*University of Essex*
**Department of Mathematical Sciences**

MA981: DISSERTATION

# Addressing Class Imbalance in Loan Default Prediction Using Machine Learning Algorithms

**Harish Vishwas Bagul**
**2310317**

Supervisor: Dr Tao Gao

September 18, 2024
Colchester

# Abstract

For any financial organisation to effectively manage risks, it is essential to predict loan defaults with accuracy. But any machine learning model may face significant difficulties when the default rate is high relative to the number of loaners—a situation known as the class imbalance problem. Several machine learning models, including Random Forest, XGBoost, Logistic Regression, and Linear Discriminant Analysis (LDA), were taken into account in this work with regards to modelling loan defaults. The issue of class imbalance was addressed with SMOTE. Model performances with and without SMOTE have been compared.

Top results show that Random Forest and XGBoost performed flawlessly or almost flawlessly, returning accuracy and AUC-ROC scores of 1.0 without the need for SMOTE. But there was overfitting. Although the recall for a minority class using SMOTE was enhanced by logistic regression and LDA, the accuracy as a whole did not considerably improve. In order to determine significant predictive elements, such as interest rate spread, upfront costs, and credit type, feature interpretation was carried out using SHAP or SHAPLEY Additive explanations. The investigation has demonstrated that while simpler models like Logistic Regression require oversampling strategies like SMOTE to improve management of the minority class, ensemble models like Random Forest and XGBoost are naturally good at handling class imbalance.

This study compares the effectiveness of different machine learning models in scenarios with imbalanced datasets,adding to the body of knowledge on the prediction of loan defaults. Based to these results,ensemble approaches can eliminate class imbalance and perform effectively without the need for additional strategies.However, simpler models can enhance the recall of the minority class by utilising SMOTE.More in-depth research could include cost-sensitive learning and more advanced oversampling techniques like ADASYN.This would lower the possibility of overfitting while additionally improving performance.

# Contents

**Abstract** **2**

**1 Introduction** **10**

1.1 Aims and Objectives . . . . . 12

**2 Literature Review** **13**

**3 Data Preprocessing and Exploratory Analysis** **18**

3.1 Raw Data Handling . . . . . 18

3.1.1 Data Acquisition . . . . . 18

3.1.2 Initial Inspection: . . . . . 18

3.1.3 Understanding Schema: . . . . . 19

3.1.4 Preliminary Assessment: . . . . . 19

3.2 Data Cleaning . . . . . 20

3.2.1 Correct Data Types: . . . . . 20

3.2.2 Remove Duplicates: . . . . . 20

3.2.3 Fix Inconsistencies: . . . . . 21

3.2.4 Normalize Data . . . . . 22

3.2.5 Review Data Types and Missing Values . . . . . 22

3.2.6 Missing/Null Value Treatment . . . . . 23

3.2.7 Decide on a Strategy . . . . . 23

3.3 Exploratory Data Analysis (EDA) . . . . . 25
3.3.1 Understand Data Distribution . . . . . 25
3.3.2 Descriptive Statistics: . . . . . 25
3.3.3 Distribution of numerical features . . . . . 26
3.3.4 Distribution of Loan Amounts . . . . . 27
3.3.5 Distribution of Interest Rates . . . . . 28
3.3.6 Box Plot of Loan Amount by Loan Purpose . . . . . 29
3.3.7 Box Plot of Rate of Interest by Region . . . . . 30
3.3.8 Heatmap of Correlation Between Features . . . . . 31
3.3.9 Count Plot of Gender . . . . . 32
3.3.10 Pie Chart Graphs to Check Frequency of Region and Loan Purpose . . 34
3.3.11 Distribution of Loan Purpose . . . . . 35

**4 Methodology 36**
4.1 Feature Engineering . . . . . 36
4.1.1 Feature Selection . . . . . 36
4.1.2 Checking Relation Between Attributes . . . . . 37
4.1.3 Drop Irrelevant Columns . . . . . 37
4.1.4 One-Hot Encoding . . . . . 38
4.2 Data Splitting . . . . . 38
4.2.1 Define Features and Target Variables . . . . . 39
4.2.2 Perform Train-Test Split . . . . . 39
4.3 Data Scaling . . . . . 40
4.3.1 Apply Standard Scaling . . . . . 40
4.3.2 Fit a StandardScaler on Training Data and Transform Both Training and Test Data . . . . . 40

4.4 Addressing Class Imbalance . . . . . . . . . . . . . . . . . . . . . . . . 41
4.4.1 Apply SMOTE . . . . . . . . . . . . . . . . . . . . . . . . . . 42
4.5 Model Development . . . . . . . . . . . . . . . . . . . . . . . . . . . 43
4.5.1 Random Forest: . . . . . . . . . . . . . . . . . . . . . . . . . . 43
4.5.2 Logistic Regression: . . . . . . . . . . . . . . . . . . . . . . . 44
4.5.3 XGBoost: . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 46
4.5.4 Linear Discriminant Analysis (LDA): . . . . . . . . . . . . . . . . 48
4.6 Model Predictions . . . . . . . . . . . . . . . . . . . . . . . . . . . . 49
4.7 Model Evaluation: . . . . . . . . . . . . . . . . . . . . . . . . . . . . 50
4.7.1 Accuracy: . . . . . . . . . . . . . . . . . . . . . . . . . . . . 51
4.7.2 Confusion Matrix: . . . . . . . . . . . . . . . . . . . . . . . . 52
4.7.3 AUC-ROC Curve: . . . . . . . . . . . . . . . . . . . . . . . . . 52
4.7.4 Comparison of Models with and Without SMOTE: . . . . . . . . . . 52
4.8 Model Interpretation: . . . . . . . . . . . . . . . . . . . . . . . . . . 52
4.8.1 SHAP Analysis: . . . . . . . . . . . . . . . . . . . . . . . . . 52

**5 Results And Discussion 57**
5.1 Model-Specific Performance (With and Without SMOTE) . . . . . . . . . . 57
5.1.1 Random Forest(With Smote) . . . . . . . . . . . . . . . . . . . . 57
5.1.2 Random Forest(Without Smote) . . . . . . . . . . . . . . . . . . . 58
5.1.3 Logistic Regression(with SMOTE) . . . . . . . . . . . . . . . . . . 59
5.1.4 Logistic Regression(without SMOTE) . . . . . . . . . . . . . . . . 59
5.1.5 XGBoost (with SMOTE) . . . . . . . . . . . . . . . . . . . . . . 60
5.1.6 XGBoost (without SMOTE) . . . . . . . . . . . . . . . . . . . . . 60
5.1.7 Linear Discriminant Analysis(With SMOTE) . . . . . . . . . . . . . 61
5.1.8 Linear Discriminant Analysis(Without SMOTE) . . . . . . . . . . . . 61

5.2 ROC Curve and AUC-ROC Comparison for Multiple Models(With SMOTE . 62
5.3 ROC Curve and AUC-ROC Comparison for Multiple Models (Without SMOTE) 63
5.4 Model Interpretation(SHAP Analysis) . . . . . . . . . . . . . . . . . . . . . . 64
5.4.1 Random Forest(with SMOTE) . . . . . . . . . . . . . . . . . . . . . . 64
5.4.2 Random Forest(without SMOTE) . . . . . . . . . . . . . . . . . . . . 65
5.4.3 Logistic Regression (with SMOTE) . . . . . . . . . . . . . . . . . . . 66
5.4.4 Logistic Regression (without SMOTE) . . . . . . . . . . . . . . . . . 67
5.4.5 XGBoost with SMOTE . . . . . . . . . . . . . . . . . . . . . . . . . . 68
5.4.6 XGBoost without SMOTE . . . . . . . . . . . . . . . . . . . . . . . . 69
5.4.7 LDA with SMOTE . . . . . . . . . . . . . . . . . . . . . . . . . . . . 70
5.4.8 LDA without SMOTE . . . . . . . . . . . . . . . . . . . . . . . . . . 71
5.5 Impact of Overfitting and SMOTE on Model Performance . . . . . . . . . . . 72
5.5.1 Overfitting in Random Forest and XGBoost . . . . . . . . . . . . . . 72
5.5.2 No Overfitting in Logistic Regression and LDA . . . . . . . . . . . . 73
5.5.3 Impact of SMOTE on Performance . . . . . . . . . . . . . . . . . . . 73
5.5.4 Similar Accuracy with and Without SMOTE . . . . . . . . . . . . . . 73

**6 Conclusion 74**
6.1 Summary of Key Findings . . . . . . . . . . . . . . . . . . . . . . . . . . . 74
6.2 Contributions and Impact . . . . . . . . . . . . . . . . . . . . . . . . . . . 74
6.3 Limitations and Future Work . . . . . . . . . . . . . . . . . . . . . . . . . 75
6.4 Final Remarks . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 75

# List of Figures

3.1 Distribution of numerical features . . . . . . . . . . . . . . . . . . . . . . . . . . 27
3.2 Distribution of Loan Amount . . . . . . . . . . . . . . . . . . . . . . . . . . . . 28
3.3 Distribution of Interest Rates . . . . . . . . . . . . . . . . . . . . . . . . . . . . 29
3.4 Box Plot of Loan Amount by Loan Purpose . . . . . . . . . . . . . . . . . . . . 29
3.5 Box Plot of Rate of Interest by Region . . . . . . . . . . . . . . . . . . . . . . . 30
3.6 Bar Plot of Loan Types Count . . . . . . . . . . . . . . . . . . . . . . . . . . . . 31
3.7 Heatmap of Correlation Between Features . . . . . . . . . . . . . . . . . . . . . 32
3.8 Count Plot of Gender . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 33
3.9 Pie Chart Graphs to Check Frequency of Region and Loan Purpose . . . . . . 34
3.10 Distribution of Loan Purpose . . . . . . . . . . . . . . . . . . . . . . . . . . . . 35

5.1 ROC Curve and AUC-ROC Analysis(With SMOTE . . . . . . . . . . . . . . . . 62
5.2 ROC Curve and AUC-ROC Analysis(Without SMOTE) . . . . . . . . . . . . . 63
5.3 Random Forest(with SMOTE) . . . . . . . . . . . . . . . . . . . . . . . . . . . . 65
5.4 Random Forest(without SMOTE) . . . . . . . . . . . . . . . . . . . . . . . . . . 66
5.5 Logistic Regression (with SMOTE) . . . . . . . . . . . . . . . . . . . . . . . . . 67
5.6 Logistic Regression (without SMOTE) . . . . . . . . . . . . . . . . . . . . . . . 68
5.7 XGBoost with SMOTE . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 69
5.8 XGBoost without SMOTE . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 70
5.9 LDA with SMOTE . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 71

5.10 LDA without SMOTE . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 72

# List of Tables

5.1 Classification Report for Random Fores . . . . . . . . . . . . . . . . . . . . . . . 57
5.2 Confusion Matrix Table . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 57
5.3 Classification Report for Random Fores . . . . . . . . . . . . . . . . . . . . . . . 58
5.4 Confusion Matrix Table . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 58
5.5 Classification Report for Logistic Regression(with SMOTE) . . . . . . . . . . 59
5.6 Confusion Matrix Table . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 59
5.7 Classification Report for Logistic Regression(without SMOTE) . . . . . . . . . 59
5.8 Confusion Matrix Table . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 60
5.9 Classification Report for XGBoost (witho SMOTE) . . . . . . . . . . . . . . . . 60
5.10 Confusion Matrix Table . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 60
5.11 Classification Report for XGBoost (without SMOTE) . . . . . . . . . . . . . . . 60
5.12 Confusion Matrix Table . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 61
5.13 Classification Report for LDA (with SMOTE) . . . . . . . . . . . . . . . . . . . 61
5.14 Confusion Matrix Table . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 61
5.15 Classification Report for LDA (without SMOTE) . . . . . . . . . . . . . . . . . 61
5.16 Confusion Matrix Table . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 62

CHAPTER 1

# Introduction

Loan Default Prediction is a difficult problem with the data for training predictive model available from thousands of applications / loans received at banks, links are made up to actual defaults by Debtor or companies. In the majority of loan datasets, far fewer examples exist where a borrower defaults (ie they do not repay) than does not default. This imbalance problematic cause most of machine learning algorithms are design to increase the overall accuracy,thus they will more tending toward majority class (non_default cases ) and as this model tendency towards scikit learn models viz logistic regression.Hence these models would be good at performing against Non-Defaults but will do a significant number of mistakes i.e out Default cases it missed.The inability to accurately assess default risks contributes to models that fail to identify such clients, and therefore undermines efforts at risk management, and in the process opens organizations to higher loss outcomes. Mitigating class imbalance is therefore important in creating accurate, better models that can easily detect potential defaulters while at the same time capturing potential eligible borrowers. Consequently, this balance is important in order to have a proper check on the side of the new risk management practices within the given financial arena to be equally fair to those provided in the past.

Forecasting loan defaults is a challenging task because a typical class imbalance characteristic of the data sets taken for training of the predictive models. Loan datasets are usually imbalanced which means that there are significantly fewer records of loan defaults or in other words, borrowers who have failed to honor their loan obligations as compared to the records of borrowers who have repaid their loans as agreed. This makes the predictive models to be biased towards the non-default cases since conventional machine learning algorithms work under the overall accuracy optimization model. In this regard these models might work perfectly for classifying non defaults but fail when it comes to classifying defaults. The inability to accurately forecast defaults has consequences of developing models that miss potential high-risk clients, thus rendering overall risk management efforts rather ineffective and, consequently, expose the financial entity to more risk and potential losses. Credit scoring is fundamental to update if class imbalance is to be overcome in order to model handles potential defaulters effectively, while at the same time keeping track of eligible borrowers. This balance is needed to adjust optimal risk management practices across the financial industry sectors with equal opportunities for all.

New trends in machine learning and data science have given new promising approaches to solve the problem of dealing with class imbalance in loan default prediction. Tools such as synthetic oversampling, outlier identification, and ensemble learning techniques provide fresh approaches to boost up the predictive model accuracy on imbalanced data set [11].For instance, the SMOTE involves creating artificial examples of the minority class to balance up as this makes the model learn from characteristics of both classes and boost up on accurate predictions. Another model is cost sensitive learning where the cost function of the algorithm is modified to increase the cost of misclassification of the minority class which in turn increases its chance of getting the defaults right [10]. In the same way, integrating these techniques with other powerful machine learning algorithms like Random Forest, GBM and XGBoost helps in the immense improvement in the quality of the predictive models. Individual methods like the ones used in XGBoost composes multiple models together in order to arrive at a better result which also reduces variance or randomness [8]. This is particularly true with XGBoost which is known to be performing well thus efficient for even high dimensional data which is the case with most financial risk datasets. Therefore, applying these advanced methods can help to enhance the financial system's loan default prediction while catering for the class imbalance problem and other related needs.

The widespread existence of big financial data and the improvement in computational power have increased the spectrum-almost indefinitely-in which loan default prediction models can improve. Big data technologies facilitate the examination of large volumes of historical and immediate lending information that allows for a more comprehensive comprehension behind borrower performance and factors affecting risk [12]. Such approach based on data can allow better risk identification and drive more targeted credit strategies. A combination with external data sources (e.g., economic indicators, social media sentiment, and transactional information) enhances the predictive models offering a more rounded perspective of borrower risk profiles. Utilizing sophisticated analytics and machine learning algorithms will also result in the ability to create much-improved real-time monitoring systems that earlier identify distress credit & enable a proactive resolution.

Loan default prediction involves some form of ethical issues which cannot be ignored. The responsibility of making the predictive models less biased is therefore essential when it comes to lending as it can improve the perception clients have towards the entire process. Of course, when approaching the analysis of the credit-risk implication of digitization, it is crucial to consider that the introduction of algorithms may bring bias and subsequently lead to discriminations or discriminating treatment of specific loans' borrowers [2]. Concerns of fairness, accuracy, and fairness are best addressed when transparency and accountability procedures are incorporated into the building and testing of the model of loan default predictions and to also make sure that the systems are developed effectively and efficiently according to the ethical and legal standards.

## 1.1 Aims and Objectives

- To investigate and address the impact of class imbalance on the performance of loan default prediction models.
- To explore various techniques for handling class imbalance in loan datasets.
- To develop a robust and reliable predictive model for loan default prediction.
- To validate the developed model using real-world loan datasets.
- To provide actionable insights for financial institutions to improve their risk management practices.

CHAPTER 2

# Literature Review

Loan default prediction has emerged to be very important since many financial institutions seek to improve on their risk management strategies. A change in trend of predictive modeling in this area can be attributed to the increase in the size and complexity of problems related to finance and the need for a form of modeling to solve such problems.

The historical procedures used in loan default prediction analysis were mainly the conventional statistical techniques. Some of the earliest models that were employed for the purpose of classification included logistic regression as well as linear discriminant analysis. Credit rating analysis was performed using logistic regression with help of which Ohlson (1980) [19] modelled the probability of default considering several financial indicators. However, there is always a limitation with logistic regression, in that the model is limited in its ability to detect the subtle, non-linear relationships that often define financial data. The second kind of theory was the discriminant analysis suggested by Altman in 1968 [1] with his famous Z-score model which employed financial ratios for the purpose of bankruptcy prognosis. However, both these techniques face some problems in case of high-dimensional data present in most of the frequently used modern approaches to financial analysis.

