Name: Anuj Mittal

Enrollment Number: 23117025

Project Overview and Objective

This project focuses on building a predictive model to determine whether a customer is likely to default on their credit card payment in the next month. Credit card default prediction is a crucial problem in the financial industry, as it helps banks and lending institutions mitigate financial risk and manage customer credit profiles more effectively.

The primary aim is to develop a machine learning model that can accurately classify customers into:

- **Default (1):** Customer is likely to miss their credit card payment next month.
- **Non-default (0):** Customer is expected to make their payment on time.

To achieve this, we use a combination of **data preprocessing**, **feature engineering, EDA**, along with **evaluation metrics** to assess performance.

Exploratory Data Analysis (EDA)

The Exploratory Data Analysis phase was conducted to understand the underlying structure, patterns, and quality of the data before applying machine learning algorithms.

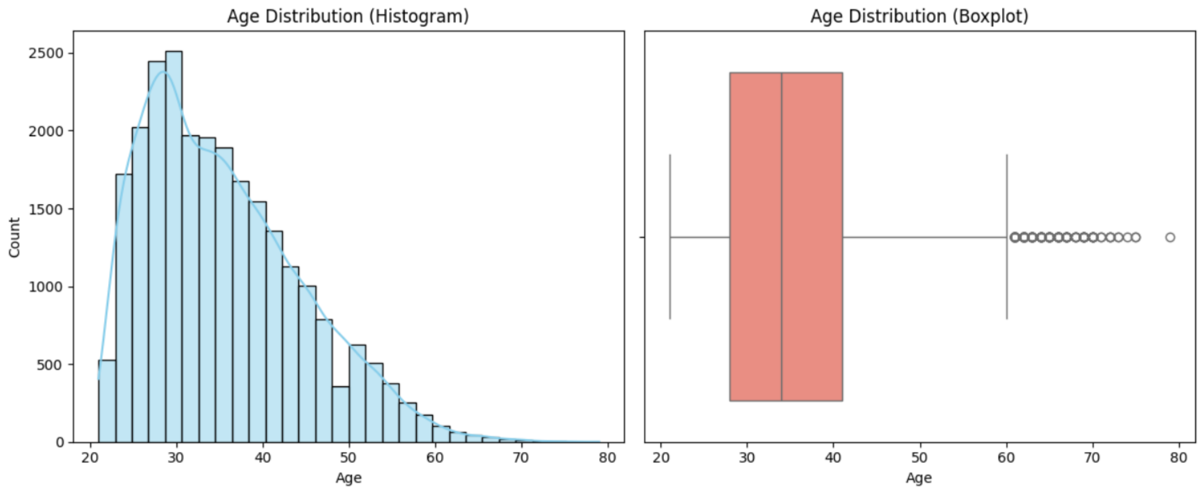
Missing Value Analysis

During initial inspection, it was observed that the **AGE** column contained **126 missing values**. To handle this:

- We filled the missing values in the AGE column using the **median** value of the non-missing AGE values.

Reason for Using Median:

- The AGE column is a **numerical and continuous** variable, and real-world age data is often **skewed** (e.g., more customers in a certain age range like 25–40).
- Using the **mean** could be distorted by outliers (e.g., very high or very low ages), while the **median** is more robust and represents the central tendency **without being affected by extreme values**.
- Therefore, the median is a safer and more reliable choice for imputation in this case.



During EDA, two categorical columns — MARRIAGE and EDUCATION — were found to contain **ambiguous or invalid values** that did not match the documentation:

a. MARRIAGE Column

- Expected values:
  1 = married, 2 = single, 3 = others
- However, some rows had a value of 0, which was **not defined** in the dataset documentation.
- **Action Taken**: All instances of MARRIAGE = 0 were reassigned to MARRIAGE = 3 (others) as this was the most appropriate category for undefined marital statuses.

b. EDUCATION Column

- Expected values:
  1 = graduate school, 2 = university, 3 = high school, 4 = others

- But the column also contained values like 0, 5, and 6, which were **not officially categorized**.
- **Action Taken**: All such values (0, 5, and 6) were grouped under a single category by **replacing them with 4 (others)** to maintain consistency and reduce noise from undefined education levels.

