Predictive Model for Loan Application Approval: A Comprehensive Analysis Using Machine Learning Techniques

Prepared for:

Delina Ivanova

Prepared by:

Team 5

Jinlun Zhang 220668810

Runze Liang (Alan) 220292181

Xinyuan Liang (Diana) 216211922

Chuhong He (Mike) 220862769

Olivia Zijun Wei 219775097

PengBo Gao (Harry) 220307831

Schulich School of Business

MBAN 6110 T

August 5, 2023

Introduction

In this project, we selected a dataset that contains historical loan application data. The data includes various characteristics of a loan application and the applicant such as the income, employment status, credit history, and loan amount. Most importantly, it indicates whether the loan application was approved or not, which is encapsulated in a binary categorical variable, which can be a potential candidate for the target variable of a loan application success prediction task.

The reason for our choice of this dataset is that it is directly relevant to our problem statement. Therefore, we need data that captures these characteristics and their corresponding loan approval status. Moreover, this dataset is not too complex because most of the features are numeric and the number of records is not large, so it does not require a lot of computational resources. Also, we believe that the selected characteristics are expected to have predictive power in determining loan approval. For example, an applicant's income and credit history may be important factors in loan approval decisions.

Problem Statement

The problem statement for this project is to determine whether a machine learning model can be trained to reliably predict the success of a loan application based on certain characteristics of the application and of the corresponding applicant. Resolving this problem can bring efficiency, speed, and objectivity to the traditionally manual and often time-consuming loan approval process. Financial institutions can use this predictive model to evaluate loan applications in a more streamlined and fair manner.

Hypothesis Statement & Objective

Based on the problem statement, our hypothesis and objective are that we can build a predictive model that accurately determines the loan approval status using the application's and the applicant's characteristics. Achieving this objective will automate the loan approval process, saving time and resources for both applicants and financial institutions. Moreover, we expect that machine learning models such as Support Vector Machine (SVM), XGBoost, and Logistic Regression will be able to individually accomplish this task. Furthermore, we hypothesize that an ensemble of these models will provide more accurate predictions than a single model.

Benefits

The benefits of this model are twofold. For loan applicants, it allows them to know immediately if their loan application is likely to be approved or not, saving them time and unnecessary effort. For financial institutions, the model streamlines the loan approval process, increasing efficiency, reducing workload, and minimizing human error and bias, resulting in more consistent and accurate decisions.

Exploratory Data Analysis

Outlier detection and removal

In this dataset, an unusual observation was detected in the feature ‘residential_assets_value’. Some records had a negative value for this feature, which is not plausible in a real-world context because asset values cannot be negative. This anomaly was likely due to human error during data collection. To maintain the integrity of the dataset and ensure robust model performance, records with negative ‘residential_assets_value’ were removed from the dataset. It's worth noting that only 20 out of 4241 records (0.5% of the total dataset) were affected by this problem and were dropped.

Addressing Formatting Inconsistencies

In the early stages of the analysis, it was identified that certain column names and categorical values in our dataset contained unnecessary leading or trailing white spaces. Such formatting issue can potentially lead to unrecognized variables during the analysis and modeling phases, disrupting the smooth progress of the project.

Visualizing Distributions and Outliers

As depicted in Figure.1-1, the histograms and boxplots were generated for each column in the dataset to visually examine the distribution and identify potential outliers. First, we examined the ‘no_of_dependents’ feature, which indicates the number of dependents for each loan applicant. We found that the histogram of ‘no_of_dependents’ shows a distribution with no clear pattern and no significant skewness. This uniform distribution suggests that the number of dependents of loan applicants is evenly distributed across the different categories, which can provide more insights to the model training.

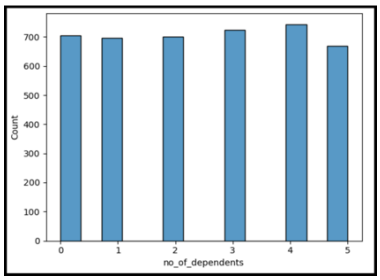

Figure.1-1

The ‘education’ feature comprises two categories: 'Graduate' and 'Not Graduate'. As illustrated by the histogram in Figure.1-2, this feature reveals a nearly balanced/uniform distribution. Out of 4241 entries, 2127 applicants are Graduates, and 2114 applicants are not, indicating that our dataset has a nearly equal representation of graduate and non-graduate loan applicants.

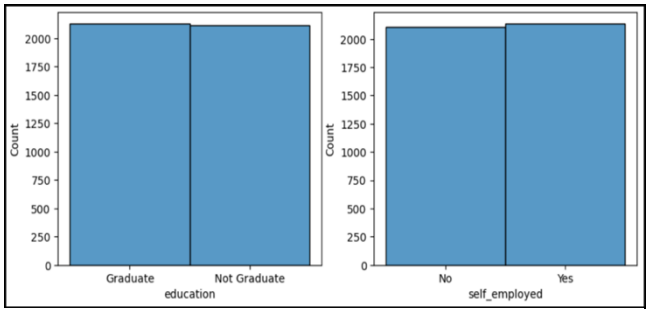

Figure.1-2

The ‘self_employed’ feature categorizes applicants based on their employment status into 'Yes'(self-employed) and 'No'(not self-employed). Like ‘education’, the ‘self_employed’ feature also exhibits a nearly balanced / uniform distribution with 2135 self-employed and 2106 non-self-employed applicants.