The use of new generation methodologies such as machine learning to forecast loan defaults also brought in new improvements in the field. There are methods such as the decision trees, random forests and the neural networks that has been proved to be more efficient as compared to the conventional methods. Some of these models include decision trees in that they provide a more interpretable model given that data is segmented on the basis of feature values. Random forests as introduced by Breiman (2001) [5] builds on decision trees and compile the results of multiple trees to make a final prediction which eliminates the problem of overfitting. This method is particularly useful where one needs to navigate through sometimes a very large number of parameters.

Other Machine Learning methods like SVMs,developed by Cortes and Vapnik (1995) [9] have also contributed towards refining the prediction models. The SVM finds the best hyperplanes that creates different classes in a high dimensional space making it ideal for detecting imbalances in datasets. This proficiency has been complemented by neural networks which are capable of modeling the non-linear relationship and are a part of the bigger picture of works in the domain of predictive modeling. It is also important to capture complex interaction between features and deep learning architectures enable this aspect.

In recent research, ensemble methods have emerged as a storehouse for better predictive accuracy by disparate grouping of an array of models together. We explain how ensemble methods, of which Gradient Boosting Machines (GBM) and specifically XGBoost are prime examples gained a lot of success in the financial prediction opposed to individual learning models. Finally, XGBoost is by far the most popular implementation in industry due to its speed and scalability. Regularization techniques in XGBoost prevents overfitting so it will get more robust models.

Class imbalance is still a big issue in the process of loan defaults. The most popular method, such as SMOTE (Synthetic Minority Over-sampling Technique) proposed by Chawla et.v Motivated by this work, and an intuition from other works [7] synthetic examples of the minority class are generated to balance two-class datasets.This makes the SMOTE algorithm yield preferable results since the augmented training set produced involves both defaulted and non-default samples. Another technique of solving the problem of class imbalance is called cost-sensitive learning, which involves modifying the cost function so as to make a higher penalty for misclassifying the minority class. Circuit 2 is designed to help models target defaults, hence, solving the imbalance problem as well.

Future developments in big data and related technologies as well as developments in computational power have also refined the capabilities of the loan default prediction. Other external data including economic indicators, social media sentiment and transactional data incorporated in the models offers a much better picture of borrower risk profile. All these approaches make it easier to assess the risk levels and, in turn develop more appropriate ways of lending.

As the predictive modeling techniques emerge the class imbalance and use of more elaborate techniques in learning algorithms remain crucial. The contemporary studies in this field are aimed at creating more stable and accurate models to predict loan defaults in order to contribute to the financial soundness of lending organisations.

Pandas is an open-source data manipulation library, present in Python language, which has a versatile importance in the Data science field for handling the structured data very effectively. It gives data structures; the Series and DataFrame which make it easy to clean and transform data and also analyze. This library has easy to use API and it provides strong functionalities that are useful for data preparation steps.

The utilisation of pandas in data analysis is quite common and the role played by this tool cannot be overemphasised. McKinney (2022) [18] first took up pandas in his book ”Python for Data Analysis” and according to him, pandas are the tools that offer high performance data structures for data Analysis. Pandas have since on been used in the data science process providing features like data concatenation, reshaping and Boolean value operation. It is very useful in the scenarios of 6 processing and exploration of data and because it is possible to easily integrate with other libraries.

I use Matplotlib, an excellent plotting library for python that allows me to create simple static plots as well animated and interactive visualizations. Highly customisable for publication quality graphics/plot. This makes it a great tool for data visualization especially with the integration with pandas and numpy. In his paper Matplotlib: A 2D Graphics Environment written in Hunter (2007) [15], he described Matplibray as a full library that enables to generate visualizations in Python. With multiple plot types available, such as line plots, scatter plots and histograms; this library can be highly useful for purposes of exploratory data analysis and model diagnostics. Its flexibility and variety of customization has become more important than ever when it comes to telling a story about the data.

Seaborn is basically an extension of Matplotlib aimed at making statistical graphics more accessible, native, and compelling. It seamlessly works with the Pandas DataFrames and includes some extra features for distribution, bivariate and categorical data. Waskom et al. (2017) [**?**] introduced Seaborn in the paper ”Seaborn: tasted in a process called “Statistical Data Visualization,” it help to oversimplify the procedure of creating the necessary graphical picture. Such capabilities of seaborn include enhanced functions for analyzing distribution, correlation, and categorical data through function sns. Heatmap, sns. pairplot, and sns. Boxplot. Due to its capability to work by reducing the necessity of writing lots of code for statistical visualization, it is useful in determining data trends and correlation.

This paper aims at showcasing how scikit-learn, which is a powerful Python machine learning library, can be helpful in data mining as well as analysis. It also consists of a plethora of algorithms that enable classification, regression, clustering, and dimensionality reduction, among others, and also tools for the selection and assessment of the models. Pedregosa et al. (2011) [20] provided a comprehensive overview of Scikit-learn in their paper ”Scikit-learn: Python for Machine Learning. ” But the main advantage of the library is its stability, simplicity and its documentation is well-done. ” Scikit-learn provides a consistent interface to different learning techniques and metrics for evaluation so that the user can actually apply and use models. Having an ability to perform preprocessing, feature selecting and cross-validation, it became one of the most important tools in machine learning pipelines.

Imbalanced-learn is another one tool that can be used to deal with class imbalance in the datasets of machine learning. They offer various types of resampling techniques that include oversampling, undersampling, and the combined methods in order to overcome the problems that arise due to imbalanced data sets.

Many imbalanced-learn techniques have been demonstrated to be very effective in previous works. SMOTE was proposed by Chawla et al. (2002) [7] where the authors designed a method for oversampling of instances of the minority class. The imbalanced-learn library extend this framework with other methods including ADASYN and under-sampling techniques mentioned earlier. Similar to He et al. (2008) [13] and Bunkhumpornpat et al.(2009) [6], these techniques promote enhanced performances of the models especially in imbalanced datasets.

The key library for scientific computing in Python,Numpy providing support for multi-dimensional arrays and matrices, along with functions to operate on these. It is a cornerstone used by many scientific computing libraries in Python.

Van Der Walt et al. (2011) [21] explained the concept of Numpy in their paper ”The NumPy Array: A Structure for Efficient Numerical Computation”, which was enables efficient numerical array manipulations and operations. Array operations of Numpy are the most fundamental feature in data preprocessing, feature engineering and some mathematical computation conceptes while working on a machine learning project. It is a key part of the data analysis and machine learning ecosystem due to its interoperability with other libraries such as pandas,sci-kit-learn.

CHAPTER 3

# Data Preprocessing and Exploratory Analysis

## 3.1 Raw Data Handling

### 3.1.1 Data Acquisition

Raw data handling is the first step of data pre-processing where we acquire the dataset from Kaggle a popular platform for datasets among data scientists machine learning engineers. The data set is usually presented in a CSV (Comma Separated Value) file format that contains attributes concerning loan application inclusive of amount, credit score and the borrower. After importing the file from Kaggle, it goes to working environment for analysis, such as using the Pandas module in python [18] or Excel program. This first step is important because it transforms the data from their storage form to a form more usable for analysis in the form of a table. Correct import of the dataset and its availability is the most basic precondition for the further operations with the data.

### 3.1.2 Initial Inspection:

After the data set has been imported into the programming environment, there is a look at the data to determine among other things the data type or the nature of the data that is being worked on. This entails looking at a few initial rows and the initial columns of the dataset to establish the nature of the dataset that is, the names of the columns and the kind of data in every particular column. Worksheets are scanned first within this inspection, any hard coded values, or formatting inconsistencies come out quickly. For example, if a particular form has a column, which has to contain numerical data but instead, it contains both numbers and letters or symbols, then this may mean that there were some data input errors or even data entry discrepancies [20]. The primary evaluation can offer an initial understanding of the data set and highlight any issues that a can be seen with naked eye and guide the subsequent cleaning process.

### 3.1.3 Understanding Schema:

It is crucial to grasp various schema of the entire dataset and its part, which are going to be processed in a definite time step. Schema explains what each column contains and what type of data it contains, which is actually given in metadata/Americans dataset include loan_amount, credit_score, borrower_details all are in numeric format If not otherwise loan_purpose and gender are all categorical data. This is made possible through the reviewing the schema since it shows how each column is being interpreted clearly hence additional information is obtained well. This step is also relevant to define deviations of the dataset's structure and content compared to the specified schema.

### 3.1.4 Preliminary Assessment:

The preliminary assessment involves examining the dataset for any initial inconsistencies or errors. This includes checking for issues such as incorrect or inconsistent column names, mismatched data types, and erroneous or outlier values. For example, a column intended to contain numerical data might have text or special characters due to data entry mistakes or formatting issues. By identifying and addressing these problems early, the data can be cleaned and prepared more effectively for detailed analysis. This assessment also involves documenting any issues found, which helps in tracking the data cleaning process and ensuring that all anomalies are resolved before moving on to more complex data processing tasks.

## 3.2 Data Cleaning

### 3.2.1 Correct Data Types:

The first step in data cleansing process is to verify that the data type for each column of the dataset is appropriate. The application of proper data types is useful because it deals with the way the data is organized and operated, which affects the quality and efficiency with which data analysis and modeling is performed.

For categorical data, such columns as Gender; loan_type, loan_purpose and so on are changed to a corresponding 'category' column data type. This is significant since such categorical data types use less memory space and speed up, a great amount of data operations which incorporate categorical variables. Essentially, this practice is adopted so as to avoid the complications that arise when manipulating these columns with too many categories or doing frequent operations such as frequency distribution and aggregating.

In contrast, such numerical columns as loan_limit, loan_amount, rate_of_interest and Credit_Score find their way to 'float' or 'int' data type. For example, columns where the values are monetary or percentages of amount and which areas are being precise or attention are granted are given a 'float' data type. This is due to the fact that 'float' data types are capable of holding fraction elements and thus is good for data whose values carry fraction parts. In contrast, integer values such as the year are converted to "int" since they do not accommodate fractions, the require writing only whole numbers.

The credit score evaluation looks at three key factors The credit facilities and financial responsibilities of an individual The gross income earned from the various income earning activities of an individual. Payments made in remittance obligations and other funded commitments for instance credit cards and mortgages

### 3.2.2 Remove Duplicates:

This is a crucial step that has to be carried out in equal measure in order to avoid instances where similar records are pulled and this has a bearing on the quality of the data set. This can happen through various reasons including, manual data entry mistakes, joining of two or more databases, or the original database containing several records of similar data. These duplicate rows have a significant impact on the analysis outcomes since the presence of such rows introduces bias towards these concrete records, and this affects statistical summaries and model training.

Removing duplicate records is critical to maintaining the integrity of the dataset. Duplicates can skew analysis results and introduce bias, making it important to identify and address them during the data cleaning process [4].

Redundance is the process of identifying particular rows in the table which contain the same information down certain or all columns, and excluding these records. This process is usually accomplished using functions that search for copy rows and that can be flagged based on similar values in one or several key fields. Thus, a data secures approach of ensuring that all entries in a dataset are distinct enhances the subsequent analysis' credibility and reliability, especially when applying techniques like Safe-Level-SMOTE to address class imbalance issues [6]. While the reader cannot see this step in the algorithm given above, it is frequently performed during the data cleaning process in order to keep the data held in a clean and precise form.

### 3.2.3 Fix Inconsistencies:

There are several inconsistencies that require to be cleared up in a dataset that is to maintain its quality. Such differences may accrue from differences in data entry procedures, or errors that accompany the data collection process. For instance, Status might contain categories that are randomly expressed, for instance, 'Approved', 'Approved', 'APPROVED' and 'Rejected', 'rejected' and 'REJECTED'. To do this values are meaned so that everyone has got the same level of values set apart from carrying out their duties independently.

Thus in the piece of code provided here the Status column is changed to binary values say 0 and 1 if the given dataset is for classification. This standardization is important because every machine learning algorithm must take input in a certain format and expect the output in another certain format. Categorical data is first transformed into binary or other standard formats to curb on errant data of the dataset and enhance its compatibility for model easing and checking.This process is similar to the strategies employed in advanced deep learning architectures, such as the Squeeze-and-Excitation networks , which adaptively recalibrate channel-wise feature responses to improve learning efficiency and accuracy [14].

### 3.2.4 Normalize Data

Data normalization is an essential process of transforming the numerical features in datasets before the analysis, particularly for the machine learning models. Normalization refers to a process whereby a range of data values is compressed or expanded towards a particular region, and this increases the efficiency and speed of convergence of many algorithms, as demonstrated in studies such as the Hyperband approach to hyperparameter optimization [16].

Normalisation such as the one depicted in the code has to be done when transforming numerical columns into appropriate data types, however there is no step to normalization for the provided code. Normalization includes shrinking all of the features to the range of between 0 and 1 or modifying them to have a mean of 0 and variance of one. This process compensates so that all the features have equal relevance while seeking the specified outcome or preventing overly potent features from entirely dominating the body. Normalization reduces such bias by making consistency in the features pertaining model performance as well as yielding more dependable outcomes.

### 3.2.5 Review Data Types and Missing Values

It is also important to note that after making type conversions a necessary follow up is to check the data types of all the columns to see if the follow up was productive. A summary of the Dataframe is provided by the df.info() function and this includes the respective datatype of each column which is vital in ensuring that every column is appropriately reformatted in preparation for analysis.

Determining if some values are missing by using the df. ISNA (). Completeness of data is highly important and achieved with the help of sum() function. This step is useful to detect any null or missing values which might have been generated during data cleaning process or were existing in the source data. Dealing with missing values is crucial for data quality with the purpose of avoiding biases and having proper scores of records. For the missing data, the data should be addressed by imputation, deletion or probably other ways based on the type of the data and the need for the analysis that is to be made.

### 3.2.6 Missing/Null Value Treatment

#### Identify Missing Values

First, missing data analysis is conducted on the data by adopting the method df. isna(). , missing() which gives the number of missing values present in each column and sum() which computes the number of missing values in a dataframe. This step provides an easy way of identifying the level of data missing across the dataset as well as its dispersion. For instance, the output may indicate that some of the columns contain missing values whereas, others do not. This assessment is very valuable as it shows which of the components of the dataset are missing and which may need to be filled.

### 3.2.7 Decide on a Strategy

1. **Imputation:**In the present work, numerical columns have missing values which are handled by median imputation. Thus, the median is used instead of the mean since the, it is less influenced by the skewness of data and outliers. The method df. Impute (df. median(numeric_only=True), inplace=True) imputes the numerical datasets within the dataframe by substituting all missing values of the given column with their median value. This approach guarantees that the dataset is retained complete and useful on the data analysis process. Median imputation helps preserving the data and using a mean value to complete the missing data which is making sense in terms of statistic.

2. **Deletion:** Within the scope of the analysis, the availability of values in the Status column could be regarded as the target element. Each hour is assigned a Status and delists rows: this avoids the problem of some models not working with unknown target values during inference. Since the Status column is the most important for this analysis, the absence of data for that variable leads to deletion of the rows containing that data. It results in full case analysis.

**Check for Missing Values Again:** This step is important and therefore the particular dataset is checked again for missing values using df. isna().sum. After performing the imputation and deletion strategies, this step was carried out in an attempt to double check how the missing values in the dataset were dealt with. The output indicates after the data has undergone the handling of missing values whether there are any values remaining that are unaccounted for. After every step including handling of missing values using appropriate strategies the data is checked. This gives room for further uses of the data as it applies to this step by properly comparing the number of missing values before and after handling them.

This method of dealing with missing values helped in the consistency of the dataset structure and integrity enhancing it hence reducing the risk of carrying out any biases while improving the quality of the analysis.

## 4. Data Review

**Validation:**

Having processed the data cleaning and pre-processing step, it is rational to transform the data on the basis of operations that have been performed on it, and then make sure that the changes . This entails going round to make sure that all the predetermined cleaning tasks that include type conversions, missing values handling, and duplicate elimination among others have been correctly performed. For example, checking an informational message – all numerical columns are actually numeric and all categorical – are actually categorical – makes certain that the dataset is in a state where it is ready for analysis. Other checks that need to be performed as part of validation are checking whether the steps taken to deal with missing values and the rows containing missing target variables were done properly. This step is particularly important so as to ascertain that the dataset is now clean, consistent and ready for analysis. In the context of model optimization, having a clean and properly validated dataset is critical for techniques like Bayesian optimization to efficiently search the hyperparameter space for the best-performing models [3].

## 3.3 Exploratory Data Analysis (EDA)

### 3.3.1 Understand Data Distribution

Deriving some insights from the distributions of various data sets is an initial step that should be done first in EDA. By gaining such knowledge, it is easy to understand how the values of various features are spread over the dataset. Usually, the first step in this case proceeds with a graphical representation of the numerical values, hence the use of histograms. The visual indicate how many data points fall into several predefined ranges or bins, and whether there is normal, skewed or outlier distribution of the data.

