## **Credit Card Users Default Prediction Using Machine Learning**

### **Participants**

* **Faseeh U Rehman Qureshi (110188603)**
* **Yuhan Xiang (110182721)**

Both participants have equally and collaboratively contributed to all aspects of the project including data loading, preprocessing, modeling, analysis, tuning, and documentation.

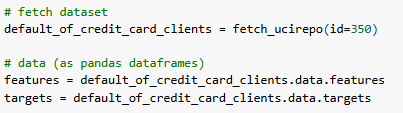
The source code and experimental notebook are available on GitHub:<https://github.com/faseeh-quraishi/Credit_Card_Users_Default_Prediction.git>.

### **1. Problem Definition**

Credit card default prediction is crucial for financial institutions to minimize credit risks and ensure financial stability. The aim of this project is to develop a machine learning model that can accurately predict whether a customer will default on their credit card payment in the next month using historical financial and demographic data.

### **2. Data Collection & Loading**

The dataset used is the **“**[**Default of Credit Card Clients Dataset**](https://archive.ics.uci.edu/dataset/350/default+of+credit+card+clients)**” (UCI ID: 350)**, accessed via the ucimlrepo Python package.



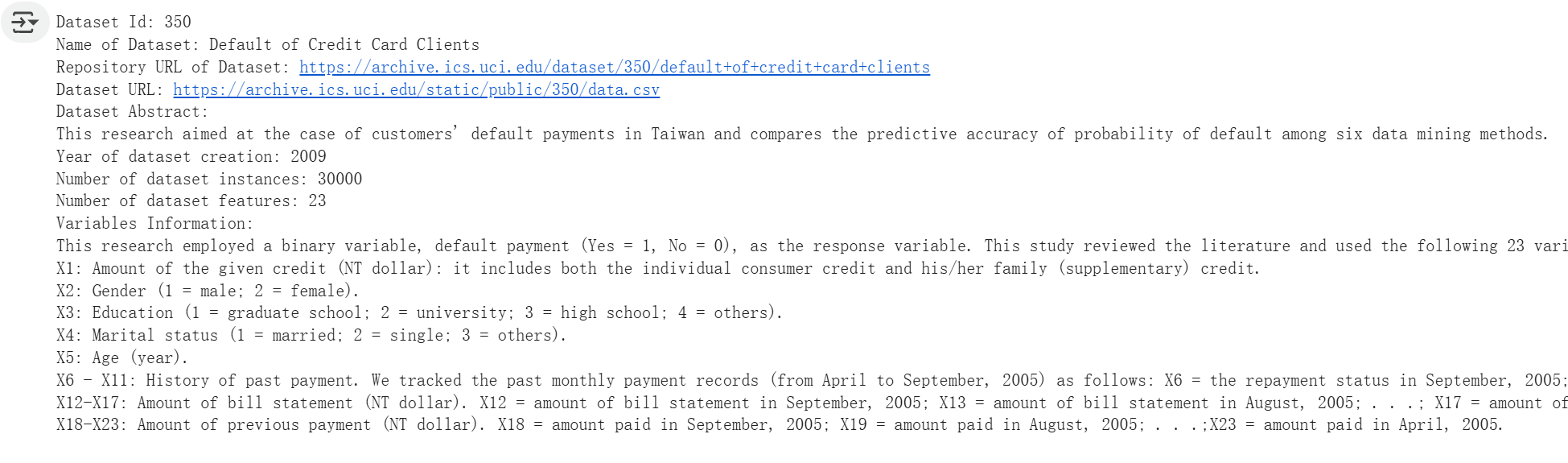
### **3. Exploratory Data Analysis (EDA)**

The EDA phase provides critical insights into the structure, quality, and relationships within the dataset, setting the foundation for robust modeling. The following analyses were performed:

#### 3.1. Dataset Metadata Overview

The analysis begins with a detailed extraction and printing of metadata attributes, which includes:

* **Dataset ID:** Retrieved from default\_of\_credit\_card\_clients.metadata.uci\_id.
* **Name:** The official name of the dataset.
* **Repository & Data URLs:** URLs for repository and direct data access.
* **Abstract:** A concise description of the dataset's purpose and background.
* **Year of Creation:** Indicates when the dataset was compiled.
* **Number of Instances and Features:** Summary statistics on the dataset’s size.
* **Variable Information:** Description of each variable in the dataset, aiding in data comprehension.

