ABSTRACT

OWEN, HAILEY MARKAY. A Machine Learning Approach to Predict Loan Default. (Under the direction of Hien Tran.)

In a world of increasing reliance on technology and a culture that wants fast results at the tip of their finger, Lending Club emerges as a fast and mutually beneficial tool to provide borrowers with unsecured personal loans without the need of interaction with a financial institution and investors with an opportunity to increase personal wealth by collecting interest on these loans. In this dissertation, we will look at the results given by Decision Tree models, which aim to predict the time until default on 36 month term loans supplied by Lending Club. First, we will explore loan default as a binary classification problem. Then, we will expand these results to see how long it will take for a loan to default, using a Multi-Class Decision Tree. We change the question slightly to determine where in the term of the loan the lender will regain their investment, and whether the loan default before or after this point. Finally, we will predict what percent gain the investor will receive regardless of the status of the loan.

A Machine Learning Approach to Predict Loan Default

by
Hailey Markay Owen

A dissertation submitted to the Graduate Faculty of
North Carolina State University
in partial fulfillment of the
requirements for the Degree of
Doctor of Philosophy

Applied Mathematics

Raleigh, North Carolina

2019

APPROVED BY:

Alen Alexanderian

Negash Medhin

Kevin Flores

Hien Tran
Chair of Advisory Committee

DEDICATION

To my husband, Jared, who always believed in me, even on days I didn’t. This is for you. I love you.

BIOGRAPHY

Hailey Owen was born 16 June 1989 in Cleveland, Tennessee. She spent her entire childhood in Cleveland through her undergraduate education at Lee University. She began college with a love of math, but no career direction. She graduated with a B.S. in Mathematics Education and began looking for a career in teaching. Shortly after graduation, she and her husband, Jared Owen, moved to Princeton, New Jersey where she began to teach grade 5 and 6 mathematics at Princeton Academy of the Sacred Heart while he completed his graduate degree. After Jared's graduation, they moved to Raleigh, North Carolina where she enrolled at North Carolina State University in the Applied Mathematics Ph.D. program. After completing Qualifying exams, she focused her attention on research in machine learning and predictive analytics.

Still having a heart for education, she accepted an internship in SAS EVAAS in the spring of 2017, which became a full-time job in spring of 2019. There she is able to use her love of math and passion for teaching to help teachers interpret student growth to better their teaching and student progress. She is excited to continue with EVAAS after graduation.

ACKNOWLEDGEMENTS

This journey would not have been possible without the unconditional support I have had from my family, my husband and my parents, who have encouraged me and watched Anderson during my long days of studying and writing. I owe all my success to you. Anderson, you came into this world during my second year of study, your existence has pushed me to pursue my dreams so that one day I can help you pursue yours. I love you and you are always my reason.

My friends that have become family, you have all helped in more ways than you know. Some of you offered mathematical help and others offered stress relief. Glenn, I owe you much gratitude for putting up with my constant doubt and questions. Thank you for your support.

My coworkers at EVAAS, you have made it possible for me to feel successful both professionally and as a student. Thank you for allowing me the time to do it all. I am grateful, and hope to support you all as you have me.

Finally, I thank my committee members, and most of all Dr. Hien Tran. Your guidance and patience has made all the difference.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER

1 INTRODUCTION

Peer-to-peer lending is the practice of borrowing or lending money from one person or company to another without going through a bank or other financial institution. Specifically, Lending Club is a peer-to-peer lending platform, headquartered in San Francisco, California [21]. This lending website pairs borrowers with investors. Customers apply for an unsecured personal loan online. Borrowers give information about their income, purpose for the loan, and debt. Lending Club collects further information about the borrower such as credit score and credit history then assesses the risk of funding the loan. If the loan is accepted, Lending Club determines the grade of the loan which assigns the interest rate and fees that must be paid to Lending Club. Investors can then search through the loan listings to find loans in which they are comfortable with investing based on information about the borrower, loan amount, grade, and purpose of the loan. Investors make money by collecting the interest from the loans. Currently, interest rates can range from 5.31% to 26.77% depending entirely on the grade that was assigned to the loan [22]. Investors can choose to fully fund a loan or partially fund a loan. It is required that investors fund at least $25 in one loan and invest at least $1,000 in total, but this could be spread over multiple loans. Borrowers can pay back the loan fully at any time without penalty [23].

For borrowers that become delinquent throughout the term of their loan, Lending Club reaches out via email, phone, and letter to collect past due payments and bring the loan back to a âĂIJcurrentâĂİ status. After 121 days of failure to pay, a loanâĂŹs status is changed to default by Lending Club and the loan is charged off when there is no âĂIJreasonable expectation of further paymentsâĂİ

[24] to Lending Club, usually within 150 days of being past due. This could happen sooner if the borrower declares bankruptcy.

Investors have only some details about the loan to make a decision about investing. Knowing if a person will default and when that could occur could make the decision process less daunting. In this paper, we will focus on trying to predict the time until charge off. We will do this in several ways. First, we will use a Decision Tree to predict if a loan is charged off, does it happen in the first, second, or third year of the loan. Second, in an attempt to attain more applicable information to the investor, we will use a second Decision Tree Model to predict if the loan would be charged off before or after the principal loan amount was paid back to the investor. Finally, we will predict the percent return on a loan. This is different than all other models run in our research. In this model, we will use a continuous label and employ a Regression Tree Model. Using this information, lenders, with some certainty, can choose investments that might default, but will be profitable before the loan goes into default.

1.1 What is Machine Learning?

Dr. Yoshua Bengio, UniversitÃľ de MontrÃľal, defines machine learning research as a part of research on artificial intelligence, seeking to provide knowledge to computers through data, observations and interacting with the world. That acquired knowledge allows computers to correctly generalize to new settings. Machine learning is based on the idea that systems can learn from data by identifying patterns within the data. This is done by determining a target function that is an approximation to the relationship between the input data and its label, in the case of supervised learning. This is all done in hopes that the system will make decisions with minimal human intervention. In general, machine learning algorithms consist of the following:

- Representation âĂŞ Classifiers, i.e. Classification and Regression Trees, Neural Networks, Support Vector Machine, and clustering

- Evaluation âĂŞ Accuracy/Error Rate, Confusion Matrix, Information Gain, Sensitivity, Specificity

- Optimization âĂŞ Search Methods, i.e. Gradient descent, Greedy search

Because of recent advancements in methods, research, and computing environments, machine learning today is not like that of the past. The ability to automatically apply calculations in less and less time with more and more data has transformed the older idea that machines can learn from patterns within data into what we see today, machines making real time predictions based on real time data. These advancements have given rise to the applications of machine learning used in everyday life.

1.1.1 Supervised vs Unsupervised

There are 3 main ways that classification algorithms operate- supervised, unsupervised, and reinforcement learning. Supervised learning is done using labeled outputs. The goal of supervised learning techniques is to learn a function that, given a sample of data and desired (labeled) outputs, best approximates the relationship between input and outputs that are observable in the data. More explicitly put, the data has a training set that is a complete set with input and labels attached. This complete data has both features and labels for each observation. The supervised learning algorithm looks at this complete data to find a pattern. This pattern is used to classify future data either for determining an accuracy measure using the testing set, which also is complete data or classifying new data that does not have a label. Given a set N of training samples $(x_1, y_1)...(x_N, y_N)$ where x_i is a feature vector and y_i is the associated label, it is assumed that there exists a function f such that $y = f(x)$ for each of the observes data. The goal of the learning algorithm is to find the best approximation g to f so that it can be applied to new data and assign labels. The function g is often called a classifier [14].

Machine learning algorithms that use this method are support vector machines, decision trees, k-nearest neighbor, and neural networks. Support vector machines build a hyperplane or hyperplanes to classify data. This idea can be seen in figure 1.1, where H_0 and H_1 are the hyperplanes, and $\frac{2}{||\vec{w}||}$ is the distance between the two hyperplanes that is to be maximized to find the optimal H_0 and H_1. The hyperplane that has the largest distance to the nearest data point of any class is chosen. This maximizes the separation and ideally achieves the lowest error rate. A decision tree classifier, which is displayed in figure 1.2, has a root node, which represents the entire population or sample of data. This root node gets divided into two or more sets depending on the split and feature that best minimizes the error. This process continues and creates decision nodes until either the node has become pure or a preset threshold has been reached. When splitting stops a leaf or terminal node is created. This process is explained in great detail in chapter 4. The k-nearest neighbor algorithm clusters the data to determine similarities. It then makes decisions on new data based on labels within each cluster. In figure 1.3, it can be seen how the algorithm clusters the data. Finally as seen in figure 1.4, a neural network makes decisions with layers. Data are passed between layers to adjust its internal weightings and ultimately minimize classification error.

Unsupervised learning is done with data that does not have labels. The goal is to infer the natural structure present within a set of data points to make classifications. The two main machine learning algorithms that use this method are principal component and cluster analysis. In principal component analysis, as illustrated in figure 1.5, the dimension of the data is reduced, but maintains as much of the complexity of the data as possible. The most significant principal components are selected by looking at how much of the data's variance they capture, as can be seen in figure 1.6. Then, by using the first few dimensions of the remapped space only, we can gain understanding of

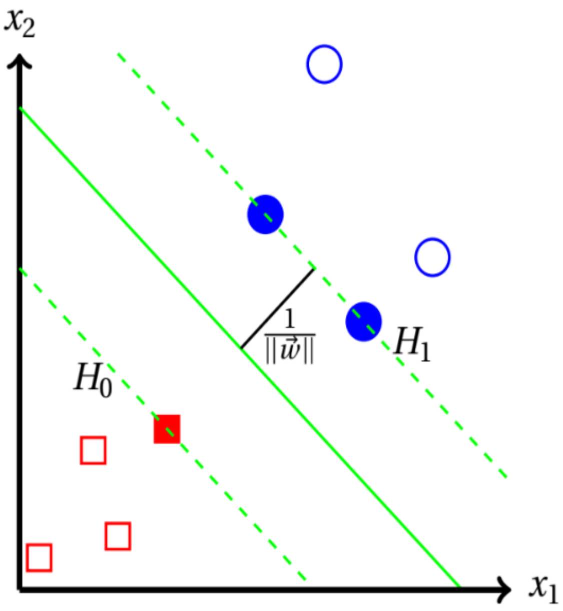


Figure 1.1 An Example of a linearly separable binary problem with support vector machines. Support vector machines build a hyperplane or hyperplanes to classify data. The hyperplane that has the largest distance to the nearest data point of any class is chosen. This maximizes the separation and ideally achieves the lowest error rate [35].

the data set's organization. Cluster analysis is a broad topic that can be accomplished many different ways. Clustering inherently uses the structure of the data to understand how data is similar, as can be seen in figure 1.7. As explained above, clustering groups data by similarities and then uses those similarities to gain understating about new data. Assuming the new data is within some distance to existing data, new data can be classified the same way.

Semi-supervised learning allows for some labeled observations in the training set and some that are not. Due to cost of labeling with some projects, fully labeled data may be near impossible, but some labels can drastically improve results. It can be seen in figure 1.8, that the black and white points are labeled while the gray points are unlabeled. From the picture, we can tell that the unlabeled points along with the label points can provide information about the structure of the data that would not be available without the few labeled data points.

Reinforcement learning refers to a goal-oriented algorithm and is used when decisions need to be made in a sequential order. The input for the model is the initial state from which the model will begin. Each decision is made based on the output of the previous decision. These algorithms are penalized when making a wrong decision and rewarded when making a right decision. An example

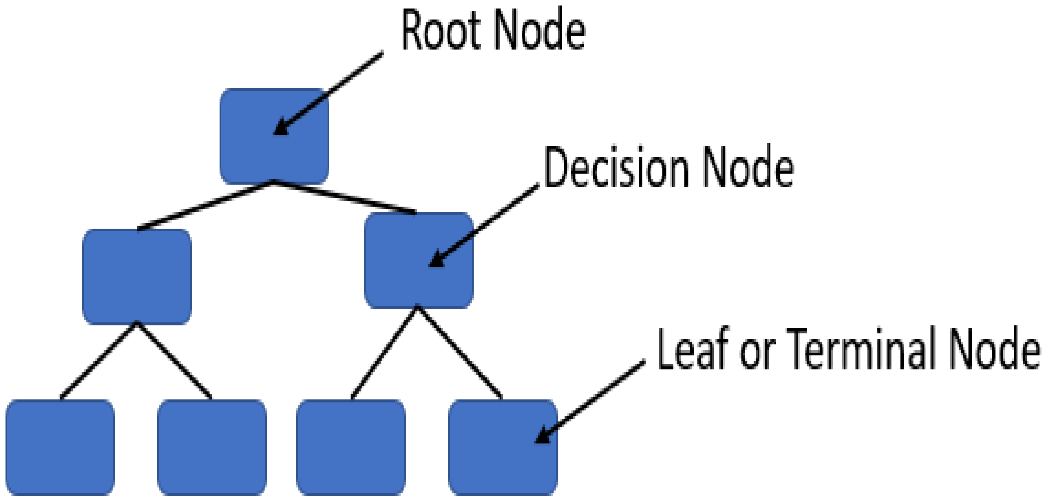


Figure 1.2 A decision tree classifier has a root node, which represents the entire population or sample of data. This root node gets divided into two or more sets depending on the split and feature that best minimizes the error. This process continues and creates decision nodes until either the node has become pure or a preset threshold has been reached. When splitting stops a leaf or terminal node is created.

of an objective function for reinforcement learning is

$$\Sigma_{t=0}^{t=\infty}\gamma^t r(x(t),a(t)), \tag{1.1}$$

where r is the reward function and t the time or steps it takes to reach the goal. x is the state at the given time and a is the action taken in that state. The objective function calculates all rewards that could be obtained by running all possible decisions. γ^t is defined as a discount factor and is multiplied by the reward function. It is designed to make future rewards worth less than immediate rewards. Reinforcement learning judges decisions that are made by results. The goal is to maximize the objective function and determine the best path to the goal. An example of a real world application would be a robot who's goal is to travel from point A to point B. Every inch that the robot moves closer to its destination could be counted as points thus minimizing the distance the robot travels from point A to point B.

1.1.2 Applications

Machine learning is used by many industries and has numerous real world applications that have significant impacts on the world around us. Figure 1.9 illustrates the methods and models used for applications of machine learning, though these are just a few.

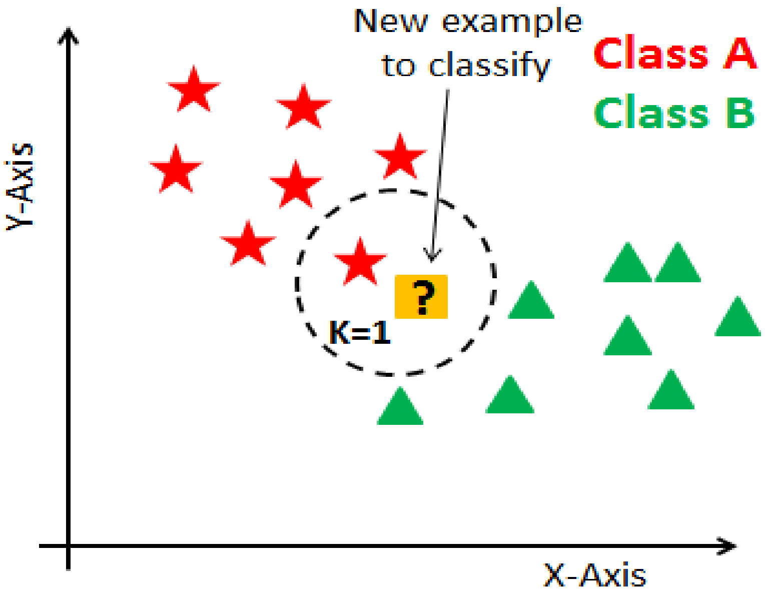


Figure 1.3 k-nearest neighbor algorithm clusters the data to determine similarities. The model then makes decisions on new data based on labels within each cluster. The cluster size depends on what integer K is determined to be. K is frequently determined by choosing the K that minimizes the error rate [30].

A small number of these applications that deserve mention because how they impact our everyday life are-

- Voice Recognition Software - Virtual Personal Assistants
- Navigation âĂŞ GPS
- Advertisement - Product Recommendations
- Face Detection âĂŞ Video Surveillance
- Search Engine Results âĂŞ Google
- Medical Predictions âĂŞ Epidemic Outbreak Prediction
- Fraud Detection - Distinguish between legitimate or illegitimate transactions

Siri and Alexa are popular applications of the virtual personal assistants. These applications use voice commands to find information. Machine learning is an important part of these personal assistants as they collect information about our lives and preferences and use that information in their later involvement with us. Typically, Deep Learning which usually involves the use of Neural

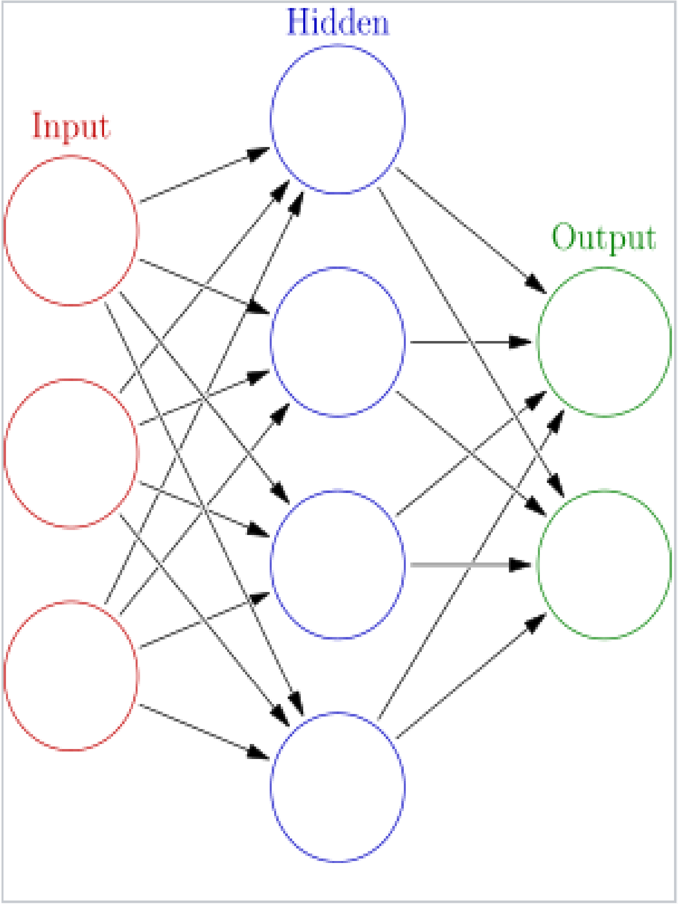


Figure 1.4 A neural network makes decisions with layers. Data are passed between layers to adjust its internal weightings and ultimately minimize classification error. The input layer contains the entire sample or population. There can by many hidden layers depending on how deep the network needs to be achieve accuracy. The output layer contains predictions or decisions made by the model.

Networks are employed to allow these virtual assistants to learn how to interpret what the user wants.

In navigation, recent advancements in machine learning have allowed the GPS to make traffic predictions. While we use the GPS, our current location and velocity, along with many other pieces of information, are recorded and processed along with thousands of other people to predict traffic and estimated time until arrival. Many methods discussed above are used in traffic pattern prediction.

Fraud detection algorithms are used to distinguish between legitimate and illegitimate transactions. Paypal being one example of a company using machine learning methods to protect against users laundering money using their services. The company compares millions of transactions using machine learning methods to distinguish between transactions that are considered normal and abnormal.

