Lending Club Loan: Exploratory Data Analysis, Classifications, Predictions

Yiru Fang, Qiyang Wang, Andrew Cao

December 2024

1 Introduction

The rapid rise of online lending platforms has revolutionized the financial industry, offering individuals and businesses convenient access to loans. However, the growth of this sector also introduces significant challenges related to profitability and risk management. With default risks and varying borrower profiles, these platforms must balance profitability while managing potential losses effectively.

Our curiosity lies in understanding how these platforms maintain their profitability despite the inherent risks in lending. Specifically, we aim to identify the key factors that contribute to successful loan repayments and highlight areas that pose a higher risk of default. By addressing these questions, this analysis provides insights that can aid in better decision-making for online lending platforms.

The objective of this project is to implement exploratory data analysis (EDA) and machine learning models to uncover key factors influencing profitability and risk. Through statistical and machine learning methods, we aim to:

- Identify features that play a critical role in predicting loan repayment.
- Evaluate and compare various models to determine the most effective approach for prediction.
- Provide actionable insights for improving risk management strategies.

This report serves as a comprehensive guide to understanding the factors that drive performance in the online loan market, bridging the gap between data insights and strategic decisions.

2 Data Description

The dataset used for this analysis originates from Kaggle's Lending Club dataset, which includes information on loan applications and their repayment statuses. The dataset comprises 26 variables, with "loan status" serving as both the target and an explanatory variable. Loan statuses are classified into three categories: "Fully Paid," "Charged Off," and "Current." Our analysis focuses on two outcomes—"Fully Paid" and "Charged Off"—to examine borrower creditworthiness by assessing loans that were either successfully repaid or defaulted. The dataset contains 77673 instances of "Charged Off" loans, revealing a significant imbalance between the two classes.

Figure 1 illustrates the distribution of loan statuses, highlighting this imbalance.

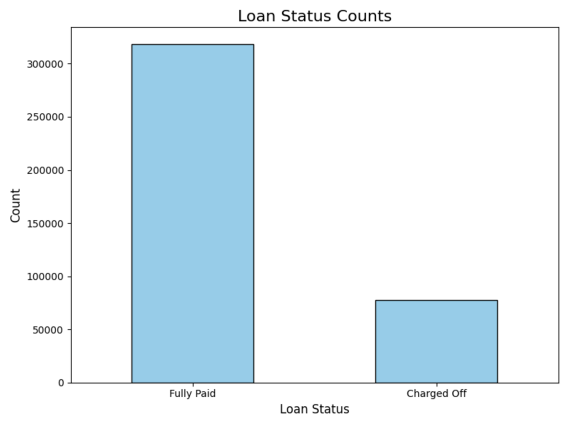


Figure 1: Count of Loan Status

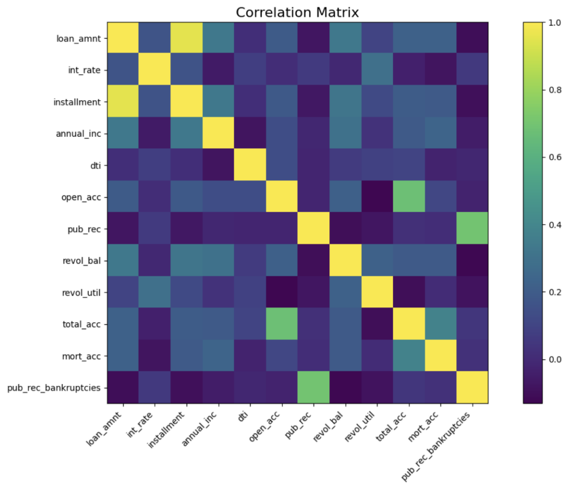


Figure 2: Distributions of `installment` and `loan_amnt`

To prevent multicollinearity among explanatory variables, we conducted a correlation analysis. The correlation matrix revealed a strong relationship between "loan amount" and "installment amount." Although these variables exhibit high correlation, their distributions relative to the target variable demonstrated notable differences. Consequently, both were retained for further analysis to preserve important distinctions in the data.The correlation analysis, as visualized in Figure 2, also informed decisions about feature selection.

