Credit Scoring

Liangxiao Li, Troy Campbell, Asiyah Sharif, Jemma Moseley, Binghan Shang

March 14, 2024

Abstract

This report includes the methodology and results for methods used to predict default probability of a default on a loan using 10 attributes for a given individual. Having read this report, the reader should be able to understand and apply the models used to predict default probability given the correct resources, as well as understand which attributes are more useful in such predictions.

Contents

1 Introduction **3**

2 Exploratory Data Analysis **4**
2.1 Data Description . . . 4
2.2 Missing Values and Data Imbalance . . . 4
2.3 Outliers . . . 5
2.4 Correlation of Variables . . . 6

3 Data Pre-processing **6**
3.1 Data Cleaning . . . 6
3.2 Analysis of Cleaned Data . . . 7
3.2.1 Outliers . . . 8
3.2.2 Correlation . . . 8
3.3 Imputation . . . 9
3.3.1 Mean and Median Imputation . . . 9
3.3.2 GBDT and XGB Imputation . . . 9
3.3.3 Imputation Result Comparison . . . 12
3.4 PCA . . . 12

4 Evaluation Metrics **14**

5 Logistic Regression Model **16**
5.1 Methods . . . 16
5.2 Results . . . 18
5.2.1 Model weights . . . 18
5.2.2 Accuracy . . . 19
5.2.3 Confusion Matrix and AUC ROC . . . 19

6 Random Forest **20**
6.1 Methods . . . 20
6.2 Results . . . 21
6.2.1 Accuracy . . . 21
6.2.2 Confusion Matrix and AUC ROC . . . 21

7 KNN **22**
7.1 Methods . . . 22
7.1.1 The Nearest Neighbor Rule . . . 22
7.1.2 The *kth* Nearest Neighbour Rule . . . 23
7.2 Results . . . 23
7.2.1 Accuracy . . . 23
7.2.2 Confusion Matrix and AUC ROC . . . 23

8 Boosting Methods **24**
8.1 Methods . . . 24
8.1.1 Ada Boost Gradient Boosting . . . 24
8.2 Results . . . 26
8.2.1 Accuracy . . . 26
8.2.2 Confusion Matrix and AUC ROC . . . 26

9 Model Evaluation **28**
9.1 Model Comparison: Accuracy . . . 28
9.2 Model Comparison: Confusion matrix and AUC-ROC curve . . . 29

10 Further Work **29**
10.1 Imbalanced Dataset . . . 29
10.2 Risk of Loans . . . 30

11 Conclusion **30**

12 Appendices **32**
12.1 Pseudocode and Descent direction of the Gradient Method/Newton's method . . . 32
12.2 The convergence of Gradient Descent . . . 33
12.3 The convergence of Newton's Method . . . 33
12.4 Gradient Descent Convergence and Learning Rate . . . 34
12.5 Regularization . . . 34
12.6 Cross Validation . . . 34
12.7 The convergence of the NN Method . . . 35

1 Introduction

In this report, we investigate the development of a statistical model to accurately predict the probability of a default on a loan. The development of the model is based on data that is available to credit lenders and we aim to identify which attributes are more useful in such predictions and discuss implications in practice.

Default probability is the probability or likelihood that a borrower is unable to pay back a debt within a given period of time (Kenton 2023). Default probability for an individual links to their credit score, and lenders subsequently use this information to decide whether or not to provide a loan. We note, however, that not every lender uses the same method to predict credit scores – in this way it is not standardised.

The importance of default probabilities, and therefore credit scores, in the current climate is large for lenders. In a recent article, the Guardian states that, in 2024, 'Britain's biggest high street lenders expect the sharpest rise in defaults on unsecured lending since 2009' (Partington 2024). The cost-of-living crisis and increased interest rates have led to an increase in default on loans in the recent two years, with predicted further increase to come, and so this is an area of significant interest for lenders. Given the importance at this moment in time, this report investigates an accurate way to predict the default on a loan, and through that provide reassurance to lenders when using the model to determine whether to provide a loan to an individual given certain attributes. We aim to make a difference in this area and help lenders through the current and predicted uncertain times.

Credit scoring is a well-researched and highly informed subject area. Within it, logistic regression is one of the most used methods to determine 'risk classes' for individuals (Kraus 2014). Kraus discusses several reasons as to why logistic regression is a popular method, including its 'good interpretability', as well as the ability to implement statistical software to perform logistic regression in an efficient way. We note also, however, limitations discussed by Kraus, such as assumption of linearity in the covariates, and potential alternatives solutions through machine learning are discussed. Initial research led to the discovery of an article on the use of logistic regression to predict default probabilities (Zhang 2020). As part of the 'International Journal of Information, Business and Management', the article not only provides confidence in the use of a logistic regression model, but also includes information about model validation and states the importance of cleaning the data. This article was a valuable starting point for the task at hand.

The report begins with Section 2, Exploratory Data Analysis, followed by Section 3, Data Preprocessing. These sections of the report contain detailed information regarding the dataset and how it was used for this task. The dataset, made up of 150,000 entries and 10 feature variables, was observed and before progress could be made with models, there were a number of tasks that were completed, such as dealing with outliers, imputation and PCA. These sections aim to describe the data and modifications that were made to it, with hopes of further setting the scene and allowing deeper understanding.

This report includes detailed methodology and results for all models used to predict default probability. Having defined important notation and metrics in Section 4, the implementation of a logistic regression model for predicting the default probability is discussed in Section 4. A summary of the logistic regression model is given and having fitted the model in python, the results are discussed. Following this, the question 'Which attributes are more useful in predicting default probability?' will be answered.

Also included, in Section 6, is the Random Forest model which was used for comparison. This model uses a number of principles such as bootstrapping and aggregation. For the third model used, Section 7 discusses the approach of 'K-Nearest Neighbours', which is a non-parametric machine-learning algorithm technique. Finally, the 'ensemble method' approach is discussed, namely through Adaboost and Gradient Boosting. This involves combining different weak learners with the aim of establishing better performance. Details of each model, their methodologies and subsequent results, are given in the relevant sections.

Having completed the analysis of the models, they are evaluated in Section 9. In this section, comparisons are drawn between models to establish which model is most useful in predicting default probabilities. Multiple evaluation metrics are determined and compared, with different metrics being more useful depending on the aim of the lender. Having interpreted results, different aims of lenders are considered and through this, recommendations for models are given. Following this, a detailed conclusion is given. The conclusion will summarise the main elements of the report and relate our findings back to the original task. Also included in the conclusion are any potential limitations of the results and suggestions for further work. References and appendices are included as relevant in Section 12 and the Reference section at the end of the report.

2 Exploratory Data Analysis

This section gives an overview of the initial dataset provided and important observations that set the foundations for our model. This part ensured that errors with the dataset were not overlooked and the data was in an appropriate format for our models and their assumptions.

2.1 Data Description

The initial dataset provided had 150,000 entries and 10 feature variables and one binary response variable (SeriousDlqin2yrs), each given below along with descriptions in Table 1.

Table 1: Variable Names and Descriptions

Variable Name	**Description**
SeriousDlqin2yrs	Person experienced 90 days past due delinquency or worse
RevolvingUtilizationOfUnsecuredLines	Total balance on credit cards and personal lines of credit except real estate and no installment debt like car loans divided by the sum of credit limits
age	Age of borrower in years
NumberOfTime30-59DaysPastDueNotWorse	Number of times borrower has been 30-59 days past due but no worse in the last 2 years
DebtRatio	Monthly debt payments, alimony,living costs divided by monthy gross income
MonthlyIncome	Monthly income
NumberOfOpenCreditLinesAndLoans	Number of Open loans (installment like car loan or mortgage) and Lines of credit (e.g. credit cards)
NumberOfTimes90DaysLate	Number of times borrower has been 90 days or more past due
NumberRealEstateLoansOrLines	Number of mortgage and real estate loans including home equity lines of credit
NumberOfTime60-89DaysPastDueNotWorse	Number of times borrower has been 60-89 days past due but no worse in the last 2 years
NumberOfDependents	Number of dependents in family excluding themselves (spouse, children etc.)

2.2 Missing Values and Data Imbalance

Upon initial inspection of the dataset, it was found that two of the variables contained missing values denoted by NA (Not Available). Specifically:

- NumberOfDependents contained 3,924 NA entries
- MonthlyIncome contained 29,731 NA entries

The proportion of missing values for Monthly Income was 19.82%, this showed that imputation would be necessary, and that it is not sufficient to remove all missing values from the dataset as this would considerably reduce the size.

Additionally, to fulfill our intention of using logistic regression, the importance of dealing with these missing values was evident because logistic regression cannot have missing values. So, having noted which variables had missing entries, potential methods of imputation began to be considered. Further details for the implementation of imputation are given in Section 3.2.

Initial inspection also showed an imbalance in the dataset. It was noted that the number of entries with a default (i.e. SeriousDlqin2yrs=1) compared to entries without a default (i.e. SeriousDlqin2yrs=0), was much lower. Specifically:

- Default entries: 10,026
- Non-default entries: 139,974

2.3 Outliers

When exploring our dataset, we also investigated any potential outliers. Investigating and removing outliers from the data before creating a model is crucial because outliers can distort analyses and therefore affect model performance. By eliminating outliers, we ensure that the model is trained on representative data, leading to more accurate predictions. To determine potential outliers we plotted box plots for each feature variable. This enabled further inspection of the data and was necessary to establish whether intervention was needed for these extreme values. In the figure below, we have shown the most interesting variable boxplots.

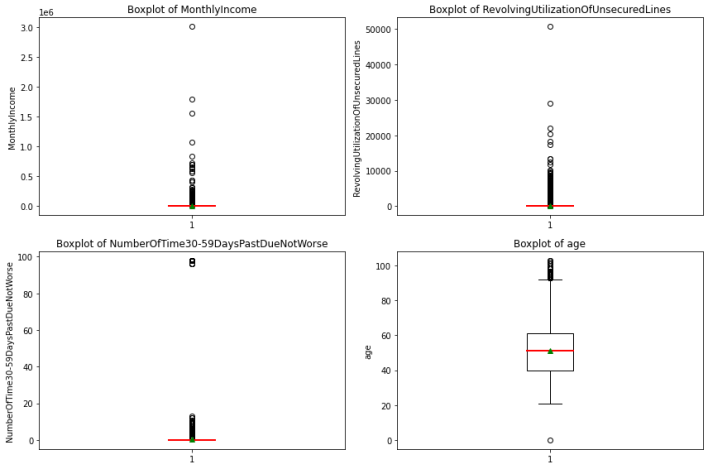

Figure 1: Boxplots

The visualisation of the four boxplots provides insight into the distribution of the variables. It can be observed that the Monthly Income and past-due variables (NumberOfTimes90DaysLate, NumberOfTime60-89DaysPastDueNotWorse, NumberOfTime30-59DaysPastDueNotWorse) have extreme point outliers, this suggests that these points might be highly influential to the model and therefore should be removed. It should be noted that on the Monthly Income boxplot, there is a $'1e6'$ at the top of the boxplot, this happens when there are extreme outliers in the dataset, causing the axis to scale to a very large range. In the RevolvingUtilizationofUnsecuredLinesboxplot we can see that the outliers are more spread out but all the outliers greater than 1000 should still be removed. Lastly, for the Age variable none of the outliers look extreme therefore these outliers will not be considered necessary to remove.

2.4 Correlation of Variables

Another step taken was to inspect the correlation between variables in the dataset. This is important because many of the models we will use later assume that all the variables are uncorrelated.

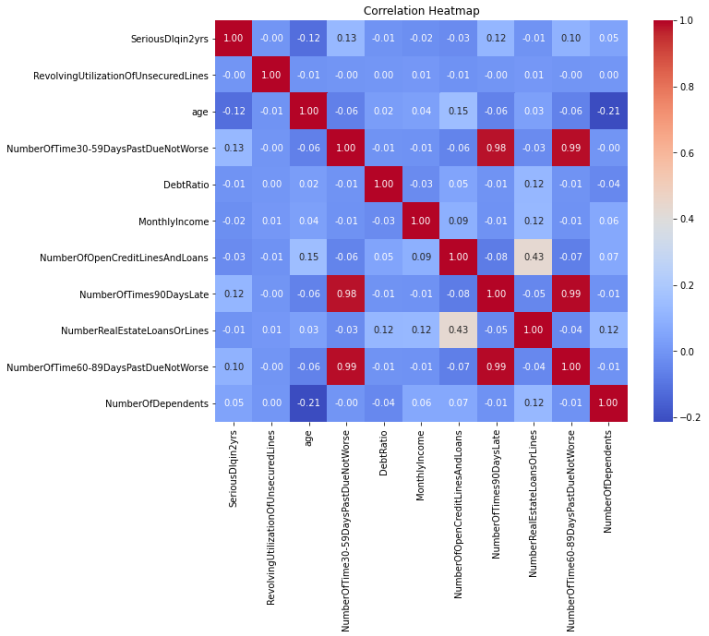

Figure 2: Correlation plot for variables

The correlation plot above shows that the past-due variables are highly correlated, while all the other variables have a low correlation. We will once again come back to the correlation of variables after cleaning the data.

3 Data Pre-processing

This section provides information on steps taken to clean the dataset in order to begin forming the models to predict default probability. The cleaned dataset is then inspected and analysis is performed on it to enable clarification and help to verify any future assumptions that may be necessary.

