Anticipate Possible Loan Delinquencies and Determine the Most Impactful Drivers

Ashwini Sharma Richard Correia Panduga Raja Tejasvi

1 Motivation

The motivation is to learn and see if there is an analytical means to anticipate the potential for a loan to become delinquent, given size, type and many other aspects of the client's management of debt. It would also be interesting to see if the differences between the initial grading, # of credit/revolving accounts, types of loans, term of loan had any sort of significant impact on the delinquencies or not. Delinquencies are a major concern for banks and other credit institutions, and so if there was an automatic and analytical means of predicting potential problems, the creditors could get ahead of the problem and better manage those potential cases and lessen the loss.

2 Significance

The significance of our project is to address the major concern faced by banks and credit institutions regarding loan delinquencies. The project intends to give creditors a way to foresee future issues and handle cases of delinquencies more effectively by using analytical models. This will contribute to minimizing financial losses and optimizing strategies for risk mitigation.

3 Objectives

The goal is to connect the data to a prediction, if possible. Given the categorical data that we have, referenced below, we want to run through the categories and determine which ones have the most correlation and predictability towards success or failure of the loan. As an example, the combination of term and interest rate may end up having a significant indication as to where loans fail or succeed, and if they do, it would also be great to see if there is truly any difference when you are comparing the different terms and rates, relative to each other, for the failures. The main objective would be to create a model that would accomplish the goal and through analysis, infer that there is a direct correlation between some of the categories and the prediction of delinquency and non-delinquency. To find if there are any differences in delinquencies or successful loans based on loan structure, client history, credit lines, amount paid relative to loan amount, and reported debt/income ratios. Test the formulated hypotheses regarding the impact of different loan terms and interest rates, varying levels of debt-to-income ratios, and distinct levels of revolving balances and credit lines on delinquencies and non-delinquencies.

4 Related Work (Background)

The reference for the existing work has been taken from the Kaggle platform. After data preprocessing and Exploratory Data Analysis on the dataset, they have fitted the training data on different models like Logistic Regression, Decision Tree, Random Forest, K-Nearest Neighbors, and Gaussian Naive Bayes models. Decision Tree Classifier achieved a perfect accuracy of 1.0 on the training data, which might indicate that the model is overfitting the training data. Random Forest Classifier also achieved a remarkably high accuracy of 0.99 on training dataset, suggesting a strong fit to the training data whereas there is drop in test dataset accuracy which is 0.908. Logistic Regression and K-Nearest Neighbors Classifier have good but slightly lower accuracies compared to the decision tree-based models. Gaussian Naive Bayes has the lowest accuracy among the models. The analysis identifies features such as 'Total Received Late Fee,' 'Loan Amount,' 'Home Ownership,' 'Interest Rate,' 'Funded Amount Investor,' and 'Collection Recovery Fee' as significant predictors for loan default. There are a few drawbacks involving the performance of the model and feature selection. To improve performance of the model on the test data, hyperparameter tuning or resampling has not been performed in some references (required as which the data is not balanced and there might be dominance of one class over the other). The model could have been fitted on various other machine learning algorithms and in-depth feature selection to avoid the issue of overfitting. Also, the effectiveness of all the columns in all combinations has not been taken into consideration. To deal with the high dimensionality datasets Principal Component Analysis could be employed to understand the variability in the dataset. Additionally, training loss or testing loss also must be analyzed to assess the model.

Dataset

When customers default on the loan repayments, the banks undergo severe financial losses which impact their stability. To understand why a certain customer defaults his on-time repayments, we have used the dataset which consists of various attributes such as interest rate, funded amount, balance etc., that are used to predict the likelihood of a person becoming a loan defaulter.

The dataset consists of a train and test dataset. There are around 67,463 rows and 35 columns in the training set and 8,913 rows and 34 columns in the testing dataset [1]. With the help such datasets which serves as an excellent tool for predictive modeling, risk evaluation, and detail financial analysis, enabling the bankers to make informed judgments about the loan approvals and risk mitigation tactics and overall strategy planning to lend loan to certain type of customers.

Attributes:

1. ID: Unique identifier for each loan.
2. Loan Amount: The amount of money requested for the loan.
3. Funded Amount: The total amount funded for the loan.
4. Funded Amount Investor: The amount funded by investors.
5. Term: The duration of the loan in months.
6. Batch Enrolled: Loan enrollment Batch ID.
7. Interest Rate: The interest rate for the loan.
8. Grade: Loan grade categorization.
9. Sub Grade: Sub classification within a given grade.
10. Employment Duration: Borrower employment duration.
11. Home Ownership: Type of home ownership.
12. Verification Status: Verification status of the borrower's information.
13. Payment Plan: Whether the loan has a payment plan or not.
14. Loan Title: Purpose of the loan.

15. Debt to Income: Ratio of debt to income.
16. Delinquency - two years: Number of delinquencies in the last two years.
17. Inquiries - six months: Number of inquiries made in the last six months.
18. Open Account: Number of open credit accounts.
19. Public Record: Number of public records.
20. Revolving Balance: Amount of revolving balance.
21. Revolving Utilities: Percentage of total available revolving credit.
22. Total Accounts: Total number of accounts.
23. Initial List Status: Initial listing status of the loan.
24. Total Received Interest: Total interest received till date.
25. Total Received Late Fee: Total late fees received.
26. Recoveries: Amount recovered after charge-off.
27. Collection Recovery Fee: Fee charged for collections.
28. Collection 12 months Medical: Collection amount for medical expenses in the last 12 months.
29. Application Type: Individual or joint application.
30. Last week Pay: Past week payment.
31. Accounts Delinquent: Number of delinquent accounts.
32. Total Collection Amount: Total amount collected.
33. Total Current Balance: Current balance of all accounts.
34. Total Revolving Credit Limit: Total revolving credit limit.
35. Loan Status: The status of the loan (e.g., defaulted (0) or paid (1)).

5 Detail design of Features

Data Preprocessing:

To clean and organize the data for efficient analysis and modeling we are performing various data preprocessing steps. This involves operations to handle the missing values, segregate numeric and categoric columns, transform and extract features.

I. Handling missing values:

We detect the missing values in the dataset. The method used to check the presence of missing values is **‘isna().sum()’** method from the pandas library. This method lists the missing values count in corresponding columns.

According to the output, it indicates 0 values across all the columns which tells us that we are having zero missing values. This is an advantage for smooth analysis and model building without having to handle the missing values with various strategies.

II. Checking for duplicate values:

To avoid the overfitting of the model, we need to check the presence of duplicates in the dataset. Using the **'duplicate().sum()'** method we can check the duplicates in our datasets. The result indicates that there are no duplicates in the dataset.

III. Segregation of Numeric and categorical columns:

Segregating the numeric and categorical columns is an important data preprocessing step as numeric and categorical columns require different handling techniques. Numeric columns must go through scaling and normalization whereas categorical columns must use encoding techniques like one-hot encoding or label encoding [2]. It can help us differentiate between the various variables to better understand the relation between each other and their influence on the output column.

When we segregate the numeric and categorical column in our dataset, we can observe that there are 9 categorical columns ('Batch Enrolled', 'Grade', 'Sub Grade', 'Employment Duration', 'Verification Status', 'Payment Plan', 'Loan Title', 'Initial List Status', 'Application Type') and 26 numeric columns.

'Grade', 'Employment Duration', 'Verification Status', 'Initial List Status', and 'Application Type' columns have a limited number of distinct values. We can thus use one hot encoding technique to handle these columns. Whereas columns such as ‘Batch Enrolled', 'Sub Grade', 'Loan Title' have high distinct values which has considerable number of unique values needs to be handled using other techniques as one hot encoding will increase the number of columns drastically.

IV. Feature relevance:

It is important to evaluate the feature relevance as it allows us to understand the significance of certain features and impact of feature on the output.

Certain attributes like ‘payment plan’ have only one value which is 'n' and is irrelevant for the analysis. This column can be dropped. Additionally, the attribute ‘accounts delinquent’ has just one value, which can also be dropped.

Feature relevance can therefore improve the performance of the model by eliminating the noisy information that may reduce the accuracy. Moreover, enabling us to understand the correlation within the dataset more effectively.

6 Analysis

Exploratory Data Analysis (EDA):
We are performing the Exploratory Data Analysis (EDA) for analyzing the dataset. EDA helps us to understand the insights, underlining data patterns and structure. We can also find anomalies or missing values which can significantly affect the accuracy of our predictive model.

Furthermore, EDA also enables us to discover the relationships between various attributes to provide insights into how these attributes can have effect on the target variable. We can therefore prioritize the most impacted attributes over the irrelevant attributes [3].

Additionally, by visualization of the distributed data and identifying their relationships, EDA supports us in developing a suitable approach for preprocessing the data which involves tasks such as normalization, handling the categorical variables, and feature engineering.

