Title
Predictive Modelling for Loan Defaults

Author
Zhu, Leon

Publication Date
2019

Peer reviewed|Thesis/dissertation

UNIVERSITY OF CALIFORNIA

Los Angeles

Predictive Modelling for Loan Defaults

A thesis submitted in partial satisfaction of the requirements for the degree

Master of Applied Statistics

by

Leon Zhu

2019

ABSTRACT OF THE THESIS

Predictive Modelling for Loan Defaults

by

Leon Zhu

Master of Applied Statistics

University of California, Los Angeles, 2019

Professor Ying Nian Wu, Chair

In this paper we explore how predictive modelling can be applied in loan default prediction. The issue of predicting the outcome of a loan to be fully paid or defaulted is one of binary classification. We explore the use of different machine learning models and their performance, namely, logistic regression, random forest, neural network, extreme gradient boost and ensemble. Additionally, as is the case with many industry data, class imbalance is an issue and data as cannot be used as such in a model otherwise the model will suffer from bias. In order to solve this issue, we explore the use of sampling techniques, such as SMOTE and ADASYN, and cost sensitive learning techniques, such as class weights. Finally, using precision, recall, G-mean, and F-measure as well as precision and recall curve AUC to examine the results of each model, it was found that there is no balancing method that is consistently superior. While all models performed well after applying a balancing method, the XGBoost with class weights model performed the best. With a robust model, there are potential opportunities for it to be leveraged in optimizing profits to produce a greater return on investment. Using the best model, return on investment was able to be improved by 83%.

The thesis of Leon Zhu is approved.

Frederic R Paik Schoenberg

Nicolas Christou

Wu Ying Nian, Committee Chair

University of California, Los Angeles

2019

Table of Contents

List of Figures

List of Tables

CHAPTER 1

1 Introduction

When a bank is making an agreement with a client for the assignment of a loan, it would be beneficial if they knew how likely the client would be able to pay back the loan in full. Banks have already been making an effort to do this through the use of FICO score and credit reports. However, not everyone has credit and not everyone with a bad FICO score is a bad borrower. Equifax, one of the three major credit bureaus, determined through a recent research study that financial institutions sometimes deny good borrowers based on this FICO criteria [1]. In this area, artificial intelligence can help modernize the criteria for loan applications and allow banks to do better loan management. In fact, Equifax recently announced they are conducting research into neural network models, producing their NeuroDecision® technology, to improve lending decisions, bring new customers into the credit markets, and help detect fraudulent activity [2].

The goal of this thesis is to show how loan approval can leverage machine learning techniques to predict whether or not a client will default on their loan. By being able to know the probability a potential client will default before their loan is assigned, the bank can leverage this information to optimize their loan approvals. Another important consideration is how to handle the opportunity cost from an incorrect prediction. Suppose a client is predicted to default but in actuality would have paid off his loan and the bank does not approve the loan. In this scenario the bank would have lost the opportunity to profit from the interest that would have otherwise been made. In this thesis, I analyse the loans of a public company, Lending Club, to create a predictive model using machine learning algorithms such as Regression, Random Forest, Neural Networks,

Gradient Boosting, and ensemble methods and evaluate each of their performances in predicting loan default. This model can then be used to minimize the opportunity cost of incorrectly classified loans and maximize profits. Furthermore, the performance of different methods that address imbalanced data will be compared to assess their effectiveness on different models. Since the criterion variable is categorical, fully paid or default, the problem is one of binary classification.

CHAPTER 2

2 Data Preparation

Given that Lending club makes their loan data publicly available on their website, the data for this analysis was obtained directly from their domain[1]; downloaded in csv file format. In particular, the 2017 Q1 and Q2 data was selected to be analysed. Each row of the data describes a single loan of a borrower tracking their financial information, loan information, and payment information. Before the data can be used, the first row, containing company link, and the last two rows, containing an arrogate total of the funding, are removed since they are not relevant to this study. Additionally, the interest of this study is in whether a potential client will ultimately default; therefore, among the various available 'Loan Status', there are only two relevant observation types of interest, 'Fully Paid' and 'Default'. The status of 'Charged Off' is equivalent to a default and will be renamed as such. Keeping only these observations, the initial 2017 Q2 data consists of 37770 rows and 145 columns and the initial 2017 Q1 data consists of 43634 rows and 145 columns. One key thing to note is the data contains an imbalanced amount of the two classes. In the 2017 Q2 data, the ratio of Fully Paid to Default is 29145:8625 or about 7:2. This will impact the analysis so to account for this, the Q2 data will be combined with only the Default observations in the Q1 data. This gives a total of 47639 rows with a Fully Paid to Default ratio of 29145:18494 or about 3:2, a slightly more balanced dataset.

[1] https://www.lendingclub.com/info/download-data.action

2.1 Exploratory Data Analysis

Of the 145 available features, I am interested in only those that would be available to the lender before the loan is assigned or, in other words, the features available at the time of loan application. Given I have limited domain knowledge of loans, I perform manual feature engineering and drop features deemed to be irrelevant. For example, the feature 'last payment' would be related to information that is acquired in the lifetime of the loan and is dropped. The feature 'employment title' contains too many unique values to be of any use categorically and is also dropped. Additionally, there are a noticeable number of features containing a lot of missing data (>30%) while some features only contain a small amount of missing data (0% - 10%). Because it is difficult to accurately infer missing entries with more than 30% of the data missing, these features are dropped. At the conclusion of this feature selection part, there remains 23 features of interest shown in figure 1.

Variable	Type	Variable	Type	Variable	Type
Loan Amount	Numerical	Verification Status	Categorical	Revolving Balance	Numerical
Term	Categorical	Loan Status	Categorical	Revolving Utility	Numerical
Interest Rate	Numerical	Purpose	Categorical	Total Account	Numerical
Installment	Numerical	Debt-to-Income Ratio	Numerical	Initial List Status	Categorical
Grade	Categorical	Delinquency	Categorical	Application Type	Categorical
Employment Length	Categorical	Earliest Credit Line	Numerical	Mortage Account	Numerical
Home Ownership	Categorical	Open Accounts	Numerical	Public Record of Bankruptcies	Categorical
Annual Income	Numerical	Public Records	Categorical		

Figure 1

Of the remaining features, those with missing values are shown in figure 2. Since these numerical features have a very minimal amount of missing values (0% - 7%), dropping the rows with missing values can be justified since compared to the overall size of the data, dropping them would have a miniscule affect.

Variable	% Missing
emp_length	7.225173%
revol_util	0.065073%
dti	0.056676%

Figure 2

I inspect each of the features individually with the goal of data visualization, summarising statistics, determining usefulness for predicting the loan status, and applying transformations where necessary. Log transformations are applied to numerical features with skewed distributions. For example, the distributions of 'annual income', 'instalment', and 'revolving balance' were skewed; likely due to the nature of money distribution in a population. In order to make the data more normally distributed, a log transformation was applied to each of these features. An example of this is shown in figure 3.