For example, in the case of analysis, the employed histograms for the purpose of exploring the distribution in the quantitative characteristics for instance loans, rates of interests and so on. Such plots are useful in determining the nature of the distribution of the data for instance the normal distribution as well as assessing the presence of outliers. For instance, when the loan amount is displayed in a histogram, it may be seen that most of the loans are within a certain intervals and several unusually small or big values may be observed.

### 3.3.2 Descriptive Statistics:

Most of the techniques come up with descriptive statistics that give an insight of the kind of data that is contained in the dataset to help in appreciating the general patterns or distribution of the data. For numerical features, first, second, third quartiles, and inter-quartile range for each of the features will tell about the central tendency and the spread of the feature's data. In the case of categorical data, use of frequency counts and mode values can be used in an attempt to determine the frequency of occurrence of each category. That is why statistical summaries are also useful, as they allow indicating the existence of

**Cross-Verification:** Through cross-checking, or validation, one reconcile the cleaned data set with the raw data in order to ensure all necessary information was not lost during cleaning. This comparison exercise ensures that the number of rows and the number of columns in a cleaned dataset are same as the numbers in the initial dataset to ensure structure integrity.

And furthermore, the comparison of some records from the raw and cleaned dataset can also reveal possible mistakes which might have occurred during the cleaning process and data loss. This procedure is vital as it checks that data cleaning exercise has not distorted the data and that all data is in the data set as it should be.

It is crucial to keep a record of the data-cleansing procedures for the sake of reproducibility. This record contains, for example, a description of each cleaning step taken, why certain decisions were made (e.g. median imputation was preferred to mean), and problems which came up and how they were solved. This facilitation by recording the methods employed, and the decisions made aids in building a knowledge structure that helps in bringing about consistency, and reducing duplication of effort while attaining targets and more importantly, enhances learning. Such transparency is paramount, in that, it provides faith that, the cleaning process is such that someone else can review it, and audit it, and even replicate it, and therefore bolster the integrity of the dataset that will be used afterward in analyzing data.

### 3.3.3 Distribution of numerical features

The histograms to visualize the distribution of numerical features of the given dataset. By using the df. iIn general, when the hist() function is applied, it generates graphs of histograms in which each graph is related to the frequency of values of a particular numerical column. To achieve this the figure size is set and the other lines of code are used to produce the figure as shown below using the command plt. show() displays these histograms. It serves for identifying where values are dispersed, for understanding distributions and finding regularities as well as skewness or suspicions of outliers. It is vital such insights are obtained to help in further data preprocessing and analysis.

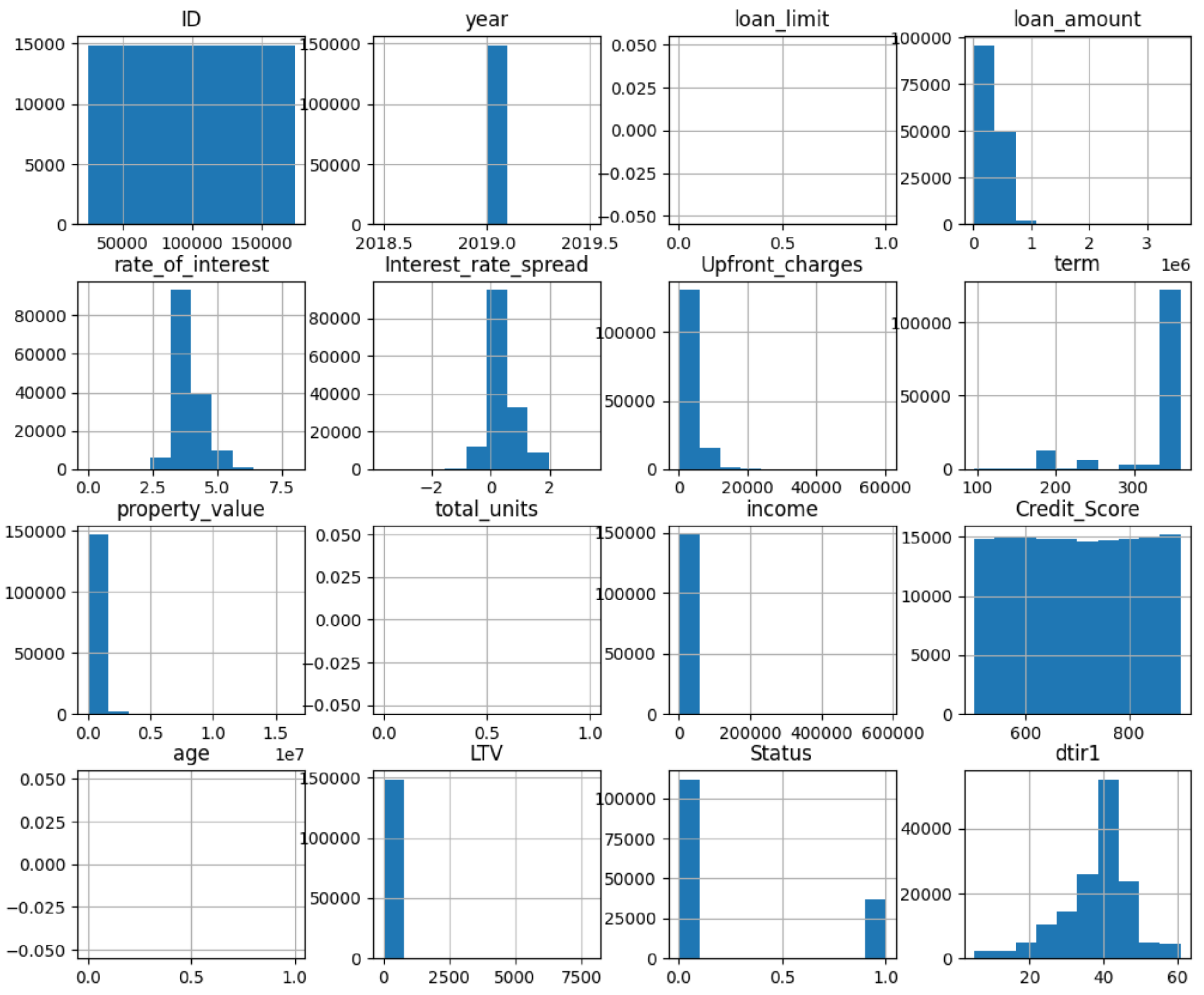

Figure 3.1: Distribution of numerical features

### 3.3.4 Distribution of Loan Amounts

It employs the use of a histogram to come up with the right distribution of loan amounts in the dataset. In that way, when using the figure that has a dimension of 10 by 6 inches, it will also make the plot visible clearly and appropriately sized. The sns. histplot function is used for the plotting of loan amounts, and the number of bins is set to 30 with overlay of a kernel density estimate (KDE) for the loan amounts. This KDE line proves beneficial in getting an overall picture about the shape and density of the loan amounts. The plot is titled "Distribution of Loan Amounts" with appropriately labeled axes: And for the vertical axis they have used labels “Loan Amount” and for the horizontal axis they have used ”Frequency”.

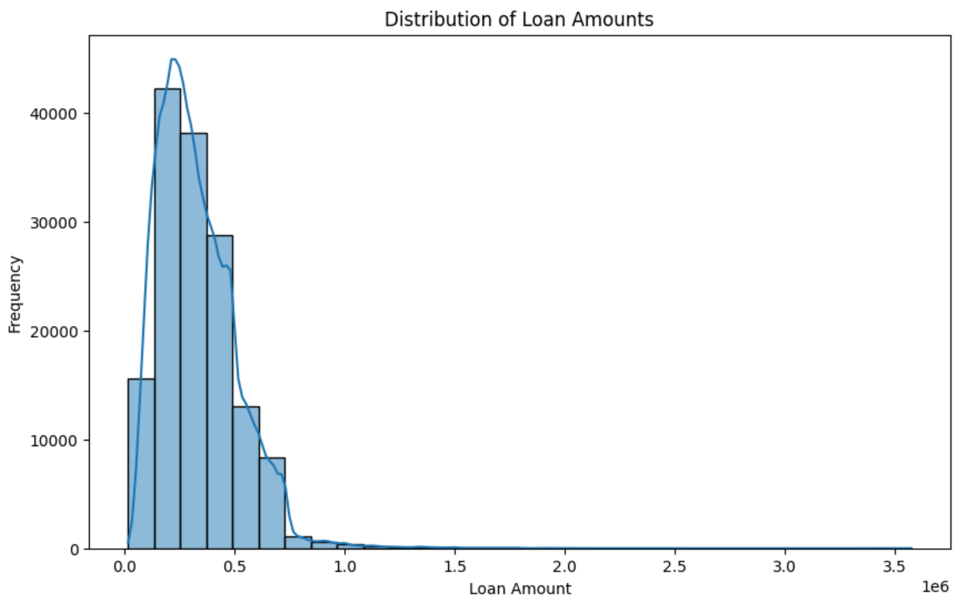

Figure 3.2: Distribution of Loan Amount

### 3.3.5 Distribution of Interest Rates

Using the hist() function provided by matplotlib the histogram is plotted in order to look into the range of interest rates (rate of interest substring) that are present in our data. In this case the figure size is set to 10 by 6 inches to avoid a compressed presentation of the plot. The sns.histplot function performs the plotting of the rate_of_interest column by fitting it into 30 bins or intervals in order to represent the variation of interest rates in the country. Some figures also come with a degree of smoothing, such as the KDE, where one tries to curve a smoother function in this case over the data to were the tendency lies such as the actual independence or concentration of data.

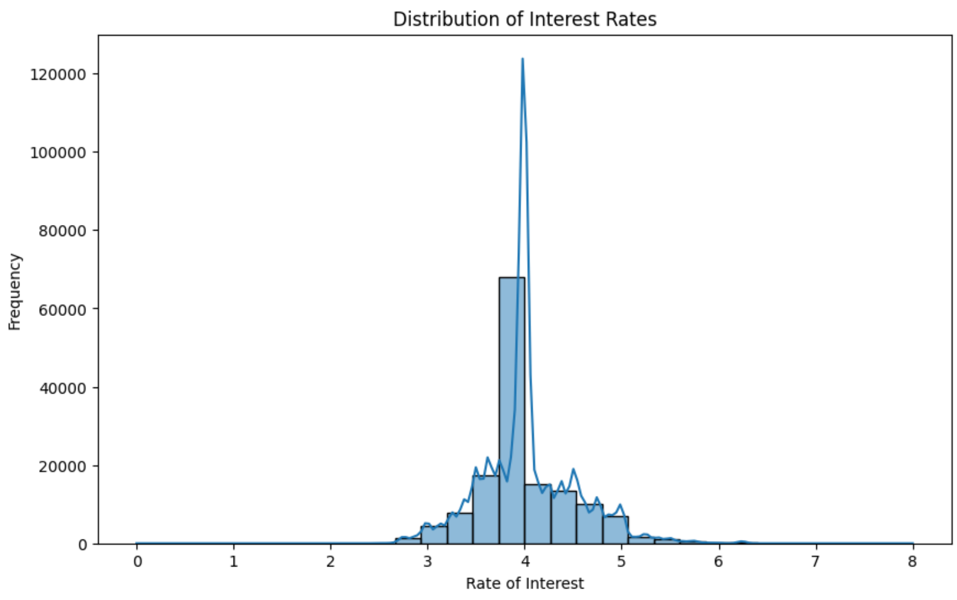

Figure 3.3: Distribution of Interest Rates

### 3.3.6 Box Plot of Loan Amount by Loan Purpose

This section gives the information in the form of graphical representation. In this instance, all the loans taken out by the borrowers are categorized into separate boxes according to the purpose of the loan and then the amounts in the boxes laid out. This extends the dot chart by adding vertical lines to each data point to depict the distribution of data points across different purposes against the amount of interest charged on every loan.

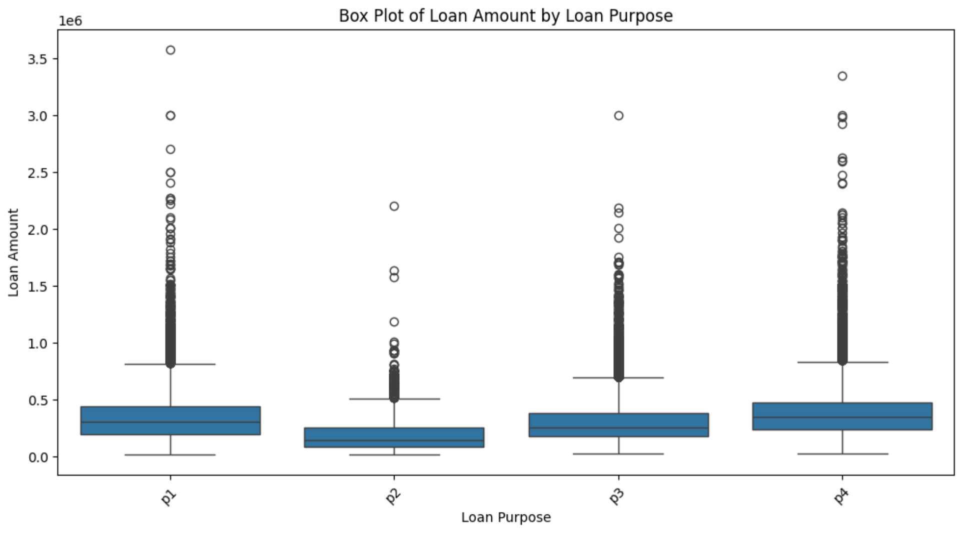

Figure 3.4: Box Plot of Loan Amount by Loan Purpose

### 3.3.7 Box Plot of Rate of Interest by Region

This enables one to generate a box plot to ascertain how the mean of interest rate varies with regard to the geographical zone. Figure size being 12 by 6 inches, shows a free and clear view of the information as seen. The region feature is used as a categorical variable and rate_of_interest is the numerical variable, from sns.boxplot depicting distribution of interest rates across different regions by drawing their box plots against the categorical variable "Region".

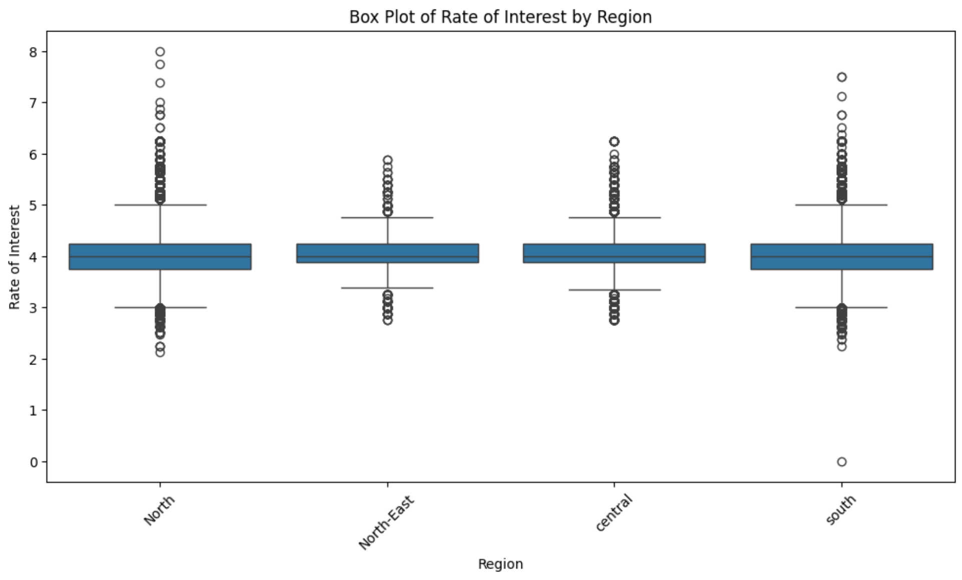

Figure 3.5: Box Plot of Rate of Interest by Region

**Bar Plot of Loan Types Count**

This generates a bar plot to visualize the count of different loan types within the dataset. By setting the figure size to 10 by 6 inches, it ensures that the plot is clear and easily readable. The sns.countplot function is employed to create the bar plot, with the x parameter set to 'loan_type' to display the different types of loans on the x-axis, and the data parameter specified as df to use the dataset for plotting.

