Improving Credit Classification Using Machine Learning Techniques

by
Adam Gregory Carl Lazure

A Thesis
presented to
The University of Guelph

In partial fulfillment of requirements
for the degree of
Master of Science
in
Mathematics and Statistics

Guelph, Ontario, Canada

ABSTRACT

IMPROVING CREDIT CLASSIFICATION USING MACHINE LEARNING TECHNIQUES

Adam Lazure
University of Guelph, 2017

Advisors:
Dr. Peter T. Kim and Dr. Zeny Feng

The quantification of credit risk is an ever expanding topic of discussion in the field of finance. In order to prevent economic loss, risk management is necessary. A popular method of risk management is the use of statistical techniques in conjunction with machine learning. This thesis takes a unique machine learning approach to credit classification. In particular, it conducts a missing information simulation study on German credit data and makes use of the random forest (RF), support vector machine (SVM), multiple imputation by chained equations (MICE) and predictive mean matching (PMM) methodologies. Results give indication that using MICE in tandem with PMM can be an optimal method of imputation within the context of credit risk data.

ACKNOWLEDGEMENTS

I would like to begin by thanking my advisors Dr. Peter T. Kim and Dr. Zeny Feng for their guidance and patience. In addition, I'd like to especially thank my parents for their invaluable support throughout my time at university. Finally, thanks goes out to the mathematics and statistics faculty as a whole, who in one way or another assisted me in my education.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Motivation

Managing risk is a constantly expanding area in the field of finance. In order to prevent economic loss, risk management is necessary. It is particularly vital for large corporations and investors where millions, or potentially billions, of dollars are at stake. An example of where the level of risk management was not adequate was the financial crisis of 2007-2008. One factor strongly associated with the crisis was the U.S. housing bubble (Holt, 2009). Credit during this period was assigned to "subprime borrowers"; individuals who had low income and or assets. Proceeding this, default rates on subprime and adjustable-rate mortgages began to rise quickly (Holt, 2009). Subprime lenders began filing for bankruptcy as more borrowers could not meet payments. It was discovered by Demyanyk and Hemert (2011) that, for six consecutive years prior to the crisis, the quality of subprime loans had been deteriorating. Estimates of the losses incurred by the housing bubble number in the trillions of dollars (Baker, 2008). A large portion of the responsibility of the housing crisis lies with poor credit risk management. Credit agencies determined mortgage backed securities based on historical mortgage default frequencies for similar pools and not by isolated mortgage quality (Holt, 2009). Said determinations were based on the assumption that housing prices would continue to rise. In contrast, when housing prices fell, the result was what is known as the "burst" of the housing bubble (Holt, 2009). The fall in housing prices translated to negative home equity for many, with homeowners not being able to make payments on upwardly adjusted monthly payments. Defaults rose dramatically and economic loss followed suite Holt (2009). Although credit risk management was not the only factor in the burst, improved management

would have aided in preventing loss.

Throughout recent history, the outstanding debt attributed to consumer credit has been continuously rising. Figure 1.1 captures the outstanding United States consumer credit debt from 2000 to 2017 sourced from the US Federal Reserve (2017).