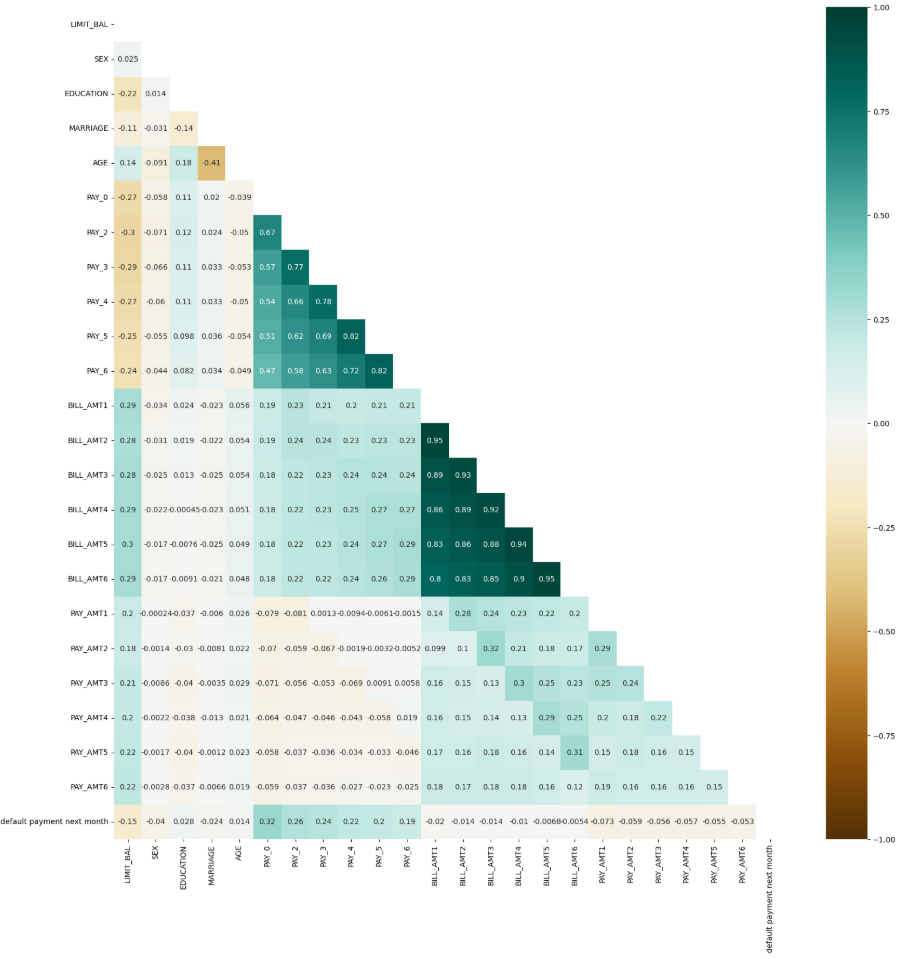


This metadata provides crucial context for understanding the nature and scope of the dataset before proceeding to detailed analysis.

#### 3.2. Correlation Matrix and Heatmap

A correlation heatmap was generated to explore linear relationships among all numeric features in the dataset. The following insights were derived:

* Strong correlations were observed among some features, particularly related to billing and payment amounts over successive months.
* Features with high inter-correlation may introduce multicollinearity and could be candidates for dimensionality reduction or feature engineering.

The relationship between individual features and the target variable was also assessed to identify potentially influential predictors.  
 

#### 3.3. Distribution of Numerical Features by Target Variable

Boxplots were used to visualize the distribution of each numerical feature (e.g., credit limit, age, bill amounts) across the two classes of the target variable (default payment next month). These visualizations highlighted:

* Clear differences in feature distributions between defaulters and non-defaulters.
* Presence of outliers and potential skewness in financial variables.
* Features that may exhibit strong predictive signals based on class separation.