The ‘income_annum’ feature captures the annual income of loan applicants. The histogram in Figure.1-3 shows that the income data is symmetrically distributed and not significantly skewed. In addition, the box plot shows that there are no outliers in the data.

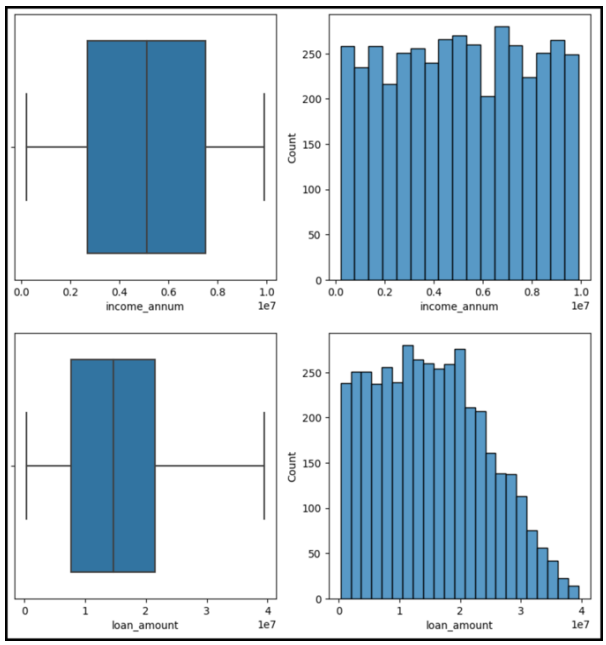

Figure.1-3

The ‘loan_amount’ feature signifies the loan amount requested by the applicants. In the histogram, the data is observed to be right-skewed, indicating that while most loan amounts are smaller and clustered towards the left of the histogram, there are some larger loan amounts that extend to the right. This right skewness suggests that some applicants are applying for significantly larger loans, potentially representing applicants with higher financial needs. In addition, the box plot shows that there are no outliers in the data feature.

The 'residential_assets_value' feature represents the value of the residential assets of loan applicants. The overall distribution is right skewed as can be seen in the histogram in Figure.1-4.
Also, in the boxplot, it seems that there are some outliers in the residential_assets_value column. However, as it is possible that some applicants have higher residential asset values and their data are relevant to the robust loan approval prediction, we decided to keep those outliers.

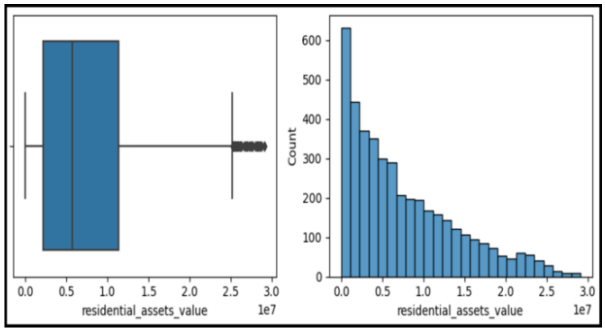

Figure.1-4

The 'commercial_assets_value' feature represents the value of the commercial assets owned by loan applicants. The plot in Figure.1-5 shows that the overall distribution is right skewed, and there are outliers, but we also decide to keep them due to the reasoning.
The 'luxury_assets_value' feature represents the value of the luxury assets owned by loan applicants. In the plot, it shows that the distribution of the data is right skewed and that there are no outliers.

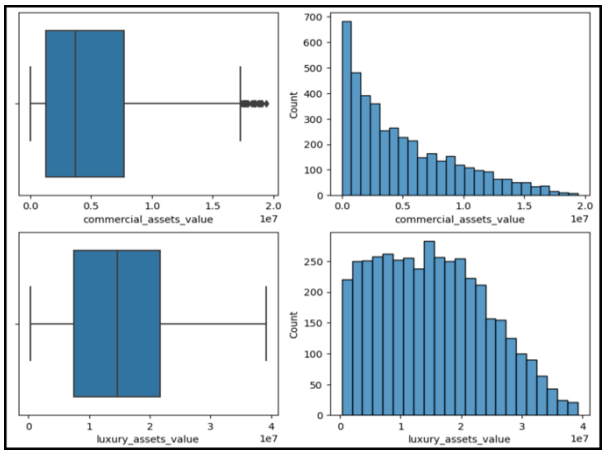

Figure.1-5

Target Variable Analysis and the Need for Resampling Techniques

The target variable in our dataset is 'loan_status', which is a categorical binary variable and represents whether a loan application was approved or rejected. However, this variable is imbalanced, as shown in the diagram, Figure.2-1. Out of all the records, over 2500 loan applications were approved, while around 1500 were rejected. This imbalance in the target variable is also reflected in the percentage of approved loans, which is approximately 62.25% of the total loan applications. The imbalance in our target variable has several implications for our analysis and model-building process:

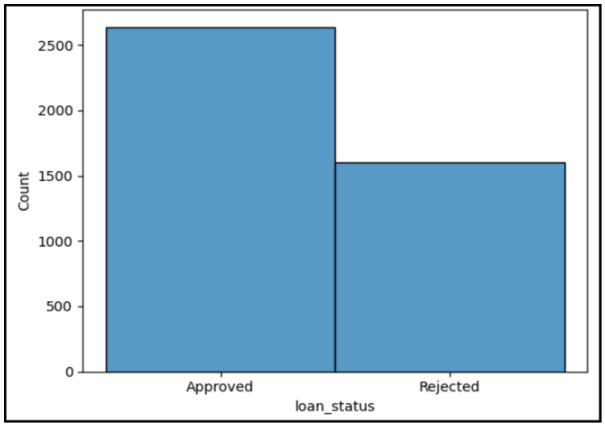

Figure.2-1

Evaluation Metrics: An imbalanced target variable suggests that we should not rely solely on the accuracy score as the evaluation metric for our models. Accuracy can be misleading when classes are imbalanced, as a model that predicts the majority class for all instances will have high accuracy but not learning any underlying patterns within the data. To overcome this, we should also consider other metrics like the F1 score or the Area Under the Receiver Operating Characteristics Curve (ROC-AUC score). These metrics can provide a more comprehensive evaluation of model performance across different thresholds and balances between precision and recall.

Resampling Techniques: The imbalance in our target variable suggests that we might need to implement resampling techniques to improve the performance of our models. Common resampling techniques include under-sampling, which reduces the number of instances from the over-represented class, and over-sampling, which increases the number of instances from the under-represented class. However, both these methods have their own disadvantages. For instance, over-sampling can lead to overfitting due to the repeated minority instances in the training set. Considering these limitations, we will implement Synthetic Minority Over-sampling Technique (SMOTE) in the following data preprocessing stage. SMOTE is a more advanced version of naive over-sampling. Unlike naive oversampling, which simply replicates minority instances, SMOTE synthesizes new minority instances.

The process of SMOTE involves the following steps:

- A random example from the minority class is first chosen.
- Then, k of the nearest neighbors of that example are found (typically k=5).
- A randomly selected neighbor is chosen and a synthetic example is created at a randomly selected point between the two examples in feature space.

This approach is effective because new synthetic examples from the minority class are created relatively close in feature space to existing actual examples from the minority class. This can help to improve the performance of our models by providing more balanced training data.

Checking for Missing Values and Dropping Irrelevant Columns

In any data analysis task, it's crucial to check for missing values, as they can lead to inaccurate analyses and conclusions. Fortunately, our dataset is clean, and there are no missing values in any of the columns. Additionally, we dropped the 'loan_id' column from our dataset because it does not provide valuable information for predicting the 'loan_status'.

Pearson Correlation Analysis

A graphical representation of correlation matrix is used to visualize the relationships between the numerical features. The heatmap in Figure.2-2 reveals several pairs of features with high correlation coefficients (above 0.7), indicating strong linear relationships, such as 'income_annum' & 'luxury_assets_value' (0.93), 'income_annum'& 'loan_amount' (0.93), etc.

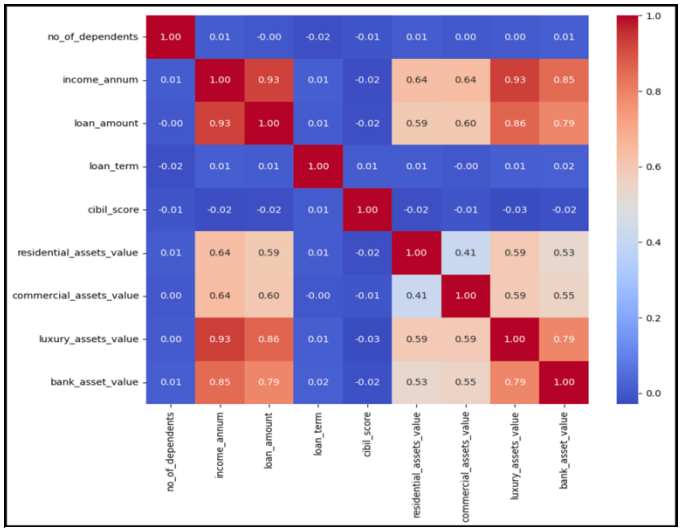

Figure.2-2

These high correlation values suggest a substantial degree of multicollinearity in our dataset. While multicollinearity doesn't affect the predictive power of certain model, it can distort the individual feature importance and result in unstable estimates of the linear model coefficients.

Visualization of Highly Correlated Features

To better understand the relationships between the highly correlated features and their impact on the 'loan_status', we created several scatter plots. Each scatter plot shows the relationship between the 'cibil_score' and one of the other highly correlated features, with the data points colored based on the 'loan_status'.

The scatter plots in Figure.2-3 indicate that 'cibil_score' is a strong differentiator for loan approval status. We can clearly see that for higher 'cibil_score', most of the loans were approved (orange), while for lower 'cibil_score', most loans were rejected (blue). This pattern is observed across all scatter plots involving 'cibil_score'.