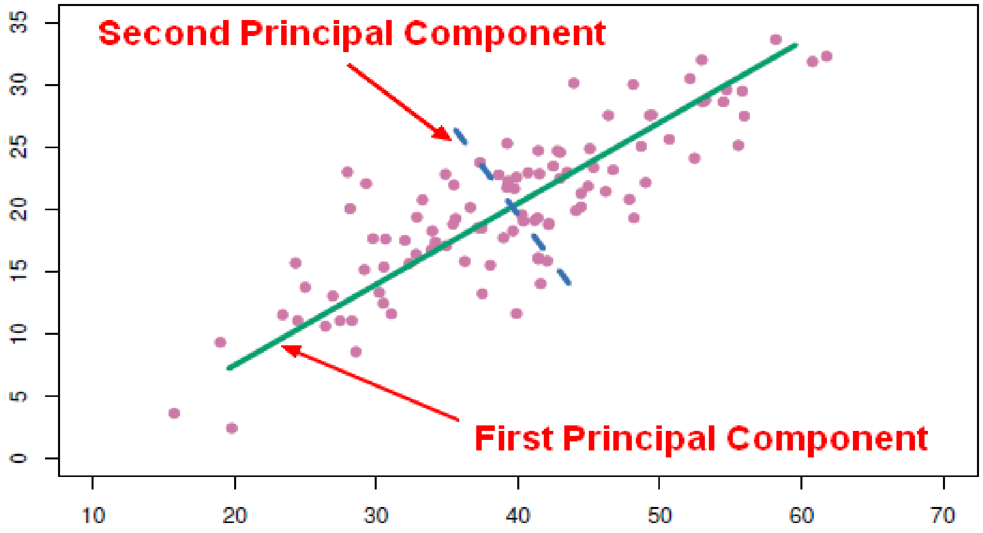


Figure 1.5 The direction with the most variance in the data is represented using the green line. This is the first principal component. The blue dotted line represents the second principal component and it is the direction that varies the most but is uncorrelated to the first component [9].

These among so many others are applications of machine learning that are changing the world in which we are living. In this dissertation, we discuss one other such domain where machine learning is already making a difference by providing more information to investors about potential loans.

1.1.3 Difficulties in Machine Learning

There are several difficulties that have to be overcome when using machine learning methods. Some methods have more limitations than others, but it general most machine learning methods struggle when presented with a class imbalance or missing data. Many methods can overfit the data if not pruned or stopped by defining parameters this can cause models that are too complex and variance in the results of the model.

Most machine learning algorithms work most efficiently when the number of instances of each class are roughly equal. When there is a class imbalance, the algorithm typically learns quickly that the error can be minimized easily by always choosing the larger class. These results are not helpful to the analyst. There are a few ways to overcome this issue and several of them are explored extensively in this dissertation. Over and under-sampling techniques are used either multiply the minority class or classes or take away from the majority class or classes. This is usually done by random sampling. Synthetic Minority Over-sampling Technique (SMOTE) is used to create new data by clustering old data and borrowing features from similar observations to add new records to the minority class or

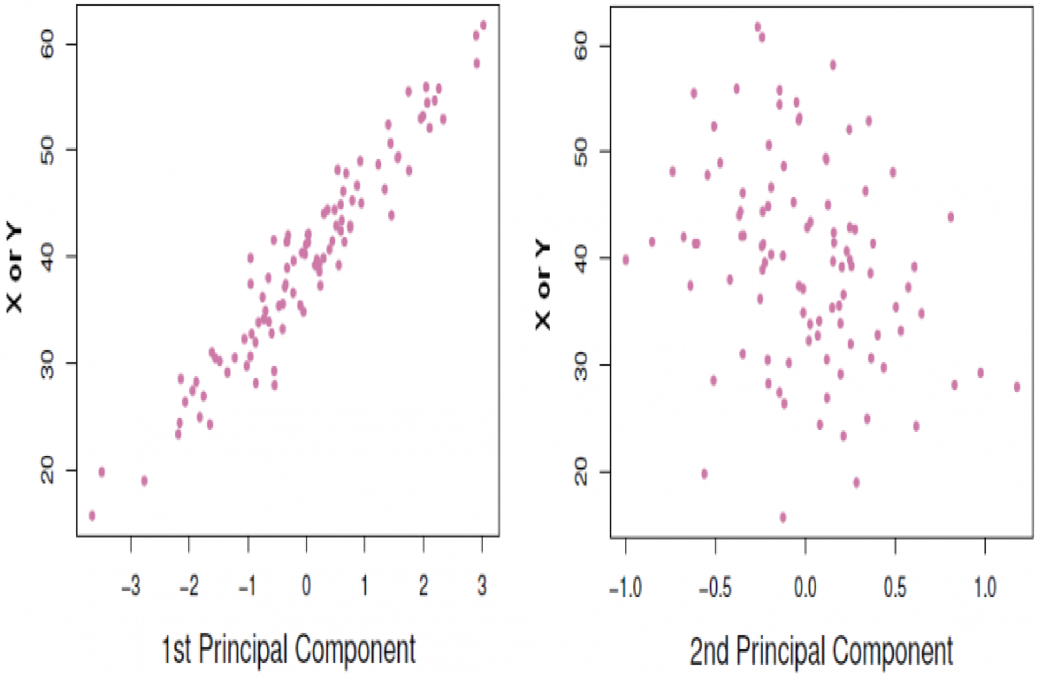


Figure 1.6 The first principal component is highly correlated with both the features in this example. The second has less correlation. As in this example, when the features are highly correlated predictions can be made with one new variable, the first principal component, because it can summarize the data well. Thus the dimension of the data is reduced [9].

classes. These methods allow the algorithm to not make decisions based on popularity of one class, but to minimize the error as they are designed to do.

The problem of having missing data is inevitable when working with real world data. Missing data can be caused by a wide variety of reasons including human error during data entry, humans withholding information on a survey or form, incorrect sensor readings, or software bugs. There are also many methods to combat this issue inducing- deletion, mean imputation, clustering methods, and Expectation Maximization. Deleting observations that have missing data is an easy way to avoid the problem all together if there are minimal missings within the data. This method causes deletion of valuable information if the problem is wide spread and not a good option for those cases. Mean imputation is a method where each missing data point is filled in with the mean of the feature it is in. Hot-Deck imputation is a clustering method that is described in detail later in this paper. Expectation maximization is the method that was ultimately chosen as the best method for our data in this dissertation, and involves an iterative method to find the maximum likelihood estimate for incomplete data.

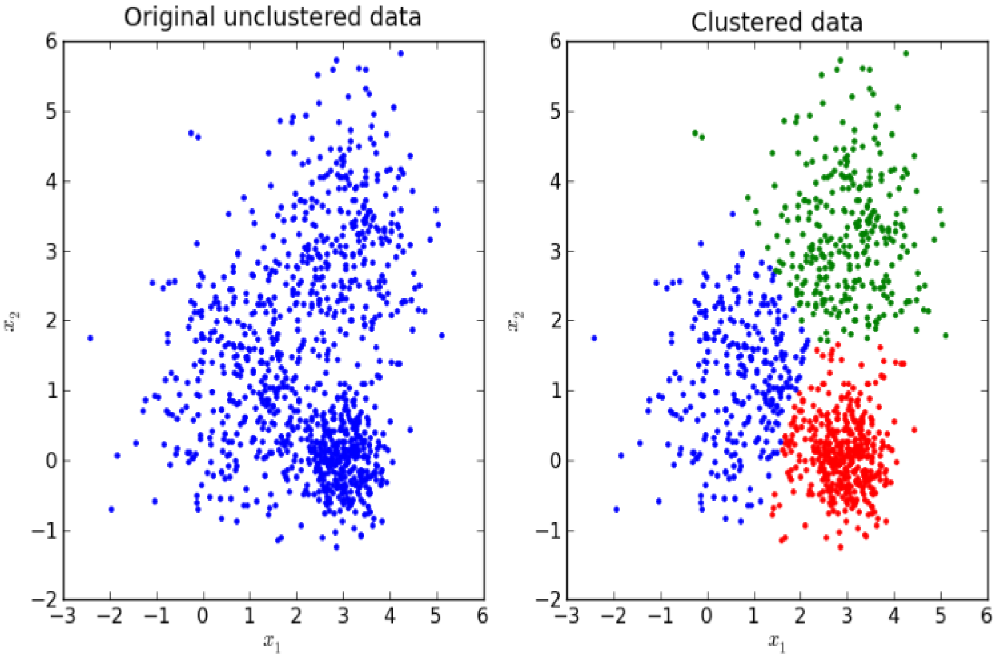


Figure 1.7 The image on the right was created using the K-Means method for clustering.The goal of the model is to determine groups of data. The number of groups is predetermined by the analyst. To produce these results, first, pick random points as cluster centers, which are called centroids. Second, calculate each inputs distance to each centroid and assign to the nearest. Third, take the average of assigned data points for each cluster and find the new center. Fourth, repeat steps two and three until the center of the clusters no longer change [19].

Overfitting often occurs when the model is more complex than necessary. The algorithm fits the training data far too closely to generalize results for any new data. It can be seen in figure 1.10, the green curve separates the blue and red data points perfectly. These points are part of the training set. While the error using the green line for the training set will be zero, the testing set or new data likely will not look the same and result in more error than if the line was more general. This is an example of an over fit classifier. The black line is still separates the training data well, and is far more general than a line that fits the data perfectly [16].

1.2 Prior Work

Many methods to predict default in loans have been purposed throughout the literature. Among the most popular are traditional statistical methods (e.g. logistic regression [36]), nonparametric statistical models (e.g. k-nearest neighbor [15] and classification trees [10]) and neural networks

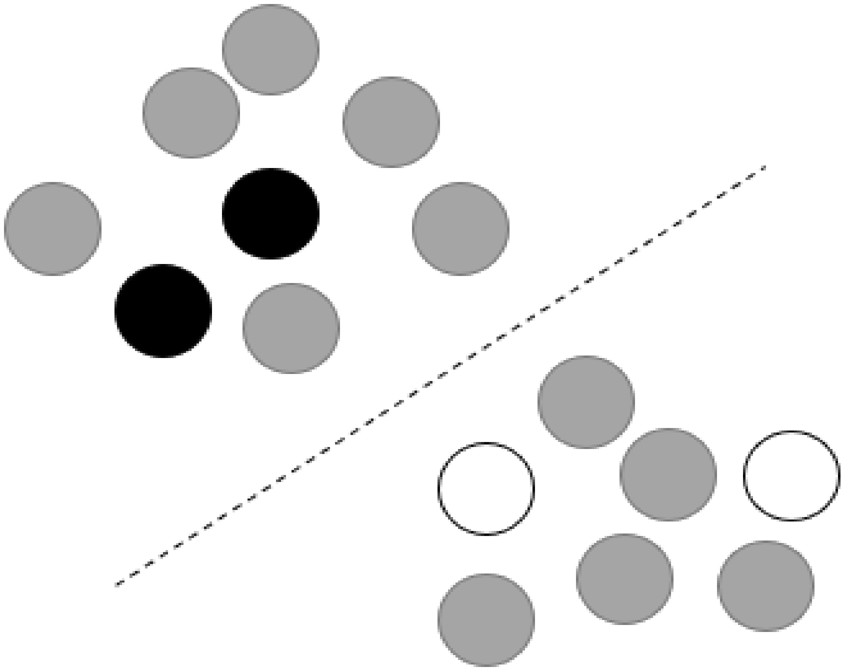


Figure 1.8 An example of a semi-supervised model. The black and white points are labeled while the gray points are unlabeled. From the picture, we can tell that the unlabeled points along with the label points can provide information about the structure of the data that would not be available without the few labeled data points.

[12]. We have chosen to focus our paper on classification and regression tree models and their adaptations. We do this because, while they are generally able to achieve high predictive accuracy rates, reasoning behind how they reach decisions is interpretable and applicable to both borrowers and lenders [28].

Classification trees were first introduced as methods to classify binary outcomes, but have been used for multi-class problems. In "Solving Multiclass Learning Problems via Error-Correcting Output Codes" [13], Dietterich et al., generalize the decision tree algorithm to handle multi-class problems. They take this approach on datasets with 6 classes up to 26 classes.

Much of the previous work has focused on predicting a binary split of loans that will be paid-in-full or will default. This is done frequently in literature and sometimes as class projects. This is an easy baseline problem from which to start. One paper that successfully makes this prediction is "Analysis of Default in Peer to Peer Lending" [33]. In this paper, Ramirez explores loan default as a binary classification problem. He builds a decision tree classifier and evaluates performance based on binary classification metrics. His error rate is 22.2% for his single tree model.

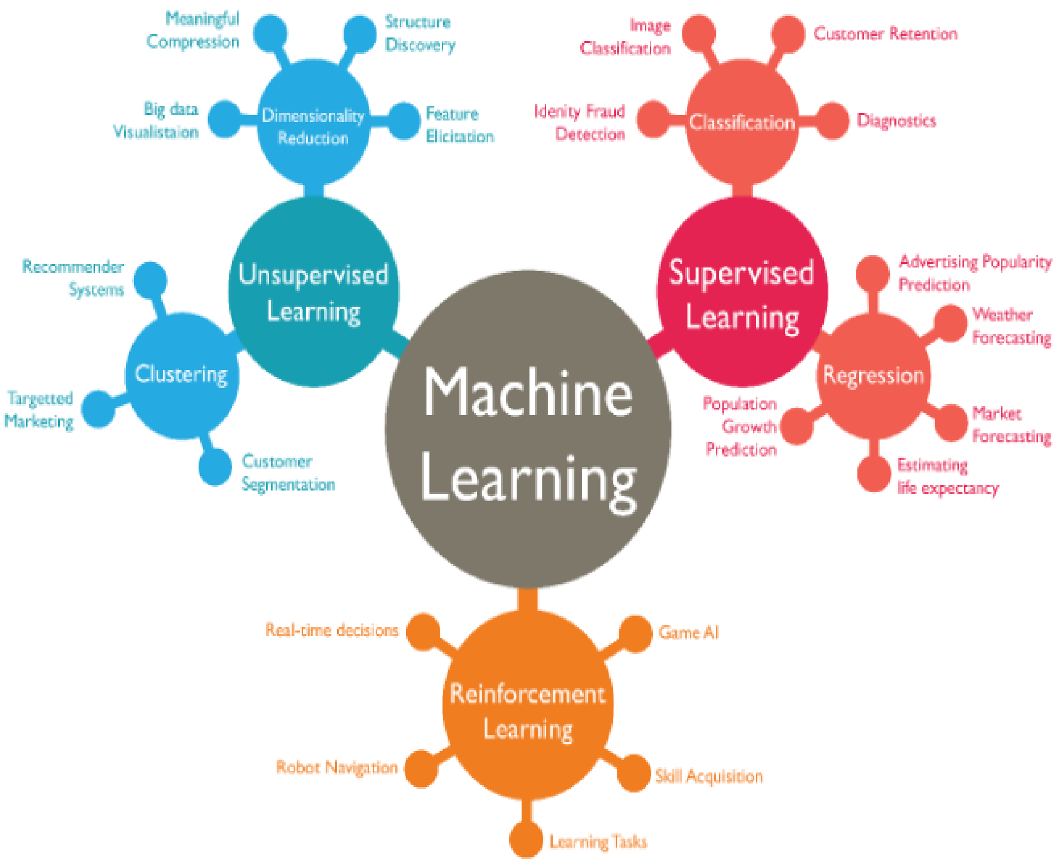


Figure 1.9 Example of Machine Learning Applications [17] for unsupervised, supervised, and reinforcement learning models. This map shows how each of these types of machine learning might be used in practice.

In "A Data-Driven Approach to Predict Default Risk of Loan for Online Peer-to-Peer (P2P) Lending" [18],the authors use Lending Club data and several multi-class models to predict if a loan will default, need attention, or be paid and compare results. Their classification/prediction models include neural networks, decision trees, and support vector machines (SVM). They compare the accuracy of these methods to conclude that SVM achieved the best performance, but only slightly. This prediction is similar to our baseline prediction though it does use the idea of a multi-class classifier to make a more robust prediction. The novelty of this paper is predicting default times using classification trees, but until now much of this work was being done using survival analysis. Narain [29] was the first author to use parametric accelerated failure time (AFT) survival method. He used data that contained 1242 borrowers with 24 month loan terms. He obtained results that were reasonable. Several authors followed his lead and started to use these techniques to make

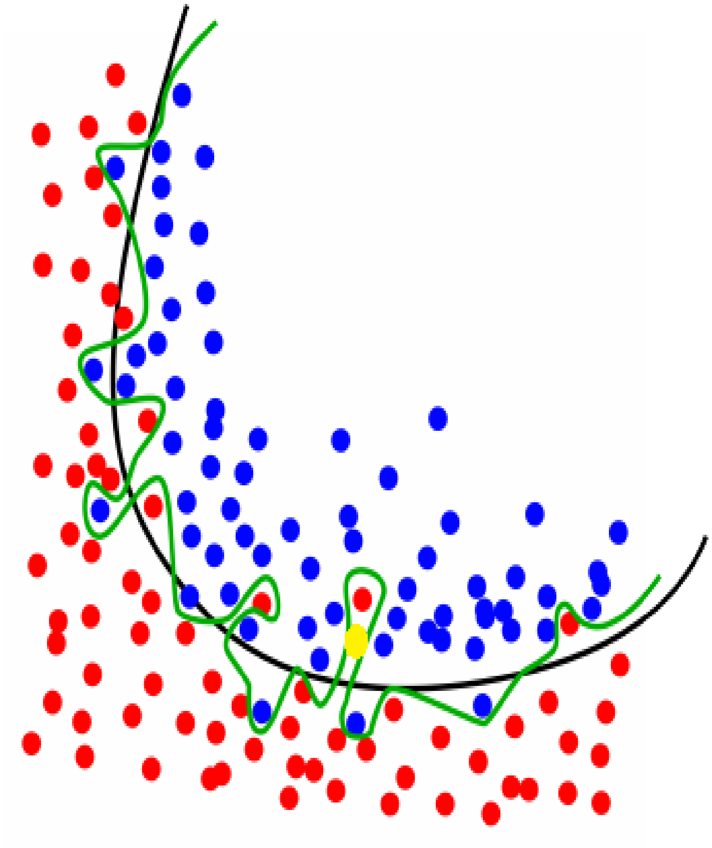


Figure 1.10 This is an example of overfitting. The green curve separates the blue and red data points perfectly. These points are part of the training set. While the error using the green line fore the training set will be zero, the testing set or new data likely will not look the same and result in more error than if the line was more general. This is an example of an over fit classifier. The black line is still separates the training data well, and is far more general. The yellow data point shows the situation where a new data point will be classified differently based on overfitting of the training data.

their predictions. In "Not if but when will borrowers default", Banasik et al. [3] purposed using an alternative to the AFT model. His work focused on using the Cox proportional hazard model for its flexible non-parametric baseline hazard. His work used a data set with 50,000 loan applications. The data was analyzed using the non-parametric proportional hazards model (no baseline hazard assumption), two parametric proportional hazards models using exponential hazards and Weibull baseline hazards, and an ordinary logistic regression approach [2].

The successes that were found in previous work, using decision trees in creative ways, set the stage for us to predict time until default using multi-class decision trees.

1.3 Overview of Contributions

While cleaning the data, imputation, and feature engineering are not novel, the way in which we employ these methods gives us the ability to apply models to data in a novel way. In literature, we find countless papers and projects that do a binary decision tree on loan data to determine if a loan will default or be paid-in-full. The accuracy of our models are better than most found in the literature. These comparisons will be made in chapter 5. The uniqueness in this dissertation comes in using multi-class classification trees to predict time until default. In the field, similar predictions are made using survival analysis. We compare our results to a survival analysis model in chapter 5. We finally predict percent gain using a regression tree. We have found in the literature this being done with random regression forests, which is similar to our model and has comparable results.