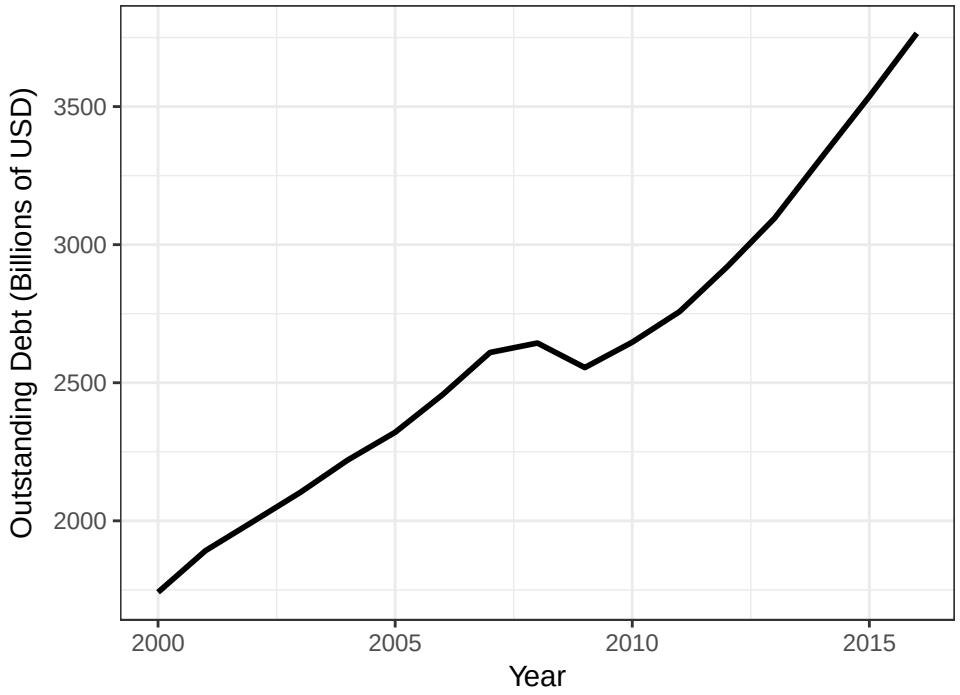


Figure 1.1 – Outstanding United States consumer credit debt in billions of USD by year (US Federal Reserve, 2017).

With the constant rise of consumer credit debt, it is clear that proper risk management is essential to prevent extreme loss. Without appropriate infrastructure in place, the potential financial loss increases by the year. This thesis takes motivation from these issues and looks to improve upon credit risk management through the use of machine learning techniques.

1.2 Credit Modeling and Objective

The use of mathematics to quantify credit risk is a relatively new topic of discussion. It can be traced back to as early as "The Pricing of Options and Corporate Liabilities" (Black and Scholes,

1973). Here the authors discuss option pricing and the use of theoretical valuation formulas. This was shortly followed by "On the Pricing of Corporate Debt: The Risk Structure of Interests Rates" (Merton, 1974). His work focused on credit defaults and yielded the popular Merton default risk model. Some time after these publications, the first international bank regulations were implemented in 1988 with the introduction of Basel I (Dionne, 2013). This accord focused solely on credit risk and maintained that each bank was to set a capital reserve of 8% of the value of securities representing credit risk in its portfolio (Dionne, 2013). The effectiveness of this accord was somewhat debated due to considerations of market risk not being considered. A reform of Basel I took place in 1996 which took market risk into consideration and more importantly allowed the use of internal market risk models (Dionne, 2013). This was followed by two more reforms, Basel II and Basel III, which involved credit risk and new capital rules to protect banks respectively (Dionne, 2013).

Since Black and Scholes (1973), many different approaches have been taken to quantify credit and market risk through machine learning. Machine learning is a branch of statistics that uses programming techniques in order to train statistical models on various types of data. Credit modeling itself is relatively straight forward; a statistical model is used to quantify the creditability of the borrower based on socioeconomic factors. These factors can include measures such as net assets, income and past default history (Dionne, 2013). The data involved in this thesis is German banking data that contains many of these variables. It revolves around a binary classification problem that is set to trend whether or not a borrower is credible or non-credible (Lichman, 2013)(Hofmann, 1994). Further information regarding this data is discussed in chapter 2.

One of the machine learning techniques this work employs is known as a random forest (RF) and was suggested by Breiman (2001). It is an extension of one of his previous works called bagging and revolves around the aggregation of classification trees. Random forests have been used extensively in the context of credit risk data with high degrees of success (Brown and Mues, 2012)(Wang et al., 2011)(Lessmann et al., 2015). Particular conclusions of these publications can be found in section 3.5. Random forests are capable of handling high-dimensional data and are nonparametric. These factors, as well as the results of other publications, make random forests an attractive option for this topic.