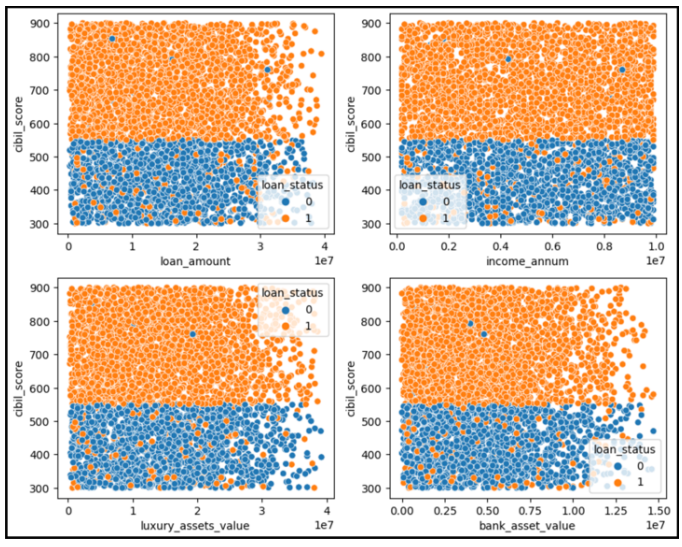

Figure.2-3

In addition, the scatter plots show that features such as 'income_annum', 'loan_amount', 'luxury_assets_value', and 'bank_asset_value' are linearly separable with 'cibil_score'. This means that we can find a threshold in the 'cibil_score' that can

effectively separate applications with different approval statuses. For instance, applicants with a 'cibil_score' below a certain threshold tend to have lower 'income_annum', 'loan_amount', 'luxury_assets_value', and 'bank_asset_value', and the applications are more likely to be rejected.

Given the strong differentiating power of 'cibil_score', one possible approach is to train a simple logistic regression model using 'cibil_score' as the sole training feature. This can serve as a baseline model, providing a benchmark for the minimal model performance we should expect.

Logistic Regression Baseline Model:

For our baseline model, we use a Logistic Regression model trained solely on the 'cibil_score' feature. Logistic regression is a suitable choice for this task since our target variable, 'loan_status', is a binary variable.

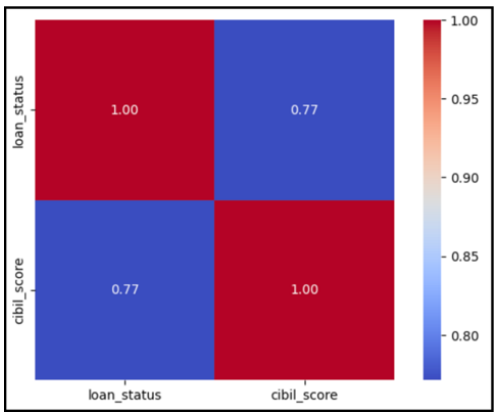

Figure.3-1

The results of the logistic regression model were surprisingly good with accuracy and F1 scores being 0.92 and 0.93 respectively, which is also unsurprising given our previous observation that 'cibil_score' is a strong differentiator between approved and rejected loan applications. This validates our initial assumption that 'cibil_score' could be a critical feature for our predictive models. However, this raised a concern that 'cibil_score' could essentially be a 'synonym' for our target variable, and hence we might need to exclude it to ensure the task remains meaningful.

To investigate this further, we visualized the correlation between 'loan_status' and 'cibil_score' using a heatmap. The heatmap in Figure.3-1 confirmed our suspicion of a high correlation between the two features.

Random Forest Model and Feature Importance

Before deciding on excluding 'cibil_score', we chose to investigate how it would impact the performance of a more sophisticated model. We trained a random forest model without excluding 'cibil_score'. The Random Forest model, even when constrained to only 3 estimators with a maximum depth of 3 each, still achieved high accuracy and F1 scores of 0.95 and 0.97 respectively, which implied that the given task might be easily solvable with simple models. However, the high performance of the models also raised concerns about potential overfitting due to the heavy reliance on the 'cibil_score' feature.

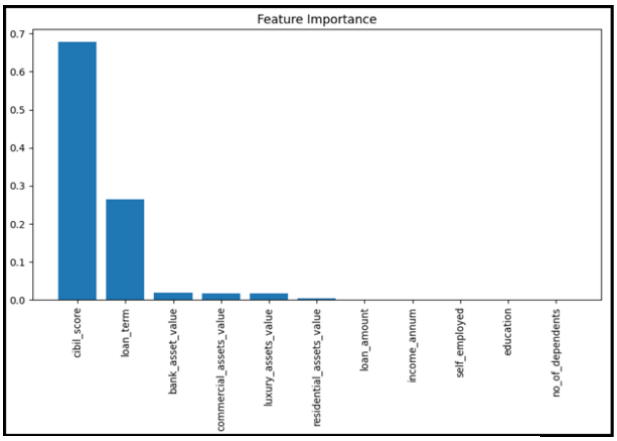

Figure.3-2

As shown in the Figure.3-2, we realize that when we included all the features in the Random Forest model, the feature 'cibil_score' had an importance of 0.677, much higher than any other feature. This implies the need to drop this feature as it will dominate the model training, leading to potential overfitting with the model relying heavily on this single feature to make predictions and failing to learn from other features.