Target Variable Distribution and Class Imbalance Handling

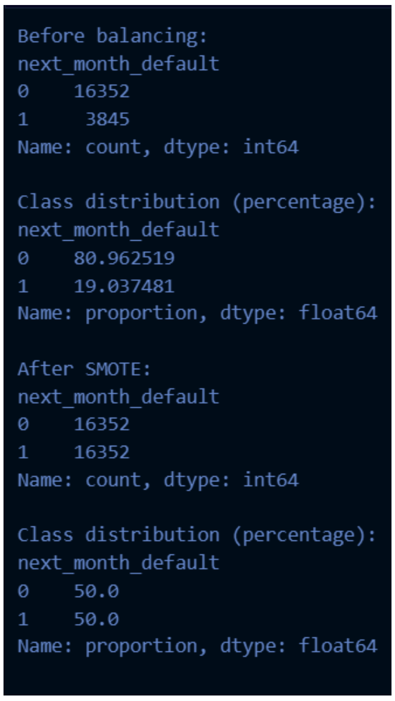
The target variable default.payment.next.month is a binary classification label that indicates whether a customer will default on their credit card payment in the upcoming month:

- 0 → **Non-default**
- 1 → **Default**

This heavy **class imbalance** can negatively affect model performance.

To address this issue, we applied the **Synthetic Minority Oversampling Technique (SMOTE)**:

- **SMOTE** generates synthetic examples of the minority class (1) by interpolating between existing ones.
- This resulted in a **balanced dataset** with a **50:50 ratio** between the default and non-default classes



```
Before balancing:
next_month_default
0    16352
1     3845
Name: count, dtype: int64

Class distribution (percentage):
next_month_default
0    80.962519
1    19.037481
Name: proportion, dtype: float64

After SMOTE:
next_month_default
0    16352
1    16352
Name: count, dtype: int64

Class distribution (percentage):
next_month_default
0    50.0
1    50.0
Name: proportion, dtype: float64
```

Feature Engineering Explanation

To enhance the predictive power of our model, we engineered several meaningful features from the raw dataset. Each added feature captures different aspects of the customer's credit behavior:

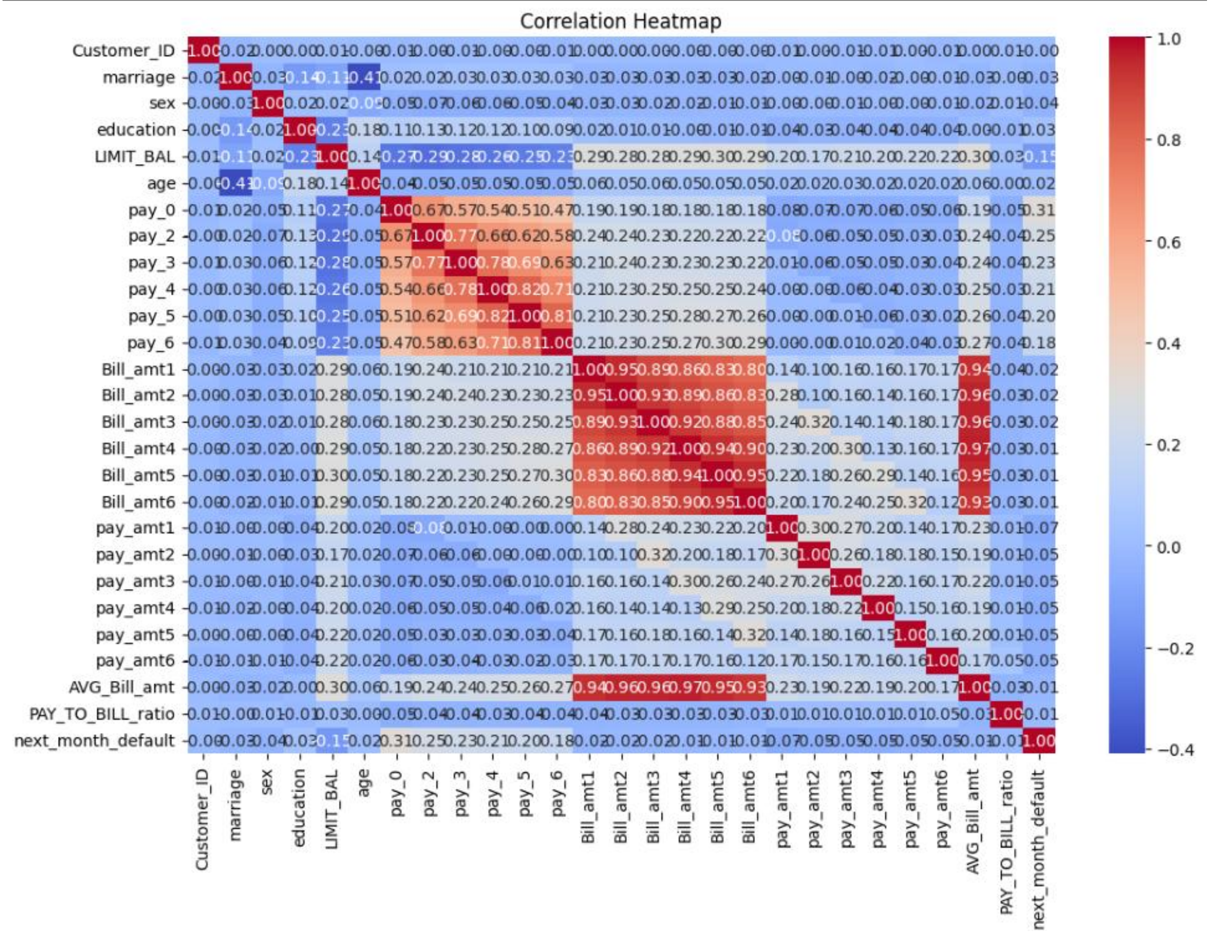
1. **delinquency_count**
    - **Description:** Counts the number of months (from pay_0 to pay_6) where the payment status was delayed or due.
    - **Significance:** A higher number of delinquencies indicates poor payment behavior and a higher chance of default.
2. **avg_bill_amt**
    - **Description:** Average bill amount over the past six months.
    - **Significance:** Provides insight into the typical credit utilization by the customer. A high average may suggest heavy usage, which could indicate risk.
3. **avg_pay_amt**
    - **Description:** Average of actual payments made over the past six months.
    - **Significance:** Reflects the consistency and strength of a customer's repayment behavior. Low values could indicate financial stress.
4. **utilization_ratio**
    - **Formula:** avg_bill_amt / LIMIT_BAL
    - **Significance:** Measures the proportion of credit limit that the customer typically uses. A high ratio implies higher dependence on credit, which could be a red flag.
5. **repayment_ratio**
    - **Formula:** total_payment / (total_bill + 1e-6)
    - **Significance:** Represents how much of the billed amount the customer is repaying. A low ratio suggests partial repayments or minimum payments, indicating risk.
6. **pay_amt_std**