Overall, EDA assists to make informed decisions regarding which approach to use for modeling, how we should prepare the data, and what concerns to solve to ensure an efficient analysis.

Additionally, we also used SPSS to compliment what was created in Python through code to break the data down and review characteristics that could help us understand connections between the data. Some of the charts and tables are remarkably similar and corroborate what we were doing through Python.

Descriptive statistics:
The descriptive or summary statistics give us insights of the variability and central

tendencies. They help to understand the distribution of numeric data.

This approach is employed to capture the core components of the data, measures of statistics, graphical representations, and quantitative evaluations. To understand the central tendencies and distribution across numerical data, summary statistics including mean, median, standard deviation, and range are computed.

Further, this approach can be used for the analysis of the categorical variables using frequency distributions graphs to identify important categories. The main objective of descriptive analysis is to plot the frequency distribution, detect anomalies, and find out necessary patterns or trends. Therefore, allowing us for further exploration.

The method **'describe ()'** provides descriptive statistics for every numerical column. The method gives the following information:

- **Count:** It displays the non-null values present in every column. It also identifies the missing values.

- **Mean:** The average of each column which gives us a measure of central tendency.

- **Std (Standard Deviation):** It measures the level of variability in the data from the mean. Greater variability is indicated by higher standard deviation values.
- **Min:** Displays the minimum value in each column.
- **25th Percentile (Q1):** This figure represents the lower quartile, or the location where 25% of the data is found.
- **The median (50th percentile or Q2)** is the value in the middle of the dataset, separating the upper and lower halves.
- **The 75th percentile (Q3)** is the upper quartile, representing the position at which 75% of the data falls.
- **Max:** Displays the maximum value in each column.

Visualization of segregated Categorical data:
On segregating the categorical data from the dataset, we can observe that there are 9 categorical columns ('Batch Enrolled', 'Grade', 'Sub Grade', 'Employment Duration', 'Verification Status', 'Payment Plan', 'Loan Title', 'Initial List Status', 'Application Type').

Categories such as ID, Batch numbers, and initial grading metrics, did not provide any relevant insights and so could be removed from the analysis.

Visualizing the categorical data will help us understand the distribution and to identify imbalances in the data. We can also find the relationship between various categorical attributes and their impact on the target variable. It will also help us eliminate certain attributes which do not contribute in predicting the outcome. Additionally, we can choose which encoding techniques to use on the attribute based in the distribution of the data.

1. **Batch Enrolled:**
   In **Figure-1**, the graph depicts the count of loan status within each batch enrolled. It shows the distribution of loan status across different batches which helps us to identify if certain batch enrollments have higher frequencies of loan defaults.

In the output we can see that there are no great differences in the frequencies of loan defaults or repayments amongst the batches. There is no unusual spike in the loan status. This distribution is homogenous with a certain uniformity. This implies that there is no significant impact of batch enrollment on the loan status.

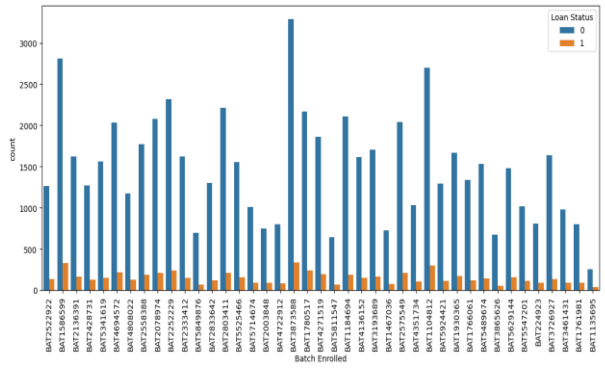


Figure-1: Count plot of loan status within each batch enrolled.

2. **Grade:**
   The plot depicts the distribution of loan status across different grades. Grade C has the largest count of loan status for defaulters, followed by Grade B and, finally, Grade A. Grades D, E, F, and G, on the other hand, have significantly smaller counts. We can thus see that certain grades have higher defaulters as shown in **Figure-2**. They indicate distinguished risk classifications, with Grade A normally representing lower risk and Grades G and F typically representing higher risk.

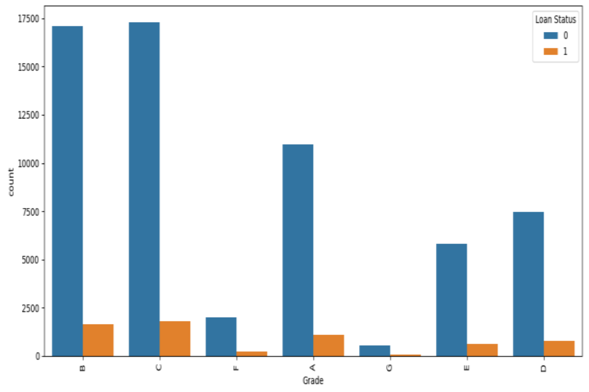


Figure-2: Distribution of loan status across different grades.

3. **Sub Grade:**
   Within each broad 'Grade' category, the 'Sub Grade' column denotes a more specific classification.
   The count plot shows the frequency of loans across several subgrades, providing a specific insight inside each broader grade. Subgrades B4, C1, and B3 appear to have the largest counts, indicating that these specific categories are more prevalent within their respective grades. Subgrades G3, G4, and G5 have significantly lower counts, indicating that they are less common within their higher-risk grade categories as shown in **Figure-3**. For example, subgrades G1 and G2 have higher ratios of defaulted loans compared to other subgrades in the G grade category.

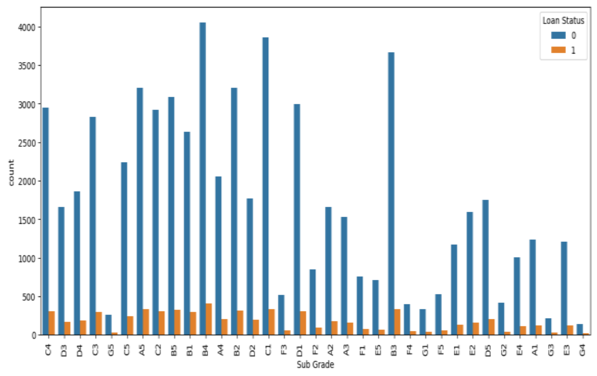


Figure-3: The count plot shows the frequency of loans across several subgrades.

4. **Employment durations (which shows home ownership):**
   In this **Figure-4**, we can observe that home ownership consists of 3 categories, mortgage, rent and own. Depending on the ratio of defaulter's vs total borrowers in each category, the defaulters are present in all the categories. Most of the loans were made to homeowners who have a mortgage, but renters were right behind the homeowners, with a difference of 10,000. Regardless of their ownership status, they all exhibit similar proportional amounts of defaults.

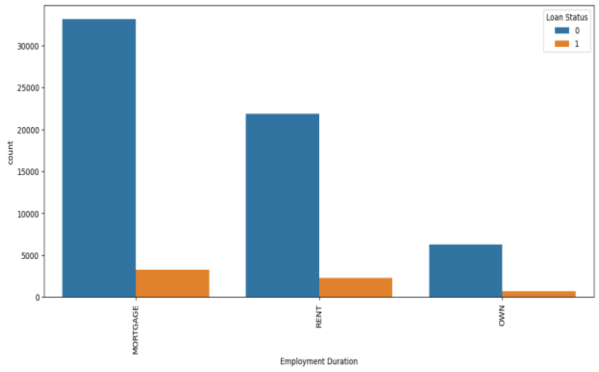


Figure-4: The count plot shows the frequency of loans across the three types of home ownership.

5. **Verification Status:**
   We can see the count plot showing the distribution of loan status based on different verification status in **Figure-5**. 'Source Verified', 'Verified', and 'Not Verified' will be displayed on the x-axis, and the count of loan statuses will be represented on the y-axis. The 'hue' parameter will differentiate the loan status within each verification status category. We can observe that the loan defaulters are more whose verification status is under source verified followed by Verified. Not Verified loan defaulters are comparatively less than verified status borrowers.

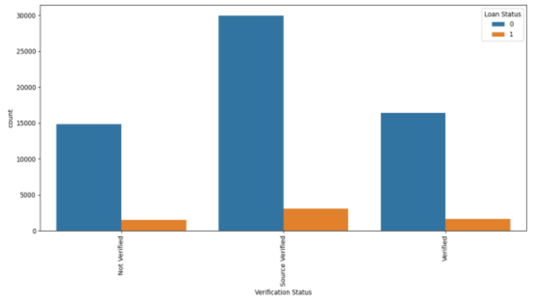


Figure-5: The count plot of the distribution of loan status based on different verification status.