When excluding the 'cibil_score' feature from the Random Forest model, as shown in Figure.3-3, we can observe a more balanced distribution of feature importance amon the training features. The 'loan_term' feature now has the highest importance (0.306), followed by 'loan_amount' (0.198), and 'income_annum' (0.187). Interestingly, 'self_employed' and 'education' features still have zero importance, suggesting that they may not contribute to the model's ability to learn and can potentially be dropped in future modeling to reduce model complexity.

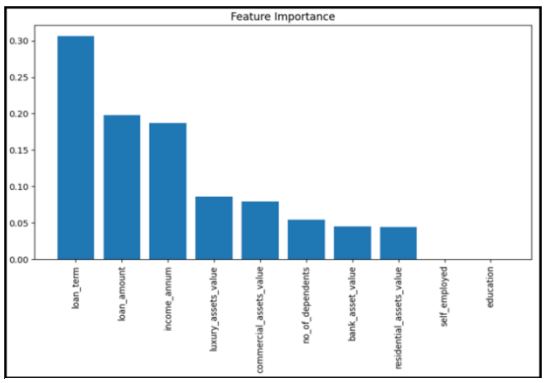

Figure.3-3

However, the random forest model's overall performance decreased, with an accuracy score being 0.63, suggesting that the model performed worse on the dataset without 'cibil_score'. Nevertheless, the precision remained as high as 0.99, indicating a low false positive rate but high false negative rate. In other words, this means that the model failed to identify most of the positive cases in the data (failed to identify as many records whose loan status is approved as possible), but the accuracy / reliability of each positive prediction is high. According to the f1 score (0.77), the general model performance is satisfactory and the feature importance it yields should be reliable as well.

In conclusion, removing 'cibil_score' led to a decrease in model performance but resulted in a more balanced distribution of feature importance.

Statistical Testing

Test based on Loan Status

In our analysis, we performed a t-test to identify features that are significantly different between loan approval and loan rejection.

The p-values for most of the variables considered, including 'Income_annum', 'Loan_amount', 'Luxury_assets_value', 'Bank_asset_value', and 'No_of_dependents', exceeded the standard significance threshold of 0.05. As a result, the null hypothesis could not be rejected for these features, indicating that there is no significant difference in these characteristics between individuals who were approved for a loan and those who were rejected.
Cibil_score: The p-value for this feature was 0.0, which is less than 0.05, indicating a significant difference in the CIBIL scores of the two groups. The average CIBIL score for

those approved for a loan was around 704, while for those rejected it was around 429. This suggests that the credit score plays a crucial role in loan approval.
Loan_term: The p-value for this feature was significantly less than 0.05, indicating a significant difference in loan terms between the two groups. The average loan term for those approved for a loan was around 10, while for those rejected it was around 11. This suggests that the loan term is also an important factor in loan approval.

Our analysis suggests that 'Cibil_score' and 'Loan_term' should be included in the training set as they are informative features (differentiators) for model training. The statistical t-test has shown that these two features have a significant impact on 'loan_status'. Furthermore, these findings could help the company identify potential high-risk or low-risk customers and tailor its services accordingly to manage risk and improve overall business performance based on these differences.

T-test on Education Status

We have used t-test to identify the features that are statistically different between the education status, Graduate and No Graduate.

Our t-test analysis indicates no statistically significant differences in the numeric variables (like "Income_annum" and "Loan_amount") between applicants who are graduates and who are non-graduates. Since all the p-values exceeded the standard significance threshold of 0.05, which prevented us from rejecting the null hypothesis for any of these variables, the educational status does not significantly influence these attributes of an application. In other words, none of these numeric characteristics of a loan application has a crucial association with education. Therefore, we can conclude that the educational status of the individuals does not play a crucial role in determining these numeric characteristics of an application.

T-test based on Self-Employment Status

We have used t-test to identify the features that are statistically different between the self-employed status, Yes and No.

Our t-test analysis of various numerical variables shows no statistically significant differences in these aspects between self-employed and non-self-employed individuals. All p-values exceeded the standard significance threshold of 0.05, which prevents us from rejecting the null hypothesis for these variables, implying that self-employment status does not have a significant impact on these attributes. In simpler terms, none of the given numerical characteristics demonstrate a strong association with an individual's self-employment status. Therefore, we can conclude that an individual's self-employment status does not play a significant role in determining these numerical aspects of the loan applicants' financial profiles.

Feature Engineering

Feature engineering is an essential step in the machine learning pipeline, such as

standardizing or normalizing raw data into a format suitable for building and optimizing machine learning models. By doing this step, the model's performance can be further enhanced, and it enables learning of complex patterns within the data.

Normalization

Normalization techniques are used. Normalization (i.e. MinMaxScaler) is a widely used data preprocessing technique that involves scaling numerical features to fit within a specific range (i.e. [0,1]) without altering the original feature distributions. This normalization helps to ensure that all features are on a comparable scale, which is beneficial for various machine learning algorithms, such as SVM.

