Analysis of the probability of default in peer-to-peer lending

Application of different classification techniques

Master's Thesis in Statistics

Data Analysis

Endre Kvåle Evjen

University of Bergen

Department of Mathematics

November 2018

Abstract

In this thesis, peer-to-peer lending is explored and analyzed with the objective of fitting a model to accurately predict if borrowers default on their loans or not. The foundation for the thesis is a dataset from LendingClub, a peer-to-peer lending platform based in San Francisco, USA. Detailed information of borrowers' financial history, personal characteristics and the specifics of each loan is used to predict the probability of default for the various loans in the portfolio. Methods used include elastic net regularization of logistic regression, boosting of decision trees, and bagging with random forests. The results are compared using accuracy metrics and a profitability measure, before a final model selection is carried out.

Acknowledgements

I have thoroughly enjoyed the work on this thesis, and have learned a lot in the process. I would like to thank my supervisor Jan Bulla for providing help and advice on forming this thesis. Thanks to my family and friends for their support, and special thanks to my partner Ann Kristin Fossan for her continued love, patience and support. I could not have done this without you.

- Endre Kvåle Evjen, November 2018

Contents

List of Tables

List of Figures

Chapter 1

Introduction

The term people-to-people (P2P; person-to-person or peer-to-peer) lending describes lending and borrowing activities that occur directly among individuals [Wang et al., 2009]. P2P lending marketplaces are platforms that facilitate interactions between lenders and borrowers so that borrowers place requests for loans online, and private lenders bid to fund these in an auction-like process [Klafft, 2008] . Since the first P2P lending platform ZOPA was launched in 2005 [Bachmann et al., 2011], there has been a large influx of new marketplaces in many places all over the world. Notably large actors today are [prosper.com] and [LendingClub] in the US, [ZOPA] in the United Kingdom and [Smava] in Germany. While it is hard to find accurate data on how many different platforms exist as of today, it has surely seen an explosive growth in recent years. For instance, in China alone there were 4856 different services reported as of 2017 [Fintechnews Singapore, 2017].

These P2P platforms can largely be divided into two types; commercial and non-commercial. The main difference between the platforms is the lender's general intention and their expectation regarding returns [Bachmann et al., 2011]. Some platforms have a largely philanthropic approach, where the main goal is to provide financial assistance to people through microfinance. A Notable example of such a platform is Kiva [Kiva], a US based non-profit that allows lenders to invest in people and specific projects in impoverished regions of the world. Another project is Trine [Trine] that attempts to reduce carbon emissions by crowdfunding solar panels in economically underdeveloped regions of the world.

For the commercial platforms, the aim of the lender is to obtain profits on their investment. These platforms are providing an alternative investment opportunity for lenders by giving direct access to borrowers, and the lender is given a presumably reasonable interest for the risk they are taking.

For this thesis we will be looking at the latter kind of lending platform, in particular LendingClub [LendingClub]. Here the motivation for the lender is to earn profit and get an adequate return on their investment. For the borrower, the most common purpose of the loan is debt restructuring and credit card consolidations, and by applying for a loan through the P2P system the borrower is often able to reduce their interest rate to make it easier for themselves to pay the owed amount.

The use of a P2P lending marketplace can prove beneficial for both lenders and borrowers. The removal of the middleman - a role usually occupied by banks - will reduce the

cost of the facilitation of the loan. While the lender on a P2P platform will be unable to take collateral to reduce the risk of the loan, the lending platform provides information on the borrower to help alleviate the risk taken by the lender. The lender will be able to decide which loans fit their level of risk willingness. This increased access to information should lead to a wider range of loans being accepted, where the more risk-willing lenders take on loans that traditionally would not have been fulfilled by banks and other credit institutions. In this way it can improve access to the credit market for individuals not usually served, while also providing an acceptable rate of interest for the lenders. Taking LendingClub as an example, they report an adjusted net annualized return over all loan grades of 5.39 % for the period of our holdout loan data (Q4 2014 – Q1 2015). This includes both 3-year and 5-year loans, and also counts payments obtained through collection agencies after a loan has been charged off. This is beyond the scope of this paper however, but it gives some indication of what returns one might expect from a balanced portfolio of loans.

Figure 1: Loan performance details

ISSUE DATE START 2014 Q4 ISSUE DATE END 2015 Q1 UNITS % of issued dollars

	TOTAL ISSUED	FULLY PAID	CURRENT	LATE	CHARGED OFF (NET)	PRINCIPAL PAYMENTS RECEIVED	INTEREST PAYMENTS RECEIVED	AVG. INTEREST RATE	ADJ. NET ANNUALIZED RETURN[1]
A	100.00%	92.34%	0.60%	0.01%	2.36%	97.02%	9.93%	7.25%	4.79%
B	100.00%	75.20%	3.82%	0.09%	5.71%	90.37%	16.24%	10.50%	6.02%
C	100.00%	65.36%	4.75%	0.18%	10.44%	84.62%	21.37%	13.59%	5.96%
D	100.00%	56.16%	5.24%	0.37%	16.07%	78.32%	26.51%	16.66%	5.29%
E	100.00%	47.72%	6.02%	0.47%	22.06%	71.45%	31.55%	19.82%	4.47%
FG	100.00%	42.28%	5.29%	0.44%	29.99%	64.28%	36.12%	24.48%	2.48%
All	100.00%	68.18%	4.05%	0.20%	10.75%	85.00%	20.60%	13.30%	5.39%

Loan performance details for loans issued in the period used as holdout set.

The aim of this thesis is to see if we can outperform the average return by using traditional credit scoring methods to predict defaults, and by extension, picking the most optimal loans for a loan portfolio. We will do this by implementing several different statistical learning techniques to try to estimate the probability of default for each loan. We use these models to classify a set of unseen data and use statistical metrics and a profitability measure to compare the results.

The structure of the thesis is as follows. In Chapter 2 the data that is the basis of this thesis is presented. Variables are explained, exploratory data analysis is performed, and changes are made to prepare the data for statistical analysis. Chapter 3 presents the theoretical and technical background for the different modeling techniques implemented in the thesis. Chapter 4 presents the results from tuning the model hyperparameters, and the accuracy statistics when prediction is made on unseen data. Chapter 5 summarizes the thesis and presents opportunities for further research on the topic.

1.1 Literature review

Hand and Henley [1997] provide an extensive and in-depth look at the credit scoring methods applied in the financial industry. Some of the techniques they mention as standard for the industry are discriminant analysis, linear regression, logistic regression and decision trees. In addition, they mention neural networks and non-parametric methods such as nearest neighbor approaches. Survival analysis is also a popular method used to predict when in a loans lifetime a default may occur. In the recent years, the growth of various P2P lending markets has led to an increase in research aiming to provide credit scoring for this market. Similar to the traditional credit markets, methods of reducing risk are vitally important here. Since the P2P markets do not allow the lender to take any collateral on the loans, they face increased risk compared to traditional institutions. To compensate the lender for the risk they are taking, the loans usually carry a high interest rate. Emekter et al. [2015] finds that the increased interest rate given for the loans belonging to the lowest credit grade is not sufficiently high to compensate for the increased risk of default. It is thus imperative that the lender can successfully identify the loans that are paid back in full to be able to obtain a profitable portfolio of loans.

A lot of research has been carried out to try to accurately define the determinants of default, and to build credit scoring models for the P2P market to help alleviate the additional risk present in this marketplace.

With data from LendingClub, Emekter et al. [2015] used a binary logit regression model to find that the variables grade, debt-to-income ratio, FICO-score web [i], and revolving line utilization were the most important variables for predicting defaults. They use an older version of the LendingClub dataset from the period May 2007 to June 2012 which includes loans that are still current, so it is not a one-to-one comparison to the dataset used in this thesis.

Using a similar approach as Emekter et al. on LendingClub data from the period 2008 to 2011, Serrano-Cinca et al. [2015] finds that the important variables explaining defaults are loan purpose, annual income, current housing situation, credit history and indebtedness. They implement a logistic regression to predict defaults and find that the grade assigned by LendingClub is the most predictive factor of defaults. They ensure intertemporal validation by dividing the available data into a training set and holdout set from a later period. They also find that loan amount and length of employment has no significant impact on the rate of defaults.

Byanjankar et al. [2015] propose a credit scoring model using a neural network to classify loan applications into defaults and non-default groups. They find that the neural network fairly successfully classifies the loans into the correct categories. They also implement a logistic regression on the same dataset and find that the neural network outperforms the logistic regression in correctly classifying defaults.

Malekipirbazari and Aksakalli [2015] presents and compares different machine learning methods, including random forests, support vector machines, logistic regression and k-nearest neighbor classifiers. They find that the random forests method outperforms the other classification methods and stands as a scalable and powerful approach for predicting borrower status. They also find that random forests outperform both FICO scores and LendingClub grading system in identification of the best borrowers in terms of low probability of default, given some restrictions on which loans are included in the subset.

Li et al. [2018] designs an ensemble learner using extreme gradient boosting, logistic regression and a deep neural network. They use data from a lending platform based in China, and their results indicate that the model can effectively increase predictive accuracy compared to other machine learning models.

Serrano-Cinca and Gutiérrez-Nieto [2016] introduce an alternative approach to credit scoring they call profit scoring. This method builds models using estimated internal rate of return to predict the expected profitability rather than trying to accurately predict the loan status of the loans. They find that the variables that has the largest effect in determining loan profitability differ from the variables that has the largest effect on determining the probability of default. This is an indication that the market is not fully efficient. They also find that the profit scoring approach outperforms standard credit scoring methods.

Chapter 2

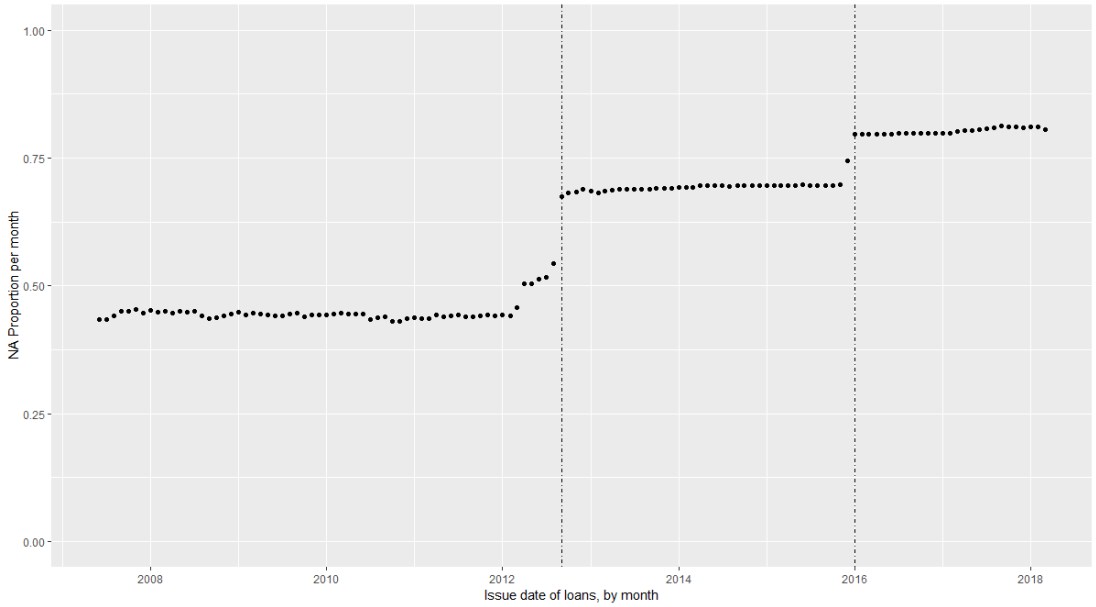
Data

This chapter describes the two datasets used in this thesis, and the transformation of the original data performed to make it suitable for my analysis. The first dataset contains the loan data for all the loans issued on LendingClub in the period September 2007 to March 2018. The second set contains the payment history for each loan contained in the first dataset. The cleaning and preparation of the data is explained, along with the reason for the choices of which variables and observations to include in the analysis. Exploratory data analysis is performed on the remaining dataset, including plots to visualize the data in more detail. The data preparation is done using `R`. The final dataset is included as an electronic attachment to the thesis.

2.1 Loan data set

The dataset is available for download directly from LendingClub [Loandata]. Creating an account and logging in to the website allows download of an extended dataset. It is this extended set that is used in this analysis. The data is split into separate files based on the issue date of the loan. The first file contains all loans issued between June 2007 and October 2011. The second file contains loans from November 2011 to October 2013. As LendingClub grew in popularity the number of loans issued increased to the point where they now issue a file containing new loans on a quarterly basis. Each separate datafile is updated quarterly to include the status of each loan. The combined dataset contains 1,870,526 loans and 128 variables. In the following subsections these variables will be explored and evaluated for use in the modeling phase.

Figure 2: Missing data evolvement over time



The average proportion of missing data in the loans for each month. Vertical dotted lines indicate the two points in time where LendingClub have extended the variable set.

Figure 2 shows the average rate of `NA` data for loans issued in each period. It is clear that additional variables have been added to the dataset twice. When these were added, they were understandably not retro-actively added to the loans issued prior to the changes. This leads to the two breakpoints seen in September 2012, and January 2016.

To get as complete a dataset as possible, the loans issued prior to September 2012 will be excluded from the final set. This amounts to 68,345 loans, which is 3.65% of the total loans. In the effort to gain consistent data for all the loans, we consider this tradeoff to be worth it. Another thing to note from Figure 2 is that the there is always missing data in the observations when the average is taken. This is due to the fact that many of the variables are concerned with the outlying cases where there are two loan applicants, so it is not surprising that we see a ceiling at about 80 % mean data presence.

Table 33 shows all the initial variables, the description of the variables, and how much of the data is missing for each variable. The count of missing observations is based on the data where loans prior to September 2012 has been removed. It is fairly simple to pick out the variables that were added in January 2016 based on the large number of variables that are missing a similar number of observations, as seen in Table 1. These missing observations are all (with a few exceptions) loans issued prior to January 2016. To keep a large dataset, and to extend the period over which the loans are issued, these variables will be removed from the dataset. The final dataset is then consistent over all the observations and variables, and should provide a good basis for analysis.

Table 1: New Variables January 2016

Variable name	Variable description	Missing
inq fi	Number of personal finance inquiries	794,938 / 44.1 %
inq last 12m	Number of credit inquiries in past 12 months	794,939 / 44.1 %
open acc 6m	Number of open trades in last 6 months	794,939 / 44.1 %
open act il	Number of installment accounts opened in past 12 months	794,938 / 44.1 %
open il 12m	Number of installment accounts opened in past 24 months	794,938 / 44.1 %
open rv 12m	Number of revolving trades opened in past 12 months	794,938 / 44.1 %
open rv 24m	Number of revolving trades opened in past 24 months	794,938 / 44.1 %
all util	Balance to credit limit on all trades	795,056 / 44.1 %
max bal bc	Maximum current balance owed on all revolving accounts	794,938 / 44.1 %
open il 24m	Number of currently active installment trades	794,938 / 44.1 %
total bal il	Total current balance of all installment accounts	794,938 / 44.1 %
total cu tl	Number of finance trades	794,939 / 44.1 %

Number of observations and proportion of missing observations for the given variables.

2.1.1 Variable analysis

The dataset contains variables which can be divided into a few broad categories; loan performance, loan characteristics, borrower characteristics, borrower credit history, current financial characteristics, and borrower assessment. Table 33 shows the variables within these categories and the following section will explore each category in turn. For this Chapter and going forwards, variables will be labeled in *italics*.

Borrower assesment

The Borrower Assesment variables are describing what the current credit rating the borrower has. In the US market, the FICO-score is a widely used rating for the credit worthiness of consumers. The dataset contains a *lower FICO range* and *higher FICO range*, which we will combine into an *average FICO rating* since the variables are perfectly correlated. The *grade* and *subgrade* are assigned by LendingClub to signify how risky each loan is. The loan grade and subgrade are the result of a formula that considers the credit score and a combination of several other indicators of credit risk from the credit report and loan application. The formula is, similarly to the formula for the FICO score, not made public. This is to prevent applicants from "gaming the system" and writing the perfect application to obtain better terms for their loan. The interest rate is also assigned by LendingClub. Interest rate is the sum of the LendingClub base interest rate and an adjustment for risk and volatility, where the adjustment is decided by what subgrade the loan is assigned to.

Borrower characteristics

Borrower characteristics describes information regarding the borrower. Examples are the *state* and *zip code* of residence, *annual income* and *home ownership* status. The zip code and state of residence of the borrower could give information relating to the probability of defaults. Especially if we included data from outside sources to go along with the analysis, such as median income by state or similar metrics. Since this data is not available, zip code and state of residence is removed from the dataset. The *employment title* variable contains 461,254 different employment titles out of which

121,735 are blank. Due to the huge number of job titles and the missing entries, this variable is dropped from the dataset.

There is not a lot of missing observations here, but the few loans that contain missing data will be excluded from the dataset. The remaining variables that will be kept are: *annual income*, *employment length*, *home ownership* and *verification status*. Description of all the variables can be found in Table 33.

Borrower indebtedness

In this category the variable *dti* gives the ratio between debt and income for the borrower. An additional variable is created to measure how large the impact of an additional debt burden is on the finances of the borrower. This variable is *monthly debt rate* and it calculates how large a percentage of the reported monthly income will be consumed by the installments on this new loan.

Borrower credit history

Borrower credit history contains information on the borrowers financial past. These variables are for the most part discrete numeric variables indicating for instance how many delinquencies the borrower has in the past two years. Other measures are how many accounts of various types are on the borrowers record, how many credit inquiries have been made, and whether the borrower has any bankruptcies on their public record. Several variables are records of how many months ago certain events occurred. For instance, how many months since the most recent installment account opened or how many months since most recent inquiry was made. For these variables there are a lot of missing entries. Some are missing as much as 80 % of the observations. In these cases the missing observations are taken to mean that the event has not occurred previously in the borrowers credit history. for instance, the variable *months since last public record* is `NA` for 83.4 % of the observations. There are also observations where the event is given 0 as value. These entries are taken to mean that the event occurred within the last month prior to applying for the loan. It then makes sense that the missing data indicates that the event has not occurred. Some modification of the variable will be necessary to properly utilize the information. This will be covered in the following section regarding variable modifications.

Furthermore, some of the variables contain similar information. For instance, number of installment accounts opened past 12 months, number of installment accounts opened past 24 months, number of installment accounts and number of currently active installment trades are all represented in different variables. There will be some reduction here to remove the redundancies introduced through the additional variables.

Borrower current financial state

Current financial state of the borrower is represented through many different variables. Among them are variables for the number of installment accounts, revolving credit accounts and number of bankcard accounts the borrower has. There are also variables for how many of them are currently active, how many are satisfactory and the maximum credit available for the various account types. These variables contain significant overlap

and there is strong correlation between some of the variables. Some are removed to reduce the redundancy.

Loan characteristics

These variables contain information about the specific loan. For instance what the loan amount is, when the loan was issued, how many installments will be paid, and the size of the monthly installments.

There are also variables concerning the amount of funding already received for each loan, whether a payment plan is in place and url for the loan application page. These, among others, are all variables that either provide information for the bidding process or variables that add no relevant information with regards to predicting defaults. They will be removed from the dataset. Notable variables that are kept are *installment*, *issue date*, *loan amount* and *loan purpose*.

Loan performance

Loan performance includes variables that give us information on how the loan is performing after being issued. This includes updates on the borrowers FICO score, payment history, total payments and whether the borrower has made any late payments. These variables provide information that would not have been available for an investor to peruse when deciding whether to invest in a given loan or not, so they cannot be included in a final model. All of these variables are removed before we start fitting models.

Secondary applicant

Secondary applicant variables are concerning the loans where there are two borrowers applying for a loan jointly. The variables are duplicates of the variables present for the main applicant, and some are presented as joint accounts. For instance, annual income is presented as joint annual income. In total there is 68,053 loans with a secondary applicant. The focus of this thesis is on loans with individual borrowers, so the loans with secondary applicants are removed from the dataset along with all the variables in this category.

Created variables

Some additional variables are created to extract as much significant information from the data as possible.

As previously mentioned, *Average FICO range* will be created as a replacement for the upper and lower bound of FICO rating originally provided.

It is calculated as the mean of the lower and upper bounds of the FICO rating.

$$\text{Average FICO range} = \frac{\text{FICO range high} + \text{FICO range low}}{2} \tag{2.1.1}$$

Another new variable is the *Monthly debt rate* variable. This variable is created to measure the amount of added debt burden the loan applied for will add to the borrowers monthly expenditure. It does not consider the already present economic obligations or savings the borrower might have from other sources, but it gives an indication on how heavy the additional burden will be for the borrower.

It is calculated using the following formula.

$$\text{Monthly debt rate} = \frac{\text{Installment}}{(\text{Annual income}/12)} \tag{2.1.2}$$

2.1.2 Variable modification

Some of the variables are unsuitable for use in the modeling phase in their initial state. In the following subsection the modifications will be explained.

Loan status

Table 2 shows the levels the variable *loan status* can take initially.

Table 2: Loan status categories

Loan status	Observations	Proportion
Charged Off	166,249	0.09
Fully Paid	674,649	0.36
Current	969,187	0.52
Default	219	0.00
In Grace Period	14,661	0.01
Late (16-30 days)	5,800	0.00
Late (31-120 days)	20,621	0.01
Issued	18,988	0.01

Loans are classified as defaults when they are 120+ days overdue. After 150 days they are charged off and there is no longer a reasonable expectation of further payments.

In this thesis the focus is on loans that have reached maturity, either through full repayment, or through being charged off or defaulted. This decision is made to be able to estimate the probability of default for the full lifetime of the loans. The levels: Current, In Grace Period, Late (16-30 days), Late (31-120) days and issued represent the loans that are still active. These loans will all be excluded from the dataset.

Two levels are kept. *Fully paid* is renamed *Non-default* and kept as is. *Charged off* and *Default* are merged into one level named *Default*. The variable is thus reduced to a binary variable where all loans fall into either the *Default* category or the *Non-default* category.

Home ownership

This variable is a factor with 6 levels: *mortgage*, *none*, *other*, *own*, *rent* and *any*.

Table 3: Home ownership categories

Home ownership	Observations	Proportion
Mortgage	922,758	0.49
None	49	0.00
Other	144	0.00
Own	206,736	0.11
Rent	740,175	0.40
Any	512	0.00

Table 3 shows that the majority of borrowers fall into the categories *mortgage*, *rent* and *own*. The interpretation of and difference between *None*, *Other* and *any* is difficult. Due to these categories containing so few observations, they are consolidated into one category named *Other*.

Continuous variables to factor

Several of the variables are stating how many months ago an event occurred. For instance, months since last delinquency, last record or most recent bank card account was opened. These variables contain a lot of `NA` entries. For these variables, `NA`-entries is taken to mean that the event has not occurred previously in the loan applicants record. The variables are listed below.

Table 4: Months since event variables

Variable name	Variable description
Mths since last delinq	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	Months since the last public record
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

Since these variables contain `NA`-entries and also entries that are 0, the variable needs to be changed to utilize the information. This is solved by converting the variable to an ordered factor. The cutoff points are set at yearly intervals, so that the first level contains all loans where the borrower has had the event occur within the last year. Second level is the year prior to that, and so on. Finally all NA observations are defined to be larger than the largest observation and this factor level is labeled *never* to indicate the event has never occurred.

Table 5: Levels of the converted factors

Factor levels after conversion					
$<$ 1 year	1-2 years	2-3 years	3-4 years	$>$ 4 years	Never

2.2 Exploratory data analysis

The variables kept after the initial reduction is listed in Table 34 in the Appendix. In this section exploratory data analysis is performed to see if there are any additional steps needed to improve the data before fitting the models. Descriptive statistics are produced for each variable, along with tests to check for significant differences between the defaulted loans and the non-defaulted loans. In addition, the variables are checked for outlying observations, and appropriate actions are taken with the results found. Finally tests for correlation and multicollinearity are done and some variables are removed due to high correlation.

2.2.1 Categorical variables

Table 35 in the Appendix shows the descriptive statistics for the categorical variables in the dataset. This table includes absolute counts of the different factor levels along with the proportion of loans within each group. There is also counts for the number of defaults and rate of default within each group. Furthermore, the table shows the ϕ coefficient for each subgroup, which is a measure of correlation between the variable and loan status. In addition, the χ^2 test-statistic and p-value for the χ^2 contingency table test performed on each subgroup is reported. Due to the large sample size the tests are very likely to report significant differences.

A contingency table is set up in the following way:

Table 6: Example contingency table

	TRUE	**FALSE**	
TRUE	a	b	e
FALSE	c	d	f
	g	h	n

Here $e = a + b$, $f = c + d$ and so on. n represents the total number of observations.

ϕ is calculated using the following formula

$$\phi = \frac{ad - bc}{\sqrt{efgh}} \tag{2.2.1}$$

and the χ^2-test is done by calculating the expected and observed observations for each combination of the two binary variables. The null hypothesis for the χ^2-test is that the

status of a loan is independent of the factor level it is tested against. Under the assumption that the null hypothesis is true, the estimated expected number of observations for a given combination of factors can be calculated by

$$\begin{aligned} \hat{e}_{ij} &= n \cdot \hat{p}_{i\cdot} \cdot \hat{p}_{\cdot j} = n \cdot \frac{n_{i\cdot}}{n} \cdot \frac{n_{\cdot}}{n} \\ &= \frac{n_{i\cdot} \cdot n_{\cdot j}}{n} = \frac{(i\text{th row total})(j\text{th column total})}{n} \end{aligned} \tag{2.2.2}$$

while the observed frequencies are readily found in the table. That leads to the χ^2 test-statistic value found by

$$\chi^2 = \sum_{\text{all cells}} \frac{(\text{observed} - \text{estimated expected})^2}{\text{estimated expected}} = \sum_{i=1}^{I} \sum_{j=1}^{J} \frac{(n_{ij} - \hat{e}_{ij})^2}{\hat{e}_{ij}} \tag{2.2.3}$$

Table 35 in the Appendix shows the results of the tests. Not all variables are as useful for predicting defaults. Many of the variables contain levels where there is little dependence between the factor level and *loan status*. No variable is completely independent from *loan status* however, so they are all kept in the dataset. The ϕ correlation coefficient indicates that the strongest correlation is found between loans belonging to grade A. Here the probability of default is lower than for the other grades. This, along with the observed proportions of defaults in the various grade levels indicates that LendingClub has a good model for evaluating the risk of lenders.

Table 7: Descriptive statistics for the levels of employment length

Emp. length	N	%	Defaults	%	Phi	χ^2	P-value
0	18,130	5.56	3,576	19.72	0.045	671.63	0.000, ***
< 1 year	25,621	7.86	3,647	14.23	0.008	18.36	0.000, ***
1 year	21,133	6.48	2,864	13.55	0.002	0.72	0.397,
2 years	29,435	9.03	3,938	13.38	0	0.01	0.921,
3 years	26,125	8.02	3,517	13.46	0.001	0.26	0.613,
4 years	18,979	5.82	2,537	13.37	0	0	0.978,
5 years	20,684	6.35	2,731	13.2	-0.001	0.44	0.506,
6 years	17,463	5.36	2,372	13.58	0.002	0.78	0.376,
7 years	18,083	5.55	2,389	13.21	-0.001	0.34	0.558,
8 years	16,146	4.95	2,174	13.46	0.001	0.16	0.692,
9 years	12,419	3.81	1,673	13.47	0.001	0.13	0.716,
10+ years	101,723	31.21	12,122	11.92	-0.029	265.33	0.000, ***

*N = Observations within group, Defaults = Number of defaults and proportion of defaults within the group , phi = correlation coefficient, χ = chi-square test critical value, *** = 0.001, ** = 0.01, * = 0.05 significance level.*

Table 7 describes the variable *employment length*. There are approximately 13 % defaults for all the levels between 1 years of employment up to and including 9 years of employment. For the borrowers that have been employed for less than one year, or that are currently unemployed, we see there is a higher rate of defaults. On the other end of the scale, where borrowers have ten or more years of employment, there is a lower rate of defaults.

Similar trends can be seen in the data for the *grade* and *subgrade* variables. The grades are set by lending club in an effort to signify the risk of each loan. This also holds true in the data, and can be seen in Figure 3 and 4. There is indeed a higher rate of default for loans of the higher grades, and the increase in defaults is fairly linear. For the subgrades within the G group, there are some discrepancies, however there are only 235 loans with grade G. For the subgrades G1 to G5 there are 108, 67, 39, 14 and 7 loans respectively, so the subset of loans here is very small.

Figure 3: Loan status by grade

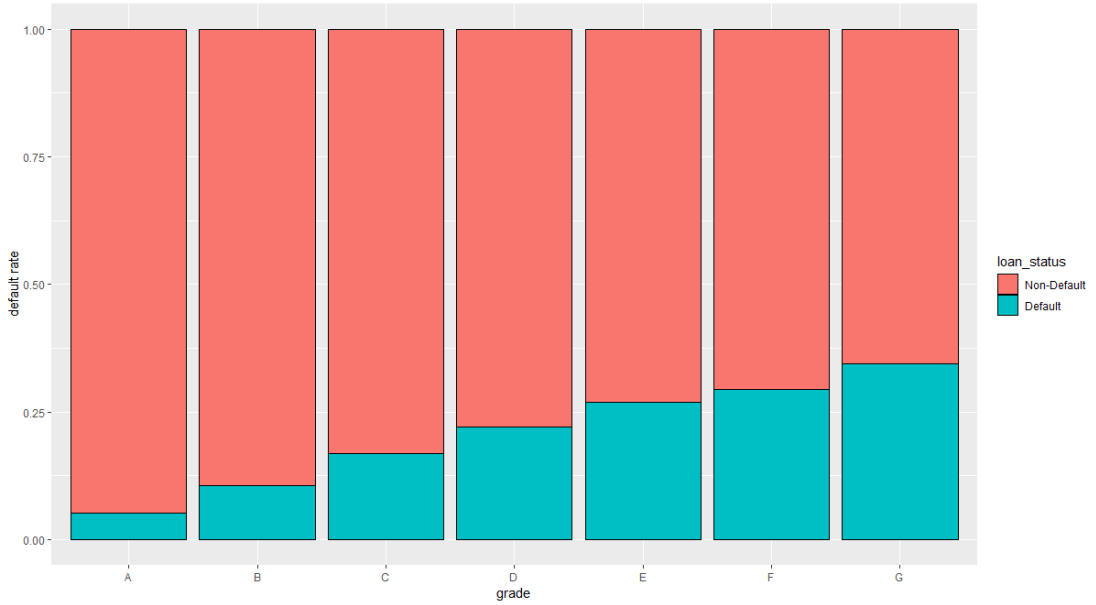
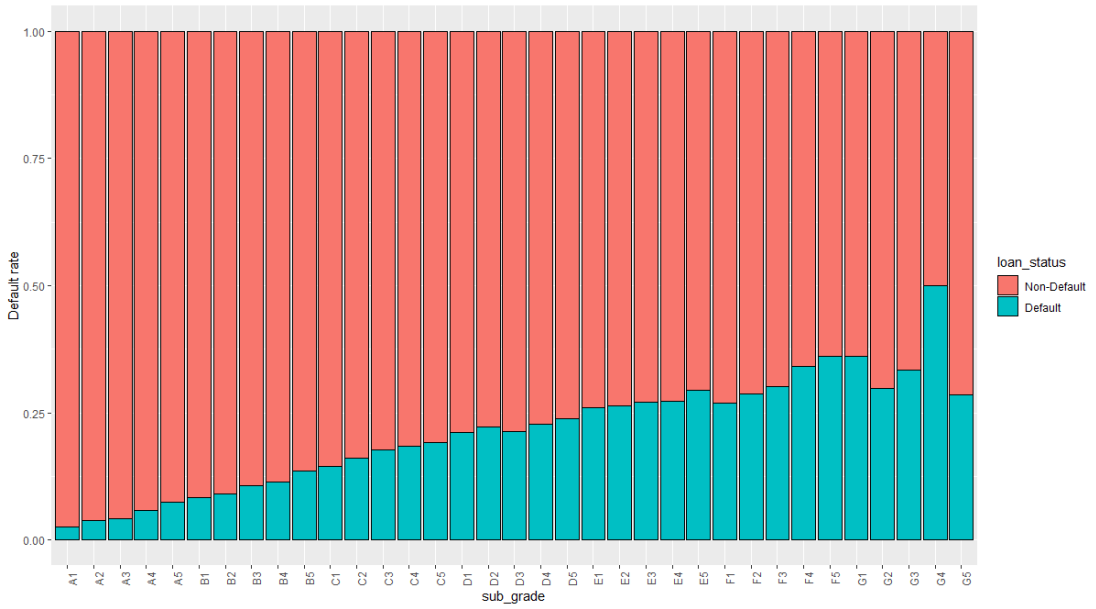


Figure 4: Loan status by subgrade



Dummy encoding of categorical variables

The categorical variables will be dummy encoded to allow them to be used in the boosting and random forests algorithms. One issue with this approach is that we lose the order of some of the variables. *Employment length* for instance has a given order to the observations, but this information will be lost when we dummy-code the factor levels. For the sake of comparison of the models however, we find that an equal dataset is preferred rather than using separate datasets for the different models. The transformation done to the categorical variables means that each separate factor level is changed to an individual variable.