6. **Payment plan:**
   The graph in **Figure-6** shows the count of payment plans where the payment plan count of loan defaulters is lesser compared to repayment loans. The payment plan also consists of a single value which is 'n'. Here, this column is an irrelevant column which does not affect the loan status in any way. Thus, we can drop this column.

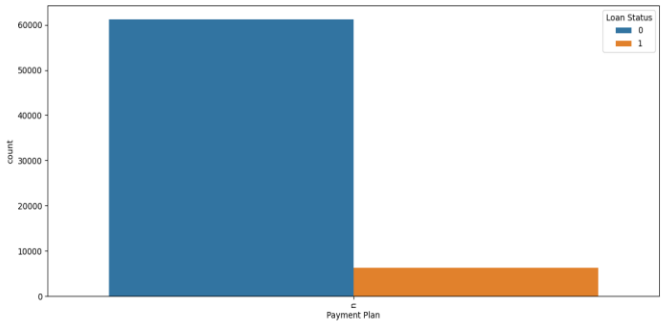


Figure-6: The count of payment plan

7. **Loan Title:**

We can observe the count plot of Loan-title where there are so many loan titles but majority of them belongs to three categories only, which are Credit card financing and debt consolidation. Credit card refinancing has the highest count among the different loan titles, suggesting that a significant number of loans were taken for this purpose. 'Debt consolidation' appears to be another major reason for loan applications, with a substantial count, although it is split into two categories with slight variations in the name ('Debt consolidation' and 'Debt Consolidation') as shown in **Figure-7** and **Figure-8**. Other specific purposes like 'Home improvement', 'Home loan', 'Personal loan', 'Getting Ahead', 'Credit', and 'bills' have comparatively lower counts.

Given the high amounts of loans made for refinancing and debt consolidation, they both carry a high level of risk, since these loans are usually designed with high interest rates and are also unsecured. This can be seen from interest rate frequencies that range from a low of 7-8% to as high as 25%.

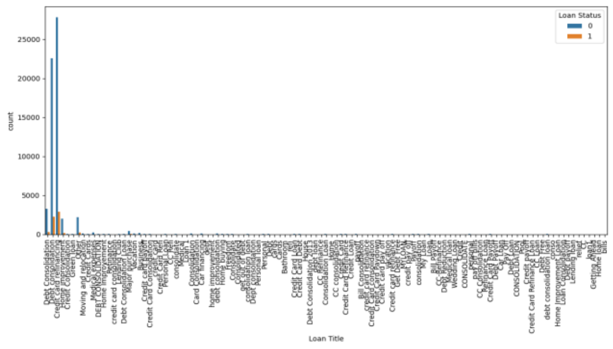


Figure-7: The count plot of Loan Title

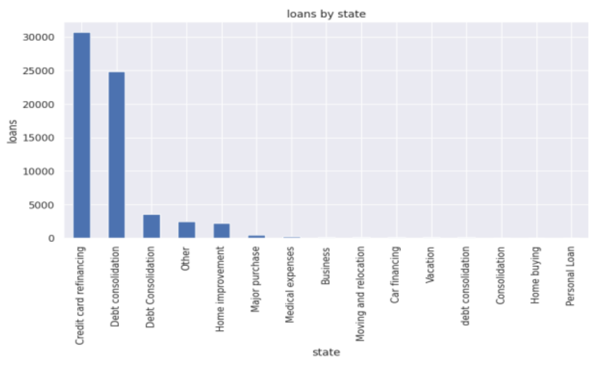


Figure-8: The count plot of each loan title

8. **Initial list status:**
   The count plot for 'Initial List Status' displays the distribution of two categories, 'w' and 'f', providing insights into how loans were initially listed in a dataset. From the graph we can observe that 'w' has a higher count compared to 'f', indicating that the initial list status 'w' is significantly more prevalent in the dataset as shown in **Figure-9**.

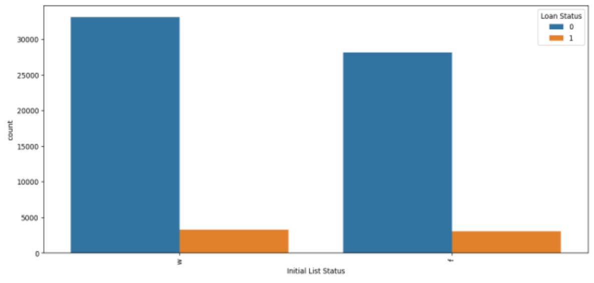


Figure-9: The count plot for Initial List Status

9.Application type:

The 'Application Type' count plot in **Figure-10** depicts the distribution of loan applications into two categories: 'INDIVIDUAL' and 'JOINT'. 'INDIVIDUAL' applications appear to be the most common, outweighing 'JOINT' applications in the dataset**.** This might indicate that individual applications are the standard mode of applying for loans within this dataset.

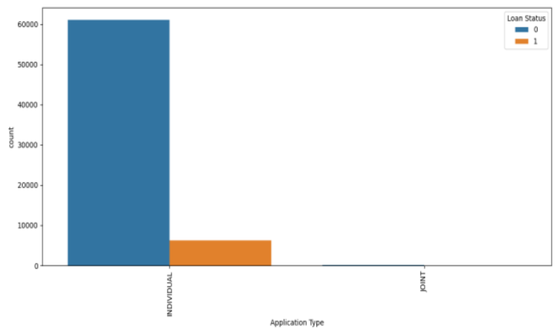


Figure-10: The count plot of Application type

Univariate Analysis:
Univariate analysis within Exploratory Data Analysis (EDA) involves the individual attributes which are considered one at a time to understand the individual distributions and observe the patterns and trends. This will help us analyze each variable's frequency distribution without considering its relationship with any other attribute. This enables us to understand the spread of values and help in decision making regarding specific variables and their impact on target variable.

Visualization of low cardinality numeric columns:

1. **Term**
   The count plot in **Figure-11** shows the distribution of 'Loan Status' across different values of Terms. The dataset consists of 3 different values like 59 months, 58 months, and 36 months. From the observation, the majority of the loans have a term of 59 months whereas smaller number of loans have a term of 36 months.

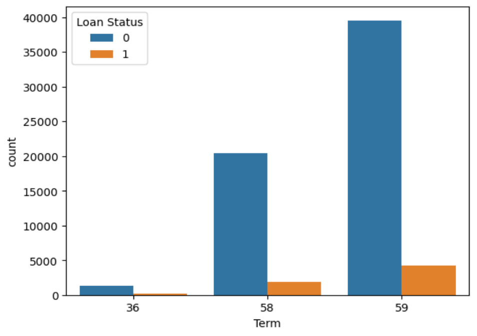


Figure-11: The distribution of 'Loan Status' across different values of Terms

2. **Delinquency - two years**
   The count plot visually represents the distribution of 'Loan Status' across different values of 'Delinquency - two years.' The majority have the loans status 0, which indicates that there are no delinquencies over the two-year period. For example, for Term 59 months there are over 40000 with no delinquencies compared to the 4000 delinquencies which is very less as shown in Figure**-12**.

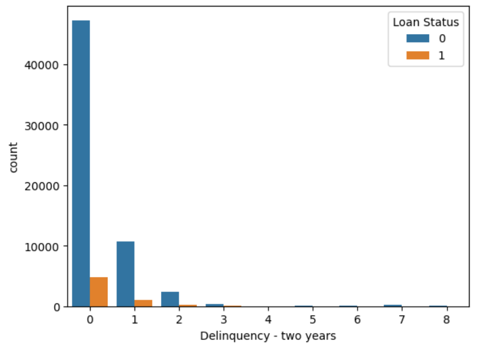


Figure-12: The distribution of Loan Status across different values of Delinquency - two years.

3. **Inquiries - six months.**
   The count plot in F**igure-13** visually represents the distribution of 'Loan Status' across different values of Inquiries - six months. Most instances

have 'Inquiries - six months' values of 0 which indicates that there are no reported credit inquiries over the six-month period. Further the count decreases as the number of reported inquiries increases.

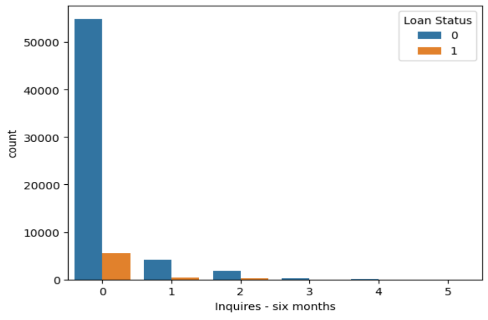


Figure-13: The distribution of 'Loan Status' across different values of Inquiries - six months

4. **Public Record**
The most frequent value with 62871 instances suggests that most of the dataset has no reported public records. We can tell that a considerable proportion of applicants have a clean public record history.