Another machine learning technique this thesis involves was developed by Cortes and Vapnik (1995) and is known as the support vector machine (SVM), or alternatively, the support vector network. SVMs use boundaries in order to perform binary classification. They have the ability to deal with complex predictor-response relationships and still maintain high accuracy (Cortes and

Vapnik, 1995). Since the data involves a binary classification problem, the SVM serves as a natural choice for analysis. The conclusions of other works expressed in section 4.6 further support this assertion.

A topic that is not discussed as frequently in the context of credit modeling is the presence of missing data. Missing data is capable of causing many issues in the machine learning process. Ignoring this data can result in a loss of otherwise useful information. On the other hand, selecting an improper technique to handle missing data could lead to misleading interpretations (Enders, 2010). This thesis makes use of an imputation technique suggested by Rubin (1987) called multiple imputation (MI). In particular, it employs an extension of this referred to as fully conditional specification, also known as multiple imputation by chained equations (MICE) (van Buuren and Oudshoorn, 1999). MICE is capable of handling complex incomplete data problems that other methods struggle with (van Buuren and Oudshoorn, 1999). There is little existing work done on using MI based procedures on credit data, however, what is available is discussed in section 5.4. This in tandem with the overall strengths of the procedure makes MICE an attractive choice for imputation.

The MICE methodology involves a series of regressions. Each variable that contains missing data is modeled conditionally with regards to the rest. The missing values are then predicted by this model. This means that the conditional distribution of each variable containing missing data must be specified. It is often convenient to use a nonparametric method, as each variable can be imputed with the same model. This thesis employs a method suggested by Rubin (1986) called predictive mean matching (PMM). PMM is semiparametric and is capable of handling numerical, ordinal and nominal data. Since the German credit data contains variables of differing types, PMM is an ideal solution for imputation.

The objective of this thesis is to test whether or not MICE in conjunction with PMM can serve as an optimal form of imputation in the credit risk setting. To answer this question, a missing information simulation is conducted. SVMs and RFs are trained on the data in a setting where no missing values are present. This is called the "pre-simulation". Missing information is then introduced into the data and subsequently imputed by MICE in tandem with PMM. SVMs and RFs are then retrained on the new data. This is the "post-simulation". The validity of imputations are assessed and model performances are compared. In order for MICE in combination with PMM to be optimal, imputations should be valid and model performances should have little change.

1.3 Overview

- Chapter 2 discusses the data involved in this thesis.
- Chapter 3 derives RFs and gives a summary of their use in similar works.
- Chapter 4 derives SVMs and gives a summary of their use in similar works.
- Chapter 5 goes over missing data mechanisms, MICE, PMM and the simulation. It then discusses the use of MICE in similar works.
- Chapter 6 details parameter tuning and feature selection.
- Chapter 7 presents the results of the thesis.
- Chapter 8 discusses the results of the work and mentions future topics of consideration.

Chapter 2

Data

In this chapter the German credit data set is explained. A dictionary containing each variable and what it relates to is also provided.

2.1 German Credit Data

This thesis makes use of 1994 German banking data. It revolves around a binary classification problem where individuals are classified as credible or non-credible based on a set of socioeconomic variables (Lichman, 2013)(Hofmann, 1994). The data contains 14 categorical and 7 numerical variables as well as 1000 observations. All observations are taken from the same bank. Corresponding definitions are viewable in section 2.2.

A problematic characteristic of this data is the presence of a class imbalance as illustrated by figure 2.1.