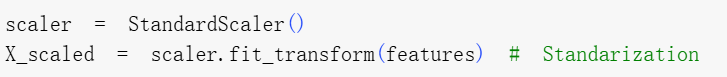
This step aids in understanding which variables may influence the likelihood of default.

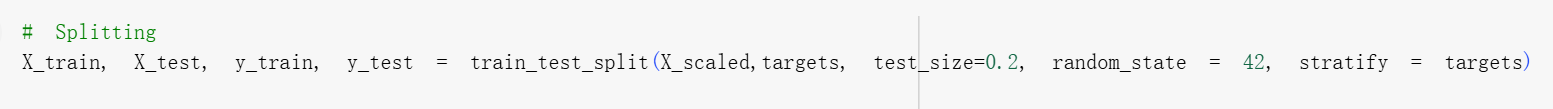
### **4. Data Preprocessing**

Effective preprocessing is vital for building robust and accurate machine learning models.

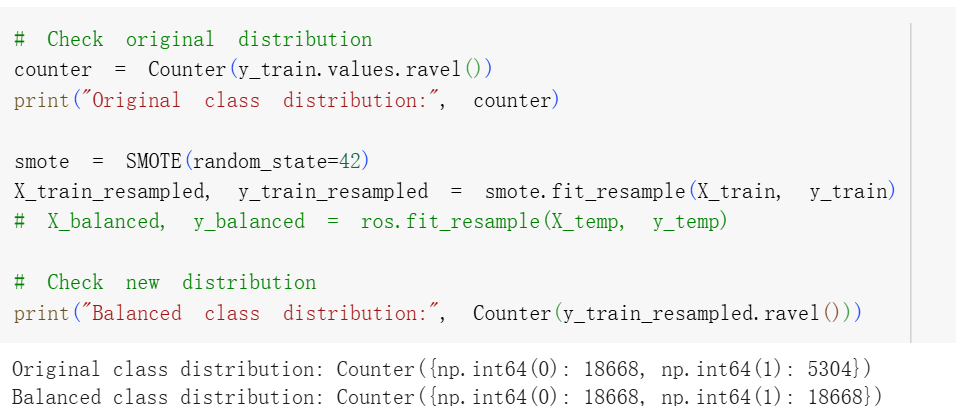
#### Data Removal: to ensure data integrity and avoid model bias from repeated records

#### Feature Scaling: StandardScaler was applied for models sensitive to magnitude (e.g., LR, KNN, GaussianNB). It helps ensure that features with larger numerical ranges (e.g., LIMIT\_BAL, BILL\_AMT1-6) do not disproportionately influence the model



* Train-Test Split: 80/20 split ensured valid model evaluation.

#### Addressing Class Imbalance: Only about 22% of clients defaulted, leading to class imbalance. To address this, SMOTE was used to generate synthetic minority samples, helping balance the data and improve the model’s ability to detect defaulters without overfitting.



#### **5.Model**

This section presents the implementation and evaluation of four machine learning models used to predict credit card default: Decision Tree, Logistic Regression, K-Nearest Neighbors, and Gaussian Naive Bayes. Each model was trained on the resampled dataset using SMOTE to address class imbalance and was evaluated on an 80:20 train-test split.

#### 5.1 Decision Model Trees

The Decision Tree Classifier was selected for its interpretability and ability to model non-linear relationships. Three versions of the model were trained with varying levels of depth and regularization:

| **Model** | **Parameters** | **Train Acc** | **Test Acc** | **Notes** |
| --- | --- | --- | --- | --- |
| **DTCa** | depth=5 | 71.03% | 77.37% | Balanced, generalizable |
| **DTCb** | depth=7 | 72.35% | 78.14% | Best trade-off |
| **DTCc** | depth=10 | 76.46% | 72.80% | Overfitting risk |

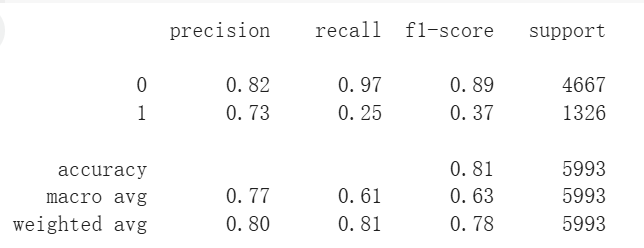
Overall, **Model DTCb** was selected as the best among the three due to its balance of bias and variance.