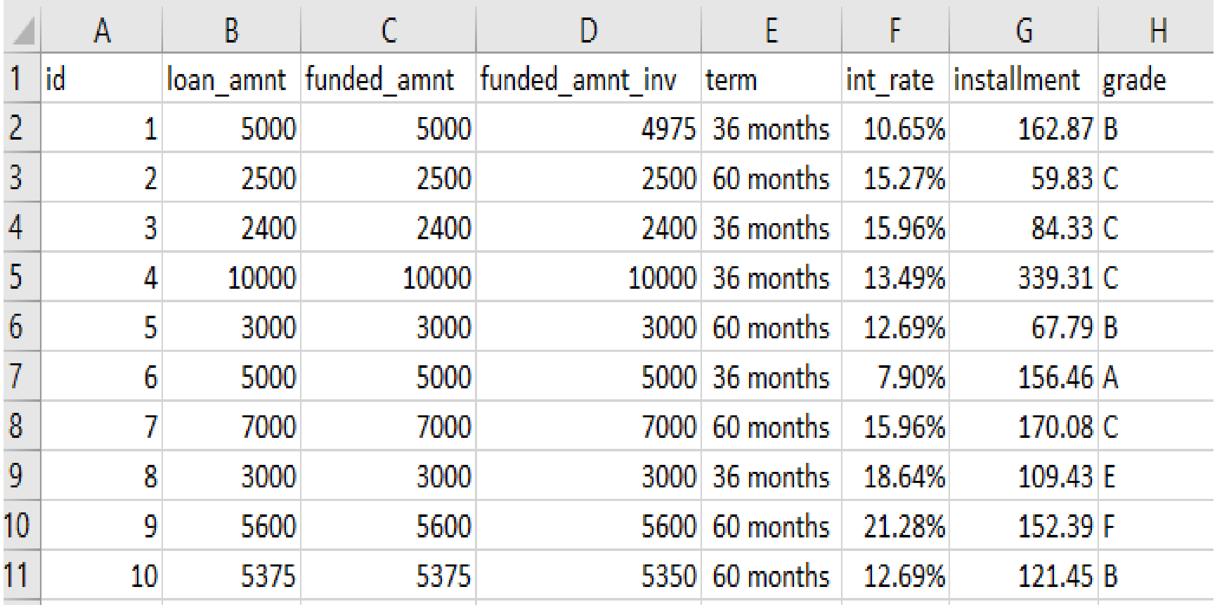
CHAPTER

2 DATA PREPARATION METHODS

Data cleaning is comprised of detecting and modifying or deleting inaccurate, incomplete, or irrelevant records or features from a data set. Lending Club's data had many irregularities and missing values as most of the features are self-reported and some information is not mandatory on the application. Also, because these data sets come from different years, regulations change which results in differing or missing entries and inconsistent features. Some, but not all, of these barriers were overcome within the cleaning and preparation process. Another reason for cleaning and managing the data is to fit the input for the model in the most efficient way possible. Some features were recoded to reduce computing time.

2.1 Cleaning

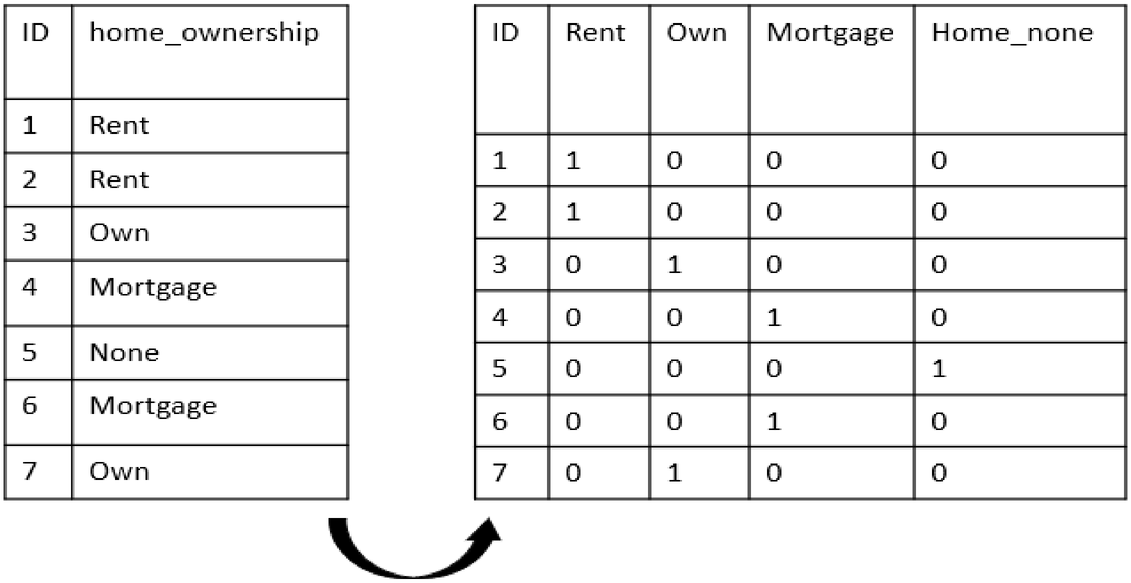
The data from Lending Club is publicly available and comes in .csv format in multiple tables. Years 2007-2011, 2012-2013 and the data dictionary were downloaded for this research. After combining all data sets there was a total of 123,386 loans. A sample of data can be seen in figure 2.1. We wanted to keep all data where the loan had either been paid-in-full or defaulted. We read in and cleaned this data using SAS 9.4. Initially, we read in and set these tables together to create one large database. After investigation, we dropped all 60 month term loans. The volume of these longer terms loans is far less than 36 month term loans. Lending Club began issuing 60-month term loans in 2010 on a very small scale. Until recently these initial loans had not reached maturity, which is one of our qualifications for analysis, thus providing very little data for analysis. After dropping all 60-month term loans, approximately 11% of loans were default and 89% were paid-in-full. We then found all the features where at least 50% of entries were missing and dropped those out of the study. We also removed 23 features that had only 31% of their entries missing, but this occurred in the same 54,084 records. The pattern here was too much to meaningfully impute this data. We also removed rows that did not appear accurate, such as if the loan amount was missing. We then found all features that were created after origination of the loan as these features were not helpful in our research and dropped those as well. You can see in Table 2.9 the features that were dropped and the reason. All features not mentioned in the table were used in analysis or used to create a feature used in analysis.



	A	B	C	D	E	F	G	H
1	id	loan_amnt	funded_amnt	funded_amnt_inv	term	int_rate	installment	grade
2	1	5000	5000	4975	36 months	10.65%	162.87	B
3	2	2500	2500	2500	60 months	15.27%	59.83	C
4	3	2400	2400	2400	36 months	15.96%	84.33	C
5	4	10000	10000	10000	36 months	13.49%	339.31	C
6	5	3000	3000	3000	60 months	12.69%	67.79	B
7	6	5000	5000	5000	36 months	7.90%	156.46	A
8	7	7000	7000	7000	60 months	15.96%	170.08	C
9	8	3000	3000	3000	36 months	18.64%	109.43	E
10	9	5600	5600	5600	60 months	21.28%	152.39	F
11	10	5375	5375	5350	60 months	12.69%	121.45	B

Figure 2.1 This figure shows an example of the data used in this project. There are 10 records in this example and a small subset of the features used in each model.

Several features came in the data sets as categorical. Features that were categorical and had no sense of order or ranking were coded with a one-hot representation. The one-hot representation creates a new feature for each category within a current feature. An example of this can be seen in figure 2.2. The category that applies to each record is indicated with a 1 and all others 0. An example of this, in the data, occurs with the feature purpose. This tells us, in 14 categories, the intent of the borrower. All 14 categories were made into separate features all consisting of 0's or 1's. Each record only has one of these 14 categories flagged as the intended use of the resource. You can see in table 2.1 the distribution of the records within the feature purpose.



ID	home_ownership
1	Rent
2	Rent
3	Own
4	Mortgage
5	None
6	Mortgage
7	Own

ID	Rent	Own	Mortgage	Home_none
1	1	0	0	0
2	1	0	0	0
3	0	1	0	0
4	0	0	1	0
5	0	0	0	1
6	0	0	1	0
7	0	1	0	0

Figure 2.2 These two tables show the progression per observation that happens to apply a one hot encoding.

Table 2.1 Loan Distributions by Loan Purpose

Purpose of Loan	Number of Loans	Total Percent	Percent Charged Off
Car	2670	1.52	0.15
Credit Card	39638	22.60	2.40
Debt Consolidation	95984	54.71	7.08
Educational	405	0.23	0.05
Home Improvement	10014	5.71	0.64
House	1099	0.63	0.08
Major Purchase	4797	2.73	0.26
Medical	1929	1.10	0.18
Moving	1468	0.84	0.13
Other	10613	6.05	0.97
Renewable Energy	186	0.11	0.02
Small Business	3492	1.99	0.45
Vacation	1192	0.68	0.10
Wedding	1939	1.11	0.12

The one-hot representation creates challenges if the number of categories within a feature is large, such as the state in which a borrow lives. If a feature like state is recoded to a one-hot representation, running the decision tree model becomes far too time consuming. To combat this issue, we grouped categories together in convenient ways. The feature state was converted

Table 2.2 Loan Distributions by Grade

Grade of Loan	Number of Loans	Total Percent	Percent Charged Off
A	37555	21.41	1.21
B	66782	38.07	4.06
C	40937	23.34	3.71
D	23442	13.36	2.73
E	5327	3.04	0.70
F	1124	0.64	0.18
G	259	0.15	0.05

Table 2.3 Loan Distributions by Employment length

Current Employment Length	Number of Loans	Total Percent	Percent Charged Off
Less than 1 year	14784	8.43	1.10
1 year	12200	6.95	0.87
2 years	16628	9.48	1.19
3 years	14467	8.25	1.04
4 years	11346	6.47	0.80
5 years	13604	7.75	0.96
6 years	10929	6.23	6.79
7 years	9788	5.58	0.73
8 years	7915	4.51	0.57
9 years	6259	3.57	0.44
10+ years	49947	28.47	3.32
Did not answer	7559	4.31	0.79

to the feature region. This was done by grouping states into 9 regions then applying the one-hot representation to the new feature and dropping the original feature of state. These regions were determined by the Census Bureau [7] and can be seen in figure 2.3.

Features that were categorical, but had meaning when ranked, such as grade, were made numeric by giving each category a meaningful numeric entry. You can see in table 2.2 the distribution of records by grade. The grade of your loan directly impacts your interest rate. Grade A borrowers will have the lowest interest rate. These borrowers have been deemed less risky by Lending Club.

2.2 Label Creation

We created four label features to use in our four different model runs. The first label was the binary label that represented if the loan was defaulted or paid-in-full. We were given the status of the loan within the data set as the variable loan_status.

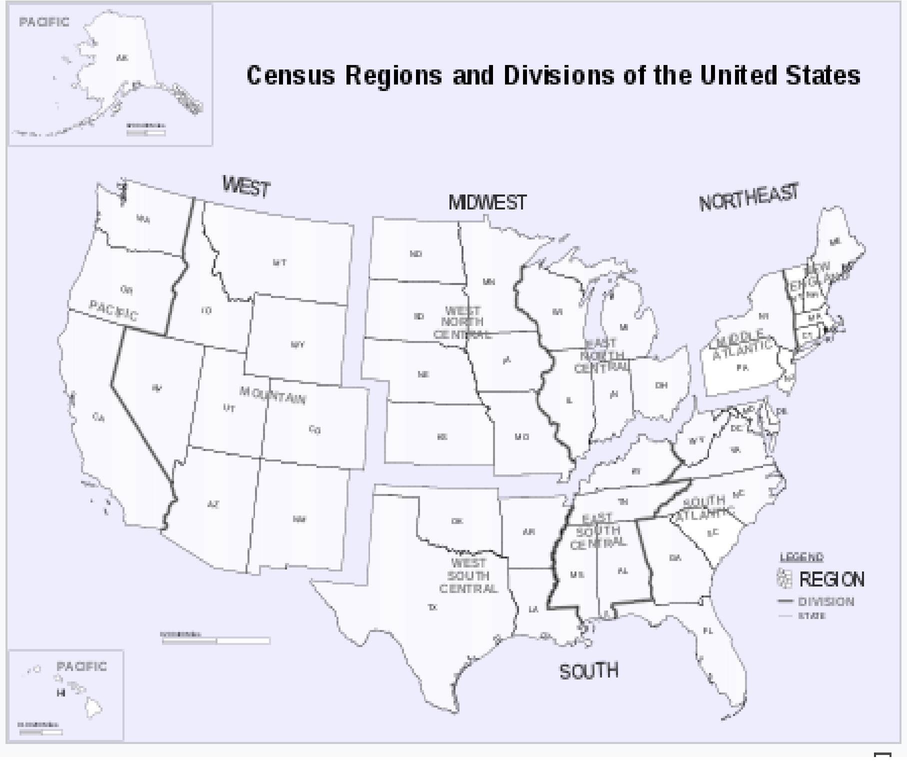


Figure 2.3 A map that of the United States from the Census Bureau showing the regions that were used for recoding of the state variable.

The second feature indicated which third of the term of the loan the borrower defaulted on payments, if they defaulted. This means that the borrower either paid the loan in full, defaulted in the first year of first third of the loan term, second year or second third of the loan term, or the third year or final third of the loan term. We then created this label by taking the date that the loan was issued and the date of last payment and calculating the amount of time between the two dates to find the total time that payments were being made on the loan. Because we would expect all loans to be paid off or defaulted approximately one or two months after the loan term, we dropped all records that far exceeded that time frame. We do not have all the payment history details on each loan; therefore, we do not know enough to proceed with these records. We can see on the Lending Club website that loans that fall delinquent are often brought back to âĂIJcurrentâĂİ by arrangement of a new payment plan. This could account for the length of some loan terms being longer than expected [25]. We then took both the binary label, we created, and the total amount of time that was paid on the loan to decide in which of four bins each record should be placed, one of the bins being paid-in-full, and the other three splitting the loan term in thirds. This is illustrated in figure 2.4.

The third label that was created was based on the date within the loan term when the original

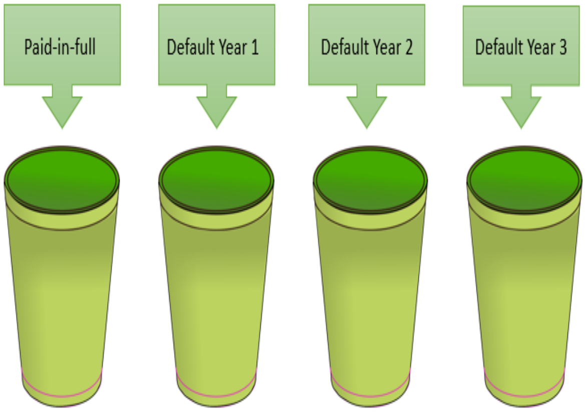


Figure 2.4 This figure illiterates what is contained in each bin we create for the multi-class classification tree and random forest model.

amount of the loan, excluding interest, was paid-in-full. Many records within the data set did have borrowers that defaulted, but did so after their payments had exceeded the amount of the original loan. In these instances, the lender still made a profit, or at least received the principal amount from the borrower, even though it was not the principal amount plus interest. To calculate if the loan was profitable or not, we multiplied the number of months the borrower made payments on the loan by installment paid each month. Then we subtracted that number from the loan amount. If the number was negative or 0, the loan was labeled 1 indicating the loan was profitable, else it was labeled 0 indicating the lender lost their investment. Default versus paid-in-full was not considered when creating this label.

The fourth label was created by considering the percent a lender will get as a return on their investment. We want to be able to predict, with some certainty, what percent of the principal amount the investor will gain or lose. We calculated this by summing the total received principal and the total received interest, then subtracting the loan amount. This gave us the amount of money that the investor received from the borrower over the amount that was lent. This total was then divided by the loan amount to find the percent that the investor either gained or lost on the investment.

2.3 Sampling

Another challenge to over come in our data was a severe class imbalance. Decision tree models create biased trees if there is a dominate class. This will frequently result in always predicting the majority class. A class imbalance was expected as Lending Club does not want to take on loans that are going to default, but conceivably some do. In three of our four created labels, we could see imbalance. When there is a severe class imbalance, re-sampling is typically performed to relieve classifier bias when training because of the simplicity of the technique as well as implementation. We chose to examine three different methods- Over-Sampling, Under-Sampling, and Synthetic Minority Over-Sampling (SMOTE). Across all models, we can see in table 2.5 that SMOTE and Over-Sampling out perform Under-Sampling. The severity of the class imbalance forces us, when under sampling, to have so few records that there are not enough to properly train the model.

Table 2.4 Class Imbalance Percentages per Model

Multi-Class Label

	Percent of Data in Each Class
Paid-in-full	87.37%
Default Year 1	4.59%
Default Year 2	5.14%
Default Year 3	2.90%

Binary Label

	Percent of Data in Each Class
Paid-in-full	87.37%
Default	12.63 %

Break-Even Label

	Percent of Data in Each Class
Gain from Investment	88.72%
Loss from Investment	11.28%

2.3.1 Over-Sampling the Minority Class

Over-sampling is done by generating new records that are copies of existing records. This process must be done after removing the validation set to prevent over fitting. We do not want the same records to be trained on as we will use to validate our models. Our over-sampling was done by first randomizing the data. We removed testing and validation sets. We then duplicated the minority class(es) until the percentages in the data were approximately equal to the majority class. We did investigate making the classes exactly equal by randomly selecting records to duplicate instead of duplicating the entire data set again, but this did not affect the accuracy of the model.

2.3.2 Under-Sampling the Majority Class

Under-sampling is done by decreasing the number of records in the majority class until it is comparable in size to the minority class. This is done by randomly selecting records to keep in the class. Our under-sampling was done by first randomizing the data a second time. We sampled using a simple random sampling method. In simple random sampling, each unit has an equal probability of selection, and sampling is without replacement. We drew only the number of records that were in the majority class. This severely depleted the number of records that were available to our model. Due to the lack of information given to the model, this method did not create favorable results. If more data were available, this method could be a better option.

2.3.3 SMOTE

SMOTE is a common method of over-sampling. Instead of duplicating records, the algorithm creates records that are synthetic to append to the minority class. Synthetic records are created by looking along the line segments, in the feature space, between minority class instances. The algorithm draws a line segment between each instance and its k nearest neighbors, then randomly chooses one of these line segments to create a synthetic instance. After determining the directions to use, the difference is taken between the instance and its nearest chosen neighbor. That difference is multiplied by a random number between 0 and 1 and added to the instance. This causes the selection of a random point along the line segment between two specific instances. This idea is depicted in figures 2.5, 2.6, and 2.7. The algorithm will continue this process until the classes are balanced. This approach effectively forces the decision region of the minority class to become more general [8]. Our implementation uses five nearest neighbors.

2.3.4 Sampling Results

Imbalance within classes was by far the biggest hurdle in processing the data. The class imbalance was so severe combined with our high dimensional data set, the SMOTE algorithm was unable to really compete with over-sampling. In a paper by Blagus and Lusa [4], it was concluded that even though SMOTE performs well on low-dimensional data, it is not effective in the high-dimensional setting for many classification methods, including classification trees. Under-sampling sharply depleted the amount of data fed into the classifier. The lack of data caused unfavorable results coming out of the model. It can be seen in table 2.5, over sampling is the superior method. It is also much less complicated and takes far less time to implement.

We are using error rate of our decision tree model to make decisions on methods to use to finalize our model. Error rate is calculated by determining the number of correctly classified examples divided by the total number of examples and subtracting that decimal from one. We want this rate

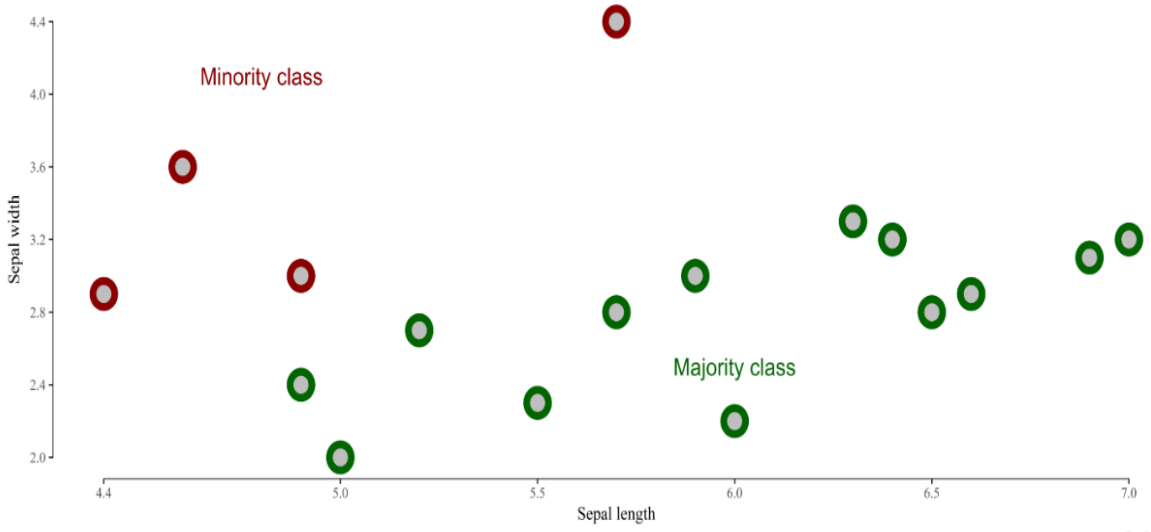


Figure 2.5 The red data points are the minority class in this example, and the green data points are the majority class [20]. This example was drawn from en example using the well known Iris data set.