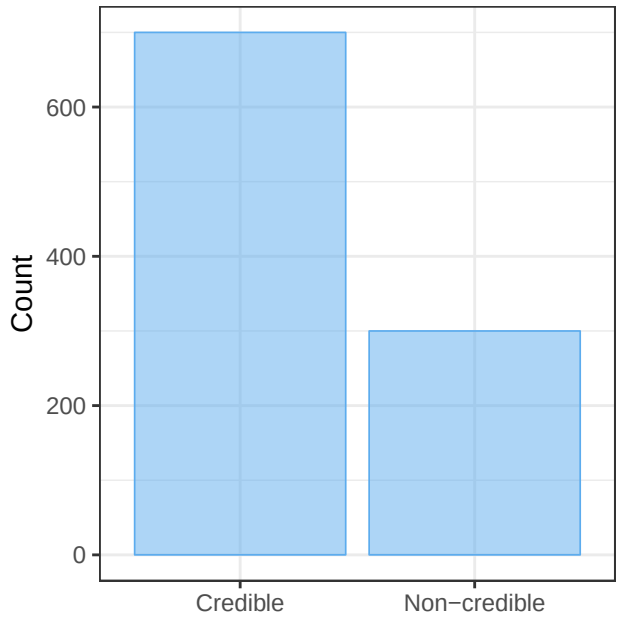


Figure 2.1 – A graph that illustrates the 7:3 class imbalance present in the German credit data.

Class imbalances like this are associated with issues when used alongside training procedures maximized around metrics such as overall accuracy. This is due to their sensitivity towards the class distribution (Fawcett, 2006). A class distribution insensitive metric known as the area under curve (AUC) is used to remedy this issue. Discussion of this subject is left to section 6.1.

2.2 Variable Dictionary

checking_acc_status A categorical variable with 4 levels that describes the status of an individual's existing checking account.

- A11: < 0 DM
- A12: $0 \leq \cdots < 200$ DM
- A13: ≥ 200 DM or salary assignments for at least 1 year
- A14: No checking account

duration_month A numerical variable that gives the duration the checking account has been open in months.

cred_history A categorical variable with 5 levels that describes the credit history of an individual.

- A30: No credits taken or all credits paid back duly
- A31: All credits at this bank paid back duly
- A32: Existing credits paid back duly until now
- A33: Delay in paying off in the past
- A34: Critical account or other credits existing (not at this bank)

purpose A categorical variable with 10 levels that describes the purpose of the loan.

- A40: Car (new)
- A41: Car (used)
- A42: Furniture or equipment
- A43: Radio or television
- A44: Domestic applications
- A45: Repairs
- A46: Education
- A48: Retraining
- A49: Business
- A410: Other

cred_amnt A numerical variable that gives the actual credit an individual has.

savings_acc_bonds A categorical variable with 5 levels that describes the status of an individual's savings account or bonds.

- A61: < 100 DM
- A62: $100 \leq \cdots < 500$ DM
- A63: $500 \leq \cdots < 1000$ DM
- A64: ≥ 1000 DM
- A65: Unknown or no savings account

emplmnt_length A categorical variable with 5 levels detailing the length of present employment.

- A71: Unemployed
- A72: < 1 year
- A73: $1 \leq \cdots < 4$ years
- A74: $4 \leq \cdots < 7$ years
- A75: ≥ 7 years

inst_rate A numerical variable indicating the installment rate in percentage of disposable income.

marital_status_sex A categorical variable with 5 levels describing personal status and sex.

- A91: Male and divorced/separated
- A92: Female and divorced/separated
- A93: Male and single
- A94: Male and married/widowed
- A95: Female and single

debtors_or_guarantors A categorical variable with three levels that involves other debtors or guarantors.

- A101: None
- A102: Co-applicant
- A103: Guarantor

current_residence_dur A numerical variable indicating how long an individual has lived at their current residence.

property A categorical with 4 levels describing the type of property an individual owns.

- A121: Real estate
- A122: Building society savings agreement or life insurance
- A123: A car or other

- A124: Unknown or no property

age_years A numerical variable indicating and individual's age in years.

inst_plans A categorical variable with 3 levels indicating the other installment plants an individual has.

- A141: Bank
- A142: Stores
- A143: None

housing A categorical variable with 3 levels indicating the type of housing an individual has.

- A151: Rent
- A152: Own
- A153: For free

num_exist_credits A numerical variable indicating the number of existing credits at this bank.

job A categorical variable with 4 levels indicating the type of job an individual has.

