Credit Card Score Prediction Using Machine Learning Models

1 Introduction

Credit score is a statistical model used by lenders to assess the creditworthiness of potential borrowers. The model typically uses a variety of factors, such as the borrower's credit history, income, and debts, to generate a score that represents the borrower's likelihood of repaying a loan. Lenders use credit scores to make decisions about whether to approve a loan application and, if so, what interest rate to charge. Credit scores can also be used to determine the credit limits on a credit card.

There are a number of different credit scoring models, but they all share some common features. First, they all use a combination of positive and negative factors to assess a borrower's creditworthiness. Second, they all assign a higher score to borrowers who have a history of paying their debts on time and in full. Third, they all take into account the borrower's current financial situation, such as their income and debts.

In recent years, machine learning models have been increasingly used in credit scoring systems to help identify and predict potential default customers. These models use a variety of data, such as the customer's credit history, income, and debts, to generate a score that represents the customer's likelihood of defaulting on a loan. Machine learning models can be more accurate than traditional credit scoring models in predicting default risk. This is because machine learning models can take into account a wider range of data and can learn from past data to improve their accuracy.

2 Methodology

The following subsections describe the machine learning models used in this study.

2.1 Logistic Regression Model

Logistic regression is a statistical and machine learning model used primarily for binary classification tasks. This model is typically employed when the class label in the problem is categorical and has only two possible outputs, commonly referred to as '0' and '1,' 'yes' and 'no,' or 'positive' and 'negative.' The primary objective of logistic regression is to predict the probability of the class label for a given set of features (input).

Logistic regression maps input variables from linear regression to a probability score between 0 and 1 using the sigmoid function, as shown in Equation (1):

$$f(z) = \frac{1}{1 + e^{-z}}$$

where

$$z = \beta_0 + \sum_{k=1}^{m} \beta_k x_k$$

Here, β_0 indicates the intercept of the algorithm, and $\beta_1, \beta_2, \ldots, \beta_k$ are the coefficients of the independent variables $(x_1, x_2, \ldots, x_k)$, and m is the number of independent variables (features).

Logistic regression is widely used in many fields, including medicine, finance, and marketing, making it worth implementing and investigating in our credit card scoring system.

2.2 Decision Tree

A decision tree is a machine learning algorithm used for both classification and regression problems. It is a graphical representation of the decision-making process that forms a complete tree structure. Each internal node in the decision tree represents a decision or a test on an input variable, and the outcomes of these tests are represented by branches. Each leaf node represents a class label of the problem domain.

The advantages of using decision trees include their ease of understanding and interpretation, minimal data preparation requirements, and low computational cost, which scales logarithmically with the number of data samples used for training. However, decision trees are sensitive to overfitting the training data if not properly pruned.

2.3 Random Forest

Random Forest is a widely used machine learning algorithm for both classification and regression tasks. It extends the concept of decision trees by constructing multiple trees on different subsets of the data and aggregating their predictions through majority voting for classification tasks. One of its primary advantages lies in its capability to handle complex datasets effectively while handling the risk of overfitting. This attribute makes it a robust model known for its high predictive accuracy

2.4 XGBoost (Extreme Gradient Boosting)

XGBoost is a powerful and high-speed machine learning algorithm belonging to the gradient boosting family. It builds a series of decision trees sequentially, with each tree correcting the prediction errors of the previous one. These models are then combined to compute the final output. XGBoost offers several advantages, including the inclusion of L_1 and L_2 regularization terms to reduce overfitting, and tree pruning to minimize complexity. The objective function is shown in Equation (3):

$$\text{obj}(t) = \sum_{i=1}^{n} l(y_i, \hat{y}_i) + \sum_{i=1}^{t} \Omega(f_t)$$

where

$$\Omega(f) = \frac{1}{2}\lambda \sum_{j=1}^{l} w_j^2$$

Here, w_j represents the weights of the leaves and λ is the regularization parameter.