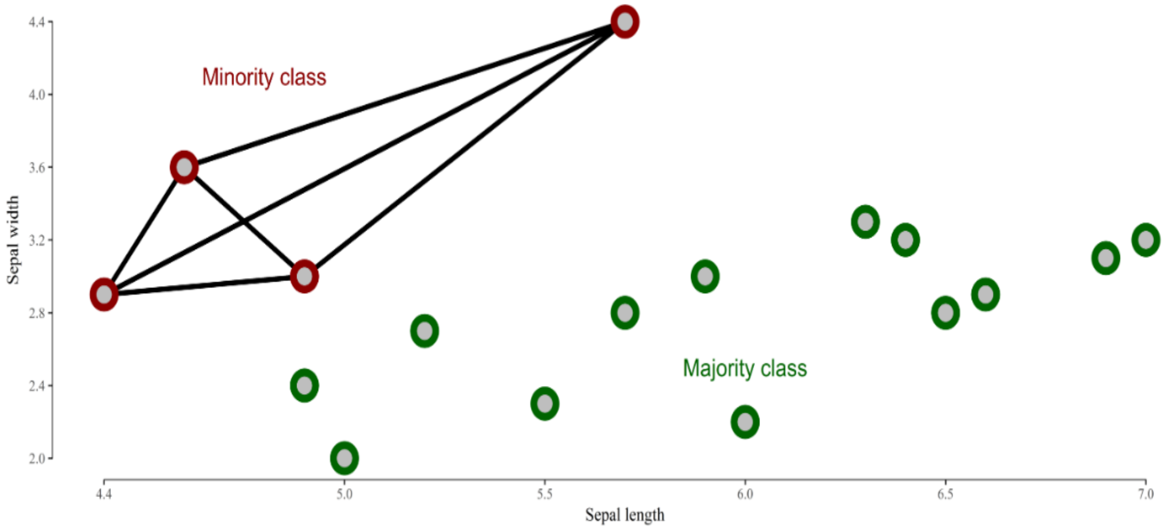


Figure 2.6 This figure shows line segments that are drawn in the feature space between minority class instances [20].

to become as small as possible while taking into account the false positive rate as well, which is explains in detail in chapter 4.

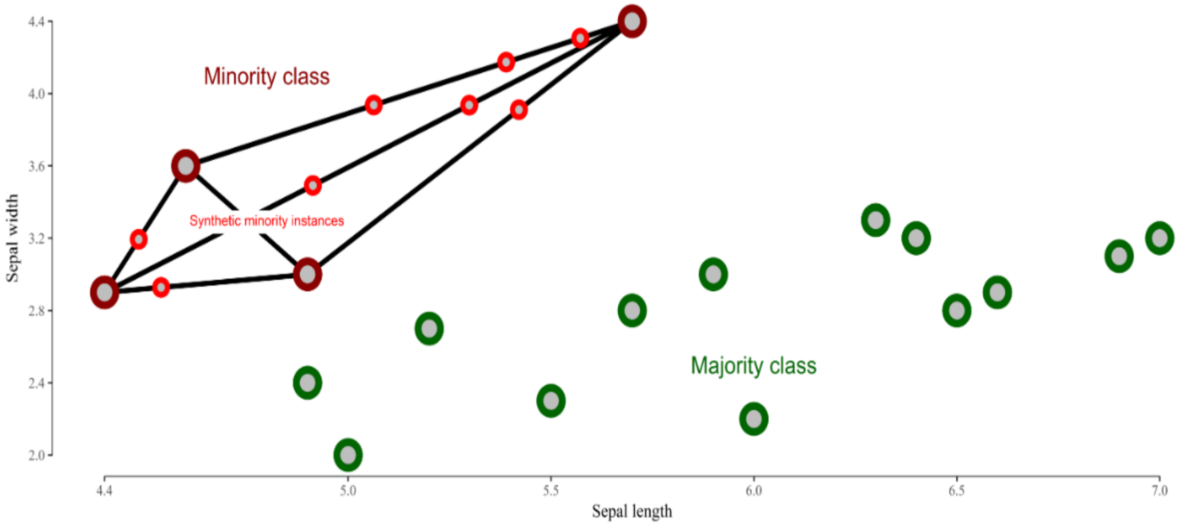


Figure 2.7 This figure shows the synthetic observations the algorithm created. The algorithm randomly chooses one of these line segments to create a synthetic instance. After determining the directions to use, the difference is taken between the instance and its nearest chosen neighbor. That difference is multiplied by a random number between 0 and 1 and added to the instance. This causes the selection of a random point along the line segment between two specific instances [20].

Table 2.5 Error Rates for Sampling Methods

	Multi-Class Label	Binary Label	Break-Even Label
SMOTE	37.23%	34.58%	33.83%
Over Sampling	23.99%	22.15%	20.49%
Under Sampling	47.48%	45.32%	44.13%

2.4 Imputation

A serious problem when working with real-world data is that there are often significant amount of data missing for various reasons. When working with a limited amount of data, it is crucial to use all available data and not discard records due to missing values if possible.

Lending Club compiles a large amount of financial and personal data from its borrowers. This data is plagued with missing entries, some explainable and some that are seemingly erroneous. In section 2.1, we discussed our method of removing features due to having more than 50% missing from the feature. In the cases where the missing information was seemingly random we elected to attempt two forms of imputation to gain precision. We employed two popular methods- Nearest Neighbor Hot-deck and Expectation Maximization. We evaluated the performance of each based on improvement in accuracy within the models based on the accuracy of simply removing all missing data. While the hot-deck method imputes missing values by clustering the data and then randomly choosing a âĂIJdonorâĂİ row to use for the missing value, the expectation maximization imputes missing values by finding the maximum likelihood estimates and iterates until maximizing the log likelihood. Then it uses this estimate to impute the data.

2.4.1 Removing Missing Data

Removing missing data involved deleting records in our data set that are missing data in any feature of interest. This is a common technique because it is easy to implement and works with any type of analysis. Because we needed a baseline for our imputation attempts, we started this process by deleting all records with missing data anywhere in them. We deleted 9,036 records.

Table 2.6 Patterns within Missing Data

Group	1	2	3	4	5	6	7	8	9	10	11	12
tax_liens	X	X	X	X	X	X	X	X				
total_acc	X	X	X	X	X	X	X	X	X	X		
collections_12_mth_ex_med	X	X	X	X	X							
acc_now_delinq	X	X	X	X	X	X	X	X	X	X		
chargeoff_within_12_mths	X	X	X	X	X							
delinq_amnt	X	X	X	X	X	X	X	X	X	X		
earliest_cr_line	X	X	X	X	X	X	X	X	X	X		
revol_util	X	X	X			X	X		X			
pub_rec_br	X	X		X	X	X						
pub_rec	X	X	X	X	X	X	X	X	X	X		
open_acc	X	X	X	X	X	X	X	X	X	X		
delinq_2yrs	X	X	X	X	X	X	X	X	X	X		
emp_length	X		X	X		X	X	X	X	X	X	X
annual_inc	X	X	X	X	X	X	X	X	X	X	X	
inq_last_6mths	X	X	X	X	X	X	X	X	X	X		
Frequency	166093	7542	1203	143	6	4	32	2	74	1	25	4

As you can see in Table 2.6, groups 11 and 12, most features are missing values in these 29 records (25 from group 11 + 4 from group 12). Although it is impossible, by looking at the data, to tell the reason these 29 records are missing values from nearly all 15 features, it might imply there is an underlying pattern of missingness among these records. Therefore, we dropped these 29 records. This also eliminates the need to impute total_acc, acc_now_delinq, delinq_amnt, earliest_cr_line, pub_rec, open_acc, delinq_2yrs, annual_inc, inq_last_6mths.

2.4.2 Expectation Maximization

The Expectation Maximization (EM) Algorithm is a technique that is used to impute missing values by finding the maximum likelihood estimates. This is done by using an iterative process with two main steps- the expectation step, or E step and the maximization step, or the M step. These two steps are repeated until convergence. Though the EM algorithm was used earlier for special circumstances, in a paper from 1977, Dempster, Laird, and Rubin [11] generalize and formally formulate the Expectation Maximization algorithm [1]. To use the EM algorithm in our incomplete data problem, we must associate it with the complete data problem to estimate the maximum likelihood. The initial step in the algorithm is to initialize the parameter vector, let this be Θ.

Let y denote the complete data samples, with $y \in Y \subseteq R^m$, and the probability density function (pdf) that is associated with y be $P_y(y;\Theta)$. y, however, cannot be directly observed. We will refer to our observed data as $x = g(y) \in X_{ob} \subseteq R^l$, where $l < m$. The pdf associated with the observed data is $P_x(x;\Theta)$. The subset of y's associated with a single x can be written as $Y(x) \subseteq Y$. The pdf of the incomplete data can be found by

$$P_x(x;\Theta) = \int_{Y(x)} P_y(y;\Theta) dy. \tag{2.1}$$

Ideally, the Maximum Likelihood estimate of Θ, denoted $\hat{\Theta}_{ML}$, would be given by

$$\Sigma_k \frac{\partial ln(P_y(y_k;\Theta))}{\partial \Theta} = 0. \tag{2.2}$$

Our problem is that the y's represent our complete data, which is not available. Here is where we use the EM algorithm to combat this issue.

For the current value of $\Theta^{(t)}$, the parameter vector, the E step computes the expected value of the observed data log-likelihood

$$Q(\Theta|\Theta^{(t)}) = E[\Sigma_k ln(P_y(y_k;\Theta)|X,\Theta^{(t)})]. \tag{2.3}$$

The M step finds the parameter vector that maximizes the log-likelihood of the imputed data. We call this vector $\Theta^{(t+1)}$ and our goal is to maximize $Q(\Theta;\Theta^{(t)})$. That is

$$\frac{\partial Q(\Theta;\Theta^{(t)})}{\partial \Theta} = 0. \tag{2.4}$$

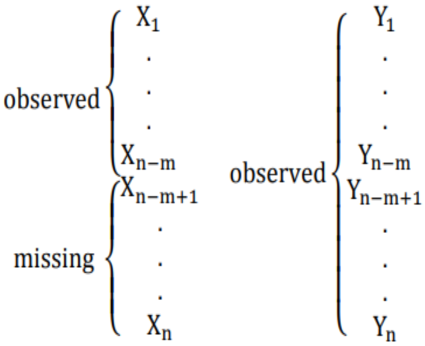
The initial estimate is begun at $\Theta^{(0)}$ and iterations are continued until the following condition is met:

$$||\Theta^{(t+1)} - \Theta^{(t)}|| \leq \epsilon. \tag{2.5}$$

Because the log-likelihood increases at each step and is bounded from above, convergence is guaranteed. Proof of this can be found in [11] on page 7.

2.4.3 Hot-Deck

Hot-deck Imputation is an intuitive method for handling incomplete data. Therefore it has been used propitiously on large data sets [27] for many years. We employed nearest neighbor hot deck imputation. In this method, missing values are replaced by values from similar complete samples. These similarities are found by clustering the data. We must define a metric to measure distance between complete data and data with missing information. Suppose our X indicates features with missing data with m missing values and Y indicates complete features.



To choose which value of X_i where $i = 1,...,n-m$ replaces the missing in X_j where $j = n-m+1,...n$, we must look at the corresponding values in the complete data Y. By calculating the closest value Y_j to Y_i, we determined which samples were most similar. The distance is calculated using the following method:

$$|Y_i - Y_j| = \min_{1\leq k\leq n-m} |Y_k - Y_j| \tag{2.6}$$

If two or more values are equidistant from Y_j, the mean of the corresponding X_i is calculated for imputation.

We looked to see what features had missing data. This can be seen in Table 2.6. We then analyzed which features were most correlated with the feature containing missing data. The feature most correlated will be Y as in our equations above. You can see in Table 2.7 the features and their correlation rates with the features being imputed.

Table 2.7 Features with Missing Data and Features Most Correlated

Feature with Missing Data	Feature Most Correlated	Correlation Rate
emp_length_num	mortgage	0.214
tax_liens_num	pub_rec_num	0.643
collections_12_mths_ex_med_num	int_rate_num	0.043
chargeoff_within_12_mths_num	delinq_2yrs_num	0.114
pub_rec_br_num	pub_rec_num	0.753
revol_util_num	int_rate_num	0.412

2.4.4 Imputation Results

As we see in Table 2.8, the EM algorithm sightly outperforms both Hot-deck Imputation and simply removing the missing data in every model. In our situation, Expectation Maximization is a good fit for the type of data we have. Features we imputed had at most 4% of the data missing, and it is important to us that they preserve the relationship with other features for its use in the decision tree model.

Table 2.8 Error Rates for Imputation Methods

	Multi-Class Label	Binary Label	Break-Even Label
Remove Missing Data	23.99%	22.39%	21.26%
EM	23.99%	22.15%	20.49%
Hotdeck Imputation	24.27%	22.15%	20.84%

2.5 Contributions

While creating the labels, sampling, and imputing missing data are not novel, the way in which we employ these methods gives us the ability to run the model to produce results that are novel. In literature, we find countless papers and projects that do a binary decision tree on loan data to determine if a loan will default or be paid-in-full. The uniqueness in this dissertation comes in the multi-class labels that allow us to predict the time to default and percent return on investment using a decision and regression tree, respectfully.

Table 2.9 Features Dropped Before Analysis

Feature Dropped	Reason
all_util	All Missing
acc_open_past_24mths	Not collected Prior to 2012
annual_inc_joint	All Missing
application_type	All entires are the same
avg_cur_bal	31% Missing
bc_open_to_buy	Not collected Prior to 2012
bc_util	Not collected Prior to 2012
collection_recovery_fee	Data recorded into the life of the loan
deferral_term	All Missing
dti_joint	All Missing
emp_title	Can't be used in analysis
hardship_amount	All Missing
hardship_dpd	All Missing
hardship_end_date	All Missing
hardship_last_payment_amount	All Missing
hardship_length	All Missing
hardship_loan_status	All Missing
hardship_payoff_balance_amount	All Missing
hardship_reason	All Missing
hardship_start_date	All Missing
hardship_status	All Missing
hardship_type	All Missing
hardship_flag	All enties are the same
id	All Missing
il_util	All Missing
inq_fi	All Missing
inq_last_12m	All Missing
last_credit_pull_d	Data recorded into the life of the loan
last_pymnt_amnt	Data recorded into the life of the loan
max_bal_bc	All Missing
member_id	All Missing
mo_sin_old_il_acct	31% Missing
mo_sin_old_rev_tl_op	31% Missing
mo_sin_rcnt_rev_tl_op	31% Missing

Table 2.9 (continued)

Feature Dropped	Reason
mo_sin_rcnt_tl	31% Missing
mort_acc	Not collected Prior to 2012
mths_since_last_delinq	58% missing
mths_since_last_major_derog	86% missing
mths_since_last_record	90% missing
mths_since_rcnt_il	All Missing
mths_since_recent_bc_dlq	84% Missing
mths_since_recent_inq	31% missing
mths_since_recent_revol_delinq	76% missing
mths_since_recent_bc	Not collected Prior to 2012
next_pymnt_d	Data recorded into the life of the loan
num_accts_ever_120_pd	31% missing
num_actv_bc_tl	31% missing
num_actv_rev_tl	31% missing
num_bc_tl	31% missing
num_il_tl	31% missing
num_op_rev_tl	31% missing
num_rev_accts	31% missing
num_rev_tl_bal_gt_0	31% missing
num_tl_120dpd_2m	31% missing
num_tl_30dpd	31% missing
num_tl_90g_dpd_24m	31% missing
num_tl_op_past_12m	31% missing
num_bc_sats	Not collected Prior to 2012
num_sats	Not collected Prior to 2012
open_acc_6m	All missing
open_il_12m	All Missing
open_il_24m	All Missing
open_il_6m	All Missing
open_rv_12m	All Missing
open_rv_24m	All Missing
orig_projected_additional_accrue	All Missing
out_prncp	Data recorded into the life of the loan
out_prncp_inv	Data recorded into the life of the loan

Table 2.9 (continued)

Feature Dropped	Reason
payment_plan_start_date	Data recorded into the life of the loan
pct_tl_nvr_dlq	31% missing
percent_bc_gt_75	Not collected Prior to 2012
policy_code	All entires are the same
pymnt_plan	Data collected after origination
recoveries	Data recorded into the life of the loan
revol_bal_joint	All Missing
sec_app_chargeoff_within_12_mths	All Missing
sec_app_collections_12_mths_ex_m	All Missing
sec_app_earliest_cr_line	All Missing
sec_app_inq_last_6mths	All Missing
sec_app_mort_acc	All Missing
sec_app_mths_since_last_major_de	All Missing
sec_app_num_rev_accts	All Missing
sec_app_open_acc	All Missing
sec_app_open_il_6m	All Missing
sec_app_revol_util	All Missing
tot_coll_amt	31% missing
tot_cur_bal	31% missing
tot_hi_cred_lim	31% missing
total_bal_il	All Missing
total_cu_tl	All Missing
total_il_high_credit_limit	31% missing
total_pymnt	Data recorded into the life of the loan
total_pymnt_inv	Data recorded into the life of the loan
total_rec_late_fee	Data recorded into the life of the loan
total_rev_hi_lim	31% Missing
total_bal_ex_mort	Not collected Prior to 2012
total_bc_limit	Not collected Prior to 2012
url	All Missing
verification_status_joint	All Missing

CHAPTER

3

FEATURE ENGINEERING

Feature engineering is a process of the analyst creating features by either using relationships among features currently in the data or looking for outside information that can be merged onto the data with already existing features. Feature engineering is the human element in machine learning. Understanding of the data, domain knowledge, and human intuition and creativity are what allow feature engineering to make a difference in model accuracy. Decision Trees are designed to find relationships among features and uses those to achieve better results. In general, a well engineered feature may be easier for the algorithm to digest and make rules from than the feature from which it was derived.

In our data, we created one relationship that increases accuracy. First credit pull (first_cr_pull) is time between earliest credit line and loan origination. This gives the model the amount of time that a borrower has had at least one credit line open. You can see in table 3.2, first credit pull is of significant importance to all three models that were run.

In our data set, we were provided with the first three digits of a zip code for each borrower. This allowed us to know what state and what region of the state each borrower was associated with at the time the loan was originated.

We also wanted to get a picture of how far the borrowers income was away from the average income in their general area. The only two location variables we had for each borrower was the first 3 numbers in the zip code and the state in which they reside. We used a zip code database that included county, zip code, and population from the year 2015. We also needed a data set that contained income information per capita. We used a data set that provided average income per county, but did not contain zip code. Both data sets were pulled from the census bureau database. We merged these two data sets by county, and calculated the average income in each 3 digit zip code region.

To create the relationship for the new feature, percent difference from the average income, we looked at each borrowers income and the average income in their region. We then calculated the percent difference and the percent increase.

$$per\_dif = \frac{|average\_income - annual\_income|}{\frac{1}{2} \times (average\_income + annual\_income)} \times 100 \tag{3.1}$$

$$per\_increase = \frac{annual\_income - average\_income}{annual\_income} \times 100 \tag{3.2}$$

As a result of adding these features, the error rate decreased, as seen in table 3.1. Also, in table 3.2, percent difference in income (per_dif) was 10th most important in all models run. This tells us that this feature is useful to the models, and made a difference in its calculations.