- A171: Unemployed or unskilled and non-resident
- A172: Unskilled and resident
- A173: Skilled employee and official
- A174: Management, self employed, highly qualified employee or officer

liable_maintenance A numerical variable indicating the number of people an individual is liable to provide maintenance for.

telephone A categorical variable with 2 levels detailing whether or not an individual owns a telephone.

- A191: None
- A192: Yes and registered under the customers name

foreign_worker A categorical variable with 2 levels indicating whether or not an individual is a foreign worker.

- A201: Yes
- A202: No

cred A categorical variable with 2 levels indicating if an individual is credible or non-credible. This is the response.

- cr: Credible
- ncr: Non-credible

Chapter 3

Random Forests

The random forest (RF) methodology is an ensemble learning method suggested by Breiman (2001). It is an extension on one of his earlier works, bootstrap aggregation, otherwise known as bagging (Breiman, 1996). RFs are nonparametric. In addition, they are capable of dealing with high-dimensional data and nonlinear relationships with relative ease.

In this chapter, RFs are derived starting from classification trees. Tree pruning is then explained, followed by the bagging procedure. The extension made from bagging to random forests is then detailed. This is followed by a review of other publications that have used this method in the credit risk field.

3.1 Classification Trees

The goal of the classification trees are to divide the predictor space containing p variables $X_1, X_2, \ldots, X_p$ into m disjoint sets $A_1, A_2, \ldots, A_m$ (Loh, 2011). In order to accomplish this, recursive partitioning, otherwise known as recursive binary splitting, is used (Friedman et al., 2001)(Loh, 2011). It is referred to as a top down approach as it starts at the top of the tree, where all observations begin in a single region known as the root node. The predictor space is then split into two new branches called daughter nodes. This process is repeated until a stopping criteria is met and final regions are determined. These final regions are referred to as terminal nodes (Loh, 2011). Figure 3.1 gives an example appearance of a simplistic tree.

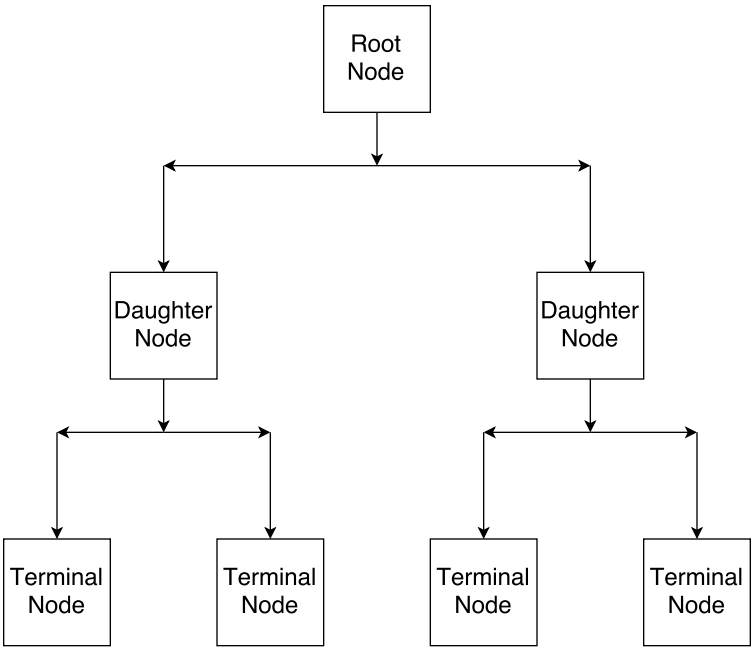


Figure 3.1 – An example appearance of a tree.

In order to conduct splits, a splitting criteria must be defined. Classification trees look to assign an observation in a given node to the most commonly occurring class of training observations in this node. First define the proportion of class k observations in node m as

$$\hat{p}_{mk} = \frac{1}{N_m} \sum_{x_i \in A_m} I(y_i = k), \tag{3.1}$$

where N_m is the number of observations in node m. Observations are then classified in node m to class $k(m) = \text{argmax}_k \hat{p}_{mk}$ which is the most prevalent class in node m (Friedman et al., 2001).