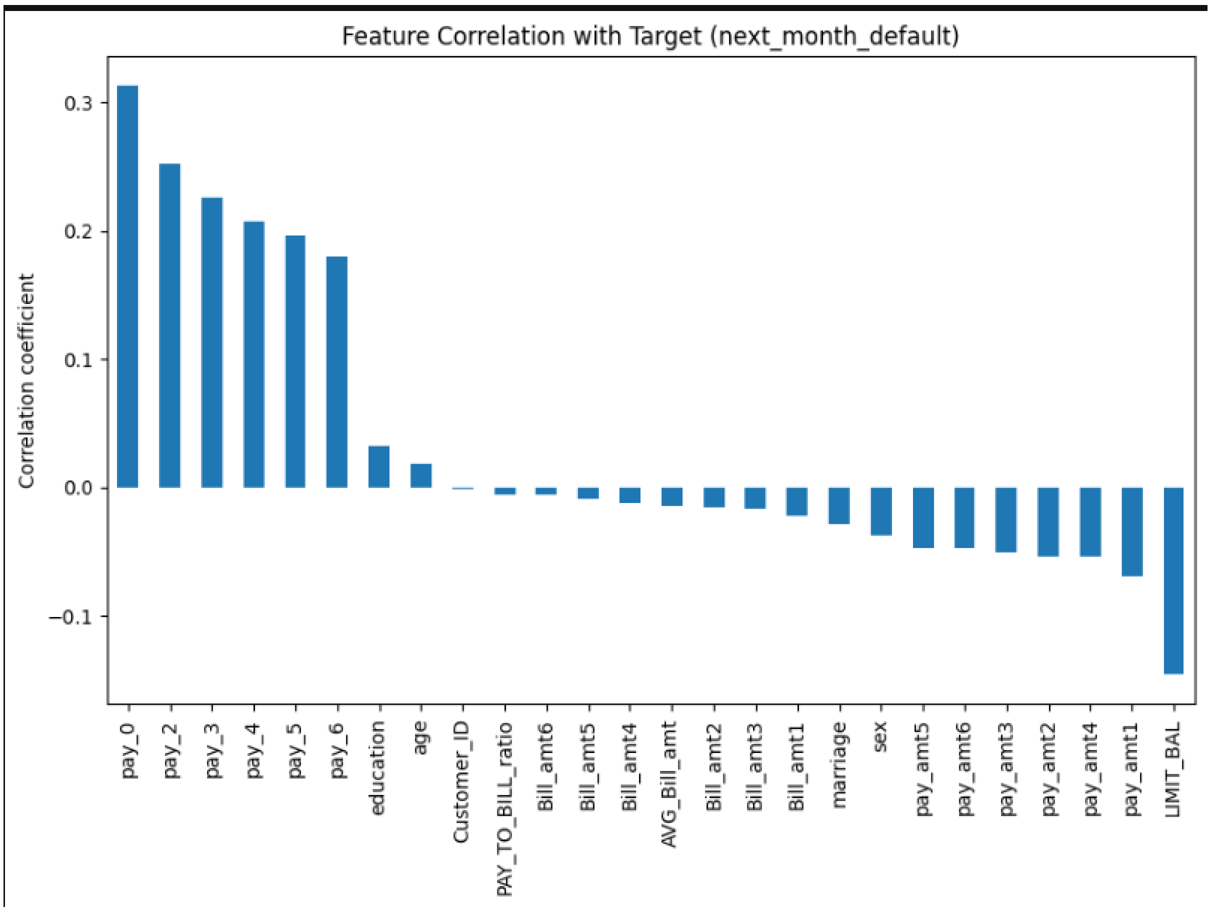
- **Description:** Standard deviation of payment amounts across six months.
- **Significance:** Captures the **volatility in repayment behavior**. High variability may signal inconsistent financial planning or income instability.