#### 5.2 Logistic Regression Model

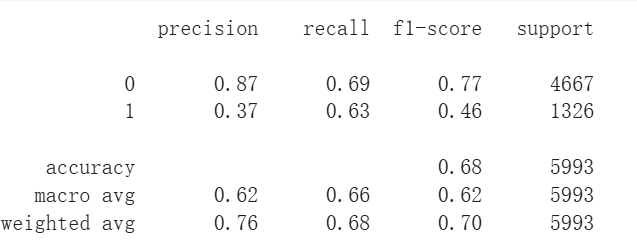
Logistic Regression was chosen for its simplicity and robustness in binary classification tasks. This model requires standardized input features, which were prepared using StandardScaler.

Logistic Regression was evaluated in four stages:

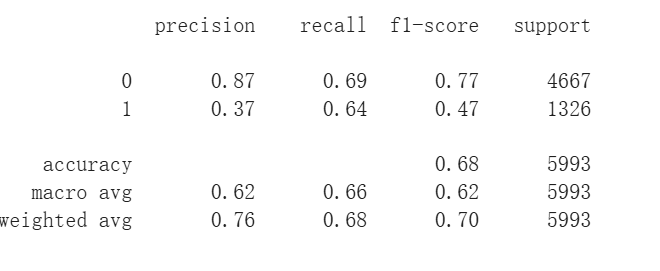
* **Performance before class balancing:**
  + The model performed well on the majority class (non-default) but poorly on the minority class.



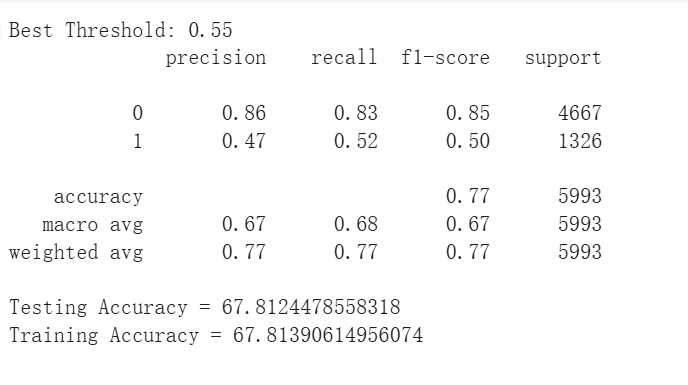
* **After SMOTE balancing:**
  + Improved recall and F1-score for the minority class.
  + Slight drop in accuracy, but better class balance.



* **After hyperparameter tuning (C=0.003):**
  + Regularization strength tuned using GridSearchCV.
  + F1-score increased to 0.47, recall increased to 0.64, and accuracy remained unchanged
  + Best parameters found C=0.003, solver='liblinear’**:**



* **Threshold tuning (Threshold=0.55):**
  + Applied Youden’s J statistic to find optimal threshold.
  + F1-score improved to **50%**, precision increased to **0.47%**.
  + Accuracy slightly increased to **77%**, indicating more balanced performance.



Logistic Regression performed well overall, offering interpretability and decent accuracy with improved fairness after rebalancing and tuning.

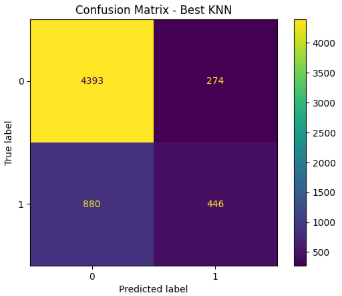
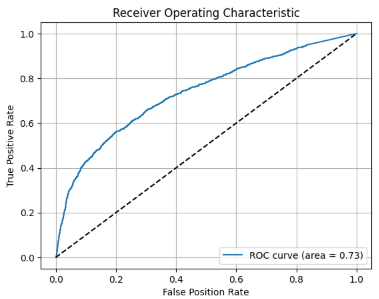
#### 5.3 K-Nearest Neighbour Model

K-Nearest Neighbors (KNN) is a non-parametric algorithm that classifies instances based on the majority class of their closest neighbors in the feature space.