2.2.2 Continuous variables

Table 36 shows the descriptive statistics for the continuous variables in the dataset split between the default loans and non-default loans. The statistics calculated are the mean, median and standard deviation for both default and non-default loans. Figure 30 and 31 shows histograms and quantile plots of the continuous variables. It is clear to see that none of the variables follow the Normal distribution closely. While the t-test is robust to deviations from normality when the sample size is large, we still opt for a non-parametric test to see if there is significant difference between the default and non-default loans. We use the Mann-Whitney test and the results are found in Table 36.

Outliers

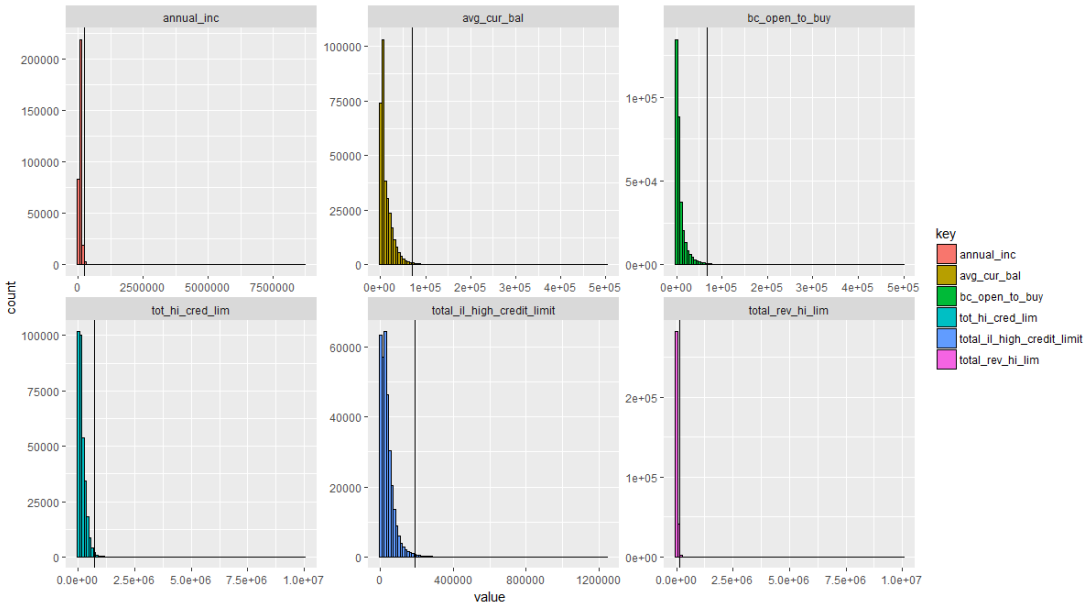
When inspecting the boxplots and histograms for all the continuous variables, found in Figure 29 and 30 respectively, it is clear that some of the variables contain outliers that might affect the fit of the models. In this thesis the models will be fit separately for the dataset containing all the observations, and also on a secondary dataset where some of the outlying observations have been removed. Both results will be reported, so that it is easy to see whether the outliers have any effect on the model fit. We justify this removal of outliers by the fact that the objective is to find a model that will optimize a portfolio of investments on the lending club platform. Therefore it is not unreasonable to, for instance remove any applications where the borrower reports an annual income of more than 1,000,000. The number of loans effected is relatively small, while the accuracy of the model might be improved by focusing on a narrower band of loans.

The histograms in Figure 30 indicates that some of the variables have values where all but a few of the loans are 0. For example the number of accounts the borrower is currently delinquent on, only 1474 loans are nonzero. These observations should not be considered outliers even though the boxplot and histogram might suggest so. Other variables that follow the above are variables concerned with collection and chargeoff done the past 12 months, and number of accounts of various types with delinquencies the past year.

There are however variables with outliers that warrant another look. *Annual income* contains heavy outliers, as does several of the variables measuring available credit and different account type balances. In Figure 5 we see the histogram for these variables. The

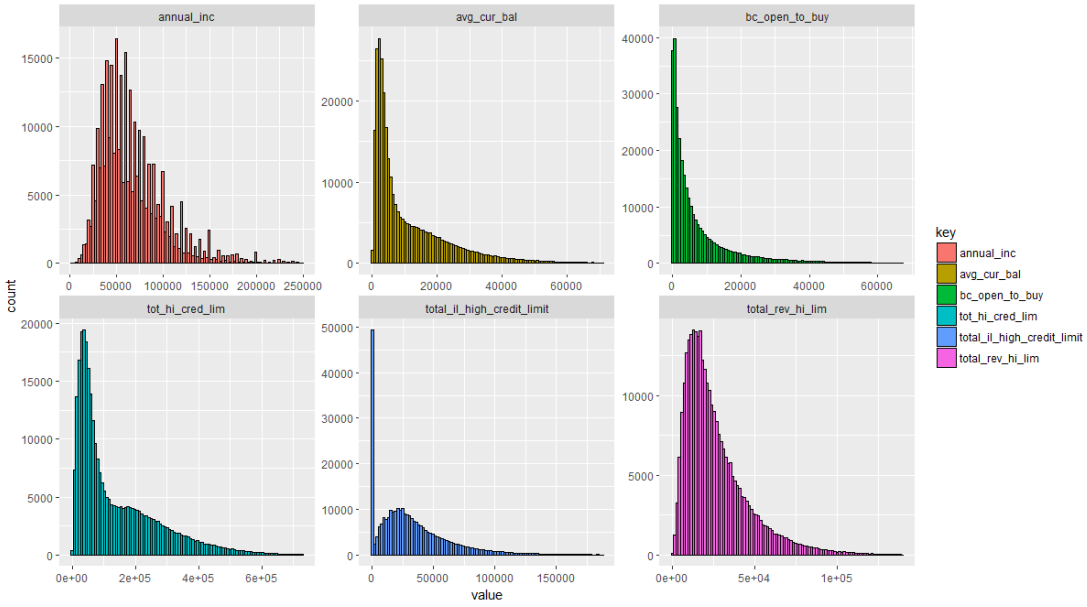
vertical line represents the 99'th quantile for each variable, while the greatest observation is at the far right of each histogram.

Figure 5: Histogram for variables with heavy outliers



In total the number of observations outside the 99'th quantile is 14,088 for these variables. Removal of these gives the distributions found in Figure 6.

Figure 6: Histogram with heavy outliers removed



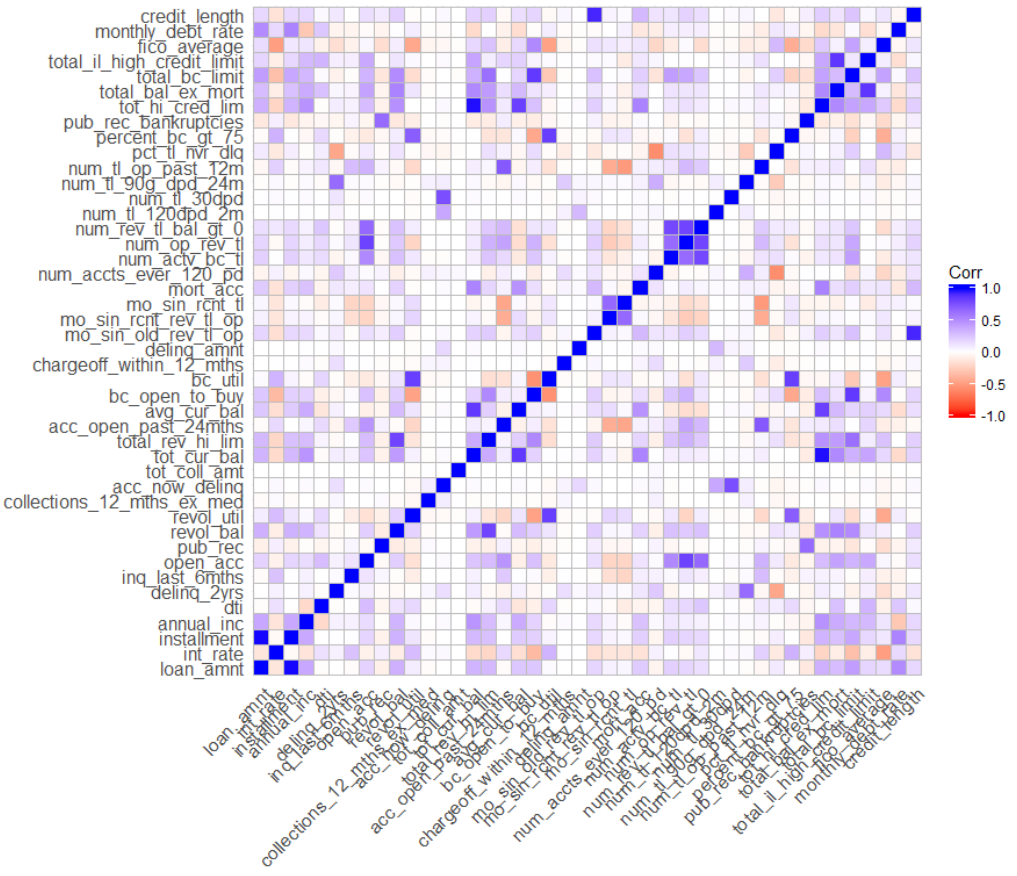
2.2.3 Initial variable selection

Before the dataset is separated into the training data and validation data, we do an initial variable selection. This can be done as long as it is not done using the response variable as a means to decide what variables to drop. The reason we should not use the response variable for selection is that we then introduce information from the entire dataset when we choose what variables to drop or not. Since we want the final validation data to be undisturbed by the model building process we should take care as to not introduce unnecessary bias to our model. Variable selection that we can do without introducing bias is to use methods which does not include any knowledge of what class the observations belong to. In this thesis the correlation between the variables will be investigated, and any highly correlated variables will be removed.

Correlation between the predictors

If some of the explanatory variables are highly correlated with each other, it is known as collinearity. If more than two variables are linear combinations of each other it is called multicollinearity. This might lead to unstable coefficient estimates and high standard errors when models are fit. Figure 7 shows the correlation between all the continuous variables in the dataset. There is very little negative correlation between the variables, but some of the variables are highly positively correlated.

Figure 7: Correlation matrix for continuous variables



Loan amount and *installment* size are two variables that understandably are interconnected, since the installment size is a function of the loan amount. Similar correlations are found in other variables, though not as strongly as for *loan amount* and *installment size*. Table 8 shows the variables that have a correlation lower than -0.5 or higher than 0.7.

Table 8: Correlation table

Correlation	Variable 1	Variable 2
0.994	loan amnt	installment
0.977	tot cur bal	tot hi cred lim
0.925	credit length	mo sin old rev tl op
0.857	total bal ex mort	total il high credit limit
0.854	avg cur bal	tot cur bal
0.854	bc open to buy	total bc limit
0.832	bc util	percent bc gt 75
0.829	bc util	revol util
0.817	tot hi cred lim	avg cur bal
0.811	num op rev tl	open acc
0.804	revol bal	total rev hi lim
0.803	num rev tl bal gt 0	num op rev tl
0.788	num rev tl bal gt 0	num actv bc tl
0.760	acc now delinq	num tl 30dpd
0.721	num tl op past 12m	acc open past 24mths
-0.583	num accts ever 120 pd	pct tl nvr dlq
-0.557	bc util	bc open to buy
-0.521	num tl op past 12m	mo sin rcnt tl

Only correlation below -0.5 and above 0.7 are included in the table

In addition to the correlation between variables, a test to check for multicollinearity is done. We calculate the Variance inflation factor (VIF), which is a measure of how much of the variance of each explanatory variable can be explained by the other variables.

The formula to calculate the VIF for each variable is:

$$VIF(\hat{\beta}_j) = \frac{1}{1 - R^2_{X_j|X_{-j}}} \tag{2.2.4}$$

where $R^2_{X_j|X_{-j}}$ is the R^2 from a regression of X_j onto all of the other explanatory variables.

The minimum value of VIF is 1 and that indicates that there is no multicollinearity for the given variable. Rules of thumb state that measures of VIF larger than 5 or 10 indicate problematic levels of multicollinearity.

The VIF for the variables that are greater than 5 are presented in Table 9. The table on the left shows the VIF before removal of any variables, while the table on the right are the VIF after *loan amount*, *total current balance*, *total balance excluding mortgage* and *total bankcard limit* has been removed.

The VIF score indicates that the correlation found between variables are likely to be what is causing most of collinearity issues, and that there is not as much multicollinearity in the dataset.

Table 9: VIF scores > 5 before and after removal.

Variable	VIF
loan amnt	312.516
installment	307.018
tot cur bal	47.191
tot hi cred lim	44.82
total bal ex mort	21.959
total il high credit limit	17.371
total bc limit	10.574
bc open to buy	9.935
revol bal	9.87
num op rev tl	7.649
mo sin old rev tl op	7.221
total rev hi lim	6.481
bc util	6.191
avg cur bal	6.012
num rev tl bal gt 0	5.929
open acc	5.533

Initial VIF scores

Variable	VIF
num op rev tl	7.52
tot hi cred lim	7.29
mo sin old rev tl op	7.205
bc util	6.034
num rev tl bal gt 0	5.617
open acc	5.338
avg cur bal	5.103

VIF-score after variables have been removed.

2.2.4 Splitting of the dataset

To accurately be able to predict the probability of default for a loan it is necessary to have the information of how the loan performed over its lifetime. To attain this, all loans that are still "live" are removed from the dataset. To have a larger set of loans to work with, only the 36-month loans are considered. Given these restrictions, the remaining time period in the dataset is loans issued in August 2012 at the earliest. For a 36-month loan to be included in the final dataset, it must have been issued in March 2015 at the latest.

A further consideration is also made with regards to the issue date of the loans. To be able to make good inference it is necessary to have a validation set that consists of loans issued in a separate time period than the loans in the training set. An intertemporal split is made to separate the training data from the validation data. This means I take all loans issued prior to a given date and use for training data, while all loans issued after this date is used as the validation set. [Joy and Tollefson, 1975].

The objective of splitting the data in this manner is to build models that are less likely to be influenced by conjunctional changes and other seasonal effects. Ideally the time period available to extract loans from would be larger. This would allow the validation set to contain loans issued from a longer period. For the final dataset, all loans issued in the period August 2012 to October 2014 are used as training data, while loans issued between November 2014 and March 2015 are kept as a validation set.

The same split is made for both the full dataset, and for the dataset where outlying observations have been removed. Table 10 shows the number of observations and percentage of observations in each of the sets.

Table 10: Intertemporal split for the datasets

	Training set	%	**Holdout set**	%
Full set	249,587	0.77	76,354	0.23
No Outliers	239,354	0.77	72,499	0.23
Period		Aug12 - Oct14		Nov14 - Mar15

Number of loans in each dataset.

Cross-validation

Cross-validation can be used to estimate the test error associated with a certain statistical learning method in order to evaluate its performance, or select the appropriate level of flexibility [James et al., 2013]. The test error is the average prediction error when we use the fitted model to predict on unseen data. The idea is that since the fitting is done without using the unseen data, the results when predicting on this subset should be as accurate as the model would be when introduced in the real world on new data.

When we fit a model we would like to keep the training dataset as large as possible, and cross-validation is a method that lets us reuse the training data. The way it works is by randomly dividing the training data into k folds of roughly equal size. One of the folds is treated as test data while the model is fit on the remaining $k-1$ folds, and the test error is calculated for the predictions made on the selected fold. This process is repeated k times, with each fold held as test data once. Finally the k test error measures are averaged to give us the estimated test error.

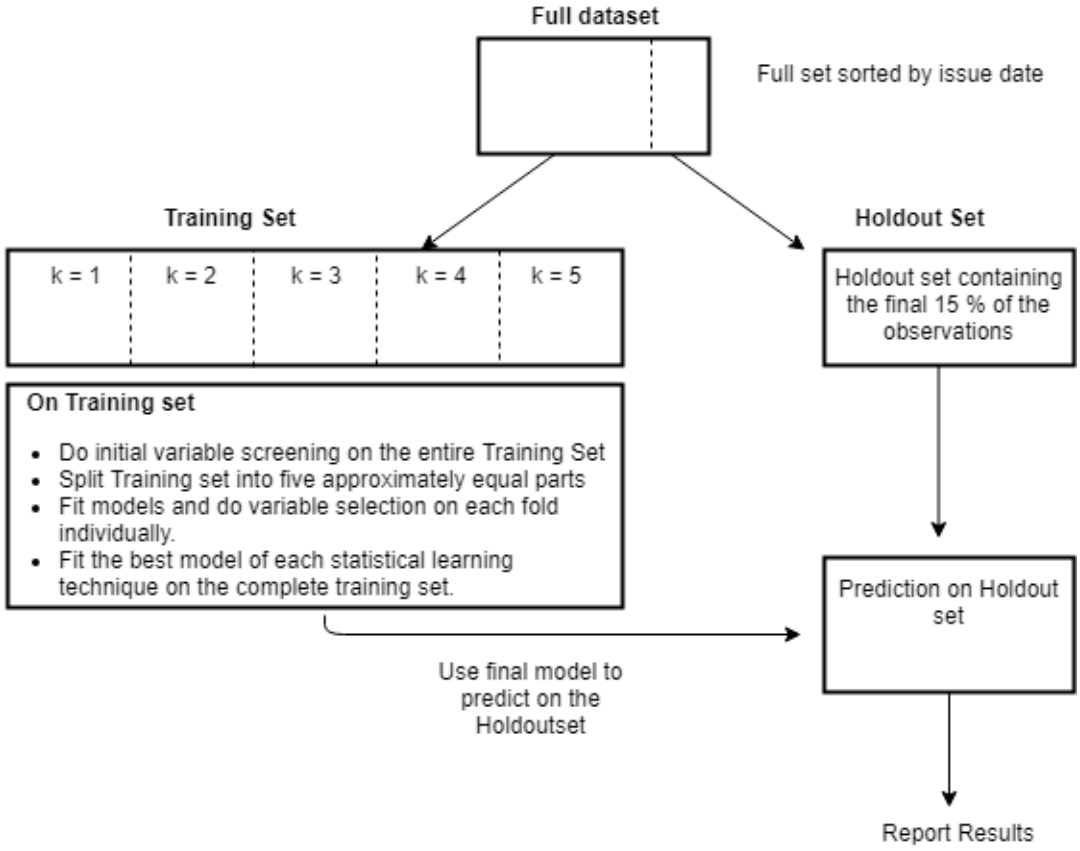
In essence, there is a bias-variance trade-off when selecting the number of folds to use. The more folds in the model, the lower the bias and higher the variance. When the number of folds is high, the training set is bound to overlap between the folds. This leads to correlated outputs from each model, which in turn will lead to high variance in the estimated test error measure when the outputs are averaged. On the one extreme end of this, if we set $k = n$ we get what is called Leave-one-out cross-validation. This involves fitting models using each observation as the hold-out data once. Since we use almost all of the training data for each fit of the model, this setup leads to approximately unbiased estimates of the test errors. It is however very computationally intensive, since we will have to fit the statistical learning model n times. The variance of the resulting test error measure will also be high, due to the highly correlated outputs from each of the k models. On the other end of the scale, with $k = 2$, we get what is called the validation set approach. This entails splitting the data into a training set and a validation set. We fit the model using the training data, and make a prediction on the validation data to estimate the test error. One issue with this approach is that it has very high bias, since the effect of selecting a new training set can have dramatic impacts on the estimated test error. Another issue with the validation set approach is that it tends to overestimate the test error rate since its fit using only a subset of the dataset.

In this thesis, 5-fold cross-validation is used. It has been empirically shown that $k = 5$ or $k = 10$ leads to test error rate estimates that suffer neither from excessively high bias

nor from very high variance [James et al., 2013]. In addition to this, the computational strain is kept within a reasonable level when using 5 folds.

Figure 8 shows the process implemented when the different models are fit.

Figure 8: The model fitting process



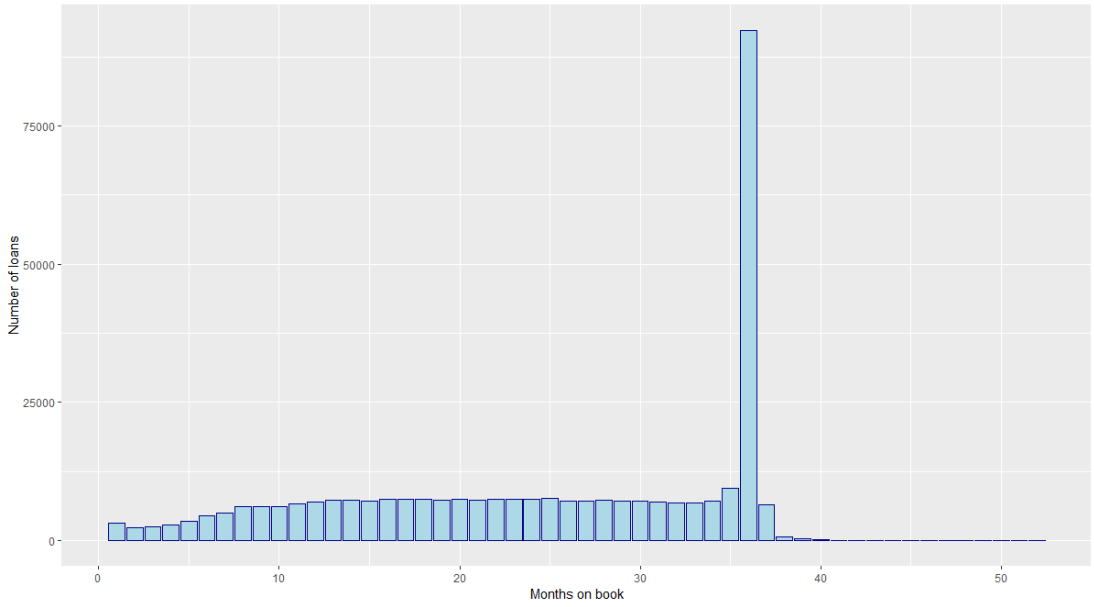
2.3 Payment history dataset

The payment history dataset which can be downloaded from [LendingClub], contains records for each individual payment made on all the loans in the database. The variables and their description can be seen in Table 37 in the Appendix. There are several variables that are duplicates of the information available in the loan data described in Section 2.1. There are also some variables that are of little interest for our purposes.

In total the payment history dataset contains 40 variables and 33,416,337 entries. The size of the dataset makes it difficult to handle, so we will extract only the variables of interest and the observations concerning the 325,941 loans that are used in the further analysis. After this initial reduction we are left with a dataset containing 8,128,494 entries, concerning the 325,941 loans we will analyze further.

Figure 9 shows the lifetime of the loans in the final dataset. All the loans are supposed to have 36 entries, but it is clear that many of the loans either default or are paid in full prior to the full lifetime. However, the vast majority of loans run for the 36 intended months.

Figure 9: Lifetime of the loans in the final dataset.



Months on book indicates the number of entries for a loan in the payment history dataset.

Table 11 shows the payment history of one loan for selected variables. Of interest to us is the variable *Received amount* which shows the actual payments made by the borrower. Using this information we are able to calculate the *net present value* and *internal rate of return* for each loan. These variables are described in detail in Section 4.2.

Table 11: Example payment history

Loan ID	Balance	Principal paid	Interest paid	Due amount	Received amount	Loan status
361542	28,000.00	701.34	163.10	864.44	864.44	Current
-	27,298.66	705.43	159.01	864.44	864.44	Current
-	26,593.23	709.53	154.91	864.44	864.44	Current
-	25,883.70	713.67	150.77	864.44	864.44	Current
-	25,170.03	717.82	146.62	864.44	864.44	Current
-	24,452.21	722.01	142.43	864.44	864.44	Current
-	23,730.20	726.21	138.23	864.44	864.44	Current
-	23,003.99	730.44	134.00	864.44	864.44	Current
-	22,273.55	734.70	129.74	864.44	864.44	Current
-	21,538.85	738.98	125.46	864.44	864.44	Current
-	20,799.88	743.28	121.16	864.44	864.44	Current
-	20,056.60	747.61	116.83	864.44	864.44	Current
-	19,308.99	751.97	112.47	864.44	864.44	Current
-	18,557.02	756.35	108.09	864.44	864.44	Current
-	17,800.67	760.75	103.69	864.44	864.44	Current
-	17,039.92	765.18	99.26	864.44	864.44	Current
-	16,274.74	769.64	94.80	864.44	864.44	Current
-	15,505.10	15,505.10	90.32	864.44	15,595.42	Fully Paid

Example of the payment history for a single loan.

2.4 Summary of the data preparation Section

In summary, what has been done in the data preparation and initial variable screening steps is as follows.

For the loan data:

1. Variable selection:
    - Variables containing information obtained post issuance of the loan are removed from the dataset, along with variables for secondary applicants and other variables with little explanatory power.
2. Dropping of observations:
    - Observations prior to introduction of variables in September 2012 are removed from the dataset.
    - Observations containing missing information are removed from the dataset.
    - Observations outside the chosen intertemporal periods are dropped to retain a subset containing only loans that have run their full lifetime.
3. Variable modification:
    - Variables are modified to be easier to use with statistical learning techniques, and new variables are created based on some of the other variables in the dataset.
4. Variable screening and initial variable selection based on the exploratory data analysis.
5. Creation of a separate subset where outlying observations have been removed as mentioned in 2.2.2
6. Splitting of the dataset into training set and holdout set.

There are now two datasets; one with outliers removed and one where the outliers are still present in the dataset. Going forward in the thesis, the presentation will be with regards to the dataset not excluding the outliers, but the process is completed for both sets. Results will be presented for both datasets when comparing the models.

From the payment history dataset we extract the payments for each individual loan. This is used to calculate the internal rate of return for each loan which will be used as a profitability measure for the different models.

Chapter 3

Models

In this section the theoretical framework for the various models implemented in this thesis is presented. First, binary regression is introduced along with the Generalized Linear Model framework, which is extended to show the implementation of the Elastic net model. Following that the concept of decision trees as a means for making binary decisions is introduced. It is further extended to include bagging and boosting models, implemented through random forests and extreme gradient boosting techniques. Finally, the K-nearest-neighbors method is presented as an alternative technique.

The Section on binary regression and the generalized linear modeling framework is based largely on Dobson and Barnett [2008] and Fahrmeir et al. [2013].

3.1 Binary regression and GLM

The main objective of this thesis is to predict whether a borrower defaults on their loan or not based on the explanatory variables available in the dataset. The dependent variable in this case is *loan status*, which is a binary variable.

A binary variable is defined:

$$Y = \begin{cases} 1 \text{ if the outcome is a success} \\ 0 \text{ if the outcome is a failure} \end{cases} \tag{3.1.1}$$

The aim of a regression analysis with binary responses $y \in \{0, 1\}$ is to model the expected value E(y), or rather the probability $P(y = 1) = P(y = 1|x_1, \cdots, x_p) = \pi$

In the case of a linear regression model: given p predictors, $(x_1, x_2, \ldots, x_p)$, the response y is predicted by

$$E(Y_i) = \mu_i = \mathbf{x}_i^T \boldsymbol{\beta}; \qquad Y_i \sim N(\mu_i, \sigma^2) \tag{3.1.2}$$

Y_i represents the independent random variables, $\mathbf{x}_i^T$ represents the i'th row of the design matrix $\mathbf{X}$ and β represents the parameters for each explanatory variable.

Since the dependent variable in our case is binary and not quantitative however, it is not appropriate to use linear regression. There are several reasons for this.

- The right hand side of the equation is not binary
- Since y_i has a Bernoulli distribution with $\pi_i = \beta_0 + \beta_1 x_1 + \cdots + \beta_p x_k$, it follows that $Var(y_i) = \pi_i(1 - \pi_i)$ depends on the values of the covariates and the parameters $\beta_0, \cdots, \beta_p$ and thus cannot have the same constant variance σ^2 for all observations i.
- The linear model allows values $\pi_i < 0$ or $\pi_i > 1$ for $\pi_i = P(y_i = 1)$ which makes little sense for probabilities.

We can avoid these issues by combining the probability π_i with the linear predictor η_i through a relation of the form

$$\pi_i = h(\eta_i) = h(\beta_0 + \beta_1 x_{i1} + \cdots + \beta_p x_{ip}) \tag{3.1.3}$$

where h is a strictly monotonically increasing cumulative distribution function on the real line. This ensures that $h(\eta) \in [0, 1]$ and that we can always express the above equation in the form $\eta_i = g(\pi_i)$ with the inverse function $g = h^{-1}$.

We are thus able to model the relationship between variables where the response variables are not normally distributed. In the generalized linear model framework, h is called the response function, while $g = h^{-1}$ is the link-function.

3.1.1 Generalized linear model

[Nelder and Wedderburn, 1972] demonstrated generalized linear models as a way of unifying various statistical models.

The model is defined in terms of a set of independent random variables $Y_1, \cdots, Y_n$, each with a distribution from the exponential family and the following properties:

1. Each Y_i has the canonical form and its distribution depends on a single parameter θ_i, such that

$$f(y_i; \theta_i = \exp[y_i b_i(\theta_i + c(\theta_i) + d(y_i)] \tag{3.1.4}$$

2. The distributions of all the Y_i's are of the same form, for instance, all being Normal distributed or all being Binomial distributed, which allows us to drop the subscript for b, c and d.

Due to the points above, the joint probability density function of $Y_1, \cdots, Y_N$ is

$$\begin{aligned} f(y_1, \cdots, y_N; \theta_1, \cdots, \theta_N) &= \prod_{i=1}^{N} \exp\left[y_i b(\theta_i) + c(\theta_i) + d(y_i)\right] \\ &= \exp\left[\sum_{i=1}^{N} y_i b(\theta_i) + \sum_{i=1}^{N} c(\theta_i) + \sum_{i=1}^{N} d(y_i)\right] \end{aligned} \tag{3.1.5}$$

Supposing that $E(Y_i) = \mu_i$, where μ_i is some function of θ_i, then there will exist for a generalized linear model, a transformation of μ_i such that $g(\mu_i) = \mathbf{x}_i^T \boldsymbol{\beta}$.

Here, g is a monotone, differentiable function called the link function, while $\mathbf{x}_i$ is a $p \times 1$ vector of explanatory variables, and $\boldsymbol{\beta}$ is the $p \times 1$ vector of parameters.

A generalized linear model thus consists of three components:

1. Response variables $Y_1, \cdots, Y_N$ that share the same exponential family distribution.
2. A set of parameters $\boldsymbol{\beta}$ and explanatory variables $\boldsymbol{X} = \mathbf{x}_i^T$
3. A monotone link function g such that $g(\mu_i) = \mathbf{x}_i^T \boldsymbol{\beta}$, where $\mu_i = E(Y_i)$

Following is a look at the useful properties of the exponential family, and some examples of the transformation of the normal and binomial distribution to the canonical form used in the generalized linear modelling framework. In addition, some of the most commonly used link functions are discussed for the case with binary response variables, as that is the most relevant for the work in this thesis.

The exponential family

For a response variable to be usable in the GLM setting, it has to belong to one of the distributions in the exponential family. If we have a single random variable Y belonging to a probability distribution depending on a single parameter θ. The distribution belongs to the exponential family if it can be written on the form

$$f(y;\theta) = s(y)t(\theta)\exp^{a(y)b(\theta)} \tag{3.1.6}$$

where a, b, s and t are known functions.

This can be rewritten in the form

$$f(y;\theta) = \exp[a(y)b(\theta) + c(\theta) + d(y)] \tag{3.1.7}$$

where $s(y) = \exp[d(y)], t(\theta) = \exp[c(\theta)]$.

The distribution is in canonical form if $a(y) = y$. $b(\theta)$ is sometimes referred to as the natural parameter. Any additional parameters in the model are considered nuisance parameters forming parts of the functions a, b, c and d, and they are treated as though they are known. As mentioned previously, many of the well-known distributions belong to the exponential family, including the Poisson, Normal and Binomial distributions.