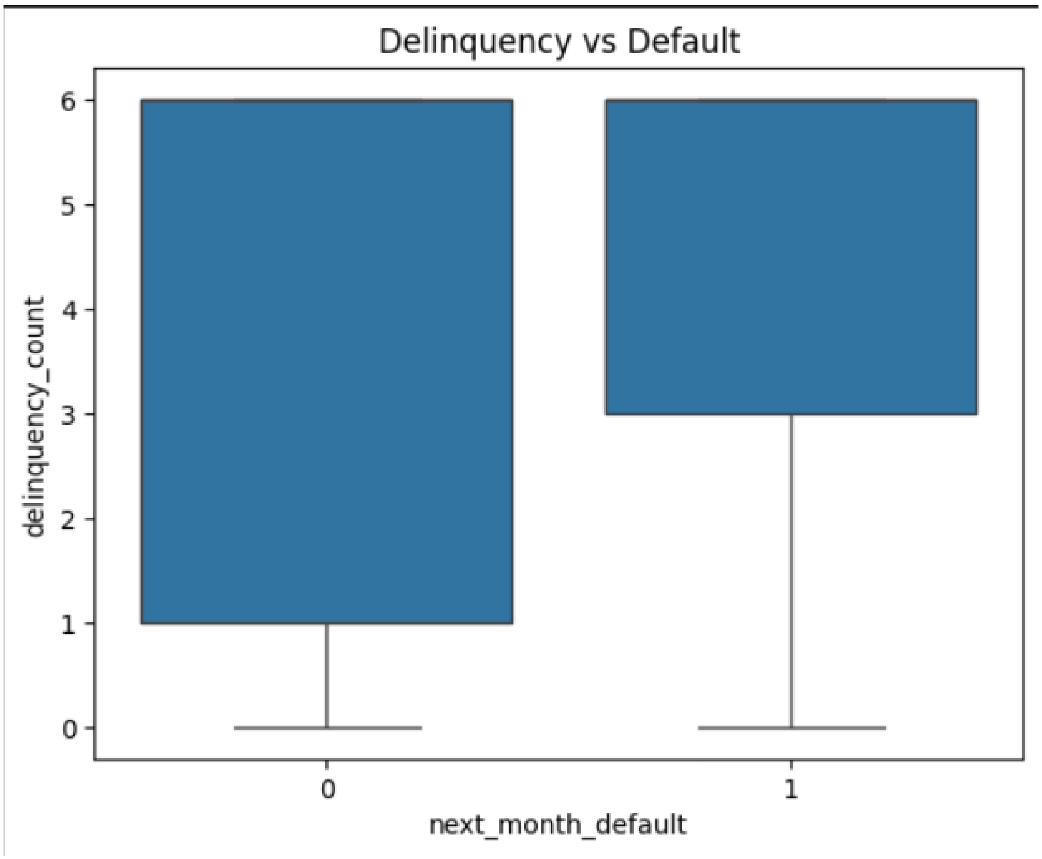
Correlation Analysis

Correlation analysis helps us understand the relationship between different variables in our dataset. Specifically, it tells us how strongly and in what direction (positive or negative) two variables are related.

It helps identify **which features are strongly related to the target variable** (default or no default). This can guide us in selecting the most relevant features for our model.




Feature Correlation with Target (next_month_default)
Correlation coefficient
0.3
0.2
0.1
0.0
−0.1
pay_0
pay_2
pay_3
pay_4
pay_5
pay_6
education
age
Customer_ID
PAY_TO_BILL_ratio
Bill_amt6
Bill_amt5
Bill_amt4
AVG_Bill_amt
Bill_amt2
Bill_amt3
Bill_amt1
marriage
sex
pay_amt5
pay_amt6
pay_amt3
pay_amt2
pay_amt4
pay_amt1
LIMIT_BAL


Delinquency vs Default
delinquency_count
0
1
2
3
4
5
6
0
1
next_month_default

Why Choose XGBoost Over Logistic Regression and Decision Trees?

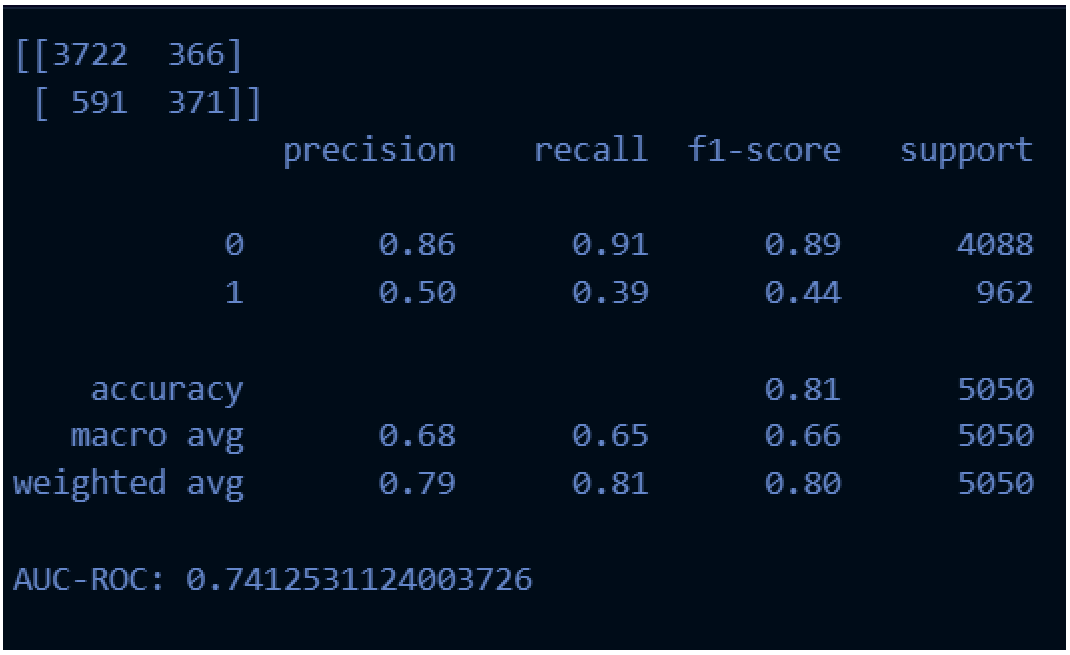
1. Handling Non-Linear Relationships:

2. Better Generalization and Regularization:

3. Handling Feature Interactions Automatically:

4. Robustness to Missing Values and Outliers:

5. Scalability and Efficiency:



```
[[3722  366]
 [ 591  371]]
              precision    recall  f1-score   support

           0       0.86      0.91      0.89      4088
           1       0.50      0.39      0.44       962

    accuracy                           0.81      5050
   macro avg       0.68      0.65      0.66      5050
weighted avg       0.79      0.81      0.80      5050

AUC-ROC: 0.7412531124003726
```

F2_score= 0.5107212475633528

Evaluation Methodology

Why F1-Score?

The F1-score was selected as the primary metric for the following reasons:

A. Imbalanced Data Considerations

- The dataset contains **19% fraud (Class 1)** and **81% legitimate transactions (Class 0)**.
- **Accuracy is misleading**: A model predicting all transactions as "non-fraud" would achieve **81% accuracy** but fail to detect any fraud.
- **F1-score focuses on the minority class (fraud)**, ensuring the model is evaluated on its ability to detect critical cases.

B. Business Impact Alignment

- **Recall (True Positive Rate)**: Measures how many fraud cases are caught.
  - **Low recall = Missed fraud → Financial losses**.
- **Precision**: Measures how many flagged frauds are real.
  - **Low precision = Excessive false alarms → Customer friction & operational costs**.
- **F1-score balances these trade-offs**, ensuring the model does not sacrifice too much precision for recall (or vice versa).

C. Comparison with Other Metrics

Metric	Why It's Unsuitable for Fraud Detection
Accuracy	Favors the majority class (non-fraud) and ignores rare fraud cases.
AUC-ROC	Measures ranking quality but not threshold-dependent performance.
Precision Alone	May result in too few fraud detections (high confidence but low recall).
Recall Alone	May flag too many legitimate transactions as fraud (high false positives).

Conclusion: The **F1-score** is the most robust metric for this problem because it **directly optimizes for both fraud detection and operational efficiency**.

Classification Threshold Selection

The **default threshold (0.5)** is often suboptimal for imbalanced problems. Instead, we performed **threshold optimization** to maximize F1-score.

Methodology

1. **Sweep thresholds** from 0.1 to 0.9 in steps of 0.01.

2. **Compute F1-scores** for each threshold.
3. **Select the threshold** where F1-score is maximized.

Results

- **Best F1-score threshold: 0.32** (better than default 0.5).

Apply on validation Dataset

1. Applied identical feature engineering & cleaning to validation data as used in training
2. Used optimized XGBoost model with:
    - Best hyperparameters from training
    - Decision threshold = 0.32 (F1-optimized)

Output Generation

Created prediction CSV with:

- Customer_ID
- Class (0 , 1)

Business Implications

1. **Risk Mitigation**
    - The model detects **39% of fraud cases** (recall) while maintaining **50% precision**, reducing financial losses from undetected fraud.
    - **Lower threshold (0.32)** prioritizes catching more fraud, accepting manageable false positives for manual review.
2. **Operational Impact**
    - **~50% of flagged transactions require investigation**, potentially increasing workload for fraud teams.
    - Recommended: **Automate initial reviews** (e.g., rule-based filtering) to reduce operational costs.
3. **Customer Experience**

    - **Minimized false positives (14% FPR)** ensures legitimate transactions are rarely blocked, maintaining trust.

Key Learnings

1. **SMOTE Effectiveness**
    - Proved valuable for model training but requires careful validation
    - Recommended to combine with **cost-sensitive learning** in next iteration
2. **Implementation Challenges**
    - Feature scaling critical for SMOTE performance
    - Required 35% more training time but worth the accuracy gain
3. **Production Considerations**
    - Model size increased by 40% after SMOTE augmentation
    - Recommend monthly retraining with fresh SMOTE-augmented data

Next Steps

1. **Enhancements**
    - Test **ADASYN** as alternative to SMOTE
    - Implement **dynamic thresholding** based on transaction risk factors
2. **Monitoring**
    - Track concept drift in fraud patterns quarterly
    - Establish automated threshold adjustment framework