Creation of New Features

loan_to_term_ratio: A new feature describing the amount of instalment the loan applicant needs to pay per month is generated. This feature is computed by dividing the loan amount of an application by the loan term.

Correlation of New Feature with Existing Ones

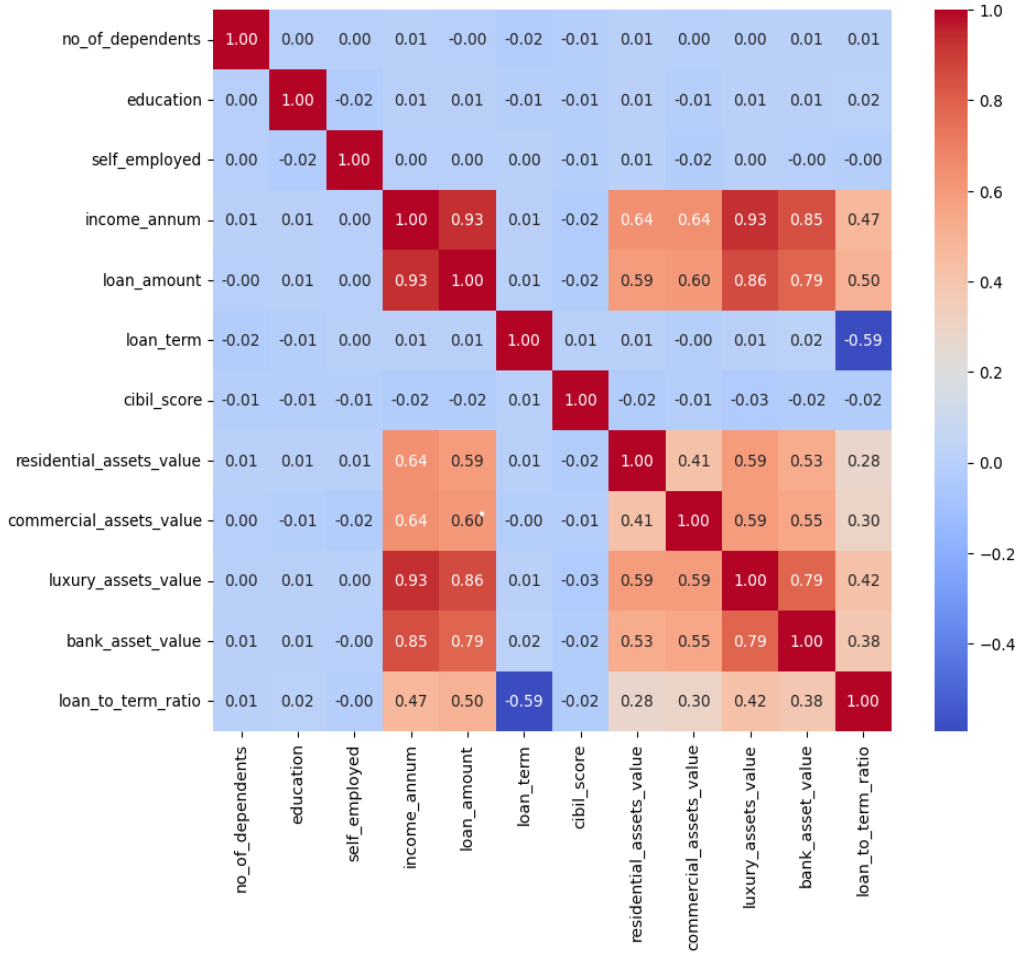

Figure.4-1

As indicated by the correlation matrix in Figure.4-1, the generated feature 'loan_to_term_ratio' is moderately correlated with many other training features like 'bank_asset_value' and 'luxury_assets_value', meaning that this feature is a good aggregation or representation of certain existing data feature values, and may yield good model

performance by itself alone.

Handling of skewed features

We have observed right-skewness within the distributions of certain features like 'residential_assets_value', 'commercial_assets_value', etc, as shown in Figure.4-2.

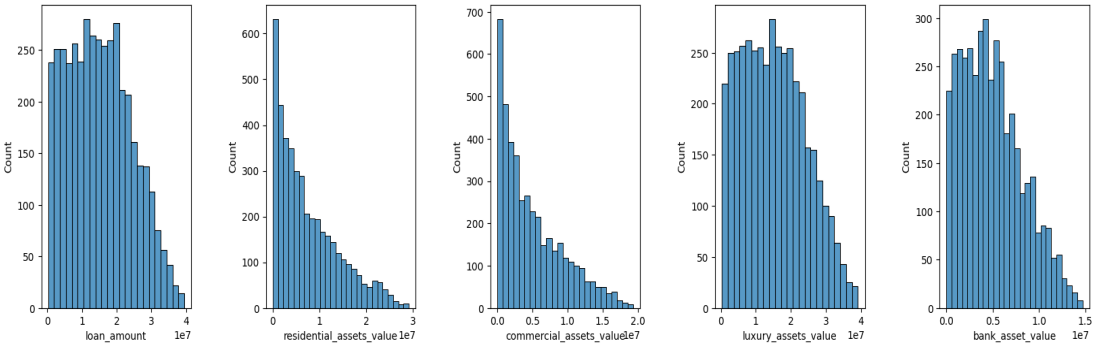

Figure.4-2

To reduce the impacts of potential outliers in those right-skewed features on the performances of the models that are sensitive to outliers such as SVM, we adopted a simple square root transformation on the skewed features to transform them to be normally distributed as shown in Figure.4-3.