Transformation of the normal distribution to canonical form

The normal distribution has the probability density function

$$f(y;\mu) = \frac{1}{(2\pi\sigma^2)^{\frac{1}{2}}} \exp\Big[-\frac{1}{2\sigma^2}(y-\mu)^2\Big] \tag{3.1.8}$$

where μ is the parameter of interest and σ^2 is considered a nuisance parameter. This can be rewritten on the canonical form as

$$f(y;\mu) = \exp\Big[-\frac{y^2}{2\sigma^2} + \frac{y\mu}{\sigma^2} - \frac{\mu^2}{2\sigma^2} - \frac{1}{2}log\ (2\pi\sigma^2)\Big], \tag{3.1.9}$$

where $a(y) = y$, $b(\mu) = \frac{\mu}{\sigma^2}$, $c(\mu) = -\frac{\mu^2}{2\sigma^2} - \frac{1}{2}log(2\pi\sigma^2)$ and $d(y) = -\frac{y^2}{2\sigma^2}$

Transformation of the binomial distribution to canonical form

The binomial distribution has the probability density function

$$f(y;\pi) = \binom{n}{y}\pi^y(1-\pi)^{n-y} \tag{3.1.10}$$

where y takes the values $0, 1, 2, ..., n$ and $\binom{n}{y} = \frac{n!}{y!(n-y)!}$
This is denoted by $Y \sim Bin(n, \pi)$

π is the parameter of interest, and n is assumed to be known. To get the probability function on the canonical form we rewrite it as

$$f(y,\pi) = \exp\left[y\ log\pi - y\ log(1-\pi) + n\ log(1-\pi) + log\binom{n}{y}\right] \tag{3.1.11}$$

by taking the exponential and logarithm of each term in the equation. We can then isolate y so that we end up with

$$f(y,\pi) = \exp\left[y\ log\Big(\frac{\pi}{1-\pi}\Big) + n\ log(1-\pi) + log\binom{n}{y}\right] \tag{3.1.12}$$

where $a(y) = y, b(\theta) = log\Big(\frac{\pi}{1-\pi}\Big), c(\theta) = nlog(1-\pi), d(y) = \binom{n}{y}$

In addition to the above distributions, several others also belong to the exponential distribution, and many can be written on the canonical form.

Link functions

Logistic regression and the logit-function
In logistic regression we model the probability that the response belongs to a particular category rather than modeling the response directly. To achieve this we must model $p(x)$ using a function that gives outputs between 0 and 1 for all values of x. Many functions fit this criteria, but in logistic regression the logistic function is used.

The logistic response function is

$$\pi = h(\eta) = \frac{\exp\eta}{1+\exp\eta} \tag{3.1.13}$$

which gives us the logit link function

$$g(\pi) = log\big(\frac{\pi}{1-\pi}\big) = \eta = \beta_0 + \beta_1 x_1 + \cdots + \beta_k x_k \tag{3.1.14}$$

This gives us a linear model for the logarithmic odds, $log\big(\frac{\pi}{1-\pi}\big)$

Transformation with the exponential function yields,

$$\frac{\pi}{1-\pi} = \exp(\beta_0) + \exp(\beta_1 x_1) + \cdots + \exp(\beta_k x_k) \tag{3.1.15}$$

which implies that the effect of the explanatory variables affects the odds in an exponential-multiplicative form.

Probit function
For the probit link-function, the response function h is defined by the standard normal cumulative distribution function Φ, i.e.,

$$\pi = \frac{1}{\sigma\sqrt{2\pi}} \int_{-\infty}^{\infty} \exp\Big[-\frac{1}{2}\Big(\frac{s-\mu}{\sigma}\Big)^2\Big] ds = \Phi\Big(\frac{x-\mu}{\sigma}\Big) \tag{3.1.16}$$

As the link function is the inverse of the response function, the probit link-function is:

$$\Phi^{-1}(\pi) = \beta_0 + \beta_1 x_1 + \cdots + \beta_k x_k, \tag{3.1.17}$$

the inverse cumulative Normal probability function. Probit models are most useful when there are natural interpretations of the model. In biological sciences, the model $x = \mu$ is called the median lethal dose model, because it corresponds to the dose that can be expected to kill half of a population.

Complementary log-log function
The complementary log-log model uses the extreme minimum-value cumulative distribution function as the response,

$$h(\eta) = 1 - \exp(-\exp(\eta)), \tag{3.1.18}$$

giving us the link function

$$g(\pi) = log(-log(1-\pi)) = \beta_0 + \beta_1 x_1 + \cdots + \beta_k x_k \tag{3.1.19}$$

This model is similar to the logit and probit models when the values of π is near 0.5, but differs when π approaches 1 or 0.

3.1.2 Elastic net

Historical background

The elastic net regularization technique is first introduced by Zou and Hastie [2005]. They present a regularization technique that enables us to implement a combination of ridge regression and lasso regression. Ridge regression is first shown in Hoerl and Kennard [1970] and it is a regularization technique that shrinks the estimated coefficients of a regression model towards zero. Lasso regression, first proposed in Tibshirani [1996] is another regularization technique, but it allows the variables to shrink all the way to zero. As such the lasso regression technique also functions as a variable selection method. Both of these regularization techniques are used to reduce the variance of a model, at the cost of introducing some bias. The elastic net attempts to keep the best parts of both regularization types.

Theoretical implementation

As previously mentioned, using the linear regression model and given p predictors, $(x1, x2, \ldots, xp)$, the response y is predicted by

$$\hat{y} = \hat{\beta}_0 + x_1\hat{\beta}_1 + \cdots + x_p\hat{\beta}_p \tag{3.1.20}$$

In the case of Ordinary least square (OLS) we obtain estimates for the vector of coefficients $\hat{\beta} = (\hat{\beta}_0, \cdots, \hat{\beta}_p)$,

by minimizing the residual sum of squares.

$$RSS = \sum_{i=1}^{n} \left(y_i - \beta_0 - \sum_{j=1}^{p} \beta_j x_{ij} \right)^2 \tag{3.1.21}$$

To evaluate the quality of a model, we typically look at two aspects:

- Can the model produce accurate predictions?
- Is the model easily interpretable?

To improve the interpretability of a model, a parsimonious model is often preferred. The fewer predictors there are, the more easily one can explain the relationship between the covariates and the response. In addition, including many variables in a model increases the amount of variability, which can lead to a less generalizable model. OLS often does poorly in both predictions and interpretation. As a means to improve the model, for both prediction and interpretability, different regularization techniques have been introduced. One such technique is called ridge regression, first introduced by Hoerl and Kennard [1970], which minimizes the residual sum of squares subject to a bound on the L2-norm of the coefficients .

$$\begin{aligned}\hat{\beta}^{ridge} &= \underset{\beta}{\operatorname{argmin}} \left(\sum_{i=1}^{n} \left(y_i - \beta_0 - \sum_{j=1}^{p} \beta_j x_{ij}\right)^2 + \lambda \sum_{j=1}^{p} \beta_j^2 \right) \\ &= RSS + \lambda \sum_{j=1}^{p} \beta_j^2\end{aligned} \tag{3.1.22}$$

It adds a penalty term to the OLS regression, so that the coefficients are penalized if they are large. The output is a shrunken version of the model, with coefficients being reduced towards zero and towards each other, without ever being completely removed from the model.

λ acts as a complexity parameter here and controls the amount of shrinkage. Larger λ leads to greater amounts of shrinkage. If we have many correlated variables in a linear regression model, the coefficients can become poorly determined and exhibit high variance. For example, one large positive coefficient in one variable can offset a large negative coefficient in another correlated variable. Ridge regression alleviates this problem by adding a size constraint on the coefficients. Ridge regression thus improves the prediction performance of the model by way of a bias-variance tradeoff. Variance is reduced, which helps improve the predictions, but we increase bias by restricting the model.

Another shrinkage method we can use to improve the model is a technique called Lasso regression which was first proposed by Tibshirani [1996]. Like ridge regression it is a penalized least squares method, but instead of L2-regularization, it imposes a L1-penalty on the regression coefficients. This regularization does both shrinkage and variable selection, and it can be expressed as

$$\begin{aligned}\hat{\beta}^{lasso} &= \underset{\beta}{\operatorname{argmin}} \left(\frac{1}{2} \sum_{i=1} n\Big(y_i - \beta_0 - \sum_{j=1} p\beta_j x_{ij}\Big)^2 + \lambda \sum_{j=1}^{p} |\beta_j| \right) \\ &= RSS + \lambda \sum_{j=1}^{p} |\beta_j|\end{aligned} \tag{3.1.23}$$

Because of the nature of the lasso constraint some of the coefficients will be shrunk to be exactly zero, resulting in some of the variables being removed from the model. The lasso regression technique has some limitations however, especially in the case of high-dimensional data where there are more predictors than observations. In this case the lasso regression can select at most n variables. Another issue is if there exists a group of highly correlated variables, it tends to keep only one of the correlated variables, while removing all the others. This might be an issue if the variables are all important.

As a compromise between lasso and ridge regression, the elastic net penalty is introduced. The formula to calculate it is

$$\hat{\beta}^{elastic} = \underset{\beta}{\operatorname{argmin}} \left(\frac{1}{2} \sum_{i=1} n\Big(y_i - \beta_0 - \sum_{j=1} p\beta_j x_{ij}\Big)^2 + \lambda \sum_{j=1}^{p} \Big(\alpha|\beta_j| + (1-\alpha)\beta_j^2\Big)\right)$$
$$= RSS + \lambda \sum_{j=1}^{p} \Big(\alpha|\beta_j| + (1-\alpha)\beta_j^2\Big) \tag{3.1.24}$$

We see from the equation above that the elastic net penalty includes an additional parameter α, which takes values between 0 and 1. This parameter controls the ratio of L1/L2 penalization. $\alpha = 0$ leads to pure ridge regression, while $\alpha = 1$ leads to pure lasso regression. Elastic net has the benefit of being able to use a mixture of the two. It retains the variable selection property of the lasso regression while also shrinking together the coefficients of the predictors like with ridge regression.

Technical implementation

In my implementation of the elastic net regularization i use a statistical learning framework called caret in R. With it, we can set up a glmnet elastic net regression, that enables us to search over a grid for the optimal values of α and λ. The hyperparameter search space and the process of tuning the parameters is found in Section 4.1.1. The hyperparameters we are tuning for are α and λ. α is the hyperparameter that defines the elastic net mixing parameter. It ranges from 0 to 1 where 0 is pure ridge regression, and 1 is pure lasso regression.λ is a measure of how strict the penalization factor should be. A higher value leads to stricter penalization, which in turn leads to fewer variables passing the lasso regularization, and stronger shrinkage towards zero for each variable from the ridge regression. The available parameter space for each hyperparameter is found in Table 12. Additional hyperparameters exist, including parameters to set restrictions on how many variables should be kept in the final model, but these are not included in our implementation.

Table 12: Tunable parameters for elastic net

Parameter name	Parameter Space
Alpha	0 to 1
Lambda	0 to Inf

The hyperparameters and parameter space

Advantages and disadvantages

Elastic net regularization works well for many datasets. It can work well on low-dimensional data as well as high-dimensional data. If there are more predictors than observations then lasso regression can at most select n variables to retain in the final model. The elastic net regularization avoids this issue. It allows us to combine the two favorable aspects of L1 and L2 regularization. It has the variable selection element

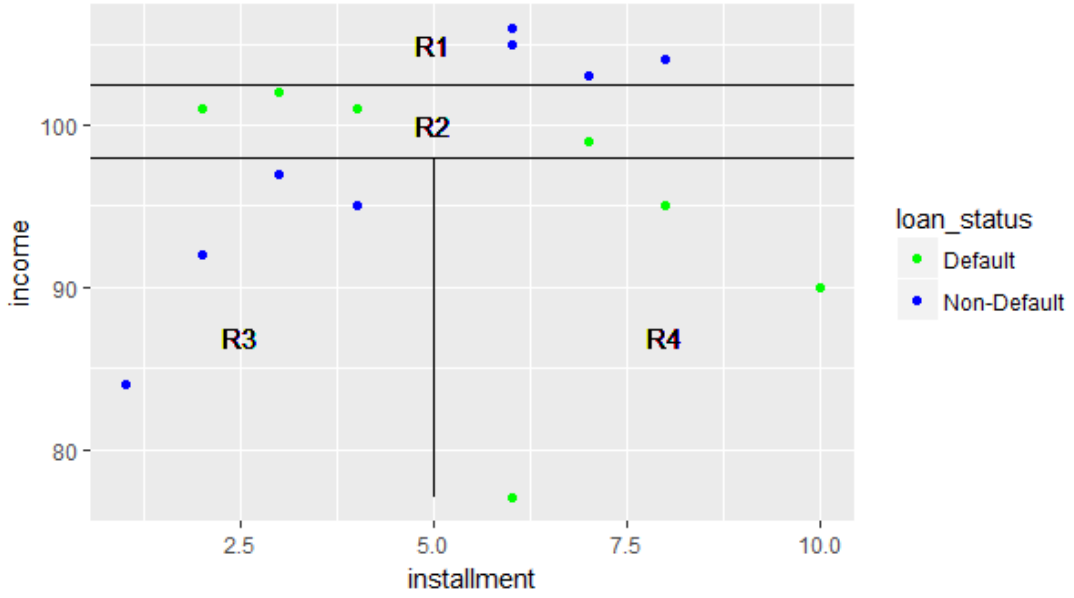
of pure lasso regression, where it allows for a more parsimonious model than ordinary least squares regression does, while also keeping the shrinkage of the coefficients enabled by the ridge regression. A disadvantage with the technique is the rather complicated parameter search space that is required to find an optimum value. Implementing elastic net rather than a pure ridge- or lasso-regression gives us an additional hyperparameter to tune for, which increases the computational cost of fitting a model.

3.2 Decision tree models

3.2.1 The basic decision tree

Tree-based methods are conceptually very simple and easy to understand, and can be a useful tool for both regression and classification tasks. In this introduction a constructed dataset is used to show how a decision tree is implemented. It contains only 16 observations for loan status, income and installment size. It does not reflect the real dataset used in the actual analysis in any way, but it is useful for explaining the process of building a decision tree. The following section is written using [James et al., 2013] as reference. A decision tree is built using a set of splitting rules to segment the predictor space into distinct and non-overlapping regions as shown in Figure 10.

Figure 10: How the decision boundaries split the dataset into regions

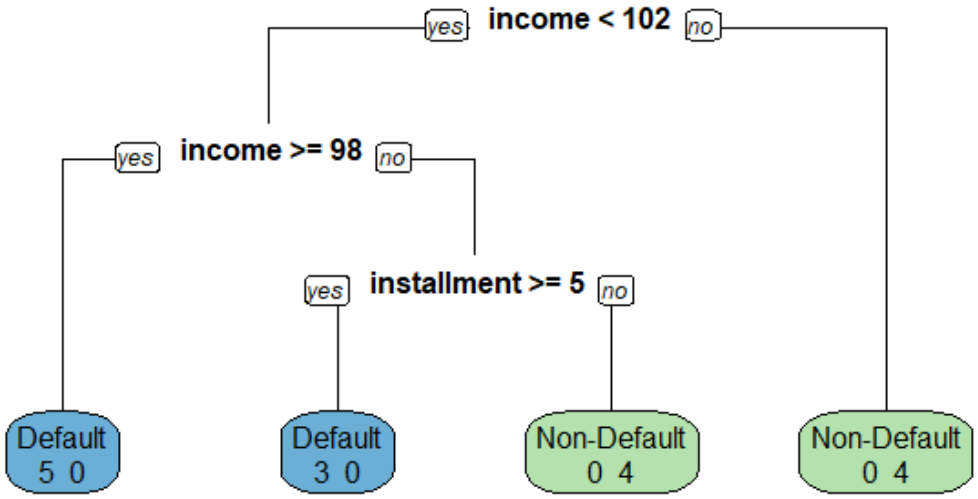


The rules can be listed in the following way:

- Income $<$ 102 which leads to the regions $R_1, (R_2 + R_3 + R_4)$
- Income $>=$ 98 which leads to the regions $R_1, R_2, (R_3 + R_4)$
- Installment $>=$ 5 which leads to the regions R_1, R_2, R_3, R_4

As shown in Figure 10 the explanatory variables are *income* and *installment size*, and the predictor space is divided into four distinct regions, R_1, R_2, R_3, R_4. Figure 11 shows the resulting decision tree and the order of the splits.

Figure 11: Example of a decision tree using simplified data



The terminal nodes at the bottom signifies the split between the two groups. Leftmost node holds 5 default and 0 non-defaults, and so on.

First the question "is *income* less than 102" is posed, and we split the data into two subsets. R_1 contains all loans that have income greater than 102, while the remaining area containing R_2, R_3, R_4 holds all the loans that has income less than 102. Then another split is made on the income variable for values greater or equal to 98, before, finally a split on which loans have installment size greater or equal to 5 is made. In the case of this constructed dataset we obtain perfect separation between the two classes loan status can take; Default and Non-Default. All of the terminal nodes, meaning the final levels of each branch, contain only one classification for loan status. This is not usually the case in a real-world data set however, so there needs to be additional rules implemented to build the tree. This includes rules for how deep we should allow the tree to be, and what criteria to use when making the decision on which variable and where to make the split.

Splitting criteria and pruning

The objective of the segmentation of the predictor space is to obtain as homogeneous regions as possible. In the regression case we want to find a region where the output prediction will be as close to the actual outcome as possible. For classification we want to find a region where the training data all indicate that the outcome belongs to the same class.

For regression trees we do this by finding the regions $R_1, \cdots, R_J$ that minimize the Residual Sum of Squares given by

$$\sum_{j=1}^{J} \sum_{i \in R_j} (y_i - \hat{y}_{R_j})^2 \tag{3.2.1}$$

Here j represents the different regions, and y_{R_j} is the mean value of all the observations that fall in region j. If we are building a classification tree on the other hand, we can't use RSS as the criterion for making the binary splits. An alternative is to use the classification error rate. This is the fraction of erroneously classified observations.

$$E = 1 - \max_k(\hat{p}_{mk}),$$

where $\hat{p}_{mk}$ represents the proportion of training observations in the m'th region that are from class k. Other splitting criteria are the Gini index and cross-entropy.

$$\text{Gini Index } = \sum_{k=1}^{K} \hat{p}_{mk}(1 - \hat{p}_{mk}), \tag{3.2.2}$$

and

$$\text{Cross-entropy } = -\sum_{k=1}^{K} \hat{p}_{mk} \log \hat{p}_{mk} \tag{3.2.3}$$

The Gini index is a measure of the total variance across the K classes. It is considered a measure of node purity, where a small value indicates that a node mostly contains observations from a single class. Cross-entropy is another measure of purity and it yields similar numerical results to the Gini index. Both of these measures are more sensitive to node purity than classification error rate, and as such they are more often used in practice.

When we build a decision tree it is not computationally feasible to calculate each possible partition of the predictor space into J separate regions. When the number of observations and predictors increase, there will be an exponential increase in the possible partitions. To combat this, we instead use a top-down greedy approach, known as recursive binary splitting. The term greedy indicates that the we always go for the best possible split at the current step of the process, so that we always choose the partitioning that provides the largest instant reduction in the error measure. As a consequence, we will not consider partitions where there might be improved results further down the line. To perform binary splitting, we have to select the predictor X_j and cut point s such that the regions $\{X|X_j < s\}$ and$\{X|X_j \geq\}$ leads to the greatest possible reduction in error.

For any j, and s we define the pair of half-planes

$$R_1(j,s) = \{X1X_j < s\} \text{ and } R_2(j,s) = \{X|X_j \geq s\}, \tag{3.2.4}$$

and seek the value of j and s to minimize

$$\sum_{i:x_i \in R_1(j,s)} (y_i - \hat{y}_{R_1})^2 + \sum_{i:x_i \in R_2(j,s)} (y_i - \hat{y}_{R_2})^2 \tag{3.2.5}$$

or similarly for a classification tree using Gini-index

$$\sum_{i:x_i \in R_1(j,s)} \hat{p}_{1k}(1-\hat{p}_{1k}) + \sum_{i:x_i \in R_2(j,s)} \hat{p}_{2k}(1-\hat{p}_{2k}) \tag{3.2.6}$$

We consider all the predictors and all possible values for the cut point s for each of the predictors, and then choose the combination that leads to the greatest reduction in error measure. After this is found, the process is repeated for each new region, until a stopping criterion is reached. This stopping criterion might be based on the degree of purity in the node, or that the node contains less than a given number of observations. Once the regions $R_1, \cdots, R_J$ have been found, we predict the response for a given test observation using the mean of the training observations for regression, or a majority vote for the classification case.

This exhaustive partitioning will often lead to a large decision tree that will produce good predictions on the training set, but poor test set performance. This is due to the complexity of the decision tree. It is overfitting the training data, and could be improved if the tree was reduced in size. This technique is called pruning and involves growing a tree to its full size before "cutting" off branches of the tree, effectively reducing its size. This process leads to a reduction in variance at the cost of some additional bias. A problem with this approach is to decide which of the possible subtrees are worth looking into. It would be too cumbersome to check every possibility, so a way to select a smaller set of subtrees for consideration is a method called cost complexity pruning. Here we consider a sequence of trees indexed by a nonnegative tuning parameter α. For each value of α there corresponds a subtree $T \subset T_0$ such that

$$\sum_{m=1}^{T} \sum_{i:x_i \in R_m} (y_i - \hat{y}_{R_m})^2 + \alpha|T| \tag{3.2.7}$$

is as small as possible. $|T|$ indicates the number of terminal nodes of the tree T, R_m is the region of the prediction space corresponding to the m'th terminal node, and y_{R_m} is the predicted response for observations in R_m.
α controls the trade-off between complexity and its fit to the training data. It balances the trade-off between variance and bias. With a value of $\alpha = 0$ the subtree will be the same as the full tree, and there will be no reduction. With increasing values of α the complexity of the tree will be reduced, along with the number of terminal nodes and branches.

3.2.2 Boosting

One of the modeling approaches I use to predict probability of default is a gradient boosting machine. I use xgboost, short for extreme gradient boosting, which is a widely used algorithm for both classification and regression tasks in statistical learning. It has proven to be computationally efficient while still being accurate. It is often used in the winning models in machine learning competitions over a wide variety of tasks. Of 29 winning solutions published at Kaggle's (Kaggle.com) blog during 2015, 17 used xgboost. Eight used solely xgboost, while the others combined it with neural nets or other algorithms in larger ensembles [Chen and Guestrin, 2016].

Historical background

Boosting originates from the question posed by Kearns and Valiant [1994] about whether a set of weak learners could be converted to a strong classifier. A weak learner is a model which can produce a hypothesis that performs only slightly better than random guessing. [Schapire, 1990]. Building an accurate classifier is difficult, but it is not as hard to create decision rules that are only moderately accurate [Schapire, 2003]. This is the concept that is applied in boosting. We use an algorithm to find many of these moderately accurate classifiers and combine the results from these in the final model.

Each iteration of the weak learner takes the results from the previous iteration into account, usually by weighing the observations that were wrongly classified more heavily. In this way, each consecutive weak learner will improve its classification on the observations that have proven harder to correctly classify. We don't really change the weak learner itself between iterations, but we manipulate the underlying training data by iteratively re-weighting the observations [Mayr et al., 2014]. When the desired number of learners have been produced we combine the prediction results from each learner, usually by taking a weighted majority vote, into the final prediction for the model. Freund and Schapire presented Adaboost in the 1990s, and it was the first adaptive boosting algorithm as it automatically adjusts its parameters to the data based on the actual performance in the current iteration. [Mayr et al., 2014]. They went on to formulate Adaboost as gradient descent with a special loss function which has later been generalized to handle a variety of loss functions. Several alternative boosting algorithms have later been developed, including xgboost. The main variation between the boosting algorithms is the way in which they weigh the training data.

Theoretical implementation

Boosting algorithms fit ensemble models of the kind

$$f(x) = \sum_{i=0}^{N} f_i(x) = f_0(x) + \sum_{i=1}^{N} \lambda f_i(x) \tag{3.2.8}$$

Where $f_0(x)$ is the initial model, λ is the shrinkage parameter, often referred to as the learning rate, and $f_i(x)$ is the successive iterations of the model. The product $\lambda f_i(x)$ denotes the "step" at iteration i.

To show how the gradient descent is implemented we can show how it works in the simple least-squares regression case. Given $(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)$, We want to build a model $F(x)$ that can predict values of Y, so that $F(x) = y$ We start with an initial model, for instance

$$F(x) = \sum_{i=1}^{n} \frac{(yi)}{n} \tag{3.2.9}$$

Using a loss function $L(y, F(x))$, in this case the mean squared error function $(y-F(x))^2$, we calculate the negative gradients:

$$-g(x_i) = -\frac{\partial L(y_i, F(x_i))}{\partial F(x_i)} \tag{3.2.10}$$

We fit another regression tree h to the negative gradients so that we get the final model $F(x) = F_0(x) + \lambda h$, where λ determines the learning rate.

In the case of least squares regression, this loss function and negative gradients is the same as the residuals, but for other loss functions it will differ. We then fit the next regression tree by using the negative gradients instead of the original training set.

Algorithmically the procedure for a boosting regression tree is as follows:

1. Set $\hat{f}(x) = 0$ and $r_i = y_i$ for all i in the training set.
2. For $i = 1, 2, ..., N$, repeat:
   (a) Fit a tree $\hat{f}_i$ to the training data (X, y_i)
   (b) Update $\hat{f}$ by adding in a shrunken version of the new tree:
   $$\hat{f}(x) < -\hat{f}(x) + \lambda \hat{f}_i(x)$$
   (c) Update the residuals:
   $$r_i < -r_i + \lambda \hat{f}^i(x_i)$$
3. Output the boosted model:
   $$\hat{f}(x) = \sum_{i=1}^{N} \lambda \hat{f}^i(x)$$

xgboost is a gradient boosting algorithm, which builds the model iteratively, similar to other boosting algorithms, and allows optimization of an arbitrary differentiable loss function. Freund and Mason [1999] described gradient boosting algorithms as iterative functional gradient descent algorithms that optimize a cost function over a function space by iteratively choosing a function that points in the negative gradient direction. When the negative gradient has been found and thus the direction in which to move to reduce the loss, the length of the step in this direction still needs to be evaluated. For gradient boosting machines, the normal approach is to use a line search to determine the optimal distance to move. For xgboost on the other hand, the gradient is calculated in a way that allows us to get both the direction and the distance of the step in one computation. This reduces the time it takes to calculate each iteration of the model, but it adds a restriction in the sense that the loss function needs to be twice differentiable.

Technical implementation

For the implementation done in R, the following hyperparameters listed in Table 13 are selected and searched for. Additional choices are made, such as deciding between a linear model fit or a tree method, but these hyperparameters are excluded from the following section.

Table 13: Tunable parameters for extreme gradient boosting

Parameter name	Par space
Eta	0 to 1
Gamma	0 to Inf
Max depth	1 to Inf
Min child weight	0 to Inf
Subsample	0 to 1
Colsample bytree	0 to 1
Nrounds	1 to Inf

The parameters that can be tuned and the possible range each parameter can take.

The *eta* hyperparameter controls the learning rate of the model. Higher values of eta lead to a model that learns slower, and takes less information from each iteration of the tree. It is used to control against overfitting. Another measure useful for controlling overfitting is the *gamma* hyperparameter. It is the minimum loss reduction required to make an additional partition on a leaf node of the tree. Larger values of gamma lead to more conservative and less complex models.

The *max depth* hyperparameter controls the number of splits that can be done in each of the trees built. The root node is considered to be level 0, and each following split along the same branch increases the depth of the tree. A deep tree is more complex and susceptible to overfitting, while a short tree is more stable but less accurate.

Minimum child weight is yet another measure to constrain the complexity of the model. If the leaf node after a partitioning is beneath this threshold, the model will not attempt further splits along this branch. Larger values lead to more conservative models.

For every iteration of the model fitting process a random subset of the samples is picked to build a tree. The size of the sample is controlled by the hyperparameter *subsample*. Setting this value to 0.5 means only half of the datapoints in the training set will be used for each tree. This prevents overfitting. Along a similar vein , *colsample by tree* controls the number of variables available for each split. This hyperparameter will sometimes exclude the more powerful predictors, which lets the algorithm extract information from the weaker variables in a dataset. This will improve the overall performance of the model.

Finally, *nrounds* is a hyperparameter that controls how many iterations of trees that are built in the model.

Advantages and disadvantages

One advantage with xgboost is that it scales extraordinary well. It can be run in parallel and on separate machines, which allows it to handle huge datasets and billions of observations. It generalizes well and obtains good results on a variety of datasets and optimization tasks. A negative property is the difficulty of explaining the output

of the model. It can be hard to explain how the conclusions are drawn due to the large number of weak learners included in the final model.

3.2.3 Random forests

Random forests is a tree-based ensemble learning method which performs well on binary classification tasks. Similar to the boosting algorithm shown in Section 3.2.2 it trains several trees and aggregates the results. However, instead of changing the weighing of the training data between trees, random forests uses bootstrap sampling to generate new training data from the same distribution for each tree. In random forests a random subset of the variables available is used for each tree. This randomization is used each time a split in the tree is considered. The reason this is done is to reduce the correlation between the different trees produced by the model. If there exists one very strong predictor in the data set, along with several other somewhat strong predictors, then it could very easily be the case that the first few splits chosen for most of the trees would be splits done on the same variables. Consequently, the trees would all look similar, even though the trees are built using different bootstrap samples. In such an event, the predictions from the trees would be highly correlated, and we would not get the desired reduction in variance.

A normal subset of variables to choose from for classification at each split is $m = \sqrt{p}$ where p is the total number of variables in the training set. By using a small number of variables for each split, we will have many splits where the strongest variable is not available, so the other predictors will have a better chance to be chosen for the split. This process decorrelates the trees and makes the average of the resulting trees less variable and therefore, more reliable. This builds on the concept of weak learners mentioned previously, since we are often building decision trees excluding the most important variables. The predictive accuracy of these trees will likely not be very good, but the results can be aggregated to form a stronger learner.

Historical background

Tin Kam Ho introduced the method of random decision forests in 1995 as a way to overcome the fundamental limitation on the complexity of tree classifiers. [Ho, 1995] If trees are grown too complex they will overfit the training data, which will lead to low bias and high variance. This in turn leads to poor generalization on unseen data. Ho showed that producing several trees with random restrictions on the variable selection for each tree could lead to improved accuracy while avoiding overfitting. An extension of the algorithm was later proposed by Breiman where he describes random forests as it is used today. [Breiman, 2001a]. He describes building a forest of uncorrelated trees using randomized variables for each split, while also combining this with bootstrap sampling. We thus get a random subset of variables and observations for each tree, which greatly reduces the correlation between trees. This leads to models that are slightly worse in bias, but have reduced variance compared to the original random decision forests proposed by Ho.

Theoretical implementation

Given a training set of size (Nxp), where N is the number of observations, and p is the number of variables, the random forests algorithm can be implemented algorithmically in the following way:

1. For $b = 1$ to B:

   (a) Draw a bootstrap sample Z^b of size N from the training data.

   (b) Grow a random forest tree T_b to the bootstrapped data by recursively repeating the following steps for each terminal node of the tree, until the minimum node size n_{min} is reached.

      i. Select m variables at random from the p available variables.

      ii. Pick the best variable/split-point among the m.

      iii. Split the node into two daughter nodes.

2. Output the ensemble of trees $\{T_b\}_{1-B}$

To make predictions at a new point x:

Regression: $\hat{f}_{rf}^B(x) = \frac{1}{B}\sum_{b=1}^{B} T_b(x)$
Classification: Let $\hat{C}_{rf}^B(x) =$ majority vote $\{\hat{C}_b(x)\}_1^B$

To find the optimal split-point and variable in step b-ii above we use a measure of node impurity to grade the different variables and split points. In a node m, representing a region R_m with N_m observations, let $\hat{p}mk$ be the proportion of class k in node m.

$$\hat{p}_{mk} = \frac{1}{N_m}\sum_{x_i \in R_m} I(y_i = k) \tag{3.2.11}$$

Three common measures of node purity are:

Classification error:

$$\frac{1}{N_m}\sum_{i \in R_m} I(y_i \neq k(m)) = 1 - \hat{p}_{mk}(m) \tag{3.2.12}$$

Gini index:

$$\sum_{k \neq k'} \hat{p}_{mk}\hat{p}'_{mk} = \sum_{k=1}^{K} \hat{p}_{mk}(1 - \hat{p}_{mk}) \tag{3.2.13}$$

Cross-entropy or deviance:

$$-\sum_{k=1}^{K}\hat{p}_{mk}\log\,\hat{p}_{mk} \tag{3.2.14}$$

Technical implementation

The implementation of the random forests algorithm in this thesis is done using the Ranger package in R. It has been shown to be less taxing computationally and it also has short training time when compared to other implementations.