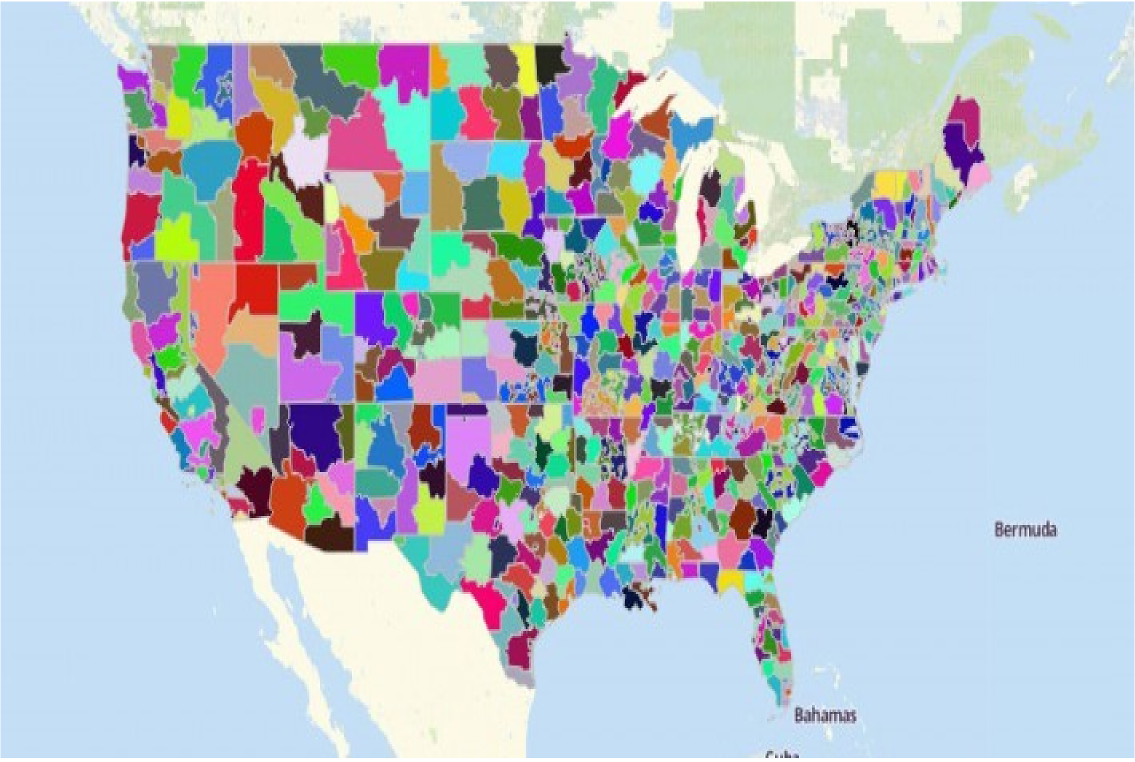


Figure 3.1 Map of United States Showing 3-digit Zip Code Regions

3.1 Feature Engineering Results

Table 3.1 Error Rates for Features Engineered

	Multi-Class Label	Binary Label	Break-Even Label
Before Feature Engineering	23.99%	22.15%	20.49%
First Credit Pull	23.54%	21.85%	20.20%
Percent difference in Income	23.48%	21.71%	20.07%
Percent Increase in Income	23.57%	21.83%	20.13%
With all Engineered Features	23.46%	21.63%	19.67%

3.2 Feature Importance

Feature importance is calculated within a decision tree model while it is splitting, which will be discussed more in chapter 4. The importance score is calculated as the decrease in node impurity weighted by the probability of reaching that node. The probability of reaching a node is calculated by the number of samples that reach the node, divided by the total number of samples. The higher the importance score the more important the feature is to the model.

In Table 3.2, we can see the top ten most important features in all decision tree models. We notice that all three models approximately score the same 10 features most important. We are asking each model run to make similar decisions all based on giving a loan to someone based on their financial characteristics, thus using the same features for each model is not a surprising result. Similarly, professionals, such as loan officers or investors, use certain criteria for loans, to decide if they are good investments to fund.

3.3 Contributions

Feature engineering is not a unique method in machine learning; however, its creative aspect does lend itself to finding new ways to allow the model to incorporate data. In this way, we calculated three features that are definitely used in the field, possibly inadvertently, to make decisions about loans, but are not typically used as features. This data is not directly collected by Lending Tree, to our knowledge, but can be sought out by investors prior to investing.

Table 3.2 Table of 10 Most Important Features

Importance Score	Feature
Multi-Class Label	
0.078	dti_num
0.078	revol_bal_num
0.075	revol_util_num
0.065	loan_amnt_num
0.064	annual_inc_num
0.053	earliest_cr_line_num
0.053	total_acc_num
0.053	first_cr_pull
0.044	int_rate_num
0.044	per_dif
Binary Label	
0.076	dti_num
0.075	revol_util_num
0.071	revol_bal_num
0.065	annual_inc_num
0.060	New_England
0.056	total_acc_num
0.056	loan_amnt_num
0.049	earliest_cr_line_num
0.049	first_cr_pull
0.046	per_dif
Break-Even Label	
0.075	dti_num
0.074	revol_bal_num
0.073	annual_inc_num
0.069	revol_util_num
0.059	New_England
0.059	loan_amnt_num
0.054	first_cr_pull
0.050	total_acc_num
0.050	earliest_cr_line_num
0.047	per_dif

CHAPTER

4

METHODS

4.1 Classification Tree

The idea of classification and regression trees was introduced by Breiman et al. in 1984 [5]. The basic goal of a classification tree is to split a population of data into smaller segments. This algorithm has a tree-like structure, hence the name. Each internal node represents features, each branch represents a rule made by the algorithm, and each leaf the outcome of those decisions. In figure 4.1 we see a basic example of the decision tree structure. A tree uses if-then statements to find patterns in data. Each of these if-then statements creates the branches. Elements to the left of that point get categorized in one way, while those to the right are categorized in another. Each non-terminal node, determined by several parameters, repeats the splitting process until a specified level of purity is reached. Decision trees are grown by examining all possible splits of all input features. The algorithm then decides if the criterion has been met or not [35]. In figure 4.2, we see a basic flow chart for how a decision tree is created as well as how the data flows though the process of creation. Each piece of this flow chart is discussed, in detail, throughout this chapter.

There are two stages to prediction. The first is training the model where the tree is built and optimized by using the training and validation sets. The second is predicting where we use the optimal parameters found for the model to predict an outcome for the testing set.

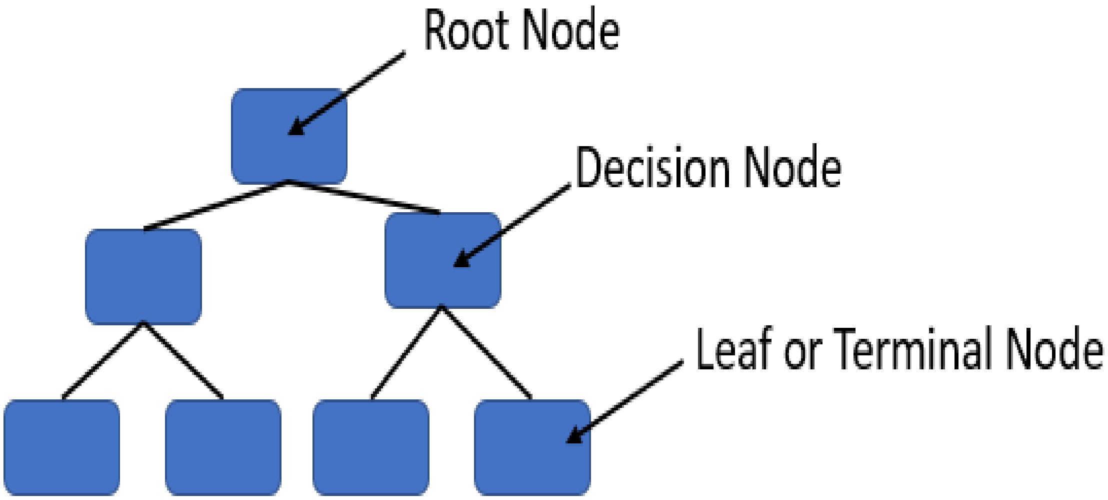


Figure 4.1 Basic decision tree structure that defines the root, decision, and leaf nodes.

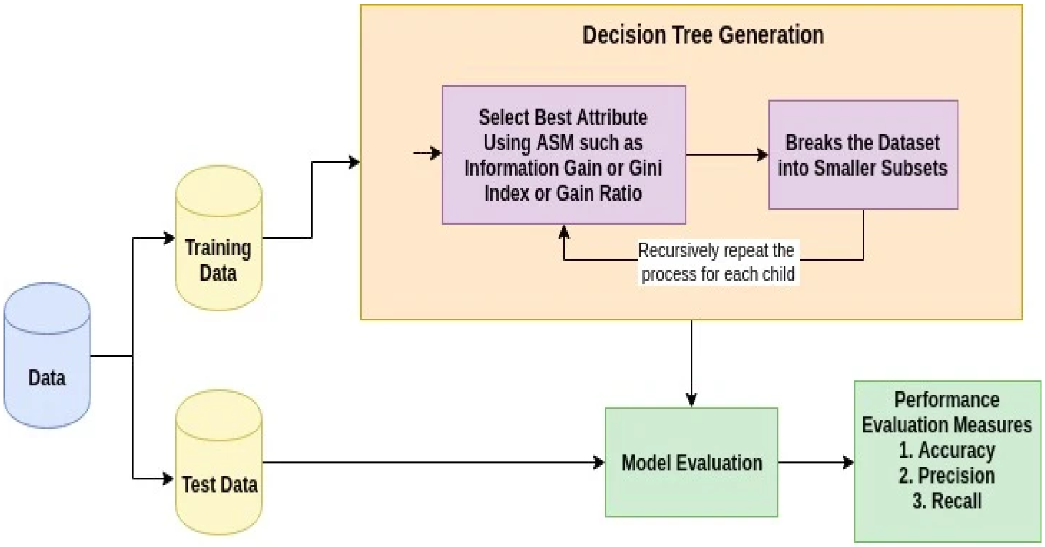

Decision Tree Generation
Select Best Attribute Using ASM such as Information Gain or Gini Index or Gain Ratio
Breaks the Dataset into Smaller Subsets
Recursively repeat the process for each child
Training Data
Data
Test Data
Model Evaluation
Performance Evaluation Measures
1. Accuracy
2. Precision
3. Recall

Figure 4.2 Basic Classification Tree Flow Chart

4.1.1 Classification Tree Model

Given the training matrix $X \in R^{l \times n}, i = 1,, l$, where n is the number of features and l is the number of observations, and a label vector $Y \in R^{l \times 1}$, a decision tree recursively partitions the space such that the samples with the same labels are grouped together.

Let the data at node m be represented by Q. For each candidate split $\Theta = (j, t_m)$ consisting of feature $j \in 1,, n$ and threshold t_m, partition the data into $Q_{left(\Theta)}$ and $Q_{right(\Theta)}$ subsets where

$$Q_{left}(\Theta) = (x, y)|x_j \leq t_m \tag{4.1}$$

$$Q_{right}(\Theta) = Q \backslash Q_{left}(\Theta). \tag{4.2}$$

The impurity at the point m is determined using an impurity function $H(\cdot)$, for which there are several choices depending on the problem being solved. This is discussed later in the chapter. The total impurity, $G(\cdot,\cdot)$, (weighted by the number of samples in the nodes) is computed using the impurity function $H(\cdot)$ as follows,

$$G(Q,\Theta) = \frac{n_{left}}{N_m} H(Q_{left}(\Theta)) + \frac{n_{right}}{N_m} H(Q_{right}(\Theta)), \tag{4.3}$$

where n is the amount of data in the (right or left) node and N is the total data points in the parent node. Select the parameters that minimizes the impurity,

$$\Theta^* = argmin_\Theta G(Q,\Theta). \tag{4.4}$$

Continue for subsets $Q_{left}(\Theta^*)$ and $Q_{right}(\Theta^*)$ until the tree is either fully grown or optimized [32].

4.1.2 Impurity

The two most commonly used ways to evaluate splits are the Gini Index and Entropy measure. The two measures are very similar but have a few notable differences. Entropy (information gain) is the measure of disorder within the data.

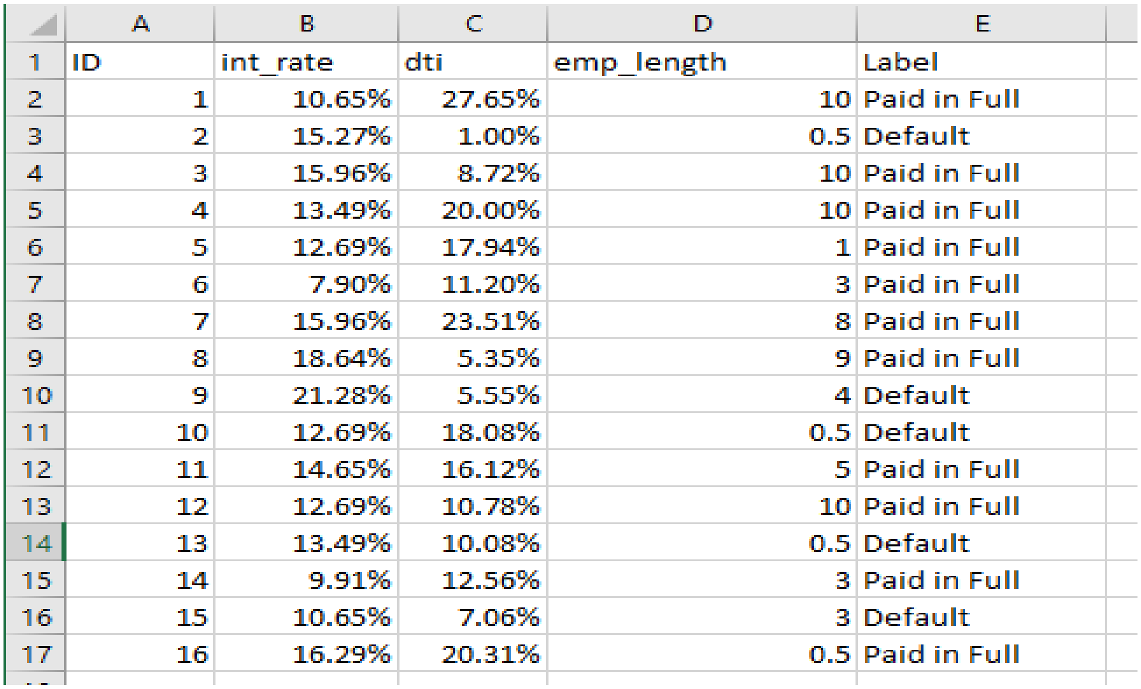
$$E = -\Sigma_{i=1}^{N} p_i log_2(p_i) \tag{4.5}$$

is the disorder of the set at each node, where N is the number of classes and p_i are the ratios of the elements of each label in the set. Minimum Entropy measure is 0 and maximum is $log_2 N$. The goal is to minimize E by splitting the observations with different cut points to determine best split. The Gini Index is the expected error rate according to the distribution of the classes at that node. The cost function

$$I = 1 - \Sigma_{i=1}^{N} p_i^2 \tag{4.6}$$

is used to evaluate splits within the data set by measuring node impurity, where N is number of classes and p_i is the fraction of each class i that reach the node. The goal is to minimize I by splitting the observations and features with different cut points to determine best split. Once a feature is chosen, use the best cut point, and repeat until end leaves are found. The Gini impurity index is the measure that was used in our research because it was faster, not necessarily more accurate. Calculating the log at each split is time consuming for large data sets. A weighted sum of the impurity can be computed for each partition. The partition that gives the minimum gini impurity index is selected.

The process of creating a decision tree begins by selecting the the feature and cut point within the feature that makes up the root node. The root node, or any decision node within the model, is chosen using an impurity measure, and in our case the Gini impurity index. In figure 4.3, we see three features and a label in which we will use to determine the best feature to place in the root node of this decision tree, which is a very small sample of our data. We will use this to simulate the real process.



	A	B	C	D	E
1	ID	int_rate	dti	emp_length	Label
2	1	10.65%	27.65%	10	Paid in Full
3	2	15.27%	1.00%	0.5	Default
4	3	15.96%	8.72%	10	Paid in Full
5	4	13.49%	20.00%	10	Paid in Full
6	5	12.69%	17.94%	1	Paid in Full
7	6	7.90%	11.20%	3	Paid in Full
8	7	15.96%	23.51%	8	Paid in Full
9	8	18.64%	5.35%	9	Paid in Full
10	9	21.28%	5.55%	4	Default
11	10	12.69%	18.08%	0.5	Default
12	11	14.65%	16.12%	5	Paid in Full
13	12	12.69%	10.78%	10	Paid in Full
14	13	13.49%	10.08%	0.5	Default
15	14	9.91%	12.56%	3	Paid in Full
16	15	10.65%	7.06%	3	Default
17	16	16.29%	20.31%	0.5	Paid in Full

Figure 4.3 This sample of data was used to choose the root note to demonstrate the uses of impurity within a decision tree.

In this example, we will use specific cut points for each feature instead of calculating the index for every cut point within each feature because of the extensive amount of calculations.

To calculate the total impurity for interest rate (int_rate), we used the same equations as above,

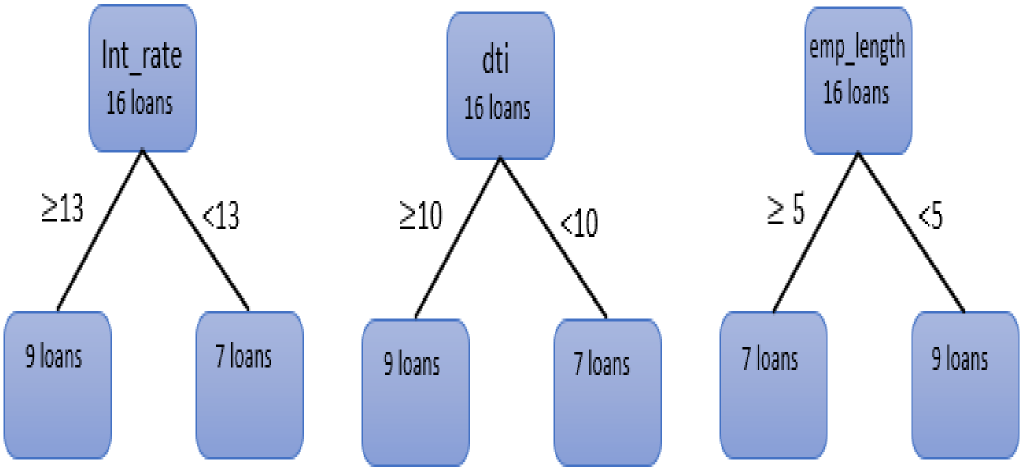


Figure 4.4 This figure shows the 3 features that are candidates for the root node along with the example cut points.

(4.6),

$$\begin{aligned} int\_rate_{\geq 13} &= 1-((\frac{6}{9})^2+(\frac{3}{9})^2) \\ &= 0.444 \end{aligned} \tag{4.7}$$

$$\begin{aligned} int\_rate_{<13} &= 1-((\frac{5}{7})^2+(\frac{2}{7})^2) \\ &= 0.408 \end{aligned} \tag{4.8}$$

$$\begin{aligned} int\_rate_{total} &= \frac{9}{16}\times 0.444+\frac{7}{16}\times 0.408 \\ &= 0.4283 \end{aligned} \tag{4.9}$$

To calculate the total impurity for debt to income ratio (dti), we used the same equations as above,

$$\begin{aligned} dti_{\geq 10} &= 1-((\frac{7}{9})^2+(\frac{2}{9})^2) \\ &= 0.3457 \end{aligned} \tag{4.10}$$

$$\begin{aligned} dti_{<10} &= 1-((\frac{4}{7})^2+(\frac{3}{7})^2) \\ &= 0.4898 \end{aligned} \tag{4.11}$$

$$\begin{aligned} dti_{total} &= \frac{9}{16}\times 0.3457+\frac{7}{16}\times 0.4898 \\ &= 0.4088 \end{aligned} \tag{4.12}$$

To calculate the total impurity for employment length (emp_length), we used the same equations as above,

$$\begin{aligned} emp\_length_{\geq 5} &= 1-((\frac{7}{7})^2+(\frac{0}{7})^2) \\ &= 0 \end{aligned} \tag{4.13}$$

$$\begin{aligned} emp\_length_{<5} &= 1-((\frac{4}{9})^2+(\frac{5}{9})^2) \\ &= 0.4938 \end{aligned} \tag{4.14}$$

$$\begin{aligned} emp\_length_{total} &= \frac{7}{16}\times 0+\frac{9}{16}\times 0.4938 \\ &= 0.2778 \end{aligned} \tag{4.15}$$

From looking at these three candidates for the root node feature the algorithm would choose employment length because it is the most pure split, or has the lowest total Gini index. This process is repeated at every node with the remaining data that made it to the node to determine the next best split.