2.5 LightGBM (Light Gradient Boosting Machine)

LightGBM is an efficient, high-speed, and scalable machine learning model that belongs to the gradient boosting family. It sequentially constructs an ensemble of decision trees to en hance predictive accuracy. LightGBM utilizes gradient-based one-sided sampling (GOSS) to expedite the training process. Another technique it employs is exclusive feature bundling (EFB), which can combine exclusive features that take nonzero values simultaneously. EFB serves as a dimensionality reduc tion method by merging certain features.

3 DATA PREPARATION

The following subsections describe the data source and the preprocessing steps that have been done to this dataset.

3.1 Data Sources

- **Development Data (Dev_data_to_be_shared.csv):**
  - Contains 96,806 records.
  - Includes the **target variable**: `bad_flag` (1 for default, 0 for non-default).
  - Features:
    - **On-us attributes**: Related to internal account characteristics, e.g., credit limit.
    - **Transaction-level attributes**: E.g., the number of transactions and their values.
    - **Bureau tradeline attributes**: E.g., product holdings and historical delinquencies.
    - **Bureau enquiry attributes**: E.g., credit inquiries in the past three months.
- **Validation Data (validation_data_to_be_shared.csv):**
  - Contains 41,792 records.
  - Features are identical to the development data but **exclude the target variable** (`bad_flag`).

3.2 Insights from the given Development Dataset

3.2.1 High Dimensionality

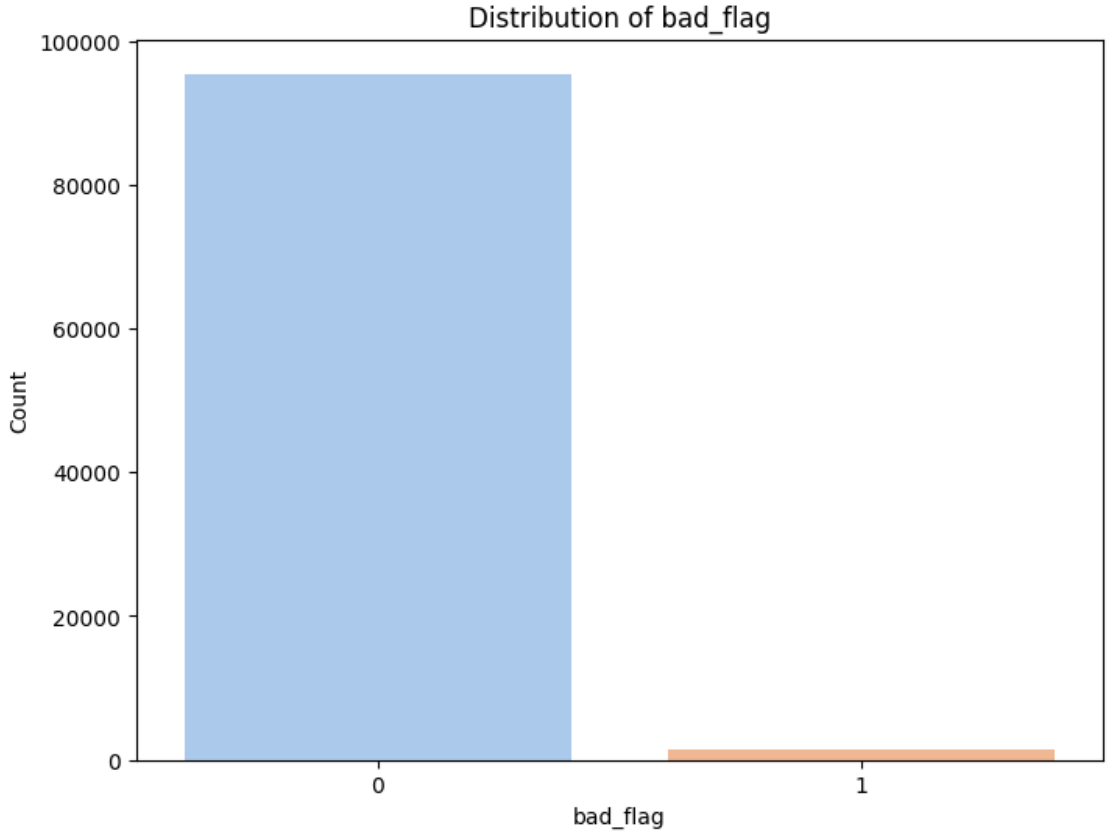
The dataset contains a large number of features, which can lead to the curse of dimensionality. This can make it challenging to process the data, especially when training machine learning models. Many features may also be irrelevant or redundant.