3.1 Data Cleaning

When cleaning the data, we had to consider not only outliers but also illogical data. In the context of credit scoring, illogical data can be determined by standard bank regulations. We applied these restrictions first and adjusted the severity of them to balance the removal of outliers while preserving a sufficient dataset for the analysis.

The age range of 18-75 was selected to encompass the primary population for the credit analysis. Individuals must be at least 18 years old to apply for loans legally. While there is no specific age limit, most banks generally agree that the maximum age should be 75 years, as the risk of the borrower passing away is relatively high. By implementing these filters, our data was reduced by approximately 10,130 entries. However, this restriction must adhere to strict legal and industry requirements regarding borrower age eligibility for loans. Although this restriction significantly narrows our dataset, it is crucial to follow legal and industrial standards.

Additionally, the RevolvingUtilizationOfUnsecuredLines criterion was set to less than 6, even though the standard is usually between 0 and 1. This was done to capture valid values while minimizing data loss, resulting in the removal of 1,237 default entries and 2,084 non-default entries. This criterion was carefully chosen to strike a balance between capturing relevant data and excluding outliers, ensuring the integrity of the dataset for analysis.

The past-due variables all contained the same extreme values of 96 and 98, this was very unusual and in total 269 entries were removed. Removing these extreme values helps maintain the accuracy of the analysis by eliminating data points that do not conform to typical patterns, thus ensuring the reliability of the results.

The DebtRatio variable was restricted to less than 2 to remove extreme outliers, ensuring a manageable level of debt relative to income while retaining sufficient data. This threshold was a loosened condition of the standard debt ratio which is between 0 and 1. This was to consider instances with reasonable outliers and disregard extreme-valued outliers.

The MonthlyIncome variable presented challenges due to missing and illogical data, such as unrealistic incomes of £1. To address this, the data was filtered to be either 0 or greater than £350, balancing the inclusion of individuals earning the minimum wage, while filtering out unrealistic data points. Allowing for a lack of income acknowledges the possibility of unemployment which is a circumstance banks offer loans to. These restrictions resulted in the removal of 883 data points with 27 defaults.

Moreover, several variables were left unrestricted in this analysis. The variables NumberofOpenCreditLinesandLoans, and NumberofRealEstateLoansorLines were devoid of constraints. This decision reflects an understanding of the complex financial situations of individuals and households, recognizing the valid reasons for differences in the number of credit lines, loans, and real estate holdings.

Finally, no limitations were imposed on the NumberOfDependents variable, which represents the count of dependents claimed by individuals on their credit applications. This deliberate approach underscores flexibility, recognizing the diverse circumstances of credit applicants. Additionally, it is noteworthy that, with the constraints applied to variables across the categories, all outliers in the NumberOfOpenCreditLinesAndLoans, NumberofRealEstateLoansOrLines, and NumberOfDependents variables were systematically eliminated, this ensured the dataset's suitability for subsequent analysis.

3.2 Analysis of Cleaned Data

In this section, we will re-inspect the outliers and correlation of the variables to ensure they are an appropriate generalisation of the initial dataset and suitable to use in our model.

3.2.1 Outliers

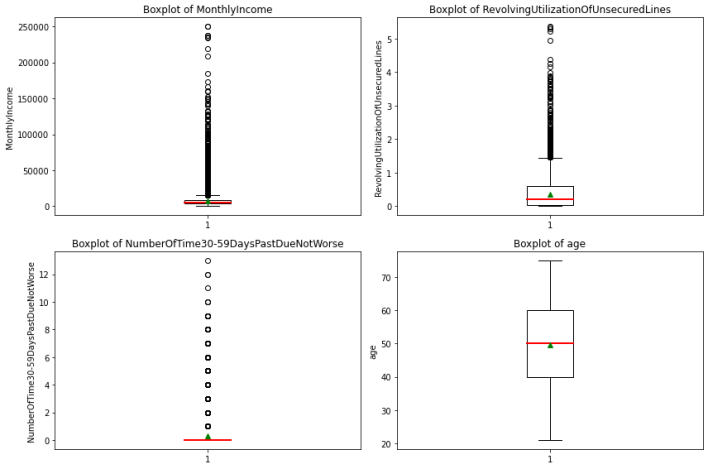

Figure 3: Modified Boxplots

The visualisation of these four modified boxplots provides insight into the distribution of the restricted variables. It can be observed that the MonthlyIncome and past-due variables no longer have extreme point outliers, this is a significant improvement from the original. In the RevolvingUtilizationofUnsecuredLines boxplot, we can see that the outliers are less spread out than the original but still encompass some reasonable outliers. Lastly, for the Age variable there are no outliers, which can be explained by the strict legal and industry requirements.

3.2.2 Correlation

The next step taken was to re-inspect the correlation between variables in the dataset. The correlation plot below, Figure 4, shows that the past-due variables are no longer highly correlated, while all the other variables maintain a low correlation.

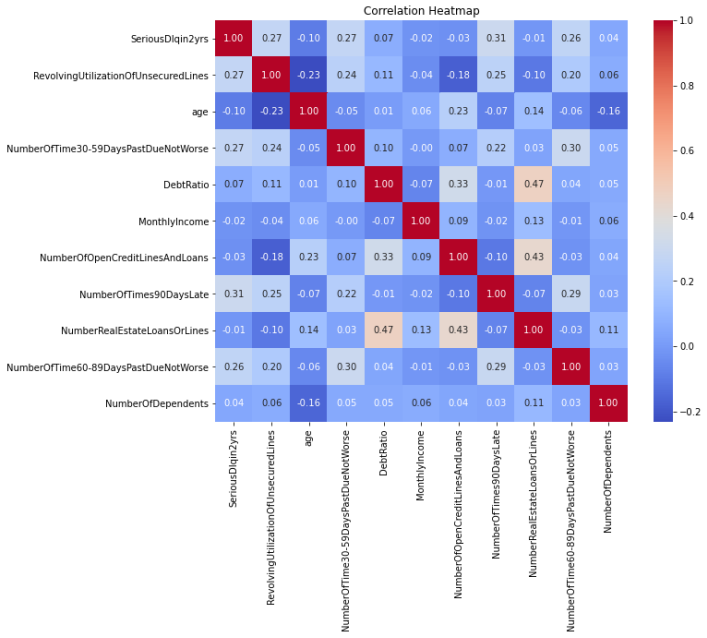

Figure 4: Modified Correlation plot for variables

3.3 Imputation

Imputation is the process of replacing missing data with substituted values, this is because it is impractical to completely remove all missing values from a dataset as it would sizeably reduce the dataset. Four methods were implemented in our study, for imputation, these include mean, median, GBDT and XGB. This section provides information for each method of imputation. Details of the evaluation to establish which method was used are given at the end of this section.

3.3.1 Mean and Median Imputation

For the mean and median imputation, we directly calculate the mean and median for the columns with missing values: NumberOfDependents and MonthlyIncome.

Table 2 shows the values which were used for imputation when mean and median were used.

Table 2: Mean and median imputation values

Imputation Method	**MonthlyIncome value**	**NumberOfDependents value**
Mean	4047	0.75
Median	2916	0

3.3.2 GBDT and XGB Imputation

To better explain our model, we'll use abbreviations SD, MI, NOD to stand for SeriousDlqin2yrs, MonthlyIncome and NumberOfDependents variables.

We first train the model XGB1 based on the cleaned dataset, where the response variable is NOD and covariates are all variables except SD, NOD and MI. Then we train the second model XGB2 after imputing NOD, where the response variable is MI and covariates are all variables except SD, MI. Notice we include the imputed NOD for the training of the second model.

We prioritize the imputation of NOD due to it having lower amounts of missing data compared to MI. This strategic approach allows us to enhance the accuracy of subsequent MI imputation, which

is faced with a higher volume of missing data. This methodology ensures a more precise and reliable data restoration process.

In the subsequent section, we explore two advanced models, Gradient Boosting Decision Tree (GBDT) and eXtreme Gradient Boosting (XGB). GBDT merges the principles of Gradient Descent with Decision Trees to enhance predictive accuracy, while XGB builds upon this by incorporating Newton's Method alongside Decision Tree methodologies for further optimization.

(a) Decision Tree - Classification

The decision tree model is a binary tree that recursively splits the dataset until we are left with pure leaf nodes.

To begin, we must outline the essential assumptions of decision tree, including absence of multi-collinearity, which has been achieved through data cleaning.

In assessing decision nodes within a classification decision tree, information theory selects the node that yields the highest information gain. This gain is predicated on the concept of entropy. Entropy, denoted as $Entropy(state)$, is computed using the formula:

$$Entropy(state) = -\sum_i (p_i * \log(p_i))$$

p_i stands for the proportion of the i^{th} group in certain state.

Usually the higher the Entropy, the less information it contains. Now we can define the Information Gain(IG) as the following:

$$IG = Entropy(parent) - \sum_i (\omega_i Entropy(child_i))$$

Here $Entropy(parent)$ and $Entropy(child)$ are defined as the Entropy of the parent/child node, while the ω_i denotes the weight based on the proportion of each classes in the node (ex. in Figure 7, the parent node has a red class of weight $\frac{1}{3}$, and the blue class has weight $\frac{2}{3}$). In Figure 7, an example calculation for $Entropy(parent)$ and $Entropy(child)$ is included.

By fitting the classification decision tree, we establish a series of decision nodes designed to facilitate the categorization of a novel dataset with indeterminate classes. This methodological approach ensures that, upon reaching the terminal leaf nodes, each node exclusively encapsulates a single class, embodying the concept of class purity.

(b) Decision Tree - Regression

After understanding the logic of classification decision tree, it's not hard to understand the regression decision tree.

In assessing decision nodes within a regression decision tree, the so-called Var_{Red} (Variance-Reduced) value is calculated to help select the node that yields the lowest group variance, which works similar to the IG in the previous section. First the variance of each node is computed using the formula:

$$variance = \frac{1}{n}\sum (y_i - \bar{y})^2$$

It's intuitive that the lower the variance, the better the selection. Therefore by following a similar fashion to the IG defined in previous section, we define the Var_{Red} as the following:

$$Var_{Red} = Var(parent) - \sum_i \omega_i Var(child_i)$$

Through the process of fitting a regression decision tree, we construct a series of decision nodes designed to efficiently classify a new data point, whose label is unknown, into a specific leaf node. Once the data point reaches its final leaf node, we assign it the mean value of that node. This refined

methodological approach not only simplifies classification but also enhances the precision of assigning values to new, unlabeled data points.

(c) GBDT & XGB

Below are the notation we'll use throughout this section to introduce GBDT and XGB:

Symbol	Description
$\underline{y}, \underline{\hat{y}}$	Label(True value),Predicted values
$L(\underline{y}, \underline{\hat{y}})$	Loss function between True and Prediction
t^k	Step size / Learning rate at step k
$\underline{d}^k$	Descent direction at step k (For Gradient Descent)
$\underline{D}^k$	Descent direction at step k (For Newton's method)
$\mathcal{L}_\phi$	Base learner used for fitting Descent direction(ϕ is the parameter for Decision Tree)
$f_k(x)$	Weak learner trained for boosting

1) From Gradient Descent to GBDT

In the canonical gradient descent optimization framework, the objective is to iteratively converge to the minimum of a loss function$\{L(\underline{y}, \underline{\hat{y}})\}$. This is achieved by updating the prediction $\hat{y}^{k+1}$ at each iteration k by $\underline{y}^{k+1} = \underline{y}^k + t^k \underline{d}^k$, where $\underline{d}^k = \nabla L(\underline{y}, \underline{\hat{y}}) = [-\frac{\partial L(\underline{y}, \underline{\hat{y}})}{\partial \underline{\hat{y}}}]_{\underline{\hat{y}}=f_k(x)}$ represents the gradient of the loss function. (The convergence proof for the gradient is in Section 12)

The Gradient Boosting Decision Tree(GBDT) model can be considered as an advanced iteration of the gradient descent methodology, augmented with principles from the regressive decision tree algorithm. The procedure for developing a GBDT model with a given dataset $\{\underline{x}\}$ paired with labels $\{\underline{y}\}$ is as follows:

Algorithm 1 GBDT

Input: Loss function $L(\underline{y}, \underline{\hat{y}})$, Base learner $\mathcal{L}_\phi$, Number of iterations M, learning rate t^k.
Output: The function $\hat{y}^M(x)$ that minimises the loss function L.
General Step: for any $k = 0, 1, 2, \ldots, M$ execute the following steps:

1. Initialize constant model: $\hat{y}^0(x) = f_0(x) = \hat{\theta}_0 = \arg\min_\theta \sum_{i=1}^n L(y_i, \theta)$,
2. Compute the steepest descent direction, $\underline{d}^k = \nabla L(\underline{y}, \underline{\hat{y}}) = [-\frac{\partial L(\underline{y}, \underline{\hat{y}})}{\partial \underline{\hat{y}}}]_{\underline{\hat{y}}=f_k(x)}$
3. Fit a base learner $\mathcal{L}_\phi$ to the descent direction d^k to produce a weak learner $f_k(x)$.
4. **Boosting**: Update prediction by adding the scaled weak learner: $\hat{y}^{k+1}(x) = \hat{y}^k(x) + t^k f_k(x)$.

The iterative process 2-4 is repeated M times, refining the model with successive 'Weak learners', such that $\underline{\hat{y}}^k$ progressively approaches the function $\hat{y}^M(x)$ that minimise $L(\underline{y}, \underline{\hat{y}})$. For further details, see Section 12.