A natural criteria to base splits on is the classification error rate. This is defined as the fraction of training observations in a region that do not belong to the most common class. This is given by

$$\frac{1}{N_m} \sum_{x_i \in A_m} I(y_i \neq k(m)) = 1 - \hat{p}_{mk(m)} \tag{3.2}$$

where $\hat{p}_{mk}(m)$ is the proportion of training observations in the mth region that are from the most prevalent class (Friedman et al., 2001). The classification error rate, however, is generally not sensitive enough for tree-growing. An alternative is known as the Gini index, formulated as

$$\sum_{k=1}^{K} \hat{p}_{mk}(1 - \hat{p}_{mk}) \tag{3.3}$$

which is a measurement of the total variance present amongst the K distinct classes (James et al., 2013)(Friedman et al., 2001). The Gini index ranges between $[0, 1]$. A value of 0 implies that a node contains observations that are all from the same class. In contrast, a value of 1 implies a node contains observations that are all from different classes. For this reason, the Gini index is known as a measure of node purity (James et al., 2013). In order to conduct splits, test splits are conducted. The Gini index is calculated for the two daughter nodes resulting from splitting upon all variables. These two values are added for every variable. The variable that has the smallest Gini index amongst daughter nodes is selected as the variable to be split upon.

3.2 Tree Pruning

Tree pruning is a method that works by growing a large tree T_0, where the split procedure is halted when some predefined minimum node size is obtained. This tree is then pruned back in order to obtain a subtree, T_a (Friedman et al., 2001)(Loh, 2011). In order to decide the best way to prune T_0, a method known as cost complexity pruning is often made of use (Friedman et al., 2001). The cost complexity criterion is defined as

$$C_\alpha(T) = \sum_{m=1}^{V} N_m Q_m(T) + \alpha V \tag{3.4}$$

where V is the number of terminal nodes in T_α, $Q_m(T)$ is a predefined cost function and $\alpha \geq 0$ is the complexity parameter which dictates a trade off between predictive capacity and tree size (Friedman et al., 2001).

The classification error rate and Gini index can be used as possible $Q_m(T)$ (Friedman et al., 2001). Since the goal of the pruning is to produce a tree that has the highest prediction accuracy possible, the classification error rate is generally preferable (Loh, 2011). The value of α is typically selected by K-folds cross validation (Friedman et al., 2001). The algorithm for building a classification tree is detailed below.

Algorithm 3.2: Classification Trees

1. Split the data into a training and test set randomly.
2. Define a sequence of complexity parameters where $\alpha_1 < \alpha_2 < \cdots < \alpha_q$ and $\alpha_1 = 0$.
3. Partition the training data randomly into K folds. By the end of the algorithm, each fold should be used as a validation set once. Begin by using the first fold as the validation set and cycle through to the Kth fold.
4. For each fold:
   (a) Grow T_0 on the fold data using recursive binary splitting. Stop when each terminal node reaches a predefined minimum number of observations. For instance, $N_m = 5$. Use the Gini index to determine splits.
   (b) Find the subtree that minimizes $C_\alpha(T)$. This will yield q subtrees.
   (c) For each subtree, predict the class of the observations in the validation set by dropping them down the tree.
   (d) For each subtree, calculate the classification error rate.
5. Find the cross validation error for each α. To do this, sum the classification error rate for each α across all folds.
6. Select the value of α that corresponds to the lowest cross validation error.
7. Using the full training set, construct a tree with the value of α selected. Predict the class of the observations in the test set by dropping them down the tree.