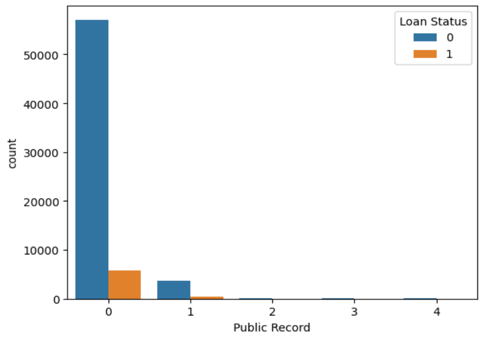


Figure-14: The distribution of 'Loan Status' across public record

5. **Collection 12 months Medical**
We can observe that the majority of instances in the dataset have no reported medical collections within the last 12 months. Loan status 1 has 1437 instances, which represents instances where there are reported medical collections within the last 12 months.

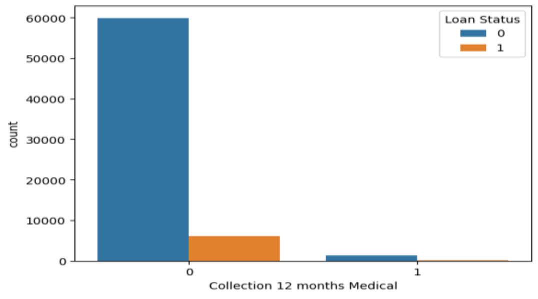


Figure-15: The distribution of 'Loan Status' across Collection 12 months Medical

Histogram:

Each histogram in **Figure-16** represents the distribution of a specific numerical feature. The x-axis of each histogram represents the range of values for the corresponding numerical feature. The y-axis represents the frequency of instances falling within each range. The frequency of instances for certain columns like 'Total Received Interest', ‘Recoveries’, and ‘Total collection amount’ are almost negligible. Most of the columns are skewed to the left side except for Loan Amount, Revolving utilities, and Last week pay which are evenly distributed.

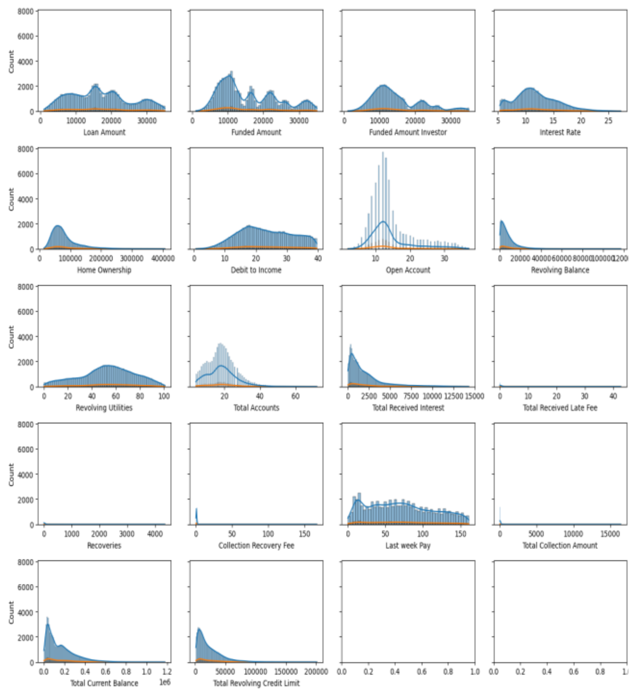


Figure-16: Histogram representation of the distribution of a specific numerical feature

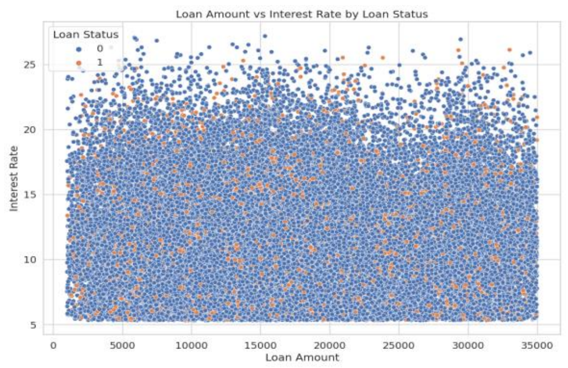


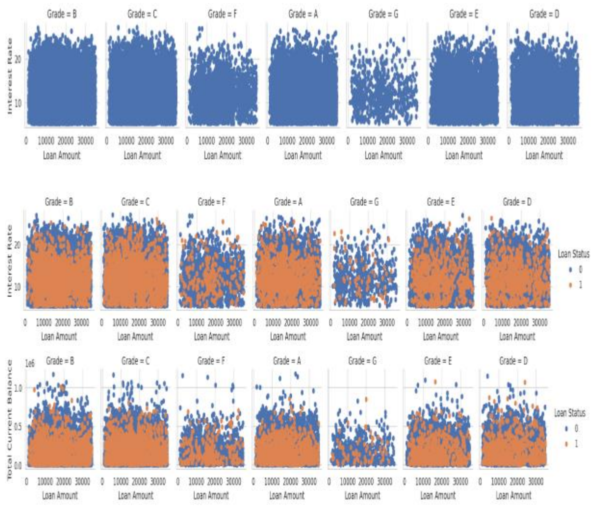
Figure-17: Scatter plot of loan amount vs Interest rate

The scatter plot in **Figure-17** is used to visualize the relationship between two numerical features, "Loan Amount" and "Interest Rate,". This color distinction helps us observe if there are patterns or trends between "Loan Amount" and "Interest Rate" based on the loan status. We can thus see that most of the dominant color data points are of loan status 0 compared to loan status 1.

Multivariate Analysis:

Multivariate analysis involves analyzing the relationship between multiple attributes. It shows us how one variable interacts with another and also the dependencies on each other. It helps us in understanding the more complex relations and dependencies unlike the univariate analysis. The most common techniques in multivariate are principal component analysis, regression analysis and other methods.

As we can observe in **Figure-18**, the multivariate analysis of a few of the columns. We try to visualize the relationship between 'Loan Amount' and 'Interest Rate' across different grades. The datapoints in the scatter plot represent the distribution of loan status based on the amount and the interest. We further analyze by other columns like 'Total Current Balance' relationship with loan amount and loan status across different grades as displayed in the graph. Other columns like Employment duration also illustrated the distribution of loans based on their loan amount, home ownership and loan status.



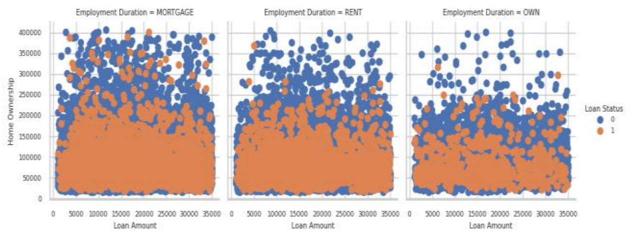


Figure-18: Scatter plot representing multivariate analysis between various attributes.

Correlation:

The correlation matrix did not identify any strong connections between the data and if anything, showed a strong amount of independence between the distinct categories.

7 Implementation

Hypothesis

When looking at the data, there are essentially two groups, one group represents customers that do not become delinquent, and the other group are the customers that do become delinquent. We came up with the following hypothesis and assumption, that the data for both groups are the same, when looking across all the distinct categories. However, if we could find any significant deviation from that assumption, with any of the categories, then we could say with confidence that there are potential predictive capabilities with some of the categories to help determine future delinquencies.

- Null hypotheses - all of the data, across each category, is the same for loan status of 1 or 0.
- Alternative hypothesis - one or more of the categories data does support significant differences between the loan status of 1 and 0.

Results of the Hypothesis:

- The #'s in bold and highlighted in yellow represent the significant values
- We reject the null and accept the alternative since there is a strong indication from a few of the categories that there is a difference in the date from the following groups:

➢ 6-month inquiries - inquiries usually mean that there are concerns and it looks like those concerns are turning into reality.

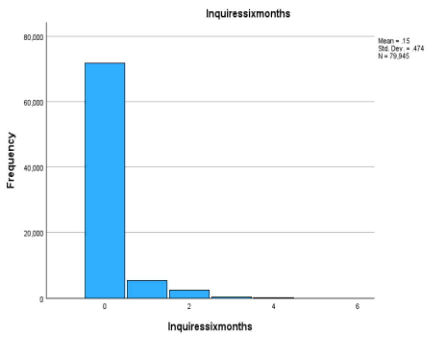


Figure-19: Six months Inquiries count plot.

➢ Total revolving credit limit - Revolving credit is not a positive sign and usually implies that customers are rotating through their credit accounts to pay other accounts, which can come crashing down.