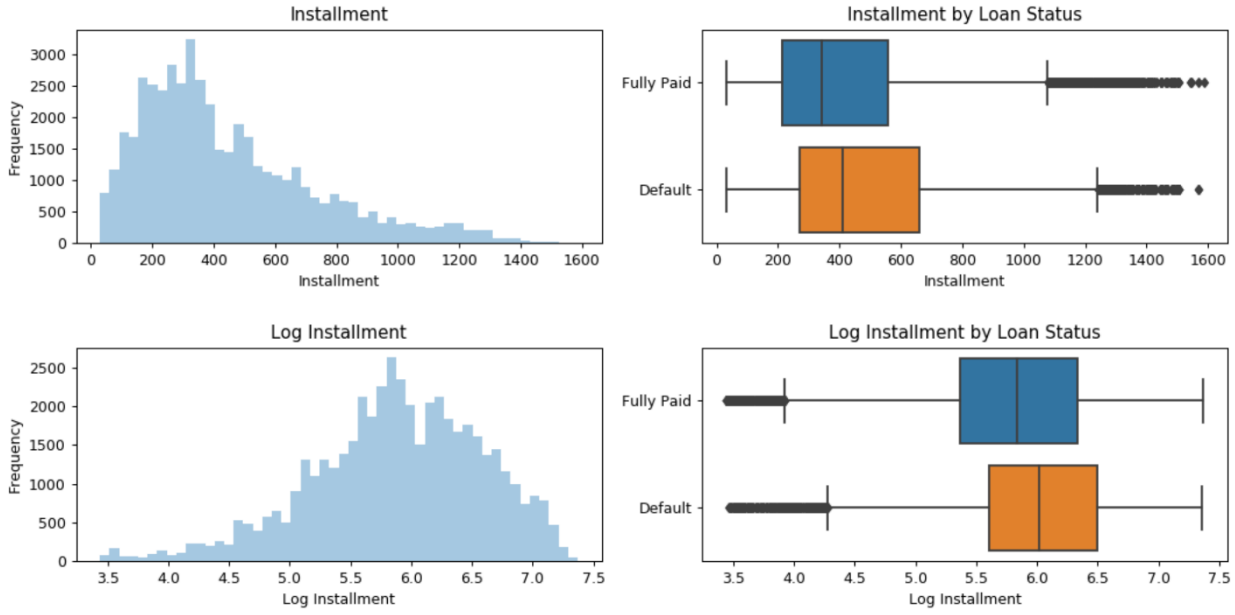

Figure 3

Additionally, the input format of some features needed to be slightly adjusted either for visualization or analysis. For example, 'earliest credit line' was inputted in such a way where only the last two digits in a year was inputted. Seeing how this could potentially create confusion when

plotted and cause issues as input for a model, it seemed reasonable to convert the year to its consecutive 4-digit format. In other features, the frequency of some occurrences was low enough to be combined into another occurrence. For example, the frequency of the 'None' category in home ownership, '10+ years' category in employment length, '10' category in mortgage account, '25' category in public records, were extremely low compared to the other categories and needed to be recut or regrouped for meaningful analysis. Finally, before the data can put input into a model, dummy variables are created for the categorical variables and the data must be split into training and test sets. The training set is used in the process of model building and the test set is used to evaluate the resulting model's performance. The entire dataset is randomly divided into training and tests set with 70% of the data going into the training set and the remaining 30% going into the test set.

An initial look of the visualized data reveals possible good indicators of the two types of response variables, default and fully paid. Potentially, it would be useful to see how these good indicators of default compare to an algorithm's interpretation of their significance or importance. For example, in regression, p-values are often used to indicate the significance of a variable in its prediction of the response variable. In other methods like random forest, feature importance can be calculated to show how often a feature was used in the algorithm's decision-making process. To visualize the data and each variable's relation to default, each predictor variable is plotted against the response variable of loan status either through a boxplot or a barchart depending on whether the variable contained continuous numerical values or categorical string values. A frequency chart of values in each variable is also plotted. An example of both forms of visualization are shown in figure 3 above and figure 4 below. For example, figure 4 shows income

verification status which consists of categorical string values. In figure 4, surprisingly, the verification status shows a higher risk of defaulting for incomes that are verified.

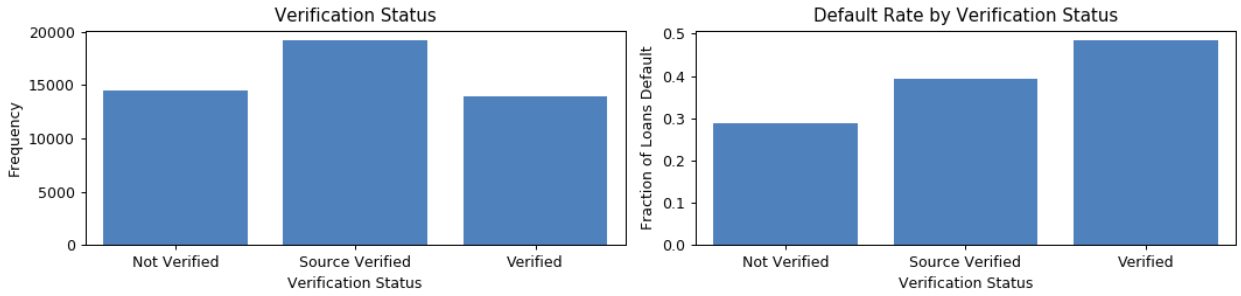

Figure 4

Other features such as debt to annual income, income ratio, home ownership, interest rate, loan amount, revolving utility, grade, term, and verification status also have similarly promising results.

2.2 Handling Imbalanced Data

In an imbalanced dataset where there are significantly more observations of one class than another, it is necessary to balance the data before training a classifier. Ideally, a classifier should provide a balanced and decent predictive accuracy for both the majority and minority class. The consequence of an imbalanced dataset, however, is that classifiers tend to have a high accuracy when predicting the majority class and a relatively low accuracy when predicting the minority class [3]. The cause of this bias is due to the majority class contributing more error signals to the loss function that classification algorithms try to minimize during training [4]. Training a model from imbalanced data results in this biased model where the training accuracy is usually high but the model often performs poorly when applied to real data. In the dataset for this analysis, there is a slight class imbalance where the ‘Fully Paid’ class consists of 61% of the data and the ‘Default’ class consists of the remaining 39%. As a result, different methods to overcome this class imbalance problem will be explored. There are many strategies to address class imbalance with current research categorizing them into five major categories, sampling strategies, synthetic data generation, cost-

sensitive learning, active learning, and kernel-based methods [5]. I explore two methods, a cost-based method, class weights, and two synthetic sampling strategies, SMOTE and ADASYN.