2) From Newton's Method to XGB

Within the framework of Newton's method, the approach remains analogous to that of Gradient Descent, with the notable distinction that the descent direction is replaced by $\underline{D}^k = -[(\nabla^2 L(\underline{y}, \underline{\hat{y}}))^{-1} \nabla L(\underline{y}, \underline{\hat{y}})]$, the Newton's direction.

The XGB model incorporates the Newton's direction within the identical iterative process as the GBDT model, the sole modification is substituting the descent direction $\underline{d}^k$ with Newton's direction

$\underline{D}^k$, thus enhancing the optimization result by utilizing the second-order derivative information. For further details, see Section 12.

3.3.3 Imputation Result Comparison

The accompanying table presents the values of evaluation metrics derived from the cleaned dataset. For the evaluation, Mean Squared Error (MSE), Mean Absolute Error (MAE), and the coefficient of determination (R^2) have been computed using the mean and median methods. In this context, the mean and median methods refer to the predictive strategy where all predicted values are set to the mean or median of the observed data, respectively. It is important to note that for the Monthly Income variable, a logarithmic transformation was applied to both the training and test datasets prior to the evaluation process.

Table 3: Imputation model Comparison

	Imputed NOD			**Imputed MI**		
	MSE	**MAE**	R^2	**MSE**	**MAE**	R^2
Mean	1.35	0.94	0	0.52	0.53	0
Median	2.14	0.89	-0.58	0.52	0.53	-0.01
GBDT	1.15	0.831	0.160	0.36	0.38	0.42
XGB	1.14	0.830	0.161	0.33	0.36	0.46

The results indicate that GBDT and XGB methods deliver superior accuracy comparing with Mean and Median imputation techniques. Because XGB and GBDT shows generally lower MSE and MAE, while generating higher R^2. Consequently, it is reasonable to conclude that the XGB method yields the highest accuracy, as evidenced by the three evaluation metrics in Table 3.

3.4 PCA

Principal Component Analysis (PCA) is a multivariate method utilized to analyze datasets with correlated variables. It serves to reduce the dimensionality of a dataset while preserving much of its variability by generating orthogonal variables called principal components (Abdi 2010). In the context of credit scoring, PCA is employed to identify the most influential features for predicting loan default probabilities based on the available data.

A problem often encountered in PCA is determining the optimal number of components to retain (Zwick 1986). In the statistical community, the efficacy of parallel analysis is no longer disputed, it offers an optimal solution to determining the number of components and is deemed superior to many other approaches. In our study, we used parallel analysis, which focuses on determining the number of components that capture the greatest amount of variance of the original data, compared to those obtained from randomly generated datasets.

The variance explained by principal components signifies the extent to which the original variability of the entire dataset is represented by the reduced set of principal components. Each principal component is linked to an eigenvalue, which quantifies the amount of variance captured by that component. By comparing the eigenvalues derived from the original dataset with those from random datasets, We can assess their significance. We generated a series of random data matrices with dimensions identical to the original dataset. These random datasets are crafted to replicate the characteristics of the original dataset and provide a basis for comparison.

Eigenvalues are computed for both the correlation matrix of the original dataset and the correlation matrices of each randomized dataset. The correlation matrix is used in PCA to standardize the variables and calculate the principal components. Eigenvalues corresponding to a specified percentile, typically the 95th percentile, are selected from the distribution of eigenvalues obtained from the randomized datasets. This percentile serves as a threshold to determine the significance of the eigenvalues. Components are retained if the eigenvalue associated with a given component in the actual dataset exceeds the eigenvalue of the corresponding component in the randomized dataset. This criterion ensures that only components capturing more variance than expected by chance are retained (O'connor 2000). This process is illustrated in Figure 5 below.

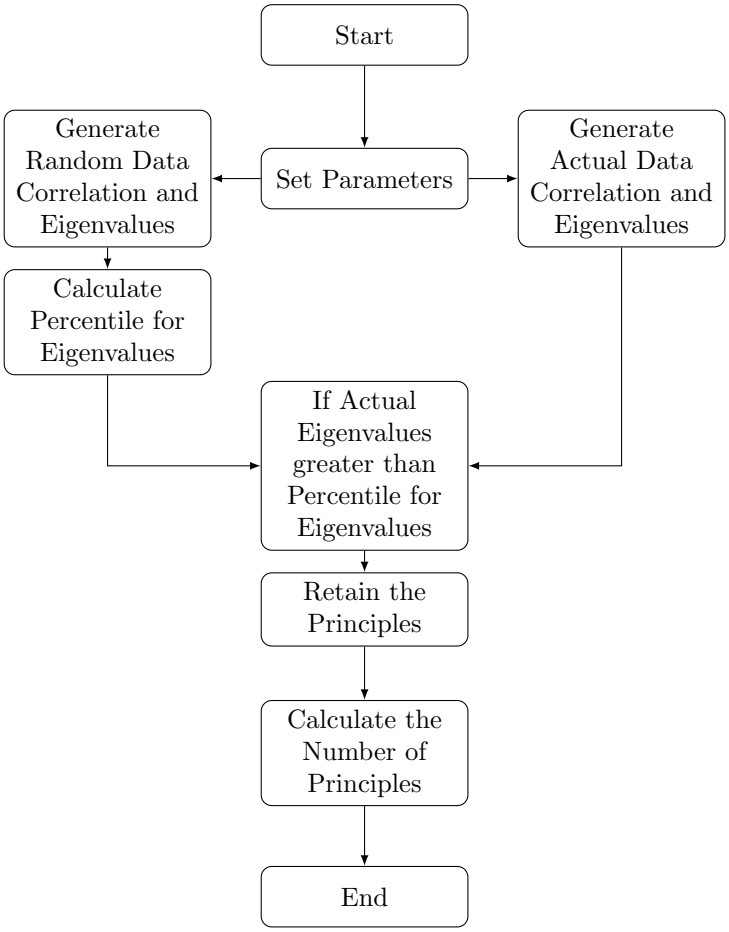

Figure 5: Process for Parallel Analysis

The graph on the left in Figure 6 illustrates that the first four eigenvalues obtained from the empirical data exceed the corresponding first four eigenvalues at the 95th percentile of the random dataset. This observation suggests that the first four components in the empirical data capture more variance than what would be expected by chance. However, the fifth eigenvalue from the empirical data falls below the fifth eigenvalue at the 95th percentile of the random dataset, indicating that the fifth component in the empirical data does not capture as much variance as expected by chance. Consequently, the decision is made to retain only the first four components. After retaining the first four principal components, the cumulative explained variance ratio is calculated, as shown in Figure 6. The cumulative explained variance ratio is approximately 67.48%, this represents the proportion of total variance in the dataset explained by the four components. However, this relatively low cumulative explained variance ratio suggests that PCA may not effectively capture a significant portion of the dataset's variability with just these four components. Therefore, PCA will not be utilized in our model.

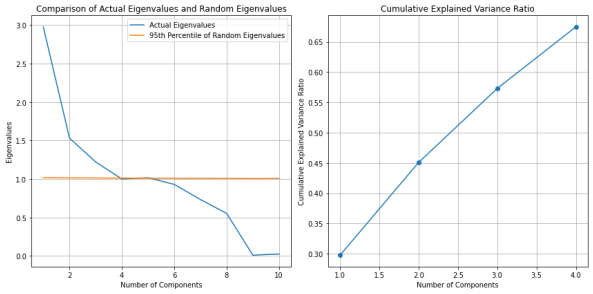

Figure 6: Visualisation of Parallel Analysis Results

4 Evaluation Metrics

This sections outlines metrics used for model results and model evaluation purposes. Multiple metrics have been considered in order to account for data imbalance, this is discussed further below.

For the model we define 0 as the non-default class (positive), where an individual does not default, and 1 as the default class (negative), where an individual does default. For interpretation purposes, we define the following:

- True Default (TD) - entries belonging to the default class and predicted as default
- True Non-Default (TND) - entries belonging to the non-default class and predicted as non-default
- False Default (FD) - entries belonging to the non-default class and predicted as default
- False Non-Default (FND) - entries belonging to the default class and predicted as non default

To understand the admissibility of different metrics, we need to fully comprehend each metric and their performances with an imbalanced dataset. The motivation for investigating these metrics arises from the problem of our model having an imbalanced dataset. The dataset we trained our models with contained 92.8% non-default entries and 7.2% default entries.

In this report, we use the minority class (Default class) as the negative class and the majority class (Non-default class) as the positive class. Following this convention, the evaluation metrics are defined as

$$\text{Accuracy} = \frac{\text{Number of Correct Predictions}}{\text{Total Number of Predictions}} = 1 - \text{Error Rate} \tag{1}$$

Accuracy measures the ratio of correct predictions to total predictions, offering a general overview of model correctness but can be misleading in imbalanced datasets due to sensitivity to data distribution.

Now we establish what are the Precision, Recall, F1-score and AUC based on Figure 7 (Demir 2022):

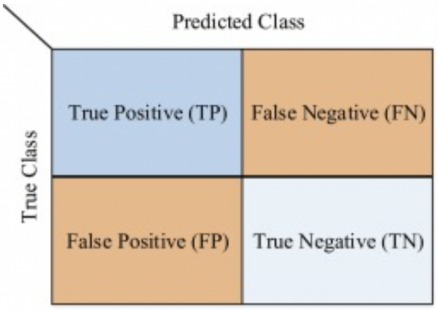

Figure 7: Confusion matrix explanation

$$\text{Precision} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Positives}} \tag{2}$$

Precision assesses the exactness of positive predictions by calculating the proportion of correctly classified positive instances among all instances predicted as positive, providing insights into the model's ability to identify positive cases.

$$\text{Recall} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Negatives}} \tag{3}$$

Recall, also referred to as sensitivity, evaluates the completeness of positive predictions by determining the ratio of correctly classified positive instances to the total number of positive instances, remaining relatively robust to changes in data distribution.

$$\text{F1 Score} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \tag{4}$$

The F1 score combines precision and recall into a single metric, offering a balanced measure of classification effectiveness sensitive to data distributions, providing comprehensive evaluation considering both correctness and completeness.

For comparison purposes, error rate (ER) is calculated. This is defined as:

$$ER = \frac{FD + FND}{TD + TND + FD + FDR}$$

Error rate is the total number of incorrectly classified entries divided by the total number of entries.

Also defined is False Non-Default Rate (FNDR). This rate gives the proportion of total defaults which are predicted as non-default. This number will help with interpreting how often banks are providing loans to those who do not pay them back, as a proportion of all people who default. We define FNDR as:

$$FNDR = \frac{FND}{TD + FDR}$$

To summarise, in an imbalanced dataset, accuracy can be misleading due to its sensitivity to class distribution, while precision and recall offer more focused insights into a classifier's performance. However, depending on the specific goals of the classification task and the relative importance of minimizing false positives and false negatives, the F1 score, which balances precision and recall, may provide the most comprehensive evaluation of classifier performance on imbalanced datasets.

In credit scoring, where the goal is to estimate the probability of defaulting for each borrower accurately, the most commonly used evaluation metric is the Area Under the Receiver Operating Characteristic (ROC) Curve (AUC-ROC). The ROC curve is a graphical representation of the true positive rate (recall) against the false positive rate at various threshold settings. The AUC-ROC measures the area under this curve, providing a single scalar value that represents the model's ability to discriminate between defaulters and non-defaulters across all possible threshold settings. A higher AUC-ROC value indicates better discrimination power: a model with an AUC-ROC value of 1 would perfectly separate defaulters from non-defaulters, while a value of 0.5 indicates a random classifier. It is rare that a score is less than 0.5 (Zhou 2007). ROC AUC is a robust evaluation metric even for imbalanced datasets, it assess the model's ability to discriminate between defaulters and non-defaulters across various threshold settings, making them suitable for imbalanced datasets where the classes are not evenly distributed.

In situations where class imbalance is present, ROC AUC offer advantages over simpler metrics like accuracy, which can be misleading due to the dominance of the majority class. By focusing on the model's discriminatory power rather than overall correctness, these metrics provide a more informative assessment of credit scoring models (He 2009).

5 Logistic Regression Model

5.1 Methods

The Logistic Regression model is a type of generalized linear model that is primarily used for binary classification problems, although it can be extended to multi-class classification problems via the One-vs-Rest technique.

To begin, we must outline the essential assumptions of logistic regression, including absence of multi-collinearity, lack of strongly influential outliers, independence of errors, and linearity in the logit for continuous variables.

When cleaning our dataset, we ensured that no extreme outliers remained. Additionally, we verified the absence of multi-collinearity using the heatmap of the cleaned dataset. The independence of errors and linearity of the logit function were confirmed as expected during the implementation of the method. With our dataset prepared and the assumptions of logistic regression validated, we can now move on to discuss the application of this method in our analysis.

To explain the intuition behind logistic regression as a model for binary classification, we first introduce the notion of odds, the favor of a particular event,

$$\text{Odds} = \frac{p}{(1-p)}$$

where p is the probability of the event that we seek to predict. Using our definition of odds we define the logit function, which is the logarithm of the odds, which we denote as $\text{logit}(p)$.

$$\text{logit}(p) = \log\left(\frac{p}{1-p}\right)$$

The logit function is a map from [0,1] to $\mathbb{R}$, which we can use to express a linear relationship between feature values and the log-odds:

$$\text{logit}(\ \mathbb{P}(y=1|x)\ ) = w_0x_0 + w_1x_1 + ... + w_mx_m$$

Here, $\mathbb{P}(y=1|x)$ is the conditional probability that a given individual falls into class 1 based on it's features.