Figure 3 depicts the distributions of "installment" and "loan amount" categorized by loan statuses:

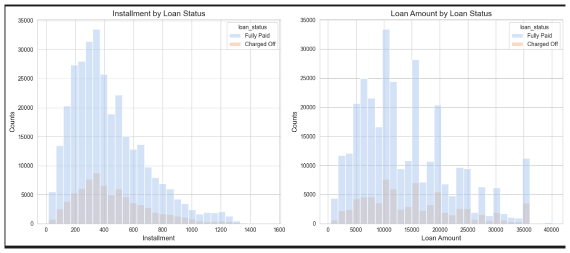


Figure 3

Upon reviewing categorical variables, we identified that `grade` and `sub_grade` offer overlapping information. While "grade" categorizes loans broadly from A to G, `sub_grade` provides finer granularity, ranging from A1 to G5. As these features showed similar trends, `grade` was removed to reduce redundancy, retaining `sub_grade` for its higher level of detail. Other variables, such as `emp_title`and `emp_length`were removed due to their limited relevance or redundancy after initial exploration.

Figure 4 below illustrates the distribution of "grade" and "subgrade" by loan status:

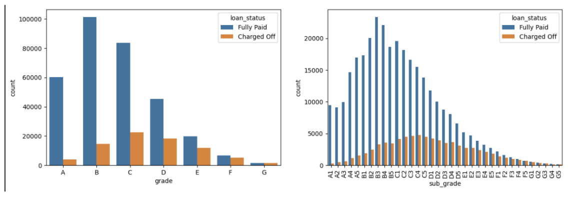


Figure 4: Districution of `grade` and `sub_grade`

3 Data Engineering

After exploring the main information in the dataset, we performed several data engineering steps to prepare the dataset for analysis. These steps are detailed below: To enhance our dataset, we began by eliminating redundant or unnecessary features like Grade, Emp_title and Emp_length; Grade was specifically removed due to its duplicative information with subgrade. Emp_title and emp_length features were removed due to their limited relevance or significant missing values that could not be reliably imputed. After that, we addressed missing data in remaining features by employing strategies such as removing rows with excessive missing data and employing statistical measures such as median or mean for continuous variables to impute missing values. Moving forward, we converted categorical variables like home_ownership and verification_status into numerical format using one-hot encoding for compatibility with machine learning models and to avoid creating ordinal relationships where none existed before. As part of our data preparation for modeling, we used the MinMaxScaler function from sklearn to scale continuous variables from 0-1 using MinMaxScaler to ensure all features have the same scale and prevent models from favoring variables with larger magnitudes. As part of this step, we divided the dataset into training and testing sets with an 80:20 split using the train_test_splin function from sklearn to ensure model performance can be evaluated on new data, providing an accurate representation of generalization. To address class imbalance between "Fully Paid" and "Charged Off" loans, we employed the Synthetic Minority Oversampling Technique (SMOTE). This technique generates synthetic samples for minority classes such as "Charged Off," by interpolating existing instances to generate synthetic samples for that minority class, thus balancing our dataset and improving model performance on minority predictions.

4 Analyzing Algorithm

In this part we applied 3 methods to analyze the most important features.

4.1 Chi-square Test

4.1.1 Why Chi-Square Test

The Chi-Square test is ideal for identifying relationships between categorical features and the target variable, `loan_status`, in this case. It helps to quickly identify whether a feature is statistically significant for predicting the target. This method is simple to implement, computationally efficient, and provides clear results in the form of Chi-Square scores and p-values, making it suitable for feature selection in this context. The Chi-Square test measures the difference between observed and expected frequencies in categorical data. The test outputs:

- **Chi-Square Score (χ^2)**: A higher score indicates a stronger association.
- **P-Value**: A small p-value (< 0.05) indicates statistical significance.

The formula for the Chi-Square statistic is:
$$\chi^2 = \sum \frac{(O_i - E_i)^2}{E_i}$$

Where O_i and E_i represent the observed and expected frequencies, respectively.

