <= 0, 'PAYMENT_STREAK'] += 1

# Delinquency Streak
df['DELINQUENCY_STREAK'] = 0
for i in range(1, 7):
    df.loc[df[f'pay_{i}'] > 0, 'DELINQUENCY_STREAK'] += 1

# Payment Volatility
payment_cols = ['pay_amt1', 'pay_amt2', 'pay_amt3', 'pay_amt4', 'pay_amt5', 'pay_amt6']
df['PAYMENT_VOLATILITY'] = df[payment_cols].std(axis=1)
```

- Credit Utilization- shows how much of available credit is being used. high utilization (>80%) = higher default risk.
- Average Utilization- **6-month average** shows consistent spending pattern. More stable predictor than single-month utilization.
- Utilization Change- **Increasing utilization** = Potential financial distress. **Decreasing utilization** = Improving financial health. Trend is often more predictive than absolute values.
- Payment Streaks- Payment history is strongest predictor of future behaviour. **Behavioural finance**: Past behaviour predicts future behaviour. Rewards customers with good payment discipline.

- Delinquency Streak- **Multiple late payments** = Systematic payment issues. Not just forgetfulness, but financial inability. **Regulatory requirement**: Banks must track delinquency patterns.
- Payment Volatility- **Stable payments** = Predictable income/budgeting. **Volatile payments** = Irregular income or poor financial management. Helps distinguish between different types of risk.

```
# 3. Delinquency Patterns
df['WORST_DELINQUENCY'] = df[['pay_1', 'pay_2', 'pay_3',
                              'pay_4', 'pay_5', 'pay_6']].max(axis=1)
df['RECENT_DELINQUENCY'] = df[['pay_1', 'pay_2', 'pay_3']].mean(axis=1)
df['DELINQUENCY_FREQUENCY'] = (df[['pay_1', 'pay_2', 'pay_3',
                                   'pay_4', 'pay_5', 'pay_6']] > 0).sum(axis=1)

# 4. Spending Patterns
bill_cols = ['Bill_amt1', 'Bill_amt2', 'Bill_amt3', 'Bill_amt4', 'Bill_amt5', 'Bill_amt6']
df['SPENDING_VOLATILITY'] = df[bill_cols].std(axis=1)
df['PEAK_UTILIZATION'] = df[bill_cols].max(axis=1) / df['LIMIT_BAL']

# 5. Advanced Features
# Risk Score (simplified version)
df['RISK_SCORE'] = (df['CREDIT_UTILIZATION'] * 0.3 +
                    df['DELINQUENCY_FREQUENCY'] * 0.4 +
                    df['PAYMENT_VOLATILITY'] * 0.3)

# Behavioral Score
df['BEHAVIORAL_SCORE'] = (df['PAYMENT_STREAK'] * 0.4 +
                          (6 - df['DELINQUENCY_STREAK']) * 0.4 +
                          (1 - df['CREDIT_UTILIZATION']) * 0.2)
```

- Worst Delinquency- Worst-case scenario often predicts future behaviour. Difference between "1 day late" vs "90+ days late". Risk segmentation: Severity matters more than frequency sometimes.
- Recent Delinquency- **Recency bias**: Recent events predict immediate future. People can change behaviour over time. **Current financial situation** more relevant than historical.
- Delinquency Frequency- **One-time late payment** = Accident/oversight. **Multiple late payments** = Systematic problem. **Pattern recognition**: Frequency reveals true risk level.
- Spending Volatility- **Stable spending** = Good budgeting/predictable income. **Volatile spending** = Irregular income or poor financial control. **Lifestyle indicator**: Consistent vs chaotic financial behaviour.

- Peak Utilization- **Peak stress test**: How high does utilization go? Even if average is low, peaks show vulnerability. **Seasonal patterns**: Holiday spending, emergencies.
- Risk Score- **Domain expertise**: Combines most predictive factors. **Weighted importance**: Delinquency gets highest weight (40%). **Single metric**: Easy to interpret and use.
- Behavioural Score- **Positive reinforcement**: Rewards good behaviour. **Balanced view**: Not just penalty based. **Customer retention**: Identifies good customers to reward.

MODEL TRAINING AND RESULTS:

	Model	Accuracy	Precision	Recall	F1 Score	ROC AUC	F2 Score
0	Logistic Regression	0.760990	0.414753	0.619543	0.496874	0.767669	0.563860
1	Random Forest	0.817228	0.523694	0.448025	0.482913	0.769005	0.461357
2	XGBoost	0.829109	0.574436	0.397089	0.469576	0.754008	0.423222
3	LightGBM	0.835842	0.597938	0.422037	0.494820	0.775005	0.448421
4	AdaBoost	0.781782	0.445568	0.595634	0.509786	0.774244	0.558044

- These are the 5 models that I used at first, Logistic regression and AdaBoost seem to give the best results.
- In this scenario, f2 score is of paramount importance. f2 score gives more weightage to recall (weightage depends on the value of beta).
- f2 score is more important because false negatives are more financially risky compared to false positives. Not being able to detect false negatives essentially means losing all the money that was given in the form of credit.
- Then I used a custom score to hyper tune model, I gave more weightage to f2 score, included f1 score to maintain balance recall and precision.