The random forests algorithm involves several hyperparameters controlling the structure of each tree, the structure of the forest, and the level of randomness in the trees and forest. in Table 14 the parameters tuned for in this thesis is listed.

Table 14: Tunable parameters for random forests

Parameter name	Parameter Space
Num.trees	1 to Inf
Mtry	1 to Inf
Replace	TRUE or FALSE
Sample.fraction	1 to Inf
Min.node.size	1 to Inf

Random forests hyperparameters and parameter space.

Num trees controls the number of trees that will be grown for the forest. It should be a sufficient number of trees so that every observation is sampled at least a few times.

The *mtry* hyperparameter controls the number of randomly drawn candidate variables that are available for selection at each split when growing a tree. As mentioned above, the default setting is to use $\sqrt{p}$, but the optimal value depends on the underlying dataset. In general, lower values of mtry lead to more different, less correlated trees, which can lead to better stability when aggregating. Having a low value of mtry can also lead to better exploitation of variables with moderate effect on the dependent variable, since the more powerful independent variables are more likely to be excluded when the number of candidate variables is low. On the other hand, low values of mtry leads to worse performing trees, since they are built on suboptimal variables. This defines the trade-off between stability and accuracy that has to be made when selecting the size of mtry.

With the *replace* hyperparameter we get to decide whether observations are sampled with replacement or not. The *sample fraction* hyperparameter sets the fraction of the dataset that will be randomly sampled for each tree.

Another complexity constraint is the *min node size* hyperparameter. It sets the minimum required size for a terminal node. If this is set to be a large number, then shorter and smaller trees are grown. This will reduce the complexity of the trees, and can help prevent overfitting. It will also improve the speed of growing the trees.

Advantages and disadvantages

Random forests is a very flexible method that can handle a variety of input data. It is able to produce fairly good results, even without tuning of the hyperparameters. Similar to xgboost, with random forests we can calculate a variable importance measure which lets us know what variables are best able to separate the target classes. The random forests technique is also resistant to overfitting, granted the number of trees in the model is sufficiently high. A disadvantage of the model is that it requires a lot of computational power to run, and the more trees you have in the model the higher the cost.

3.3 Alternative approaches

Several alternative learning techniques exist that could be used in a binary classification setting. As mentioned in the literature review, discriminant analysis, neural networks and support vector machines among others have been implemented with success for credit scoring. Another alternative method is the k nearest neighbor algorithm, which will be explored further in this following Section.

3.3.1 K-nearest neighbors

Historical background

Fix and Hodges Jr [1951] are the first to formulate a rule of the nearest neighbor type. They investigated a rule which assigns to an unclassified point, the class most heavily represented among its k nearest neighbors. The basic idea is that the proportion of the classified points which falls in a stated (small) neighborhood around the unknown observation may be used to estimate the probability that the unknown point belongs to a given class. They discuss the problem of defining the optimal size of the region around an unknown observation to use. Instead of defining a region of a fixed size, they propose choosing a number k, and take in the neighborhood of the unknown point a region containing a total of k points of known observations.

In an often-cited paper, Cover and Hart [1967] does an extensive study on the properties of the nearest neighbor rule. They find that the probability of error for a nearest neighbor rule has a lower bound in the Bayes probability of error and an upper bound that is twice the Bayes probability of error.

Building on the previous research, Dudani [1976] introduces a weighting function to define a distance-weighted k-nearest-neighbor (KNN) rule. They give the k nearest neighbors a weight to signify the closeness of the known observation to the unknown observation. Among other methods for distance-weighting they use the inverse of the distance as the weight for each of the k neighbors. The distance-weighted k-nearest-neighbor rule then assigns the unknown observation to the class for which the weights sum to the greatest value. They found that using distance-weighted k-nearest-neighbor lead to reduced probability of misclassification compared to the simple majority vote traditionally implemented.

Theoretical implementation

Given a positive integer k, and a test observation x_0, the KNN classifier first identifies the k points in the training data that are the shortest distance away from x_0, represented by $\mathcal{N}_0$. The distance measure usually applied is the Euclidian distance, defined as

$$d_i = ||x_i - x_0|| \tag{3.3.1}$$

It then estimates the conditional probability for class j as the fraction of points in $\mathcal{N}_0$ whose response values equal j:

$$Pr(Y = j|X = x_0) = \frac{1}{K} \sum_{i \in \mathcal{N}_0} I(y_i = j) \tag{3.3.2}$$

Using the inverse of the distance as weights for each of the k neighbors, we get

$$w_i = \frac{1}{d_i}, d_i \neq 0 \tag{3.3.3}$$

It then follows that for a distance-weighted KNN classifier that the probability of Y belonging to a class j is given by

$$Pr(Y = j|X = x_0) = \frac{1}{K} \sum_{i \in N_0} I(y_i = j) \cdot w_i \tag{3.3.4}$$

Finally, KNN classifies the test observation x_0 to the class with the largest probability. Despite the fact that it is a very simple approach, KNN can often produce classifiers that work surprisingly well.

The hyperparameter k is very important for the KNN algorithm. A value of k that is too low will lead to an extremely flexible model, with a decision boundary that will be very closely fit to the training data. This gives the model high variance, and it is likely that the model overfit the data. A k value too high on the other hand will lead to an inflexible model that generalizes too much. The decision boundary will be too rough, and will not follow the data close enough for it to make accurate predictions on the test data. The hyperparameter k thus controls the bias-variance trade-off in the knn model.

Technical implementation

To implement a KNN model we use the package fastknn in R, which provides the fastest (to our knowledge) implementation of knn available. With this implementation there are a few hyperparameters that can be tuned for optimal performance. Table 15 summarizes the hyperparameters and the parameter space for each of them.

k is the number of neighboring observations that are considered for each classification of a test observation. If k is set to 1, then we will only use the observation closest to the test observation to determine the predicted classification of the test observation. The possible values of k is 1 to n where n is the total number of observations in the training set.

Table 15: Tunable parameters for k nearest neighbor

Parameter name	Par space
K	0 to n
Normalization	Null, std, minmax, maxabs, robust
Method	Vote or dist

The tuneable hyperparameters and parameter space.

Method determines the technique used for calculating the prediction for the test observation. The possible values this hyperparameter can take are vote and dist. *Vote* implements the traditional majority vote method, where we simply count how many of the k nearest neighbors are from each class and classify the test observation as the same class as the majority of the neighbors. The *dist method* can be thought of as a weighted-voting rule. It weights the influence of each of the neighbors differently. The probability of the test observation belonging to a class is computed from the inverse of the nearest neighbor distances. This lets the neighbors closer to the observation play a more important role in predicting the classification outcome.

The final hyperparameter we can tune is the *normalization technique*. This is recommended when the variables are measured in different units. Since we have a lot of different units in our dataset, we will likely benefit from using a normalization technique. Table 16 shows the different options for normalization, and the way they are implemented in practice.

Table 16: KNN - normalization techniques

Normalization technique	Description
Null	No normalization implemented. Values are kept as they are in the original dataset.
Std	Variables are standardized by removing the mean and scaling to the unit variance.
Minmax	Variables are transformed by scaling each one between 0 and 1.
Maxabs	Variables are transformed by scaling each one by its maximum absolute value.
Robust	Variables are transformed by removing the median and scaling each variable by its interquartile range.

Normalization techniques available in the fastknn package.

Advantages and disadvantages

An advantage with the KNN modeling approach is that it is a non-parametric method that makes almost no assumptions about the dataset. It is highly flexible which allows for use in a variety of cases. It is an intuitive model that is simple to implement, while still providing good results in many cases. One disadvantage is that KNN is sensitive to class-outliers. Observations that don't fit the general consensus can have a large impact on the predicted outcome of classifications. Another potential issue is that KNN does not

do any form of variable selection, and will not make any judgement on which variables are more important than others. As a consequence of this it is very sensitive to irrelevant variables. The KNN model is computationally a very expensive method. Since it is a memorization-based technique, it needs to store all the data in the training set. It is also expensive in the sense that it can take a long time to calculate all the distances required to make the predictions. In addition to this the model will become slower and slower the more data is included in the training set. These computational issues are further exacerbated by the fact that the time spent by the model is all in the testing phase, not the training phase. This means that unlike other modelling approaches, each new test set we make predictions on requires the full "training" time to run.

Chapter 4

Results and model comparison

In this Section the results from the various modeling approaches are presented. First the hyperparameter tuning process and results are discussed for each model. Then an introduction to the accuracy metrics calculated is given, before the results from the individual models is presented. After that a comparison is done between the models to see which has the best overall performance.

4.1 Hyperparameter tuning

As mentioned in Chapter 3, each of the statistical learning techniques implemented have a set of hyperparameters that can be tuned to optimize the performance of the model. We make use of cross-validation to find the optimal settings. We make sure to use the same partitions for the Cross-validation for each model we tune hyperparameters for. This ensures that we get a fair comparison between the various models.

In addition to the resampling strategy, we can also set the search strategy for the hyperparameter tuning. In this thesis the strategies random search, grid search, or a combination of the two is implemented. The random search sets the hyperparameters to random values within a given range, and performs tuning for a set number of random combinations. The grid search splits the parameter space into uniformly distributed partitions, and tunes the parameters for each combination of points.

The strategies used and the extent of the hyperparameter tuning varies between the different model implementations due to the fact that the computational effort required to train the models vary to a great extent. Some models require tuning of more parameters, which in turn leads to an exponential increase in possible permutations. For the elastic net model, only two parameters with reasonably small search spaces are tuned for. This allows the use of a grid-search, giving an exhaustive look at the optimal parameter settings. For xgboost and random forests on the other hand there are more parameters to consider, so the initial parameter tuning is done by a random search. This tuning is then followed by a narrow grid search for a few chosen parameters in the parameter space near the previously found values.

As the Cross-validation is set to do five folds, each parameter setting is trained on the different training sets, and tested on the remaining folds. The average is taken of the test error estimates for each parameter setting, and the best model is selected

and trained on the complete training set. This model is then used to make predictions on the observations in the holdout set. Following is the results of the hyperparameter optimization for each of the learning techniques implemented in this thesis.

4.1.1 Elastic net

For the elastic net model there are only two hyperparameters that we tune to find the optimal settings, namely α and λ. These control the degree of LASSO-regularization versus ridge-regularization and the penalty term respectively. To select the optimal α we use a cross-validation method and test five different values to optimize the AUC (description of the AUC and other metrics is found in Section 4.2). Table 17 lists the results from the tuning process.

Table 17: Hyperparameter settings for elastic net

Parameter name	Search space	Result	Var. count	Test set AUC
Alpha = 0				
Lambda.min	0.0060 to 58.7000	0.0060	151	0.68232
Lambda.1se	0.0060 to 58.7000	0.0870	151	0.68000
Alpha = 0.25				
Lambda.min	0.0007 to 0.2351	0.0007	118	**0.68289**
Lambda.1se	0.0007 to 0.2351	0.0082	45	0.68024
Alpha = 0.5				
Lambda.min	0.0004 to 0.1175	0.0004	118	0.68246
Lambda.1se	0.0004 to 0.1175	0.0041	40	**0.68028**
Alpha = 0.75				
Lambda.min	0.0002 to 0.0783	0.0002	116	0.68247
Lambda.1se	0.0002 to 0.0783	0.0027	39	0.68024
Alpha = 1				
Lambda.min	0.0002 to 0.0587	0.0002	116	0.68247
Lambda.1se	0.0002 to 0.0587	0.0020	39	0.68022

The hyperparameters and final tuning results for various levels of α

Since the model with $\alpha = 0.25$ and $\lambda = 0.0007$ obtains the highest test set AUC, we will use it to make a prediction on the holdout set. Another model of interest is the model using $\alpha = 0.5$ and $\lambda = 0.0041$. This model has a test set AUC of 68.0257, which is slightly lower than the first mentioned setting. It does however also reduce the model from the initial 151 variables to a model consisting of only 40 variables. Since we would prefer a parsimonious model, we will see how this model performs on the holdout set. The 40 variables that are left and their coefficients can be seen in Table 18.

Table 18: Elastic net model, $\alpha = 0.5$

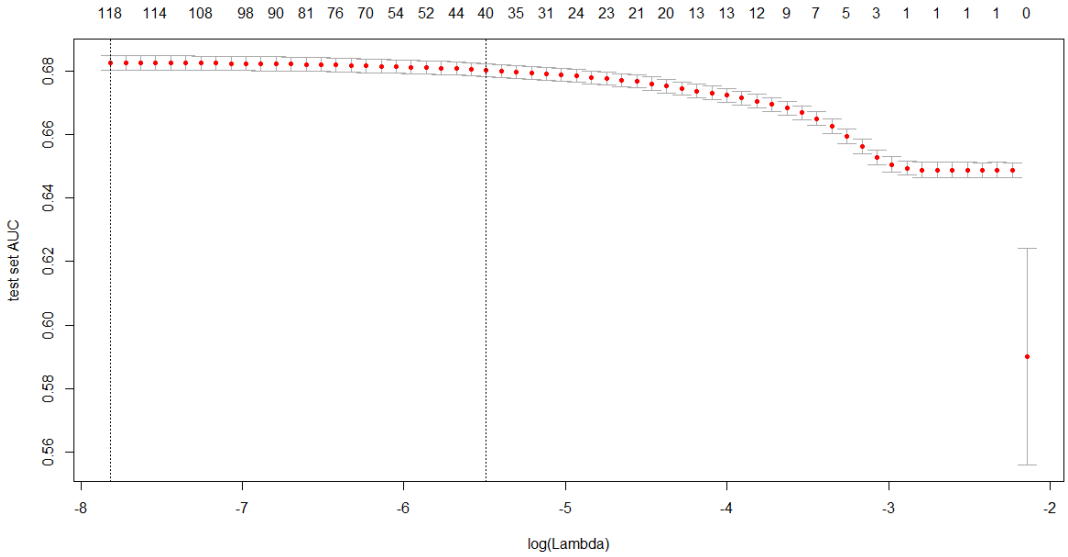
Variable name	Coefficient	Variable name	Coefficient
Intercept	-4.4329	grade.A	-0.1023
int rate	0.0729	grade.C	0.0780
installment	-0.0001	grade.D	0.0547
issue d	0.0002	sub grade.A1	-0.0599
dti	0.0119	sub grade.A3	-0.0425
delinq 2yrs	0.0106	sub grade.C3	0.0025
inq last 6mths	0.0399	sub grade.D1	0.0121
total rev hi lim	-0.0000	sub grade.D2	0.0189
acc open past 24mths	0.0415	emp length.0	0.2804
bc open to buy	-0.0000	emp length.10..years	-0.0130
mo sin old rev tl op	-0.0004	home own MORTGAGE	-0.0222
mo sin rcnt tl	-0.0025	home own RENT	0.1172
mort acc	-0.0112	verif. status Source	0.0236
num rev tl bal gt 0	0.0058	purpose.credit card	-0.0337
num tl op past 12m	0.0231	purpose.medical	0.0177
percent bc gt 75	0.0005	purpose.small business	0.3593
tot hi cred lim	-0.0000	mths sin last delinq...1.12.	0.0355
total il high credit limit	-0.0000	mths sin last record.3.4.years	-0.0241
fico average	-0.0031	mths sin recent bc...4.years	-0.0905
monthly debt rate	3.9950	mths sin recent bc dlq.3.4.years	-0.0351
-	-	mths sin rcnt revol delq.3.4.yrs	-0.0104

The variables left in the model with $\alpha = 0.5$, $\lambda = 0.0041$.

In Figure 12 the results from the cross-validation when setting alpha to 0.5 and using different values of λ are shown. The left dotted vertical line represents the λ value that gives the best AUC. The second line indicates the largest λ within one standard error from the optimal λ. The numbering along the top of the plot shows how many nonzero values are present in the model at the given level of λ.

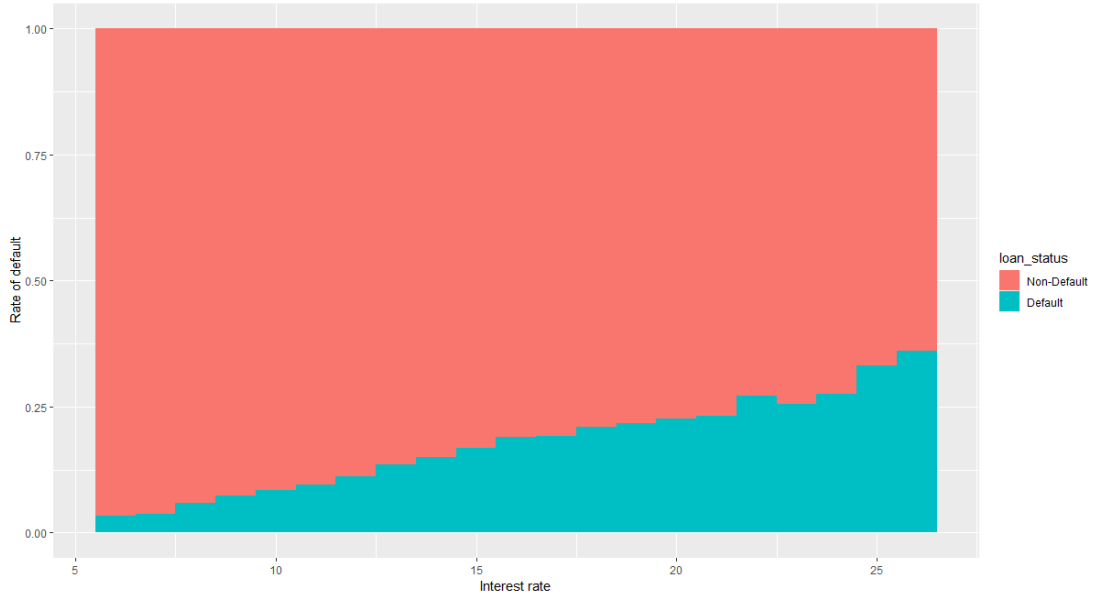
We see there is a significant decrease in the number of variables present between the two suggested values of λ. The reduction in the AUC is sparse, so in the interest of a parsimonious model, the lambda value one standard error away from the minimum might be preferred. In addition to this we see that we are able to reduce the number of variables in the model all the way down to 1, while still retaining an AUC of approximately 0.65. This variable is interest rate, and as expected is closely related to the probability of default. Figure 13 shows that there is a near-linear relationship between the interest rate and the rate of defaults in the training set. Since this single variable model performs very well, we will include this as a baseline model for which we can compare the other techniques implemented.

Figure 12: test set AUC for a range of λ values, given $\alpha = 0.5$



Vertical dotted lines indicates the λ-value that gives the highest AUC and the largest λ-value within one standard error of the optimal value. Number of remaining variables at the different levels of λ is noted at the top of the plot.

Figure 13: Rate of default for given interest rates



4.1.2 Xgboost

Table 13 in Section 3.2.2 shows the different hyperparameters available for tuning when implementing the extreme gradient boosting algorithm in MLR. Other parameters exist, but these are the ones that are optimized for in this thesis. Table 19 summarizes the results of the model optimizations, and the parameter space searched through for each of the hyperparameters. The search path and results from the initial search are found in Table 38 in the Appendix.

Table 19: Hyperparameter settings for the xgboost model

Parameter name	Search Space	Tune result	Test set AUC
Initial search			
Max depth	2 to 6	5	
Nrounds	100 to 500	234	
Eta	0.01 to 0.99	0.0457	
Gamma	0.01 to 0.3	0.2997	
Min child weight	0.5 to 10	2.7110	
Subsample	0.5 to 1	0.6120	
Colsample bytree	0.5 to 1	0.8111	0.68886
2-level tree			
Max depth	2 to 2	2	
Nrounds	200 to 700	644	
Eta	0.03 to 0.6	0.1791	
Gamma	0.275 to 0.35	0.3001	
Min child weight	2.5 to 3.25	3.1499	
Subsample	0.5 to 0.9	0.6083	
Colsample bytree	0.7 to 0.9	0.7123	0.68940
5-level tree			
Max depth	5 to 5	5L	
Nrounds	200 to 700	513	
Eta	0.01 to 0.99	0.0517	
Gamma	0.01 to 0.3	0.3202	
Min child weight	0.5 to 10	3.1437	
Subsample	0.5 to 1	0.6291	
Colsample bytree	0.5 to 1	0.7165	**0.68950**

The hyperparameters and final tuning results

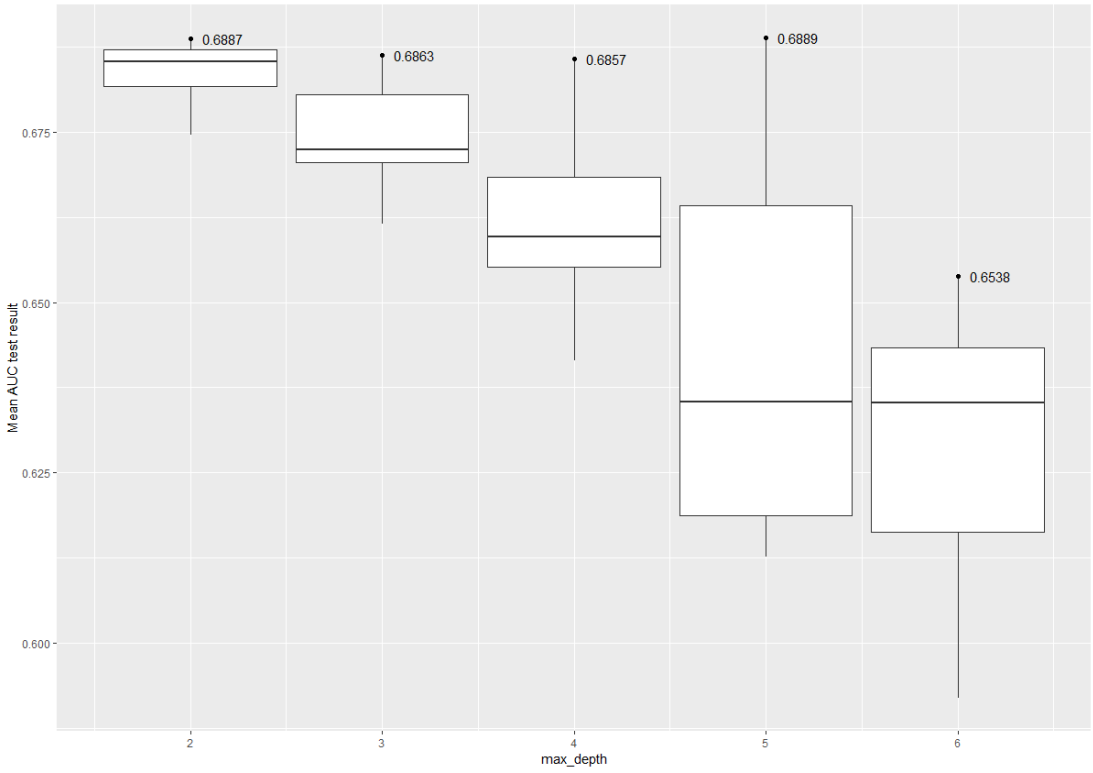
The first random search was done for 50 combinations within the parameter space listed for each of the parameters in Table 19. The hyperparameter settings that performed best are listed under initial search in the table.

Figure 14 displays the distribution of the test results from the initial tuning, divided between the different levels of tree depth. It is clear that the trees with only two levels are far more reliable in their performance. This is consistent with the theory for boosting models, where low complexity models often yield more reliable results. High complexity models on the other hand can provide more accurate results, but are susceptible to more variability. We see that the settings that perform the best is a setup with max depth being set to 5, but there is a lot of variability within the 5-level models.

To see if we can optimize further, secondary searches for models with *max depth* = 2 and models with *max depth* = 5 are done. Random searches for 50 iterations in the region close to the previously found hyperparameters gives the results listed in Table 19 under 2-level tree and 5-level tree.

Using random searches is admittedly not the most optimal approach, since there are interaction effects between the parameters. We would ideally do a grid search for the optimal settings, but with so many parameters the number of combinations would quickly lead to unreasonable computation time. The method applied does however give us an indication of how well the boosting technique works on our dataset.

Figure 14: Test AUC for the initial random search, sorted by depth of the trees.



Boxplots of the initial tuning settings. The numbers at the top indicate the best results within the group.

The results obtained when using these models to make predictions on the holdout set is found in Section 4.2.2.

4.1.3 Random forests

Table 20 shows the different hyperparameters we tune in our implementations of random forests. The results of the initial search and the parameter space for each parameter is shown along with the result for the subsequent retuning done on a smaller model and the final large model.

What we attempt to achieve when tuning the hyperparameters is to obtain the optimal compromise between low correlation and reasonable strength of the trees. This is known

Table 20: Hyperparameter settings for the random forests

Parameter name	Search Space	Tune result	Test set AUC
Default parameters			
Mtry	-	12	
Num.trees	-	500	
Min.node.size	-	1	
Sample.fraction	-	0.632	0.67679
Small Model			
Mtry	1 to 25	4	
Num.trees	250	250	
Min.node.size	1 to 500	84	
Sample.fraction	0.07	0.07	0.67874
Final model			
Mtry	1 to 25	4	
Num.trees	700 to 2000	1000	
Min.node.size	1 to 500	84	
Sample.fraction	0.35 to 0.35	0.35	**0.68263**

The hyperparameters and final tuning results

as the bias-variance trade-off, and can be controlled for by adjusting the hyperparameters *mtry*, *sample fraction* and *minimum node size*.

The default value for *mtry* is usually set to $\sqrt{p}$ where p is the number of independent variables in the model. This is usually a reasonable value, but it can sometimes be improved depending on the dataset used. It depends on the real number of relevant predictors in the dataset. If there are many predictors with a large influence on the dependent variable, then *mtry* should be set small so that not only the strongest predictors are chosen in the splits. On the other hand, if there are few influential predictors then *mtry* should be set to a large number so that the trees can find the relevant predictors. For the dataset used in this thesis $\sqrt{p} \approx 12$, so we will do a search in the region around 12 to look for the optimal value for *mtry*.

The *sample fraction* decides how many sample observations are drawn to build each tree. Decreasing the sample size leads to more diverse trees which in turn leads to less correlation when aggregating the trees. Larger sample size leads to more accuracy and higher correlation, so again there is a trade-off here between bias-variance.

Martínez-Muñoz and Suárez [2010] claim that there is no substantial performance difference between sampling with replacement or without replacement when the sample size parameter is set optimally. However, Janitza et al. [2016] finds that in cases where there are categorical variables with varying number of categories, sampling with replacement might induce a slight variable selection bias. Since we have categorical variables in our dataset, we use sampling without replacement as our sampling scheme.

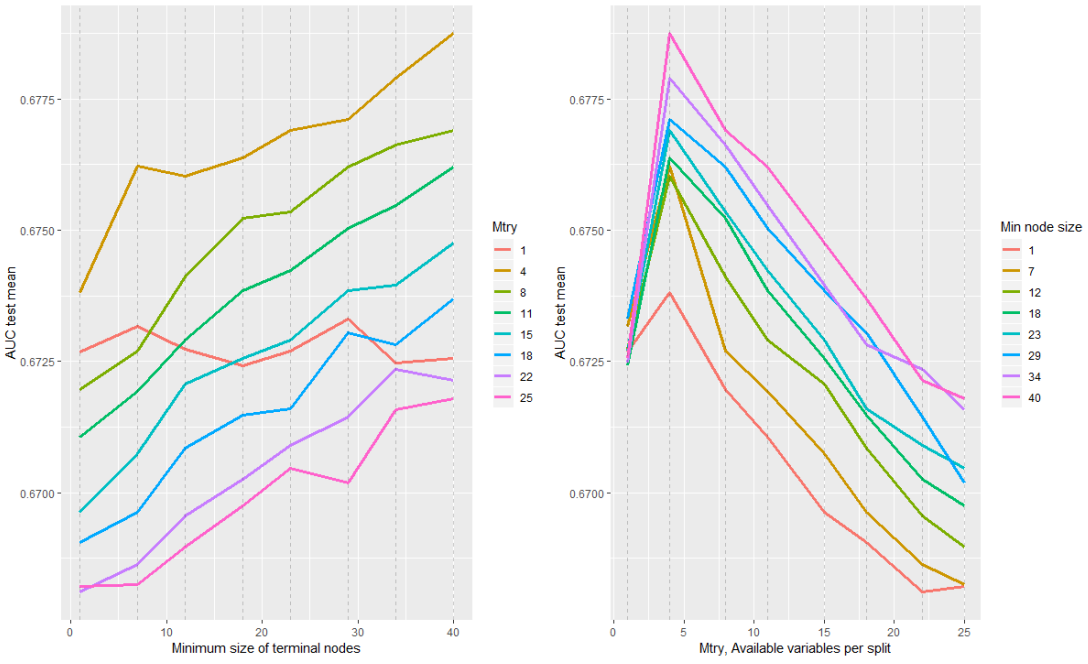
The *minimum node size* defines the minimum number of observations required in a terminal node in the trees, and it is by default set to 1 for classification tasks. Probst et al. [2018b] finds that computation time decreases approximately exponentially when increasing minimum node size. For large samples, they find that it might be advantageous to set this parameter to a value higher than the default as it decreases the runtime substantially, often without loss of prediction performance.

The number of trees grown and consequently aggregated obviously has a large impact on the performance of the model. Probst and Boulesteix [2017] proves theoretically that more trees are always better. In the same paper they show that the biggest performance gain is achieved when growing the first 100 trees. The computation time increases linearly with the number of trees.

Fernández-Delgado et al. [2014] finds that the performance increases obtainable by tuning the hyperparameters of a random forests model is far less than other machine learning algorithms. Nevertheless, when testing on 38 different datasets, they find a small performance gain from tuning compared to using the default hyperparameters. They find that tuning the parameter *mtry* gives the biggest average improvement, followed by *sample size*.

Based on the above we don't expect to get a large improvement in performance from tuning the various hyperparameters, but attempts are made to improve upon the default hyperparameter settings. We start our search for the optimal settings by first tuning *mtry* and *minimum node size* while keeping the number of trees and fraction of samples used low. The idea is that we can speed up the learning process by first optimizing using a smaller model, and then later on train a model on a larger number of trees using a larger fraction of the available observations for each tree. We expect to see an improvement in performance when increasing the size of the model.

Figure 15: Tuning results for mtry and minimum node size



Test set AUC for combinations of mtry and min node size.

First we train a model using the default values as implemented in the Ranger package in R. Since the expected improvement from tuning is fairly low in random forests models, this default model is used as a baseline model to compare to. The search space and tuning results can be found in Table 20.

To search for optimal hyperparameters, we continue by performing a grid search for the

optimal settings of *mtry* and *minimal node size*. We set the *sample fraction size* to be 0.07 and *number of trees* to be 250. This is a large reduction when compared to the default settings, but it will greatly improve the speed of tuning the models.

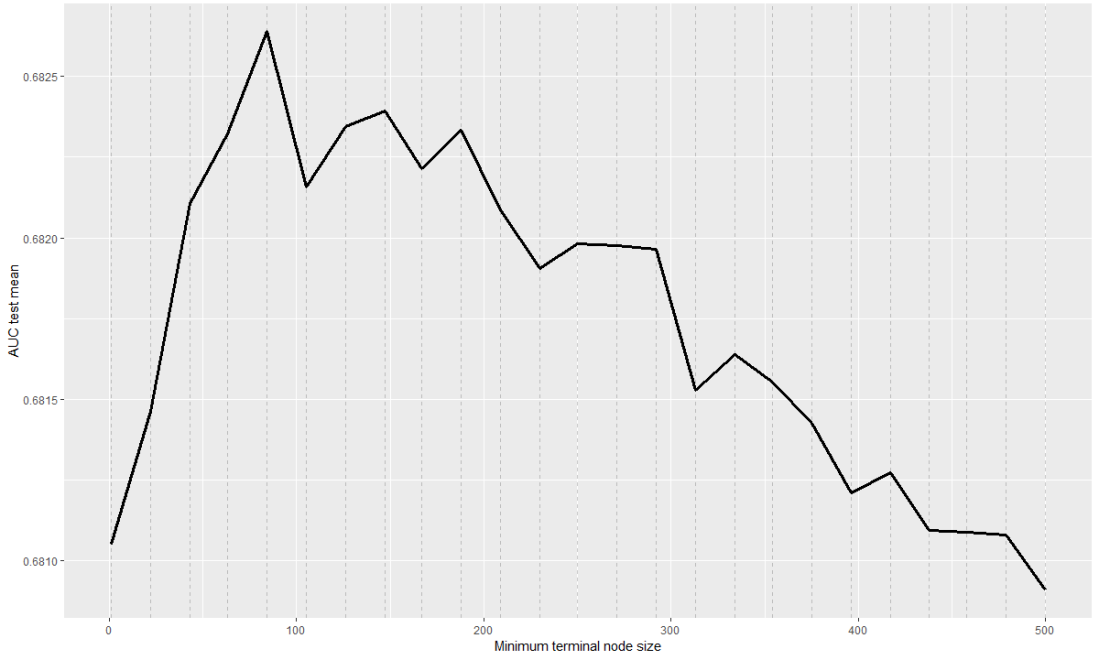
The initial results can be found in Figure 15 and in Table 20 labeled small model.