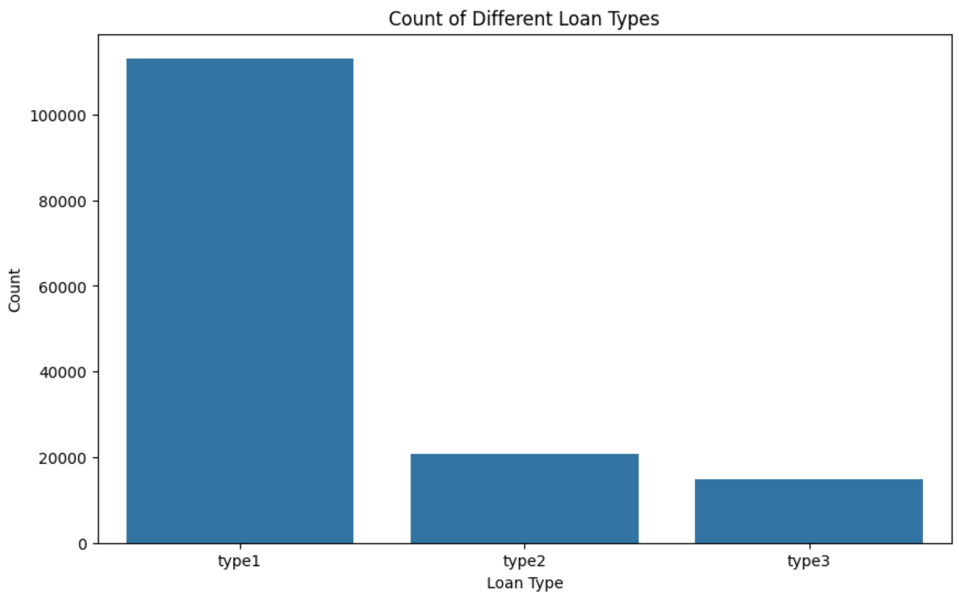

Figure 3.6: Bar Plot of Loan Types Count

### 3.3.8 Heatmap of Correlation Between Features

This is done by using data frame df to choose only numerical column from the dataset. select_dtypes (include=['number']). This step is important as the correlation heatmap only requires quantitative variables so as to represent the relations between features. The output is a numerical_dexterously that same contains all columns of data type which is numeric.

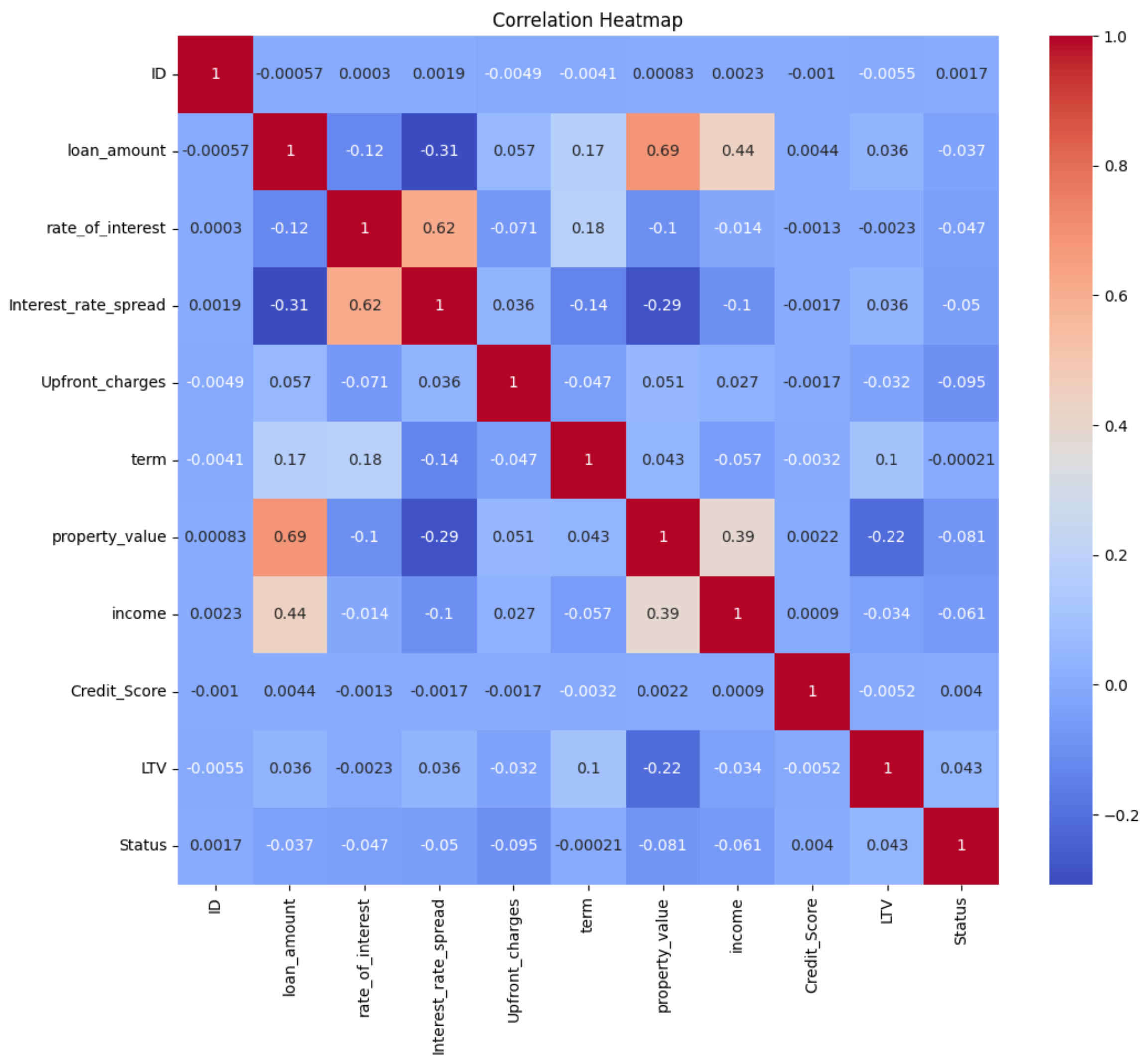

Figure 3.7: Heatmap of Correlation Between Features

### 3.3.9 Count Plot of Gender

First, it draws figure with the dimension of 10*6 inches in order to make plot unambiguous and adequate. The sns. The actual plot is plotted using countplot function where x-axis is the ‘Gender’ column of the data set and the data parameter is defined as the data frame df. This function tells the number of occurrences, thus gives the frequency of each gender in the ‘Gender’ column.

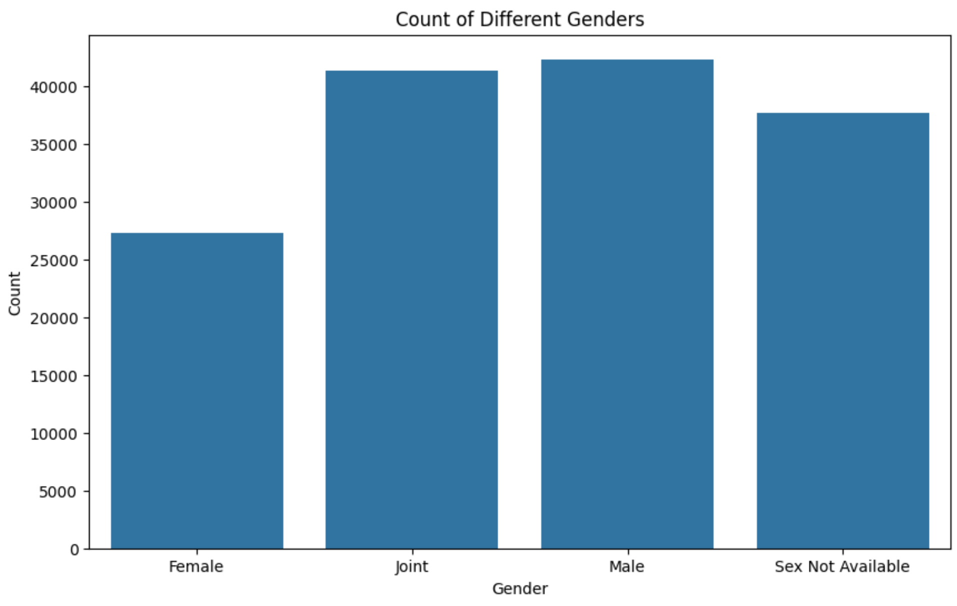

Figure 3.8: Count Plot of Gender

### 3.3.10 Pie Chart Graphs to Check Frequency of Region and Loan Purpose

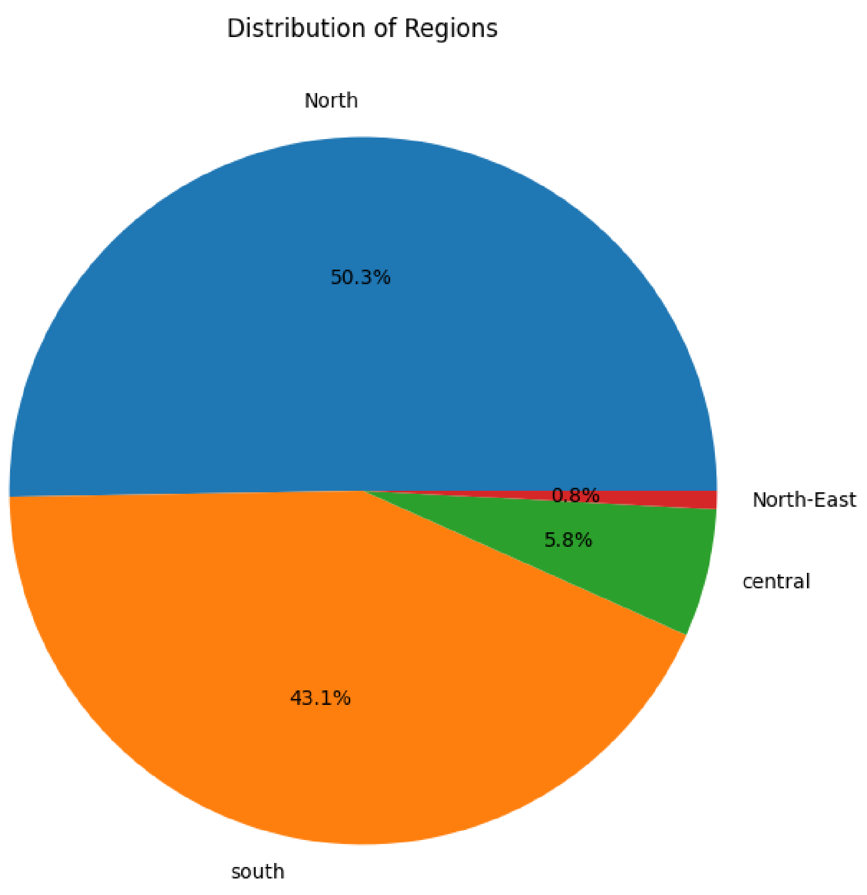

Figure 3.9: Pie Chart Graphs to Check Frequency of Region and Loan Purpose

First, in the analysis, the function of value_counts() is used to estimate the count of occurrences for each of the unique regions. These counts describe the level of loan application formation as a function of the specific region. The plt. A pie chart is then displayed with the 'pie()' function The size of each slice reflects the number of applications from a region. The autopct='%1. 1f%%' parameter represent the mathematic relation of the each slice and whole pie chart fraction, so it is convenient in the making an analysis of the relative proportions of the areas. Last of all, the pie chart is presented with a title helping to give a concise idea of distribution of loan applications by geography.

### 3.3.11 Distribution of Loan Purpose

The code creates a pie chart in order to find out the proportion of different loan purposes in the dataset. For the first time, it counts the frequency of every loan purpose using the value_counts() function, which counts the unique values in the loan_purpose column. Pie Chart is made with the help of Matplotlib in which each slice corresponds to a loan purpose and the size of the slice is the number of loans made for that purpose as a percentage of all loans made. It describes what each loan is used for and the percentage of the total amount of the entire category. This graph is called ‘Distribution of Loan Purpose’ and it states how loans are classified according to the purpose for which they would be used.

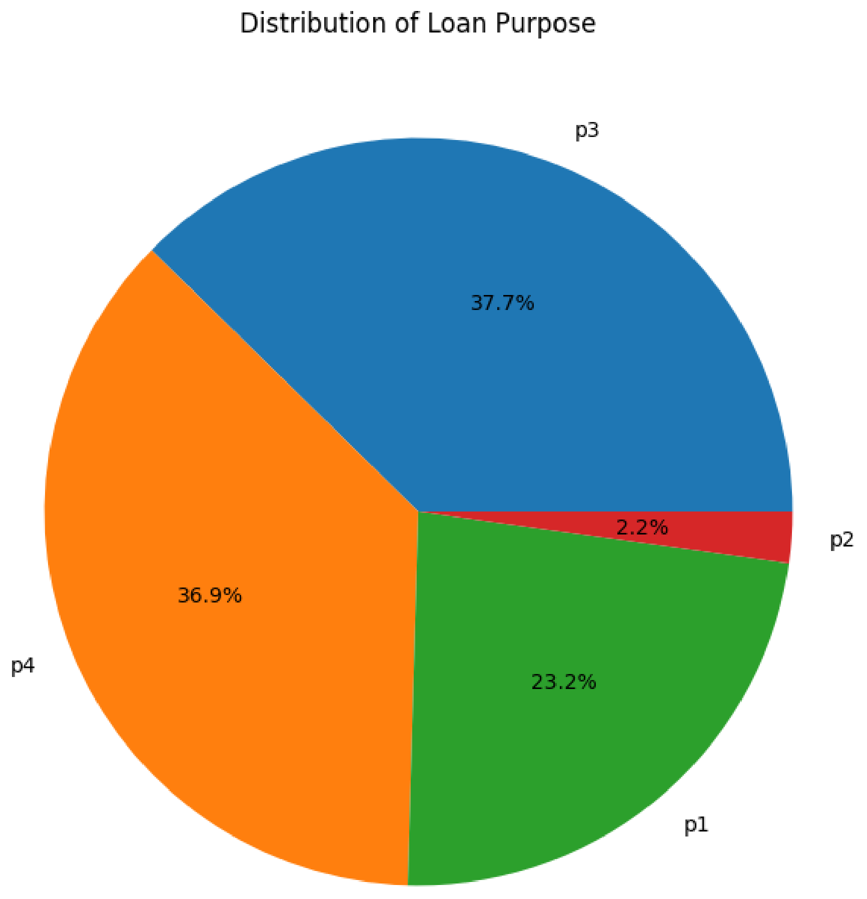

Figure 3.10: Distribution of Loan Purpose

CHAPTER 4

# Methodology

## 4.1 Feature Engineering

Feature engineering is a very important one for it involves the creation of features from extracted data which will be use to train a machine learning model. The main goals in this step are to define the input and output variables, find out co-dependencies between attributes, excludes unnecessary data and transform categorical variables to make them suitable for models. All of these tasks help to enhance the model that is used for the prediction in order to ensure that the best model is achieved.

### 4.1.1 Feature Selection

Feature selection is Drop Irrelevant Columns the screening of selecting several fruitful factors in a dataset which considerably influences the performance of the model. When come to the project, there was much more emphasis on deciding which numerical features should be included in the dataset because generally, numerical features offer quantitative data regarding the model which is quite crucial for training a model. By using the include_data_type filter only the float64 and int64 type columns are included into the dataset to analyze which of the features are most informative and which of them can be mathematically manipulated. This step is however very important because it minimizes the model's dimensions by removing features without correlation to the outcome, thus enhancing both the comprehensibility of the model and the model's effectiveness. If it is necessary to predict something, then numerical features will present the key characteristics of patterns and trends of data.

### 4.1.2 Checking Relation Between Attributes

There is another important part of feature engineering which is to understand how different numerical attributes are connected. This is achieved by using feature correlation analysis which assist in evaluating how one or more features are linked to each other. As for the feature engineering process in our project, we calculated correlation matrix of the numerical features and then display it in the form of the heat map. The heat map shows PCC, values ranging between -1 meaning that degree of negative linear correlation and + 1 meaning there is degree of positive linear correlation. This kind of feature evaluation also shows high positive or negative correlation between features and points to redundancy situation when one feature is a good predictor of another one. This information is useful in times when the program needs to be pruned by combining, transforming or dropping features to eliminate some problems which are common such as multimillionaire. By recognizing these relationships it allow us to construct model on feature that contribute exclusivity to the prediction which makes the model more reliable.

### 4.1.3 Drop Irrelevant Columns

There may be columns in a dataset with no relevance or very little relevance to the predictive modeling. Such columns might include identifiers, redundant information or those that do not relate to the target variable. At this point, we have taken into consideration the dataset again and have pointed out this limitation and sought for its alleviation. The 'submission_of_application' and the 'dtir1' columns were eliminated on the other hand as they were irrelevant to the analysis or indeed added no value in terms of predictive ability. It is highly important to remove such non-informative features as such actions help to deal with the technological problem of dimensionality reduction in the dataset which not only helps in hastening the computation but reduces the chances and effects of over fitting by getting rid of noise and unhelpful data. The model can now work with a diminished dataset that can contain only the important variables which increases the model accuracy on new data set and also its generalization ability.