In our task, predicting the probability of a given individual belonging to a particular class is of more use. This is in the inverse form of the logit function, known as the logistic sigmoid function, which is given below:

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

where z denotes the net input (linear combination) of weights and features associated with the individuals. The sigmoid function being the inverse of the logit function, is a map from $\mathbb{R}$ to [0,1].

Once the sigmoid function has been applied to the linear combination of weights and features for a given individual in the dataset, the predicted probability can then simply be converted into a binary outcome via a threshold function:

$$\hat{y} = \begin{cases} 1 & \text{if } \phi(z) \geq 0.5 \\ 0 & \text{otherwise} \end{cases}$$

The diagram below summarised the steps taken in a logistic regression model. The only aspect left to be explained is how the error is used to continuously update the weights throughout training.

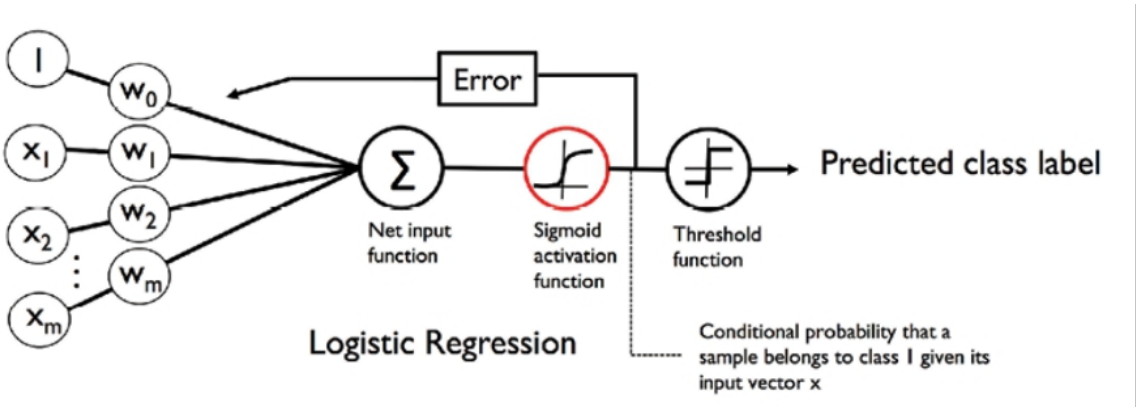

Figure 8: Process for Logistic Regression

In logistic regression a cost function is used that we wish to minimise throughout training via updating of the weights. To explain how the cost function is derived we first define likelihood.

$$L(\mathbf{w}) = P(\mathbf{y}|\mathbf{x};\mathbf{w}) = \prod_{i=1}^{n} P(y^{(i)}|x^{(i)};\mathbf{w}) = \prod_{i=1}^{n} \left(\phi(z^{(i)})\right)^{y^{(i)}} \left(1-\phi(z^{(i)})\right)^{1-y^{(i)}}$$

where $\phi(z^i)$ denotes the logistic sigmoid function evaluated at the weighted sum of features for the ith individual. This likelihood value needs to be maximised to build a logistic regression model. We apply the natural logarithm to the likelihood, which makes maximisation easier :

$$\ell(\mathbf{w}) = \log L(\mathbf{w}) = \sum_{i=1}^{n} \left[y^{(i)} \log\left(\phi(z^{(i)})\right) + (1-y^{(i)}) \log\left(1-\phi(z^{(i)})\right)\right]$$

An optimization algorithm such as gradient descent is applied to maximize this log-likelihood- function. We will take the negative log-likelihood as a cost function that can be minimised using gradient descent, see Section 12. We define the cost function as:

$$J(\mathbf{w}) = \sum_{i=1}^{n} \left[-y^{(i)} \log\left(\phi(z^{(i)})\right) - (1-y^{(i)}) \log\left(1-\phi(z^{(i)})\right)\right]$$

Using gradient descent, we can now update the weights by taking a step in the opposite direction of the gradient of our cost function $J(\mathbf{w})$ with the following updates rule:

$$\mathbf{w} := \mathbf{w} + \Delta_w$$

The weight change, Δ_w, is defined as the negative gradient multiplied by the learning rate (step size), Δ:

$$\Delta_w = -\eta \nabla J(w)$$

To compute the gradient of the cost function, we compute the partial derivatives of the cost function with respect to each weight,w_j.

Identifying the optimal value for the learning rate is crucial, as it directly impacts how well the gradient descent algorithm converges, see Section 12 for further details. Packages such as Scikit Learn in Python can find the optimal weights without the need for manual tuning of hyper-parameters like step size, as it learns efficient optimization algorithms.

Gradient descent is an algorithm that benefits from feature scaling, by applying standardization to our dataset, we shift the mean of each feature so that it is centered at zero and each feature has unit variance, as illustrated below:

$$x'_j = \frac{x_j - \mu_j}{\sigma_j}$$

Here x_j is a vector consisting of the jth feature values of all the training data. This standardization technique is applied to each feature in our dataset. Standardizing features is essential for effective regularization, ensuring that all features are on comparable scales.

Regularization is applied to mitigate over-fitting, improving the method's generalization. Over-fitting occurs when a model has an excessive number of parameters, making it unnecessarily complex relative to the underlying data. Conversely, under-fitting, or high bias, arises when the model is too simple to capture patterns in the training data, resulting in poor performance on unseen data. Further details on the regularization process can be found in Section 12.

5.2 Results

5.2.1 Model weights

The magnitude of each weight reflects the strength of the association between that feature and the outcome, with the sign indicating the direction of the association (positive or negative). These weights are the model's learned parameters after considering the data provided during the training process.

Table 4: Fitted weights for Logistic Regression

Variable	**Value**
RevolvingUtilizationOfUnsecuredLines	0.64
Age	-0.23
NumberOfTime30-59DaysPastDueNotWorse	0.31
DebtRatio	0.13
MonthlyIncome	-0.27
NumberOfOpenCreditLinesAndLoans	-0.10
NumberOfTimes90DaysLate	0.34
NumberRealEstateLoansOrLines	0.06
NumberOfTimes60-89DatsPastDueNotWorse	0.19
NumberOfDependents	0.07
Intercept	-3.08

In the context of predicting defaults, the feature most indicative appears to be the RevolvingUtilizationOfUnsecuredLines. Its positive weight (0.64) suggests that an increase in the ratio of credit card utilization is associated with a higher likelihood of default. This implies that borrowers' credit card utilization could serve as a significant predictor, with higher utilization potentially elevating the risk of default.

Subsequently, the NumberOfTimes90DaysLate also emerges as highly influential. With a positive weight of 0.34, it signifies that an increase in the number of times a borrower is 90 days late in payment correlates with a heightened probability of default. This underscores the importance of recent payment behavior in forecasting defaults, as past delinquencies may persistently impact future credit performance.

Other noteworthy predictors include NumberOfTime30-59DaysPastDueNotWorse (positive weight: 0.31), NumberOfTime60-89DaysPastDueNotWorse (positive weight: 0.19), and DebtRatio (positive

weight: 0.13). These features also exhibit positive weights, indicating that an escalation in past delinquency occurrences and debt ratio leads to an augmented default risk.

Conversely, Age and MonthlyIncome demonstrate negative weights (-0.23 and -0.27, respectively), suggesting that an increase in age and monthly income is associated with a reduced likelihood of default. This implies that older borrowers and those with higher incomes tend to have lower default risks, potentially owing to greater financial stability.

In summary, while numerous features contribute to predicting default, the RevolvingUtilizationOfUnsecuredLines feature appears to be the most influential, followed closely by NumberOfTimes90DaysLate and NumberOfTime30-59DaysPastDueNotWorse. These features could be prioritized in risk assessment and management strategies. However, a comprehensive evaluation of all features is essential to develop robust risk management policies.

5.2.2 Accuracy

The accuracy of the logistic regression model is 0.9316, or 93.16% accuracy. It is clear to see that this is a high accuracy for the model and through this the model is correctly predicting the class (default or non-default) for an individual 93.16% of the time. However, it is important to note that a large proportion of the dataset is within the non-default class, and so this imbalance may be one of the reasons for this high accuracy value.

5.2.3 Confusion Matrix and AUC ROC

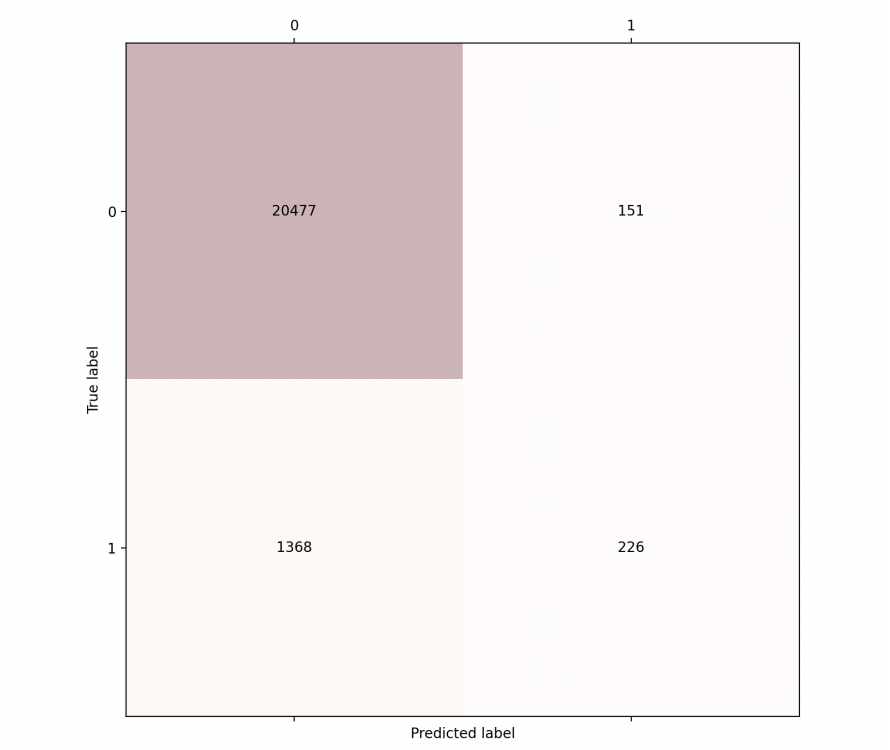

Figure 9: Logistic Regression Model Confusion Matrix

The above confusion matrix, shown in Figure 9, displays the number of predicted and actual defaults and non-defaults for the logistic regression model.

It is important to note that banks or lenders would want to minimise those who are predicted as non-default but subsequently default, ie to minimise False Non-Default results. Keeping this in mind, it can be seen that the number of False Non-Default entries for the logistic regression model is 1,368. We note that this could be due to the imbalance in the dataset, having a minority of entries as defaults

which could lead to issues with predictions. Applying this to context, it would be important for banks or lenders to be cautious of this situation and avoid large loans due to this uncertainty.

It can also be seen that the majority of entries in the confusion matrix are True Non-Defaults (20,477). This is important as it shows that, through the logistic regression model, it is often accurately predicted when individuals will not default. This ensure that banks are providing loans to those who will indeed pay them back, and with this is good for the profit of the lender. However, again we note that this large number could be partially due to the imbalance in the dataset, with the majority of entries being non-default, or belonging to class 0.

For the logistic regression model, we have an error rate of 0.068 (i.e. the model is making false predictions 6.8% of the time). Aditionally, for the logistic regression model, we have a false non-default rate of 0.858. For interpretation purposes, this shows that out of the total number of defaults, 85.8% of these individuals are predicted as a non-default. This is a noticeably high percentage and something which will be compared against other models.

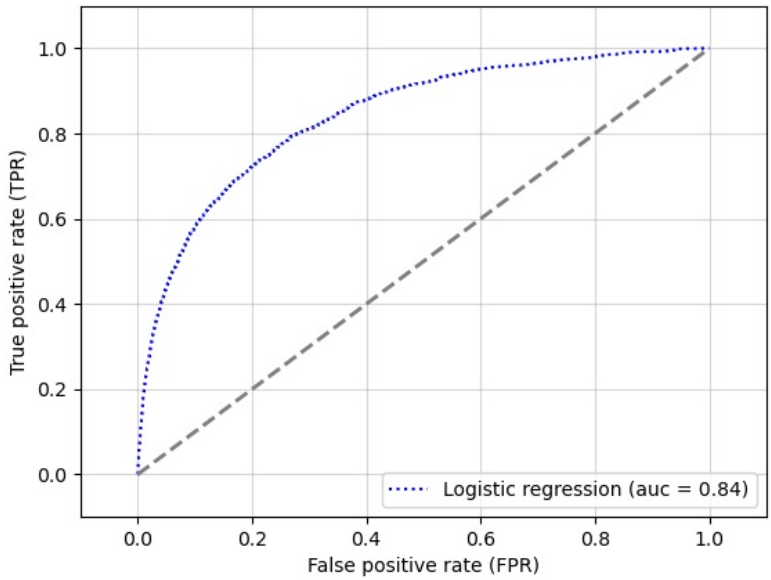

Figure 10: ROC Curve: Logistic Regression Model

The AUC score for logistic regression model is 0.84. It is clear to see that the model performs better than if it were to randomly choose a class for each prediction (0.5) and so can be of use to banks or lenders. Equally, it is noted that the score is not perfect and could be closer to 1 to obtain better prediction.