4.1.2 Result

The results of applying the Chi-Square test are shown below:

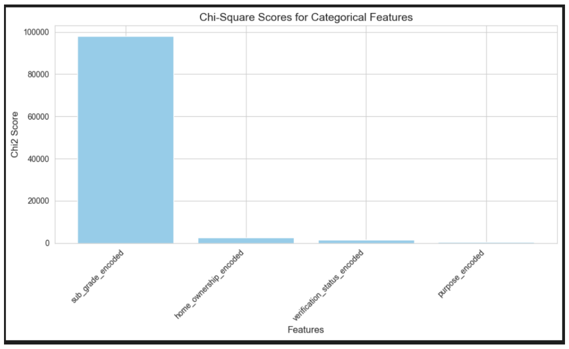
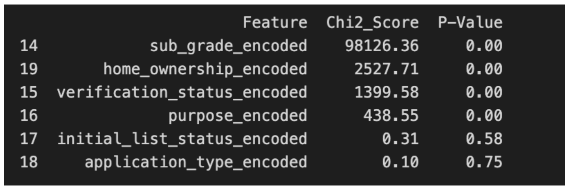


Figure 5: Chi-Square Scores for Categorical Features



```
                        Feature  Chi2_Score  P-Value
14            sub_grade_encoded    98126.36     0.00
19       home_ownership_encoded     2527.71     0.00
15  verification_status_encoded     1399.58     0.00
16              purpose_encoded      438.55     0.00
17  initial_list_status_encoded        0.31     0.58
18     application_type_encoded        0.10     0.75
```

Figure 6: Chi-Square Test Results Table

We find:

- `sub_grade_encoded` has an extremely high Chi-Square score (98126.36) and a p-value of 0.00, indicating a strong correlation with loan status.
- Features like `home_ownership_encoded`, `verification_status_encoded`, and purpose also show significant relationships (low p-values).
- Features like `initial_list_status_encoded`and `application_type_encoded`have high p-values (¿ 0.05), suggesting no significant relationship with loan status. These features can be disregarded in the modeling phase.

The Chi-Square test provided clear insights into which categorical features significantly impact the target variable. Features with high scores and low p-values will be retained for the predictive model, while insignificant features will be excluded to simplify the model and avoid noise.

4.2 Principle component analyze (PCA)

4.2.1 Why PCA

Principal Component Analysis (PCA) is a powerful technique for dimensionality reduction. It is used here to simplify the dataset by reducing the number of features while retaining the majority of the variance in the data. This helps to:

- Improve computational efficiency.
- Reduce multicollinearity between features.
- Identify and focus on the most impactful variables for modeling.

PCA works by finding new axes (principal components) that maximize the variance in the data. These axes are linear combinations of the original features and are orthogonal to each other. Mathematically, PCA involves:

1. **Standardizing the data** to have zero mean and unit variance.

2. **Calculating the covariance matrix** (Σ) of the features:

$$\Sigma = \frac{1}{n}\sum_{i=1}^{n}(X_i - \bar{X})(X_i - \bar{X})^T$$

3. **Performing eigen decomposition** of the covariance matrix to obtain eigenvalues (λ) and eigenvectors (v).

4. **Sorting principal components**: Eigenvalues represent the variance explained by each principal component. Components are ranked by decreasing eigenvalues.

5. **Choosing the number of components** based on a cumulative variance threshold (e.g., 95%).

The total variance explained by the first k components is given by:

$$\text{Cumulative Variance} = \frac{\sum_{i=1}^{k}\lambda_i}{\sum_{i=1}^{m}\lambda_i}$$

4.2.2 Result

The results of applying the PCA are shown below:

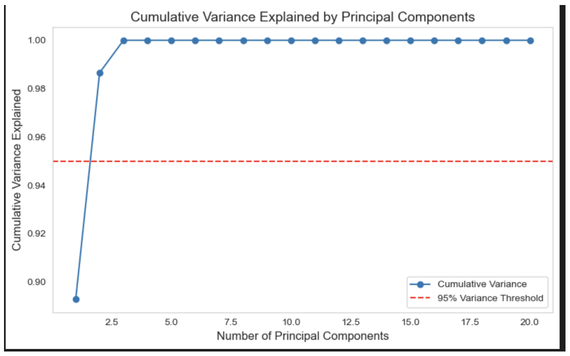


Figure 7: Cumulative Variance Explained by PCs

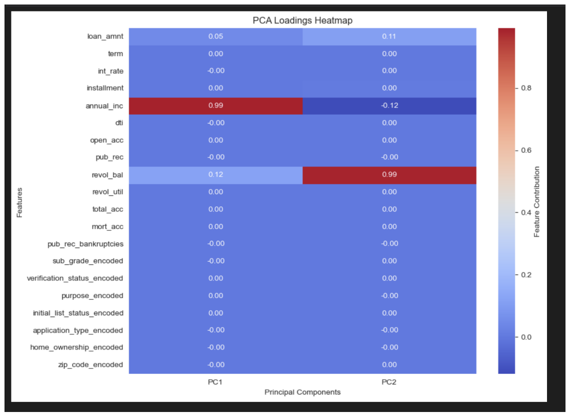


Figure 8: PCA Heatmaps

We find:

- A 95% explained variance threshold was set to identify the optimal number of principal components (PCs).

- As shown in Figure 8, two principal components were sufficient to capture 95% of the variance in the dataset.

- The PCA loadings heatmap in Figure 9 highlights the contributions of features to the principal components: PC1 is heavily influenced by annual income (0.99);PC2 is strongly associated with revolving balance (0.99).

4.3 Lasso Regularization

4.3.1 Why Lasso Regularization

Least Absolute Shrinkage and Selection Operator (Lasso) is used here for feature selection and dimensionality reduction. By adding a penalty term to the loss function, Lasso shrinks less important feature coefficients to zero, effectively removing irrelevant features. This helps in:

- Simplifying the model by reducing the number of features.
- Enhancing interpretability and avoiding overfitting.

Lasso regression modifies the standard linear regression loss function by adding an L_1-norm penalty. The objective function for Lasso is:

$$\text{Loss} = \frac{1}{2n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 + \lambda\sum_{j=1}^{p}|\beta_j|$$

Where:

- y_i: Observed value,
- $\hat{y}_i$: Predicted value,
- β_j: Coefficient of the j-th feature,
- λ: Regularization strength (controls the penalty).

When λ increases:

- Less significant coefficients are forced to zero.
- Only the most relevant features are retained.

4.3.2 Result

The results of applying the Lasso Regularization are shown below:

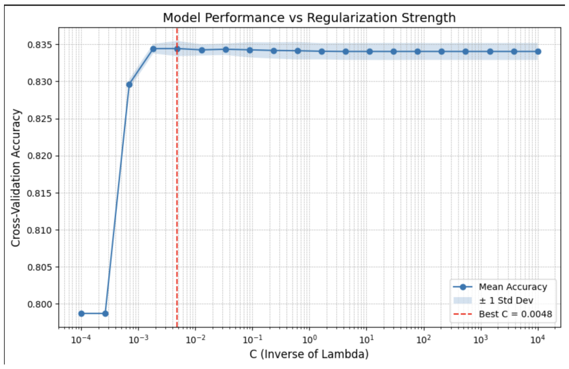
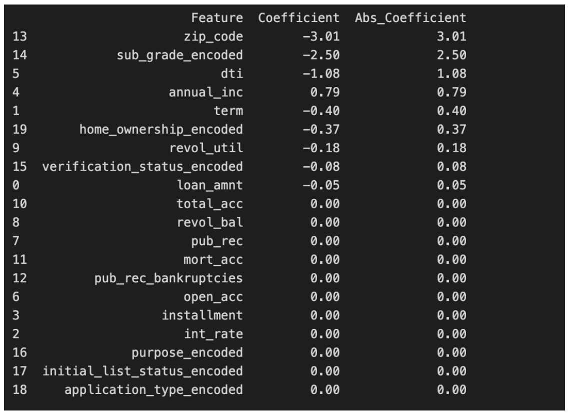


Figure 9: Model performance v.s. C



```
                      Feature  Coefficient  Abs_Coefficient
13                   zip_code        -3.01             3.01
14          sub_grade_encoded        -2.50             2.50
5                         dti        -1.08             1.08
4                  annual_inc         0.79             0.79
1                        term        -0.40             0.40
19     home_ownership_encoded        -0.37             0.37
9                  revol_util        -0.18             0.18
15 verification_status_encoded       -0.08             0.08
0                   loan_amnt        -0.05             0.05
10                  total_acc         0.00             0.00
8                   revol_bal         0.00             0.00
7                     pub_rec         0.00             0.00
11                   mort_acc         0.00             0.00
12       pub_rec_bankruptcies         0.00             0.00
6                    open_acc         0.00             0.00
3                 installment         0.00             0.00
2                    int_rate         0.00             0.00
16            purpose_encoded         0.00             0.00
17 initial_list_status_encoded        0.00             0.00
18   application_type_encoded         0.00             0.00
```