2.2.1 Cost Sensitive Learning

Certainly better than having no method to address class imbalance, in some instances, studies show that cost-based methods can produce better results than sampling methods [5] [6]. As such, the use of both cost-sensitive learning along with sampling methods are considered and the results of both are compared to determine which is more appropriate. In the cost-based method, weights are assigned to each class as a learning cost; the class with fewer observations, 'Default', will be assigned a higher weight. During training, mistakes made by the classifier on the minority class will be penalized based on the weight. By assigning weights to each class, mistakes made will play a role in the way by which the learning algorithm minimises its loss function. Since the minority class holds more weight, this class is valued higher and therefore prediction mistakes made by the classifier will have a higher error cost. A cost matrix or in other words, a matrix describing the class weights, is built to hold the penalization cost for misclassifications [5]. The weight is determined by the function where w_a and w_b represents the weights of the majority and minority class, N represents the total number of observations in the data, k represents the number of classes, and n_a and n_b represents the cardinality of a and b.

$$[w_a \quad w_b] = \frac{N}{k \cdot [n_a \quad n_b]} \tag{1}$$

2.2.2 Synthetic Minority Over-sampling Technique (SMOTE)

Synthetic Minority Over-sampling Technique (SMOTE) is a sampling strategy built upon the concept of undersampling and oversampling. In undersampling, a subset of the majority class is randomly chosen with or without replacement until each class has an equal number of observations. Contrarily, in oversampling, observations from the minority class is chosen with or without

replacement until both classes have an equal number of observations. Shown below in figure 5 is a visualization of undersampling and oversampling.

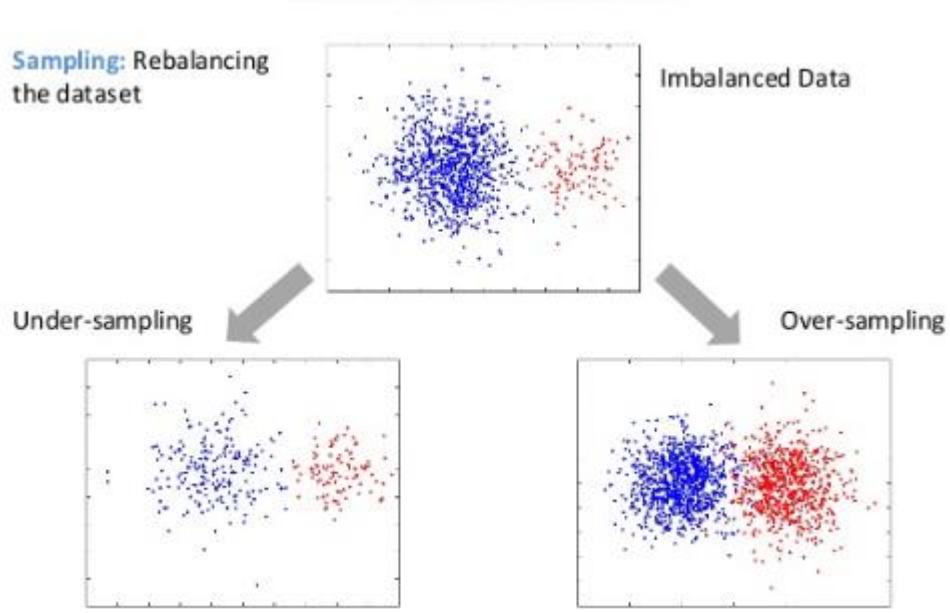

Figure 5

The SMOTE method utilizes a variation of this approach to address the issue of imbalanced classes. As an oversampling technique, the SMOTE approach synthesis new data in the minority class using the k-nearest neighbours technique to find observations similar to the existing observation [3]. Rather than simply adding copies of existing observations to the data, a synthetic data point is created between the current data point and one of its k nearest neighbours. By doing so, this provides larger decision regions rather than small dense and more specific regions created by repeated replication in oversampling. In experiments, this has shown to improve learning as well as perform better than other sampling methods [3]. There are, however, notable limitations to the SMOTE technique. Issues arise when SMOTE generates synthetic samples near the border between classes as it generates samples without consideration for the class of neighbouring examples [6]. The consequence of this is a risk of generating overlapping classes. Many

modifications aimed to improve or fix some of these issues have been made since, such as Borderline-SMOTE and ADASYN.

2.2.3 Adaptive Synthetic (ADASYN)

As a method derived from SMOTE, Adaptive Synthetic (ADASYN) sampling aims to build upon SMOTE in an effort to resolve some of its limitations. Similar to SMOTE, the goal is to provide a way to compensate for a skewed data distribution. There are two key differences between ADASYN and SMOTE. The first is that ADASYN will generate more synthetic data for minority class observations that are harder to learn than those that are easier to learn [5]. The second is that it biases the sample space by adaptively shifting the classification decision boundary toward the more difficult examples [5]. In contrast, SMOTE will generate equal numbers of synthetic samples for each minority data sample. Since ADASYN adaptively updates based on the data distribution characteristics, it is more efficient than SMOTE which relies on evaluating an underlying hypothesis performance to update [5]. In their proposal of ADASYN, the researchers behind ADASYN ran simulations on five different datasets and evaluated performance using overall accuracy, precision, recall, F-Measure, and G-Mean. Through their simulation testing comparing ADASYN and SMOTE, they found that ADASYN was able to provide improved overall accuracy, F-Measure, and G-Mean. On an average of 100 simulations, ADASYN improved accuracy for both majority and minority classes and did not sacrifice one class in preference for another [5]. Building upon their study where ADASYN was compared to SMOTE and decision trees, in this study, ADASYN is compared to SMOTE as well as cost sensitive learning to see if it can still produce similarly successful results.

CHAPTER 3

3 Criteria to measure performance

3.1.1 Assessment Metrics

Although accuracy and error rate metrics usually provide a method of describing the performance of a classifier, in an imbalanced dataset, this evaluation method does not provide adequate information [6] [7]. Since these metrics are highly sensitive to changes in the data [6], the classifier will be biased in predicting the majority class at the cost of the minority class. As such, it is simple to get a high accuracy which leads to an inaccurate evaluation of model performance. For instance, given a dataset with 100 observations, of which 85 is from the positive class and 15 is from the negative class, naively choosing to always predict the positive class results in an accuracy of 85%. However, this method is not robust and is dependent on the class distribution in the data so it would not perform well should the distribution of classes change. The expectation is for the model to perform well on both classes rather than only one at the cost of the other [6] [7]. Since accuracy is not a good indicator of model quality or performance in an imbalanced dataset, instead, other assessment metrics such as receiver operating characteristics curves, precision-recall curves, F-measure, and G-mean must be used [3] [6]. These metrics can be defined with the use of the confusion matrix described as:

	Predicted Positive	Predicted Negative
Actual Positive	True Positive (TP)	False Negative (FN)
Actual Negative	False Positive (FP)	True Negative (TN)

Figure 6

$$Precision = \frac{TP}{TP + FP} \quad (2)$$

$$Recall = \frac{TP}{TP + FN} \quad (3)$$

$$Specificity = \frac{TN}{TN + FP} \quad (4)$$

$$F1 = \frac{2 \cdot Recall \cdot Precision}{Recall + Precision} \quad (5)$$

$$GMean = \sqrt{\frac{TP}{TP + FN} \times \frac{TN}{TN + FP}} \quad (6)$$

$$Accuracy = \frac{TP + TN}{TP + FN + TN + FP} \quad (7)$$

Precision is a measure of exactness, taking the number of true positives divided by all positive predictions. Recall is a measure of completeness, taking the number of true positives divided by the number of positive values. The F-Measure combines precision and recall providing a weighted harmonic average of the two. G-Mean provides a measure of bias. In the case of loans, the cost of false negatives and false positives each have their own cost but it is likely better to have a fully paid mislabelled as default rather than have a default mislabelled as fully paid since this can incur a greater financial loss. In this case, a false positive, or a false prediction of default, would be more preferable to a false negative, or a false prediction of fully paid, so recall will be an important measure. Additionally, since there is an uneven data distribution, F-Measure will also be an important measure.