6 Random Forest

6.1 Methods

Decision trees, while transparent and intuitive, suffer from high sensitivity to training data variations, meaning that even minor changes can result in drastically different models. This susceptibility stems from the decision tree's reliance on the specific structure of the training dataset to make splits at each node. To mitigate this issue, the random forest algorithm was introduced.

Random Forest is a distinguished machine learning algorithm that combines the output of multiple decision trees to reach a single result. In the context of this task, we will use the random forest method as a classification method. To understand random forests, it is necessary to first understand decision tree learning. In Section 3.3 Imputation, we explain decision tree learning in detail. Before we begin,

we quickly quote the essential assumption of the random forest, which is the same as decision trees as it's an ensemble of them. (Ali 2012).

The random Forest algorithm addresses the high variance of Decision trees by constructing multiple trees, each based on a random subset of the training data and a random selection of features, followed by an aggregation of their predictions. This process embodies the principles of ensemble learning, specifically through two key mechanisms: bootstrapping and aggregation.

Step 1) Bootstrapping:

The algorithm creates numerous distinct datasets, named $Random_1, Random_2, ...$ by performing random sampling with replacement from the original dataset. Each bootstrapped dataset is designed to have the same number of instances as the original dataset, but will likely include some instances more than once due to the replacement of the sampling process.

Furthermore, the algorithm introduces randomness in feature selection. For a dataset containing 'm' instances and 'n' features, each bootstrapped dataset contains a subset of $\sqrt{n}$ features chosen at random without replacement.

The above two pictures demonstrate the process of bootstrapping, for instance, the first randomly generated dataset $Random_1$ contains 6 rows and 2 features x_0, x_1.

Step 2) Aggregation:

After the creation of multiple decision trees, the Random Forest algorithm aggregates their predictions. This step typically involves employing a majority voting system for classification tasks for averaging the predictions for regression tasks.

6.2 Results

6.2.1 Accuracy

The accuracy of the random forest model is 0.9311, or 93.11% accuracy. It is clear to see that this is a high accuracy for the model and through this the model is correctly predicting the class (default or non-default) for an individual 93.1% of the time. However, it is important to note that a large proportion of the dataset is within the non-default class, and so this imbalance may be one of the reasons for this high accuracy value.

6.2.2 Confusion Matrix and AUC ROC

The confusion matrix for the random forest model is shown above, in Figure 11. from this figure we can draw a number of results that can be used for model comparison. The majority of entries in the confusion matrix are True Non-Defaults (20,434), this gives confidence as these are being predicted correctly, however this may be partially due to the large imbalance in the dataset.

For the random forest model, we have an error rate of 0.069 (i.e. the model is making false predictions 6.9% of the time). It is noted that the number of False Non-Default entries for the random forest model is 1,335, and further to this random forest has a false non-default rate of 0.838. For interpretation purposes, this shows that out of the total number of defaults, 83.8% of these individuals are predicted incorrectly as a non-default. This is a noticeably high percentage and something which will be compared against other models.

Now looking to the ROC curve, shown in Figure 12, the AUC score for the random forest model is 0.84. It is clear to see that the model performs better than if it were to randomly choose a class for each prediction and so can be of use to banks or lenders.

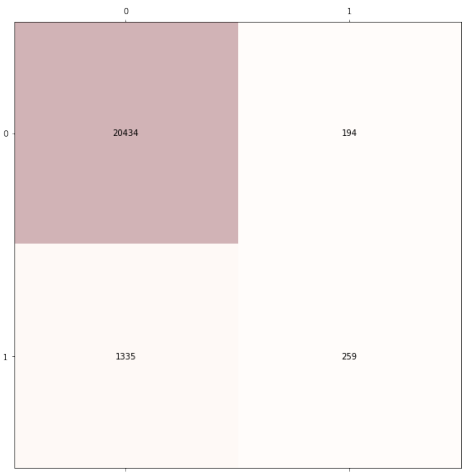

Figure 11: Random Forest Confusion Matrix

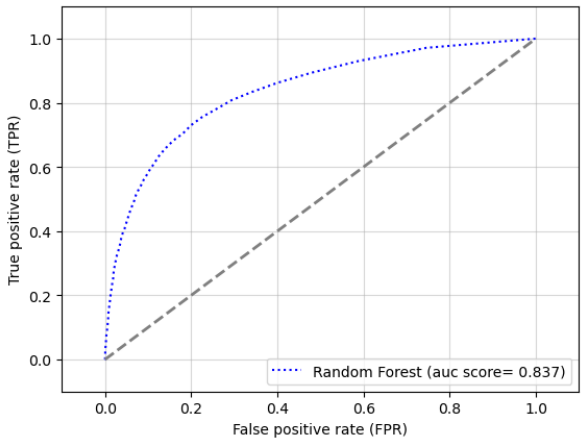

Figure 12: ROC Curve: Random Forest Model

7 KNN

7.1 Methods

The K Nearest Neighbours (KNN) method is a generalisation of the Nearest Neighbour (NN) method. It is a non-parametric machine-learning technique that assigns an unclassified sample point to a previously classified point. The NN algorithm is one of the simplest classification algorithms, but it can give highly competitive results. To understand how KNN works, we will first explain the NN method in context.

7.1.1 The Nearest Neighbor Rule

A set of n pairs $(x_1, \theta_1), \ldots, (x_n, \theta_n)$ is formed, where the x_i's take values in a metric space X upon which is defined a metric d, and the θ_i's take values in the set $\{0, \ldots, m\}$. In the context of our model, the value of n for the x_i's is the size of the training data set, and $m = 1$, which represents the respective class (Srivastava 2024)

The θ_i is defined as - $\theta_i = \begin{cases} 1 & \text{if the } i\text{th individual defaults on their loan} \\ 0 & \text{if the } i\text{th individual does not default on their loan} \end{cases}$

In each pair (x_i, θ_i), each θ_i is considered to be the class to which the ith individual belongs. The ith individual with observed outcome x_i, belongs to category θ_i. In our dataset, the first entry would be (x_1, θ_1) where the 1st individual did not default on their loan and therefore belongs to the $\theta_1 = 0$ class. All the other existing data points in the training set will similarly be assigned their class and formed into the remaining $n - 1$ pairs.

When a new point x is given, a new pair (x, θ) is formed, where only the measurement x is known, and it is desired to estimate θ by utilizing the information contained in the set of correctly classified points. We shall call $x'_n \in \{x_1, x_2, \ldots, x_n\}$ a nearest neighbor to x if

$$\min d(x_i, x) = d(x'_n, x) \quad \forall\, i = 1, 2, \ldots, n.$$

where d is the chosen distance metric for X. The event of a tie for the nearest neighbour will not be discussed in this report as it has little significance to our specific model. Applying to context, the distance metric that was chosen was the standard Euclidean norm. This was the logical distance metric for our dataset due to its emphasis on feature independence, but it also proved to have the highest classification accuracy against other distance metrics (Steinbach 2009).

The nearest neighbor rule decides x belongs to the category θ_n of its nearest neighbor x_n. A mistake is made if $\theta'_n \neq \theta$. Given in the appendix is a theorem of convergence of the NN method explaining when $x'_n \to x$ with probability one. Knowing that $x'_n \to x$ with probability one provides insights into

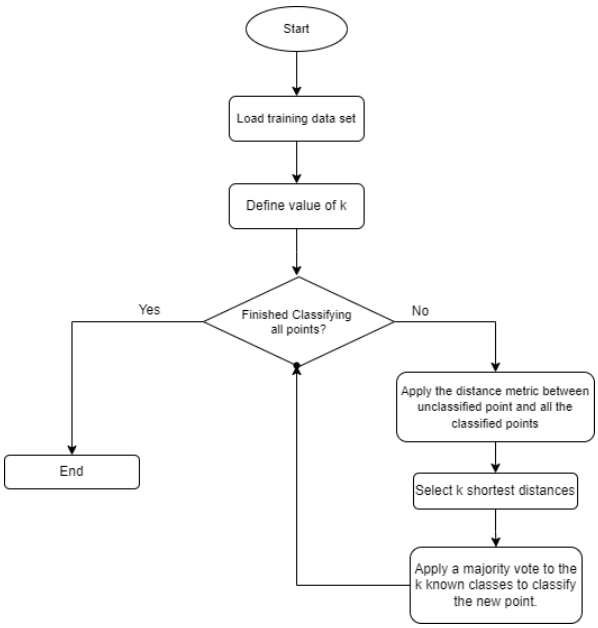

Figure 13: Process for KNN

the robustness of the NN method, it shows that as the dataset grows the NN method tends to make better predictions (Hart 1967).

7.1.2 The kth Nearest Neighbour Rule

The kth Nearest Neighbour rule follows closely from the Nearest Neighbour rule, with the difference being that for each new point x we inspect the k nearest neighbours and use a majority vote to classify the new point.

Multiple approaches can be used to determine the optimal value of k, the approach we employed was a grid-search method and we discovered that $k = 4$ was the optimal value. At first glance, an even value of k may seem counter-intuitive for a model that uses majority voting; however, upon closer inspection, it is a reasonable choice. An even value of k provides a more balanced neighborhood of the data point being classified. In contrast, an odd value of k always skews the neighborhood, resulting in a majority always being formed. Therefore, an even value of k leads to more stable predictions, which explains the higher levels of model accuracy.

7.2 Results

7.2.1 Accuracy

The accuracy of the K Nearest Neighbours model is 0.928, or 92.8% accuracy. It is clear to see that this is a high accuracy for the model and through this, the model is correctly predicting the class (default or non-default) for an individual 92.8% of the time. However, as mentioned before, this high accuracy value is likely to be due to this imbalance of the dataset.

7.2.2 Confusion Matrix and AUC ROC

The confusion matrix, shown in Figure 14, displays the number of predicted defaults, actual defaults, predicted non-defaults, and actual non-defaults. We know that banks would want to minimise the amount of False Non-Default results, and in the KNN model, it can be seen that there are 1,393 False Non-Default entries.

It can also be seen that the majority of entries in the confusion matrix are True Non-Defaults (20,401). This is important as it shows that, the KNN model often accurately predicted when individuals will not default. This ensures that banks are providing loans to those who will indeed pay them

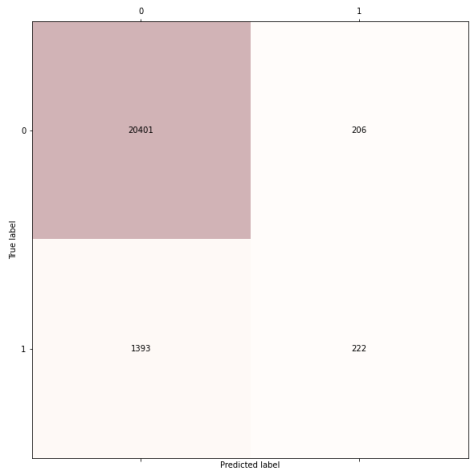

Figure 14: KNN Model Confusion Matrix

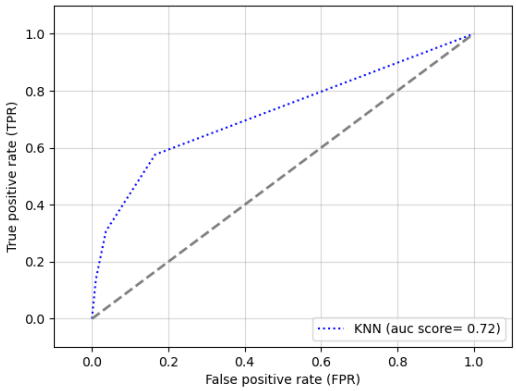

Figure 15: ROC Curve: KNN Model

back, and this is good for the profit of the lender. However, again we note that this large number could be partially due to the imbalance in the dataset, with the majority of entries being non-default, or belonging to class 0.

For comparison purposes, the error rate (ER) is calculated. This error rate is the total number of incorrectly classified entries divided by the total number of entries. For the KNN model, we have an error rate of 0.072 (i.e. the model is making false predictions 7.2% of the time). This means the remaining 92.8% of instances were correctly classified.

Also calculated is False Non-Default Rate (FNDR). This rate gives the proportion of total defaults which are predicted as non-default. For the KNN model, we have a false non-default rate of of 0.863. For interpretation purposes, this shows that out of the total number of defaults, 86.3% of these individuals are predicted as a non-default. The remaining 13.7% of default instances are correctly classified by the model as defaults. This is a noticeably high percentage and something which will be compared against other models.

In order to measure model performance, we can look at the AUC (Area Under The Curve) ROC (Receiver Operating Characteristics) curve. This method can be used to ascertain how well the model distinguishes between classes, in our case between defaults and non-defaults.

The AUC score for the KNN model is 0.72. It is clear to see that the model performs better than if it were to randomly choose a class for each prediction (0.5) and so can be of use to banks or lenders. Equally, it is noted that the score is not perfect and could be closer to 1 to obtain a better prediction.

8 Boosting Methods

8.1 Methods

8.1.1 Ada Boost Gradient Boosting

In the context of classification problems an 'ensemble method' can be thought of as the process of combining different classifiers into a meta-classifier that has better generalization performance than each individual classifier alone. The ensemble technique chosen to tackle our problem was boosting, which predominately falls into two main forms; adaptive boosting (Adaboost) and gradient boosting.