```python
def custom_score(y_true, y_pred):
    f2 = fbeta_score(y_true, y_pred, beta=2)
    acc = accuracy_score(y_true, y_pred)
    f1 = f1_score(y_true, y_pred)

    final_score = 0.4 * f2 + 0.3 * acc + 0.3 * f1
    return final_score
```

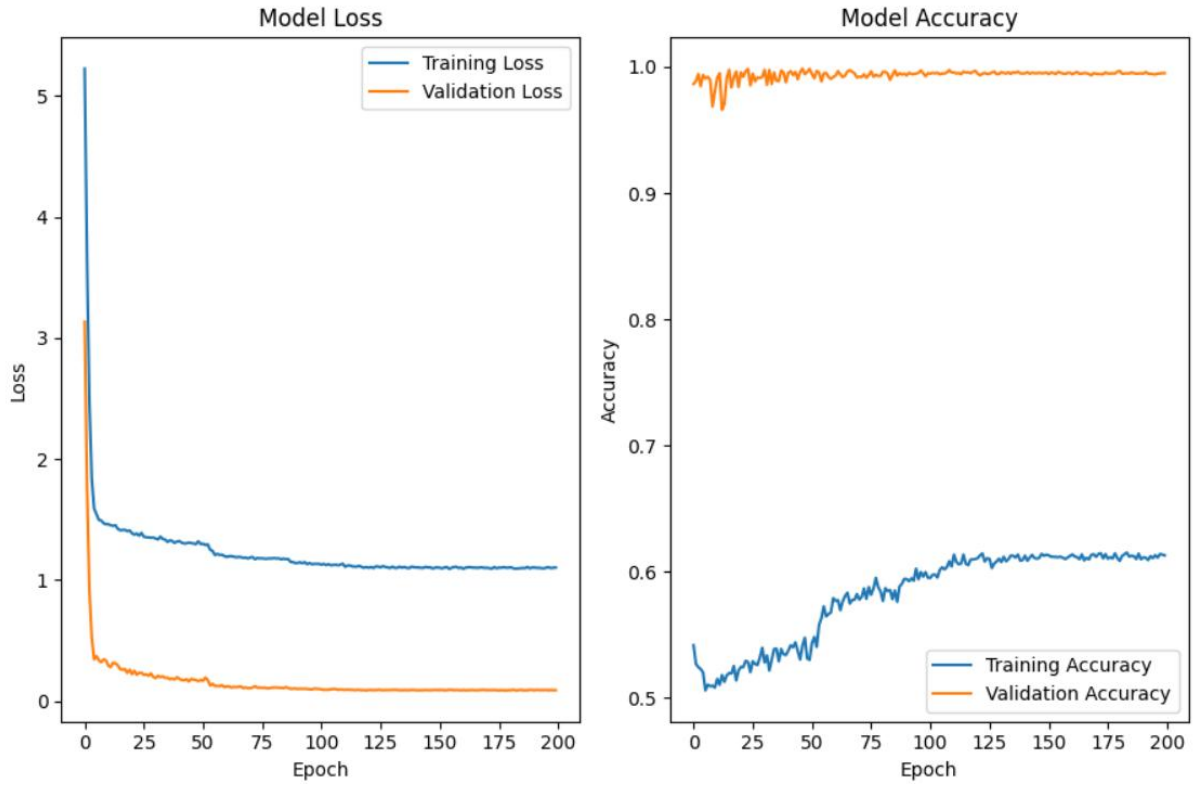
- Also included accuracy so that model doesn't flag too many false positives.

```
Final Results:
                 Model  Custom Score  Accuracy  Precision    Recall  F1 Score
2              XGBoost      0.611655  0.798020   0.474650  0.564449  0.515670
1        Random Forest      0.608271  0.792673   0.463891  0.567568  0.510519
0  Logistic Regression      0.603486  0.761188   0.415160  0.620582  0.497500