3.1.2 Receiver Operator Characteristic (ROC) Curves and Precision-Recall (PR) Curves

Receiver Operator Characteristic (ROC) curves provides a way to visualize benefits and costs by showing how the number of correctly classified positive examples varies with the number of incorrectly classified negative examples [6] [8]. The ROC curve is created from plotting the false

positive rate by the true positive rate, therefore, being closer to the upper left corner of the plot represents a better classifier. However, when there is a large class imbalance, ROC curves may be misleading when trying to make conclusions on the reliability of the classifier. As an alternative, the Precision-Recall curve can be used to provide a more informative assessment of the classifier's performance [8]. The PR curve is created from plotting the precision rate over the recall rate. In the PR space, the closer to the upper right corner, the better the classifier. Furthermore, for both of these curves, we can assess their area under the curve (AUC) to assess the performance of the model [8]. The AUC provides a measure of the model's capability of correctly predicting each class. Since in this study there is an imbalance of classes, PR curve is preferred over ROC curve as one of the indicators of classifier performance.

CHAPTER 4

4 Classifiers

To create a model, the performance of four different classifiers are compared to find a suitable model. The models chosen to be explored in this study are logistic regression, random forest, neural network, and extreme gradient boost. Furthermore, it has been shown that an ensemble method has the potential to perform better than an individual classifier. As such, the potential of an ensemble model made from the top three performing classifiers is explored. Since classifier performance results are not universal and depends on the properties of the data, it is difficult to know which classifier is the most suitable. However, previous studies on similar topics can provide direction and insight on which to explore. The motivating factor for using these particular classifiers are discussed in each following section.

4.1 Logistic Regression

Originally developed by Joseph Berkson, logistic regression, also called logit regression, is a technique used to model the probability of a categorical event. In its basic form, it is used to model a binary outcome but can be expanded to a multi-outcome model. As the focus of this study is not on logistic regression itself, the technical details of how logistic regression works are left out. There are several advantages of considering logistic regression. As a rather simple, fast to train, and straightforward model, it is easy to interpret; potentially offering business value. Furthermore, it is often recommended to start with a simple model and then progressively explore more complex models. Despite the flaws of a simple model, it will allow us to explore, learn about the data, and gain exposure to issues at a baseline level rather than at a harder to deal with complex level. When

considering the performance trade-off between a slow complex model and a fast simple model, if logistic regression performs well enough, it is sometimes preferable to choose the simpler model as it is easier to understand, faster, and possibly be more robust and less prone to overfitting in the long run.

Credit scoring is an area where logistic regression is quite popular to use and has shown to provide impressive performance [4] [9]. Similar to the goal of this analysis, the goal of credit scoring is to estimate whether an individual is a good or bad credit risk. Thus, much like loans, credit data also suffers from an imbalance of classes. In one study on credit score modelling conducted by Crone and Finlay, logistic regression was utilized and its performance was compared to discriminant analysis, neural networks, and decision trees [4]. In particular, they focused on the effect sampling and balancing has on predictive accuracy of logistic regression. This study found that balancing has a negligible effect on the performance of logistic regression which seemed to perform well regardless of the balancing strategy applied. On the other hand, they noted that, in general, oversampling methods improved the performance over an unbalanced dataset. Furthermore, the result seemed to be dependent on the sample size of the data. Oversampling performed better in their datasets with larger samples and undersampling performed better in those with smaller samples. Additionally, in another study on the same subject of credit scoring, the performance of logistic regression was shown to be comparable to those of a neural network approach in classifying percentage of good and bad loans [9]. Given that credit score modelling and loan default prediction are closely related, as well as being motivated by the results of previous studies, this study will also utilize logistic regression. Much like the previous studies, how balancing and sampling methods affect logistic regression with respect to the dataset used in this study will be explored as well as compare the performance results to those of the other classifiers

utilized in this study. Additionally, the role of logistic regression in an ensemble method will also be explored.

An initial base model is built to serve as the basis for the configuration of the other models as well as serves to provide the basis performance metrics to evaluate against. In other words, the models built from using the class weights, SMOTE, and ADASYN balancing methods will use the same model parameters as the base model with an addition of model balancing method applied. After reviewing the results of the base model, the p-values indicating the significance of the predictor variables was studied. An attempt was made at feature selection using these p-values, selecting only significant features for the model, however, this proved to show very little improvement in subsequent model performance and so was not adapted. While all three methods of addressing class imbalance clearly showed substantial improvement in recall and F-measure, overall, SMOTE produced the best results as presented in table 1 below. The effect of SMOTE and ADASYN on the imbalanced data is similar, with SMOTE doing slightly better. Recall improved by 103% and F-measure improved by 39%.

Table 1

Logistic Regression	**Balancing Method**			
	SMOTE	***ADASYN***	***Class Weights***	***Base***
Accuracy	64.49	63.95	64.03	**65.70**
Precision	0.53	0.52	0.52	0.59
Recall	**0.63**	0.61	0.59	0.31
Specificity	0.65	0.66	0.67	0.87
F-Measure	**0.57**	0.56	0.56	0.41
G-Mean	**0.64**	0.63	0.63	0.52

4.2 Random Forest

Random forests, in general, are very good at providing good predictive power, low overfitting by nature of randomness, as well as provides a model that is highly interpretable [10]. This ease of

interpretability comes from providing an importance measure of each predictor variable. As a result, it is easy to understand how much each variable is contributing to the classifier's prediction output. This capability of being able to identify and interpret relevant predictor variables is of interest and importance in many business applications who want to know which variables weigh the heaviest in order to make adjustments from a business standpoint. By knowing which variables are the most important, businesses can improve the data collection process to emphasis collecting the important variables more accurately.

There are two main methods of measuring feature importance, mean decrease impurity (impurity-based importance) and mean decrease accuracy (permutation feature importance) [11]. In classification, the measure of impurity is either the Gini impurity or the information gain/entropy and in regression, the measure of impurity is variance. In Gini impurity, how much each feature decreases the weighted impurity in a tree is calculated. Impurity can be thought of as a measure of the accuracy when classifying a variable if the variable was classified based on the distribution of labels in the dataset. The impurity decrease from each feature is averaged and the features are ranked according to this measure. The more a feature decreases the impurity, more important the feature is. However, feature selection in the Gini impurity method is biased towards preferring variables with a large number of values or scale of measurement and overestimates the importance of correlated variables [11] [10]. When there are correlated features, one of the correlated features will be highly important while the others will have a lowered importance. Feature selection using permutation importance provides a more reliable technique that is applicable to any model and is reasonably efficient though computationally more expensive than impurity-based importance. In this method, the values of each feature are permuted and measured to see how much the permutation affects the accuracy of the model.