Figure 15 shows that the model performs best when the available variables for each split is set to 4. We also notice a distinct increase in performance from higher values of *minimum node size*. An additional tuning is done with *minimum node size* set to 40, for values of *mtry* in the range 1 to 8. This second search confirms that a *mtry* value of 4 is the optimal value.

Following these results, subsequent tuning is done for the optimal value of *minimum node size*. We search in the range 20 to 500, since there was a trend of increased performance when node size was increased. The results from this tuning of node size can be found in Figure 16. From the Figure the best performance is found when *minimum node size* is set to be 84.

Using the above results, we train the model using a larger sample fraction and a larger number of trees. This final model has the best performance found so far, as shown in Table 20.

Figure 16: Tuning minimum node size.



4.1.4 K-nearest neighbors

To find the optimal hyperparameter settings for our K-nearest neighbors model (KNN), we use the fastknn package in R. This package has an integrated cross-validation function, that allows for easy tuning to find the optimal value for k. The other hyperparameters we tune for are normalization strategy and classification method, as mentioned in Table 15.

We will tune the hyperparameters using a smaller subset of the data. This is to reduce the computational effort required to find the best parameters. The KNN-algorithm does not train a model in the same way that random forests or boosting techniques do. Instead the training is based around "memorizing" the different observations and the distance between each. Since the tuning is rather fast when the sample size is reduced, we are able to do a thorough search for the optimal parameter settings. We do the initial tuning on a subset consisting of 15 % of the training set. When we fit the final model we will increase the fraction to 40 % of the training set, and similarly make the prediction on 40 % of the holdout data. In Table 21 and Figure 17 we see the search space and tuning results for the different normalization strategies and a range of values for k.

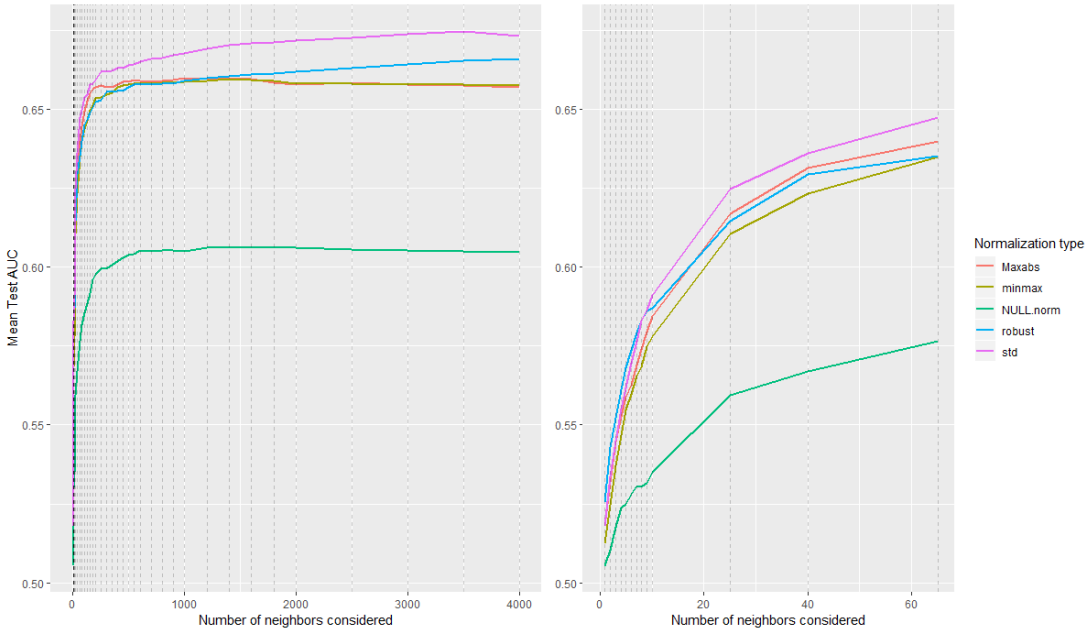
Table 21: Test AUC results for the various normalization and voting methods

Model	Optimal K	Test set AUC
Normalization technique = NULL		
Classification method = vote	1800	0.6040
Classification method = dist	2000	0.6061
Normalization technique = minmax		
Classification method = vote	1400	0.6599
Classification method = dist	1400	0.6596
Normalization technique = maxabs		
Classification method = vote	1400	0.6585
Classification method = dist	1000	0.6597
Normalization technique = robust		
Classification method = vote	4000	0.6653
Classification method = dist	4000	0.6658
Normalization technique = std		
Classification method = vote	3500	0.6740
Classification method = dist	3500	0.6746
KNN final model	3500	0.6740

The tuning results for the different normalization techniques for values of k in the range 1 to 4000.

Figure 17 shows that the standardize normalization technique gives the best results for this dataset. We also see from Table 21 that the weighted models perform better for all the models, except the minmax normalized model. Based on these findings, we optimize the model using standardizing normalization further by fitting it on a larger dataset, while also fine-tuning the number of neighbors to include. The final results are shown in Table 21 under *normalization technique* = *std*, labeled KNN final model.

Figure 17: Test AUC results for combinations of k and normalization techniques



Left plot: The tuning results for k in the range 1 to 4000.
Right: Same tuning results as above, shown for values of k in the range 1 to 65.
Vertical dotted lines signify the actual k-values tuned for.

4.2 Accuracy metrics

Several different metrics exist for measuring the success of a classification algorithm. Following is a reference for some of the most used accuracy metrics, and the formulas for calculating each. True positive is a measure of how many of the predictions that were made for the positive class were really positive observations. In this case, we are looking at how many of the observations that our models predict as default were really loans that defaulted. Similar to the true positive classification, the true negative measures how many of the observations that were predicted as non-default, were actually non-default loans. The false positives are a measure of how many of the loans the model erroneously predicted to be default that were actually non-default. Finally, the false negative is a measure of how many of the defaulted loans that were erroneously labeled non-default.

Confusion matrix

The confusion matrix is a useful tool for collecting and measuring classification accuracy statistics. An example is found in Table 22. The rows are labeled with the true values, while the columns hold the predictions made for each class of the outcome. In the table we then get a measure of how many true positive, true negative, false positive and false negative observations the fitted model predicted. The confusion matrix is also the first step in calculating several other metrics, such as specificity, sensitivity, precision, recall and the AUC metric which are all presented below.

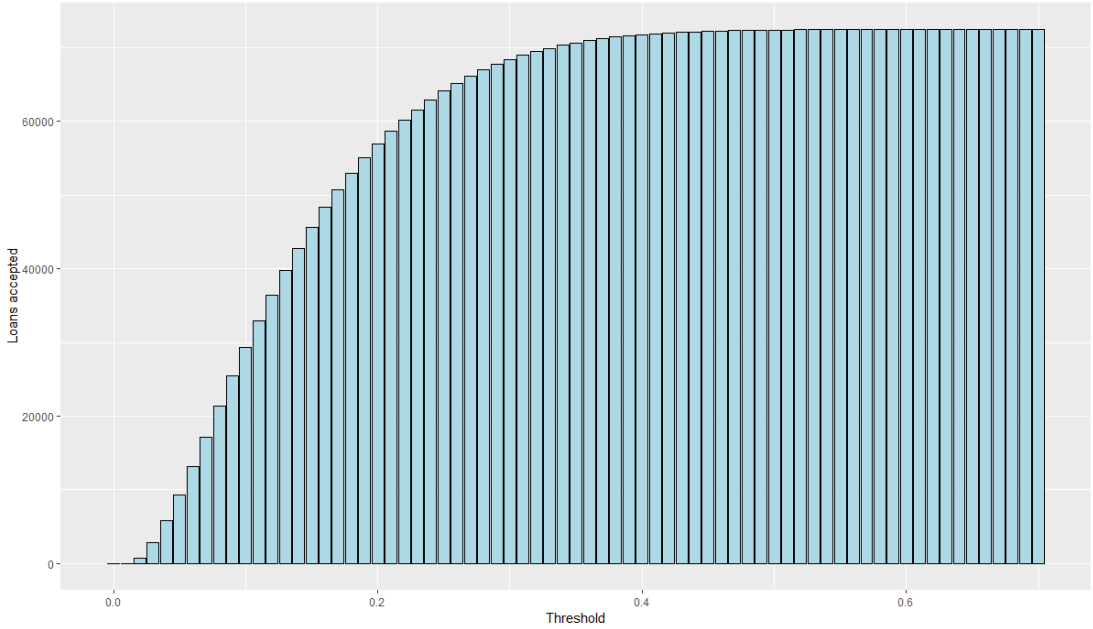
Table 22: Confusion matrix example

		PREDICTED	
		Default	**Non-Default**
TRUTH	**Default**	True positive	False negative
	Non-Default	False positive	True negative

The models fit in this thesis output a probability for each observation in the range 0 to 1. The way to obtain the predicted class labels is then done by setting a threshold for how large the probability of an observation belonging to a class needs to be to be labeled as that class. Shifting the threshold can drastically change the predicted output of the classes.

Essentially, the threshold selection is a trade-off where a toleration level is set for the number of misclassifications of a given class. If we set the threshold very low, then all observations will be classified as default. This will give a near-perfect accuracy on the defaulted loans, but we would get close to zero correct non-default predictions, which would make the model useless. Similarly, we could set the threshold too high, and end up classifying all loans as Non-Default. Figure 18 shows how many loans are accepted at different threshold settings. In this case the probability of default needs to be lower than the threshold for the loan to be classified as non-default.

Figure 18: Number of loans accepted at different thresholds



A representation of how many loans are accepted and declined at various thresholds.

In the case of default prediction, it is far costlier to misclassify defaults than to erroneously classify a non-default loan as default. An incorrectly classified default leads to a loss of money, while a misclassified non-default leads to a loss of potential income. Seeing as we are looking to optimize a portfolio of loans and maximize profits, we are better off when keeping the misclassified defaults low while still accepting a fair amount of loans for investment. This means that the threshold for classifying a loan as default will have to be quite low.

If we are very strict and only allow 5 % of actual defaulted loans to be classified as non-default in the test set, meaning we require the true positive rate to be 0.95, we would decline to invest in a lot of loans that would not actually default. On the other end, if we require a lower rate of defaults to be accurately predicted we might end up investing in too many loans that default. In Table 23 we see the classification results when setting the threshold to obtain a given rate of true positive predictions.

Table 23: Loan acceptance at different threshold values

TPR	Loans accepted	Bad loans accepted	Loans declined	Good loans declined
0.95	13,224	548	59,275	49,451
0.85	26,369	1,590	46,130	37,348
0.70	39,270	3,080	33,229	25,937
0.50	52,264	5,144	20,235	15,007
0.25	64,039	7,748	8,460	5,836

Number of loans accepted and declined at various levels of true positive rate.

As Table 23 shows, adjusting the true positive rate requirement has a huge impact on how many loans will be accepted by our models. Making use of this setting we can get a decent basis for comparing the different models if we set some restrictions on how

many of the defaults we require to be accurately classified. We decide to use a true positive rate of approximately 0.85 for all the models, so that we get even grounds for comparison. At this setting we should have a healthy amount of loans accepted, while keeping the rate of default loans accepted fairly low.

Following is a brief introduction to the different accuracy metrics presented in the Sections for each individual model.

Accuracy

Accuracy is the ratio of correct classifications over all classifications. The formula to calculate it is:

$$\text{Accuracy} = \frac{\text{True positive} + \text{True negative}}{\text{All observations}} \tag{4.2.1}$$

In the case of the lending club data set, this measure is not very useful since the dataset is quite unbalanced. The accuracy measure doesn't discriminate between correctly classified defaults or non-defaults. As such, any model that predicts that all loans are non-default will obtain an accuracy of approximately 85 %. An accuracy higher than that is hard to obtain if we value the cost of misclassifications differently, which is the case for default prediction.

Sensitivity

The sensitivity measure is the ratio of the number of correct positive examples to the number classified as positive. It is also called the true positive rate or recall, and it is calculated by:

$$\text{Sensitivity} = \frac{\text{True positive}}{\text{True positive} + \text{False negative}} \tag{4.2.2}$$

It is a useful measure for cases where there is an unbalanced dataset, since it allows us to measure how well the model performs on the positive class in the dataset.

Specificity

Specificity measures the ratio of correct negative examples to the number classified as negative. Also known as the true negative rate, it is calculated by:

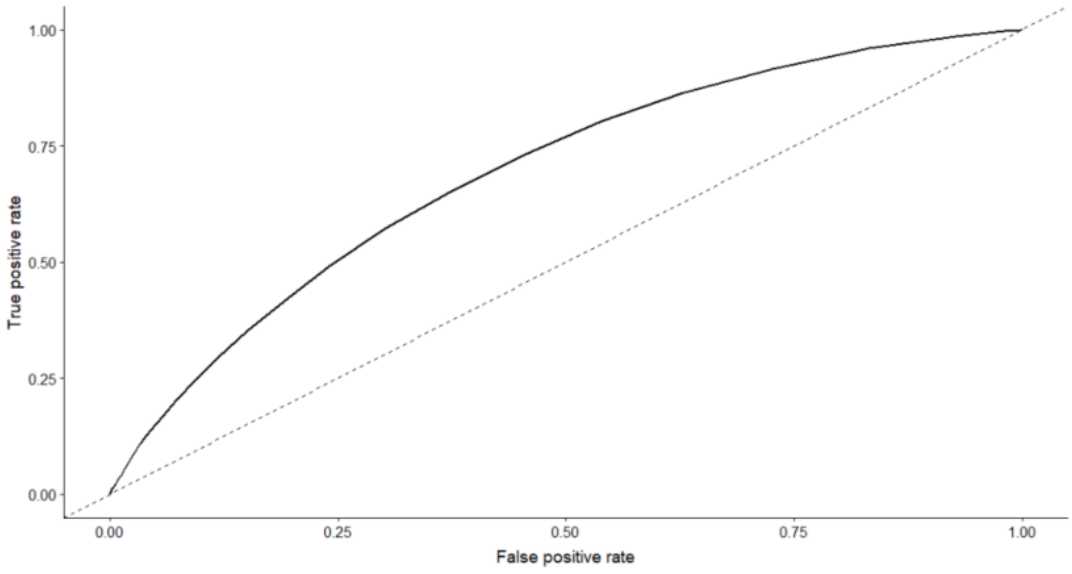
$$\text{Specificity} = \frac{\text{True negative}}{\text{True negative} + \text{False positive}} \tag{4.2.3}$$

AUC

The Area under the receiver operator characteristics curve (AUROC, referred to as the AUC from here on out) is a measure that can be used when we have a binary classification problem like in this thesis. The ROC-curve is a function of the sensitivity (or true positive rate) on the y-axis, and false positive rate (1-specificity) on the x-axis for a range of different thresholds.

In Figure 19 we see an example of how the AUC plot looks. The dashed line in the middle represents a classifier that splits the observations into two equal classes. If a model performs better than random chance, it would have a ROC-curve that would tend towards the upper left corner of the plot. A perfect classifier would show as a straight line from (0,0) to (0,1) to (1,1), indicating that it obtains complete separation between the classes.

Figure 19: Example AUC plot



F1 measure

The F1 measure is the weighted average of the recall and precision measures. It is calculated by:

$$F_1 = 2 \cdot \frac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}} \tag{4.2.4}$$

This can be useful because of the relation between the precision and recall statistic. Precision is the ratio of true positives over all predicted positives. Recall is the ratio of

all positives over all actual positive observations. Since these measures take false positives and false negatives into account, the F1 measure can be a useful measure when the distribution between the classes is unequal. The score one can obtain from the f-measure ranges from 0 to 1, where 1 is the optimal score.

IRR as a profit measure

From the payment history dataset we extract the cashflow for each individual loan. With this knowledge of what payments were made, we are able to calculate the internal rate of return (IRR) for each loan to get a tangible representation of which loans are more profitable. This can then be used to compare which model selects the most profitable loan portfolio by calculating the overall IRR given a situation where we invest the same amount in each of the accepted loans.

IRR is a well-known financial formula that may be fairly easily computed for investments that have an initial cash outflow followed by several cash inflows. To find the IRR we first need to calculate the net present value (NPV) of the loans.

The NPV is found by

$$\text{NPV} = \sum_{t=1}^{T} \frac{C_t}{(1+r)^t} - C_0, \tag{4.2.5}$$

where C_t is the cashflow during the period t, C_0 is the initial cash outflow, r is the discount rate and T is the total number of payments.

The NPV is a measure of the value of an investment when the time-cost is considered. It builds on the idea that future money has less value than presently available funds, since the money today can be invested to increase its value. It is a useful way to compare different investment opportunities when you know or are able to accurately estimate the future cashflow. In the case of the LendingClub data, we have access to the exact cashflow from the payment history dataset. Based on this we can calculate the NPV and use a root function to find the IRR, since the IRR is the discount rate required for the NPV to be equal to zero. Generally speaking, the higher the IRR of a loan is, the more desirable the project is to invest in.

Figure 20 shows the distribution of the annualized internal rate of return for all the loans in the dataset. We see that all non-default loans have a positive IRR along with some of the defaulted loans. Almost 10 % of the defaulted loans has a positive IRR, indicating that they are profitable even though they eventually default.

Figure 20: The distribution of annualized internal rate of return in the loan portfolio.

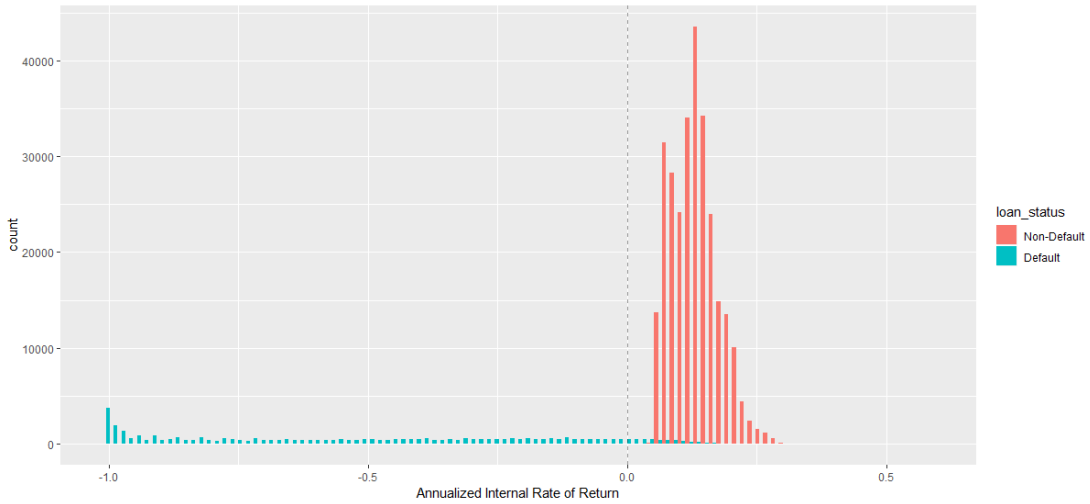


Table 24: Summary statistics - IRR

Minimum	Q1	Median	Mean	Q3	Maximum
-1.00	0.08	0.12	0.05	0.15	0.58

The summary statistics for the IRR variable.

From the summary statistics, we see that the highest IRR is 0.58. This is a loan that is fully repaid with the first down payment. Similar themes hold true for the other loans that return the highest IRR. They are all paid back early, leading to good returns for the lender. On the other end of the scale, we see that there are quite a few loans that have an IRR of -1. All of these are loans that defaulted before the first repayment was made. The mean is 0.05, which is considerably skewed towards the negative due to the large number of loans that default prior to the first repayment being made. In fact, the mean is smaller than nearly 87 % of the observations. The results from using this measure for the various models can be found in Section 4.3.2.

4.2.1 Elastic net

Table 25 shows the accuracy metrics for the two elastic net models and the univariate interest rate model. The difference in the AUC between the two elastic net models is now lower than it was during training. Both models achieve a better AUC on the holdout set than they did on the training data. Since the model with $\alpha = 0.5$ also implemented a stricter degree of variable selection, it is the preferred model out of the two.

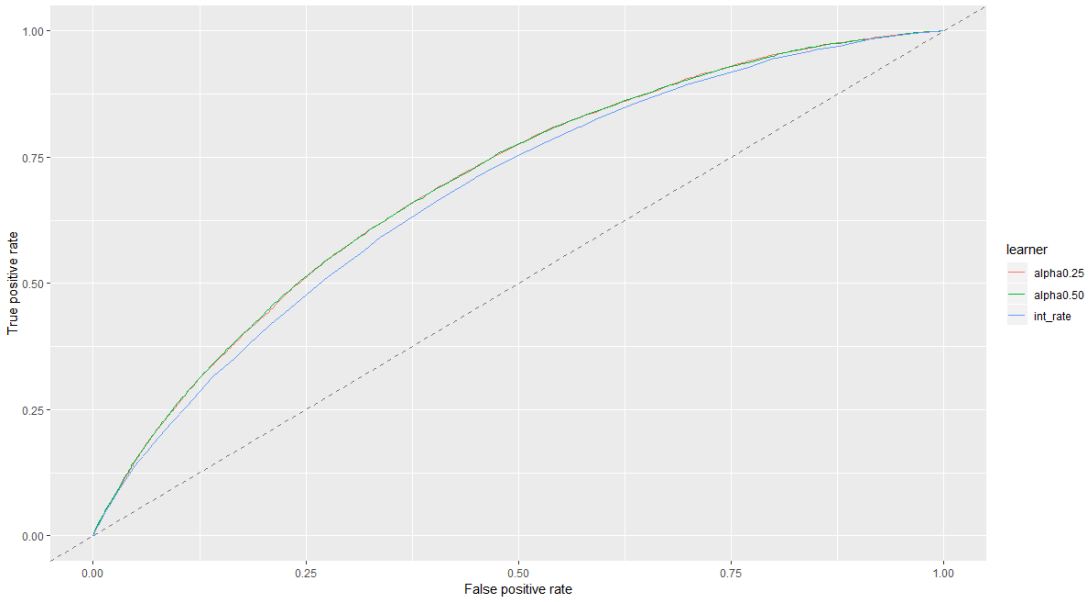
Table 25: Accuracy statistics - elastic net

Model	Accuracy	Sensitivity	Specificity	AUC	f1
Alpha = 0.25	0.46035	0.84996	0.39475	0.69500	0.31122
Alpha = 0.50	0.45879	0.84914	0.39306	0.69495	0.31141
Int rate univar	0.43550	0.85560	0.36470	0.67760	0.30400

Accuracy results on the holdout set for the chosen parameter settings.

In Figure 21 we see the ROC-curves for the two elastic net models, and the univariate model built using interest rate as the only predictor. The two elastic net models are very close, while the univariate model performs a bit worse.

Figure 21: AUC plot for the two elastic net models and the univariate interest rate model.



4.2.2 Xgboost

Similar to the other models, we set the threshold for correct classification of defaults to be 85 %, so that we get equal grounds for comparison between the models. Using the parameter settings listed in Table 19 we get the following accuracy statistics.

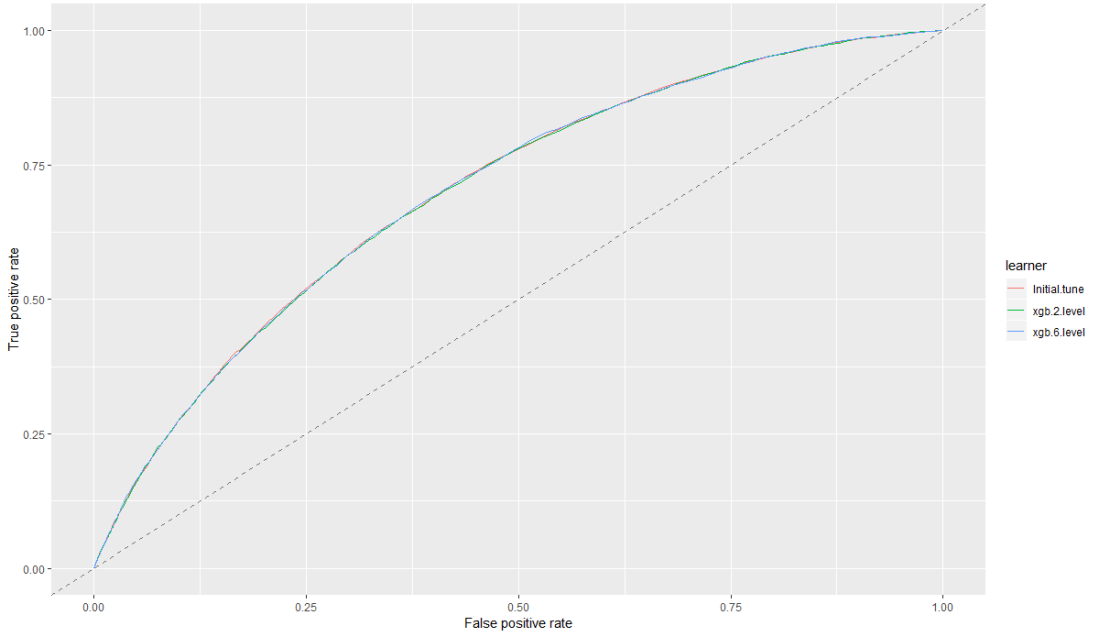
Table 26: Accuracy statistics - xgboost

Model	Accuracy	Sensitivity	Specificity	AUC	f1
Initial tuning	0.46284	0.85078	0.39752	0.69965	0.31343
2-level trees	0.46276	0.85050	0.39747	0.69795	0.31333
5-level trees	0.46947	0.84978	0.40543	0.70027	0.31585

Xgboost accuracy results on the holdout set for the three chosen parameter settings.

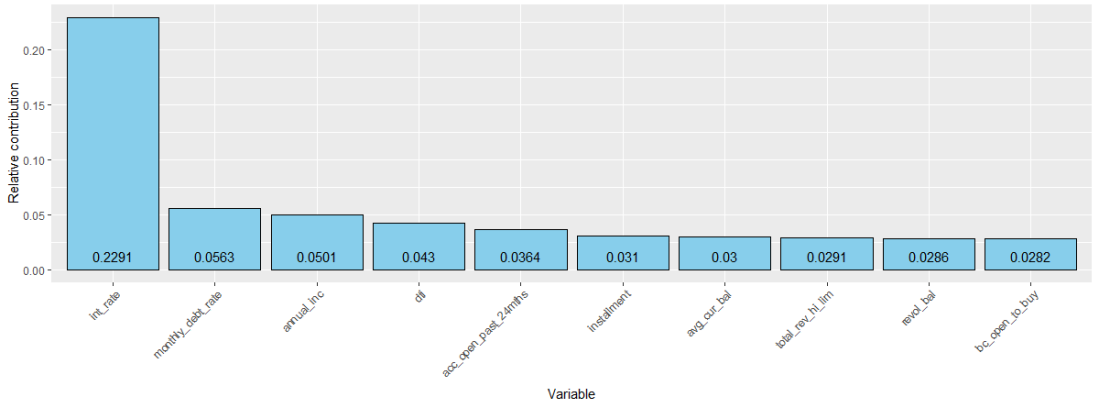
In Figure 22 we see that the ROC-curves for the three models are almost identical. There is also very little difference between the models in the other accuracy metrics calculated.

Figure 22: AUC for the three xgboost models



Looking at the variable importance measure for the initial tuning model, we can get a sense of which of the variables are providing the most information for predicting the outcome. Interest rate is clearly the variable that has the largest impact on the model. For all three extreme gradient boosting models fit, the five most important variables were the same. Other variables with large impacts are the monthly debt rate, annual income, debt to income rate and whether the loan is grade A or not.

Figure 23: Variable importance measure for the 5-level xgboost model.



A measure of how often each variable was chosen for a split in a tree.

4.2.3 Random forests

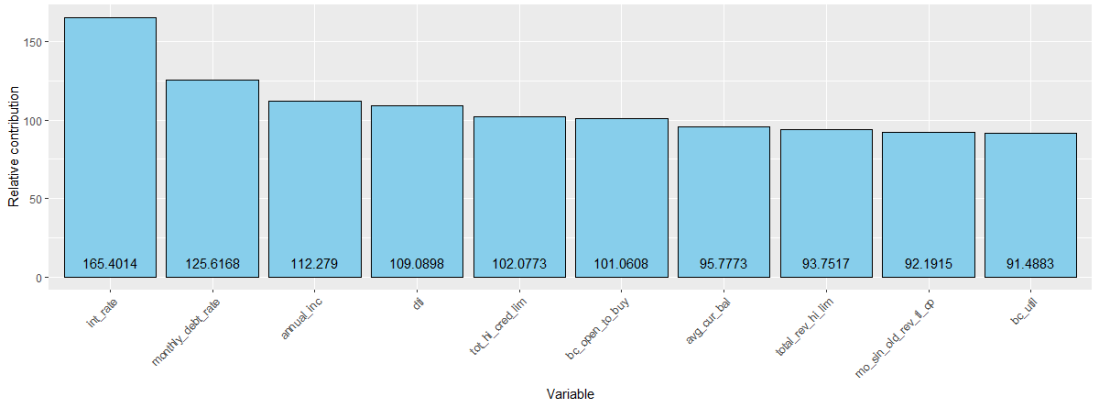
For the random forests algorithm we test the results for each of the three models on the holdout set. The accuracy statistics for each model can be found in Table 27. The output is fairly consistent with the results obtained on the test sets during the cross-validation. We see that when comparing the default settings with the small model and the final model, there is an increase of the AUC of ≈ 0.006 and 0.009, respectively. This is in line with what was found in Probst et al. [2018a]. In the cited study they found an average increase in AUC from tuning of 0.012 when testing on 38 different datasets, so there might still be room for improvement in our model.

Table 27: Accuracy statistics - random forests

Model	Accuracy	Sensitivity	Specificity	AUC	f1
Default model	0.44554	0.85050	0.37735	0.68659	0.30658
Small model	0.45355	0.85050	0.38671	0.69244	0.30969
Final model	0.46002	0.85050	0.39427	0.69571	0.31224

Random forests accuracy results on the holdout set for the three chosen parameter settings.

Figure 24: Variable importance measure for the final random forests model

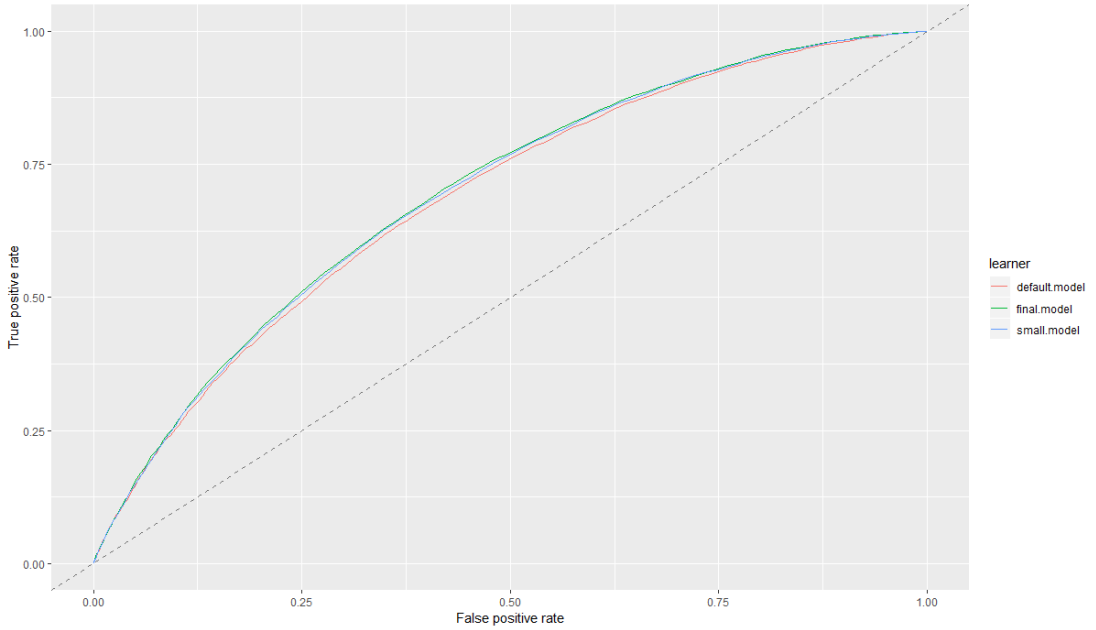


A measure of how often each variable is chosen for the split.