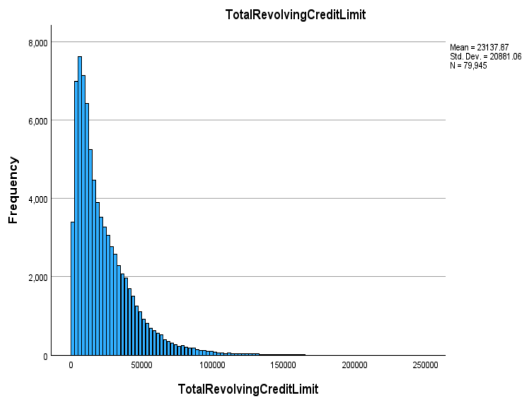


Figure-20: Total Revolving Credit limit count plot

- ➢ Revolving utilities - actual usage of the revolving accounts

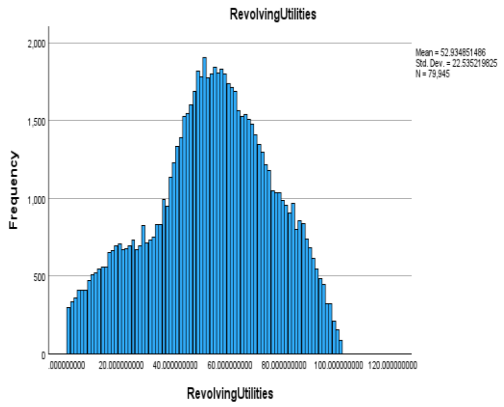


Figure-21: Revolving Utilities count plot.

- ➢ Open accounts - the overall number of accounts that people have open is another sign of people rotating between accounts to pay outstanding balances.

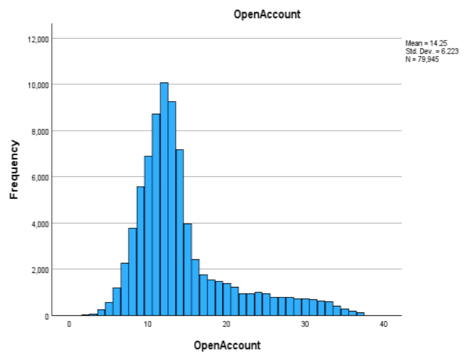


Figure-22: Open account limit count plot

- ➢ 12-month collections - if someone has been in collections, then that is another sign, that it could happen again

- ● Inquiressixmonths
- ➢ Equal variances assumed .322
  .571 -.239 79943 .406
  .811 -.001
- ➢ Equal variances not assumed
  -.237 30769.972 .406
  .812 -.001
- ● TotalRevolvingCreditLimit
- ➢ Equal variances assumed .424
  .515 -.602 79943 .274
  .547 -104.938
- ➢ Equal variances not assumed
  -.606 31350.790 .272
  .545 -104.938
- ● RevolvingUtilities
- ➢ Equal variances assumed .077
  .781 -1.703 79943 .044
  .089 -.320484115821
- ➢ Equal variances not assumed
  -1.704 31071.587 .044
  .088 -.320484115821
- ● OpenAccount
- ➢ Equal variances assumed .146
  .703 2.924 79943 .002
  .003 .152

- Equal variances not assumed
  2.927 31086.875 .002
  .003 .152
- Collection12monthsMedical
- Equal variances assumed .323
  .570 .284 79943 .388
  .776 .000
- Equal variances not assumed
  .285 31240.222 .388
  .776 .000

Model Implementation

The model building involves steps to create a predictive model that is, capable of classifying the loan status depending on the various input features. The process of step by step of building the model is given below:

Firstly, we initialize a data frame 'train_df' which contains several attributes which can help us predict the loan status. It involves 35 columns and 65,463 rows. The categorical columns are encoded into numeric columns. The columns which do not provide much information, and which have no relevance are removed.

For categorical columns we use one-hot encoding to convert them into numeric columns. The process of one-hot encoding involves creating new binary columns which indicate the presence of a particular category alone. It will create a new individual column for every unique category present in the original categorical attribute. This automatically expands the number of columns in the dataset. Therefore, the current dataset shape is 67463 rows with increased columns up to 43.

After converting the categorical columns, the next step involves dividing the data into train and test sets with input feature ‘X’ and target feature ‘y’.

Iterations in Feature Selection and Model Training:

The selection of the features (to be dropped) is based on eigen values from SPSS tool which were shown least importance. We tried removing a few columns in each iteration from the data-frame and then checked the model's performance. This approach is used to identify the most important and crucial features that will help predict the outcome and drop irrelevant columns in the dataset to improve the performance.

Process of feature selection involves:

Column Removal Approach:

Columns for removal were selected as per recommendations from SPSS analysis. Multiple iterations were performed by excluding differences such as 'ID', 'Payment Plan', 'Accounts Delinquent', 'Batch Enrolled', 'Sub Grade', 'Loan Title' are dropped to assess the impact on model performance. In our model we are using several classification models as listed below.

Iteration 1:
Models:

- Decision Tree
- Random Forest
- K Neighbors Classifier
- Gaussian NB
- Logistic regression
- Support Vector Machine
- XGB Classifier.

These models are trained on the training data and later we analyze the performance on both training and testing data [5]. Performance metrics like accuracy and ROC score are calculated. The ROC-AUC score measures the ability of a model to distinguish between the classes.

Observations of iteration - 1:

Decision Tree and KNN Models:
Despite eliminating the columns, the performance of the models, Decision Tree and K-Nearest Neighbors (KNN) remained consistent. Both the models showed high accuracy on the training set but slightly lower accuracy on the test set. This indicates that both the models are overfitting.

Gaussian NB and XGB Classifier:
Gaussian Naive Bayes (NB) and XGB Classifier models also maintained similar performance to the above models even after eliminating the columns. The models' accuracy and ROC-AUC scores remained similar across training and testing datasets, without any significant improvement.

Random Forest:
This model displays high training accuracy but better performance on the test set compared to the Decision Tree model, implying less overfitting [4].

ROC-AUC values of approximately 50% for all models indicate that the models perform no better than randomness. The outcomes above imply that the models are not properly differentiating between classes.

The models are not learning the patterns in the data efficiently, and removing individual columns did not increase prediction or reduce overfitting. Thus, we perform further iterations of feature selection and model training.

Logistic Regression:
We have applied the logistic regression model in phase-2 of our project. The model attained a training accuracy of 51 % and testing accuracy of 50% and the ROC-Score is 50.8%. The training and testing losses are also huge with 17.6% and 17.7% loss respectively. These results suggest that the model is performing poorly.

To improve the model, we have performed hyperparameter tuning on this model. Firstly, we tried to use Lasso regularization (L1) by defining 'C' values ranging from 0.1 to 1.0 in steps of 0.3. The accuracy recorded was 53% with F1 score of 52.8%, ROC-AUC score of 52.8%. The training and testing losses of 16.9% and 17.0% loss respectively are still huge. Even after adjusting 'C', the model gave better results than previous results without hyperparameter tuning but overall, the results still indicate that the model is performing very poorly.

We further changed the parameters to see if the model improves, but still the model didn't perform well. The accuracy and ROC-AUC score observed is still less which 53% and 52.7%. The training and testing losses are still very high. In conclusion, logistic regression with or without hyperparameter tuning didn’t perform well on the dataset. Since the data is complex, logistic regression is not fitting the data.

Support Vector Machine:

This model also displayed poo performance similar to logistic regression. It obtained an accuracy of just 47% and ROC-AUC score of 50.9%. Even after hyperparameter tuning and adjusting ‘C’ value ranging from 0.1 to 1, it didn’t help improve the performance of the model. Overall, SVM also showed poor performance on the data. By far, XG Boost classifier has produced better results compared to all the other models.

Iteration-2:
Column Removal:
In this iteration we are removing different columns such as ‘Public Record’,

‘Delinquency - two years’ and ‘Collection 12 months Medical'.

Overfitting continues to exist in both Decision Tree and XGB Classifier, especially demonstrated by significant differences in training and testing accuracies. Random Forest outperforms Decision Tree and XGB Classifier on the test set, showing more generalist model behavior. We can conclude that the removed columns were not the cause of overfitting of the model.

Iteration-3:
Column Removal: In this iteration we are further dropped more columns such as 'Debit to Income', 'Total Accounts'

Similar to the previous iterations, the models did not show any substantial changes in the model performance even though we removed additional columns.

Model comparison over all iterations:
Throughout the iterations, we saw that there is no significant difference in the performance of the model after eliminating the columns that did not contribute significantly to the model. Every iteration showed only minimum change in accuracy and the ROC-AUC scores with not much improvement in the models' performance.

The Decision Tree models showed the maximum training accuracy of 100% in all the three iterations but had lower test accuracy and ROC-AUC scores. Other models like the Random Forest also maintained higher training accuracy (around 99.99%) but varied slightly in test accuracy. K Neighbors Classifier consistently showed around 90% accuracy in both training and testing but did not exhibit significant improvement in the performance even after eliminating a few other columns. Gaussian NB indicated lower accuracy. However, the XGB Classifier had consistent higher accuracy on both training and testing sets across all iterations, although it has not shown any significant improvement in the ROC-AUC score.