   F2 Score   ROC AUC
2  0.543870  0.782002
1  0.543284  0.779945
0  0.564699  0.767596
```

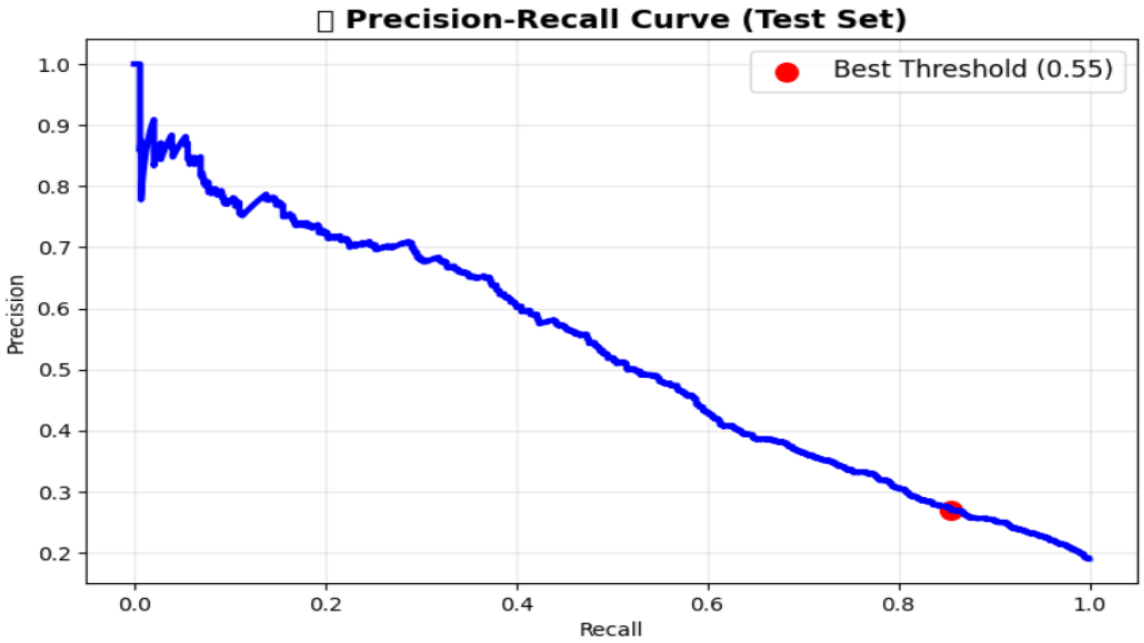
- Then I decided to make XGBoost, Random Forest and Logistic regression a bit better by increasing n_estimators and iterations. XGBoost gave the best results in this case.

```python
# Weighted Score Optimized Model
def build_weighted_score_model():
    model = Sequential([
        # Input Layer - optimized for F2/F1 balance
        Dense(1024, activation='relu', input_shape=(X_train.shape[1],),
              kernel_regularizer=tf.keras.regularizers.l2(0.003)),
        BatchNormalization(),
        Dropout(0.45),

        # Hidden Layers - balanced for recall and precision
        Dense(512, activation='relu', kernel_regularizer=tf.keras.regularizers.l2(0.003)),
        BatchNormalization(),
        Dropout(0.35),

        Dense(256, activation='relu', kernel_regularizer=tf.keras.regularizers.l2(0.003)),
        BatchNormalization(),
        Dropout(0.3),

        Dense(128, activation='relu', kernel_regularizer=tf.keras.regularizers.l2(0.003)),
        BatchNormalization(),
        Dropout(0.25),

        Dense(64, activation='relu', kernel_regularizer=tf.keras.regularizers.l2(0.003)),
        BatchNormalization(),
        Dropout(0.15),

        # Output Layer
        Dense(1, activation='sigmoid')
    ])
```

- Then I also used a Neural Network with a very basic structure with Dropout Regularisation and Batch norm.
- It has a basic NN structure of 512->256->128->64 nodes in the hidden layers with ReLU activation function then output layer has one node which gives the probability using sigmoid activation function.
- Also changed the custom score to 0.7*f2 + 0.2*f2 + 0.1*acc.
- Ran this model for 200 epochs, and binary cross entropy as the loss function.



- model performs much better on validation set, its most likely because of regularization. It has a healthy composition.
- The only issue is that it has low precision for good values of recall, and this kind of relationship is expected between recall and precision.
- So, the next most important step is to select the best threshold for classifying, and obviously threshold = 0.55 works really bad as it flags most of the dataset as positive, which is very conservative and not very practical.



- So, I conducted an analysis to find best threshold, for which I conducted testing and got 0.87 as the best threshold for the balance of conservative and safety.

```
🎯 FINDING OPTIMAL THRESHOLD USING TRAINING DATA:
================================================
632/632 ━━━━━━━━━━━━━━━━━━━━ 1s 1ms/step
Training probability range: 0.0003 to 0.9908

🎯 OPTIMAL THRESHOLDS:
Best F1 Score Threshold: 0.860 (F1: 0.4560)
Best F2 Score Threshold: 0.790 (F2: 0.5915)
Precision-Recall Optimal Threshold: 0.862
ROC Optimal Threshold (Youden): 0.861
Custom Weighted Score Optimal Threshold: 0.840 (Score: 0.5392)
```

- For best f2 score of 0.5915, I got the threshold as 0.79.

CONCLUSION:

Instead of pursuing metric optimization in a vacuum (only f2 score), I implemented a comprehensive evaluation framework that considers the broader business context and stakeholder needs. This balanced approach produced a model that not only demonstrates strong technical capabilities but

also addresses the practical challenges and strategic priorities of credit risk management.

All the code files are attached in the folder and I’ve tried to make it as organised as possible, and even the predicted submission file is also there.