Once more, just like in logistic regression, a base model is built to serve as a basis for comparisons. The parameters for the base random forest model are determined through configuration to optimize the model for the input data. First, the number of trees used by the model will have to be optimized in order to balance the trade-off between increasing performance and increasing computation cost as well as reduce the possibility of overfitting. The model is initially trained using all the predictor variables where out of bag error is used to determine the optimal number of trees for the classifier. Figure 7 shows that after about 500 trees, increasing the number of trees no longer yields significant improvement in error rate. Based on these results, the number of estimators that is most reasonably suitable for this data seems to be 500 so the base model can be fitted to 500 trees.

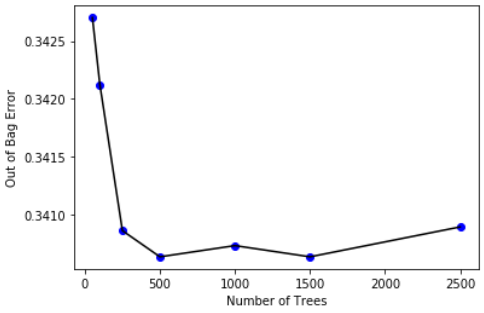

Figure 7

After fitting the classifier, the second step is to calculate the importance of each feature by method of mean decrease accuracy, or permutation feature importance. In this model, permutation importance is utilized rather than mean decrease impurity to account for any bias that may be attributed to variable cardinality, scale of measurement, or correlation. Figure 8 below illustrates the results of permutation importance for the top 15 features. It is interesting to note that most of these features determined to be important had previously shown to be good possible indicators during the data exploration phase.

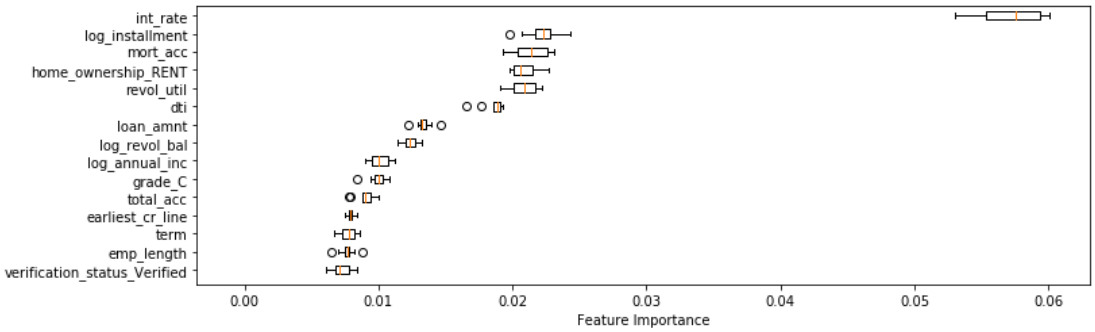

Figure 8

From these 15 features, the top 10 are selected and a training set consisting only of these features are used in training the model. After utilizing this method to produce three trained models, similarly, to the case of logistic regression, all three methods once again showed substantial improvement in recall and F-Measure over the base model as can be seen in table 2. Unlike previously, the class weights version clearly out performs the others, improving the recall by 60% and F-Measure by 20%.

Table 2

Random Forest	**Balancing Method**			
	SMOTE	***ADASYN***	***Class Weights***	***Base***
Accuracy	67.99	**68.01**	65.67	67.73
Precision	0.59	0.59	0.54	0.60
Recall	0.52	0.52	**0.69**	0.43
Specificity	0.78	0.78	0.64	0.83
F-Measure	0.55	0.55	**0.60**	0.50
G-Mean	0.64	0.64	**0.66**	0.36

4.3 Neural Network

The reason for choosing neural network as a classifier to be used in this study is a result of the properties of neural networks as well as motivated by the results from similar studies where neural networks had been applied with success. Classification has been demonstrated to be one of the areas neural networks have become increasingly applicable in. In some studies, they even

outperform traditional statistical methods such as discriminant analysis or regression [12] [13]. In a paper [13] on neural networks, Zhang outlines 4 strengths of neural networks. The first strength of neural networks is their robustness as they do not rely on any underlying assumptions or specifications and can self-adapt to the data. Secondly, they are applicable to any function and can be used for any approximation. Third, the nonlinear nature of neural networks allows them to model even complex relationships within data, often a property of real-world data. Finally, neural networks are able to provide posterior probability estimates which is useful in allowing the output to be used to minimize alternative risk functions or suggest alternative measures of network performance. Given these advantages, naturally, neural network becomes very viable as a classifier for the data in this study. The theoretical concepts of neural network have also been successfully tested using real world data and successfully applied in many business applications. Furthermore, in a study on building credit scoring models using neural networks and traditional techniques such as regression, when comparing the model performance, researchers found that neural networks were superior to logistic regression in correctly classifying bad loans [9]. Due to the similar nature between bankruptcy prediction and default prediction and the success of their results, in this study a neural network model is also trained.

The general structure of a feedforward neural network model consists of an input layer, hidden layer, and the resulting output. In the base model, serving the same functions as before, the number of these layers to add and other model parameters are determined. Although a single layer would be sufficient, the training time required could be high. Therefore, two hidden layers are added to reduce the training time. In this model there are 20 processing elements in the input layer, 15 in the first hidden layer, 4 processing elements in the second hidden layer, and 2 in the output layer. ReLu is chosen for the activation function for the hidden layers and since this is a binary

classification problem, the sigmoid activation function will be used. With these parameters set, the other three models, one for each balancing method, can be built using the base model as a base. The results can be compared and is shown in table 3. Unlike the results from the other models, in this case for neural network, class weights did not result in any improvement over the base model. Instead, SMOTE and ADASYN seemed to be more effective at improving model performance with ADASYN producing the best results. ADASYN improved the recall by 86% and F-measure by 28%.

Table 3

Neural Network	**Balancing Method**			
	SMOTE	***ADASYN***	***Class Weights***	***Base***
Accuracy	63.85	63.02	66.74	**67.16**
Precision	0.52	0.51	0.60	0.61
Recall	0.68	**0.71**	0.39	0.37
Specificity	0.62	0.58	0.84	0.86
F-Measure	0.59	**0.59**	0.47	0.46
G-Mean	**0.65**	0.64	0.57	0.56

4.4 XGBoost

Although a relatively new method, Extreme Gradient Boosting offers several advantages over the other algorithms applied in this study as well as comparable performance to the other algorithms. Tree boosting techniques have been shown to produce good results on many standard classification benchmarks [14]. Extreme Gradient Boosting, or otherwise known as XGBoost, is a boosting algorithm created by Tianqi Chen that builds upon the gradient boosting algorithm. Boosting algorithms aim to build a strong model through an ensemble of weak learners. Weak learners are added to correct existing errors and this process is repeated until no improvements can be made to the prediction accuracy. Illustrated in figure 9, is an example of this process.