In figure 4.5, we show two example splits, A and B. Here we are show that nodes with multiple classes are calculated exactly the same way as a binary tree. The Gini Index can be calculated in the

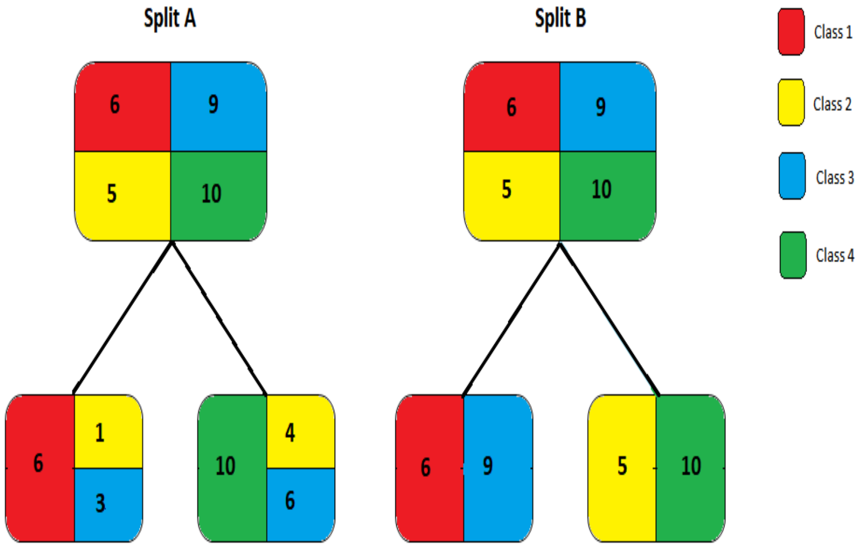


Figure 4.5 Two Possible Splits to be Made by a Classification Tree

following way-

$$\begin{aligned} SplitA_{left} &= 1-((\frac{6}{10})^2+(\frac{1}{10})^2+(\frac{3}{10})^2) \\ &= 0.54 \end{aligned} \tag{4.16}$$

$$\begin{aligned} SplitA_{right} &= 1-((\frac{10}{20})^2+(\frac{4}{20})^2+(\frac{6}{20})^2) \\ &= 0.62 \end{aligned} \tag{4.17}$$

The total impurity for Split A is weighted by the number of samples in each node.

$$\frac{10}{30}*0.54+\frac{20}{30}*0.62 = .5933 \tag{4.18}$$

$$\begin{aligned} SplitB_{left} &= 1-((\frac{6}{15})^2+(\frac{9}{15})^2) \\ &= 0.48 \end{aligned} \tag{4.19}$$

$$\begin{aligned} SplitB_{right} &= 1-((\frac{5}{15})^2+(\frac{10}{15})^2) \\ &= 0.44 \end{aligned} \tag{4.20}$$

The total impurity for Split B is weighted by the number of samples in each node.

$$\frac{15}{30}*0.48+\frac{15}{30}*0.44=0.46. \tag{4.21}$$

Since the total impurity is smaller (.46 compared to .5933) Split B is the better split, and will be chosen by the algorithm, because it has higher proportions of fewer classes in each leaf.

4.1.3 Over-fitting and Parameter Optimization

One of the biggest problems in machine learning involves overfitting, which is fitting predictors so well to the training set that the model cannot adapt to the testing set. To overcome this issue, pruning is necessary. ÂăPruning a classification tree often results in a smaller size tree and avoids unnecessarily complex trees. There are two methods for pruning- pre-pruning and post-pruning.

Pre-pruning, also called forward pruning, stops growing the tree earlier, before it perfectly classifies the training set. This prevents non-significant branches from being formed. This method involves using stopping criterion to stop growing prematurely as to not classify the training set perfectly. When constructing the tree, some significant measures can be used to assess the goodness of a split. If partitioning the tuples at a node would result the split that falls below a prespecified threshold, then further partitioning of the given subset is halted; otherwise, it is expanded. High threshold results in oversimplified trees, whereas low threshold results in very little simplification [31]. This threshold, ideally, should be determined by tuning the parameters within the model to find what threshold is best for the data.

Post-pruning, also known as backwards pruning, allows the tree toÂăperfectly classify the training set and then eliminates non-significant branches. There are various methods of post-pruning. One of the most common approaches is called Reduced Error Pruning. This method looks at each node, from the leaves up, in the classification tree and calculates the error rate of the new tree if that node was a leaf. If the new error rate is equal to or smaller than the original tree then the node is turned into a leaf. The branches and leaves below this node are discarded [31].

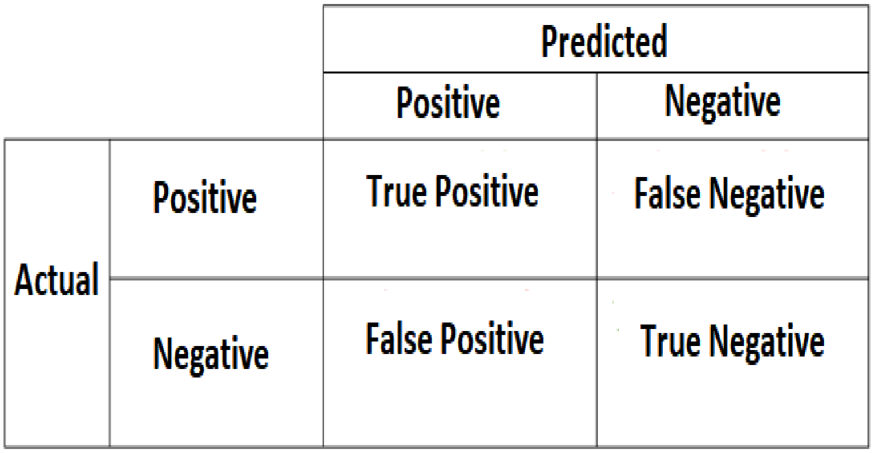
We used a pre-pruning technique. Data were separated into two sets: a training set, which is used to build the decision tree, and a validation set, which is used to evaluate the impact of pruning the tree.ÂăWe tuned parameters on the validation set using stopping criterion before using our tree to predict our testing set. Our stopping criterion were,

- Minimum Impurity Decrease - induces a split at a node if a decrease of the impurity greater than or equal to some set threshold value. Therefore we are not making splits that are not significant.
- Minimum Samples - sets a threshold for the minimum number of samples required to be a leaf node.

The specific thresholds used varied by model.

4.1.4 Model Evaluation

We are evaluating our classification trees using the error rate, which is the number of correctly classified examples divided by the total number of examples subtracted from one.



		Predicted	
		Positive	Negative
Actual	Positive	True Positive	False Negative
	Negative	False Positive	True Negative

Figure 4.6 Confusion Matrix

We also want to look at the confusion matrix, which will also help us to interpret the performance of our model. When the model is binary, a loan that is predicted as default (or a loss of investment) when it actually is paid in full (or gain from investment) is called a false negative and is displayed above the diagonal. When the model is binary, a loan that is predicted as paid in full (or a gain from investment), but actually defaults (or is a loss of investment) is called a false positive and is displayed below the diagonal. When our model is multi-class, we still want to look at the confusion matrix for more explanation, but these terms no longer apply. In our specific case, as we will see in Chapter 5, the column of the confusion matrix that was predicted paid in full, but actually defaulted is a good indicator of if our model will be helpful. These loans that end up in these cells are loans that we predicted would be paid in full, but have actually defaulted.

We also want to look at our binary loans using the following two rates:

- Sensitivity (True Positive Rate) - The fraction of loans that are predicted as paid-in-full loans and were actually paid-in-full (True Positive) over loans that were actually paid-in-full (True Positive + False Negative). This rate is sometimes called Recall or probability of detection and it measures how often a test correctly generates a positive (paid-in-full) result for loans that will be paid-in-full.

- Specificity (True Negative Rate) - The fraction of loans that are predicted as default and were actually default (False Negative) over loans that were actually default (False Positive + True Negative). This measures a model's ability to correctly generate a negative result (default) for loans that will default.

- Precision - The fraction of loans that are predicted as paid-in-full and were paid-in-full (True Positive) over the total number of loans that were predicted as paid-in-full (True Positive + False Positive).

These rates give us an overall picture of how our model is performing. Telling investors that loans will be paid in full, when they will actually default will cost investors. However, telling investors that loans will default when they will actually be paid in full will cost borrowers. These rates give a better picture of incorrectly classified loans than accuracy alone.

4.1.5 Interpretation

In figure 4.7, we can see a basic example of a decision tree built, using loan data, to predict year of default. A decision tree is said to be a "white box" of machine Learning algorithms because of the simplicity of extracting rules and interpreting results. By visualizing the tree, decision-making logic is available and simple to follow because it mimics the human-like thinking. The value and likelihood of outcomes can be quantified directly on the flow chart. When dealing with the domain of finance, it is imperative that the model and outcomes be easily interpretable and thus transparent to both borrower and investor.

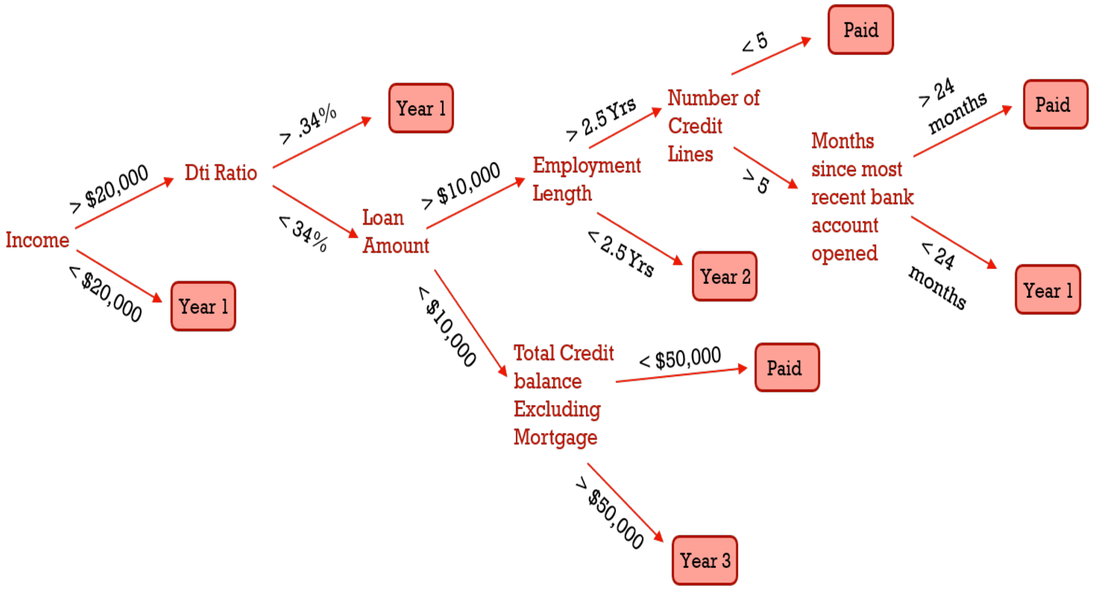


Figure 4.7 Basic Classification Tree Example for Year of Default Selection

4.2 Regression Tree

We typically use regression trees for problems where we attempt to predict the values of a continuous variable from one or more continuous and/or categorical predictor variables. We build these with very similar methods as the classification tree. Previously we were concerned with the misclassification rate when discussing classification trees. When talking about regression, we need a similar measure for node assignment. We used the Mean Squared Error in our model to determine these assignments. We will begin this explanation with the estimate for the mean squared error, $R(d)$, defined by

$$R(d) = \frac{1}{N}\Sigma_n (y_n - d(x_n))^2. \tag{4.22}$$

We want to find the value of $d(x_n) = y(t)$, which is the predicted value for each observation, to minimize $R(d)$, where $y(t)$ is the predicted response value at the node t.

Theorem 4.2.1. *The value of $y(t)$ that minimizes $R(d)$ is the average of y_n for all cases (x_n, y_n) falling into t; that is, the minimizing $y(t)$ is*

$$\bar{y}(t) = \frac{1}{N(t)}\Sigma_{x_n \in t}\, y_n, \tag{4.23}$$

where the sum is over all y_n such that $x_n \in t$ and $N(t)$ is the total number of cases in t.

The proof of this theorem can be found in [5], but this proof hinges on seeing that the number a which minimizes $\Sigma_n(y_n - a)^2$ is $a = \frac{1}{N}\Sigma_n y_n$. This equation can be written in the more familiar form

$$R(t) = \frac{1}{N}\Sigma_{x_n \in t}(y_n - \bar{y}(t))^2 \tag{4.24}$$

More simply explained, within each of these nodes, t, a sum of squares can be calculated

$$\Sigma_{x_n \in t}(y_n - \bar{y}(t))^2). \tag{4.25}$$

Then we can calculate the average by summing over $t \in T$ for the total within node sum of squares then dividing by the N, the number of samples. A regression tree is formed iteratively so as to maximize the decrease in $R(t)$ [5]. To use the tree as a predictor, after building the tree, data can be followed down the tree until it reaches a terminal node. At that time, the predicted value is the average of all the responses within that terminal node. We will use this method to predict the potential percent gain for an investor from each loan. For this model, there is no class imbalance because there are no classes. We did not need to upsample before running the regression tree model. We did use all features and we used Expectation Maximization to impute missing data.

4.3 Random Forest

Random forests are structured very similarly to classification trees. They were introduced by Breiman in 2001 [6] and purposed to reduce error by growing a large number of classification trees and compiling the results from the individual models to provide a final outcome. An illustration of the structure of the random forest model is in figure 4.8. Growing a large number of trees typically results in significant improvements in accuracy over the single classification tree. The trees are set up by first selecting some number of best features and the best splits to grow a tree [35]. Then subsets of the samples are randomly chosen. This can be done different ways, with replacement or without replacement. For our purposes, we randomly selected with replacement and sub-sample size was always the same as the original input sample size. [32]. This method is called bootstrapping and will reduce over-fitting and variability within the model.

Breiman defines the margin function

$$mg(X,Y) = av_k I(h_k(X) = Y) - max_{j \neq Y} av_k I(h_k(X) = j), \tag{4.26}$$

where we denote our testing set X with coordinating label set Y and the ensemble of random forest classifiers $h_1(x), h_2(x), ..., h_k(x)$, where each $h(x)$ is trained on a random selection from X. The function $I(\cdot)$ is an indicator function. The margin function finds the difference between the average number of votes for the correct class and the average vote for any other class. The larger the margin

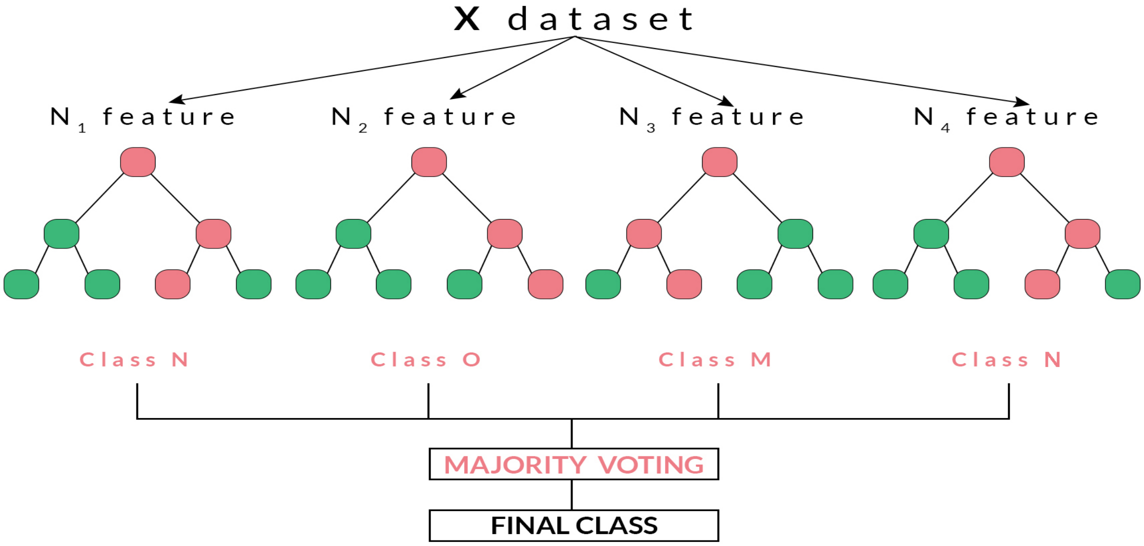


Figure 4.8 Basic Random Forest, taken from [37]

the more certainty we have that the classifications are correct. In Breiman's paper, he defines the generalization error

$$PE^* = P_{X,Y}(mg(X,Y)) < 0, \tag{4.27}$$

where the subscript X, Y indicates that the probability is over the X, Y space [6].

Theorem 4.3.1. *As the number of trees N increases, for almost surely all sequences $\Theta_1, \Theta_2, ..., \Theta_N$, then PE^* converges to*

$$P_{X,Y}(P_\Theta(h(X,\Theta) = Y) - max_{j \neq Y} P_\Theta(h(X,\Theta) = j) < 0). \tag{4.28}$$

The proof for this theorem can be found in [6]. Here $h_k(X)$ can be written $h_k(X, \Theta_k)$ where θ_k denotes the random subsets of X to be used by the algorithm. This result explains why random forests do not over-fit as more trees are added but instead produce a limiting value of the generalization error [6]. While the random forest model combats the issue of overfitting that we see in the classification tree model, it also creates an issue with interpretability.

4.4 Contributions

While our machine learning methods that have been introduced in this chapter, have all been used before, no where in the literature are researchers using multi-class classification trees to predict time until default. Also using a Regression Tree to predict percent gain from an investment in a loan is, to our knowledge, a novel way to make this prediction.

CHAPTER

5 PREDICTION RESULTS

We completed all of our model runs in Python with the exception of the Survival Analysis model, which was implemented in SAS. In Python, we used the machine learning packages imported from scikit-learn. In SAS, we used PROC PHREG to obtain results for our comparison with survival analysis. In all models we implemented sampling to combat the class imbalance where appropriate. We used Expectation Maximization to impute missing data for all models run. All features that were created through feature engineering were used to create all of the following model results.

5.1 Results for Classification Trees

We are predicting several things with these models. First, we are binarily predicting if loans will default or not using data from Lending Club. Second, we are predicting if an investor will make a profit or not if they choose to invest in a loan. This is also a binary prediction, but not the same as the first. In this prediction a loan could still default, but the investor make a profit. It all depends on when the loan defaults. Our third model is a multi-class classification tree that aims to predict a time until default.

In table 5.1, we can see the error rates for all classification tree models. All models include Sampling, imputation using Expectation Maximization, features that were engineered, and all were pruned using the same methods.

As we would expect, both binary models have an error rate that is less than the multi-class error rate. This is expected, because there is more opportunity for error when we introduce the possibility of predicting more classes within the default class. This is notably the same between the binary prediction and the multi-class prediction, because the only variance is the increased number of bins for default.

Table 5.1 Error Rates for Classification Tree

	Multi-Class Label	Binary Label	Break-Even Label
Classification Tree	23.46%	21.63%	19.67%

In the confusion matrix in table 5.2, we can see the distributions of loans into each class as they were predicted vs actual. Loans that appear in red are predicted as being paid-in-full but are actual defaulted loans. We must remember that funding a loan is a non secured investment. The loans in red are far more detrimental to investment strategy than loans in green. A loans in green simply result in a missed opportunity to make a profit. In [33], Ramirez explores loan default as a binary classification problem. He builds a decision tree classifier and his error rate is 22.2% for his single tree model.Our error rate is lightly lower.