Figure 10: Feature Coefficients

We find:

- Setting C=0.0048 (as shown in Figure 11) provided the best trade-off between regularization strength and model accuracy.
- After applying Lasso, features such as `zip_code`(-3.01), `sub_grade_encoded`(-2.50), `dti`, and `anual_inc` and other 5 variables had non-zero coefficients (Figure 12), indicating their importance in predicting loan status.

- Features like `total_acc` were shrunk to zero, indicating they are less impactful and can be excluded from further analysis.

Lasso Regularization effectively reduced the dimensionality of the dataset while retaining key variables like `zip_code`, `sub_grade_encoded` and `dti`. This method enhances model efficiency and interpretability by focusing only on the most influential features. These selected features should be prioritized in the final model.

5 Prediction Method

In this part we apply 4 methods for prediction, we only considered the important features we obtained from the previous part.

5.1 K-Nearest Neighbor (KNN)

5.1.1 Why KNN

KNN is a simple, non-parametric classification algorithm that makes no assumptions about the data distribution. It is well-suited for datasets where relationships between data points can be effectively captured by proximity in feature space. This method:

- Classifies objects based on the majority vote of their nearest neighbors.
- Allows flexibility by tuning the parameter k, the number of neighbors.

KNN assigns a class to a data point based on the majority class of its k nearest neighbors. The key steps include:

1. **Calculate Distance:** Compute the distance between the query point and all other points in the dataset. A common metric is the Euclidean distance:
$$d(i,j) = \sqrt{\sum_{m=1}^{n} (x_{i,m} - x_{j,m})^2}$$
2. **Sort Neighbors:** Rank all data points by their distance to the query point.
3. **Select k Nearest Neighbors:** Identify the k closest data points.
4. **Vote:** Assign the class label that occurs most frequently among the k neighbors.

5.1.2 Result

The results of applying the KNN are shown below:

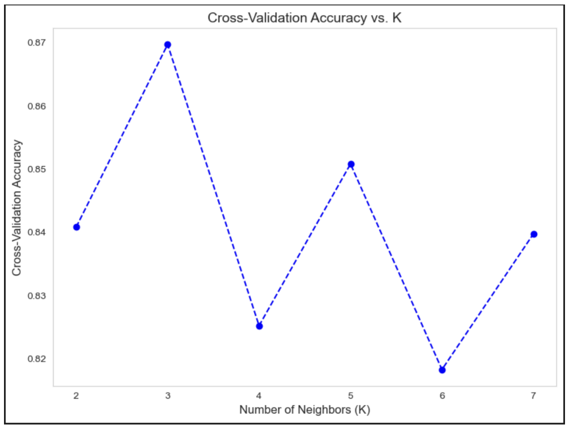
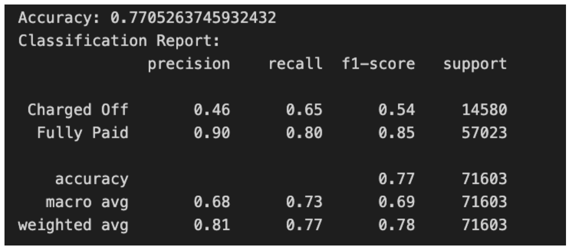


Figure 11: Cross-Validation Accuracy v.s. K



```
Accuracy: 0.7705263745932432
Classification Report:
              precision    recall  f1-score   support

 Charged Off       0.46      0.65      0.54     14580
  Fully Paid       0.90      0.80      0.85     57023

    accuracy                           0.77     71603
   macro avg       0.68      0.73      0.69     71603
weighted avg       0.81      0.77      0.78     71603
```

Figure 12: Classification Report

- Cross-validation (Figure 14) identified k=3 as the optimal number of neighbors, achieving the highest accuracy of 0.87.
- F1-score for the majority class (”Fully Paid”) is high (0.85), indicating strong performance. F1-score for the minority class (”Charged Off”) is lower (0.54), highlighting challenges in handling imbalanced datasets.
- Overall accuracy: 0.77, with a macro-average F1-score of 0.69.