Figure **??** shows the ten variables that contribute most in the final model. The four most important variables; interest rate, monthly debt rate, annual income and dti are the same as the four most important variables found by the xgboost model in Section 4.2.2.

The AUC-plot for the random forests models is found in Figure 25. The models are very close to each other, but the final model has the best performance.

Figure 25: AUC for the three random forests models



4.2.4 K nearest neighbors

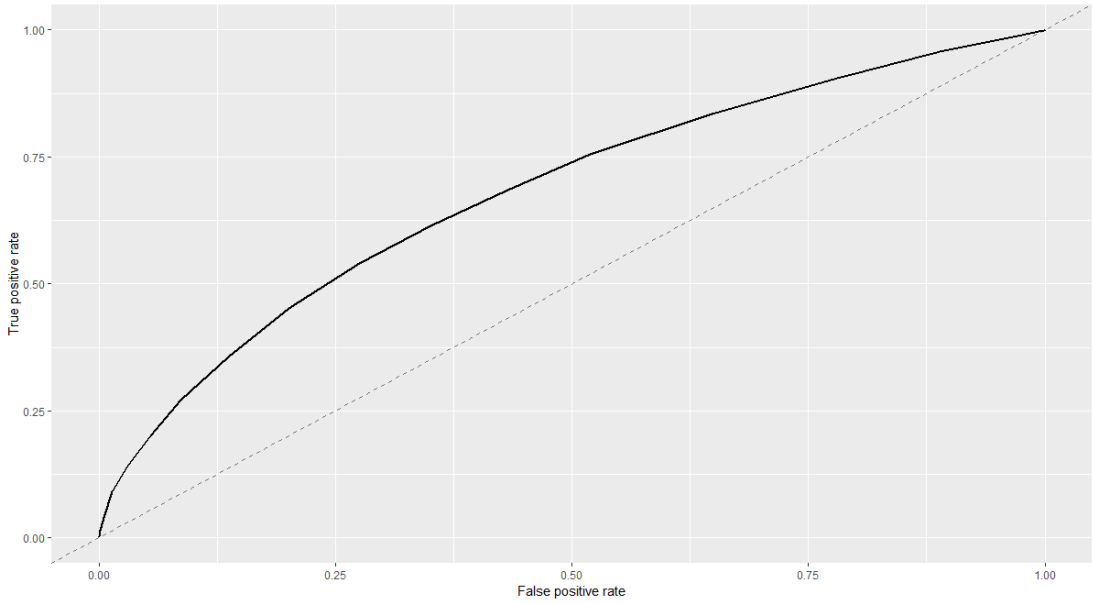
Table 21 shows that the hyperparameter settings for KNN that gave the best results is the model using the standardize normalization technique. We retuned the model on a larger subset of the data before testing the prediction accuracy on the holdout data set. The accuracy statistics obtained when testing the model is listed in Table 28. The results are similar to what was found on the test sets during the fitting process.

Table 28: Accuracy statistics - KNN

Model	Accuracy	Sensitivity	Specificity	AUC	f1
Final model	0.4468	0.8500	0.3782	0.6808	0.3088

KNN accuracy results on the holdout set for the parameter settings found previously.

Figure 26: AUC for the final KNN model



The KNN technique is a non-parametric learning method that gives us no indication of which variables are important for the accuracy of the model.

4.3 Comparison of results

In this Section the different models will be compared. First we look at the accuracy statistics produced by each model. This is followed up by testing if there is significant difference between the better performing models using McNemar tests. Finally, using the *internal rate of return* variable, we compare the profitability of the models for various levels of investment.

Table 29: Combined accuracy metrics

Model	Accuracy	Sensitivity	Specificity	AUC	f1
Elastic net					
Alpha = 0.25	0.46035	0.84996	0.39475	0.69500	0.31122
Alpha = 0.50	0.45879	0.84914	0.39306	0.69495	0.31141
Interest rate	0.43550	0.85560	0.36470	0.67760	0.30403
Xgboost					
Initial tuning	0.46284	0.85078	0.39752	0.69965	0.31343
2-level trees	0.46276	0.85050	0.39747	0.69795	0.31333
5-level trees	0.46947	0.84978	0.40543	0.70027	0.31585
Random forests					
Default model	0.44554	0.85050	0.37735	0.68659	0.30658
Small model	0.45355	0.85050	0.38671	0.69244	0.30969
Final model	0.46002	0.85050	0.39427	0.69571	0.31224
KNN					
Final model	0.44680	0.85000	0.37820	0.68080	0.30880

The accuracy metrics for the different models.

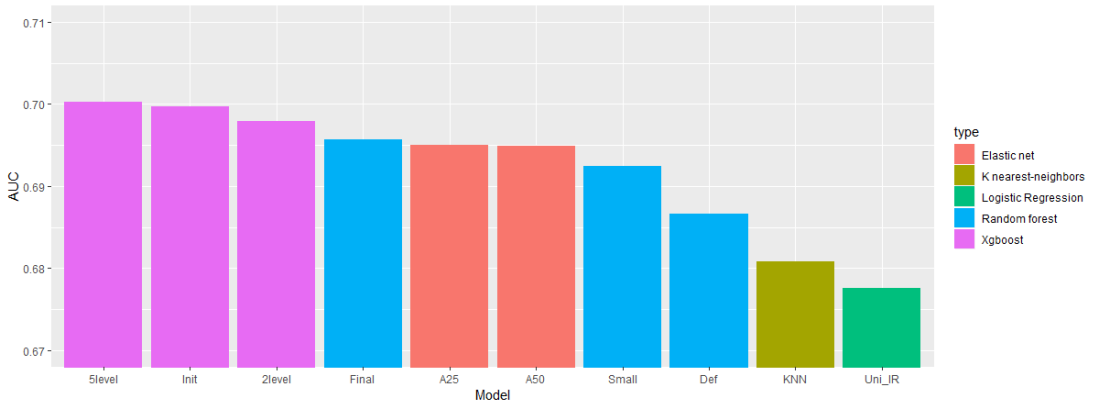
Table 29 shows the accuracy statistics for the different models. Table 40 in the Appendix holds the results for the models on the dataset where the outliers have been removed. We find no particular improvement for any of the models. We did not perform as rigorous tuning of hyperparameters on the dataset of the outliers, so there is a possibility that it could be improved further. However, as the models currently performs worse than the full dataset, we conclude that the original model settings are fairly robust to outlying observations and will not pursue the alternative set any further.

We tuned all the models to get a sensitivity of approximately 0.85. This was done to get even grounds for comparison of the different models. Some of the models deviate slightly from 0.85, since it is difficult and sometimes impossible to find a threshold that gives the desired split. Looking at the accuracy, we see that all the models misclassify more than half the observations. This is expected since we define the default predictions to be more important than the non-default. As a result, there are many loans that are erroneously classified as default. This is reflected in the specificity in the table. The specificity is (in our situation) a measure of how many of the loans classified as non-default that are really non-default. We see that there is not a lot of difference in the results for the various models. Most of the models produce results that lie in the range between 0.39 and 0.40. The small discrepancies in the sensitivity between the models will also resonate in the specificity rating. The model that has the highest specificity is the xgboost model with 5 levels in the trees.

When we look at the AUC-score we see similar results as we did for the specificity. The

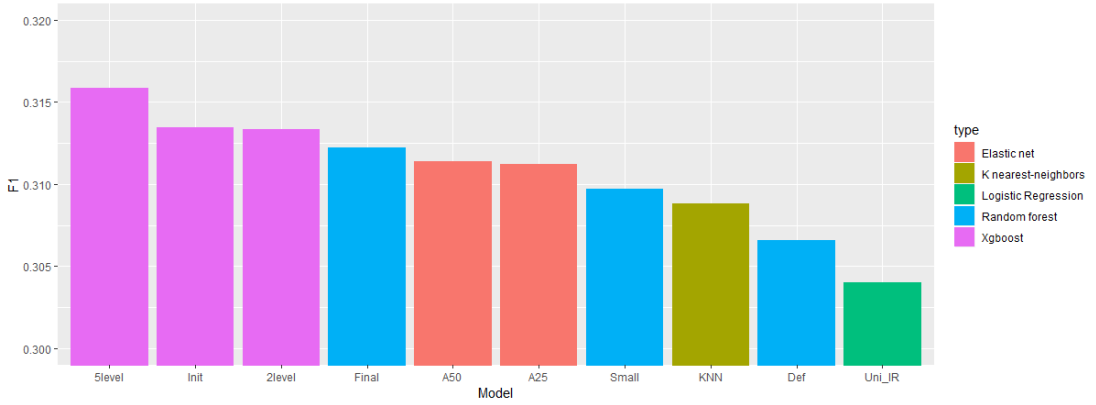
performance of the various models is fairly close, but we see that the xgboost models are best overall. The xgboost model with max depth of 5 has the largest AUC on the holdout set. Figure 27 gives a graphical representation of the AUC for the various models.

Figure 27: Comparison of the AUC for all models



Looking at the F1 statistic we see similar results as we did for the AUC. The xgboost models obtain the best results, closely followed by the random forests models and the elastic net models. Figure 28 shows the differences graphically.

Figure 28: The F1 scores for the different models



4.3.1 McNemar tests

Since the models get results that are fairly close to one another we would like to apply a test to see if there is significant difference between the better models. We can do this by using McNemar test, first introduced in McNemar [1947]. The McNemar test is a non-parametric statistical test for paired comparisons that can be used to compare the performance of two classification models.

It compares the predictions of two models, by building a 2x2 contingency table that

shows which observations both models classified correctly, which either model misclassified and the other got correct, and which observations both models misclassified. Table 30 illustrates the setup of the contingency table.

Table 30: Example McNemar contingency table

		Model 2	
		Correct	**Wrong**
Model 1	**Correct**	A	B
	Wrong	C	D

An example of how the McNemar contingency table is set up.

Based on these numbers, we use a chi-squared test to see if there is significant difference between B and C in Table 30. From our models, the xgboost models were the best performing. The model with max depth of 5 gave the highest AUC. Of the other techniques, the final random forests model had the highest AUC followed closely by the elastic net models. We will see if there is a significant difference between the 5 level xgboost model, the final random forests model, and the elastic net model with alpha set to 0.5.

Table 31: McNemar contingency table - xgboost vs random forests

		Random forests	
		Correct	**Wrong**
Xgboost	**Correct**	31,751	3,374
	Wrong	4,095	37,134

Prediction overlap between xgboost 5-level and random forests final models.

From Table 31 we see that there is a fairly significant amount of observations that one model predicts correct while the other misclassifies. Testing for significant difference between the two values we get a p-value significantly below 0.01. Similar results are obtained for the other comparisons, which indicates that the performance of the models is significantly different. The contingency table for the other comparisons can be found in the Appendix.

4.3.2 IRR as profitability measure

To utilize the IRR as a profitability measure we need to consider what classification the model gives each loan. Based on the loans that each model predicts to be non-default, we can calculate the total profitability of each model.

We consider each loan classified as non-default as a loan that will be invested in. This will give a measure of the profitability of a classifier over the complete dataset. This might not realistically reflect an investment situation though, since the total number of loans invested in will be very large.

A second approach sets a maximum number of total loans each model will invest in to reflect a situation where the lender has limited funds to invest. Table 32 displays the results for 500, 1,000, 5,000 and 10,000 loans accepted, so that we get an overview of the profitability of the different models at various settings.

The loans are selected based on their predicted probability of default given by each model. In a real-world situation we might be better off by setting a threshold for which loans we will accept, but for comparison of models this approach is difficult since the given probability for each loan varies by a lot in the different models.

Table 32: Profit measures

Model	**Profit**	**Total loans**	**500**	**1,000**	**5,000**	**10,000**
Elastic net, $\alpha = 0.25$	5.20	27,298	5.81	5.70	5.61	5.48
Elastic net, $\alpha = 0.5$	5.07	27,173	**6.22**	**6.13**	5.53	5.59
Univariate model	4.74	13,462	5.04	5.27	4.93	4.63
Rf-default	5.20	26,226	5.77	5.93	5.70	5.56
Rf-final	5.09	27,599	5.83	5.92	**5.85**	5.67
Rf-small	5.04	27,113	5.77	5.81	5.75	5.49
Xgboost-2level	5.39	27,507	5.92	5.99	5.72	**5.74**
Xgboost-5level	**5.56**	28,495	5.90	6.07	5.82	5.68
Xgboost-Initial	5.29	27,828	5.90	6.08	5.74	5.66
Flat investment	2.78	76,354	-	-	-	

Profits for each model for different numbers of accepted loans.

Table 32 shows that the model that returns the largest profit changes based on how many loans are accepted and invested in. We also see that the flat investment, where all loans are accepted gives an IRR of 2.78 %. All the other models improve upon this result. For the full holdout-set the 5 level xgboost model returns the largest IRR. For lower volume of investment the elastic net model with reduced variable set gives good returns, while the final random forests model gives the best return when we invest in 5,000 loans. Overall the 5 level xgboost model gives the largest returns.

Chapter 5

Summary and conclusion

In this thesis peer-to-peer banking and credit-scoring has been explored. Here, the main findings will be summarized and potential shortcomings and directions for future studies are suggested.

5.1 Summary

In the first part of this thesis the datasets are presented. The variables are explained, and changes are made where necessary to make the variables usable in the model building stage. Some variables contain data gathered after the loan has been issued. These are removed to have a dataset that reflects the investment situation a lender would experience when searching for loans to invest in. Missing observations are removed from the dataset and decisions are made regarding what loans to keep. We trim the dataset so that we only keep loans that have reached maturity. In addition, we include only 36-month loans so that we can do an intertemporal split of the data to help protect the model against seasonal cycles that might affect the stability of the models.

Exploratory data analysis is performed, and we find that some variables are highly colinear. These are removed to simplify the modeling phase, even though some of the learning techniques are capable of handling multicollinearity. The payment history dataset is used to extract the individual loan payments for each loan. These are then used to calculate the Internal rate of return which is later implemented as a profitability measure to be used when comparing the different models.

In Chapter 3 the different models are implemented. First binary regression is presented before the extension into generalized linear models and elastic net models are shown. After that we explore the decision tree models, specifically boosting using extreme gradient boosting and bagging with random forests models. Finally, a K nearest neighbors model is implemented.

The models are tuned to find the optimal settings for the hyperparameters. For the elastic net we use a grid search to find the optimal value for alpha and lambda. We find that we obtain the best results with alpha set to 0.25, which reduces the variable set to 118 variables. A secondary model is also found where the lasso regularization is stronger. This leads to a model with slightly lower performance with regards to the AUC, but it reduces the number of variables included in the model to 40.

For the extreme gradient boosting models, we use a random search to find the best hyperparameter settings. The first search indicates that models with a maximum depth of 5 gives the best results, but they also far more varied. More consistent and nearly as good are models with max depth of 2. This leads us to do extended searches for both versions. We find the highest AUC in the 5-level model, although the 2-level model is very close.

Continuing with decision trees we build random forests models. We find the optimal parameters by doing a grid search on a smaller subset of the loans. This gives us clear indication that using 4 variables per split in the tree gives the best results. Following some additional tuning to find the optimal number of observations in the terminal nodes we land on a final model that obtains a decent AUC score.

When training the K nearest neighbors algorithm we use a smaller subset of the data to find the optimal number of neighboring observations to include when classifying new data. In addition to tuning for k, we tested various settings for normalization of the data and two different voting techniques to decide what classification to give the target observation. We found that the best normalization technique is one that standardizes the data by removing the mean and scaling each observation to the unit variance. The best voting technique we found was using the inverse of the distance between the target and the training data to weigh the importance of the training data. The KNN model did not reach the same AUC as the other models, but it performed fairly well for being such a simple technique.

Finally, in Chapter 4 a comparison is made between the different models. We look at the accuracy statistics for all the models and find that the extreme gradient boosting models does the best overall. They obtain the highest AUC and also does comparably well when comparing the F1 measure. The difference between the performance of the elastic net, random forests and boosting models is fairly small, so a test to verify that the difference is significant is performed. The test concludes that the best performing models from xgboost, random forests and elastic net are all significantly different to each other. While the xgboost method obtains the best results, the models can be difficult to understand. A case can thus be made for the elastic net model. Here we get a model that is easier to understand, with readily interpretable variables and coefficients. The performance is also very close to that of the other methods, so it might be preferable to use in practice.

In addition to the accuracy statistics, we calculate the internal rate of return one would obtain if an investment was made in the loans selected from each model. Here the results are more varied. If one would invest in all the loans the model predicts as non-default, the largest return would be from the 5-level boosting model. With more stringent restrictions however, we see that the elastic net model with lasso regularization does the best. If investing in the 5,000 loans deemed safest by each model, we see that the random forests model gives the best results.

When we compare the profitability of each model to the baseline flat investment in all the loans, we see that all the suggested models return a larger profit. We also see that a univariate model using only the interest rate set by LendingClub gives decent results for both accuracy and profitability.

5.2 Shortcomings

While we are able to find models that have decent prediction accuracy and give larger returns on profit than the baseline model, the increase in performance is rather small. Since the univariate interest rate model performs comparably well, it indicates that the credit scoring done by LendingClub is fairly accurate. The interest rate thus constitutes a decent indicator of the risk involved with the loan, and the improvements from using the learning techniques are somewhat limited.

Another issue is the period for which the data is collected. Ideally we would like a larger sample, so we could use a holdout set that covered a longer period. A longer period would provide better protection against seasonal effects that might affect the probabilty of default. On the other hand, we've seen that LendingClub updates the variable sets, which might become an issue if using a larger dataset.

When we look at the performance of the models on the training data and the holdout set, we see that all the models perform better on the holdout set than the training set. This is odd, and we can't really explain why this would be the case. Special attention has been made to not include any of the data from the holdout set in the training of the models to ensure that no information is transferred. We have also made sure that the results of any predictions are not influencing the selection of hyperparameters. All selection of hyperparameters is done based on the results from the cross-validated test results on the training set.

5.3 Further research

The analysis done in this thesis may be a good starting point for further research into peer-to-peer lending and credit scoring in this market. A possible approach could be to extend the dataset to cover a longer period of time. An extension to the boosting models implemented here could be to investigate ensemble methods where several models are combined to produce a final prediction. Models of this kind allows us to take advantage of the strengths of the different modeling techniques. From the result from the McNemar test we see that there is a lot of observations that one model predicts correctly while the other model predicts wrong. An ensemble of these learners might be able to improve on the total performance if these models are applied at different areas of the dataset.

Another approach that could be implemented is a cost-sensitive classifying system. In this modeling method we give different costs to misclassifying loans. For instance we set the cost of erroneously classifying a defaulted loan as non-default to be higher than the converse. Further than that we can set the cost of misclassifying loans that are from a higher credit grade to be more expensive than loans in lower grades. Since it is known that the loans in the higher grades are riskier we should be especially certain that they won't default before we invest in them. This could reduce the overall risk and possibly improve the performance of the models.

Appendix A

Tables

A.1 Variable description

Table 33: All available variables

Variable name	Variable description	Missing
Borrower Assesment		
fico range high	The upper boundary range the borrower's FICO at loan origination belongs to	2,213 / 0.1 %
fico range low	The lower boundary range the borrower's FICO at loan origination belongs to	2,213 / 0.1 %
grade	LC assigned loan grade	- / - %
int rate	Interest Rate on the loan	- / - %
sub grade	LC assigned loan subgrade	- / - %
Borrower Characteristics		
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	- / - %
emp length	Employment length in years	- / - %
emp title	The job title supplied by the Borrower when applying for the loan	4 / 0 %
home ownership	The home ownership status provided by the borrower during registration or obtained from the credit report	- / - %
verification status	Indicates if income was verified by Lending Club	- / - %
zip code	The first 3 numbers of the zip code provided by the borrower	2,214 / 0.1 %
Borrower Indebtedness		
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	3,054 / 0.2 %
Credit History		
acc open past 24mths	Number of trades opened in past 24 months	2,213 / 0.1 %
chargeoff within 12 mths	Number of charge-offs within 12 months	2,213 / 0.1 %
collections 12 mths ex med	Number of collections in 12 months excluding medical collections	2,213 / 0.1 %
delinq 2yrs	The number of 30+ days past-due incidences of delinquency in the borrower's credit file for the past 2 years	2,213 / 0.1 %
inq fi	Number of personal finance inquiries	794,938 / 44.1 %
inq last 12m	Number of credit inquiries in past 12 months	794,939 / 44.1 %
inq last 6mths	The number of inquiries in past 6 months (excluding auto and mortgage inquiries)	2,214 / 0.1 %
mo sin old il acct	Months since oldest bank installment account opened	56,848 / 3.2 %
mo sin old rev tl op	Months since oldest revolving account opened	2,293 / 0.1 %

Continued on next page

Variable name	Variable description	Missing
mo sin rcnt rev tl op	Months since most recent revolving account opened	2,293 / 0.1 %
mo sin rcnt tl	Months since most recent account opened	2,292 / 0.1 %
mths since last delinq	The number of months since the borrower's last delinquency	897,716 / 49.8 %
mths since last major derog	Months since most recent 90-day or worse rating	1,311,977 / 72.8 %
mths since last record	The number of months since the last public record	1,495,377 / 83 %
mths since rcnt il	Months since most recent installment accounts opened	824,611 / 45.8 %
mths since recent bc	Months since most recent bankcard account opened	20,542 / 1.1 %
mths since recent bc dlq	Months since most recent bankcard delinquency	1,359,878 / 75.5 %
mths since recent inq	Months since most recent inquiry	196,943 / 10.9 %
mths since recent revol delinq	Months since most recent revolving delinquency	1,179,601 / 65.5 %
num accts ever 120 pd	Number of accounts ever 120 or more days past due	2,292 / 0.1 %
num tl 90g dpd 24m	Number of accounts 90 or more days past due in last 24 months	2,292 / 0.1 %
num tl op past 12m	Number of accounts opened in past 12 months	2,292 / 0.1 %
open acc 6m	Number of open trades in last 6 months	794,939 / 44.1 %
open act il	Number of installment accounts opened in past 12 months	794,938 / 44.1 %
open il 12m	Number of installment accounts opened in past 24 months	794,938 / 44.1 %
open rv 12m	Number of revolving trades opened in past 12 months	794,938 / 44.1 %
open rv 24m	Number of revolving trades opened in past 24 months	794,938 / 44.1 %
pct tl nvr dlq	Percent of trades never delinquent	2,446 / 0.1 %
pub rec	Number of derogatory public records	2,213 / 0.1 %
pub rec bankruptcies	Number of public record bankruptcies	2,213 / 0.1 %
tax liens	Number of tax liens	2,213 / 0.1 %
tot coll amt	Total collection amounts ever owed	2,292 / 0.1 %
Current financial characteristics		
acc now delinq	The number of accounts on which the borrower is now delinquent	2,213 / 0.1 %
all util	Balance to credit limit on all trades	795,056 / 44.1 %
avg cur bal	Average current balance of all accounts	2,332 / 0.1 %
bc open to buy	Total open to buy on revolving bankcards	21,728 / 1.2 %
bc util	Ratio of total current balance to high credit/credit limit for all bankcard accounts	22,686 / 1.3 %
delinq amnt	The past-due amount owed for the accounts on which the borrower is now delinquent	2,213 / 0.1 %
il util	Ratio of total current balance to high credit/credit limit on all install acct	934,812 / 51.9 %
max bal bc	Maximum current balance owed on all revolving accounts	794,938 / 44.1 %
mort acc	Number of mortgage accounts	2,213 / 0.1 %
num actv bc tl	Number of currently active bankcard accounts	2,292 / 0.1 %
num actv rev tl	Number of currently active revolving trades	2,292 / 0.1 %
num bc sats	Number of satisfactory bankcard accounts	2,213 / 0.1 %
num bc tl	Number of bankcard accounts	2,292 / 0.1 %
num il tl	Number of installment accounts	2,292 / 0.1 %
num op rev tl	Number of open revolving accounts	2,292 / 0.1 %
num rev accts	Number of revolving accounts	2,293 / 0.1 %
num rev tl bal gt 0	Number of revolving trades with balance >0	2,292 / 0.1 %
num sats	Number of satisfactory accounts	2,213 / 0.1 %
num tl 120dpd 2m	Number of accounts currently 120 days past due (updated in past 2 months)	76,755 / 4.3 %
num tl 30dpd	Number of accounts currently 30 days past due (updated in past 2 months)	2,292 / 0.1 %
open acc	The number of open credit lines in the borrower's credit file	2,213 / 0.1 %
open il 24m	Number of currently active installment trades	794,938 / 44.1 %
percent bc gt 75	Percentage of all bankcard accounts > 75% of limit	22,168 / 1.2 %
revol bal	Total credit revolving balance	2,213 / 0.1 %
revol util	Revolving line utilization rate, or the amount of credit the borrower is using relative to all available revolving credit	3,450 / 0.2 %
tot cur bal	Total current balance of all accounts	2,292 / 0.1 %
tot hi cred lim	Total high credit/credit limit	2,292 / 0.1 %
total acc	The total number of credit lines currently in the borrower's credit file	2,213 / 0.1 %
total bal ex mort	Total credit balance excluding mortgage	2,213 / 0.1 %
total bal il	Total current balance of all installment accounts	794,938 / 44.1 %
total bc limit	Total bankcard high credit/credit limit	2,213 / 0.1 %

Continued on next page

Variable name	Variable description	Missing
total cu tl	Number of finance trades	794,939 / 44.1 %
total il high credit limit	Total installment high credit/credit limit	2,292 / 0.1 %
total rev hi lim	Total revolving high credit/credit limit	2,292 / 0.1 %
Dependent variable		
loan status	Current status of the loan	- / - %
Loan characteristics		
earliest cr line	The month the borrower's earliest reported credit line was opened	2,213 / 0.1 %
application type	Indicates whether the loan is an individual application or a joint application with two co-borrowers	- / - %
desc	Loan description provided by the borrower	- / - %
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	- / - %
id	A unique LC assigned ID for the loan listing	- / - %
initial list status	The initial listing status of the loan. Possible values are – W, F	- / - %
installment	The monthly payment owed by the borrower if the loan originates	- / - %
issue d	The month which the loan was funded	- / - %
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	- / - %
member id	A unique LC assigned Id for the borrower member	1,802,029 / 100 %
policy code	publicly available policy code=1, new products not publicly available policy code=2	2,213 / 0.1 %
purpose	A category provided by the borrower for the loan request	- / - %
pymnt plan	Indicates if a payment plan has been put in place for the loan	- / - %
term	The number of payments on the loan. Values are in months and can be either 36 or 60	- / - %
title	The loan title provided by the borrower	- / - %
url	URL for the LC page with listing data	- / - %
Loan performance		
collection recovery fee	post charge off collection fee	2,213 / 0.1 %
last credit pull d	The most recent month LC pulled credit for this loan	2,286 / 0.1 %
last fico range high	The upper boundary range the borrower's last FICO pulled belongs to	2,213 / 0.1 %
last fico range low	The lower boundary range the borrower's last FICO pulled belongs to	2,213 / 0.1 %
last pymnt amnt	Last total payment amount received	2,213 / 0.1 %
last pymnt d	Last month payment was received	29,442 / 1.6 %
next pymnt d	Next scheduled payment date	839,235 / 46.6 %
out prncp	Remaining outstanding principal for total amount funded	2,213 / 0.1 %
out prncp inv	Remaining outstanding principal for portion of total amount funded by investors	2,213 / 0.1 %
recoveries	post charge off gross recovery	2,213 / 0.1 %
total pymnt	Payments received to date for total amount funded	2,213 / 0.1 %
total pymnt inv	Payments received to date for portion of total amount funded by investors	2,213 / 0.1 %
total rec int	Interest received to date	2,213 / 0.1 %
total rec late fee	Late fees received to date	2,213 / 0.1 %
total rec prncp	Principal received to date	2,213 / 0.1 %
Secondary applicant		
annual inc joint	The combined self-reported annual income provided by the co-borrowers during registration	1,733,976 / 96.2 %
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	1,733,980 / 96.2 %
revol bal joint	Sum of revolving credit balance of the co-borrowers, net of duplicate balances	1,746,665 / 96.9 %

Continued on next page

Variable name	Variable description	Missing
sec app chargeoff within 12 mths	Number of charge-offs within last 12 months at time of application for the secondary applicant	1,746,664 / 96.9 %
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	1,746,664 / 96.9 %
sec app earliest cr line	Earliest credit line at time of application for the secondary applicant	1,746,664 / 96.9 %
sec app fico range high	FICO range (low) for the secondary applicant	1,746,664 / 96.9 %
sec app fico range low	FICO range (high) for the secondary applicant	1,746,664 / 96.9 %
sec app inq last 6mths	Credit inquiries in the last 6 months at time of application for the secondary applicant	1,746,664 / 96.9 %
sec app mort acc	Number of mortgage accounts at time of application for the secondary applicant	1,746,664 / 96.9 %
sec app mths since last major derog	Months since most recent 90-day or worse rating at time of application for the secondary applicant	1,783,037 / 98.9 %
sec app num rev accts	Number of revolving accounts at time of application for the secondary applicant	1,746,664 / 96.9 %
sec app open acc	Number of open trades at time of application for the secondary applicant	1,746,664 / 96.9 %
sec app open act il	Number of currently active installment trades at time of application for the secondary applicant	1,746,664 / 96.9 %
sec app revol util	Ratio of total current balance to high credit/credit limit for all revolving accounts	1,747,614 / 97 %
Created variables		
Monthly debt rate	Installment amount divided by monthly income to see added debt burden	- / - %
Average FICO	Mean of the FICO lower and FICO upper limit provided by LC	- / - %

This table contains all the variables initially in the dataset along with information on how many observations are missing for each variable. Missing count is done prior to any reduction in the dataset.

Table 34: Final set of variables

Variable name	Variable description
Borrower assesment	
grade	LC assigned loan grade
int rate	Interest Rate on the loan
sub grade	LC assigned loan subgrade
Borrower characteristics	
annual inc	The self-reported annual income provided by the borrower during registration
emp length	Employment length in years
home ownership	The home ownership status provided by the borrower during registration or obtained from the credit report
verification status	Indicates if income was verified by Lending Club
Borrower indebtedness	
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
Credit history	
acc open past 24mths	Number of trades opened in past 24 months
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
inq last 6mths	The number of inquiries in past 6 months (excluding auto and mortgage inquiries)
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
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 recent bc	Months since most recent bankcard account opened
mths since recent bc dlq	Months since most recent bankcard delinquency
mths since recent revol delinq	Months since most recent revolving delinquency
num accts ever 120 pd	Number of accounts ever 120 or more days past due
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
pct tl nvr dlq	Percent of trades never delinquent
pub rec	Number of derogatory public records
pub rec bankruptcies	Number of public record bankruptcies
tot coll amt	Total collection amounts ever owed
Current financial characteristics	
acc now delinq	The number of accounts on which the borrower is now delinquent
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
delinq amnt	The past-due amount owed for the accounts on which the borrower is now delinquent
mort acc	Number of mortgage accounts

Continued on next page

Variable name	Variable description
num actv bc tl	Number of currently active bankcard accounts
num op rev tl	Number of open revolving accounts
num rev tl bal gt 0	Number of revolving trades with balance >0
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)
open acc	The number of open credit lines in the borrower's credit file
percent bc gt 75	Percentage of all bankcard accounts > 75% of limit
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
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 bc limit	Total bankcard high credit/credit limit
total il high credit limit	Total installment high credit/credit limit
total rev hi lim	Total revolving high credit/credit limit
Dependent variable	
loan status	Current status of the loan
Loan characteristics	
installment	The monthly payment owed by the borrower if the loan originates
issue d	The month which the loan was funded
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
purpose	A category provided by the borrower for the loan request
Created variables	
Monthly debt rate	Installment amount divided by monthly income to see added debt burden
FICO average	Mean of the FICO lower and FICO upper limit provided by LC

The variables included in the final dataset.