Table 5.2 Confusion Matrix for Multi-class Classification Tree

		Predicted				
		Paid-in-Full	Default Year 1	Default Year 2	Default Year 3	All
Actual	Paid-in-Full	13121	741	903	493	15258
	Default Year 1	642	111	54	17	824
	Default Year 2	731	51	111	27	920
	Default Year 3	409	19	20	60	508
	All	14903	922	1088	597	17510

We have computed several metrics for our binary predictions based on the confusion matrix. For the Binary classification the confusion matrix can be found in table 5.3. The Sensitivity/Recall score is 87.38%, the Specificity score is 17.20%, and the Precision score is 87.75%. These scores show that while we are predicting actual paid-in-full loans as such at a high rate, we are not predicting loans that will default correctly nearly as often. Because the idea of having false positives are far

Table 5.3 Confusion Matrix for Binary Classification Tree

		Predicted		
		Paid-in-Full	Default	All
Actual	Paid-in-Full	12652	1827	14479
	Default	1767	367	2134
	All	14419	2194	16613

more detrimental to investors, our low specificity score may indicate another model may be more beneficial where we can penalize the prediction of paid-in-full when the loan defaults.

Table 5.4 Confusion Matrix for Break-Even Classification Tree

		Predicted		
		Gain from Investment	Loss from Investment	All
Actual	Gain from Investment	13755	1734	15489
	Loss from Investment	1710	311	2021
	All	15465	2045	17510

Again, we calculated the Sensitivity score (88.80%), Specificity score (15.39%), and Precision score (88.94%) for the break-even binary model. The confusion matrix for this model can be seen in table 5.4. These results are similar to the last. While our model is accurate 80.33% of the time, the specificity score tells us that we are frequently saying that an investor will gain from their investment when they will not.

5.2 Survival Analysis

In Survival Analysis, the aim is to estimate the distribution of the event times $f(t)$ of a group of subjects. $f(t)$ can also be referred to as the probability distribution function (pdf). That is, the function that describes the likelihood of observing Time (T) at time (t) relative to all other survival times. We will define $F(t)$ as follows:

$$F(t) = \int_0^t f(t)dt. \tag{5.1}$$

$F(t)$ is the probability of observing Time (T) less than or equal to some time (t). $F(t)$ is also called the cumulative distribution function (cdf). To derive the Survival Function, $S(t)$, we must transform the cdf to

$$S(t) = Pr(T > t) = 1 - F(t). \tag{5.2}$$

The Survival Function describes the probability of surviving past time t. It is a monotone decreasing function of t with $S(0) = 1$ and $lim_{t->\infty} S(t) = 0$ [2]. The Hazard Function, $h(t)$, takes the limit as $dt \rightarrow 0$ of the probability of an event happening in a time interval $[t, t+dt)$ given that the subject has survived until time t divided by dt since the probability increase with dt. By taking the limit, the hazard function quantifies the instantaneous rate of occurrence at time t [2].

$$h(t) = lim_{dt \rightarrow 0} \frac{Pr(t \leq T < t+dt | T \geq t)}{dt}. \tag{5.3}$$

The conditional probability in the numerator may be thought of as a ratio. The numerator of which is the joint probability that T is in the interval $[t, t+dt)$ and $T \geq t$, which, by definition, can be written as $f(t)dt$, for small dt. The denominator is the probability of $T \geq t$, which is $S(t)$. Thus we have,

$$\begin{aligned} h(t) &= lim_{dt \rightarrow 0} \frac{\frac{f(t)dt}{S(t)}}{dt} \\ &= \frac{f(t)}{S(t)} \end{aligned} \tag{5.4}$$

This is the connection between the survival function and the hazard function [34].

There are several variations to the survival analysis model, but the most commonly used variation, when attempting to predict time until default, is the Cox proportional hazard model (Cox model). In Chapter 1, our prior work survey describes this work. The Cox model, has a hazard function of the form

$$h(t, x_i) = h_0(t) exp(\beta^T x_i) \tag{5.5}$$

This equation says that the hazard for a subject i at time t is the product of an unspecified, positive baseline hazard function $h_0(t)$, and a linear function of a vector of inputs x_i which is exponentiated. The baseline hazard $h_0(t)$ is a function of time only, and is assumed to be the same for all subjects. The name "proportional hazard" stems from the fact that the hazard of any individual is a fixed proportion of the hazard of any other individual over time. The β parameters of the proportional hazards model can be estimated without having to specify the baseline hazard function $h_0(t)$. Therefore, the proportional hazards model is most often called a semi-parametric model. The estimation of the β coefficients is done by using the partial likelihood principle [2], which can be found in [33].

For the purpose of comparing our results to more typical results in the field, we are using a survival analysis model.

5.3 Comparison of Results

To compare our results, we first ran a survival analysis model on our Lending Club data. We used the same data that we used to run the classification model. A graph of the results can be seen in figure 5.1. As time (in months) progresses the probability of survival decreases. Near the end of the curve, we see a dramatic drop off. This is caused by the loans that exceed 36 month term due to collections efforts. They, while is still classified under the 36 month category, are past due and frequently default. Our curve, has removed all loans that exceed the 38 months.

To compare our classification tree model results to the survival analysis results, we subtracted loans cumulatively that were predicted to default at each month from the total number of loans. We then divided that by the total number of loans and multiplied by 100 to get a percent.

$$SP = \frac{T - \Sigma_0^n t_n}{T} \times 100, \tag{5.6}$$

where T is the total number of loans and t_n are the loans that default up to that month. We plotted the Survival probability from the classification tree by time in months in figure 5.3. We can see that, in our classification tree plot in figure, as time progresses the probability of survival decreases, but less sharply than that of the survival analysis curve. In figure 5.2, we plot the actual survival plot using equation 5.6 but instead of subtracting the loans that were predicted to default, we subtract loans that actually defaulted using the label on each observation.

By looking at the three graphs, it is clear that the survival analysis results are closer to the actual results which show the probability of survival. However, because these models are made to predict somewhat different things, survival analysis being the probability of survival given certain attributes and classification tree being in what year are you likely to default, we are limited on our comparison. The classification tree model makes predictions that are difficult to obtain using the survival analysis results. Using the classification tree results, an investor can follow a potential investments attributes through the tree to see how our model would classify them and determine with some level of accuracy within what year they are likely to default, if at all. It can be seen, through this comparison, that while the model's outputs create different curves there are consistences in the curvature, which tells us that the models are picking up many of the same patterns present in the data.

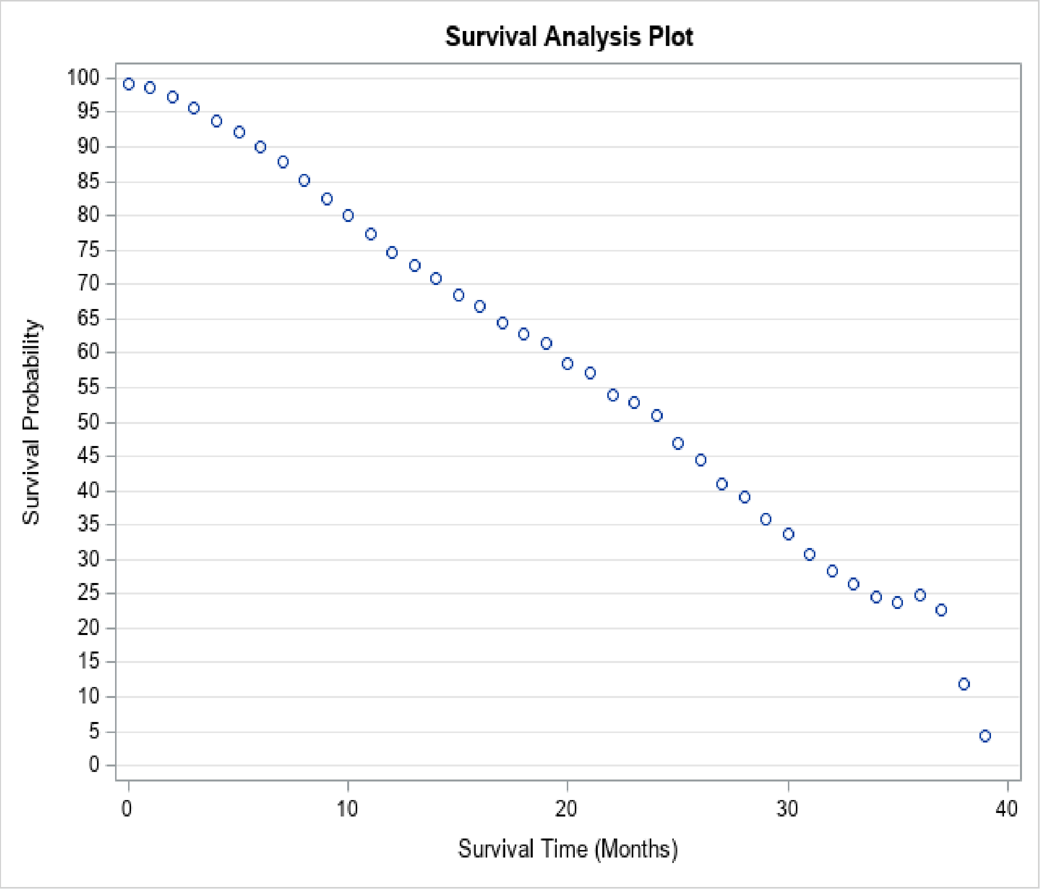


Figure 5.1 Survival Curve by month is created using the survival analysis model.

5.4 Results for Random Forests

We made the same predictions using a random forest model. While the error rate is somewhat lower than with the classification tree and the specificity score is higher, the interpretability is less simple for a random forest because of the number of trees that are created by each model. Each tree is trained on bagged data using a random selection of features, thus gaining a full understanding of the decision process by examining each individual tree is far more difficult than a classification or regression tree.

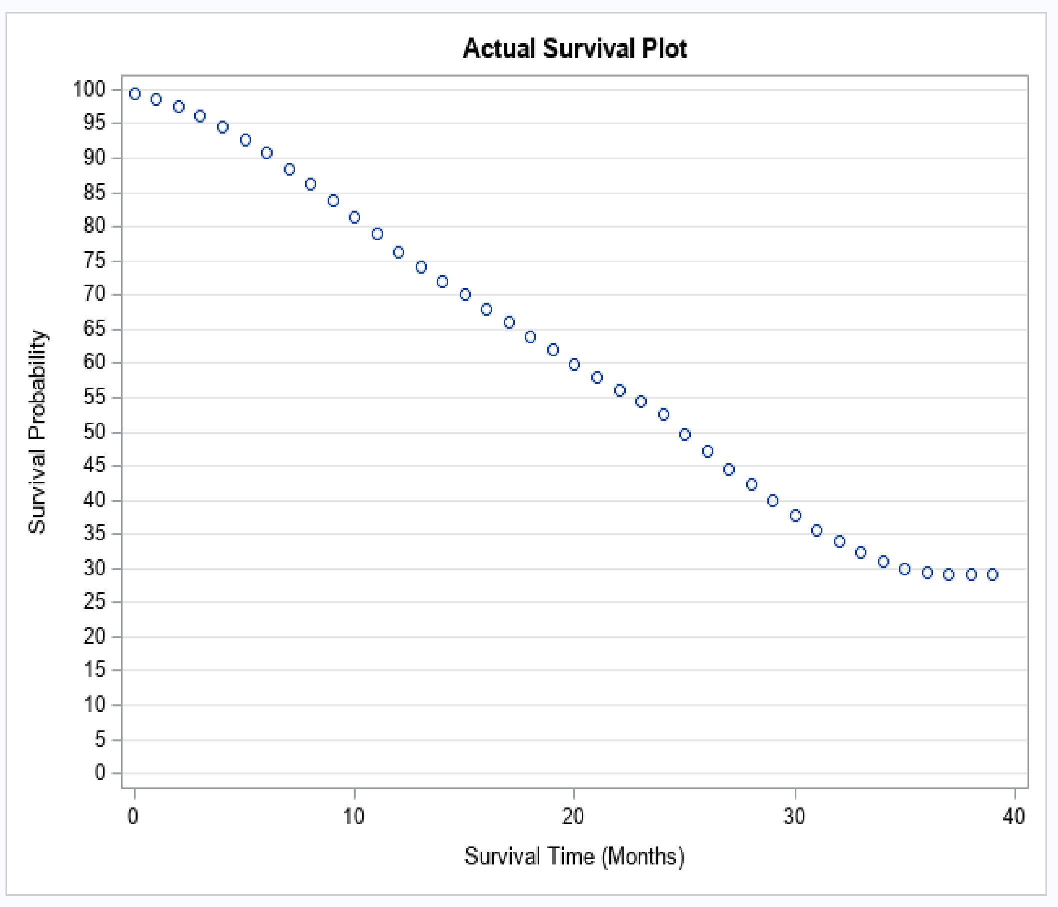


Figure 5.2 Actual Survival Plot is based on label instead of prediction.

Table 5.5 Error Rates for Random Forest Model

	Multi-Class Label	Binary Label	Break-Even Label
Random Forest	21.91%	18.99%	16.77%
Number of Trees Grown	7	50	30

In the confusion matrix in table 5.6, we can, again, see the distributions of loans into each class as they were predicted vs actual. Loans that appear in red are predicted as being paid-in-full but are actual defaulted loans. We must remember that funding a loan is a non secured investment. The loans in red are far more detrimental to investment strategy than loans in green. A loans in green simply result in a missed opportunity to make a profit.

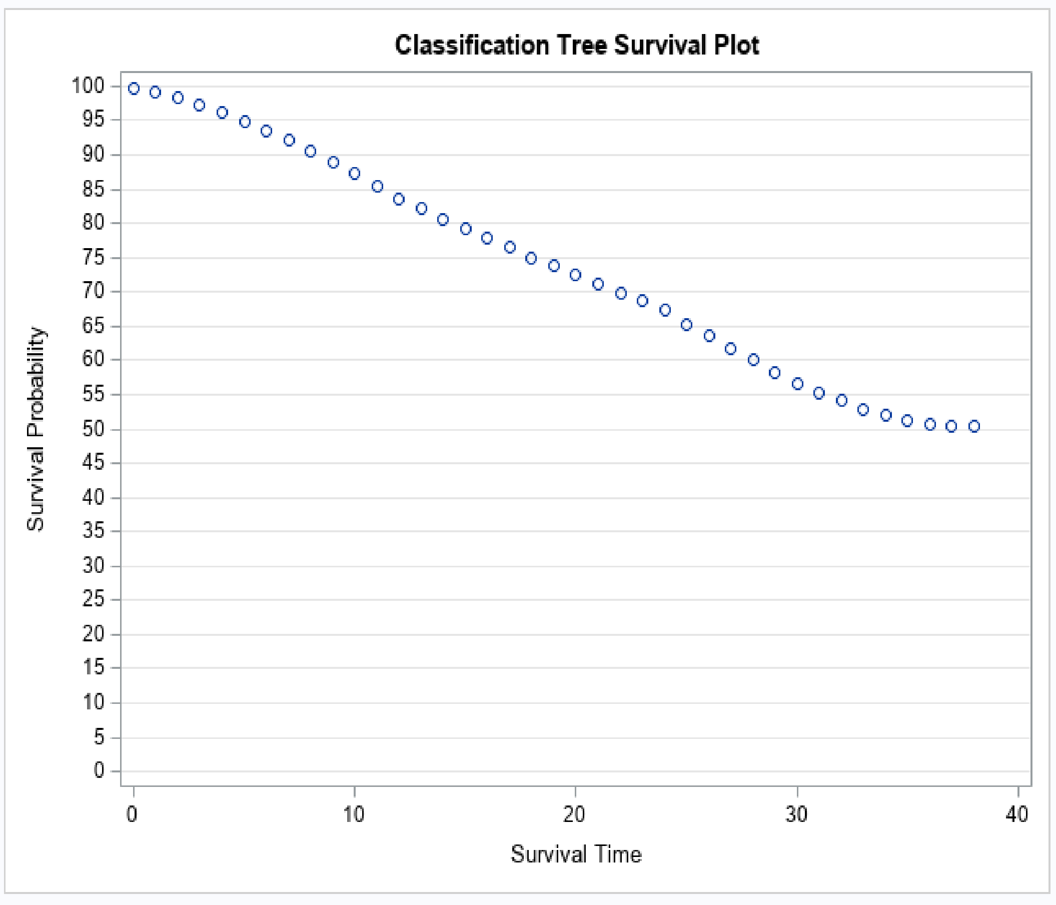


Figure 5.3 Classification Tree Survival Plot is created using the classification tree model.

Table 5.6 Confusion Matrix for Multi-class Random Forest

		Predicted				
		Paid-in-Full	Default Year 1	Default Year 2	Default Year 3	All
Actual	Paid-in-Full	13494	726	801	237	15258
	Default Year 1	650	88	68	18	824
	Default Year 2	751	65	80	24	920
	Default Year 3	418	39	39	12	508
	All	15313	918	988	291	17510

Table 5.7 Confusion Matrix for Binary Random Forest

		Predicted		
		Paid-in-Full	Default	All
Actual	Paid-in-Full	12977	1502	14479
	Default	1653	481	2134
	All	14630	1983	16613

Table 5.8 Performance Metrics for Binary Random Forest

Sensitivity/Recall	89.63%
Specificity	22.54%
Precision	88.70%

Table 5.9 Confusion Matrix for Break Even Random Forest

		Predicted		
		Gain from Investment	Loss from Investment	All
Actual	Gain from Investment	13504	1197	14701
	Loss from Investment	1590	322	1912
	All	15094	1519	16613

Table 5.10 Performance Metrics for Break-Even Random Forest

Sensitivity/Recall	91.86%
Specificity	16.84%
Precision	89.47%

5.5 Regression Tree Results

In our regression tree model we are trying to predict the net annualized return (NAR) or percent gain. We did this by looking at two metrics coming out of our model- the mean squared error (MSE), which is defined in chapter 4 and again here, and the coefficient of determination R^2.

$$R^2 = 1 - \frac{\Sigma_{i=1}^{m}(\hat{y}_i - y_i)^2}{\Sigma_{i=1}^{m}(y_i - \bar{y})^2} \tag{5.7}$$

$$MSE = \frac{1}{m}\Sigma_{i=1}^{m}(\hat{y}_i - y_i)^2, \tag{5.8}$$

where m is the number of samples that are considered for the split, $\bar{y}$ is the mean of the label, and $\hat{y}$ is the models prediction of the label.

To predict the percent gain for each loan, we first calculated the percent of the loan amount that the investor would make. We did this by determining the amount above principle the investor would make. Then dividing by the loan amount. This percent would be negative if the investor lost the investment because of early default. We used that as the target to train and determine the accuracy of our regression model.

Though it is slight, it can be seen in table 5.11 that as we allow the depth to grow deeper, bias increases while the MSE decreases. This is likely due to the idea that as the decision tree is allowed to grow it is able to capture non-linear relationships among the data.

Table 5.11 Regression Tree Results

	MSE	R^2
Regression Tree (Max Depth 4)	0.059304	0.0101
Regression Tree (Max Depth 8)	0.059096	0.01358
Regression Tree (Max Depth 10)	0.059093	0.01362

The root of the MSE for the Regression Tree with max depth 10 is $\sqrt{0.059093} = 0.24309$, which tells us that our predicted percent gain differs from the true percent gain on average by 0.24. If a loan is going to default, most do soon after payments begin and have percent "gain" on average of -0.49. If we think of our root of the MSE as somewhat of a confidence interval around the average percent "gain" of -0.49, we can see our model predicts, on average, if the investment will result in gain or in a loss.

The entire output is slightly too large when printed to read on one page, therefore an example of the output of a tree model is located in figure 5.4. On each node that is displayed, the MSE, number of samples, and the feature cut point is printed, making interpretation straightforward, similar to the classification tree [26].