The purpose of boosting is to improve 'weak' learners, basic models slightly better than random chance, like decision tree stumps, to 'strong' learners. It's an iterative process, whereby a set of weak classifiers are consecutively trained on the training set, the training set is then re-weighted after each classifier (round of boosting) , with increased weight being assigned to entries misclassified by the previous weak leanrer, with the intuition of having the following weak classifier learn from all the prior misclassifications up until that point, therefore attempting with greater chance to now classify

these previous misclassified entries correctly. The two approaches mainly differ regarding how the weights are updated and how the ensemble of weak learners are combined to produce a more accurate predictive model.

Figures 16 and 17, given below, outline the steps taken in Adaboost and gradient boosting respectively.

Algorithm 1 AdaBoost

Initialize k: the number of AdaBoost rounds
Initialize $\mathcal{D}$: the training dataset, $\mathcal{D} = \{\langle \mathbf{x}^{[1]}, y^{[1]} \rangle, ..., \mathbf{x}^{[n]}, y^{[n]} \rangle\}$
Initialize $w_1(i) = 1/n, \quad i = 1, ..., n$, $\mathbf{w}_1 \in \mathbb{R}^n$

for r=1 to k **do**
  For all $i : \mathbf{w}_r(i) := w_r(i) / \sum_i w_r(i)$ [normalize weights]
  $h_r := FitWeakLearner(\mathcal{D}, \mathbf{w}_r)$
  $\epsilon_r := \sum_i w_r(i)\ \mathbf{1}(h_r(i) \neq y_i)$ [compute error]
  if $\epsilon_r > 1/2$ then stop
  $\alpha_r := \frac{1}{2} \log[(1 - \epsilon_r)/\epsilon_r]$ [small if error is large and vice versa]
  $w_{r+1}(i) := w_r(i) \times \begin{cases} e^{-\alpha_r} & \text{if } h_r(\mathbf{x}^{[i]}) = y^{[i]} \\ e^{\alpha_r} & \text{if } h_r(\mathbf{x}^{[i]}) \neq y^{[i]} \end{cases}$
Predict: $h_k(\mathbf{x}) = \arg\max_j \sum_r \alpha_r \mathbf{1}[h_r(\mathbf{x}) = j]$

Figure 16: Adaboost Algorithm

Algorithm 1 Gradient Boosting

Initialize T: the number of trees for gradient boosting rounds
Initialize $\mathcal{D}$: the training dataset, $\{(\mathbf{x}^{(i)}, y^{(i)})\}_{i=1}^n$
Choose $L\big(y^{(i)}, h(\mathbf{x}^{(i)})\big)$, a differentiable loss function
Step 1: Initialize model $h_0(\mathbf{x}) = \underset{\hat{y}}{\operatorname{argmin}} \sum_{i=1}^n L\big(y^{(i)}, \hat{y}\big)$ [root node]
Step 2:
for t=1 to T **do**
  A. Compute pseudo residual $r_{i,t} = -\left[\frac{\partial L(y^{(i)}, h(\mathbf{x}^{(i)}))}{\partial h(\mathbf{x}^{(i)})}\right]_{h(\mathbf{x}) = h_{t-1}(\mathbf{x})}$, for $i = 1$ to n
  B. Fit tree to r_{i_t} values, and create terminal nodes $R_{j,t}$ for $j = 1, ..., J_t$.
  C.
  for j=1 to J_t **do**
    $\hat{y}_{j,t} = \underset{\hat{y}}{\operatorname{argmin}} \sum_{\mathbf{x}^{(i)} \in R_{i,j}} L\big(y^{(i)}, h_{t-1}(\mathbf{x}^{(i)}) + \hat{y}\big)$
  D. Update $h_t(\mathbf{x}) = h_{t-1}(\mathbf{x}) + \alpha \sum_{j=1}^{J_t} \hat{y}_{j,t}\, \mathbb{I}(\mathbf{x} \in R_{j,t})$
Step 3: Return $h_t(\mathbf{x})$

Figure 17: Gradient Boosting Algorithm

Illustrated in the algorithms above we can see, we can see that adaboost and gradient descent differ in the following:

- Weight adjustment: Adaboost adjusts weights of instances after each round of boosting based on misclassifications which enables correcting of the mistakes of the previous learners by giving more weight to the instances that were previously misclassified. On the other hand, gradient boosting computes residuals based on the gradient of the loss function with respect to the model predictions, and fits trees to these residuals rather than adjusting weights. This enables each tree to fit the residual errors of the entire ensemble up to that point, effectively reducing the model's loss.

- Model update: Adaboost updates the classifier weights based on the exponential loss function, whereas gradient boosting updates the model by adding a new tree that is scaled by a learning rate α and fits the current residuals.

- Final prediction: Adaboost's final model is a weighted majority vote of the weak learners where the weights are determined by the learner's accuracies, but gradient's boostings final model is the sum of the predictions from all individual trees, hence it directly models the prediction of interest rather than voting.

As already stated, our dataset is heavily imbalanced, a large majority of individuals are classified as non-default whereas less than 10% of samples fall into the default class. Ensemble methods such as boosting can offer improved performance on such datasets due to the diversity of decision boundaries, by combining multiple weak learners, like trees in a random forest. Each tree may overfit to a different part of the training data; so when they are aggregated (gradient boosting) or combined via majority vote (Adaboost) they tend to cancel out each other's errors leading to more generalized decision boundaries.

For this reason, when implementing our two boosting models, two variations each were used. One where the number of weak learners and the maximum depth size were kept low, reducing the likelihood of over-fitting the training dataset therefore reducing variance. The second more complex variation of the models used 500 weak learners and increased tree depth, enabling an increased likelihood of over-fitting the training dataset. This increase in complexity would ideally enable the models to capture the characteristics of the minority class better, correctly identifying more defaulting individuals, which would likely be of high importance to credit lenders.

8.2 Results

8.2.1 Accuracy

The Adaboost and gradient boosting method were tested twice, once with 150 weak learners and a second time with 500 weak learners. We can observe the accuracy for each variation of the model given below:

Number of Weak Learners	AdaBoost	Gradient Boosting
150	93%	93%
500	90%	89.5%

Table 5: Comparison of AdaBoost and Gradient Boosting accuracy with different numbers of weak learners.

In table 5 above, the accuracies between both models are very similar, this can be explained by the imbalanced nature of this dataset. As mentioned earlier in this study, accuracy isn't the best evaluation metric. Nonetheless, it does provide an initial understanding of the overall effectiveness of the model, particularly when compared to a baseline model, confirming that the model is not making arbitrary predictions.

Additionally, note the roughly 3% declines in accuracy introduced by increasing the number of weak learners in each algorithm and the maximum depth of the weak learners (decision trees), may in fact be desirable. In regular circumstances, increasing the number of weak learners can result in over fitting. In gradient boosting, weak learners are added and the algorithm reduces the training error by fitting to the noise in the training data. This can degrade generalization to unseen data, as reflected in the decreasing in final accuracy. However, as we will see in the section below where we analyse the respective confusion matrices and ROC curves, that from the perspective of most risk averse credit lender's the most important priority would in fact be minimising the number of loans given to individuals that are likely to default. So the introduction of more complex models that better identifies such individual's at the expense of misclassifying individuals that wouldn't of defaulted as now defaulting, may therefore be justifiable.

8.2.2 Confusion Matrix and AUC ROC

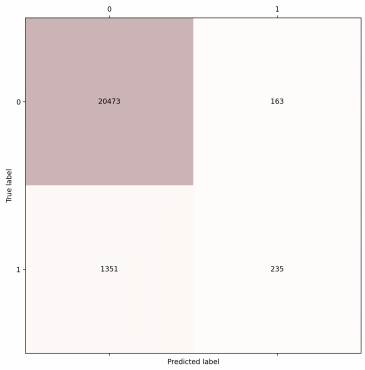

Figure 18: Adaboost ,max tree depth =1, weak learners = 150

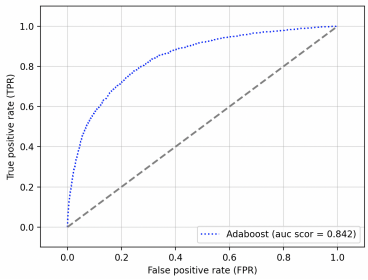

Figure 19: Adaboost ,max tree depth =1, weak learners = 150

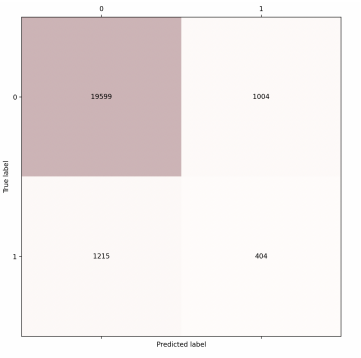

Figure 20: Adaboost, max tree depth =34, weak learners = 500

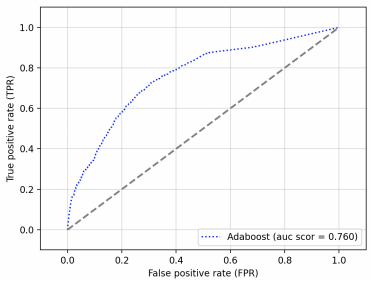

Figure 21: Adaboost , max tree depth = 34, weak learners =500

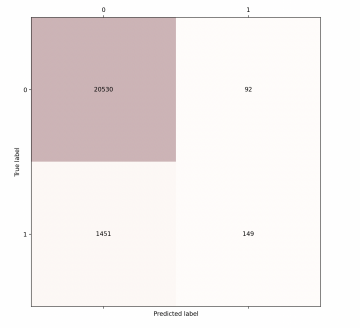

Figure 22: Gradient boosting, max tree depth = 8, weak learners= 150

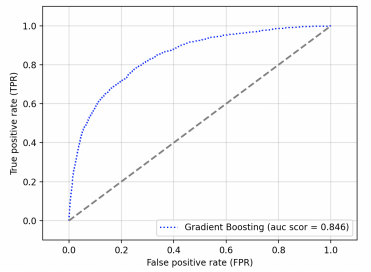

Figure 23: Gradient boosting, max tree depth = 8, weak learners = 150

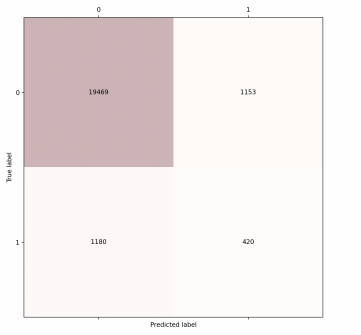

Figure 24: Gradient boosting, max tree depth = 34, weak learners = 500

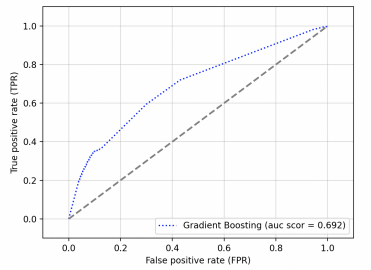

Figure 25: Gradient boosting, max tree depth = 34, weak learners = 500

As we can see from our evaluation metrics, the algorithm that has the fewest false positives, in other words fewest amount individuals predicted as non defaulting when they actually default is indeed our algorithm with overall lowest accuracy; gradient boosting with 500 weak learners, figure 25.

This is without surprise as a gradient boosted learning algorithm with such a large number of weak learners as well as high maximum depth, was bound to over fit the training dataset, as highlighted from the drop in ROC AUC score of 0.85 to 0.69. The decline in false positives, from the 150 weak learner version of gradient boosting to the 500 weak learner variation was 1451 individuals (Figure 22) to 1180 (Figure 24) which reflects roughly a 18% decline.

Our 500 weak learner Adaboost model does also reflect a significant drop in false positives in

comparison to it's 150 weak learner variation with the amount of false positives falling roughly by 10%, however the fall in overall accuracy is less steep in comparison to the gradient boosting algorithm.

So although the decline in false positives is larger both relative to it's weak learner counterpart and it's absolute value in gradient boosting, it's important to note that in the 150 weak learner variation of the models gradient boosting had far more false positives than Adaboost, a 100 more to be precise (figures 17 21), so perhaps the more significant decline in false positives of gradient boosting can be explained partially through its poorer initial performance in comparison to Adaboost.

Additionally, we would like to highlight that due to the expensive nature in regards to training ensemble methods like gradient boosting and Adaboost, it wasn't possible to apply searching techniques like grid search to find for certain the exact hyper-parameter configurations that would enable the fewest amount of false positives, or maximize our alternative evaluation metrics like ROC AUC.

9 Model Evaluation

We established a null model class, which incorporates a prediction method. In this prediction method, we employed a random selection approach, whereby, for each sample, a category is randomly chosen as the prediction result based on the sample count, specifically randomly selecting either 0 or 1 as the prediction outcome, and returning a list of the same length as the data. This methodology aims to construct a simple and intuitive baseline model for comparison and evaluation against other complex models.

Subsequent to constructing the null model, we utilized it to predict samples within the dataset and compared its outcomes with those of other trained models. Through such comparisons, we assessed the performance superiority or inferiority of other models relative to the baseline model, thereby enhancing our understanding and interpretation of the predictive capabilities and effects of the models. The performance of the trained models were assessed using diverse metrics, encompassing accuracy, precision, recall, F1-score, and the area under the ROC curve (AUC). These metrics are defined in Section 4. The results are presented in the table below. Overall, all our models significantly surpassed the null model in predictive performance.