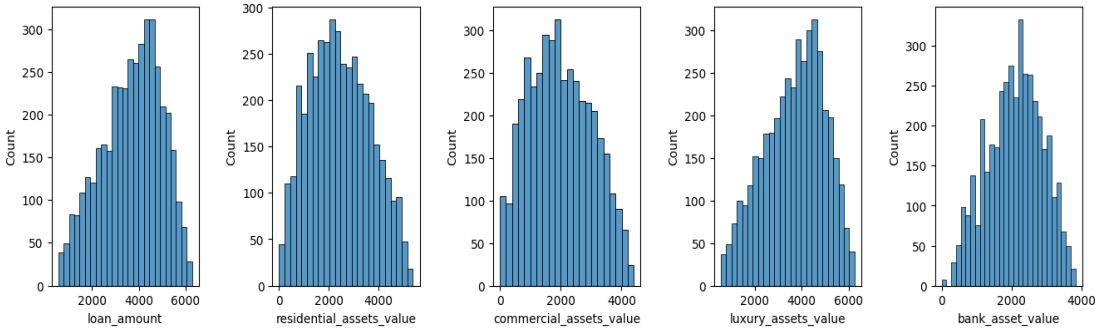

Figure.4-3

For the normalization process, we will employ a MinMaxScaler, but only on the training set. This decision is based on three main reasons. First, the MinMaxScaler will not alter the original distributions of feature values, which is essential for preserving the inherent patterns in the data. Second, since MinMaxScaler can restrict the feature values within a specified range, and the numerical features like ‘loan_amount’ and ‘bank_asset_value’ should always be positive to be interpretable, MinMaxScaler is considered to be a better normalization method than the StandardScaler that yields a normal distribution (with zero mean and unit variance) with possible negative standardized feature values, compromising certain interpretability and the suitability to the business problem on hand. Third, the MinMaxScaler should only be trained on the training set, because it should not have any knowledge of the testing set to ensure that the testing set truly simulates unseen data. Therefore, we will refrain from training the MinMaxScaler until we have completed the train-test split.

Feature Selection

All features post-engineering were retained for model development, except for 'education'

and 'self_employed', as they were demonstrated to be unimportant according to the EDA section and statistical testing. Future work might involve applying techniques such as recursive feature elimination or using feature importance from tree-based models to select the most impactful features.

Model Training & Evaluation:

Dropping Unnecessary Columns:

The 'cibil_score' column was deemed extraneous and was thus removed from the dataset to avoid any potential overfitting or irrelevant correlations. It's crucial in predictive modelling to only incorporate features that add substantive predictive value.

Handling Imbalanced Data using SMOTE:

As anticipated, our dataset displayed a significant imbalance in the 'loan_status' classes. An imbalanced dataset can skew the results, leading the model to be biased towards one class as it can predict the majority class more frequently due to its overrepresentation. To combat this, we used the aforementioned SMOTE method to generate synthetic samples in the minority class. The resampled resulting dataset now contains equal numbers of approved and rejected records as shown in Figure.5-1.

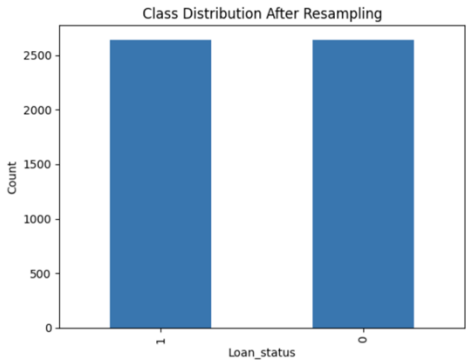

Figure.5-1

Train-Test Split:

The balanced dataset (from the previous SMOTE step) was divided into training and testing sets, with 80% of the records allocated for training and the remaining 20% for testing purposes.

Data Normalization:

To ensure the algorithms perform optimally, the aforementioned MinMaxScaler is trained on the training data to normalize it, while the same scaler is subsequently used on the testing data without training to maintain consistency and simulate performance on unseen data.

Model Training & Hyperparameter Tuning:

We have decided to train 3 models with random search CV on the hyper-parameter tuning: Support Vector Machines (SVM), XGBoost, and Logistic Regression. Upon training and optimization of these individual models, we further leveraged an ensemble method called Majority Voting to aggregate the predictions for enhanced performance. To be more specific, Majority Voting means that each classifier / model will vote for a class, and the final prediction of a record is determined by the class label with the most votes.