Regardless of the removal of specific columns or features in each iteration, there was minimal influence on the model's predictive capacity. However, the above attempts have not drastically improved the performance of the model or generalized to new data. But the XGB Classifier model displayed the most promising results throughout the iterations, consistently outperforming other models with fewer variations. It indicates that to further improve the models’ predictions, it is necessary to make other additional improvements like resampling, hyperparameter tuning, or trying out various feature engineering techniques like scaling.

Resampling:
We are performing the resampling technique to address the problem of class imbalance; Resampling is used in cases especially where one class outnumbers another class and leads to poor or biased predictions. Using this we can adjust the class distribution equally amongst all the classes to make sure that they are balanced, and the predictions are not biased towards a class.

We have two types of resampling techniques:

Under sampling: In this method we reduce the number of instances in the majority class to match the number of instances in the minority class. This technique will remove valuable information from the majority class and might lead to the loss of information.

Over sampling: On the other hand, this technique involves increasing the instance in the minority to match the majority class instances. To balance we need to replicate the already existing samples. There is no loss of information using this technique.

As our data is not balanced, we are using the over sampling technique method 'RandomOverSampler()' to increase the number of instances in the minority class. We can confirm by checking the mean, the mean before the resampling is 0.09 and after applying the resampling method, the mean is 0.5. Therefore, resampling will help us balance the distribution of the classes in the data.

Observations of the model's performance after resampling:

The models showed improvement in the testing accuracy and ROC scores after resampling. Resampling significantly improved the models, decision tree and random forest accuracy and ROC scores. But we can observe significant difference in the training losses and testing losses which can lead to overfitting.

Whereas K Neighbors classifier and XGB Classifier did not show substantial improvement in the performance with 84% ROC score but there is less difference in the training loss and testing loss which indicates stable performance. GaussianNB did not perform well in any of the iterations or resampled models.

Since XGB Classifier is showing consistent improvement in overall performance, we will now try to do hyperparameter tuning to further optimize the model and decrease the difference between training loss and testing loss.

Hyperparameter Tuning:

Hyperparameters regulate the process of learning, influencing the training process and the generalization of a model towards original input. The performance of a model can be substantially affected through changing these hyperparameters. By setting the ideal parameters we can thus improve the accuracy of the model. Hyperparameters can also prevent the problem of overfitting or underfitting. The model can be made less vulnerable to noise or small variations in the training data by fine-tuning the hyperparameters. Predictions based on irregular variations are less possible.

There are specific hyperparameters for every model that can be modified. For instance, the gradient boosting technique XGBoost contains several parameters, like learning rate, maximum depth, and number of trees. The model's performance can be significantly improved by tuning these hyperparameters.

I. **First set of Hyperparameters tuning: {'max_depth': 7, 'n_estimators': 250}:**

Firstly, we set the hyperparameters of max_depth as 7 and n_estimators as 250. A higher max_depth indicates a deeper number of trees to capture more complex relations whereas n_estimators signify the number of boosting rounds in the model. This will help improve the performance as is evident in the output where the accuracy increased to 97% and f1 score is 96.8%. There is also improvement in the ROC AUC score which is 96.7%. However, the testing loss is greater than the training loss indicating risk of overfitting.

II. **Second Set of Hyperparameters: {'colsample_bytree': 0.8, 'max_depth': 8, 'n_estimators': 250, 'subsample': 0.8, 'learning_rate': 0.1}:**

In the second set of tuning the parameters we have increased the max_depth to 8 and added the colsample_bytree parameter which is set to 0.8 additionally adding learning rate of 0.1 and setting subsample to 0.8. The performance has lowered compared to previous tuning with accuracy of 96% and f1 score of 096.2 %. However, the testing loss has reduced to 1.38 which indicates that the issue of overfitting can be mitigated.

On comparing both the sets of hyperparameter tuning we can observe differences in the performance, where the first set showed better predictive metrics and the second set aimed at mitigating the problem of overfitting which is important for generalization of unseen data.

Testing on Original Dataset with Best Model:
Testing the tuned model on the original dataset showed consistent performance. It achieved an accuracy of 98%and ROC AUC score of 98.8% which indicates that the model correctly predicted the outcome. Additionally, the f1 score of 0.914 represents a balanced mean of precision and recall considering both the false positives and false negatives. These values indicate that the model performs exceptionally well in the original dataset. Thus, hyperparameter tuning and training on resampled data provides more robust predictions on testing data.

Scaling on best model with resampling and tuned hyperparameter:
Scaling makes sure that all the features have similar scale. It is an optimization technique where algorithms like XGBoost which is an ensemble algorithm that uses gradient boosting, and these types of algorithms often use gradient descent for optimization. Thus, scaling the input features can help the algorithm to converge much faster. Hence, improving the performance of the model.

In order to further increase the performance of the XGBClaassifier after resampling we also perform scaling on certain numeric columns like 'Loan Amount', 'Funded Amount', 'Funded Amount Investor', 'Home Ownership', 'Total Received Interest', 'Recoveries', 'Total Collection Amount', 'Total Current Balance', and 'Total Revolving Credit Limit' using a StandardScaler from the sklearn.preprocessing module. In the code, the numerical features listed above were scaled using a ‘StandardScaler’ method. This step will ensure that the input features are scaled uniformly.

After scaling, we split the dataset into the training and testing sets. We are applying the Random Oversampling technique once again. The best performing XGBoost model from the previous steps (with tuned hyperparameters) was fitted to the resampled data for classification.

The accuracy of the model after scaling and resampling is 97% which is lesser than the model on which scaling was not applied. The F1 score 0.967 is also slightly lower than the non-scale model. This indicates that there is a decrease in the balance between precision and recall. Additionally, ROC-AUC score of 0.966 also decreased slightly compared to the previous non -scaled model. The training and testing losses also increased marginally, indicating an increase in error and decrease in the model’s performance.

We can conclude that scaling didn’t have a substantial impact on the performance of the model. The difference in the metrics of the non-scaled and scaled models didn’t have any significant difference or improvement.

8 Preliminary Results

Building the model involves several steps which include pre-processing the data, feature selection, model training and hyperparameter tuning. Overview of the results obtained by the model.

Firstly, we have performed the data pre-processing on the original dataset. We have converted all the categorical columns to numeric columns using the one-hot encoding. Later, irrelevant columns are removed from the dataset based on the SPSS principal component analysis (PCA). However, the PCA did not provide any consolidation recommendations for the features.

The next step involves model training through several iterations. In the first iteration we used several classification models such as Decision Tree, Random Forest, K Neighbors Classifier, GaussianNB and XGB Classifier that are trained on the dataset.

The results of the first iteration are listed below:

```
training accuracy for Decision Tree is 1.0
testing accuracy for Decision Tree is
0.8195068926330352
roc_score for Decision Tree is Decision Tree is
0.5039002681423373
training accuracy for Random Forest is
0.9999364729798408
testing accuracy for Random Forest is
0.9081476357527546
roc_score for Random Forest is Random Forest is
0.5
training accuracy for K Neighbors Classifier is
0.9087751990513299
testing accuracy for K Neighbors Classifier is
0.9022184890557834
roc_score for K Neighbors Classifier is K
Neighbors Classifier is 0.4998784403844252
training accuracy for Gaussian NB is
0.900643740470947
testing accuracy for Gaussian NB is
0.9018726221651268
roc_score for Gaussian NB is Gaussian NB is
0.5011385659623712
training accuracy for XGB Classifier is
0.9243816703371167
testing accuracy for XGB Classifier is
0.9071594446365927
roc_score for XGB Classifier is XGB Classifier is
0.499455930359086
```

Based on the correlation analysis, we eliminated certain columns in the iterations. However, even with the removal of columns, the performance of the models remained the same. Decision Tree and K Neighbors Classifier results showed overfitting, while Random Forest outperformed on the testing set. The results of the second and third iteration after eliminating more irrelevant columns are given below.