Table 6: Model Evaluation Results

Model	Accuracy	Precision	Recall	F1-score	FNDR	AUC
Logistic Regression	93.16%	93.74%	99.27%	96.43%	85.82%	0.840
KNN	92.80%	93.61%	99.00%	96.23%	86.25%	0.720
Random Forest	93.11%	93.87%	99.06%	96.40%	83.75%	0.837
AdaBoost (150 weak learners)	93.19%	93.81%	99.21%	96.43%	85.18%	0.842
AdaBoost (500 weak learners)	90.01%	94.16%	95.13%	94.64%	75.05%	0.760
Gradient Boosting (150 weak learners)	93.06%	93.40%	99.55%	96.38%	90.69%	0.846
Gradient Boosting (500 weak learners)	89.50%	94.29%	94.41%	94.35%	73.75%	0.692
Null Model	49.28%	49.32%	92.63%	64.37%	93.15%	0.512

9.1 Model Comparison: Accuracy

When comparing accuracy we can see that the model that performs best is Adaboost with 150 learners with 93.16% accuracy,however this is closely followed by logistic regression which has 93.16% accuracy, not far short from Adaboost despite being much less expensive to use in practice in comparison. Reiterating once more that high accuracy levels across all models may be influenced by the large imbalance in the dataset. Therefore depending on the overall objective of the credit lender being considered, accuracy may not be important at all. For example, if a credit lender is primarily focused reducing resources lost on individuals that turn out to default, an algorithm's ability to correctly distinguish between such individuals would be valued at a higher degree than accuracy.

It's important to note, however, that although the accuracy of logistic regression doesn't mean much due to the nature of the problem and imbalances within the dataset, logistic regression does hold the benefit of being more interpretable than the other models, via the analysis of it's fitted coefficients.

Through this it was revealed that the most important feature in determining likelihood of defaulting was RevolvingUtilizationOfUnsecuredLines.

9.2 Model Comparison: Confusion matrix and AUC-ROC curve

First we note that an AUC score greater than 0.5 implies the model has a good measure of separability. It's capable of distinguishing between positive and negative classes with a degree of accuracy better than random chance, with models with AUC scores closer to 1 being better at predicting positive classes as positive and negative classes as negative. It's clear, from first glance at Table 6, that all our models have much greater prediction capability than random guessing with ROC AUC scores being much greater than 0.5, therefore to determine the differences in ability between models we may need to take a closer look at their respective confusion matrices.

For each model, confusion matrices allowed a number of pieces of information to be extracted and conclusions to be drawn including the error rate (ER) and the False Non-Default Rate (FNDR). This enabled clearer vision of which model was predicting incorrectly most frequently and which model was most often predicting defaults as non-defaults.

The False Non-Default rate for all models was high: logistic regression had a rate of 0.858, random forest 0.838, and KNN 0.863. Adaboost with 150 learners was 0.8518 and gradient boosting with 150 learners was 0.9069. The vast majority of lenders would prioritise reducing the false non default rate as much as possible, this is because it results in giving loans/credit to individuals that later go on to default. Unfortunately, potentially due to the highly imbalanced nature of the dataset the, the false non default rate is high across all of our models. However, by adding an abundance of weak learners to our gradient boosted model, we were able to establish a gradient boosted model comprised of 500 weak learners that had the lowest false non default rate across all the models, with a value of 0.7375. This means that for a typical credit lender seeking to minimise loss through defaulting borrowers, gradient boosting with 500 lenders would be the ideal model.

On the other hand, if the objective of a lender was to identify the factors most important in determining defaulting in borrowers then logistic regression would be by far the most sensible choice. The output of logistic regression can be expressed in terms of odds ratios, and the coefficients of the feature variables indicate the strength and direction of the association with the target variable.

Lastly, on the rare chance that a credit lender perhaps is focused on maximising the amount of loans they're giving out despite the risk of individuals defaulting, then the metric we'd be most interested in maximising is recall, the proportion of actual positives that are correctly identified as such. In other words it answers the question 'Of all actual positives, how many did we identify correctly?' maximising recall means you are trying to predict as many positives as possible, even if it results in more false positive. In this case the most desirable model would be gradient boosting with 150 weak learners, as this has the greatest recall percentage among all models.

In addition to incorporate a holistic view of model performance, the AUC (Area Under The Curve) ROC (Receiver Operating Characteristics) curve can be used. The AUC score aggregates the performance of the model across all thresholds, providing a single measure of overall performance. Comparing AUC scores, the greatest among the models was gradient boosting with 150 learners, with an AUC score of 0.846. However, gradient boosting can be very expensive to train and in practice perhaps a simpler model like logistic regression will be more desirable to use as it's much simpler to train. It can also be seen that the AUC score for logistic regression was almost identical to the gradient boosting with 150 weak learners model.

10 Further Work

10.1 Imbalanced Dataset

In the Exploratory Data Analysis section, it was observed that the dataset exhibits a significant imbalance with respect to the target variable, SeriousDlqin2yrs. Specifically, the dataset includes over 7,000 instances with a response variable value of one and in excess of 100,000 instances with a value of

zero. This pronounced disproportion between the two classes underscores the challenge of an extremely imbalanced dataset.

Having an imbalanced dataset may lead to a predictive model that is biased towards the majority class, meaning it will be more likely to predict the majority class than the minority class. This can result in a high overall accuracy rate, which might be misleading if the minority class predictions are poor. (This is why we focused on evaluating the True Positive rate in the model evaluation section)

To tackle with this problem, we'll implement oversampling techniques such as SMOTE (Synthetic Minority Over-sampling Technique), which involves adding more copies of the minority class to the dataset by creating synthetic samples.

10.2 Risk of Loans

We were tasked to calculate the probability of default, however future work that may be more beneficial to financial institutions would also be calculating the risk of the loans. When determining the most appropriate course of action for banks dealing with these individuals, we were forced to adopt the safest approach, which entailed minimizing the False Non-Default value. This approach prioritizes minimizing the instances in which the model incorrectly predicts that an individual will not default on their loan when they do. It is important to minimise this type of error; however, it is also important to recognize that this approach may not most profitable. By incorporating information about the size of the loan an individual is seeking to borrow, a more dynamic approach can be adopted. This new approach seeks to balance the opportunity cost and the risk of default. For instance, individuals seeking larger loans may be subject to harsher criteria to reflect the elevated risk associated with a larger loan, while individuals seeking smaller loans may be afforded more leniency. In summary, while minimizing False Non-Default values is a valid and prudent approach in risk assessment, integrating additional contextual factors such as loan size enables financial institutions to adopt a more nuanced and adaptive approach to managing risk and maximizing opportunities in lending decisions.

11 Conclusion

In conclusion, the main aim of this report was to develop a statistical model to predict the probability of a default on a loan, given data that is available to investors. Following this, another aim was to establish which attributes, from the 10 in the dataset, are most useful in these predictions.

In order to begin developing potential methods for predicting default probability, it was essential to complete preliminary data analysis and subsequently clean the original data. Key parts of this data cleaning involved applying restrictions to the data to remove illogical entries, these restrictions are given in Section 3.1, with relevant justifications. As well as these, outliers were removed and missing values were imputed. Having tried four methods of imputation, the XGB method was determined to be the best and was used for development of our statistical models. With data pre-processing complete, the following models were developed and fitted to the dataset: Logistic regression, random forest, K nearest neighbours and boosting methods. Multiple models were used in order for comparison purposes.

Once the logistic regression model was fitted to the data, the weights of each variable were inspected in order to establish which attributes are most useful in prediction of whether an individual will default on a loan. It was found that RevolvingUtilizationOfUnsecuredLines appears to be the most influential feature in prediction of a default, with positive weight (0.64). Having a high revolving Utilization means that an individual has higher credit debt compared to their total credit limit. Putting this into context, it is clear that those with high revolving Utilization, i.e. those going over their credit limit, are more likely to default on a loan. This is something that lenders will therefore be able to keep in mind when deciding whether to provide a loan to an individual.

It was also noted that Age and MonthlyIncome demonstrate negative weights (-0.23 and -0.27, respectively), suggesting that an increase in age and monthly income is associated with a reduced likelihood of default. This implies that older borrowers and those with higher incomes tend to have lower default risks, potentially owing to greater financial stability. Establishing these key attributes,

both positive and negative weights, is important and useful for lenders and it is hoped that these findings can be used in practice in future.

One of the first steps for model evaluation was to compare all models to the null model, which uses a random selection approach. This helped to solidify comparisons and show the extent of improvement when the other models are used. Having used a number of different methods for model comparison, it can be seen that all models performed better than the null models. Looking to accuracy as a metric, we can see that the model that performs best is AdaBoost (150 weak learners), closely followed by logistic regression. However we note that high accuracy levels across all models may be influenced by the large imbalance in the dataset.

It is noted that the vast majority of lenders would prioritise reducing the false non default rate as much as possible. The model with the lowest FNDR is gradient boosting with 500 weak learners, with a value of 0.7375 and so would be recommended to reduce giving loans to individuals that later go on to default. However, looking to another perspective, if the objective of a lender was to identify the factors most important in determining defaulting in borrowers then logistic regression would be by far the most sensible choice due to obtaining coefficients for features in the model. Finally, for those lenders who seek to only maximise the amount of loans they are giving out, regardless of risking those who default, we look to the recall metric. The model with the highest recall was gradient boosting with 150 weak learners and so would be recommended in this instance. Through model evaluation it was determined how different evaluation metrics may prove more or less useful to lenders depending on their overall objective.

For an overall view of model performance, AUC scores can be inspected. Comparing AUC scores, the greatest among the models was gradient boosting with 150 learners, with an AUC score of 0.846. This high AUC value gives evidence to using gradient boosting with 150 learners as an effective model, also backed up by the high recall, however it is an expensive model to train. Therefore, in practice, it may be preferred to look to a simpler model such as logistic regression, with an only slightler lower AUC score of 0.840.

Reflecting on initial research, it is clear that this task led to conclusions similar to those in previous work in this area of statistics. When looking to Zhang's article on logistic regression to predict default probabilities, (Zhang 2020), it gives confidence to further show how logistic regression is a successful model in such predictions. Not only this, but the work in this report solidifies the importance of data cleaning before models can be fitted, and this was something discussed by Zhang. Additionally, it was noted that models for predicting default probability are not standardised and that different lenders use different methods for this. Having evaluated the models in this report, and discussed why some models may be preferred to others, it is evident why lenders use different methods. We note that for a lender, establishing their main aim or focus may indicate which model may be preferable to them.

The large imbalance in the dataset between number of non-defaults and number of defaults has been noted as a potential limitation in this work, since an imbalanced dataset may lead to a predictive model that is biased towards the majority class. This limitation may have an influence on accuracy rates in models, and in future work the Synthetic Minority Over-sampling Technique will be implemented and accuracy will be reviewed. This would ensure that lenders are able to maximise profit through more accurate predictions for which individuals will default or not default. The importance of this work for lenders cannot be overstated and with the use of a model, the choice of which depending on the main aim of the lender, will provide an effective way to predict default probabilities.

12 Appendices

12.1 Pseudocode and Descent direction of the Gradient Method/Newton's method

The Gradient Descent/Newton's method aims to find optimal solution toward $\min\{f(\boldsymbol{x}) : \boldsymbol{x} \in R^n\}$ (Ruder 2017)

Algorithm 2 The Gradient Method

Input: A tolerance parameter $\epsilon > 0$, $x^0 \in \mathbb{R}^n$ and constant stepsize t^k
Output: The point x^{k+1} satisfying the stopping condition.
General Step: for any $k = 0, 1, 2, \ldots$ execute the following steps:

1. Compute the steepest descent direction $d^k = -\nabla f(x^k)$
2. Set $x^{k+1} = x^k + t^k d^k$. (t^k is the stepsize chosen by certain methods)
3. If $\|\nabla f(x^{k+1})\| \leq \epsilon$, then STOP and output x^{k+1}.

The unit Steepest descent direction d^k is derived by minimising directional derivative $f'(x; d)$:

$$d^k = \arg\min_{d \in \mathbb{R}^n} \{f'(x; d) : ||d|| = 1\} = \arg\min_{x \in \mathbb{R}^n} \left\{\nabla f(x)^T d : ||d|| = 1\right\}$$

By applying Cauchy-Schwarz inequality, $\nabla f(x)^T d \geq -||\nabla f(x)|| \cdot ||d|| = -||\nabla f(x)||$, while by setting $d = -\frac{\nabla f(x)}{||\nabla f(x)||}$, we obtain $f'(x; d) = -||\nabla f(x)||$, which achieves the lower bound of the Cauchy-Schwarz inequality.

Algorithm 3 The Newton's Method

Input: A tolerance parameter $\epsilon > 0$, $x^0 \in \mathbb{R}^n$
Output: The point x^{k+1} satisfying the stopping condition.
General Step: for any $k = 0, 1, 2, \ldots$ execute the following steps:

1. Compute the Newton direction $D^k = -(\nabla^2 f(x^k))^{-1} \nabla f(x^k)$.
2. Set $x^{k+1} = x^k + t^k D^k$. (t^k is the stepsize chosen by certain methods)
3. If $\|\nabla f(x^{k+1})\| \leq \epsilon$, then STOP and x^{k+1} is the output.

The Newton's direction D^k is derived by solving the minimization problem(2nd order Taylor):

$$x^{k+1} = \arg\min_{x \in \mathbb{R}^n} \{f(x)\} \approx \arg\min_{x \in \mathbb{R}^n} \left\{ f(x^k) + \nabla f(x^k)^T (x - x^k) + \frac{1}{2}(x - x^k)^T \nabla^2 f(x^k)(x - x^k) \right\}$$

By equating the derivative of the target function $\nabla f(x)$ to zero, we get $\nabla f(x^k) + \nabla^2 f(x^k)(x - x^k) = 0$,thus leading to $x^{k+1} = x^k - \left(\nabla^2 f(x^k)\right)^{-1} \nabla f(x^k)$, which leads to the newton's direction.