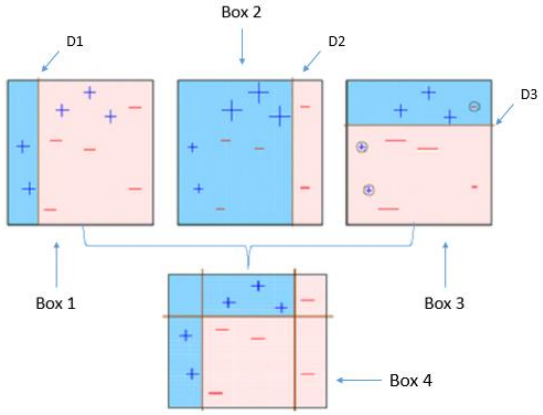

Figure 9

Gradient boosting builds trees that adds models that predict the misclassification error of prior models and tries to minimize the error when adding new models. In this way, the final model is the combined result of all previous parts.

The advantage of XGBoost is its speed and performance which can be attributed to its utilization of parallel computing that makes learning faster [14]. Regardless of the size of the data or number of machines, XGBoost runs relatively faster than other algorithms. XGBoost has been shown to run over ten times faster than other algorithms as well as outperform them [14]. XGBoost differs from traditional gradient boosting algorithms through its innovative use of parallel tree boosting, which allows for faster learning and data processing, a weighted quantile sketch procedure, which enables handling instance weights in approximate tree learning, and an effective cache-aware block structure for out-of-core tree learning [14]. Most importantly, these are combined into a highly scalable end-to-end tree boosting system [14]. Similar to random forests, XGBoost is also able to produce a measure of feature importance. Given these advantages, XGBoost applied to this dataset has the potential to reduce run times and produce competitive performance.

In the base XGBoost model, parameter tuning and feature selection is done to maximize the potential capabilities of the classifier. Using the classification error and AUC results shown below in figure 10, it can be inferred that although the training results still improve, improvement in the test set begins to slow, suffering from diminishing returns and potentially overfitting the data. As the number of estimators increase, the discrepancy between train and test grow larger. As a result, the number of estimators is set to 300 as this seems sufficient where there is not a significant discrepancy between train and test and the classification error and AUC no longer improve a significant amount from increasing the number of estimators. Using a combination of a max tree depth of three, learning rate of 0.05, and 300 estimators was shown to give the best results.

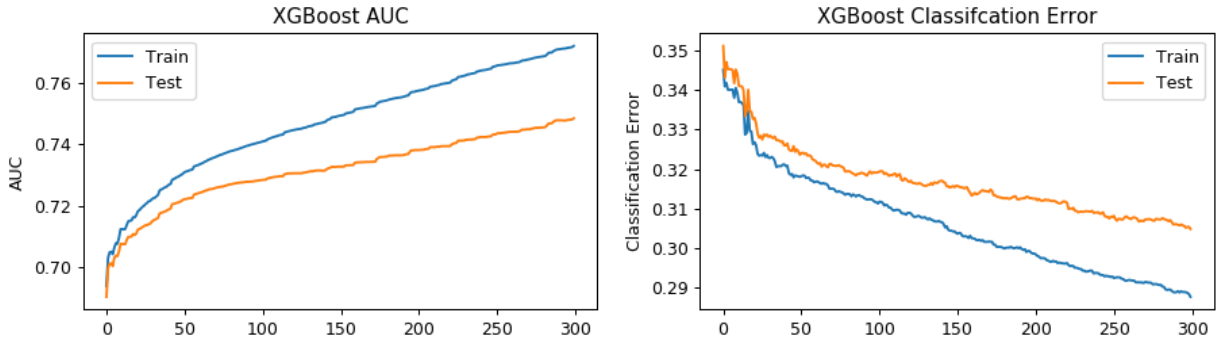

Figure 10

Additionally, since XGBoost is able to provide a measurement of the features important in its decision-making process, feature importance can be referenced to build a subset of optimal features. Notably, the feature importance shows a similar pattern to the features that random forest determined were also importance. Both algorithms have overlaps in the features they deem important. Just as was in the case for Random Forest, many of the features deemed important or useful by XGBoost was also notable in the exploration phase. Finishing parameter tuning and feature selection in the base model, the other models with balancing methods applied can be adapted and built using the same configuration. Each of the balancing methods had great positive

effect on the underlying XGBoost model and the results are presented in table 4. Unsurprisingly, once again, all three balancing methods produced some improvement over the base model; although it should be noted the base model, in the absence of any balancing method, still produced decent results. SMOTE and ADASYN performed similarly but class weights produced exceedingly well results with an improvement of 49% to recall and 15% in F-Measure.

Table 4

XGBoost	**Balancing Method**			
	SMOTE	***ADASYN***	***Class Weights***	***Base***
Accuracy	66.77	66.83	67.63	**69.51**
Precision	0.56	0.56	0.56	0.63
Recall	0.56	0.57	**0.73**	0.49
Specificity	0.74	0.73	0.64	0.82
F-Measure	0.56	0.56	**0.63**	0.55
G-Mean	0.64	0.64	**0.68**	0.63

4.5 Ensemble

Although each model individually can perform decently well, it is possible to improve overall performance by combining the decisions from each model into an ensemble model [15]. Ensemble methods help reduce errors due to noise, bias, variance and can possibly reduce degree of overfitting. Although this is not consistently true for all datasets and models, it is worth exploring as it can lead to substantial performance improvement. In a credit scoring study [15], ensemble learning was studied to determine its effect on performance. In their comparative assessment, they found their ensemble models was an improvement over their individual learners and produced the best performance. Since there are similarities in the properties of their credit dataset and the loans dataset in this study, exploring ensemble models can possibly also have similar results in improving performance.

An ensemble model is made from a combination of individual models. There are different methodologies to create an ensemble model. For instance, one method is to take the prediction probabilities of each model and average them and another is to take the class prediction of each model and take the majority vote. A more complex ensemble can be created through a method known as stacking. In the credit scoring study with ensemble models, despite bagging and boosting being more popularly used [15], it was their stacking ensemble model that became one of their best performing models. In stacking, base leaners are made of different learning algorithms individually fitted on the same data. Their prediction outputs are then collected and combined into a new dataset to be used as a new set of predictors by another learning algorithm, the meta learner. The meta learner, is then trained on this data to produce a final prediction. The entire process can be summarised in figure 11 below.