Table 35: Descriptive statistics for categorical variables

Variable	N	%	Defaults	%	Phi	χ^2	P-value
Grade							
A	69,473	21.31	3,661	5.27	-0.124	4,990.13	0.000, ***
B	115,017	35.29	12,222	10.63	-0.059	1,145.83	0.000, ***
C	83,809	25.71	14,133	16.86	0.061	1,197.16	0.000, ***
D	42,318	12.98	9,313	22.01	0.098	3,142.34	0.000, ***
E	12,271	3.76	3,299	26.88	0.079	2,014.49	0.000, ***
F	2,818	0.86	831	29.49	0.044	637.66	0.000, ***
G	235	0.07	81	34.47	0.017	88.73	0.000, ***
Sub-grade							
A1	9,870	3.03	251	2.54	-0.056	1,027.68	0.000, ***
A2	10,572	3.24	412	3.9	-0.051	844.22	0.000, ***
A3	11,635	3.57	495	4.25	-0.051	863.21	0.000, ***
A4	17,191	5.27	1,002	5.83	-0.052	888.31	0.000, ***
A5	20,205	6.2	1,501	7.43	-0.045	653.78	0.000, ***
B1	20,297	6.23	1,679	8.27	-0.039	483.31	0.000, ***
B2	23,624	7.25	2,159	9.14	-0.035	391.37	0.000, ***
B3	25,518	7.83	2,712	10.63	-0.023	178.08	0.000, ***
B4	24,542	7.53	2,804	11.43	-0.016	85.5	0.000, ***
B5	21,036	6.45	2,868	13.63	0.002	1.45	0.229,
C1	20,933	6.42	3,022	14.44	0.008	22.37	0.000, ***
C2	18,717	5.74	2,995	16	0.019	119.63	0.000, ***
C3	16,913	5.19	2,989	17.67	0.03	286.52	0.000, ***
C4	14,357	4.4	2,651	18.46	0.032	337.93	0.000, ***
C5	12,889	3.95	2,476	19.21	0.035	396.54	0.000, ***
D1	11,343	3.48	2,390	21.07	0.043	603.22	0.000, ***
D2	9,996	3.07	2,213	22.14	0.046	686.17	0.000, ***
D3	8,529	2.62	1,821	21.35	0.039	482.68	0.000, ***
D4	7,028	2.16	1,601	22.78	0.041	550.12	0.000, ***
D5	5,422	1.66	1,288	23.76	0.04	514.04	0.000, ***
E1	3,684	1.13	959	26.03	0.04	515.97	0.000, ***
E2	3,184	0.98	838	26.32	0.038	465.56	0.000, ***
E3	2,244	0.69	606	27.01	0.033	362.42	0.000, ***
E4	1,664	0.51	455	27.34	0.029	281.44	0.000, ***
E5	1,495	0.46	441	29.5	0.032	336.64	0.000, ***
F1	1,040	0.32	279	26.83	0.022	162.36	0.000, ***
F2	573	0.18	164	28.62	0.019	114.22	0.000, ***
F3	671	0.21	202	30.1	0.022	161.47	0.000, ***
F4	354	0.11	121	34.18	0.02	130.96	0.000, ***
F5	180	0.06	65	36.11	0.016	78.6	0.000, ***
G1	108	0.03	39	36.11	0.012	46.38	0.000, ***
G2	67	0.02	20	29.85	0.007	14.36	0.000, ***
G3	39	0.01	13	33.33	0.006	11.78	0.001, ***
G4	14	0	7	50	0.007	13.23	0.000, ***
G5	7	0	2	28.57	0.002	0.39	0.530,
Employment length							
0	18,130	5.56	3,576	19.72	0.045	671.63	0.000, ***
< 1 year	25,621	7.86	3647	14.23	0.008	18.36	0.000, ***
1 year	21,133	6.48	2,864	13.55	0.002	0.72	0.397,
2 years	29,435	9.03	3,938	13.38	0	0.01	0.921,
3 years	26,125	8.02	3,517	13.46	0.001	0.26	0.613,
4 years	18,979	5.82	2,537	13.37	0	0	0.978,
5 years	20,684	6.35	2,731	13.2	-0.001	0.44	0.506,
6 years	17,463	5.36	2,372	13.58	0.002	0.78	0.376,
7 years	18,083	5.55	2,389	13.21	-0.001	0.34	0.558,

Continued on next page

Variable	N	%	Defaults	%	Phi	χ^2	P-value
8 years	16,146	4.95	2,174	13.46	0.001	0.16	0.692,
9 years	12,419	3.81	1,673	13.47	0.001	0.13	0.716,
10+ years	101,723	31.21	12,122	11.92	-0.029	265.33	0.000, ***
Home ownership status							
Mortgage	154,686	47.46	16,942	10.95	-0.067	1471.8	0.000, ***
Other	75	0.02	13	17.33	0.002	0.71	0.400,
Own	31,693	9.72	4,426	13.97	0.006	11.12	0.001, ***
Rent	139,487	42.8	22,159	15.89	0.064	1345.84	0.000, ***
Verification status							
Not verified	122,719	37.65	14,499	11.81	-0.035	404.91	0.000, ***
Source verified	103,022	31.61	14,312	13.89	0.011	37.04	0.000, ***
Verified	100,200	30.74	14,729	14.7	0.026	224.74	0.000, ***
Purpose							
car	3,045	0.93	331	10.87	-0.007	16.22	0.000, ***
credit_card	81,206	24.91	9,195	11.32	-0.034	386.81	0.000, ***
debt_consolidation	190,357	58.4	26,460	13.9	0.019	116.02	0.000, ***
home_improvement	16,719	5.13	1,959	11.72	-0.011	40.86	0.000, ***
house	1,242	0.38	199	16.02	0.005	7.42	0.006, **
major_purchase	5,806	1.78	730	12.57	-0.003	3.08	0.079, .
medical	3,302	1.01	556	16.84	0.01	34.6	0.000, ***
moving	2,121	0.65	410	19.33	0.014	65.27	0.000, ***
other	15,888	4.87	2,509	15.79	0.016	85.24	0.000, ***
renewable_energy	208	0.06	32	15.38	0.002	0.57	0.449,
small_business	3,354	1.03	764	22.78	0.028	259.03	0.000, ***
vacation	1,976	0.61	296	14.98	0.004	4.38	0.036, *
wedding	717	0.22	99	13.81	0.001	0.09	0.765,
Months since last delinquency							
< 1 year	27,494	8.44	3,963	14.41	0.009	28.82	0.000, ***
1-2 years	34,465	10.57	4,637	13.45	0.001	0.3	0.585,
2-3 years	29,044	8.91	3,857	13.28	-0.001	0.16	0.687,
3-4 years	25,124	7.71	3,225	12.84	-0.004	6.36	0.012, *
> 4 years	43,049	13.21	6,021	13.99	0.007	16.85	0.000, ***
Never	166,765	51.16	21,837	13.09	-0.008	20.48	0.000, ***
Months since last record							
< 1 year	823	0.25	116	14.09	0.001	0.33	0.568,
1-2 years	1,609	0.49	231	14.36	0.002	1.31	0.253,
2-3 years	2,696	0.83	374	13.87	0.001	0.58	0.448,
3-4 years	4,657	1.43	637	13.68	0.001	0.39	0.532,
> 4 years	39,362	12.08	6,027	15.31	0.021	147.42	0.000, ***
Never	276,794	84.92	36,155	13.06	-0.021	138.97	0.000, ***
Months since last major derogatory							
< 1 year	6,656	2.04	982	14.75	0.006	11.31	0.001, ***
1-2 years	12,474	3.83	1,755	14.07	0.004	5.6	0.018, *
2-3 years	14,302	4.39	1,957	13.68	0.002	1.34	0.248,
3-4 years	15,335	4.7	2,085	13.6	0.002	0.77	0.381,
> 4 years	32,967	10.11	5,048	15.31	0.019	120.81	0.000, ***
Never	244,207	74.92	31,713	12.99	-0.019	116.39	0.000, ***
Months since recent Bankcard							
< 1 year	149,464	45.86	22,334	14.94	0.043	598.55	0.000, ***
1-2 years	75,245	23.09	10,067	13.38	0	0.03	0.854,
2-3 years	35,761	10.97	4,374	12.23	-0.012	43.98	0.000, ***
3-4 years	17,982	5.52	2,049	11.39	-0.014	63.22	0.000, ***

Continued on next page

Variable	N	%	Defaults	%	Phi	χ^2	P-value
$>$ 4 years	47,487	14.57	4,715	9.93	-0.042	564.42	0.000, ***
Never	2	0	1	50	0.003	0.23	0.628,
Months since recent Bankcard delinquency							
$<$ 1 year	10,375	3.18	1,485	14.31	0.005	8.36	0.004, **
1-2 years	13,936	4.28	1,973	14.16	0.005	7.96	0.005, **
2-3 years	13,826	4.24	1,872	13.54	0.001	0.39	0.530,
3-4 years	14,701	4.51	1,878	12.77	-0.004	4.48	0.034, *
$>$ 4 years	30,136	9.25	4,354	14.45	0.01	33.96	0.000, ***
Never	242,967	74.54	31,978	13.16	-0.01	31.87	0.000, ***
Months since recent revolving delinquency							
$<$ 1 year	19,053	5.85	2,706	14.2	0.006	12.38	0.000, ***
1-2 years	22,460	6.89	3,045	13.56	0.002	0.81	0.369,
2-3 years	19,934	6.12	2,666	13.37	0	0	0.954,
3-4 years	18,655	5.72	2,371	12.71	-0.005	7.13	0.008, **
$>$ 4 years	32,123	9.86	4,572	14.23	0.008	23.46	0.000, ***
Never	213,716	65.57	28,180	13.19	-0.007	15.92	0.000, ***

Phi is the correlation coefficients with loan status factor, χ^2 = Chi-squared critical value, P = P-value.

Table 36: Descriptive statistics for contiunous variables

Variable	Default, N=43,540			Non-Default, N = 282,401			Wilcoxon	
	Mean	Med	StDev	Mean	Med	StDev	p	Sign
Borrower assessment								
int rate	14.38	14.16	3.82	12.3	12.12	3.87	0	***
Borrower characteristics								
annual inc	63,282.5	53,000	68,876.02	73,383.75	62,000	56,285.01	0	***
Borrower indebtedness								
dti	19.11	18.91	8.21	17.25	16.73	7.97	0	***
Credit history								
acc open past 24mths	4.82	4	3.05	4.11	4	2.78	0	***
chargeoff within 12 mths	0.01	0	0.11	0.01	0	0.11	0.55525	***
collections 12 mths ex med	0.01	0	0.13	0.01	0	0.13	0	
delinq 2yrs	0.33	0	0.88	0.3	0	0.83	0	***
inq last 6mths	0.86	1	1.06	0.7	0	0.97	0	***
mo sin old rev tl op	166.92	150	92.09	181.29	163	92.68	0	***
mo sin rcnt rev tl op	10.99	7	13.68	13.56	8	16.27	0	***
mo sin rcnt tl	7.12	5	7.85	8.6	6	9.45	0	***
num accts ever 120 pd	0.5	0	1.24	0.46	0	1.21	0	***
num tl 90g dpd 24m	0.09	0	0.46	0.08	0	0.44	0.0001	***
num tl op past 12m	2.25	2	1.75	1.88	2	1.58	0	***
pct tl nvr dlq	94.55	100	8.11	94.7	100	8.06	0.00005	***
pub rec	0.22	0	0.61	0.19	0	0.57	0	***
pub rec bankruptcies	0.14	0	0.38	0.12	0	0.35	0	***
tot coll amt	209.44	0	1778.99	221.83	0	17319.42	0	***
Current financial characteristics								
acc now delinq	0.01	0	0.09	0	0	0.08	0.02549	*
open acc	11.43	11	5.13	11.33	10	5.06	0.00016	***
revol bal	13,869.29	10,193	18,356.32	15,871.71	11,235	21,260.72	0	***
revol util	58.19	59.1	22.47	55.04	55.7	23.17	0	***
avg cur bal	9,457.02	4,578.5	12,206.75	12,993.23	6,691	16,064.46	0	***
bc open to buy	5,909.11	2,466.5	9,899.86	9,032.35	3,859	14,282.85	0	***
bc util	68.15	73.3	25.85	63.66	67.8	26.7	0	***
delinq amnt	11.18	0	623.96	10.42	0	604.9	0.00457	
mort acc	1.32	0	1.9	1.73	1	2.12	0	***
num actv bc tl	3.81	3	2.18	3.68	3	2.08	0	***
num op rev tl	8.39	8	4.25	8.15	7	4.16	0	***
num rev tl bal gt 0	6.07	5	3.19	5.62	5	3	0	***
num tl 120dpd 2m	0	0	0.03	0	0	0.03	0.20108	
num tl 30dpd	0	0	0.07	0	0	0.06	0.00035	***
percent bc gt 75	55.99	57.1	34.29	49.46	50	35.03	0	***
tot hi cred lim	122,136.12	67,123	135,126.17	162,589.85	98,494	172,738.84	0	***
total il high credit limit	33,915.11	25,001	36,694.33	36,685.07	27,124	39,854.91	0	***
Loan characteristics								
installment	420.3	348.56	259.38	424.33	357.72	257.03	0.00002	***
Created variables								
fico average	686.93	682	23.32	697.13	692	30.56	0	***
monthly debt rate	0.09	0.08	0.04	0.08	0.07	0.04	0	***

**** = 0.001, ** = 0.01, * = 0.05, ' ' = > 0.1, significance level, p = p-value for the Wilcoxon test.*

Table 37: Payment history variable descriptions

Variable name	Variable description
Loan id	A unique identifier for the loan
Pbal beg period	Remaining balance owed at beginning of the period
Prncp paid	Principal paid
Int Paid	Interest paid
Fee paid	Fees paid
Due amt	Amount due this period
Received amt	Amount paid this period
Received d	Date payment is receieved
Period end lstat	Loan status at the end of the period
Month	Month of payment
Pbal end period	Principal balance at the end of the period
Mob	Which entry of the loan it is
Co	Is the loan charged off?
Coamt	Amount charged off
Interest Rate	Interest rate on the loan
Issued Date	Issue date of the loan
Monthly contract amt	Monthly installment size
Dti	Debt to income ratio
State	State of residence
Home ownership	Home Ownership status
Monthly income	Borrowers monthly income
Earliest credit line	Earliest recorded credit line
Open credit lines	Number of open credit lines
Total credit lines	Total number of credit lines
Revolving credit balance	Revolving credit balance
Revolving line utilization	Revolving credit line utilization
Inquiries 6M	Number of inquiries past 6 months
Dq 2yrs	Number of delinquencies past 2 years
Months since Dq	Months since last delinquency
Public rec	Number of public records
Months since last rec	Months since last public record
Employment length	Borrowers employment length
Current policy	Current LendingClub assigned loan policy
Grade	LendingClub assigned loan grade
Term	Number of terms of payment
Appl FICO band	FICO range of applicant
Last FICO band	Latest known FICO range of applicant
Vintage	Loan vintage
Pco recovery	Post charge-off recovery
Pco collection fee	Post charge-off recovery fee

The variables contained in the payment history dataset along with their descriptions.

Table 38: Results of random search for hyperparameters - xgboost

eta	max depth	colsample bytree	nrounds	min child weight	gamma	subsample	AUC test mean
0.4323	4	0.7489	346	4.7149	0.0815	0.6971	0.6629
0.7360	2	0.9443	399	7.1615	0.2093	0.7664	0.6746
0.0734	2	0.5190	133	6.4457	0.2589	0.8016	0.6808
0.3263	2	0.8939	485	2.1389	0.2287	0.6425	0.6853
0.3856	4	0.6782	201	3.2506	0.0866	0.5542	0.6702
0.1746	2	0.5805	210	6.2676	0.0281	0.9529	0.6874
0.5456	3	0.5560	209	3.3118	0.0861	0.6182	0.6713
0.5142	4	0.6168	460	4.5040	0.1740	0.8953	0.6571
0.4307	2	0.8779	305	5.1802	0.1145	0.7198	0.6842
0.5593	3	0.9059	290	5.3423	0.2556	0.8982	0.6723
0.2754	2	0.5844	331	1.3957	0.1738	0.7538	0.6875
0.7590	3	0.7248	118	5.7232	0.0217	0.9008	0.6747
0.4426	2	0.8149	242	5.3176	0.0297	0.6047	0.6829
0.6198	4	0.9157	290	9.8845	0.0408	0.9378	0.6569
0.0263	2	0.9423	455	7.1248	0.0497	0.7178	0.6827
0.0365	3	0.6435	340	8.8902	0.2110	0.6573	0.6863
0.6808	2	0.7983	471	9.6934	0.2701	0.9562	0.6759
0.5629	5	0.6749	186	2.2137	0.1243	0.9150	0.6566
0.5501	3	0.5720	496	8.7268	0.2718	0.6150	0.6615
0.5961	3	0.8893	104	6.3387	0.0398	0.7041	0.6796
0.6501	3	0.6160	228	4.2771	0.2120	0.7020	0.6696
0.4631	3	0.8153	445	6.7030	0.1375	0.8413	0.6724
0.1298	2	0.9957	197	7.5478	0.0150	0.5218	0.6868
0.3904	2	0.9947	138	8.9010	0.1437	0.9928	0.6875
0.2826	4	0.5689	483	7.4612	0.0970	0.6887	0.6715
0.0659	4	0.6843	122	0.5405	0.1030	0.7902	0.6857
0.4608	2	0.7179	128	8.2269	0.1170	0.6922	0.6856
0.6742	3	0.8697	142	6.4424	0.1968	0.5095	0.6680
0.4929	3	0.9519	125	3.7907	0.1105	0.7465	0.6815
0.8605	2	0.9388	164	5.5134	0.2783	0.8956	0.6789
0.0600	5	0.7790	151	0.7174	0.1162	0.9656	0.6871
0.2623	3	0.9941	129	5.6680	0.2657	0.5102	0.6857
0.0458	**5**	**0.8111**	**234**	**2.7111**	**0.2997**	**0.6320**	**0.6889**
0.4703	4	0.6044	289	5.6667	0.1451	0.6811	0.6621
0.5236	2	0.6663	177	6.0673	0.2712	0.9922	0.6861
0.2423	**2**	**0.9688**	**260**	**1.2900**	**0.2911**	**0.8865**	**0.6887**

Results from the initial random search. Only settings with a test AUC-score greater than 0.655 are included, and ***bold*** *indicates the two best result.*

A.2 Confusion matricies

Table 39: Confusion Matricies

		PREDICTED	
		Default	**Non-Default**
TRUTH	**Default**	9364	1640
	Non-Default	39692	25658

Confusion matrix for elastic net model with alpha set to 0.25

		PREDICTED	
		Default	**Non-Default**
TRUTH	**Default**	9364	1640
	Non-Default	39817	25533

Confusion matrix for elastic net model with alpha set to 0.5.

		PREDICTED	
		Default	**Non-Default**
TRUTH	**Default**	10419	585
	Non-Default	52473	12877

Confusion matrix for the univariate interest rate model.

		PREDICTED	
		Default	**Non-Default**
TRUTH	**Default**	9356	1648
	Non-Default	39491	25859

Confusion matrix for the 2-level xgboost model.

		PREDICTED	
		Default	**Non-Default**
TRUTH	**Default**	9350	1654
	Non-Default	38509	26841

Confusion matrix for the 5-level xgboost model.

		PREDICTED	
		Default	**Non-Default**
TRUTH	**Default**	9368	1636
	Non-Default	40760	24590

Confusion matrix for the default random forest settings.

		PREDICTED	
		Default	**Non-Default**
TRUTH	**Default**	9335	1669
	Non-Default	39906	25444

Confusion matrix for the small random forest model.

		PREDICTED	
		Default	**Non-Default**
TRUTH	**Default**	9341	1663
	Non-Default	39414	25936

Confusion matrix for the final random forest settings.

Table 40: Combined accuracy metrics - outliers removed

Model	Accuracy	Sensitivity	Specificity	AUC	f1
Elastic net					
Alpha = 0.25	0.45063	0.85066	0.38384	0.68790	0.30702
Alpha = 0.50	0.45344	0.84902	0.38740	0.68675	0.30770
Interest rate	0.41500	0.85548	0.34146	0.66233	0.29499
Xgboost					
Initial tune	0.45929	0.84960	0.39413	0.69431	0.31015
5-level model	0.45533	0.85104	0.38927	0.69457	0.30895
2-level model	0.45272	0.84960	0.38647	0.69159	0.30757
Random forest					
Default model	0.44276	0.85123	0.37457	0.68331	0.30415
Small model	0.44482	0.85085	0.37703	0.68607	0.30484
Final model	0.45301	0.84988	0.38676	0.69021	0.30775
KNN					
Final model	0.44860	0.84990	0.38210	0.67790	0.30485

The accuracy metrics for the different models on the dataset where outliers are removed.

Table 41: McNemar contingency tables

		Elastic net	
		Correct	**Wrong**
Xgboost	**Correct**	31,922	3,525
	Wrong	3,924	36,983

McNemar contingency table - Prediction overlap between xgboost 5-level and elastic net $\alpha = 0.5$ models.

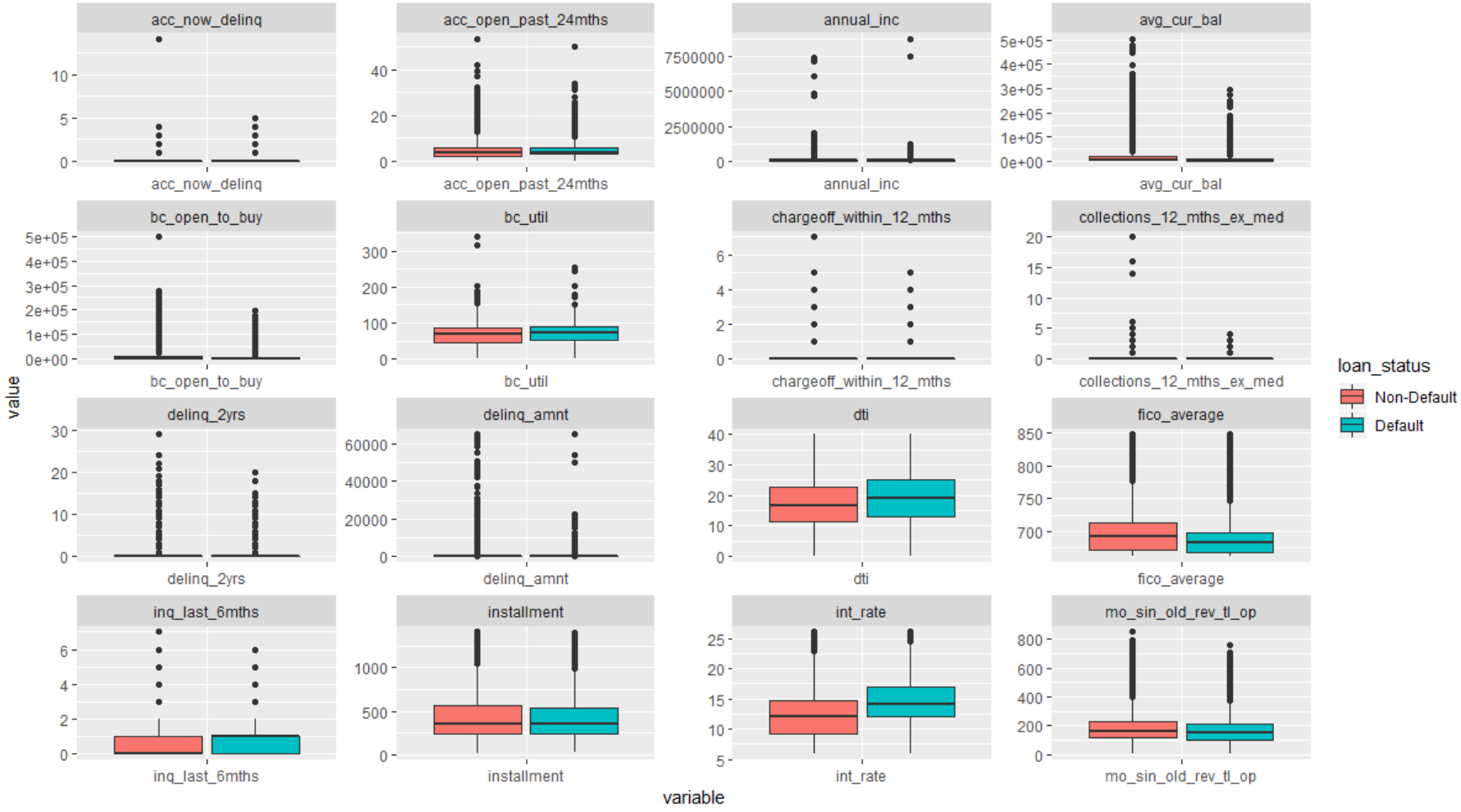
		Elastic net	
		Correct	**Wrong**
Random forest	**Correct**	31,943	3,504
	Wrong	3,182	37,725

McNemar Contingency table - Prediction overlap between random forest final and elastic net $\alpha = 0.5$ models.

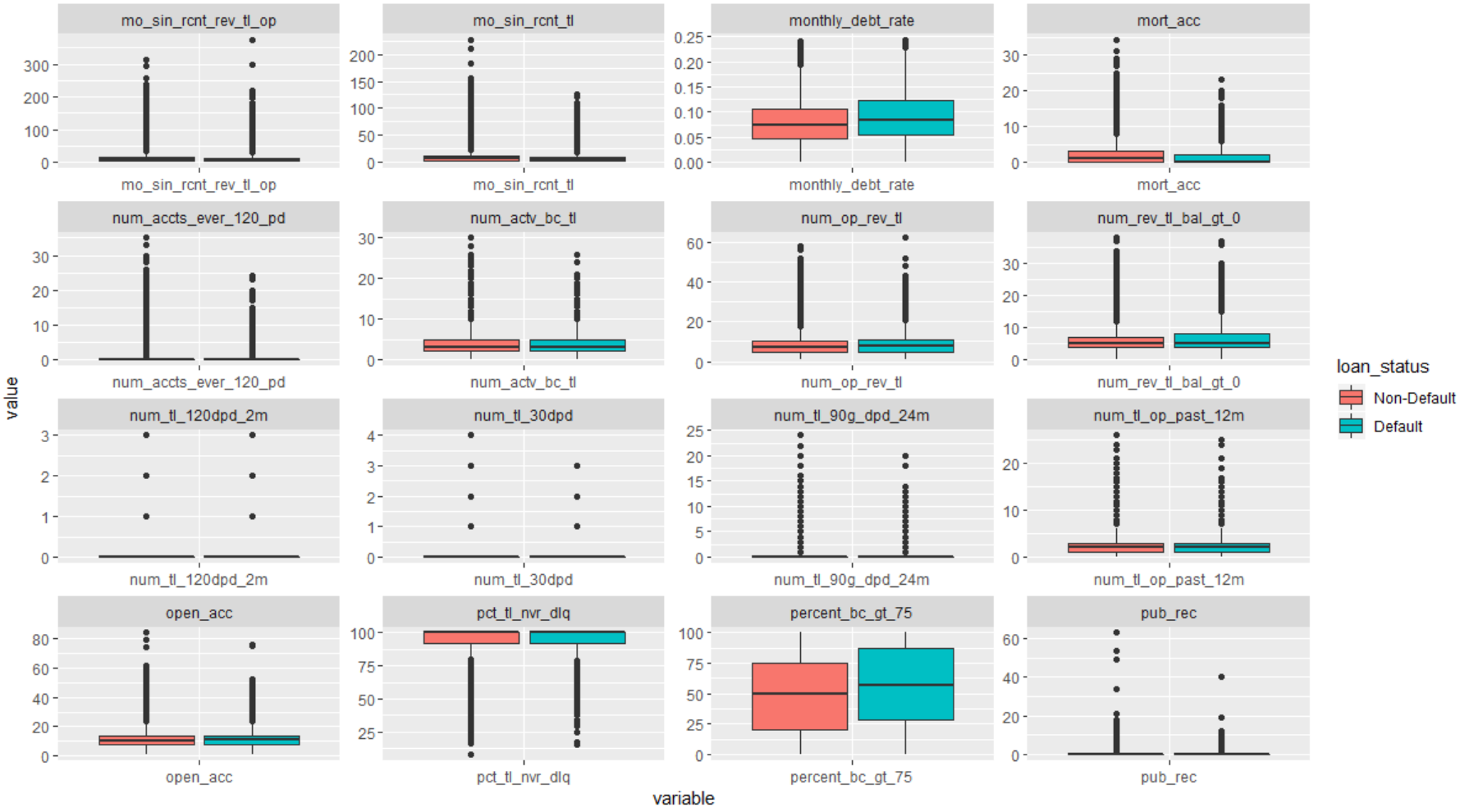
Appendix B

Figure output

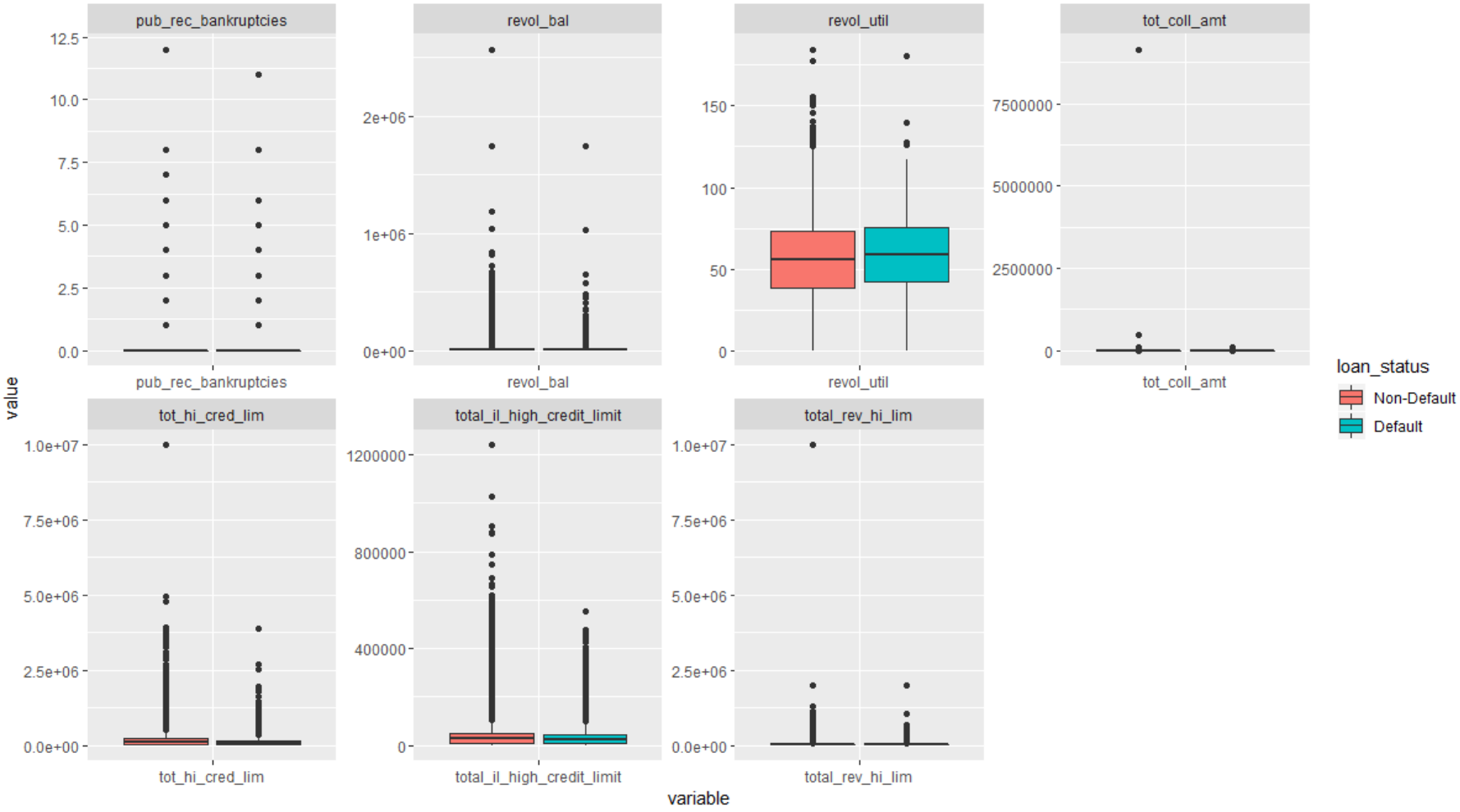
Figure 29: Boxplots for continuous variables



Boxplots for variable 1-16.