### 4.1.4 One-Hot Encoding

It meant that input features needed to be in a numerical form in order for the machine learning models to be used. Categorical data, which are non-parametric information in the form of text string, must be encoded on the numerical format before being used in modeling. A technique that may be used for this is called one hot encoding [**?**]. In this project, we performed one hot encoding on categorical variables because it was observed that the variables along with their different categories are different columns with the value 1 if the category has occurred and vice versa. This approach proves to be useful because it does not distin guish the categories in terms of a fake ordinal measure that can mislead the model. While building this model, the drop_first=True option was employed so as to make the model to have the least multicollinearity index. After that, the dataset becomes fully numeric and can be immediately fed into the machine learning algorithms since all the information translated from the categorical variables is stored in a form of numbers which will be more easily interpreted by a machine learning algorithm. This is crucial for the model that is intended for making the predictions based on all the gathered data during the experiments.

## 4.2 Data Splitting

Split of data is a very important stage in the process of creating an ML model as it allows to check whether the model performs well enough on samples which it has never seen. This means by using separate subsets of data the model is checked and validated each time by using separate subsets, normally referred to as the training set and the testing set. The reason for this step is to avoid overfitting, a scenario whereby the model gives excellent results for the training data but performs dismally for the actual data sets that it is supposed to predict. Data splitting helps in a right assessment of the measure of accuracy the model has completed so as to yield credible results when tested on other sets of data.

### 4.2.1 Define Features and Target Variables

Once data has been preprocessed, the step that follows is defining the features as well as the target variable. A successful model of machine learning will depend on the data that is fed as input to it. These are referred to as features also known as independent variables or predictors of the model being used. In this project, after encoding the categorical variables, the dataset df_encoded was prepared with both the features and the target variable. The features matrix X was constructed by removing the 'Status' column from df_encoded. This matrix includes all the relevant factors that are likely to affect the outcome, these are loan amount, interest rate, loan purpose etc. However, the target variable, which depicts the status we want to predict, whether a loan will default or the user will repay the loan or not, was the column titled 'Status'. It is very important to make this distinction between features and the target variable as it helps the model to learn the relation between predictors and the outcome and not run through the data.

### 4.2.2 Perform Train-Test Split

The third and final step is to divide the set of features and the target variable; the data set can then be split into training and testing categories. The division of the dataset was done with the help of train_test_split function from the sci-kit-learn library which comprises of this splitting type. Here in this project, we further split 70 % of the data into training set data (X_train, y_train) and the remaining 30 % as test/evaluation data set (X_test, y_test). The parameter for the test size was left as 0. 3 which calls for a 30 % of the entire data set for testing purposes. Further, random_state of 42 was applied so as to make the split to be consistent that is each time the code is executed the data split should be similar. This consistent process is advisable in order to avoid errors due to partitioning of data and also help in gaining comparable results between different experiments. Training dataset is employed for training the model where the model acquires the knowledge of the existence of certain patterns in the data. As for validation, on the other hand, it's aimed at checking the model's ability to make correct estimations on the data it was not initially trained on and the testing set is used for this purpose. In this way, the division of this dataset in such a precise manner helps in establishing a strong framework for the machine learning model that does not overspecialize and is thus more universal and strong.

## 4.3 Data Scaling

Scaling techniques are one of the most important pre-processing steps in most of the machine learning, especially for the algorithms that are highly sensitive to the distance between the feature space and the target or those algorithms that involve optimizations of some gradient descent methods (e. g. , logistic regression, neural networks). Data scaling is namely about making all the features equally relevant to the model learning by normalizing the feature values to the same range. This step aids in accelerating the optimization of convergence of gradient descent, increases the model's accuracy, and reduces the skewing effects of features of larger magnitude in the learning process.

### 4.3.1 Apply Standard Scaling

In this project, there was the use of the StandardScaler from the sci-kit-learn library in the process of feature scaling, to make use of. Standard scaling another conventional method of normalization, in which each of the variables in a given set of data is made to have a mean of zero and a standard deviation of one. This makes certain that the values are brought, on average, to zero and that they are scaled by the amount of variation existing in the feature. Standard scaling makes the model more stable and accurate since it does not give any special attention to large features as compared to small features of data. This is especially relevant in this project because things like loan amount or interest rates could have drastically different ranges and normalizing will prevent a single feature from dominating the model's decision making process.

### 4.3.2 Fit a StandardScaler on Training Data and Transform Both Training and Test Data

Before applying the scaling operation, StandardScaler was fitted on training data using fit_transform() method In this stage, the means and standard deviations are calculated solely based upon the training data The fit_transform() method was applied upon X_train to both fit the scaler and transform the training data given these means and standard deviations While doing so, the scaling is derived from the training set to preserve the purity of the model assessment On applying the above The resulting sets, which are X_train_scaled and X_test_scaled represent the training and tests sets for the model training and tests respectively to enable feed the model with standardized inputs which increases the model's reliability.

## 4.4 Addressing Class Imbalance

This problem of class imbalance is very common in most original datasets when it comes to real word applications such as financial analysis, health studies and fraudulent activities [**?**]. The next type is class imbalance where one of the classes has many more instances than the other classes making the model prediction bias. When it comes to loan default prediction, this means that the number of samples not in the loan default class is much bigger than the number of samples that are in the loan default class. This can be inconvenient because a majority of methods implemented in machine learning presume that classes contain a comparable number of records. Consequently, a model may end up giving highly accurate models at the aggregate level (if they accord highly with the non-defaults), but very low accuracy to the exceptions (the defaults), which are typically of more concern in credit risk management.

In order to address this issue, a Synthetic Minority Over-sampling Technique (SMOTE) was adopted in this project. When it comes to SMOTE, it is much more refined than the methods described above since it not only balances the dataset, but also increases the model's capability to generalize because it artificially creates new samples rather than simple copies.

SMOTE works by generating synthetic samples for the minority class. It does this by selecting random data points from the minority class, finding their nearest neighbors in the feature space, and generating new synthetic data points by interpolating between the original data points and their neighbors.

**Key Steps of SMOTE**

1. **For each minority class sample** $x_i$**:**

- Identify the k-nearest neighbors from the minority class using a distance metric (e.g., Euclidean distance).

2. **Random Selection**:

   - Randomly select one of the k-nearest neighbors $x_{NN}$ of $x_i$.

3. **Synthetic Sample Creation**:

   - Generate a new synthetic sample $x_{\text{new}}$ by interpolation, using the formula:

$$x_{\text{new}} = x_i + \lambda \times (x_{NN} - x_i)$$

   where $\lambda$ is a random number between 0 and 1. This generates new data points along the line segment joining $x_i$ and its neighbor $x_{NN}$, ensuring that the synthetic data falls within the feature space of the minority class.[?]

### 4.4.1 Apply SMOTE

The application of SMOTE is carried out with the belief that mere oversampling of the minority class is likely not to be enough as this involves the creation of more copies of the same instances present in the minority class. This can result in overfitting where the by the model becomes overly optimized to the training data and performs poorly on unseen data. However, SMOTE is much more refined cause it actually generates synthetic data points. All the new examples are generated selecting two or more similar examples in the feature space then averaging to generate a synthetic instance. This makes the class probabilities to be more spread out in the feature space which increases probability of the model to distinguish between meaningful patterns that define both majority and minority class.

However, for SMOTE to work efficiently, it was necessary to check and remove any missing value within the training dataset (X_train_scaled) and the target labels (y_train). When certain data is unaccounted for, then the models developed are usually off-track and may not generalize well. In light of this, the SimpleImuter from sci-kit learn was used This imputer works in removing outliers which are beyond the range defined for each feature. This tool helps in imputing the missing values in the dataset with mean, median or mode of the column. In this project, the mean imputation procedure was adopted which involves replacing of any missing value by mean of the relevant feature column. This step was important in order to eliminate any possibility of synthing samples from incomplete data which could result in very high density areas to be covered with the synthezied samples.

This was done after imputing for any missing values that the training set can have after the initial data splitting was done. For the minority class the SMOTE function from the imbalanced-learn library was employed to generate new samples of the feature space. This method proved useful in equalising the class distribution of the training data so as to give X_train_smote and y_train_smote which comprised of an equal number of instances from the majority and minority classes. This balanced training set is important because as the model is trained, it does not rely mainly on defaults but also on non-defaults so as to increase accuracy.

Not only does applying SMOTE correct the class imbalance it also helps improve the ability of the model to identify finer details and patterns within the minority class which could go unnoticed in the highly imbalance data set. This makes the model more accurate and reliable when it is implemented in a real scenario thus improving the performance of the business.

## 4.5 Model Development

### 4.5.1 Random Forest:

In considering its capacity to manage difficult data and provide accurate forecasts using multiple decision tree polls, Random Forest was chosen for this investigation. In order to address class imbalance, two different versions of the Random Forest models have been developed: one that made use of the SMOTE-Synthetic Minority Oversampling Technique and another that does not. Comparing the model's performance with and without balanced classes will be given as a result.An additional ensemble algorithm is Random Forest.To improve classification, it constructs a large number of decision trees and then puts their result together. In terms of mathematics, it is expressed as

The function $f(X)$ is computed as follows:

$$f(X) = \frac{1}{T} \sum_{t=1}^{T} f_t(X)$$

where T is the number of trees built and $f_t(X)$ is the decision tree's output for the tree. A random selection of the data and features served as the foundation for each tree [5].

The Gini Impurity, which is defined as follows, is a common way to quantify the quality of splits in a decision tree:

$$\text{Gini}(S) = 1 - \sum_{i=1}^{c} p_i^2$$

Where $p_i$ is the proportion of instances of class node S In this case,Further, Random Forest was used both with and without SMOTE. It performed exceptionally well in classification tasks overall, but when trained with SMOTE,it tended to overfit.

1. **With SMOTE:**
   SMOTE was used to generate synthetic samples of the minority class against the original training dataset because the dataset is unbalanced and has a significantly higher proportion of non-defaults than defaults. In order to create new synthetic instances, SMOTE interpolates between already-existing examples of the minority class. By doing this, the data is balanced and prevented from becoming partial throughout training towards any one class. Next, using SMOTE to train a Random Forest model using balanced training data that had been resampled came next. To evaluate a model's performance under practical generalisation conditions, an unbalanced test set was used for testing.

2. **Without SMOTE:**
   In the second iteration, SMOTE was not used; instead, the Random Forest algorithm was built directly from the unbalanced dataset. As a result, the raw data distribution was used for model training, and there were many times as many non-defaults as defaults. This is done in order to investigate how an imbalanced class distribution, which causes class imbalance, affects model predictions and performance.

### 4.5.2 Logistic Regression:

Loan defaults are predicted using logistic regression, one of the most widely used linear classification techniques. A linear relationship between independent factors and the log-odds of the dependent variable is assumed, and the probability of a default is obtained for a given input set of attributes. Logistic regression is an excellent starting point for more complicated models in evaluation because of its simplicity of use and interpretability. A Logistic Regression model with SMOTE for class imbalance and another without any sample modification were the two versions of the model that we used.classification algorithms of the Logistic Regression type are widely used. By employing the Sigmoid function to transfer any real-valued number into the range of 0 to 1, logistic regression models binary categorisation. The following is the equation for logistic regression:

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

where:

- $P(y = 1 \mid X)$ indicates the chance of the positive class (defaults on loans)
- $X_1, X_2, \ldots, X_n$ are the input features
- $\beta_0$ is the intercept term, and $\beta_1, \beta_2, \ldots, \beta_n$ are the input feature-corresponding coefficients.

The parameter estimates for  have been obtained through maximum likelihood estimation by maximizing the log-likelihood: (Hosmer et al., 2013)

$$\text{Log-Likelihood} = \sum_{i=1}^{n} [y_i \log(P(y_i)) + (1 - y_i) \log(1 - P(y_i))]$$

In the present study, loan defaults are predicted using logistic regression. Using an over-sampled minority class, the SMOTE algorithm was developed to address issues with class imbalance [**?**]

1. **With SMOTE:**
   The training dataset is tested with SMOTE in order to address class imbalance. Due to the very large proportion of non-default entries in the original data compared to default records, the model would probably be biassed towards predicting non-defaults. To provide a nearly balanced dataset in the train set for the approach, synthetic points relating to the minority class—that is, loan defaults—will be created using SMOTE.For giving equal opportunities for learning from both classes, this balance improved the accuracy of the Logistic Regression model's loan default prediction. After that, the

model was trained using the SMOTE-preprocessed dataset to guarantee that learners received effective models from both the majority and minority classes. During training, the log-loss function is optimised to reduce the gap between the expected and actual outcomes. Following that, the model's generalisation performance on real-world data was tested using the unbalanced initial test set as a surrogate.

2. **Without SMOTE:**
   On the original, imbalanced data from the present model, logistic regression was performed without the use of any oversampling techniques. The data was used to directly train the model since it is highly unbalanced, in more instances of "no default" than those of default. The next steps have been made to follow the model's performance in a typical scenario without extra steps taken to correct the imbalance. The absence of SMOTE would result in a bias in the model it favours the majority class, or non-defaults, and affects recall for the minority class of interest, or defaults. This model learns to minimise a log-loss function; but, given the huge imbalance between both classes, it may become less accurate in recognising loan defaults as it becomes more focused on the prevailing class, or non-defaults. With the help of this version, the predictive efficacy of a linear model such as logistic regression can take into account the impact of a class imbalance.

### 4.5.3 XGBoost:

One of the most robust and effective methods for creating an ensemble of weak learners that would gradually reduce the classification error is XGBoost, which is based on the gradient boosting methodology. It is a powerful model for loan default prediction due to its ability to take into account complex interactions between data, feature interaction, and class-imbalanced datasets. To determine if the algorithm would perform properly in predicting loan defaults under various class distribution scenarios, XGBoost is run both with and without SMOTE in this investigation. A sophisticated use of gradient-boosted decision trees, XGBoost (Extreme Gradient Boosting) aims to maximise speed and performance [8].The objective function in XGBoost consists of a loss function $L$ and a regularization term $\Omega$: .

$$\text{Obj}(\theta) = \sum_{i=1}^{n} L(y_i, \hat{y}_i) + \sum_{k=1}^{K} \Omega(f_k)$$

Where:

- $L$ is the loss function; typically, logistic loss is utilized for binary classification.
- $\Omega(f_k)$ regularizes the complexity of the model.
- $\hat{y}_i$ is the predicted output.

XGBoost is incredibly effective at optimising model performance because it uses a second-order Taylor expansion of the loss function. According to[17], the relevance of a feature in XGBoost is determined by how frequently it appears in splits and how much it adds to the total gain.

1. **With SMOTE:**
   first being used in the XGBoost model's training, the training data completed SMOTE to address the dataset's class imbalance between the default and non-default classes. By applying existing cases, SMOTE will generate synthetic examples of the class that demonstrate loan defaults, balancing the dataset. As a result, there is less bias towards the majority class and the model is ensured to be trained fairly on the basis of defaults and non-defaults. Further, the ensemble implementation improves XGBoost's capacity to handle unbalanced data. Also, the model's capacity to generalise and predict better for the minority class—particularly with regard to loan defaults—should improve by the combination of SMOTE and XGBoost. While keeping their strong performance in predicting non-defaults, XGBoost improves learning from the minority class with data balanced by SMOTE.

2. **Wihout SMOTE:**
   Without using SMOTE, this second variation was trained using the initial unbalanced data. In actual use, XGBoost is relatively robust to the issue of class imbalance because it offers and strengthens difficult classification samples. Since non-default situations still made up the majority of the training data, the model was predicted to predict non-defaults.The model trained without SMOTE will first allow evaluations of XGBoost's

efficiency in more realistic cases, where class imbalances would typically be common. The model would probably have trouble accurately predicting the minority class of interest—conditioning loan defaults—even if XGBoost is able to withstand a substantial amount of class imbalance. In order to measure how much SMOTE truly improves the balance of classes and loan default prediction accuracy, this variability then serves as an ideal level.

### 4.5.4 Linear Discriminant Analysis (LDA):

The linear classification method known as "linear discriminant analysis" looks for a linear feature combination that has the most ability to divide two or more classes. Classes that are linearly separable are when LDA really shows. The study has tried using LDA to predict loan defaults, both with and without SMOTE for checking purposes and with SMOTE for a class imbalance issue.

The basis of LDA, a classification method, is the assumption that data belonging to various classes are produced by Gaussian distributions with the same covariances. LDA seeks to maximise the distance between classes while projecting the data onto a lower-dimensional space [4].

$$\delta_k(x) = x^T \Sigma^{-1} \mu_k - \frac{1}{2} \mu_k^T \Sigma^{-1} \mu_k + \log \pi_k$$

Where:

- $\mu_k$ is the mean vector of class $k$,
- $\Sigma$ is the shared covariance matrix,
- $\pi_k$ is the prior probability of class $k$.

LDA has been applied for the prediction of loan defaults, to which SMOTE was applied to improve its capability in capturing the minority class of loan defaults.

1. **With SMOTE:**

   The SMOTE algorithm is used to synthesise examples of the minority class of interest, defaults, because the loan default data points are highly imbalanced, with the majority coming from the non-default class. This variation first uses SMOTE to balance the dataset before training the LDA. The LDA was able to acquire the signature of both classes much more effectively by resampling the data.In addition to providing a wide range of default situations during training, this will ensure that the LDA model has no bias towards the current class. Classes are balanced to allow it to anticipate a minority class better, which is represented by loan defaults, without favouring the majority class, which is shown by non-defaults.