3.2.2 Missing Data and Inconsistencies

There are missing values (NaNs) and possible inconsistencies in the data, which can cause issues during model training. Handling missing data through imputation or deletion can be time-consuming.

3.2.3 Imbalanced Data

The dataset is highly imbalanced, where one class is significantly more frequent than the other. This can impact model performance, leading to biased predictions favoring the majority class.



3.3 Data Preparation

3.3.1 Load Data

Loaded the development and validation datasets into Pandas DataFrames.

3.3.2 Missing Value Analysis

Computed the percentage of missing values for each feature, dropping columns with missing values exceeding 1.232%. This threshold was determined through iterative experimentation to balance data retention and quality. It ensured that essential information was preserved

while removing columns with excessive missing data that could undermine model reliability. Also, the thought here was that if some columns have lot of missing values then it must be less important to the bank.

- Computed the percentage of missing values for each feature.
- Imputed missing values with column medians for both datasets.
- Dropped columns with missing values exceeding 1.232%.

3.3.3 Seperating the Target Variable

- Dropped irrelevant columns from X (e.g., account_number) which have no role in prediction.
- Separated features ($\mathbf{X}$) and target ($\mathbf{y}$), where `bad_flag` was the target variable.

3.4 Exploratory Data Analysis (EDA)

3.4.1 Feature Importance

Used a Random Forest Classifier to rank features by importance. Retained the top 100 features based on their importance.

3.5 Dimensionality Reduction

- Standardized features using `StandardScaler`.
- Performed PCA to capture 97% of variance.
- Identified features contributing less than 1% to PCA components and removed them.

3.6 Data Balancing

3.6.1 Addressing Class Imbalance

Observed that the `bad_flag` target variable was highly imbalanced, with the majority class significantly outnumbering the minority class. To address this, SMOTE was used to oversample the minority class to 3% of the majority class. This specific percentage was chosen to balance the need for additional data points with the risk of overfitting. Increasing the representation of the minority class to 3% ensured that the models had enough data to learn from, without creating an artificially balanced dataset that might skew real-world performance. This approach also helped maintain computational efficiency while improving the model's ability to predict default cases.

- Observed that `bad_flag` was imbalanced.
- Applied SMOTE to oversample the minority class to 3% of the majority class.

3.7 Model Development

3.7.1 Custom Validation Set

Created a balanced validation set with equal representation of both classes (50 samples each). Ensured no overlap between training and validation sets.

3.7.2 Model Training

Selected five models for training:

- Logistic Regression
- Decision Tree Classifier
- Random Forest Classifier
- XGBoost
- LightGBM

These models were chosen for their complementary strengths. Logistic Regression served as a baseline due to its interpretability. Decision Tree Classifier provided insights into hierarchical decision-making. Random Forest and LightGBM were included for their robust performance on structured datasets, leveraging ensemble learning and gradient boosting techniques, respectively. XGBoost was selected for its superior handling of class imbalance, scalability, and ability to extract complex patterns from the data.