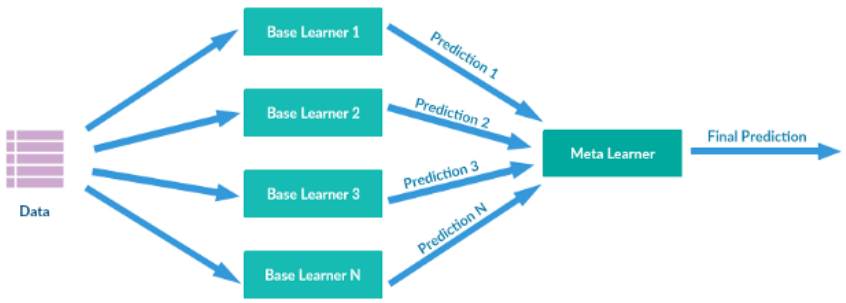

Figure 11

Based on these techniques, three ensemble models were built, one simple ensemble model using the averaging method and two stacking ensemble models. The base learners for the ensemble models consist of the random forest with class balancing, neural network with ADASYN, and XGBoost with class balancing models built previously. For the two stacking models, the meta learners uses logistic regression and XGBoost trained with class weights. The results and a comparison of the results is shown below in table 5. Overall, the ensemble models all performed

well with almost equivalent performance results. Although ensemble methods did not result in substantial improvement over any of the individual classifiers that they are made from, they performed almost equivalently well to best models produced by each classifier. The ensemble model that performed the best was from the ensemble method of averaging which performed slightly better than either of the stacking methods.

Table 5

Ensemble	**Method**		
	Average	***LR Stack***	***XGB Stack***
Accuracy	**66.66**	66.52	65.67
Precision	0.55	0.55	0.54
Recall	**0.71**	0.68	0.69
Specificity	0.64	0.65	0.64
F-Measure	**0.62**	0.61	0.60
G-Mean	**0.67**	0.66	0.66

CHAPTER 5

5 Results

An initial look at the data showed that there are several important predictor variables clients provide that could be potential powerful indicators of default. Among the available variables, the analysis showed the top 3 most influential indicators were the client's interest rate, annual income, and home ownership. Applying class weights, SMOTE, and ADASYN to balance the data showed varying degrees on improvements for each model. For some models, class weights worked well, while for others, SMOTE worked well. The varying degree of success for each balancing method shows no one method is necessarily superior but rather, in the case of imbalanced data, it is important to determine which method is suitable for some particular dataset and model. Out of logistic regression, random forest, neural network, XGBoost, and ensemble models, XGBoost with class weights performed the best; achieving an overall accuracy of 68% when attempting to predict each client's loan status. Illustrated in figure 12 is a comparison of ROC and PR that also shows XGBoost performing above the others. The ensemble model was also able to achieve similarly decent results.

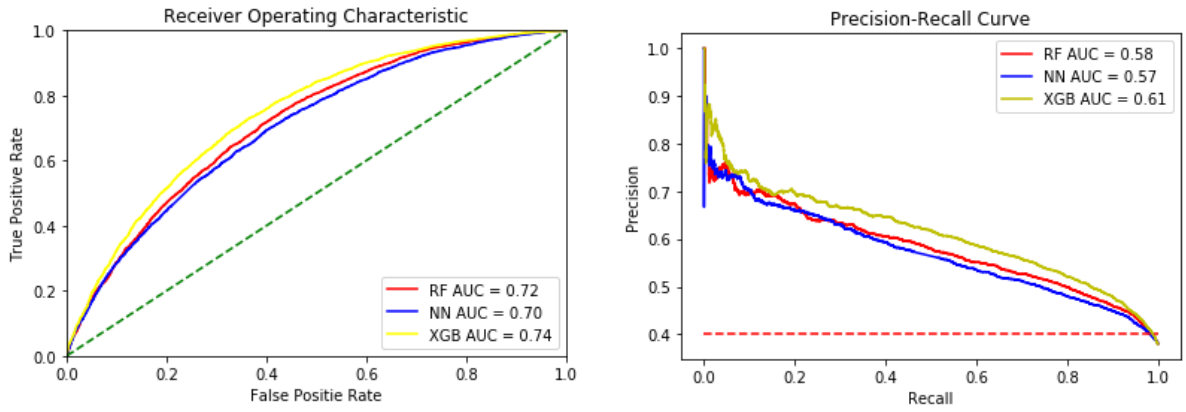

Figure 12

Additionally, for any individual, the model is able to provide a probability of its prediction. The interpretation of the model's output is as follows: an output of default 78% suggests a client will have a 78% chance to default. The more confident the model is, the more accurate the prediction is. Although the overall accuracy of the best performing model, XGBoost, was 68%, the model's individual predictions are fairly accurate. Prediction outputs between 80% – 100% was able to, on average, result in a correct prediction over 90% of the time. Figure 13 shows highly confident predictions, however, are rare and there are much more predictions in the 50% – 70% range which explains the overall accuracy of 68%.

The practical application of the model and these results can be applied to improving return on investment (ROI). There are different ways to improve returns and possible optimization that can be done in this area. In this study, overall investment return was compared as well as average improvement on ROI by prediction categories. Overall, if one was to take the raw brute force approach and invest in every loan in the test data, one would incur a loss of 21%. Alternatively, if one was to apply the results of the model and invest only on the loans the model predicts to be fully paid, one would instead incur a loss of 8%. In other words, applying the model prediction to avoid investing on loans predicted to default results in a return on investment growth of 83%. Additionally, the model's output of probabilities was categorised into ranges. For example, if the

model predicted fully paid with 75% probability, this would be categorised to be in the 70% - 80% range. After each loan is categorised into each of the five categories seen in figure 13, returns on loans within each category with and without model prediction can be calculated. For example, for all loans that had predictions between 50% – 60%, first without model prediction assistance, an average return is calculated assuming one makes an investment on every loan. Next, with the advantage of using the loan predictions, an average return is calculated assuming one invests only in loans predicted to be fully paid. Finally, using this, the improvement of average return from using model prediction in each category can be calculated. As can be seen in figure 13, predictions with probabilities in the 50% – 60% range resulted in an 12% improvement on return on average over investing on all loans in that range. With increasing probabilities, there is an increasing improvement on return until the 80% – 100% range. In this range, few to no loans are defaulted on so there is not much opportunity to improve returns.