KNN performed well on the majority class but struggled with minority classes due to the dataset’s imbalance. While it is straightforward and interpretable, its reliance on distance metrics and sensitivity to class imbalance suggest it could benefit from additional techniques like oversampling or distance-weighted voting in future iterations.

5.2 K Means

5.2.1 Why K Means

K-Means is a widely used unsupervised learning algorithm for clustering data points into groups based on their similarity. It is simple, efficient, and performs well for datasets where clusters are spherical and separable in feature space. This method was selected to identify inherent patterns or groupings within the data. K-Means aims to partition the dataset into k clusters by minimizing the sum of squared distances (inertia) between each data point and its assigned cluster centroid. The key steps are:

1. **Initialize Centroids:** Randomly select k points as initial centroids.
2. **Assign Clusters:** Assign each data point to the nearest centroid based on the Euclidean distance:

$$d(i,c) = \sqrt{\sum_{m=1}^{n}(x_{i,m} - c_m)^2}$$

3. **Update Centroids:** Compute the new centroid for each cluster as the mean of all data points assigned to it.
4. **Iterate:** Repeat steps 2 and 3 until centroids converge or a maximum number of iterations is reached.

The optimal k is determined using the **Elbow Method**, which plots inertia against k. The point where inertia starts decreasing more slowly is chosen as k.

5.2.2 Result

The results of applying the K Means are shown below:

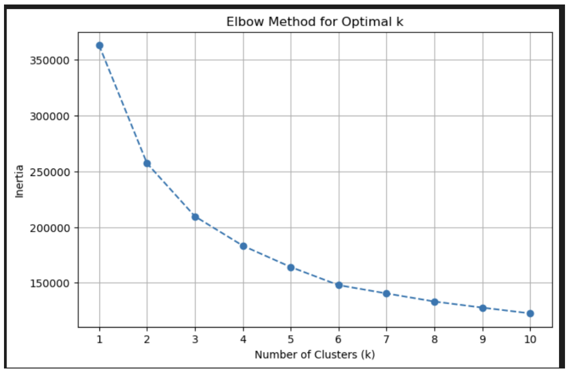
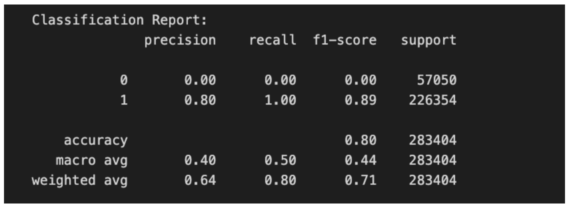


Figure 13: Wlbow Method v.s. K



```
Classification Report:
              precision    recall  f1-score   support

           0       0.00      0.00      0.00     57050
           1       0.80      1.00      0.89    226354

    accuracy                           0.80    283404
   macro avg       0.40      0.50      0.44    283404
weighted avg       0.64      0.80      0.71    283404
```

Figure 14: Classification Report

We find:

- As shown in Figure 17, the inertia sharply decreases until k=6, after which the rate of decrease becomes more gradual. Hence, k=6 was selected as the optimal number of clusters.
- Despite applying resampling techniques to balance the data, K-Means failed to effectively separate clusters corresponding to different labels. This is evident in Figure 18, where F1-scores for one class are excellent (0.89), but the other class has an F1-score of 0, indicating complete misclassification for that group and overall accuracy is 0.40, with a macro-average F1-score of 0.44.

5.3 Logistic Regression

5.3.1 Why Logistic Regression

Logistic regression is a widely used statistical method for binary classification tasks due to its simplicity, interpretability, and effectiveness in estimating probabilities of binary outcomes. It is particularly useful for datasets with linear relationships between features and the target variable. This method was chosen for its ability to provide clear probabilities and insights into the importance of each feature through coefficients.