4 EVALUATION METRICS

Evaluation metrics are important measures to assess the performance of machine learning models. These metrics also play a crucial role in comparing ML models against each other and discussing how well each model is performing. All evaluation metrics are calculated based on four types of classifications: true positive (TP), false positive (FP), true negative (TN), and false negative (FN). The explanation of these four is as follows:

- **TP:** This represents positive data that has been correctly classified as positive.
- **FP:** These are negative data points that have been incorrectly classified as positive.
- **TN:** This category encompasses negative data points that have been correctly classified as negative.
- **FN:** FN includes positive data points that have been incorrectly classified as negative.

The following four subsections explain the four evaluation metrics that are used in this project.

4.1 Accuracy

Accuracy is the most widely used classification performance metric. It calculates the number of correctly classified samples by the model out of the total samples in the dataset. Accuracy is calculated as explained in Equation 1.

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{FP} + \text{TN} + \text{FN}} \tag{1}$$

4.2 Precision

Precision is a metric used to evaluate the accuracy of correctly classified positive samples by the model. It is calculated by dividing the number of TP by the sum of TP and FP. In simpler terms, precision measures how many of the classified TP instances are actually positive. Equation 2 explains how precision is calculated.

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}} \tag{2}$$

4.3 Recall

Recall is a metric used to evaluate the ability of the model in classifying all positive samples correctly. It is also known as sensitivity or the true positive rate. Recall is calculated by dividing the number of TP by the sum of TP and FN. The equation for recall is explained below in Equation 3:

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}} \tag{3}$$

In simpler terms, recall measures how well the model identifies all positive examples in the dataset. A high recall value indicates that the model is effective at minimizing false negatives and is successful in correctly identifying most of the positive cases.

4.4 Area Under the ROC Curve (AUC-ROC)

Area Under the ROC Curve (AUC-ROC) is an evaluation metric used to measure the area under the Receiver Operating Characteristic (ROC) curve. This curve plots the True Positive Rate (Sensitivity) against the False Positive Rate (1 — Specificity) at various threshold values. The area under the ROC curve, often referred to as the AUC, quantifies the model's ability to distinguish between positive and negative samples. The AUC value falls within the range of 0.5 to 1. A perfect classifier is represented by an AUC value of 1, while an AUC of 0.5 indicates that the model performs no better than random guessing. In general, a higher AUC-ROC score indicates a better-performing model, and it is often used to compare different models' performance on the same dataset.

Model	Training Accuracy	Validation Accuracy	AUC-ROC
Logistic Regression	0.97	0.50	0.70
Decision Tree	1.00	0.77	0.76
Random Forest	1.00	0.72	0.96
XGBoost	0.79	0.88	0.91
LightGBM	0.99	0.74	0.95