2. **Wihout SMOTE:**
   In that case, the unbalanced dataset was used directly to train the LDA model, and oversampling methods like SMOTE were not used. The majority of non-defaulters vastly outnumbers the minority class of defaulters in this instance, given to the model's analysis of the data's natural distribution.Because LDA primarily uses class distributions to create decision limits, the training dataset's majority of non-default cases may have a biassed effect on the model. Since the model did not have enough opportunities to learn from the training, it was unable to correctly identify the instances of default cases at the time of prediction. Because class balances play a significant role in improving predictive performances, these variations allow the model to compare its performance when dealing with imbalanced data without any problems.

## 4.6 Model Predictions

With the test data properly prepared, we proceed to make predictions using each of the four models:With the test data properly prepared, we proceed to make predictions using each of the three models:

- Random Forest Predictions:Random forests is the second modelling method developed from an ensemble learning method that is capable of handling high dimensionality data and possesses good robustness in handling the target variable (Status). This model combines the outputs of numerous decision trees which can contribute in enhancing prediction accuracy besides avoiding over training. Applying the model to impute the test data we get y_pred_rf which depicts the prediction that the model has about the data.

- Logistic Regression Predictions: Logistic Regression is a frequently used linear model which gives an estimation of probability of occurrence of a binary event. It is quite useful when it comes to assessing correlation of target variable with the features. The predicted values, y_pred_logreg are the predicted values of the test set or in other words the test data itself which have been classified by the model on the basis of probabilities.
- XGBoost Predictions: XG Boost is one of the best gradient boosting models that work best on structured data. It constructs the trees one at a time, where each tree is a correction of the previous one, making the predictions as accurate as possible. The XGBoost model is also used predict the test data and the predicted value is stored in the variable y_pred_xgb after which its performance is assessed.
- Linear Discriminant Analysis (LDA) Predictions:Linear Discriminant Analysis (LDA) in its turn searches for linear combination of features in order to separate the classes in the best manner. The model then determines the possibility of the test points being in a particular class then assigns it to that class with greatest probability density. This method is most effective when the data has a of 'bell shaped' distribution and an linearly separable.

## 4.7 Model Evaluation:

The performance of the machine learning models used in this work to predict loan defaults has been measured using a number of evaluation indicators. For a thorough understanding of model performance, normal and class-sensitive combination metrics were used because the dataset was very unbalanced, with non-default situations outnumbered default ones. The evaluation's primary objectives were overall correctness and the minority class's proper classification. In applications pertaining to finance, this is important.

### 4.7.1 Accuracy:

The ratio of accurate default and non-default forecasts to total predictions is known as accuracy. The primary statistic is accuracy, which is a direct measure but can potentially be misleading when applied to an unbalanced dataset. The accuracy was computed in this study to give a general indication of model performance; yet, due to the extremely skewed nature of the class distribution, the focus is mostly on the class-specific metrics. These implementations tested the effects of data balancing on the overall correct classification rate using the classifiers that were given, both with and without SMOTE.

**Precision, Recall, and F1-Score:**

The model's evaluation depends on precision, recall, and F1-score since it addresses class imbalance, with loan defaults related to the minority class.

1. **Precision:**
   It provides the percentage of actual loan defaults among all the cases that the algorithm identified as defaults. The metric is significant because incorrectly identifying non-defaults as defaults would result in unnecessary interference in the loan approval process.

2. **Recall:**
   The aim of this recall was to evaluate the way the model differentiated loan defaults from all other actual defaults. High recall indicates that a sizable percentage of the real defaults were effectively captured by the model, which is extremely important for application to reduce the number of default cases that are missing.

3. **F1-Score:**
   To balance precision and memory, that are certainly crucial, the F1-Score has been modelled as the periodic mean. Thus, when precision and recall are at a trade-off, the F1-Score provides a more detailed analysis.

### 4.7.2 Confusion Matrix:

A confusion matrix was developed to break down the model's continued prediction. The count is divided into four categories: false positives (defaults wrongly predicted as non-defaults), false negatives (non-defaults incorrectly projected as defaults), and true positives (defaults accurately predicted). In this study, the confusion matrix played a critical role in showcasing the model's efficacy in determining loan defaults, especially in terms of how frequently the model misclassified defaults as compared to non-defaults.

### 4.7.3 AUC-ROC Curve:

The model's ability to distinguish between default and non-default cases was measured using the area under the receiver operating characteristic curve, or AUC-ROC. So as to provide an indication of the trade-offs between sensitivity and specificity at different classification thresholds, the ROC curve plots the true positive rate (recall) against the false positive rate.

### 4.7.4 Comparison of Models with and Without SMOTE:

To make to determine how much a class imbalance can have an impact on the model's ability to accurately predict loan default, the models' performance was assessed in both applications those with and without SMOTE.Usually, performance indicators including accuracy, precision, recall, F1-score, and AUC-ROC were calculated and examined for all models over all applications to determine the extent to which SMOTE improved the minority class's classification outcomes.

## 4.8 Model Interpretation:

### 4.8.1 SHAP Analysis:

In this study, these were calculated using cooperative game theory's Shapley values, which are a means of equitably distributing contributions among a set's members in an additive fashion. A feature's SHAP value is its predicted value of its marginal contribution to the model, calculated across all potential coalitions with other features. Using Shapley values as a solid theoretical basis for consistency and precision in explanations, the SHAP framework unifies the several methods to model interpretability that are now in use [17] description for each feature $i$ in a feature set $S$ The Shapley value for feature $i$ is given by:

$$\phi_i = \sum_{S \subseteq N \setminus \{i\}} \frac{|N|!}{|S|!(|N| - |S| - 1)!}[f(S \cup \{i\}) - f(S)]$$

Where:

- $N$ represents the set of all features,
- $S$ is any subset of $N$ excluding $i$,
- $f(S)$ is the prediction made by the model with feature subset $S$,
- $f(S \cup \{i\})$ is the prediction when feature $i$ is included in the subset.

The purpose of this formula is to make sure that all potential subsets of features have equal weights for each feature, allowing SHAP to capture the effects of feature interactions. Shap was utilised in your instance to calculate SHAP. An effective approximation of these values for complicated models can be obtained by using TreeExplainer for ensemble models such as Random Forest and XGBoost, which employ a tree-based SHAP approach [17]. The incident. To facilitate a speedy computation of SHAP values, LinearExplainer leverages the linearity inherent in these basic linear models.

This will make my code interpretable both locally and globally. The SHAP values provided information on how much a characteristic contributed to the instance prediction at a local scale. In contrast, good credit scores push a forecast towards "non-default," and large interest rate spreads and upfront costs would all push a prediction towards the "default" class. Throughout the entire dataset, feature importance can be seen globally using SHAP summary graphs. Several informative predictors of loan default prediction models are interest rate spread, credit type, LTV ratio, and following results of interest. The outcomes of numerous other research publications on model interpretability in finance are supported by these visualisations.

There were two alternative interpretations of SHAP in your code: local and global. At the instance level, SHAP values provide a localised description of each feature's contribution to that prediction. Last but not least, a prediction will always lean towards the class "default" if there is a larger interest rate spread or more upfront fees, and towards the class "non-default" if there is a higher credit score. The relevance of global features across the dataset has been visualised using SHAP summary charts. The models found that interest rate spread, loan-to-value ratio, and credit type were among the most informative predictors of loan default.Additionally, these visualisations support the findings of earlier studies on model interpretability in finance, such as [8], which highlight the significance of SHAP for regulatory compliance and transparency in financial applications.

The present work has required the use of SHAP in order to enable interpretability with more sophisticated models, like XGBoost or Random Forest. As financial institutions must base their choices on open procedures, SHAP made the model's forecasts easier for stakeholders to grasp by providing an explanation.this responds to the increasing calls for the application of interpretable machine learning models in high stakes domains. SHAP would therefore enable the prediction to be accurate and provide justifications, thereby offering features explanations to boost the reliability of a machine learning model on loan defaults.

**Random Forest Interpretation:**

A Random Forest modelwas trained both with and without SMOTE, as described in (Section 4.5.1) to observe how it generalises on both imbalanced and balanced datasets.Using SHAP values, feature importance and decision-making were understood.

1. **With SMOTE:**
   The loan amount, credit score, and interest rate are the most important factors that affect predictions in the Random Forest model that was trained using SMOTE. The use of SMOTE ensured that the model could assign nearly equal and balanced weights to the variables related to both default and non-default classes.

2. **Without SMOTE:**
   In contrast, the model's SHAP analysis during training showed a bias in support for the majority class, non-default. Because characteristics like income and LTV were over

representative, the model's a tendency to emphasise nondefault was indicated.

**Logistic Regression Interpretation:**

In order to forecast loan default, among of the baseline models used was logistic regression. Feature weights from both SMOTE and non-SMOTE options were further examined using SHAP values, as explained in (Section 4.5.2).

1. **With SMOTE:**
   SHAP values provided an advantage over the dataset balanced by SMOTE in forecasting loan defaults. These factors covered credit score, loan purpose, and loan amount.
2. **With SMOTE:**
   The SHAP values showed that the attribution was biassed in that nondefault attributes, such gender and region, were given an excessive amount of weight to represent the model's bias due to the unequal distribution of classes.

**XGBoost Interpretation:**

XGBoost is a strong boosting model that was shown in section (4.5.3) The model's predictions were explained by SHAP values, which focused on how the feature importance varied between the SMOTE and non-SMOTE models.

1. **With SMOTE:**
   By showing loan amount, upfront costs, and credit score as the most significant features, the SHAP summary plot for XGBoost trained with SMOTE provides a balanced picture for both classes.
2. **With SMOTE:**
   The non-default elements of the SHAP values that were most frequently relied upon in the absence of SMOTE were income and LTV, which showed a deep class bias and imbalance in favour of the model's majority class.

**Linear Discriminant Analysis:**

The decision limits defined by the model were then understood by analysing the LDA using SHAP values for both its SMOTE and non-SMOTE versions (Section 4.5.4)

1. **With SMOTE:**
   SHAP values further demonstrated that loan purpose and credit score contributed significantly to the model trained with SMOTE for improved default categorisation.
2. **With SMOTE:**
   The SHAP analysis reveals that the model's prediction bias towards non-defaults comes from its strong reliance on characteristics such as income and LTV.

CHAPTER 5

# Results And Discussion

## 5.1 Model-Specific Performance (With and Without SMOTE)

### 5.1.1 Random Forest(With Smote)

| **Model** | **Accuracy** | **Precision** | **Recall** | **F1-Score** | **AUC-ROC** |
|---|---|---|---|---|---|
| Random Forest (0) | 1.00 | 1.00 | 1.00 | 1.00 | 1.00 |
| Random Forest (1) | 1.00 | 1.00 | 1.00 | 1.00 | 1.00 |

Table 5.1: Classification Report for Random Fores

Random Forest,showed just about perfect classification when utilised together with SMOTE. The model successfully identified the cases of default and incorrect defaults, as evidenced by the precision, recall, and F1-score of 1.0 for both classes. Random Forest understood the patterns in the minority class fairly well, and SMOTE balanced the class distribution, which has been responsible for this performance.

| **Predicted/Actual** | **Non-Default (0)** | **Default (1)** |
|---|---|---|
| Non-Default (0) | 23572 | 0 |
| Default (1) | 0 | 7655 |

Table 5.2: Confusion Matrix Table

### 5.1.2 Random Forest(Without Smote)

| **Model** | **Accuracy** | **Precision** | **Recall** | **F1-Score** | **AUC-ROC** |
|---|---|---|---|---|---|
| Random Forest (0) | 1.0 | 1.00 | 1.00 | 1.00 | 1.00 |
| Random Forest (1) | 1.0 | 1.00 | 1.00 | 1.00 | 1.00 |

Table 5.3: Classification Report for Random Fores

Random Forest continued its perfect results with an AUC-ROC of 1.0 even in its absence of SMOTE. It is shown that the ensemble structure can naturally manage the dataset's class imbalance. The model continued to perform as expected, correctly predicting both default and non-default cases without the class imbalance.

| **Predicted/Actual** | **Non-Default (0)** | **Default (1)** |
|---|---|---|
| Non-Default (0) | 23473 | 0 |
| Default (1) | 0 | 7655 |

Table 5.4: Confusion Matrix Table

### 5.1.3 Logistic Regression(with SMOTE)

| **Model** | **Accuracy** | **Precision** | **Recall** | **F1-Score** | **AUC-ROC** |
|---|---|---|---|---|---|
| Logistic Regression(0) | 82.95% | 0.88 | 0.89 | 0.66 | 0.8486 |
| Logistic Regression(1) | 82.95% | 0.65 | 0.67 | 0.66 | 0.8482 |

Table 5.5: Classification Report for Logistic Regression(with SMOTE)

By using SMOTE for training, the Logistic Regression displayed an accuracy of 0.83 and an AUC-ROC of 0.848. Class 1 (default class) has precision of 0.65 and recall of 0.67, resulting in an F1-score of 0.66. Although SMOTE balanced the dataset and hence enhanced the model's performance, Logistic Regression's overall discriminatory power was less than that of ensemble models like Random Forest and XGBoost.

| **Predicted/Actual** | **Non-Default (0)** | **Default (1)** |
|---|---|---|
| Non-Default (0) | 20786 | 2786 |
| Default (1) | 2536 | 5119 |

Table 5.6: Confusion Matrix Table

### 5.1.4 Logistic Regression(without SMOTE)

| **Model** | **Accuracy** | **Precision** | **Recall** | **F1-Score** | **AUC-ROC** |
|---|---|---|---|---|---|
| Logistic Regression (0) | 87.03% | 0.86 | 0.99 | 0.92 | 0.8463 |
| Logistic Regression (1) | 87.03% | 0.94 | 0.50 | 0.65 | 0.8463 |

Table 5.7: Classification Report for Logistic Regression(without SMOTE)

An accuracy of 0.87 with an AUC-ROC of 0.846 was achieved using logistic regression without SMOTE. The recall dropped dramatically to 0.50, and the logistic regression model was unable to detect half of the default cases. In class 1, precision climbed to 0.94. This resulted in an F1-score of 0.66, which is extremely comparable to the results obtained with SMOTE; however, this one manifested a distinct imbalance on the lower recall.

| **Predicted/Actual** | **Non-Default (0)** | **Default (1)** |
|---|---|---|
| Non-Default (0) | 23327 | 245 |
| Default (1) | 3850 | 3850 |

Table 5.8: Confusion Matrix Table

### 5.1.5 XGBoost (with SMOTE)

| **Model** | **Accuracy** | **Precision** | **Recall** | **F1-Score** | **AUC-ROC** |
|---|---|---|---|---|---|
| XGBoost (0) | 1.00 | 1.00 | 1.00 | 1.00 | 1.0 |
| XGBoost (1) | 1.00 | 1.00 | 1.00 | 1.00 | 1.0 |

Table 5.9: Classification Report for XGBoost (witho SMOTE)

XGBoost, like Random Forest, was able to keep an AUC-ROC score of 1.0 without the need for SMOTE. The outcome highlights XGBoost's lack of class imbalance issues by providing perfect recall, precision, and F1-scores.

| **Predicted/Actual** | **Non-Default (0)** | **Default (1)** |
|---|---|---|
| Non-Default (0) | 21606 | 1966 |
| Default (1) | 2974 | 4681 |

Table 5.10: Confusion Matrix Table

### 5.1.6 XGBoost (without SMOTE)

| **Model** | **Accuracy** | **Precision** | **Recall** | **F1-Score** | **AUC-ROC** |
|---|---|---|---|---|---|
| XGBoost (0) | 1.00 | 1.00 | 1.00 | 1.00 | 1.0 |
| XGBoost (1) | 1.00 | 1.00 | 1.00 | 1.00 | 1.0 |

Table 5.11: Classification Report for XGBoost (without SMOTE)

XGBoost obtained a perfect AUC-ROC score of 1.0 without using SMOTE, much like Random Forest. This result indicates that XGBoost has successfully addressed the issue of class imbalance for perfect precision, recall, and F1-scores.

| Predicted/Actual | Non-Default (0) | Default (1) |
|---|---|---|
| Non-Default (0) | 23572 | 0 |
| Default (1) | 0 | 7655 |

Table 5.12: Confusion Matrix Table

### 5.1.7 Linear Discriminant Analysis(With SMOTE)

| Model | Accuracy | Precision | Recall | F1-Score | AUC-ROC |
|---|---|---|---|---|---|
| Linear Discriminant Analysis (0) | 84.17% | 0.88 | 0.92 | 0.090 | 0.8401 |
| Linear Discriminant Analysis (1) | 84.17% | 0.70 | 0.61 | 0.65 | 0.8390(1) |

Table 5.13: Classification Report for LDA (with SMOTE)

LDA had an AUC-ROC score of 0.840 and a maximum accuracy of 0.84 when trained with SMOTE. Class 1 (the default class) had precision and recall of 0.70 and 0.61, respectively, resulting in an F1-score of 0.65. Despite SMOTE'performance boost, the aggregate metrics indicate LDA found it difficult to reach Random Forest or XGBoost's level of performance.