Logistic regression models the relationship between independent variables (X) and a binary target variable (y) using the logistic function:

$$P(y=1|X) = \frac{1}{1+e^{-(\beta_0+\beta_1 X_1+\beta_2 X_2+\cdots+\beta_n X_n)}}$$

Where:

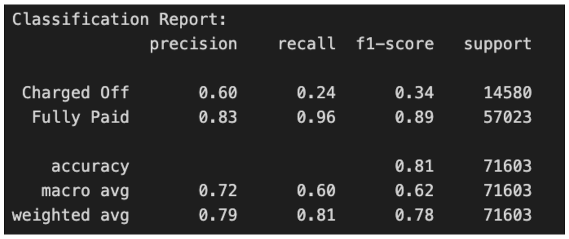
- $P(y=1|X)$: Probability of the positive class,
- β_0: Intercept,
- $\beta_1, \beta_2, \ldots, \beta_n$: Coefficients for features $X_1, X_2, \ldots, X_n$.

The model optimizes the log-loss function to estimate coefficients:

$$\text{Log-Loss} = -\frac{1}{n}\sum_{i=1}^{n}[y_i \log(\hat{y}_i) + (1 - y_i)\log(1 - \hat{y}_i)]$$

5.3.2 Result

The results of applying the Logistic Regression are shown below:



```
Classification Report:
              precision    recall  f1-score   support

 Charged Off       0.60      0.24      0.34     14580
  Fully Paid       0.83      0.96      0.89     57023

    accuracy                           0.81     71603
   macro avg       0.72      0.60      0.62     71603
weighted avg       0.79      0.81      0.78     71603
```

Figure 15: Classification Report

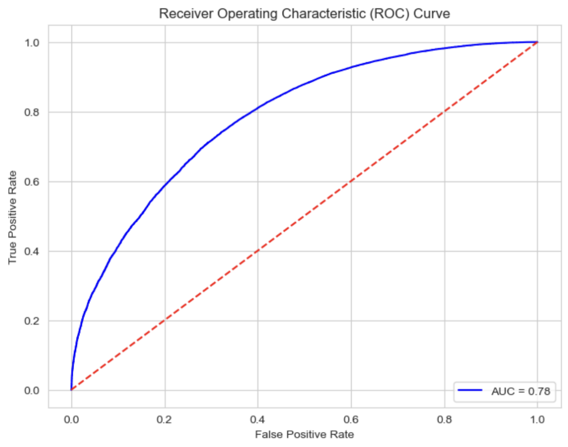


Figure 16: AUC-ROC

We find:

- Logistic regression achieved high precision (0.83) and F1-score (0.89) for the majority class ("Fully Paid"). However, performance on the minority class ("Charged Off") was poor, with an F1-score of 0.34 and recall of 0.24, indicating that the model struggles with imbalanced data.
- The ROC curve shows moderate performance, with an AUC of 0.75, reflecting a decent ability to distinguish between the two classes.

5.4 Random Forest

5.4.1 Why RF

Random Forest is an ensemble method to improve classification accuracy by creating multiple decision trees. This approach excels at handling datasets containing complex linear and nonlinear relationships and helping identify crucial features. By training each tree with different data and feature subsets (bagging and feature sampling), overfitting is reduced while generalization increases significantly. The core steps of the algorithm are:

1. **Bootstrap Sampling:** Create multiple random subsets of the data.
2. **Grow Decision Trees:** Train individual decision trees on these subsets.
3. **Split at Each Node:** Use a random subset of features to determine the best split, minimizing Gini impurity:

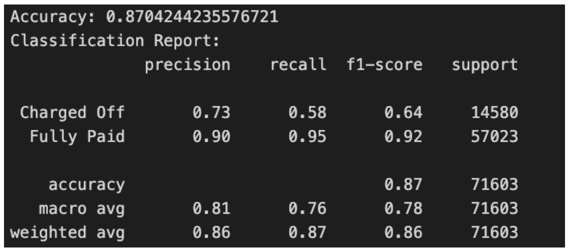
$$G = 1 - \sum_{i=1}^{C} p_i^2$$

   Where p_i is the proportion of samples in class i at a node.

4. **Aggregate Predictions:** Combine outputs of all trees (majority voting for classification).