12.2 The convergence of Gradient Descent

Before we provide the theorem for the convergence of Gradient Descent, we first define the notation $C_L^{k,p}(\mathbb{R}^n)$.

Definition 1 (Functions with Lipschitz gradient $C_L^{k,p}(\mathbb{R}^n)$)**.** *Let $C_L^{k,p}(\mathbb{R}^n)$ be the class of functions such that*

- *any $f \in C_L^{k,p}(\mathbb{R}^n)$ is k times continuously differentiable on $\mathbb{R}^n$.*
- *The pth derivative of $f \in C_L^{k,p}(\mathbb{R}^n)$ is Lipschitz continuous on $\mathbb{R}^n$ with constant L:*

$$\|f^{(p)}(x) - f^{(p)}(y)\| \leq L\|x - y\|.$$

L is called the Lipschitz constant.

Now we're ready to prove the convergence of Gradient Descent, by combining the following **Theorem 1** and **Theorem 2**, we see that each step of gradient descent decreases the objective function by at least $M||\nabla f(\boldsymbol{x}^k)||^2$, and ultimately the norms of the gradients $\nabla f(\boldsymbol{x}^k)$ to zero.

Theorem 1 (Sufficient decrease of the gradient method)**.** *Let $f \in C_L^{1,1}(\mathbb{R}^n)$. Let $\{x^k\}_{k\geq 0}$ be the sequence generated by the gradient method for solving $\min_{x\in\mathbb{R}^n} f(x)$ with constant stepsize $t \in (0, \frac{2}{L})$ Then*

$$f(x^{k+1}) \leq f(x^k) - M\|\nabla f(x^k)\|^2$$

where

$$M = t\left(1 - \frac{tL}{2}\right)$$

Theorem 2 (Convergence of the Gradient Method)**.** *If the following two conditions are met: (1)$f \in C_L^{1,1}(\mathbb{R}^n)$. (2) f is bounded below over R^n, then*

- *for any k, $f(\boldsymbol{x}^{k+1}) < f(\boldsymbol{x}^k)$ unless $\nabla f(\boldsymbol{x}^k) = 0$*
- *$\nabla f(\boldsymbol{x}^k) \to 0$ as $k \to \infty$*

12.3 The convergence of Newton's Method

At the very least, Newton's method requires that $\nabla^2 f(x) \succ 0$ for every $x \in \mathbb{R}^n$, which in particular implies that there exists a unique optimal solution x^*. However, this is not sufficient to guarantee convergence. Strong assumptions are required about f in order to guarantee convergence of Newton's method. In particular, we require that

A1: there exists $m > 0$ for which $\nabla^2 f(x) \succeq mI$, for any $x \in \mathbb{R}^n$,

A2: there exists $L > 0$ for which $\|\nabla^2 f(x) - \nabla^2 f(y)\| \leq L\|x - y\|$ for any $x, y \in \mathbb{R}^n$.

A1 is

$$\nabla^2 f(x) - mI \succeq 0 \quad \Rightarrow \quad x^T \nabla^2 f(x) x \geq m x^T x \text{ for all } x \in \mathbb{R}^n.$$

This assumption essentially states that the eigenvalues of the Hessian must be uniformly bounded away from 0 for all x.

A2 states that $\nabla^2 f(x)$ is Lipschitz continuous, which is a strong form of uniform continuity (i.e. that $f \in C^{2,2}$).

Theorem 3 (Quadratic Local Convergence of Newton's Method)**.** *Let f be a twice continuously differentiable function defined over $\mathbb{R}^n$, and assume that A1 and A2 hold for f. Let $\{x^k\}$ be the sequence generated by Newton's method and let x^* be the unique minimiser of f over $\mathbb{R}^n$. Then for $k = 0, 1, \ldots$*

$$\|x^{k+1} - x^*\| \leq \frac{L}{2m}\|x^k - x^*\|^2.$$

12.4 Gradient Descent Convergence and Learning Rate

Please note that in practice selecting an optimal value for the learning rate is crucial, as it directly impacts how well the gradient descent algorithm converges, shown below

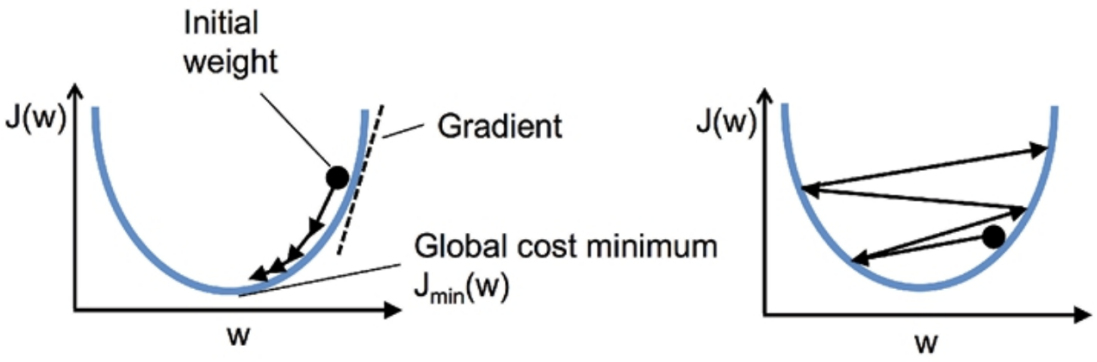

Figure 26: Importance of Learning rate

We can see in the figure above that the if the learning rate is too big for instance then we can over shoot the minimum, however on the other hand if it is too small it can take a long time for our algorithm to converge.

12.5 Regularization

The intuition behind regularization is to introduce additional information (bias) to penalize extreme parameter (weight) values. In our case implemented the most common form of regularization known as L2- Regularization, which can be written as the following:

$$\frac{\lambda}{2}\|\mathbf{w}\|^2 = \frac{\lambda}{2}\sum_{j=1}^{m} w_j^2$$

here λ is the so-called regularization parameter. So the final cost function for logistic regression on which gradient descent is applied is the following:

$$J(\mathbf{w}) = \sum_{i=1}^{n}\left[-y^{(i)}\log(\phi(z^{(i)})) - (1-y^{(i)})\log(1-\phi(z^{(i)}))\right] + \frac{\lambda}{2}\|\mathbf{w}\|^2$$

In regards to selecting the best value of lambda (regularization strength) that enables greatest performance, we implemented a search procedure known as grid search, which is a brute-force exhaustive search paradigm where we specify a list of values for different hyper parameters, and the computer evaluates the model performance for each combination to obtain the optimal combination of values from the set.

As Scikit-learn already abstracts away the need to optimize the learning rate, we only need to focus on the fine tuning the regularization strength via grid search. In practice, we run grid search using 10-fold stratified cross validation to tune our model by finding the optimal value for regularization strength based on our grid values, lastly we retrain the model on the entire training dataset to leverage the maximum amount of information available. The independent test dataset is then used to estimate the performance of the model.

12.6 Cross Validation

We have outlined the process of stratified 10-fold cross validation in figure 27 below:

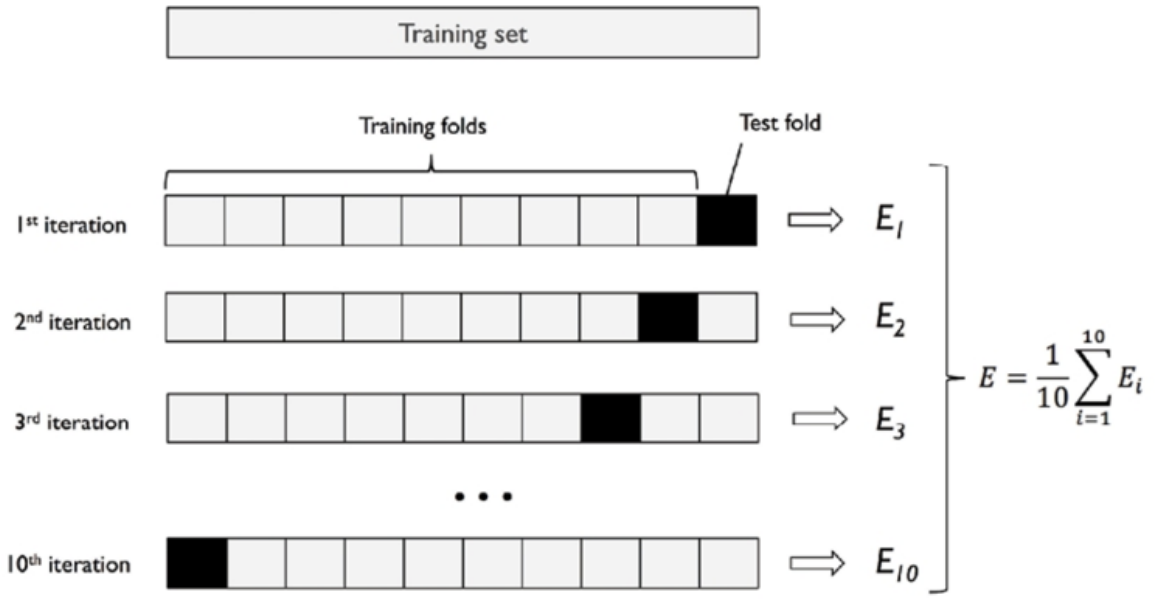

Figure 27: 10-fold cross validation

where the term 'stratified' simply means class label proportions are preserved in each fold to ensure that each fold is representative of the class proportions in the training dataset, which is a crucial step when implementing hyper parameter tuning on imbalanced datasets like this one.

12.7 The convergence of the NN Method

In our model, $x \in \mathbb{R}^n$ where n is finite and d is the Euclidean norm leading to a separable Euclidean space. A Euclidean space $\mathbb{R}^n$ equipped with the Euclidean distance metric $d(x, y) = \| x - y \|_2$, where $x, y \in \mathbb{R}^n$, is separable if there exists a countable dense subset in $\mathbb{R}^n$. Our data has no restrictions in $\mathbb{R}^n$ therefore as $\mathbb{R}^n$ is dense in itself it is separable and the theorem below holds.

Theorem 4 (Convergence of Nearest Neighbour)**.** *Let x and x_1, x_2, ... be independent identically distributed random variables taking values in a separable metric space X. Let x'_n denote the nearest neighbor to x from the set $\{x_1, x_2, \ldots, x_n\}$. Then x'_n approaches x with probability one.*

References

Abdi, Hervé (2010). *Principal component analysis*. URL: https://wires.onlinelibrary.wiley.com/doi/10.1002/wics.101.

Ali, Jehad (2012). *Random Forests and Decision Trees*. URL: https://www.researchgate.net/publication/259235118_Random_Forests_and_Decision_Trees.

Demir, Fatih (2022). *Deep autoencoder-based automated brain tumor detection from MRI data*. URL: https://www.analyticsvidhya.com/blog/2018/03/introduction-k-neighbours-algorithm-clustering/.

Hart, P. E. (1967). *Nearest Neighbour Pattern Classification*. URL: https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=1053964&tag=1.

He, Haibo (2009). *Learning from Imbalanced Data*. URL: https://ieeexplore.ieee.org/document/5128907.

Kenton, Will (2023). *Default Probability: Definition for Individuals Companies*. URL: https://www.investopedia.com/terms/d/defaultprobability.asp#:~:text=Default%20probability%20is%20the%20likelihood,management%20or%20credit%20analysis%20scenarios.

Kraus, Anne (2014). *Recent Methods from Statistics and Machine Learning for Credit Scoring*. URL: https://ebookcentral.proquest.com/lib/nottingham/reader.action?docID=5018539.

O'connor, Brian P. (2000). *SPSS and SAS programs for determining the number of components using parallel analysis and Velicer's MAP test*. URL: https://link.springer.com/article/10.3758/BF03200807.

Partington, Richard (2024). *UK banks expect sharp rise in defaults on unsecured debt*. URL: https://www.theguardian.com/money/2024/jan/18/unsecured-debt-defaults-uk-banks-lenders-forecast.

Ruder, Sebastian (2017). *An overview of gradient descent optimization algorithms*. URL: https://arxiv.org/abs/1609.04747.

Srivastava, Tavish (2024). *A Complete Guide to K-Nearest Neighbors (Updated 2024)*. URL: https://www.analyticsvidhya.com/blog/2018/03/introduction-k-neighbours-algorithm-clustering/.

Steinbach, Michael (2009). *kNN: k-Nearest Neighbours*. URL: https://vincentqin.gitee.io/blogresource-2/cv-books/8-kNN.pdf.

Zhang, Alan (2020). *Development of logistic regression model to predict default probabilities of loan accounts*. URL: https://www.proquest.com/docview/2348381305?accountid=8018&parentSessionId=ZxVG&pq-origsite=primo&sourcetype=Scholarly%20Journals.

Zhou, Kelly H. (2007). *Receiver-Operating Characteristic Analysis for Evaluating Diagnostic Tests and Predictive Models*. URL: https://www.ahajournals.org/doi/full/10.1161/CIRCULATIONAHA.105.594929.

Zwick, William (1986). *Comparison of five rules for determining the number of components to retain.Psychological Bulletin,99, 432–442*. URL: https://www.tandfonline.com/doi/abs/10.1207/s15327906mbr1702_5.