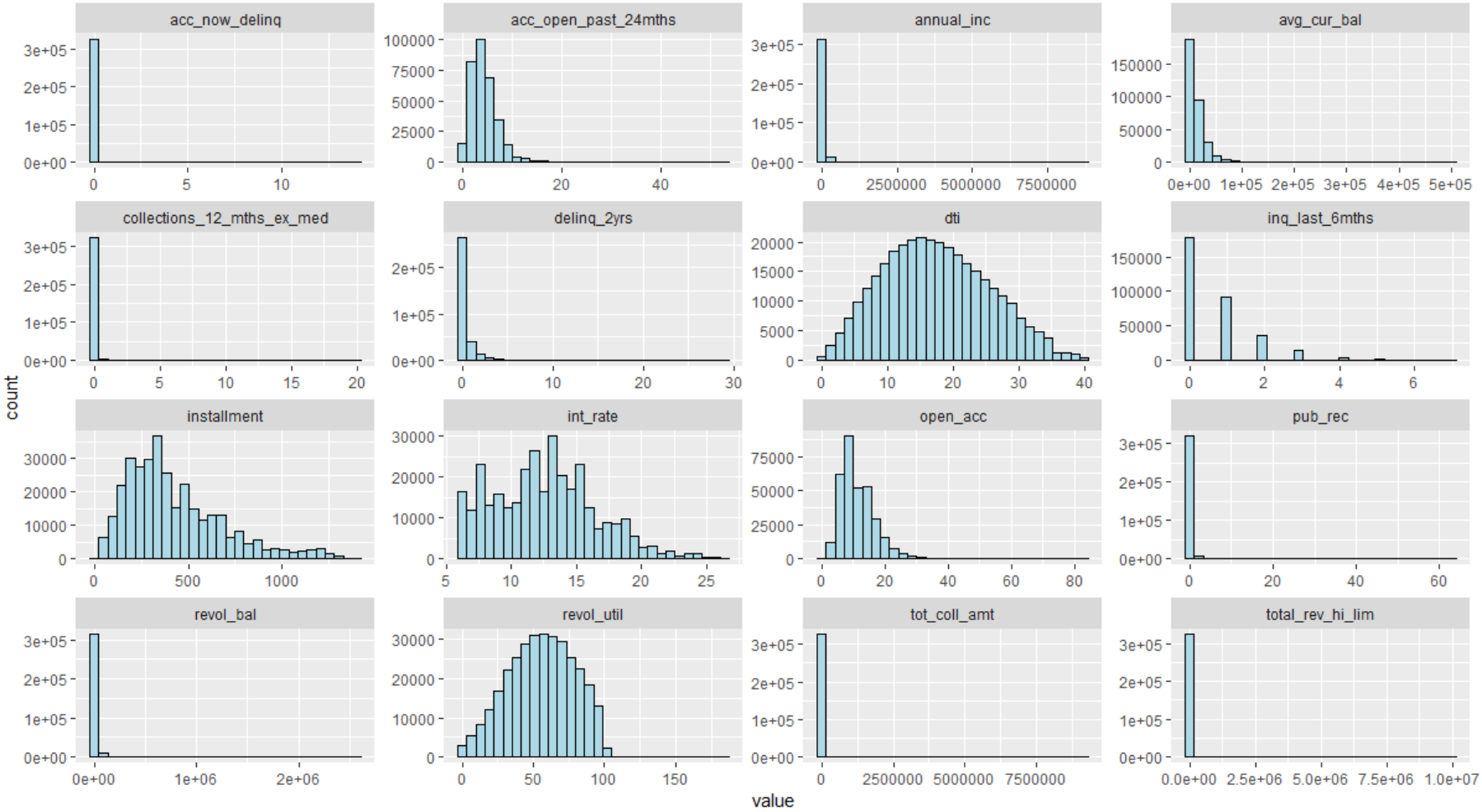


Boxplots for variable 17-32.

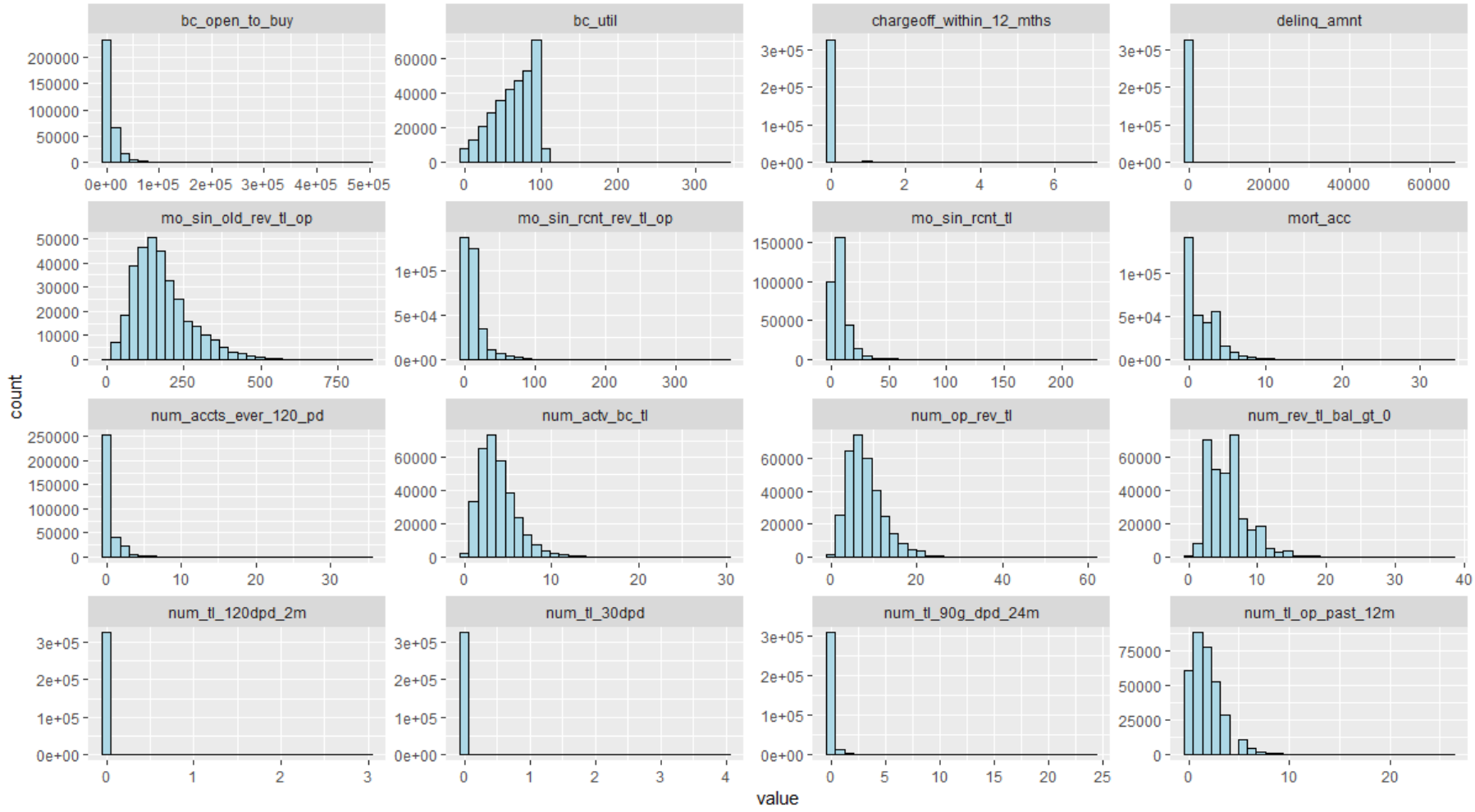


Boxplots for variable 33-39.

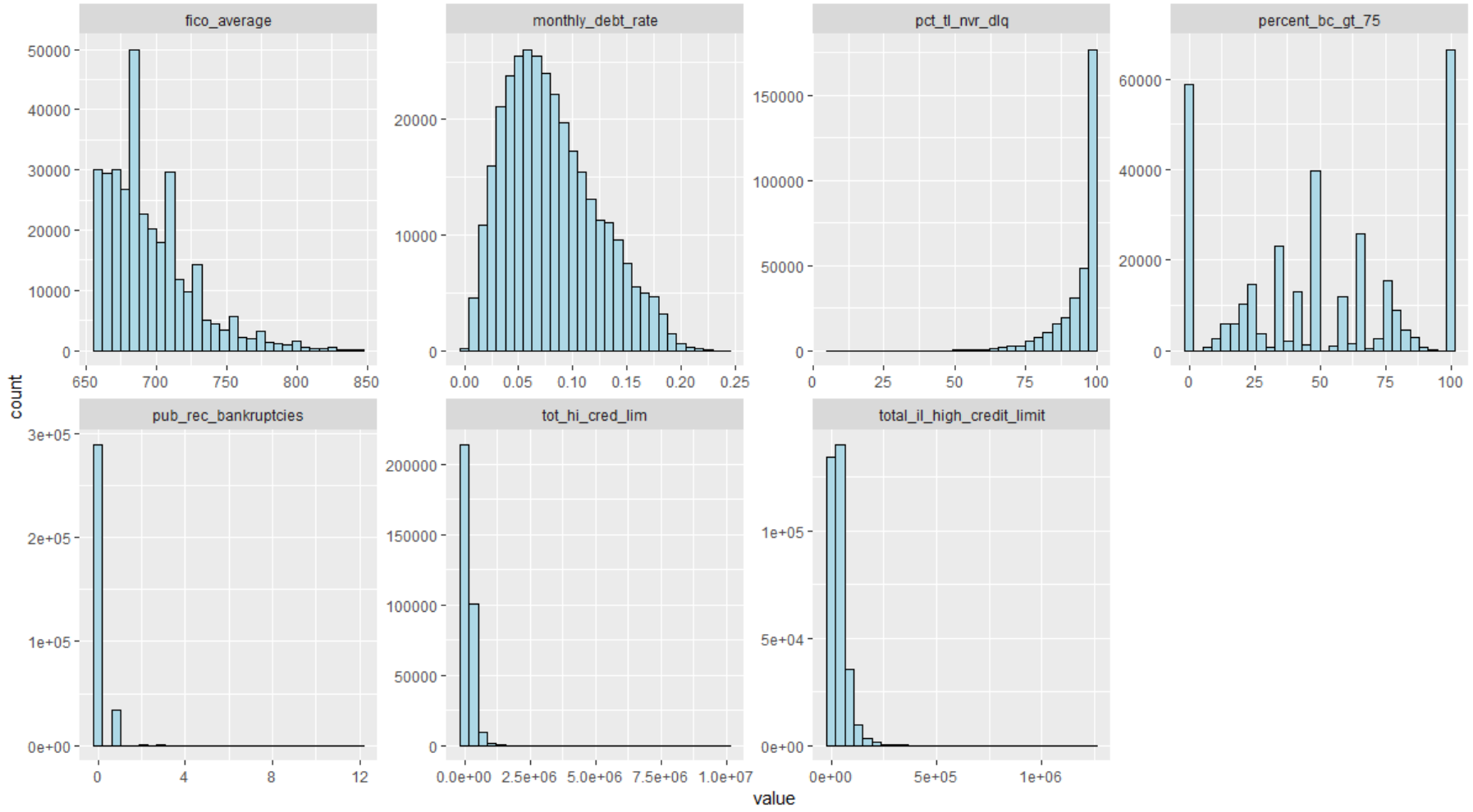
Figure 30: Histograms for continuous variables



Histograms for the continuous variables 1-16.

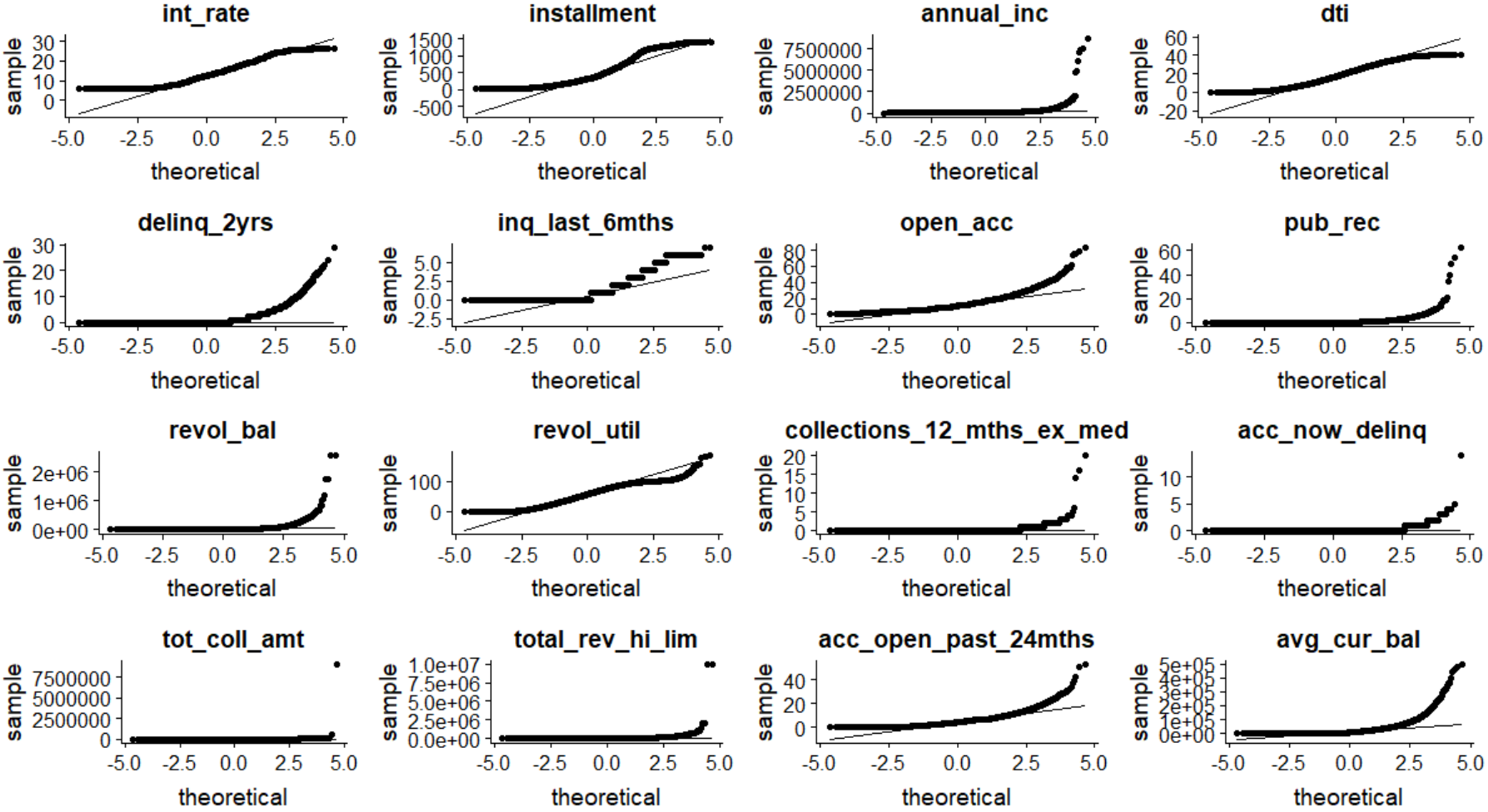


Histograms for the continuous variables 17-32.

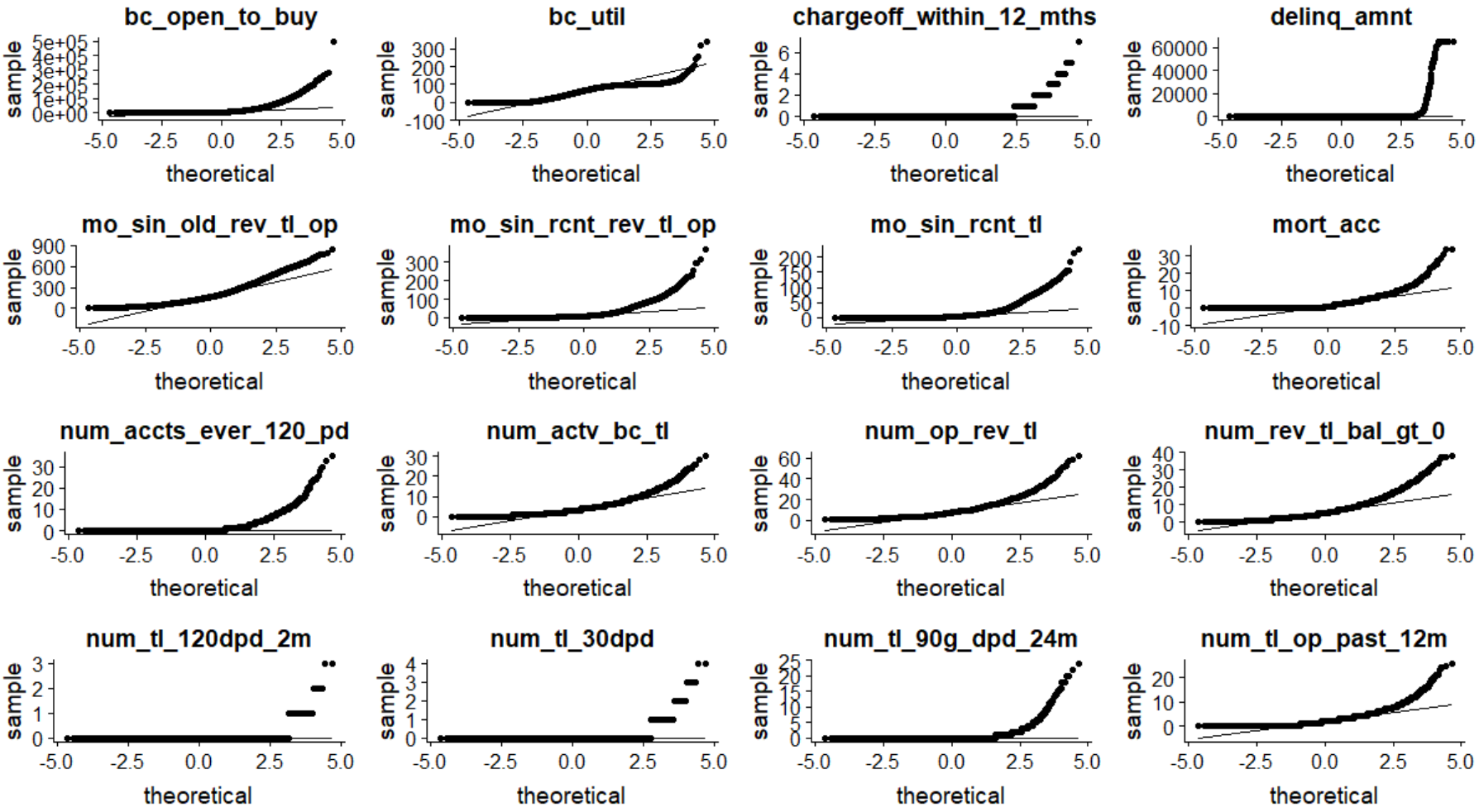


Histograms for the continuous variables 33-39.

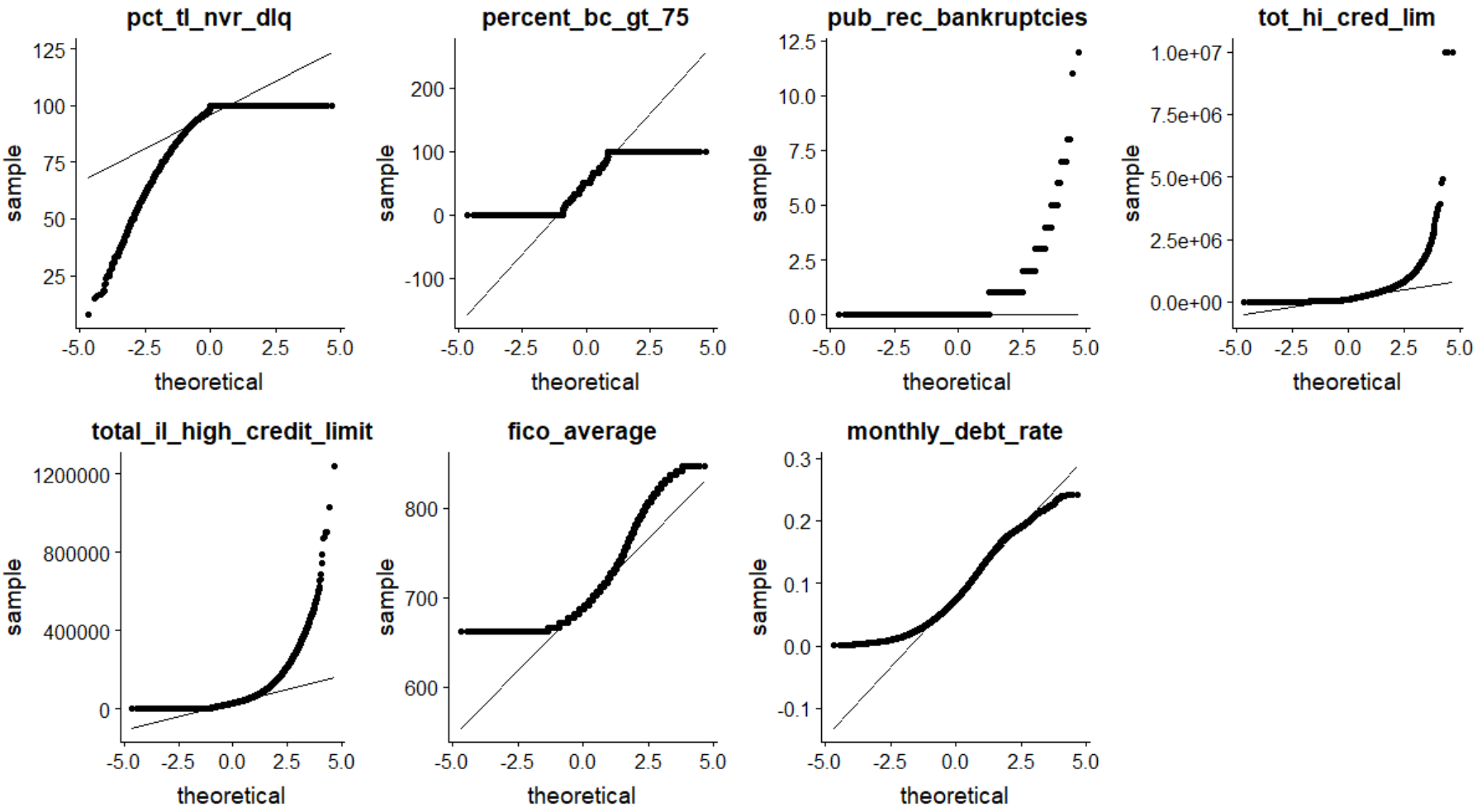
Figure 31: Quantile plots for continuous variables



Quantile plots for the continuous variables 1-16.



Quantile plots for the continuous variables 17-32.



Quantile plots for the continuous variables 33-39.

Appendix C

Use of R

The open source programming language `R` has been a vitally important part of the work in this thesis. It has been used for every stage in the process, including pre-processing, exploratory data analysis, variable modification, model implementation and data visualization.

In this section a description of the two files complementing the thesis is presented.

loan_data.rds is an `R` database object that contains the prepared dataset used in this thesis. It consists of 153 variables and 325,941 observations, which thus holds both the training and validation set used to fit the models.

The second file is *mlr.R*, which shows an example implementation of the xgboost algorithm using the MLR-framework. This includes tuning of the hyperparameters, training of the model and prediction on the holdout data. In addition it shows the cross-validation process for the elastic net models using caret, and the subsequent training and predictions on the holdout set done with MLR.

Bibliography

prosper.com. https://www.prosper.com, a. [Online; accessed 17-Nov-2018].

Smava. https://www.smava.de, b. [Online; accessed 17-Nov-2018].

Kiva. https://www.kiva.org, c. [Online; accessed 17-Nov-2018].

Trine. https://www.jointrine.com, d. [Online; accessed 17-Nov-2018].

Lending club. https://www.lendingclub.com, e. [Online; accessed 17-Nov-2018].

Lending club loan-data dataset download location. https://www.lendingclub.com/info/download-data.action, f. [Online; accessed 17-Nov-2018].

Lending club payment history dataset download location. https://www.lendingclub.com/company/additional-statistics, g. [Online; accessed 17-Nov-2018, (requires account)].

Grade calculation. https://www.lendingclub.com/foliofn/rateDetail.action, h. [Online; accessed 17-Nov-2018].

Fico. https://www.fico.com/en/products/fico-score, i. [Online; accessed 17-Nov-2018].

Zopa. https://www.zopa.com, j. [Online; accessed 17-Nov-2018].

A. Bachmann, A. Becker, D. Buerckner, M. Hilker, F. Kock, M. Lehmann, and P. Tiburtus. Online peer-to-peer-lending - a literature review. 2011.

Bernd Bischl, Michel Lang, Lars Kotthoff, Julia Schiffner, Jakob Richter, Erich Studerus, Giuseppe Casalicchio, and Zachary M. Jones. mlr: Machine learning in r. *Journal of Machine Learning Research*, 17(170):1–5, 2016. URL http://jmlr.org/papers/v17/15-066.html.

Leo Breiman. Random forests. *Machine Learning*, 45(1):5–32, Oct 2001a. ISSN 1573-0565. doi: 10.1023/A:1010933404324. URL https://doi.org/10.1023/A:1010933404324.

Leo Breiman. Random forests. *Machine learning*, 45(1):5–32, 2001b.

Ajay Byanjankar, Markku Heikkilä, and Jozsef Mezei. Predicting credit risk in peer-to-peer lending: A neural network approach. In *Computational Intelligence, 2015 IEEE Symposium Series on*, pages 719–725. IEEE, 2015.

Tianqi Chen and Carlos Guestrin. Xgboost: A scalable tree boosting system. *CoRR*, abs/1603.02754, 2016. URL http://arxiv.org/abs/1603.02754.

Thomas Cover and Peter Hart. Nearest neighbor pattern classification. *IEEE transactions on information theory*, 13(1):21–27, 1967.

Annette J. Dobson and Adrian G. Barnett. *An Introduction to Generalized Linear Models, Third Edition*. Texts in Statistical Science. Chapman & Hall/CRC Press, Boca Raton, FL, 2008. URL `https://eprints.qut.edu.au/15448/`. For more information about this book please refer to the publisher's website (see link) or contact the author.

Sahibsingh A Dudani. The distance-weighted k-nearest-neighbor rule. *IEEE Transactions on Systems, Man, and Cybernetics*, (4):325–327, 1976.

Riza Emekter, Yanbin Tu, Benjamas Jirasakuldech, and Min Lu. Evaluating credit risk and loan performance in online peer-to-peer (p2p) lending. *Applied Economics*, 47 (1):54–70, 2015. doi: 10.1080/00036846.2014.962222. URL `https://doi.org/10.1080/00036846.2014.962222`.

Ludwig Fahrmeir, Thomas Kneib, Stefan Lang, and Brian Marx. *Introduction*, pages 1–19. Springer Berlin Heidelberg, Berlin, Heidelberg, 2013. ISBN 978-3-642-34333-9. doi: 10.1007/978-3-642-34333-9_1. URL `https://doi.org/10.1007/978-3-642-34333-9_1`.

Manuel Fernández-Delgado, Eva Cernadas, Senén Barro, and Dinani Amorim. Do we need hundreds of classifiers to solve real world classification problems? *The Journal of Machine Learning Research*, 15(1):3133–3181, 2014.

Evelyn Fix and Joseph L Hodges Jr. Discriminatory analysis-nonparametric discrimination: consistency properties. Technical report, California Univ Berkeley, 1951.

Yoav Freund and Llew Mason. The alternating decision tree learning algorithm. In *icml*, volume 99, pages 124–133, 1999.

David J Hand and William E Henley. Statistical classification methods in consumer credit scoring: a review. *Journal of the Royal Statistical Society: Series A (Statistics in Society)*, 160(3):523–541, 1997.

Trevor Hastie and Junyang Qian. glmnet vignette, 2014. URL `https://web.stanford.edu/~hastie/glmnet/glmnet_alpha.html`.

Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *Random Forests*, pages 587–604. Springer New York, New York, NY, 2009. ISBN 978-0-387-84858-7. doi: 10.1007/978-0-387-84858-7_15. URL `https://doi.org/10.1007/978-0-387-84858-7_15`.

Klaus Hechenbichler and Klaus Schliep. Weighted k-nearest-neighbor techniques and ordinal classification. 2004.

Tin Kam Ho. Random decision forests. In *Proceedings of 3rd International Conference on Document Analysis and Recognition*, volume 1, pages 278–282 vol.1, Aug 1995. doi: 10.1109/ICDAR.1995.598994.

Arthur E. Hoerl and Robert W. Kennard. Ridge regression: Biased estimation for nonorthogonal problems. *Technometrics*, 12(1):55–67, 1970. ISSN 00401706. URL `http://www.jstor.org/stable/1267351`.

Gareth James, Daniela Witten, Trevor Hastie, and Robert Tibshirani. *Tree-Based Methods*, pages 303–335. Springer New York, New York, NY, 2013. ISBN 978-1-4614-7138-7. doi: 10.1007/978-1-4614-7138-7_8. URL https://doi.org/10.1007/978-1-4614-7138-7_8.

Silke Janitza, Harald Binder, and Anne-Laure Boulesteix. Pitfalls of hypothesis tests and model selection on bootstrap samples: causes and consequences in biometrical applications. *Biometrical Journal*, 58(3):447–473, 2016.

O. Maurice Joy and John O. Tollefson. On the financial applications of discriminant analysis. *Journal of Financial and Quantitative Analysis*, 10(5):723–739, Dec 1975. doi: 10.2307/2330267.

Michael Kearns and Leslie Valiant. Cryptographic limitations on learning boolean formulae and finite automata. *Journal of the ACM (JACM)*, 41(1):67–95, 1994.

Michael Klafft. Online peer-to-peer lending: A lenders' perspective. In *Proceedings of the International Conference on E-Learning, E-Business, Enterprise Information Systems, and E-Government, EEE 2008*, pages 371–375. CSREA Press, 2008.

Cheng Li. A gentle introduction to gradient boosting. URL http://www.chengli.io/tutorials/gradient_boosting.pdf.

Wei Li, Shuai Ding, Yi Chen, and Shanlin Yang. Heterogeneous ensemble for default prediction of peer-to-peer lending in china. *IEEE Access*, 2018.

Milad Malekipirbazari and Vural Aksakalli. Risk assessment in social lending via random forests. *Expert Systems with Applications*, 42(10):4621–4631, 2015.

Gonzalo Martínez-Muñoz and Alberto Suárez. Out-of-bag estimation of the optimal sample size in bagging. *Pattern Recognition*, 43(1):143–152, 2010.

Andreas Mayr, Harald Binder, Olaf Gefeller, and Matthias Schmid. The evolution of boosting algorithms-from machine learning to statistical modelling. *arXiv preprint arXiv:1403.1452*, 2014.

Quinn McNemar. Note on the sampling error of the difference between correlated proportions or percentages. *Psychometrika*, 12(2):153–157, 1947.

J. A. Nelder and R. W. M. Wedderburn. Generalized linear models. *Journal of the Royal Statistical Society. Series A (General)*, 135(3):370–384, 1972. ISSN 00359238. URL http://www.jstor.org/stable/2344614.

Philipp Probst and Anne-Laure Boulesteix. To tune or not to tune the number of trees in random forest? *arXiv preprint arXiv:1705.05654*, 2017.

Philipp Probst, Bernd Bischl, and Anne-Laure Boulesteix. Tunability: Importance of hyperparameters of machine learning algorithms. *arXiv preprint arXiv:1802.09596*, 2018a.

Philipp Probst, Marvin Wright, and Anne-Laure Boulesteix. Hyperparameters and tuning strategies for random forest. *arXiv preprint arXiv:1804.03515*, 2018b.

Robert E. Schapire. The strength of weak learnability. *Machine Learning*, 5:197–227, 1990. doi: 10.1007/BF00116037. URL https://doi.org/10.1007/BF00116037.

Robert E. Schapire. *The Boosting Approach to Machine Learning: An Overview*, pages 149–171. Springer New York, New York, NY, 2003. ISBN 978-0-387-21579-2. doi: 10.1007/978-0-387-21579-2_9. URL https://doi.org/10.1007/978-0-387-21579-2_9.

Carlos Serrano-Cinca and Begoña Gutiérrez-Nieto. The use of profit scoring as an alternative to credit scoring systems in peer-to-peer (p2p) lending. pages 113–122, Sep 2016.

Carlos Serrano-Cinca, Begona Gutierrez-Nieto, and Luz López-Palacios. Determinants of default in p2p lending. *PloS one*, 10(10):e0139427, 2015.

Fintechnews Singapore. Peer-to-peer lending in china, hong kong and southeast asia. http://fintechnews.sg/9584/crowdfunding/peer-peer-lending-china-hong-kong-southeast-asia/, 2017. [Online; accessed 17-Nov-2018].

Robert Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society. Series B (Methodological)*, 58(1):267–288, 1996. ISSN 00359246. URL http://www.jstor.org/stable/2346178.

Hui Wang, Martina Greiner, and Jay E. Aronson. People-to-people lending: The emerging e-commerce transformation of a financial market. In *Value Creation in E-Business Management*, pages 182–195. Springer Berlin Heidelberg, 2009.

Hui Zou and Trevor Hastie. Regularization and variable selection via the elastic net. *Journal of the Royal Statistical Society. Series B (Statistical Methodology)*, 67(2):301–320, 2005. ISSN 13697412, 14679868. URL http://www.jstor.org/stable/3647580.