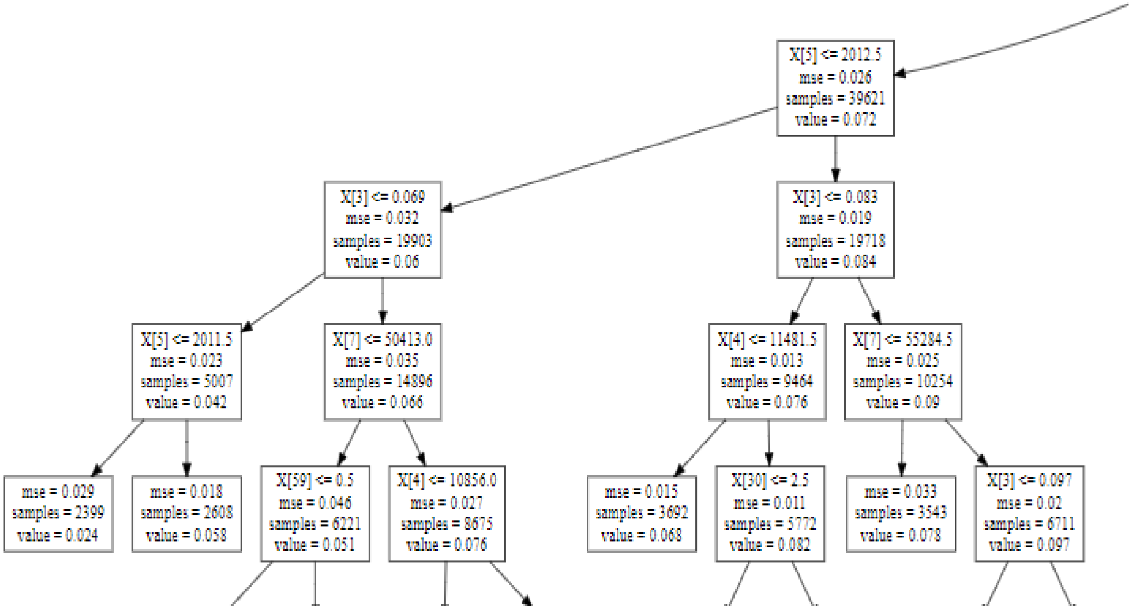


Figure 5.4 On this regression tree result plot, each node displays, the MSE, number of samples, and the feature cut point, making interpretation straightforward.

5.6 Contributions

In this chapter we have shown that our models are comparable with similar methods for solving these problems in the field, and in most cases, can supply investors with useful information needed to make educated decisions on investments based on attributes of borrowers. In some cases, as we have noted, our models do not make a prediction that, by some accuracy measures, is reliable enough to solely base a decision on investment. In these cases, domain knowledge as well as other models used in the field would be a more appropriate consult.

CHAPTER

6

CONCLUSION AND FUTURE WORK

6.1 Conclusion

Online Peer-to-Peer lending is rapidly becoming a more popular way to borrow money and invest. Loan default by extension is a growing concern for the peer-to-peer loans. In our study we use data from Lending Club to predict loan default in 5 main ways. We use a data-driven approach to build two binary classification tree models that predict if a loan will default and if the investor will make money on the loan regardless of the loan status. We then build a multi-class classification tree that predicts the time until default. This is only granular by year. We replicate these predictions using random forest models and find the results are somewhat better, but harder to interpret. Lastly, we predict the percent gain an investor is likely to receive from his/her investment with a regression tree.

We have made a few key contributions in our work.

- Through data processing, imputation, and feature engineering, we set up our data in such a way to apply models and reach a greater efficiency and accuracy.
- We incorporate new features into our models through feature engineering and show how they are relevant to each model run.
- We extend binary results to multi-class classification trees to predict time until default in a unique way. We then compare our results to those methods that are used in the field, namely

survival analysis models.

- We also experiment with the use of the random forest models to create the same prediction. Our metrics show that this is a better model, but as discussed, possibly less interpretable.

- We use a regression tree to predict percent gain from an investment in a loan. We have shown why our model is relevant in the field based on our results.

6.2 Future Work

Countless people can benefit monetarily by using methods described in this dissertation to determine default, not just in peer-to-peer lending. The study could be continued on longer loan term data and less risky loans, for example, commercial loans, mortgages, student loans. Also, early repayment reduces the amount of profit on a loan, by reducing the amount of interest that is paid. Expected profit could be incorporated into this analysis and more explicitly predicted.

Further exploration into ways of using classification trees to predict time until default could be a next step. As expected we found the the more granular the prediction of time became the less accurate the model. Investigating the idea of multiple binary predictions could be a next step. This would allow the first prediction to be default or not, which had a low error rate. Then a separate model could be run to take the output of the first model and predict time until the default will take place.

BIBLIOGRAPHY

[1] Attarian, A. “Patient Specific Subset Selection, Estimation and Validation of an HIV-1 Model with Censored Observations under an Optimal Treatment Schedule”. *NC State University, Ph.D. Thesis* (2012).

[2] Baesens Bart, e. a. “Neural Network Survival Analysis for Personal Loan Data”. *Journal of the Operational Research Society* (2004), pp. 1089–1098.

[3] Banasik, J. e. a. “Not if but when will borrowers default”. *The Journal of the Operational Research Society* (1999), 1185âĂŞ1190.

[4] Blagus Rok, e. a. “SMOTE for high-dimensional class-imbalanced data”. *BMC Bioinformatics* (2013).

[5] Breiman Leo, e. a. *Classification and Regression Trees*. Brooks/Cole, 1984.

[6] Breiman, L. “Random Forests” (2001).

[7] *Census Regions and Divisions of the United States*. URL: `http://www.census.gov/geo/maps-data/maps/pdfs/reference/us_regdiv.pdf`.

[8] Chawla Nitesh, e. a. “SMOTE:Synthatic Minority Over-sampling Technique”. *Journal of Artificial Intelligence Research* (2002), pp. 321–357.

[9] “Data Mining - Principal Component (Analysis Regression) (PCA)” (2018).

[10] David R.H., e. a. “Machine learning algorithms for credit-card applications”. *IMA Journal of Mathematics Applied in Business and Industry* (1992), pp. 43–51.

[11] Dempster, A. et al. “Maximum Likelihood from Incomplete Data via the EM Algorithm”. *Journal of the Royal Statistical Society* (1977), pp. 1–38.

[12] Desai., e. a. “A comparison of neural networks and linear scoring models in the credit union environment”. *European Journal of Operational Research* (1996), pp. 24–37.

[13] Dietterich Thomas G., e. a. “Solving multiclass learning problems via error-correcting output codes”. *The Handbook of Brain Theory and Neural Networks* **2** (2002).

[14] Dietterich, T. G. “Ensemble Learning”. *Journal of artificial intelligence research* **2** (1995), pp. 263–286.

[15] Henley W.E., e. a. “Construction of a k-nearest neighbour credit-scoring system”. *IMA Journal of Mathematics Applied in Business and Industry* (1997), pp. 305–321.

[16] Hoang, P. “Supervised Learning in Baseball Pitch Prediction and Hepatitis C Diagnosis”. *NC State University, Ph.D. Thesis* (2015).

[17] Jha, V. “Machine Learning Algorithm - Backbone of emerging technologies” (2017).

[18] Jim Yu, e. a. “A Data-Driven Approach to Predict Default Risk of Loan for Online Peer-to-Peer (P2P) Lending”. 2015 Fifth International Conference on Communication Systems and Network Technologies. IEEE, 2015.

[19] “K-Means Clustering in Python” (2017).

[20] Kunert, R. “SMOTE explained for noobs - Synthetic Minority Over-sampling TEchnique line by line” (2017).

[21] *Lending Club*. URL: `https://www.lendingclub.com/company/contact-us/corporate-headquarters`.

[22] *Lending Club*. URL: `https://www.lendingclub.com/investing/investor-education/interest-rates-and-fees`.

[23] *Lending Club*. URL: `https://www.lendingclub.com/investing/investor-education/what-are-the-terms-of-the-loans`.

[24] *Lending Club*. URL: `https://help.lendingclub.com/hc/en-us/articles/216127747`.

[25] *Lending Club*. URL: `https://help.lendingclub.com/hc/en-us/articles/216127917-What-to-expect-when-a-loan-is-late-`.

[26] Li, P. & Han, G. “Lending Club Loan Default and Profitablily Prediction” ().

[27] Little R.J.A., e. a. *Statistical analysis with missing data*. John Wiley Sons, Inc., 2002.

[28] Mues Christophe, e. a. “Decision diagrams in machine learning: an empirical study on real-life credit-risk data”. *Expert Systems with Applications* (2004), pp. 257–264.

[29] Narain, B. “Survival Analysis and the Credit Granting Decision”. *Credit Scoring and Credit Control* (1992), pp. 109–121.

[30] Navlani, A. “KNN Classification using Scikit-learn” (2018).

[31] Patel Nikita, e. a. “Study of Various Decision Tree Pruning Methods with their Empirical Comparison in WEKA”. *0975-8887, International Journal of Computer Applications* (2012).

[32] Pedregosa, F. e. a. “Scikit-learn: Machine Learning in Python”. *Journal of Machine Learning Research* **12** (2011), pp. 2825–2830.

[33] Ramirez, A. “Analysis of Default in Peer to Peer Lending”. *University of Califorinia Los Angeles, Masters Thesis* (2016).

[34] Rodriguez, G. *Lecture Notes on Generalized Linear Models*. 2007. URL: http://data.princeton.edu/wws509/notes/.

[35] Sidle, G. "Using Multi-Class Machine Learning Methods to Predict Major League Baseball Pitches". *NC State University, Ph.D. Thesis* (2017).

[36] Steenackers A., e. a. "A credit scoring model for personal loans". *Insurance: Mathematics and Economics* (1989), pp. 31–34.

[37] Tahsildar, S. *Random Forest Algorithm In Trading Using Python*. 2019. URL: https://blog.quantinsti.com/random-forest-algorithm-in-python/ (visited on 05/23/2019).

APPENDIX

APPENDIX

A

FULL FEATURE DESCRIPTION

Table A.1 All Features Used with Description.

Feature Name	Decription
acc_now_delinq	The number of accounts on which the borrower is now delinquent.
addr_state	The state provided by the borrower in the loan application
annual_inc	The self-reported annual income provided by the borrower during registration.
chargeoff_within_12_mths	Number of charge-offs within 12 months
collections_12_mths_ex_med	Number of collections in 12 months excluding medical collections
delinq_2yrs	The number of 30+ days past-due incidences of delinquency in the borrower's credit file for the past 2 years
delinq_amnt	The past-due amount owed for the accounts on which the borrower is now delinquent.
dti	A ratio calculated using the borrower's total monthly debt payments on the total debt obligations, excluding mortgage and the requested LC loan, divided by the borrower's self-reported monthly income.

Table A.1 (continued)

Feature Name	Decription
earliest_cr_line	The month the borrower's earliest reported credit line was opened
emp_length	Employment length in years. Possible values are between 0 and 10 where 0 means less than one year and 10 means ten or more years.
grade	LC assigned loan grade
home_ownership	The home ownership status provided by the borrower during registration or obtained from the credit report. Our values are: RENT, OWN, MORTGAGE, OTHER
id	A unique LC assigned ID for the loan listing.
initial_list_status	The initial listing status of the loan. Possible values are âĂŞ W, F (Whole and fractional)
inq_last_6mths	The number of inquiries in past 6 months (excluding auto and mortgage inquiries)
int_rate	Interest Rate on the loan
issue_d	The month which the loan was funded
last_pymnt_d	Last month payment was received
loan_amnt	The listed amount of the loan applied for by the borrower. If at some point in time, the credit department reduces the loan amount, then it will be reflected in this value.
loan_status	Current status of the loan
open_acc	The number of open credit lines in the borrower's credit file.
pub_rec	Number of derogatory public records
pub_rec_bankruptcies	Number of public record bankruptcies
purpose	A category provided by the borrower for the loan request.
revol_bal	Total credit revolving balance
revol_util	Revolving line utilization rate, or the amount of credit the borrower is using relative to all available revolving credit.
sub_grade	LC assigned loan subgrade
tax_liens	Number of tax liens
term	The number of payments on the loan. Values are in months and can be either 36 or 60.
total_acc	The total number of credit lines currently in the borrower's credit file

Table A.1 (continued)

Feature Name	Decription
verification_status	Indicates if income was verified by LC, not verified, or if the income source was verified
zip_code	The first 3 numbers of the zip code provided by the borrower in the loan application.

Table A.2 All Features Unused with Description.

Feature Name	Decription
acc_open_past_24mths	Number of trades opened in past 24 months.
all_util	Balance to credit limit on all trades
annual_inc_joint	The combined self-reported annual income provided by the co-borrowers during registration
application_type	Indicates whether the loan is an individual application or a joint application with two co-borrowers
avg_cur_bal	Average current balance of all accounts
bc_open_to_buy	Total open to buy on revolving bankcards.
bc_util	Ratio of total current balance to high credit/credit limit for all bankcard accounts.
collection_recovery_fee	post charge off collection fee
desc	Loan description provided by the borrower
dti_joint	A ratio calculated using the co-borrowers' total monthly payments on the total debt obligations, excluding mortgages and the requested LC loan, divided by the co-borrowers' combined self-reported monthly income
emp_title	The job title supplied by the Borrower when applying for the loan.
fico_range_high	The upper boundary range the borrower's FICO at loan origination belongs to.
fico_range_low	The lower boundary range the borrower's FICO at loan origination belongs to.
funded_amnt	The total amount committed to that loan at that point in time.
funded_amnt_inv	The total amount committed by investors for that loan at that point in time.

Table A.2 (continued)

Feature Name	Decription
il_util	Ratio of total current balance to high credit/credit limit on all install acct
inq_fi	Number of personal finance inquiries
inq_last_12m	Number of credit inquiries in past 12 months
installment	The monthly payment owed by the borrower if the loan originates.
last_credit_pull_d	The most recent month LC pulled credit for this loan
last_fico_range_high	The upper boundary range the borrowerâĂŹs last FICO pulled belongs to.
last_fico_range_low	The lower boundary range the borrowerâĂŹs last FICO pulled belongs to.
last_pymnt_amnt	Last total payment amount received
max_bal_bc	Maximum current balance owed on all revolving accounts
member_id	A unique LC assigned Id for the borrower member.
mo_sin_old_il_acct	Months since oldest bank installment account opened
mo_sin_old_rev_tl_op	Months since oldest revolving account opened
mo_sin_rcnt_rev_tl_op	Months since most recent revolving account opened
mo_sin_rcnt_tl	Months since most recent account opened
mort_acc	Number of mortgage accounts.
mths_since_last_delinq	The number of months since the borrower's last delinquency.
mths_since_last_major_derog	Months since most recent 90-day or worse rating
mths_since_last_record	The number of months since the last public record.
mths_since_rcnt_il	Months since most recent installment accounts opened
mths_since_recent_bc	Months since most recent bankcard account opened.
mths_since_recent_bc_dlq	Months since most recent bankcard delinquency
mths_since_recent_inq	Months since most recent inquiry.
mths_since_recent_revol_-delinq	Months since most recent revolving delinquency.
next_pymnt_d	Next scheduled payment date
num_accts_ever_120_pd	Number of accounts ever 120 or more days past due
num_actv_bc_tl	Number of currently active bankcard accounts
num_actv_rev_tl	Number of currently active revolving trades
num_bc_sats	Number of satisfactory bankcard accounts
num_bc_tl	Number of bankcard accounts

Table A.2 (continued)

Feature Name	Decription
num_il_tl	Number of installment accounts
num_op_rev_tl	Number of open revolving accounts
num_rev_accts	Number of revolving accounts
num_rev_tl_bal_gt_0	Number of revolving trades with balance >0
num_sats	Number of satisfactory accounts
num_tl_120dpd_2m	Number of accounts currently 120 days past due (updated in past 2 months)
num_tl_30dpd	Number of accounts currently 30 days past due (updated in past 2 months)
num_tl_90g_dpd_24m	Number of accounts 90 or more days past due in last 24 months
num_tl_op_past_12m	Number of accounts opened in past 12 months
open_acc_6m	Number of open trades in last 6 months
open_il_12m	Number of installment accounts opened in past 12 months
open_il_24m	Number of installment accounts opened in past 24 months
open_act_il	Number of currently active installment trades
open_rv_12m	Number of revolving trades opened in past 12 months
open_rv_24m	Number of revolving trades opened in past 24 months
out_prncp	Remaining outstanding principal for total amount funded
out_prncp_inv	Remaining outstanding principal for portion of total amount funded by investors
pct_tl_nvr_dlq	Percent of trades never delinquent
percent_bc_gt_75	Percentage of all bankcard accounts > 75% of limit.
policy_code	"publicly available $policy\_code = 1$ new products not publicly available $policy\_code = 2$"
pymnt_plan	Indicates if a payment plan has been put in place for the loan
recoveries	post charge off gross recovery
title	The loan title provided by the borrower
tot_coll_amt	Total collection amounts ever owed
tot_cur_bal	Total current balance of all accounts
tot_hi_cred_lim	Total high credit/credit limit
total_bal_ex_mort	Total credit balance excluding mortgage
total_bal_il	Total current balance of all installment accounts
total_bc_limit	Total bankcard high credit/credit limit

Table A.2 (continued)

Feature Name	Decription
total_cu_tl	Number of finance trades
total_il_high_credit_limit	Total installment high credit/credit limit
total_pymnt	Payments received to date for total amount funded
total_pymnt_inv	Payments received to date for portion of total amount funded by investors
total_rec_int	Interest received to date
total_rec_late_fee	Late fees received to date
total_rec_prncp	Principal received to date
total_rev_hi_lim	Total revolving high credit/credit limit
url	URL for the LC page with listing data.
verified_status_joint	Indicates if the co-borrowers' joint income was verified by LC, not verified, or if the income source was verified
revol_bal_joint	Sum of revolving credit balance of the co-borrowers, net of duplicate balances
sec_app_fico_range_low	FICO range (high) for the secondary applicant
sec_app_fico_range_high	FICO range (low) for the secondary applicant
sec_app_earliest_cr_line	Earliest credit line at time of application for the secondary applicant
sec_app_inq_last_6mths	Credit inquiries in the last 6 months at time of application for the secondary applicant
sec_app_mort_acc	Number of mortgage accounts at time of application for the secondary applicant
sec_app_open_acc	Number of open trades at time of application for the secondary applicant
sec_app_revol_util	Ratio of total current balance to high credit/credit limit for all revolving accounts
sec_app_open_act_il	Number of currently active installment trades at time of application for the secondary applicant
sec_app_num_rev_accts	Number of revolving accounts at time of application for the secondary applicant
sec_app_chargeoff_within_-12_mths	Number of charge-offs within last 12 months at time of application for the secondary applicant
sec_app_collections_12_-mths_ex_med	Number of collections within last 12 months excluding medical collections at time of application for the secondary applicant

Table A.2 (continued)

Feature Name	Decription
sec_app_mths_since_last_major_derog	Months since most recent 90-day or worse rating at time of application for the secondary applicant
hardship_flag	Flags whether or not the borrower is on a hardship plan
hardship_type	Describes the hardship plan offering
hardship_reason	Describes the reason the hardship plan was offered
hardship_status	Describes if the hardship plan is active, pending, canceled, completed, or broken
deferral_term	Amount of months that the borrower is expected to pay less than the contractual monthly payment amount due to a hardship plan
hardship_amount	The interest payment that the borrower has committed to make each month while they are on a hardship plan
hardship_start_date	The start date of the hardship plan period
hardship_end_date	The end date of the hardship plan period
payment_plan_start_date	The day the first hardship plan payment is due. For example, if a borrower has a hardship plan period of 3 months, the start date is the start of the three-month period in which the borrower is allowed to make interest-only payments.
hardship_length	The number of months the borrower will make smaller payments than normally obligated due to a hardship plan
hardship_dpd	Account days past due as of the hardship plan start date
hardship_loan_status	Loan Status as of the hardship plan start date
orig_projected_additional_accrued_interest	The original projected additional interest amount that will accrue for the given hardship payment plan as of the Hardship Start Date. This field will be null if the borrower has broken their hardship payment plan.
hardship_payoff_balance_amount	The payoff balance amount as of the hardship plan start date
hardship_last_payment_amount	The last payment amount as of the hardship plan start date
disbursement_method	The method by which the borrower receives their loan. Possible values are: CASH, DIRECT_PAY
debt_settlement_flag	Flags whether or not the borrower, who has charged-off, is working with a debt-settlement company.

Table A.2 (continued)

Feature Name	Decription
debt_settlement_flag_date	The most recent date that the Debt_Settlement_Flag has been set
settlement_status	The status of the borrowerâĂŹs settlement plan. Possible values are: COMPLETE, ACTIVE, BROKEN, CANCELLED, DENIED, DRAFT
settlement_date	The date that the borrower agrees to the settlement plan
settlement_amount	The loan amount that the borrower has agreed to settle for
settlement_percentage	The settlement amount as a percentage of the payoff balance amount on the loan
settlement_term	The number of months that the borrower will be on the settlement plan