5.4.2 Result

The results of applying the Random Forest are shown below:



```
Accuracy: 0.8704244235576721
Classification Report:
              precision    recall  f1-score   support

 Charged Off       0.73      0.58      0.64     14580
  Fully Paid       0.90      0.95      0.92     57023

    accuracy                           0.87     71603
   macro avg       0.81      0.76      0.78     71603
weighted avg       0.86      0.87      0.86     71603
```

Figure 17: Classification Report

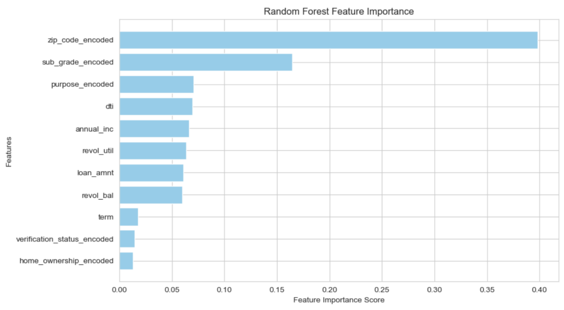


Figure 18: Feature Importance

We find:

- Random Forest achieved the highest accuracy (0.87) among all models tested. The F1-score for the minority class ("Charged Off") was 0.64, significantly better than other methods, showing improved performance on imbalanced data.
- Key features contributing to the model include `zip_code_encoded`, `sub_grade_encoded`, and `purpose_encoded` as shown in the importance chart.

Random Forest outperformed other models with its robust accuracy and F1-scores, demonstrating its capability to handle complex, non-linear relationships in the data. Its feature importance analysis added value by highlighting the most critical predictors, making it the optimal model for this study. Future improvements could involve fine-tuning hyperparameters to further enhance performance.

6 Conclusion

In this project, we employed seven methods - three for capturing feature importance and four for prediction. With the analyzing method we capture the most important features and applied them to prediction. Here is a comparison table for the four prediction method:

Table 1: Performance Comparison of Models

Method	**Accuracy**	**F1-Score (Minority)**	**Strengths**	**Weaknesses**
KNN	77%	0.54	Simple, effective for majority class	Poor for minority class, sensitive to imbalance
K-Means Clustering	40%	0.00	Identifies clusters without supervision	Failed to separate meaningful clusters, even after resampling
Logistic Regression	81%	0.34	Interpretable, good for linear relationships	Weak performance on minority class, sensitive to imbalance
Random Forest	87%	0.64	High accuracy, balanced performance across classes, feature importance insights	Computationally intensive, requires hyperparameter tuning

From the comparison, Random Forest stands out as the optimal model, achieving the highest accuracy (87%) and a significantly better F1-score (0.64) for the minority class. While Logistic Regression provided interpretability, its poor recall for the minority class limited its effectiveness. KNN showed reasonable performance but struggled with class imbalance. K-Means failed to identify meaningful clusters, demonstrating its unsuitability for this dataset. For future work, there are some potential improvements. Addressing Class Imbalance has had a dramatic effect on all models; using techniques such as Adaptive Synthetic Sampling (ADASYN) or undersampling the majority class could create more balanced datasets and strengthen minority class predictions. Hyperparameter Tuning allows Random Forest's performance to be further improved by carefully tweaking hyperparameters such as number of trees, maximum depth, and minimum samples per split using techniques such as Grid Search or Bayesian Optimization. Feature Engineering also presents an opportunity to enhance performance across models by adding interaction terms or normalizing continuous variables in order to boost predictive power. Random Forest has proven effective when used alongside advanced models; however, more experimental models such as Gradient Boosting (e.g. XGBoost and LightGBM) or Neural Networks may provide greater potential outcomes. Furthermore, Explainability frameworks such as SHAP can provide more comprehensive insight into feature importance and model behavior, helping guide decision-making processes more easily. By addressing these aspects, the overall model performance can be improved, and more reliable, actionable insights can be derived from the data. This holistic approach ensures the solution is robust, interpretable, and aligned with the dataset's characteristics and real-world constraints.

References

[1] Nathan George, "Lending Club Dataset," Kaggle. [Online]. Available: `https://www.kaggle.com/datasets/wordsforthewise/lending-club/data`. [Accessed: Dec. 10, 2024].