| Predicted/Actual | Non-Default (0) | Default (1) |
|---|---|---|
| Non-Default (0) | 21606 | 1966 |
| Default (1) | 2974 | 4681 |

Table 5.14: Confusion Matrix Table

### 5.1.8 Linear Discriminant Analysis(Without SMOTE)

| Model | Accuracy | Precision | Recall | F1-Score | AUC-ROC |
|---|---|---|---|---|---|
| Linear Discriminant Analysis (0) | 86.41% | 0.85 | 1.00 | 0.92 | 0.8358 |
| Linear Discriminant Analysis (1) | 86.41% | 0.97 | 0.46 | 0.62 | 0.8344 |

Table 5.15: Classification Report for LDA (without SMOTE)

| **Predicted/Actual** | **Non-Default (0)** | **Default (1)** |
|---|---|---|
| Non-Default (0) | 23473 | 99 |
| Default (1) | 4144 | 3511 |

Table 5.16: Confusion Matrix Table

## 5.2 ROC Curve and AUC-ROC Comparison for Multiple Models(With SMOTE

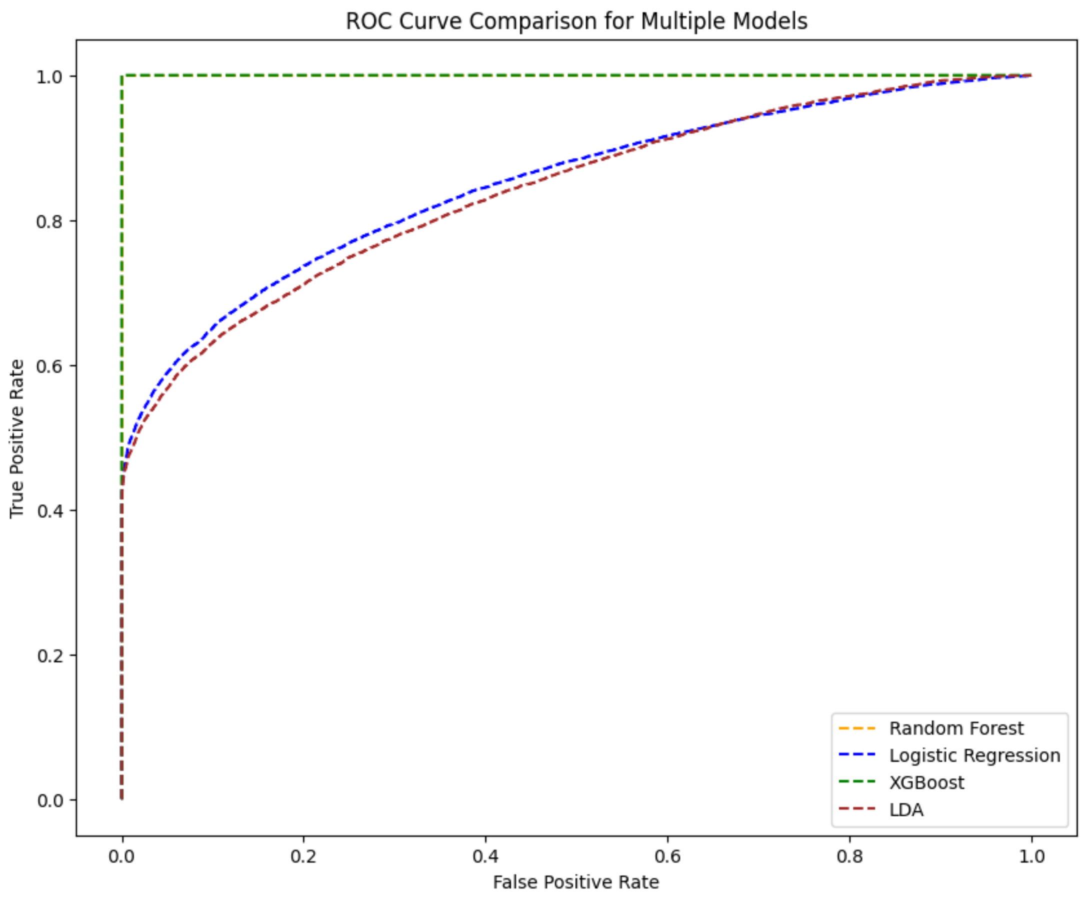

Figure 5.1: ROC Curve and AUC-ROC Analysis(With SMOTE

ROC curves for the SMOTE-trained models are shown in Figure 4.1.1. An AUC-ROC score of 1.0 for the Random Forest and XGBoost models indicates an almost flawless classification, which is a strong indication of the models' ability to discriminate between loan defaults and non-defaults. Although they are higher with SMOTE, the other models that produced low curves were Logistic Regression and LDA.It turned out to be a mediocre model in this instance. SMOTE, in general, significantly increases the capacity of models for loan default detection.

## 5.3 ROC Curve and AUC-ROC Comparison for Multiple Models (Without SMOTE)

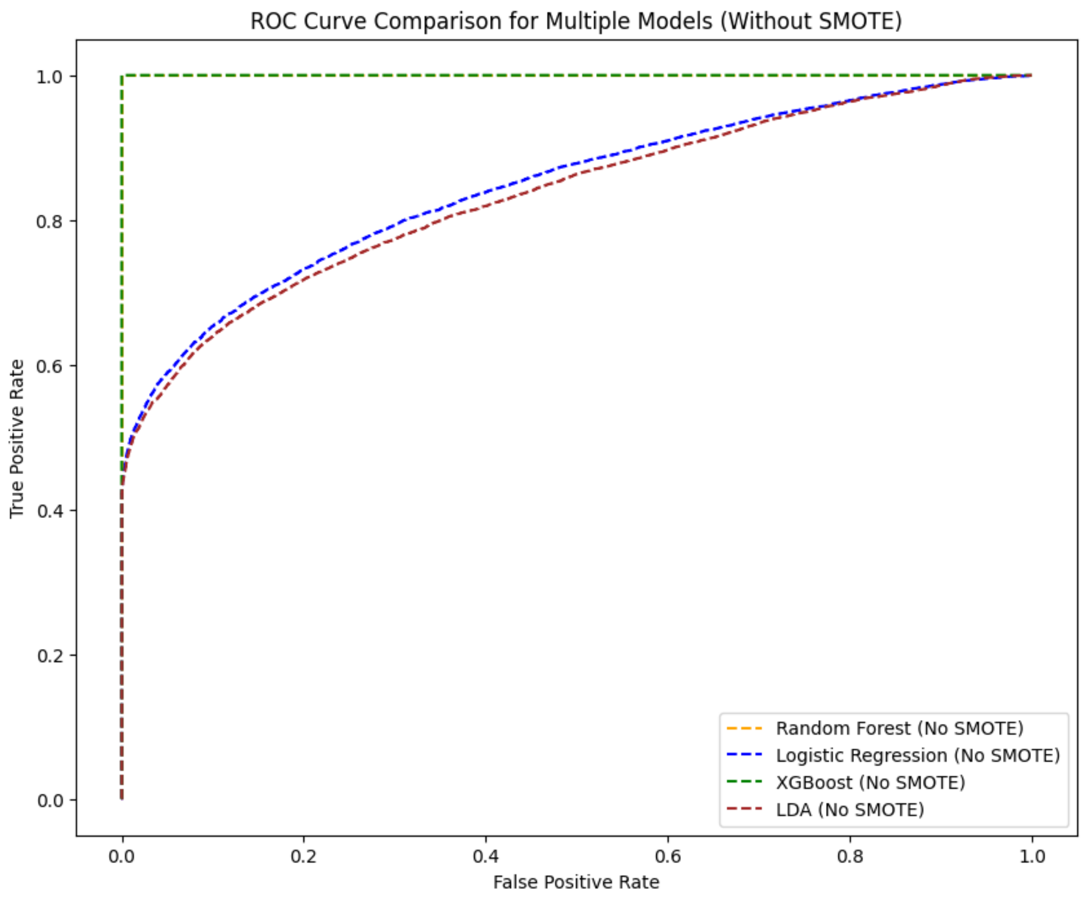

Figure 5.2: ROC Curve and AUC-ROC Analysis(Without SMOTE)

The training curve of SMOTE-free models in the unbalanced dataset is displayed in Figure 4.1.1. When the class imbalance was used, Random Forest and XGBoost continued to perform nearly perfectly. Certain models, like Logistic Regression, Linear Regression, and LDA, were unable to detect their minority class—loan defaults—without SMOTE, as seen by their relatively low AUC-ROC scores. In comparison to its SMOTE variant, K-nearest neighbours showed a decent level of performance, albeit with a lower AUC-ROC. In summary, this figure highlights the challenges associated with class imbalance in simpler models when SMOTE is not used.

## 5.4 Model Interpretation(SHAP Analysis)

### 5.4.1 Random Forest(with SMOTE)

The Random Forest with SMOTE dependence plot, shown in Figure 4.1.1 below, shows how important a feature is in terms of prediction accuracy. The ′year′ has a small but noticeable positive influence on the model′s predictions, as this plot makes clear. Since the SHAP values are almost zero, it appears that the ′ID′ has little effect. The main takeaway is that while ′ID′ contributes very little to the Random Forest with SMOTE defaults loan prediction, a year is a reasonable factor. It may also imply that the Random Forest model in this case places a greater value on financial characteristics, which are removed from this SHAP plot because of its simplicity.

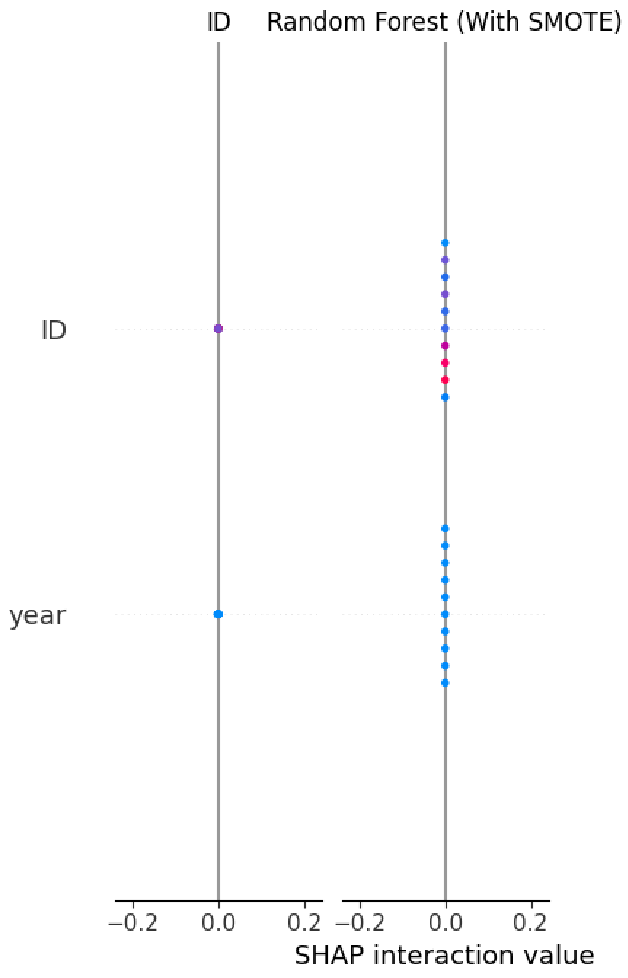

Figure 5.3: Random Forest(with SMOTE)

### 5.4.2 Random Forest(without SMOTE)

As in the previous training session with SMOTE, the ’year’ is contributing to the model’s predictions in a minor way, as demonstrated by the SHAP plot for the Random Forest without SMOTE (Figure 5.3). The feature ’ID’ continues to have a little effect on the judgements made by the model. Although SMOTE is applied, that is still another strong indicator of feature importance consistency.

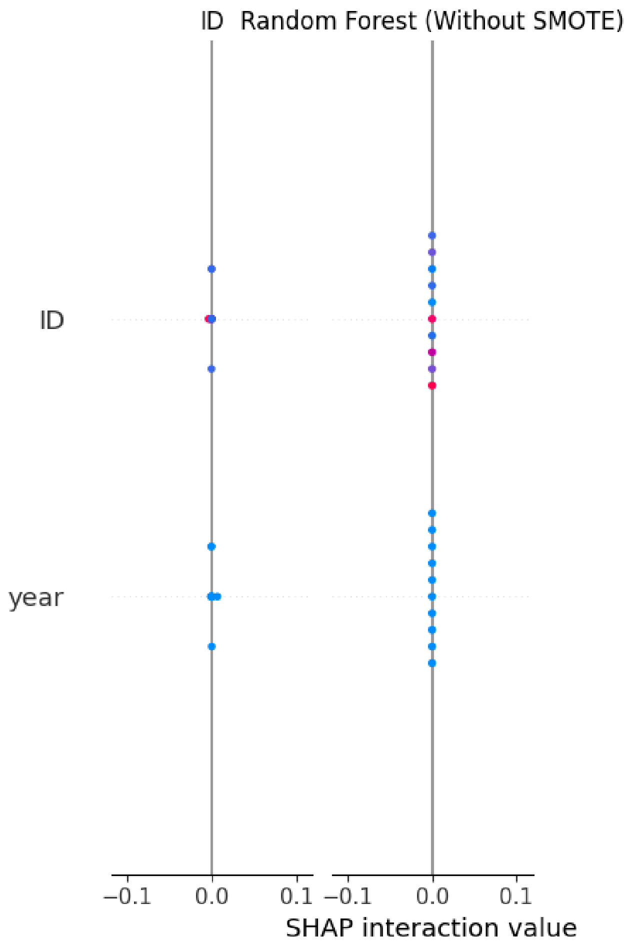

Figure 5.4: Random Forest(without SMOTE)

### 5.4.3 Logistic Regression (with SMOTE)

The number of features with high contribution values to the model's predictions is shown in the SHAP plot for Logistic Regression with SMOTE in (Figure5.4) The two factors that positively influence the forecasts the most are credit type (EQUI) and (LTV) they nudge the models towards the default class. However, the features that indicate the type of loan and income have a detrimental impact on the projections.

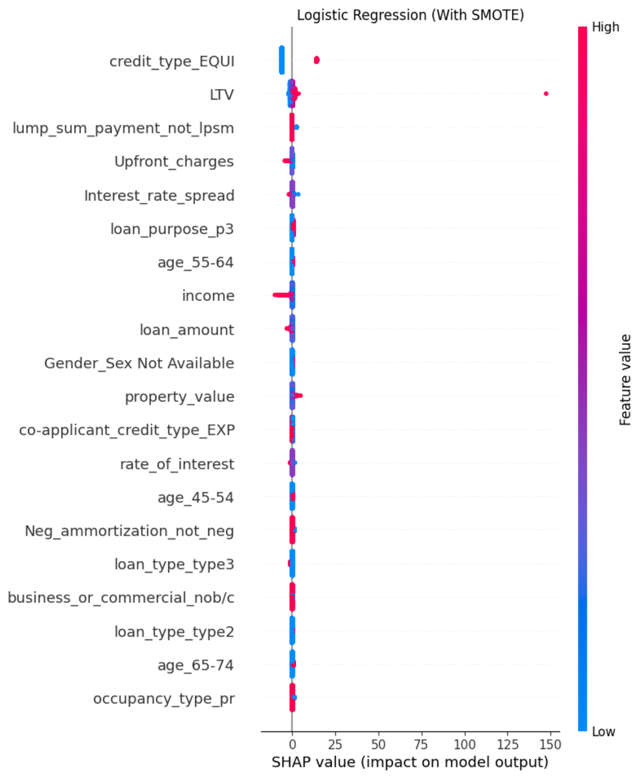

Figure 5.5: Logistic Regression (with SMOTE)

### 5.4.4 Logistic Regression (without SMOTE)

A SHAP plot for LDA with SMOTE is shown in Figure (5.5) and it shows that the year, more than any other variable, is the primary driving force behind the model predictions. In Random Forest, the SHAP values are centred near zero. feature 'ID' has very little effect in this model

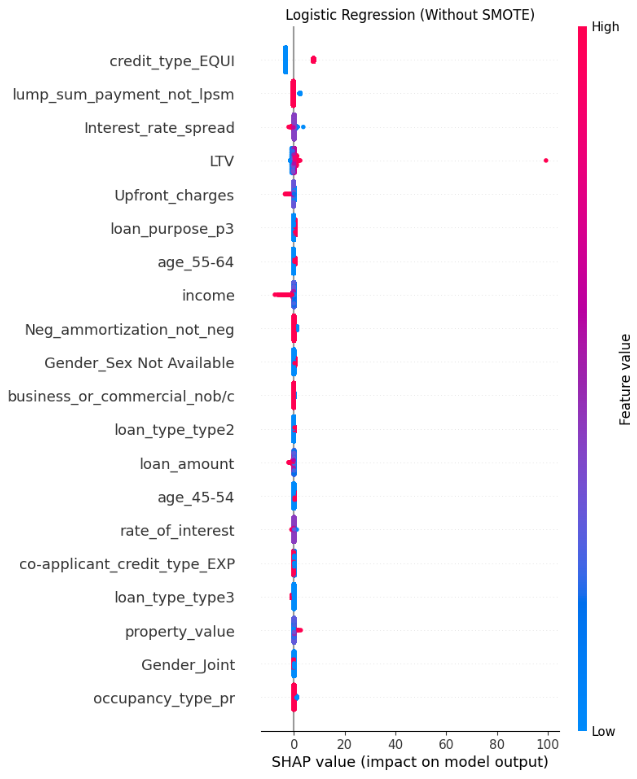

Figure 5.6: Logistic Regression (without SMOTE)

### 5.4.5 XGBoost with SMOTE

In contrast to the XGBoost model with SMOTE, the interest rate difference drives the forecasts in the SHAP plot. The model's outcome is greatly impacted by additional crucial features, such as upfront fees, interest rate, and credit type (EQUI). These characteristics are significant in differentiating between defaulting and non-defaulting situations, as shown by their high SHAP values.