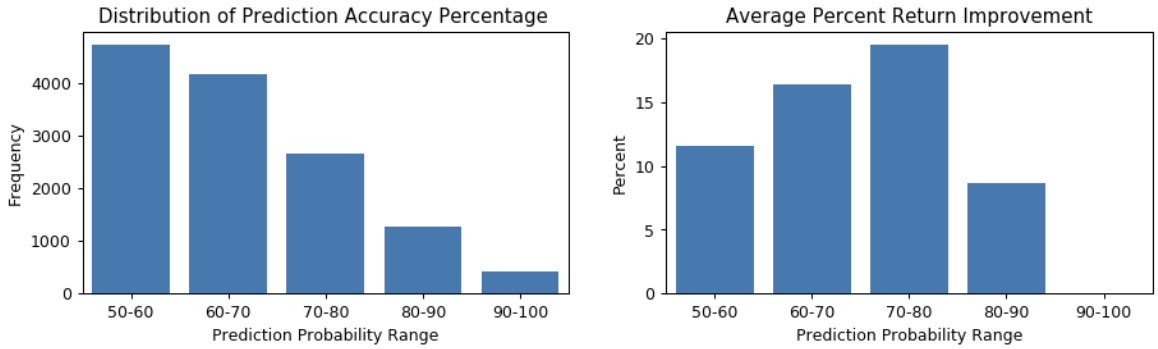

Figure 13

CHAPTER 6

6 Further Study

This study explored using logistic regression, random forest, neural network, extreme gradient boosting, and ensemble models but other models can be explored and it is possible for those to produce better results than the models used in this study. The best-performing model of this study achieved an accuracy of about 68%. Although this is not bad in terms of performance, it is possible that it can be further improved through different methods of parameter tuning, feature selection, or model selection. Besides feature importance measures, there are other ways of feature selection that, just like class balancing methods, can result in varying degrees of success. Principle component analysis (PCA) which seeks to reduce data dimensionality by reducing a set of possibly correlated variables into a set of uncorrelated variables. Variants of PCA, such as principal feature analysis, have also been developed in order to address some of the limitations of PCA. Further exploration into feature selection methods can possibly yield improvements in model performance. Another potential area of interest is parameter tuning as this can greatly alter model behaviour. Techniques such as grid search or random search provide methods to evaluate a pre-set search space of combinations of parameters. These two methods, however, have long run times and other methods have since been developed, such as Bayesian optimization. Exploring the result of these parameter tuning methods could result in insight of which parameters are important to tune and how they affect the performance of the model or speed of training. Furthermore, the results of this study can be used in further analysis of profit optimization. Since the scope of this study is on loan

prediction using various models and comparing methods to handle imbalanced data, further research into methods that can optimize return on investment is left to further research.

CHAPTER 7

7 Conclusion

In the financial sector, the application of artificial intelligence and machine learning techniques with the goal of maximizing return on investment and profit optimization has seen a rise in interest over the years. An increasing number of studies have been conducted in risk management, bankruptcy prediction, and credit scoring systems backed by artificial intelligence. Similar to the nature of these concepts, machine learning also has potential application in loan default prediction. A better understanding of loan defaults, what contributes to the risk of loan defaults and how to predict them, will have tremendous benefits to the lending market industry.

In this paper we explored the use of machine learning techniques applied to loan default prediction. We studied the performance of logistic regression, random forest, neural network, extreme gradient boosting, and ensemble in their capability to predict the outcome of the loan, default or fully paid. In the loan dataset, by nature, loans that were fully paid greatly outnumbered loans that were defaulted. Because these machine learning classifiers are heavily dependent on the distribution of the dataset, this presented us with an imbalanced learning challenge. A classifier trained on imbalanced dataset would be subject to bias toward the majority class at the cost of the minority class. As such, we addressed the problem of imbalanced data classification using and comparing the results of sampling techniques like SMOTE and ADASYN and cost sensitive learning like class weights. Although all balancing methods resulted in model improvement, there was no one method that consistently worked best for every learning algorithm. Since we are working with imbalanced data, the usual methods of describing classifier performance, like

accuracy, could not represent a true representation of performance. Instead, classifier performance was evaluated through the use of the confusion matrix; utilizing metrics such as precision, recall, F-measure, G-mean, precision-recall curves, and ROC. Based on these evaluation metrics, the XGBoost model with class weights showed the best performance but ultimately the ensemble model built through the averaging method was chosen as the best model to use as it performed nearly as well as XGBoost and more robust. Using the results from the predictive results from the best performing model, ROI was shown to be able to be improved by about 83% with potential for further improvement.

8 References

[1] G. Press, “Equifax And SAS Leverage AI And Deep Learning To Improve Consumer Access To Credit,” 20 Februrary 2017. [Online]. Available: https://www.forbes.com/sites/gilpress/2017/02/20/equifax-and-sas-leverage-ai-and-deep-learning-to-improve-consumer-access-to-credit/.

[2] J. Hawkins, “Equifax Receives Utility Patent for Innovative NeuroDecision® Technology,” 11 December 2018. [Online]. Available: https://investor.equifax.com/news-and-events/news/2018/12-11-2018-130158756.

[3] N. Chawla, K. Bowyer, L. Hall and W. Kegelmeyer, “SMOTE: Synthetic Minority Over-Sampling Technique,” *Journal of Artificial Intelligence Research,* vol. 16, pp. 321-357, 2002.

[4] S. F. Crone and F. Steven, “Instance Sampling in credit scoring: An empirical study of sample size and balancing,” *International Journal of Forecasting,* vol. 28, no. 1, pp. 224-238, 2012.

[5] H. He, Y. Bai, E. Garcia and S. Li, “ADASYN: Adaptive Synthetic Sampling Approach for Imbalanced Learning,” *IEEE International Joint Conference on Neural Networks (IEEE World Congress on Computational Intelligence),* pp. 1322-1328, 2008.

[6] H. He and E. A. Garcia, “Learning from Imbalanced Data,” *IEEE Transactions on Knowledge and Data Engineering,* vol. 21, no. 9, pp. 1263-1284, 2009.

[7] M. Kubat and S. Matwin, “Addressing the Curse of Imbalanced Training Sets: One-Sided Selection,” in *Fourteenth International Conference on Machine Learning*, Morgan Kaufmann, 1997, pp. 179-186.

[8] J. Davis and M. Goadrich, “The Relationship Between Precision-Recall and ROC Curves,” *Proceedings of the 23rd international conference on Machine Learning,* pp. 233-240, 2006.

[9] V. S. Desai, J. N. Crook and G. A. J. Overstreet, “A comparison of neural networks and linear scoring modles in the credit union environment,” *European Journal of Operational Research,* vol. 95, no. 1, pp. 24-37, 1996.

[10] C. Strobl, A.-L. Boulesteix, T. Kneib, T. Augustin and A. Zeileis, “Conditional variable importance for random forests,” *BMC Bioinformatics,* vol. 9, no. 1, p. 307, 2008.

[11] C. Strobl, A.-L. Boulesteix, A. Zeileis and T. Hothorn, “Bias in Random Forest Variable Importance Measures: Illustrations, Sources and a Solution,” *BMC Bioinformatics,* vol. 8, no. 1, p. 25, 2007.

[12] G. Zhang, M. Y. Hu, B. E. Patuwo and D. C. Indro, “Artificial neural networks in bankruptcy prediction: General framework and cross-validation analysis,” *European Journal of Operational Research,* vol. 116, no. 1, pp. 16-32, 1999.

[13] G. P. Zhang, “Neural networks for classification: a survey,” *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews),* vol. 30, no. 4, pp. 451-462, 2000.

[14] T. Chen and C. Guestrin, “XGBoost: A Scalable Tree Boosting System,” in *Proceedings of the 22nd ACM Sigkdd International Conference on Knowledge Discovery and Data Mining*, San Francisco, CA, USA, 2016.

[15] G. Wang, H. Jinxing, J. Ma and H. Jiang, “A comparative assessment of ensemble learning for credit scoring,” *Expert Systems with Applications,* vol. 38, no. 1, pp. 223-230, 2011.