Iteration-2 Output:

```
training accuracy for Decision Tree is 1.0
testing accuracy for Decision Tree is
0.8207421315282376
roc_score for Decision Tree is Decision Tree is
0.5007122219874375
training accuracy for Random Forest is
0.9999364729798408
testing accuracy for Random Forest is
0.9081476357527546
roc_score for Random Forest is Random Forest is
0.5
training accuracy for K Neighbors Classifier is
0.9087751990513299
testing accuracy for K Neighbors Classifier is
0.9022184890557834
roc_score for K Neighbors Classifier is K
Neighbors Classifier is 0.4998784403844252
training accuracy for Gaussian NB is
0.900643740470947
testing accuracy for Gaussian NB is
0.9018726221651268
roc_score for Gaussian NB is Gaussian NB is
0.5011385659623712
training accuracy for XGB Classifier is
0.9243181433169575
testing accuracy for XGB Classifier is
0.9073570828598251
roc_score for XGB Classifier is XGB Classifier is
0.5000482609380241
```

Iteration-3 Results:

```
training accuracy for Decision Tree is 1.0
testing accuracy for Decision Tree is
0.8110084490340432
roc_score for Decision Tree is Decision Tree is
0.4919685194691472
training accuracy for Random Forest is 1.0
testing accuracy for Random Forest is
0.9081476357527546
roc_score for Random Forest is Random Forest is
0.5
training accuracy for K Neighbors Classifier is
0.9087751990513299
testing accuracy for K Neighbors Classifier is
0.9022184890557834
roc_score for K Neighbors Classifier is K
Neighbors Classifier is 0.4998784403844252
training accuracy for Gaussian NB is
0.900643740470947
testing accuracy for Gaussian NB is
0.9019220317209349
roc_score for Gaussian NB is Gaussian NB is
0.5011657694444168
training accuracy for XGB Classifier is
0.9230264272403862
testing accuracy for XGB Classifier is
0.9072088541924008
```

```
roc_score for XGB Classifier is XGB Classifier is
0.499966650491887
```

As further columns were eliminated, Decision Tree and XGB Classifier continued to show signs of overfitting whereas Random Forest consistently outperformed on test data.

Logistic Regression Results:

training accuracy is 0.5104538560261347
testing accuracy is 0.5086296074481407
roc_score for is 0.5086754950450166
train_log_loss is 17.64503153137284
test_log_loss is 17.710784114813645

I. First set of hyperparameters for Logistic Regression {'C': 0.7, 'multi_class': 'auto', 'penalty': 'l1', 'solver': 'liblinear'}:

F1 Score: 0.5288330887548236
roc_auc_score: 0.5280107799737783
Train_loss: 16.943586964192264
Test_Loss: 17.01216481898277

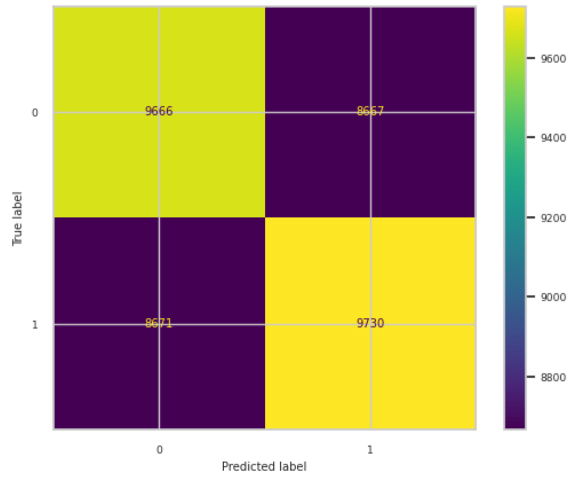


Figure-22: Confusion Matrix of first set of Hyperparameter tuning.

II. Second set of hyperparameters for Logistic Regression {'C': 0.7, 'multi_class': 'auto', 'penalty': 'l2', 'solver': 'newton-cg'}:

F1 Score: 0.5282557286145315
roc_auc_score: 0.527548445409532
Train_loss: 16.961249237466596
Test_Loss: 17.028845335877605

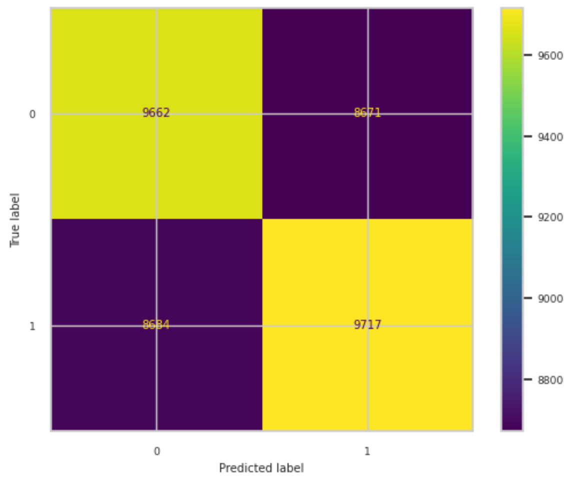


Figure-23: Confusion Matrix of second set of Hyperparameter tuning.

Support Vector Machine:

F1 Score: 0.4702872768972919
roc_auc_score: 0.509721471545319

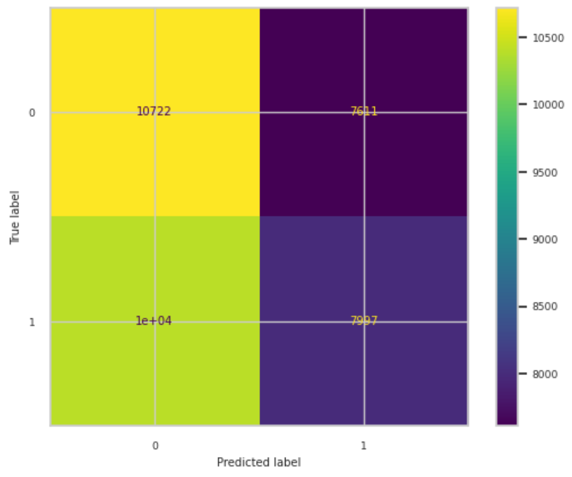


Figure-24: Confusion Matrix of SVM Model.

SVM Hyperparameter tuning:

F1 Score: 0.04766860949208993
roc_auc_score: 0.50270843515215559

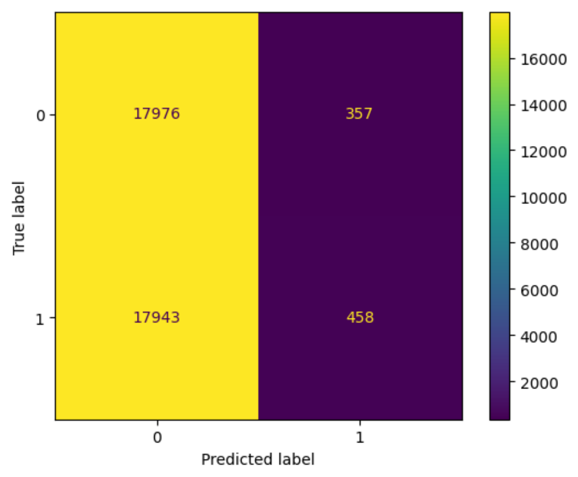


Figure-25: Confusion Matrix of best set of hyperparameters in SVM Model.

On the dataset, the Support Vector Machine (SVM) and Logistic Regression models performed less than ideal. In spite of adjusting hyperparameters both the models gave poor accuracy results. On the other hand, the XGBoost classifier obtained better results than these models, indicating that it is more appropriate given the complexity of the dataset.

The XGB Classifier showed consistent performance throughout all the iterations, suggesting a model that fits the best. We further perform the resampling on the XGB Classifier where it maintains stable performance even after resampling unlike other models as shown below.

Resampling Output:

```
training accuracy for Decision Tree is 1.0
testing accuracy for Decision Tree is
0.9333042957478086
roc_score for Decision Tree is Decision Tree is
0.9331813087917782
train_log_loss for Decision Tree is
2.2204460492503136e-16
test_log_loss for Decision Tree is
2.4039568466090544
training accuracy for Random Forest is 1.0
testing accuracy for Random Forest is
0.9995372134806991
roc_score for Random Forest is Random Forest is
0.9995380685832291
train_log_loss for Random Forest is
2.2204460492503136e-16
test_log_loss for Random Forest is
0.01668051689483856
training accuracy for K Neighbors Classifier is
0.8815074087037685
testing accuracy for K Neighbors Classifier is
0.826863396308597
roc_score for K Neighbors Classifier is K
Neighbors Classifier is 0.8265844296802947
train_log_loss for K Neighbors Classifier is
4.270905889859687
test_log_loss for K Neighbors Classifier is
6.240475732421873
training accuracy for Gaussian NB is
0.5092871310232179
testing accuracy for Gaussian NB is
0.5056623291773289
roc_score for Gaussian NB is Gaussian NB is
0.5063134821471017
train_log_loss for Gaussian NB is
17.68708456297839
test_log_loss for Gaussian NB is
17.817735664315848
training accuracy for XGBClassifier is
0.8944113872360284
testing accuracy for XGBClassifier is
0.8365002450046278
roc_score for XGBClassifier is XGBClassifier is
0.8364228851371808
train_log_loss for XGBClassifier is
3.8057993603023004
test_log_loss for XGBClassifier is
5.893128498258769
```

Hyperparameter Tuning Results:

Hyperparameter tuning was performed on the best fit XGBClassifier model to further optimize it. The first set of hyperparameters, including {'max_depth': 7, 'n_estimators': 250}, improves accuracy to 97%, F1 score to 97%, and ROC AUC score to 97.04%.