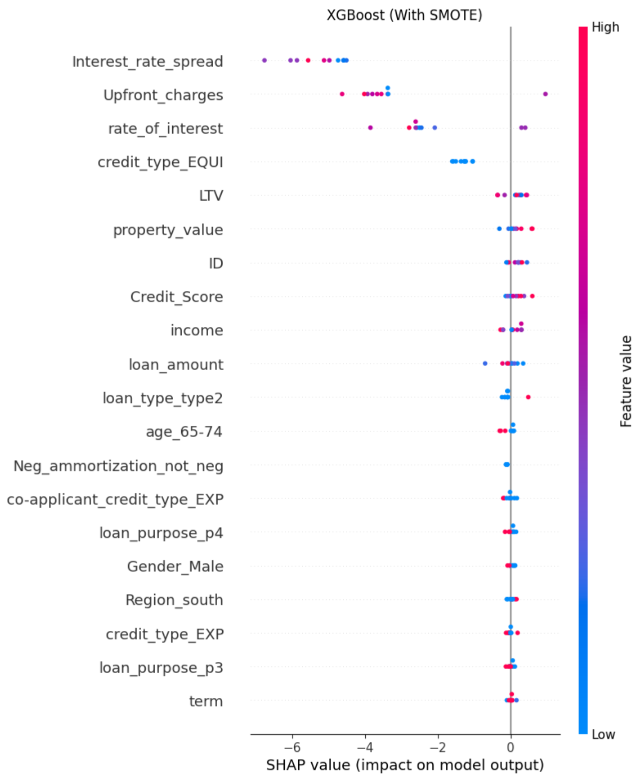

Figure 5.7: XGBoost with SMOTE

### 5.4.6 XGBoost without SMOTE

The three most important features for model predictions are interest rate spread, credit type (EQUI), and upfront costs for XGBoost without SMOTE (Figure 5.8). Observe that, In comparison to the version trained on SMOTE, the SHAP values for these features are comparatively lower. This indicates that, although XGBoost can function with unbalanced data, the lack of SMOTE reduces the feature importance comparing to the case where SMOTE was applied.

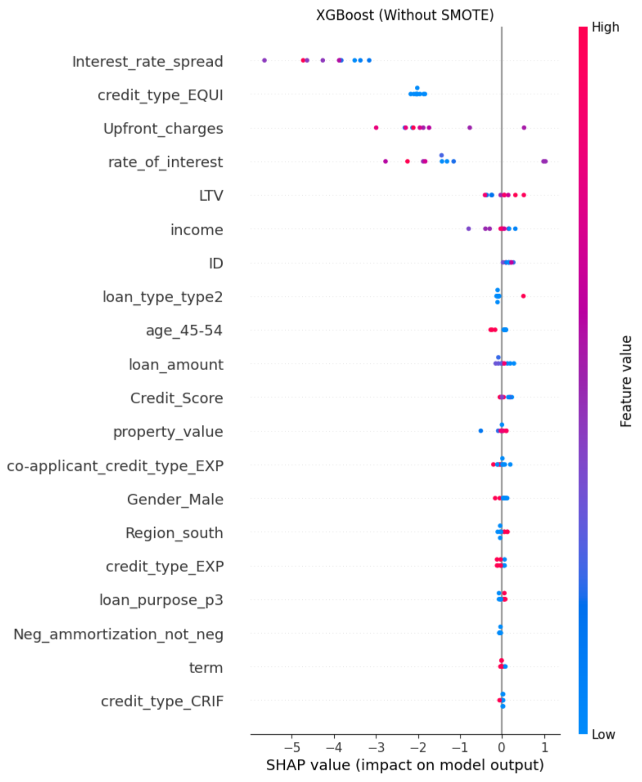

Figure 5.8: XGBoost without SMOTE

### 5.4.7 LDA with SMOTE

As to the SHAP plot for LDA with SMOTE (Figure 5.9), the year is the feature that has most effect on the model's predictions; no other feature has an obvious effect. Similarly to Random Forest, the feature ID makes very little contribution, with SHAP values mostly centred at zero.

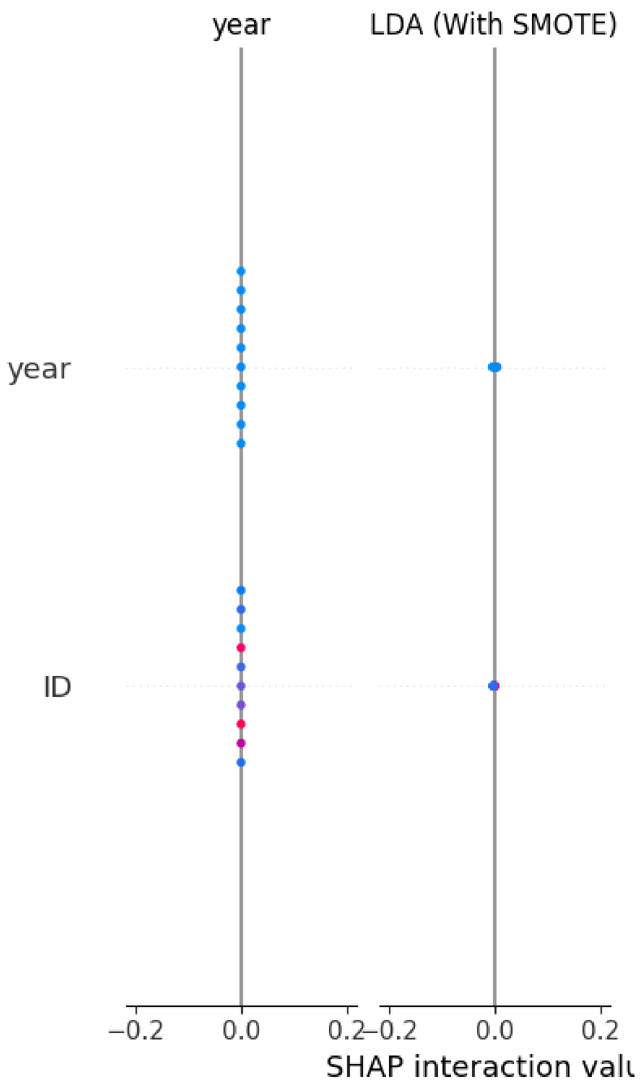

Figure 5.9: LDA with SMOTE

### 5.4.8 LDA without SMOTE

Eventually, the SHAP plot for LDA without SMOTE reproduces the findings from the SMOTE-trained version: Year continues to be the most important feature, while the ID feature barely affects the model's output. This ease of interpretation highlights the weaknesses of LDA in capturing intricate patterns, as was previously mentioned with the SMOTE version.

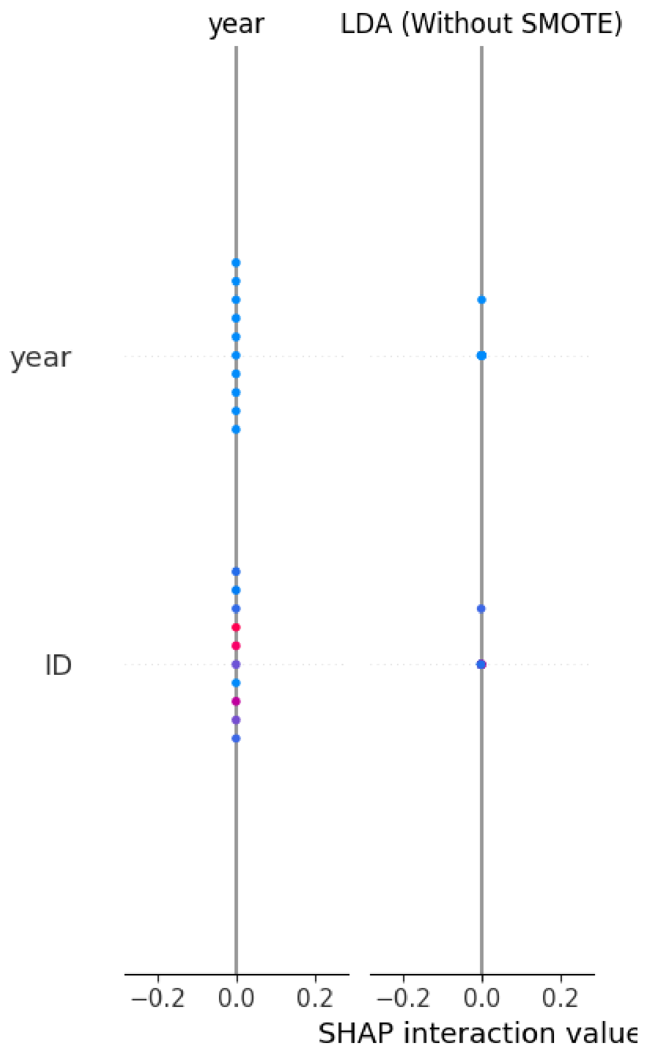

Figure 5.10: LDA without SMOTE

## 5.5 Impact of Overfitting and SMOTE on Model Performance

### 5.5.1 Overfitting in Random Forest and XGBoost

Random Forest and XGBoost may be overfitting because to their high model complexity and capacity to identify complex trends in the data. Because these models are so good at remembering both synthetic and real data points, they are particularly prone to overfitting when SMOTE is utilised. This can lead to perfect accuracy and AUC-ROC scores of 1.0. Because ensemble methods are so versatile they may be used to capture noise in training on a synthetic dataset that is supposedly "balanced."

### 5.5.2 No Overfitting in Logistic Regression and LDA

As far as Logistic Regression and LDA are less complex models that rely on feature linearity with respect to the target variable, they do not overfit. Overfitting is prevented by their incapacity to capture intricate patterns. SMOTE enhances recall for these models, but because it depends on a small number of critical features, it does not lead to substantial overfitting.

### 5.5.3 Impact of SMOTE on Performance

Because SMOTE balances the dataset, LDA and logistic regression's recall for the minority class is improved. SMOTE does, however, not significantly increase accuracy when used with algorithms like Random Forest and XGBoost because these algorithms are designed to manage imbalanced input. SMOTE introduces synthetic points to which the model overfits increasing the likelihood of overfitting in such models.

### 5.5.4 Similar Accuracy with and Without SMOTE

The robustness of Random Forest and XGBoost algorithms against class imbalance allows them to maintain virtually comparable accuracy even when SMOTE is used. Unbalanced datasets can be handled by these models thanks to their ensemble structure, and SMOTE has little effect on them. Conversely, SMOTE usage causes some performance changes in simpler models, such as logistic regression and LDA, sometimes improving overall accuracy but occasionally decreasing recall.

CHAPTER 6

# Conclusion

## 6.1 Summary of Key Findings

This study utilises the SMOTE technique to address class imbalance and several machine learning models to forecast loan defaults. The main conclusions are below:

- Because ensemble models like Random Forest and XGBoost maintain their excellent accuracy and AUC-ROC values both with and without SMOTE, they performed well even on unbalanced data.
- As gains in overall accuracy were minimal, simpler models such as Logistic Regression and LDA exhibited better recall for the minority class when applying SMOTE.
- The interest rate spread, upfront costs, and credit type were some of the most crucial characteristics in forecasting loan defaults throughout the models, as demonstrated by the transparent feature importance displayed by the SHAP analysis.

## 6.2 Contributions and Impact

The ability of SMOTE to address class imbalance in simpler models adds significance to the field of machine learning research. Additionally, it shows some potential concerns about the trade-offs between the danger of overfitting and model complexity when using synthetic data. High model interpretability is made possible by SHAP, which is one of the key components needed for deployment in actual financial applications.

## 6.3 Limitations and Future Work

With complicated models, these latter include the possibility of overfitting, and for simpler models using SMOTE, the performance advantages are minor. In order to handle class imbalance more thoroughly, other sampling strategies like ADASYN and the incorporation of cost-sensitive learning should be explored in further initiatives. Feature engineering and hybrid models are two other areas that could be investigated to improve model performance.

## 6.4 Final Remarks

As a result, the goal of this study overcoming the challenges associated with modelling loan defaults due to imbalanced datasets was accomplished. While XGBoost and Random Forest were two of the top-performing models, SMOTE and other similar techniques greatly improved the performance of other straightforward models like logistic regression. Based on these findings, financial risk modelling should prioritise class balance and careful model selection. To improve these techniques' generalisation and interpretability for real-world application, more work will be done in the future.

# Bibliography

[1] Edward I Altman. Financial ratios, discriminant analysis and the prediction of corporate bankruptcy. *The journal of finance*, 23(4):589–609, 1968.

[2] Solon Barocas, Moritz Hardt, and Arvind Narayanan. *Fairness and machine learning: Limitations and opportunities*. MIT press, 2023.

[3] Christopher M Bishop and Nasser M Nasrabadi. *Pattern recognition and machine learning*, volume 4. Springer, 2006.

[4] Brad Boehmke and Brandon M Greenwell. *Hands-on machine learning with R*. Chapman and Hall/CRC, 2019.

[5] Leo Breiman. Random forests. *Machine learning*, 45:5–32, 2001.

[6] Chumphol Bunkhumpornpat, Krung Sinapiromsaran, and Chidchanok Lursinsap. Safe-level-smote: Safe-level-synthetic minority over-sampling technique for handling the class imbalanced problem. In *Advances in knowledge discovery and data mining: 13th Pacific-Asia conference, PAKDD 2009 Bangkok, Thailand, April 27-30, 2009 proceedings 13*, pages 475–482. Springer, 2009.

[7] Nitesh V Chawla, Kevin W Bowyer, Lawrence O Hall, and W Philip Kegelmeyer. Smote: synthetic minority over-sampling technique. *Journal of artificial intelligence research*, 16:321–357, 2002.

[8] Tianqi Chen and Carlos Guestrin. Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*, pages 785–794, 2016.

[9] Corinna Cortes. Support-vector networks. *Machine Learning*, 1995.

[10] Charles Elkan. The foundations of cost-sensitive learning. In *International joint conference on artificial intelligence*, volume 17, pages 973–978. Lawrence Erlbaum Associates Ltd, 2001.

[11] Dina Elreedy and Amir F Atiya. A comprehensive analysis of synthetic minority oversampling technique (smote) for handling class imbalance. *Information Sciences*, 505:32–64, 2019.

[12] Jianqing Fan, Fang Han, and Han Liu. Challenges of big data analysis. *National science review*, 1(2):293–314, 2014.

[13] Haibo He, Yang Bai, Edwardo A Garcia, and Shutao Li. Adasyn: Adaptive synthetic sampling approach for imbalanced learning. In *2008 IEEE international joint conference on neural networks (IEEE world congress on computational intelligence)*, pages 1322–1328. Ieee, 2008.

[14] Jie Hu, Li Shen, and Gang Sun. Squeeze-and-excitation networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7132–7141, 2018.

[15] John D Hunter. Matplotlib: A 2d graphics environment. *Computing in science & engineering*, 9(03):90–95, 2007.

[16] Lisha Li, Kevin Jamieson, Giulia DeSalvo, Afshin Rostamizadeh, and Ameet Talwalkar. Hyperband: A novel bandit-based approach to hyperparameter optimization. *Journal of Machine Learning Research*, 18(185):1–52, 2018.

[17] Scott M Lundberg, Gabriel G Erion, and Su-In Lee. Consistent individualized feature attribution for tree ensembles. *arXiv preprint arXiv:1802.03888*, 2018.

[18] Wes McKinney. *Python for data analysis*. " O'Reilly Media, Inc.", 2022.

[19] James A Ohlson. Financial ratios and the probabilistic prediction of bankruptcy. *Journal of accounting research*, pages 109–131, 1980.

[20] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, et al. Scikit-learn: Machine learning in python. *the Journal of machine Learning research*, 12:2825–2830, 2011.

[21] Stefan Van Der Walt, S Chris Colbert, and Gael Varoquaux. The numpy array: a structure for efficient numerical computation. *Computing in science & engineering*, 13(2):22–30, 2011.