Table 1: Model Comparison Results

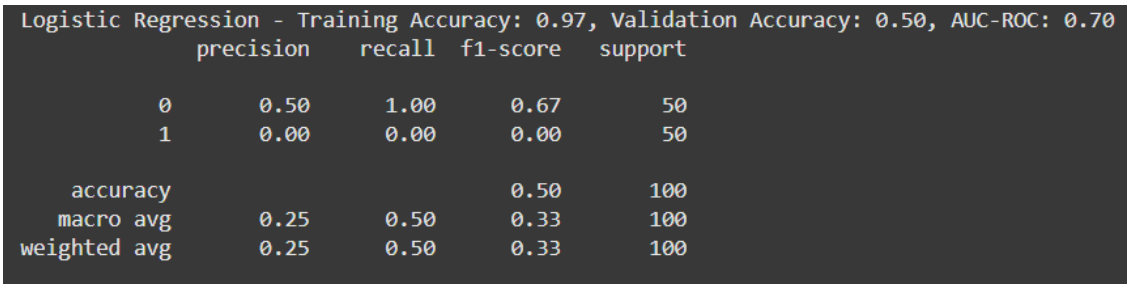


Figure 1: Logistic Regression

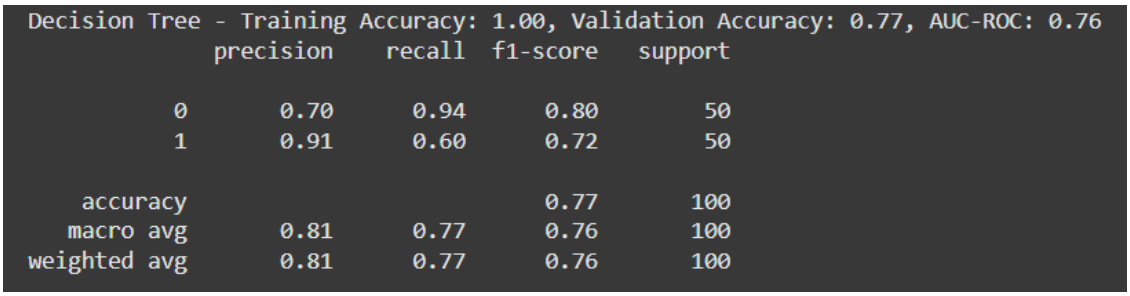


Figure 2: Decision Tree

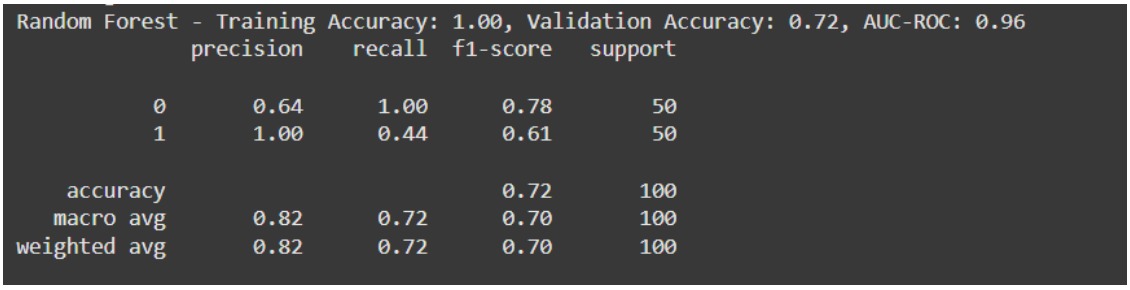


Figure 3: Random Forest

XGBoost - Training Accuracy: 0.79, Validation Accuracy: 0.88, AUC-ROC: 0.91

	precision	recall	f1-score	support
0	0.90	0.86	0.88	50
1	0.87	0.90	0.88	50
accuracy			0.88	100
macro avg	0.88	0.88	0.88	100
weighted avg	0.88	0.88	0.88	100

Figure 4: XGBoost

LightGBM - Training Accuracy: 0.99, Validation Accuracy: 0.74, AUC-ROC: 0.95

	precision	recall	f1-score	support
0	0.66	1.00	0.79	50
1	1.00	0.48	0.65	50
accuracy			0.74	100
macro avg	0.83	0.74	0.72	100
weighted avg	0.83	0.74	0.72	100

Figure 5: LightGBM

5 Conclusion

- **Logistic Regression:** This showed an impressive Training Accuracy but failed miserably in the validation Accuracy. Also, its AUC-ROC is pretty low.
- **Decision Tree:** This model showed a perfect 1.00 accuracy in the Training Set which is a clear indication of Overfitting. Also, its AUC-ROC is pretty low.
- **Random Forest:** This model also showed a perfect 1.00 accuracy in the Training Set while the Validation set showed only 0.72 accuracy which is a clear indication of Overfitting
- **XGBoost:** It has a pretty good Training accuracy which is neither too high nor too low. Also, it has the highest Validation Accuracy. This makes the model robust and accurate for different datasets. Also, it has a high AUC-ROC.
- **LightGBM:** This model also showed a near perfect 0.99 accuracy in the Training Set while the Validation set showed only 0.74 accuracy which is a clear indication of Overfitting

Final Selection : XGBoost