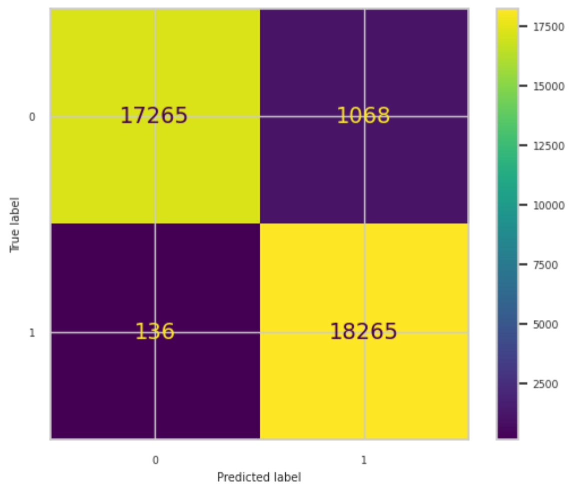


Figure-26: Confusion Matrix of first set of Hyperparameter tuning.

Hyperparameter Set-1 Results:

F1 Score: 0.9712815889263081
roc_auc_score: 0.9704718677866708

Train_loss: 0.07948022973449027
Test_Loss: 1.0626470468888194

The second set of hyperparameters, {'colsample_bytree': 0.8, 'max_depth': 8, 'n_estimators': 250, 'subsample': 0.8, 'learning_rate': 0.1}, aims at mitigating overfitting but results in accuracy (96%) , slightly lower F1 score (96.6%, and ROC AUC score to 96.5%.

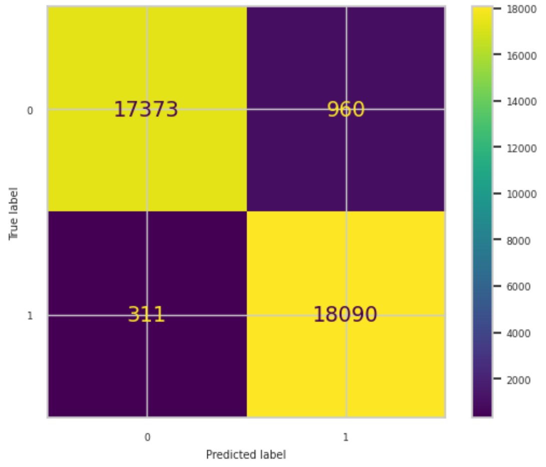


Figure-27: Confusion Matrix of second set of Hyperparameter tuning.

Hyperparameter Set-2 Results:

F1 Score: 0.96606232143333396
roc_auc_score: 0.9653670780931906
Train_loss: 0.3095103126168504
Test_Loss: 1.247113939608208

From the observations of high ROC-AUC Score, we can conclude that the first set of hyperparameter tuning produced better results compared to second set of hyperparameter. Therefore, we are proceeding to use the first set of hyperparameters for the XGBoost Classifier model.

Performance of Trained XGBoost Model with Optimized Hyperparameters on Original Dataset:

F1 Score: 0.9146853146853147
roc_auc_score: 0.9885198941154953

Consistent performance was observed when the tuned model was tested using the original dataset. With a ROC AUC score of 98.8% and an accuracy of 98%, the model successfully predicted the result.

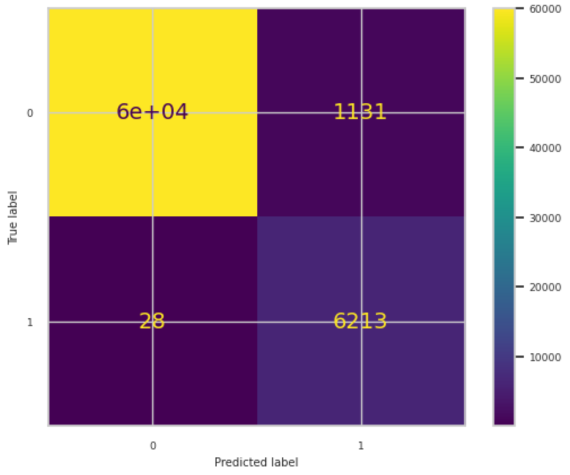


Figure-28: Confusion Matrix Trained XGBoost Model with Optimized Hyperparameters

Scaling Results:

F1 Score: 0.9677915330169923
roc_auc_score: 0.9668778495430678
Train_loss: 0.11648689754737453
Test_Loss: 1.192166354542858

Comparing the results of with scaling and the performance of the model without scaling we can draw conclusion that scaling didn't improve the model's performance.

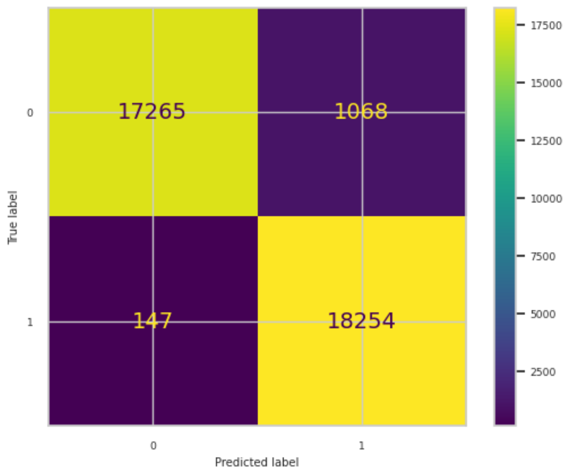


Figure-29: Confusion Matrix output after scaling.

Conclusion:

In our project, to analyze the dataset, we used several machine learning models such as: Decision Tree, Random Forest, K Neighbors Classifier, Gaussian NB, XGBClassifier, Logistic Regression, and SVM. Among all the models XGBClassifier performed the best. Notably, the XGBoost model had the best ROC-AUC score of 0.9705 especially after performing the hyperparameter tuning. This score is substantially higher than the pre-tuning ROC-AUC of 0.8380, hyperparameter optimization is clearly effective from the results. The training and testing losses for this model are also less, which indicates the performance of the model. Contrary to expectations, scaling did not lead to any significant gain in performance.

9 Project Management

Description:
We have done all the preliminary analysis and discovery of the data, created several classification models based on our hypothesis and found results using the data that can predict potential delinquencies. In the second phase of our project, we performed hyperparameter tuning to improve the results for XGBoost model. Additionally, tried scaling to see if it's helpful to improve the performance of the model. We have also tested two additional models which are logistic regression and SVM.

Responsibility & Contributions (Phase-1)

Phase	Description	Action item	Member	%
1	Research and Exploration	Find relevant dataset	Richard	5
		Research papers	Richard	5
		Tutorials	Tejasvi	5
2		Finalize Approach	Ashwini	5
	Thinking exercise for Design part	Finalize Hypotheses	Richard	5
3	Implementation - Python	EDA	Tejasvi/Ashwini	20
		Model Training & Validation	Ashwini	10
		Result Comparison	Ashwini	5
	Implementation - SPSS	EDA	Richard	5
		Model	Richard	5
		Hypotheses Results	Richard	5
4	Documentation	Report - First Draft	Tejasvi	15
5	Implementation	Test accuracy for logistic regression and SVM	Tejasvi/ Richard	20
		Hyperparameter tuning	Ashwini	20
		Scaling	Ashwini	20

Responsibility (Phase-2):

Phase	Description	Action Item	Member	%
3	Implementation	Test accuracy for logistic regression and SVM	Tejasvi/ Richard	20
		Hyperparameter tuning	Ashwini	20
		Scaling	Ashwini	20
4	Documentation	Summarization	Richard	20
	Documentation	Report - Final	Tejasvi	20

Issues/Concerns:
PCA in SPSS not producing better outcomes (zero correlation)
For more details about tasks, please refer specific sections (added only action items in contribution table)

GitHub Project Repository Link:

https://github.com/SharmaAshwini/CSCE5310_Project

References

[1] Loan Prediction Dataset
[2] Reference Notebook - 1: https://www.kaggle.com/code/inagib21/predicting-loan-default

[3] Reference Notebook - 2: https://www.kaggle.com/code/kartikanagawadi/loandefault
[4] Reference Notebook - 3: Random-forest-loan prediction
[5] Xu Zhu, Qingyong Chu, Xinchang Song, Ping Hu, Lu Peng, Explainable prediction of loan default based on machine learning models, Data Science and Management, Volume 6, Issue 3, 2023, Pages 123-133, ISSN 2666-7649, https://doi.org/10.1016/j.dsm.2023.04.003
[6] Reference Notebook - 4: hyperparameter-grid-search
[7] Reference Notebook - 5: xgboost-hyperparameter-tuning-scenarios-by-non-exhaustive-grid-search

Note: Please refer iPYNB for high quality plots/charts