* **Model Training**The KNN model was initially trained using default parameters. It predicted test instances by comparing them to the most similar training samples. Basic evaluation metrics showed acceptable performance.
* **Hyperparameter Tuning**To improve accuracy, Grid Search with 5-fold cross-validation was used. The tuning explored different values of k, with distance-based weighting and Euclidean distance. The best configuration (metric:euclidean, k:15, weight:distance) was selected based on the F1 score.

#### **Performance**

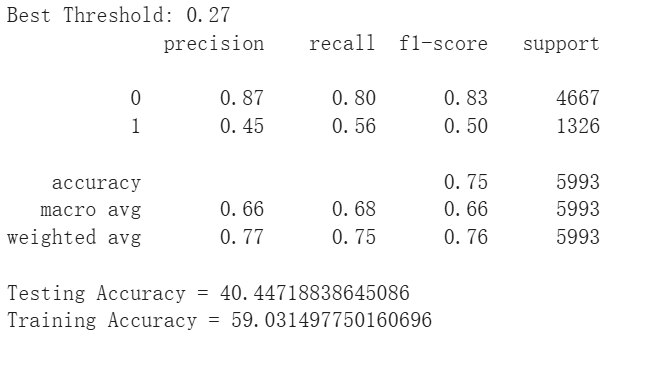
* + **Train Accuracy:** 100.0%, **Test Accuracy:** 80.5%
  + High precision, poor recall (conservative predictions)
  + Likely overfitting due to perfect training accuracy
  + Sensitive to feature scaling and computationally slow on large datasets

The model's performance was also assessed using ROC-AUC and confusion matrix plots.

#### 5.4 Gaussain Naive Bayes

Gaussian Naive Bayes was applied as a probabilistic baseline model assuming feature independence.Assuming Gaussian and independent features:

* **Train Accuracy**: 59.0%, **Test Accuracy**: 40.5%
* High precision, poor recall (conservative)
* Fast but underperforms due to strong assumptions



This model offered fast training and prediction times but suffered from its strong independence assumptions. It served as a good baseline but underperformed compared to other models.

### **6. Model Comparison**

| **Model** | **Train Acc** | **Test Acc** | **Precision(y=1）** | **ROC-AUC** |
| --- | --- | --- | --- | --- |
| Logistic Regression (C=0.003) | 68% | 68% | 0.47 | 0.72 |
| Decision Tree (DTCb) | 72% | 78% | 0.51 | 0.73 |
| KNN (k=15) | 100% | 81% | 0.62 | 0.73 |
| Gaussian NB（ var\_smoothing=1e-05） | 59% | 40% | 0.45 | 0.72 |

### **7. Conclusion & Insights**

This study evaluated four classification models—Logistic Regression, Decision Tree, K-Nearest Neighbors, and Gaussian Naive Bayes—for predicting credit card default using UCI dataset #350. The performance of each model was compared using accuracy, precision, and ROC-AUC.

Among all models, **Decision Tree (DTCb)** achieved the best test accuracy (78%) and solid balance between bias and variance. **Logistic Regression**, with threshold tuning and regularization (C=0.003), offered interpretable results and stable performance (AUC = 0.72), making it suitable for practical deployment. **KNN (k=15)** showed the highest test accuracy (81%) but suffered from severe overfitting (100% train accuracy), limiting its generalizability. **Gaussian Naive Bayes** offered fast computation and reasonable baseline performance, though its strong independence assumptions limited its recall and overall effectiveness.

In conclusion, **Decision Tree and Logistic Regression** emerged as the most reliable models for default prediction, offering a trade-off between interpretability, robustness, and performance. Future improvements may include incorporating ensemble methods like Random Forest or XGBoost, and applying feature selection to reduce dimensionality and overfitting.