Model \ Testing Score	Accuracy	F1-score	Precision	Recall
Support Vector Machines	0.59	0.5309	0.464	0.6203
XGBoost Classifier	0.6496	0.6365	0.6136	0.6612
Logistic Regression	0.5294	0.5005	0.4716	0.5332
Ensemble Modeling using Majority Voting	0.589	0.5373	0.4773	0.6146

Table 1: Evaluation Metrics Scores of the Models on the Testing Set

Support Vector Machines (SVM)

The SVM model underwent training with an array of kernel methods, gamma values, and other parameters. The test set evaluation result of the best tuned estimator from the Random Search CV is shown on the above Table 1. It is noteworthy that the best estimator only yields a training F1 score of 0.4884, indicating underfitting in terms of balance between precision and recall. Moreover, the test evaluation result suggests that the SVM model has a higher recall (0.6203) compared to precision (0.4640), meaning that the model has a higher tendency to classify a loan as approved, even if it's not, resulting in a greater number of false positives.

XGBoost Classifier

This XGBoosting model was optimized over varying learning rates, estimators, and other parameters. The test set evaluation result of the best tuned estimator from the Random Search CV is shown on the above Table 1. It is noteworthy that the best estimator yield a training F1 score of 0.6349, revealing that the model achieves a better balance between precision and recall compared to the SVM model. The test set metrics proves the model's effectiveness, especially with a promising recall of 0.6612. A high recall suggests the model has a higher accuracy in identifying the positive class, which is loan approvals in this context. The optimal hyperparameters ('max_depth': 231) reveals a deep tree model, suggesting complex trees. However, since the training and testing F1 scores are similar in value, the XGBoost model does not show signs of overfitting, meaning the underlying patterns within the data are intricate and demand deep trees to capture.

Logistic Regression

This Logistic Regression model was optimized over various solvers and regularization strengths. The test set evaluation result of the best tuned estimator from the Random Search CV is shown on the above Table 1. It is noteworthy that the best estimator yields a training F1 score of 0.5299, leading the logistic regression model sits between SVM and XGBoost in terms of balance between precision and recall. Test set evaluation result highlights a balanced performance but with a relatively lower accuracy. This might hint that the linear boundary assumption might not be the most apt for this data.

Ensemble Modeling using Majority Voting

The 3 tuned models were assimilated into an ensemble model through a hard majority voting classifier. Note that the Voting Classifier will not retrain the supplied individual models but fitting the voting classifier itself. The test set evaluation result of the Majority Voting

classifier is shown on the above Table 1. It is noteworthy that the ensemble's accuracy of 0.5890 is not the highest comparing to the individual models, suggesting that there might be discordance among the models' predictions, which is reasonable based the differences between the testing F1 score of XGBoost and that of the other 2 models. The ensemble's testing recall of 0.6146 indicates that the combined model still maintains a good rate of correctly identifying approved loans (lower false negative rate). Interestingly, the testing ensemble's precision is slightly better comparing to the individual SVM or logistic regression model, which suggests that the ensemble could reduce some of the false positives generated by these two models.

Conclusion:

Both SVM and Logistic Regression underperformed in the binary classification task, with accuracy metrics hovering around 50%, suggesting that their predictive powers are not much better than random guessing. These results lag behind the benchmark set by the Random Forest model, which achieved an F1 score of 0.77. Various factors could contribute to this, including limited hyper-parameter tuning, the normalization procedure adopted, or potential overfitting of the Random Forest model.

The ensemble method using majority voting did not elevate performance. Instead, it was overshadowed by the standalone XGBoost model, hinting that SVM and Logistic Regression might have introduced more noises than complementary information to the majority voting ensembling. This underscores the possibility that while both SVM and Logistic Regression naturally benefit from feature normalization, they might not be inherently well-suited for this specific dataset or task.

In addition, SVM and Logistic Regression aim to define hyperplanes or boundaries for classification, XGBoost's ensemble of decision trees captures intricate patterns, enabling it to perform differently and potentially better. However, blindly combining XGBoost's predictions with the others, as seen in the majority voting method, can be counterproductive due to potentially conflicting insights, leading the majority voting classifier to perform no better than the individual models.

Furthermore, it is evident that XGBoost's relative insensitivity to feature scaling gives it an edge over SVM and Logistic Regression in this scenario, implying the undesirability of data normalization in this case and the reason why XGBoost outperforms the others (as the feature scaling can be compromising the model performances in this case). Moving forward, fine-tuning the model weights in the hard majority voting classifer or exploring more sophisticated ensemble techniques like stacking might improve the ensemble predicition result. Hence, adjustments in preprocessing strategies and re-evaluation of normalization methods are recommended to bolster the model performance.

In conclusion, the problem statement is partially addressed, because a Random Forest model can be trained to reliably predict the success of a loan application based on characteristics of the application and of the corresponding applicant, including the key feature "cibil_score"

(credibility score). However, without including this key feature, a reliable predictive model that accurately determines the loan approval status using the application's and the applicant's characteristics cannot be obtained.

Reference

Brownlee, J. (2021, March 16). SMOTE for imbalanced classification with Python. Machine Learning Mastery. https://machinelearningmastery.com/smote-oversampling-for-imbalanced-classification/