3.3 Bagging

Classification trees often have high variance. Bootstrap aggregation, otherwise known as bagging, is a method of finding a prediction model with a reduced variance through repeated sampling. It is often used in conjunction with classification trees (Breiman, 1996). If L is considered to be a sample from a given population, then one way to reduce the variance of the prediction is to resample as many times as possible from the population. A separate prediction model could then be created for each sample and the resulting predictions would be averaged (Breiman, 1996). Consider B training set response predictions $\hat{L}^1(x), \hat{L}^2(x), \ldots, \hat{L}^B(x)$. These predictions could be aggregated through

$$\hat{L}_{avg}(x) = \frac{1}{B}\sum_{b=1}^{B} \hat{L}^b(x) \tag{3.5}$$

which is impractical, as access to multiple L is often not a reality (Breiman, 1996). An alternative is bootstrapping, which takes repeated samples from the training data set. These samples represent the B bootstrapped training data sets. The tree is then trained on the bth bootstrapped training set in order to get $\hat{L}^{*b}(x)$. Predictions are then averaged to obtain

$$\hat{L}_{bag}(x) = \frac{1}{B}\sum_{b=1}^{B} \hat{L}^*_b(x) \tag{3.6}$$

which is what is known as bagging (Breiman, 1996). In the classification context a procedure such as voting is generally used. Let $N_k = \#\{*b; \hat{L}^{*b} = k\}$ be the count of the kth class amongst the $*b$ bootstrapped response predictions. The aggregated prediction for the response is then

$$\hat{L}_{bag}(x) = \text{argmax}_k(N_k) \tag{3.7}$$

which is the most common class for the predicted response amongst all bootstraps (Breiman, 1996).

3.4 The Random Forest

When discussing random forests, it is most natural to explain how they deviate from the bagging procedure previously discussed in section 3.3. This method improves over bagging classification trees through a straight-forward modification which de-correlates the trees.

Consider a set of predictors that contains a continuous variable whose correlation to the response is much higher than the other predictors. In this scenario, the trees from the bagging procedure will conduct splits almost exclusively on this variable (Breiman, 2001). This will yield predictions that are highly correlated amongst the trees. When taking the average of B i.i.d random variables attributed with same variance σ^2, the average variance is $\frac{1}{B}\sigma^2$. If the assumption of independence is invalid, then correlations between variables must be considered. The aggregated variance is then

$$\rho\sigma^2 + \frac{1-\rho}{B}\sigma^2 \tag{3.8}$$

where ρ is the correlation between bagged predictors (Friedman et al., 2001). It is clear from equation 3.8 that as the number of bootstraps rises, the variance lowers. If the predictors among bagged trees are highly correlated, the effects of the variance reduction is limited.

Random forests take motivation from these issues and seek to eliminate the correlation between bagged trees. This is accomplished by modifying the split process. When each split is considered, a random sample of $m \leq p$ predictors is taken as candidates for the split (Breiman, 2001). In order to obtain a reasonable subset of split candidates, typically $m \approx \sqrt{p}$ (Breiman, 2002). With this specification, the highly correlated variable will not always be considered as a split candidate. Some trees may still be correlated, however, this effect is greatly diminished through the split sampling alteration. When $m = p$, the mechanism of the random forest methodology is identical to bagging. The aggregation of bootstraps is defined as

$$\hat{L}_e^*(x) = \frac{1}{B}\sum_{b=1}^{B} \hat{L}_b^*(x) \tag{3.9}$$

where e is commonly referred to the ensemble and x refers to training or test set inputs (Breiman, 2001). In order to obtain a reasonable aggregation, a large amount of bootstraps, typically 500, are taken (Breiman, 2002). Another important concept to make note of is the out-of-bag (OOB) samples. This procedure is used as an alternative to the typical training and test set method of error quantification. An OOB observation is an observation that does not appear in a set of particular bootstrapped samples. The OOB ensemble is defined by

$$\hat{L}_e^{**}(x) = \frac{\sum_{b=1}^{B} I_{i,b} \hat{L}_b^*(x)}{\sum_{b=1}^{B} I_{i,b}} \tag{3.10}$$

where $I_{i,b} = 1$ if i is an OOB case for b, otherwise $I_{i,b} = 0$.

Due to the class imbalance mentioned in chapter 2, a modified bootstrap procedure is used. Typically, bootstrap samples are sampled with replacement from the training set. This does not take the class imbalance of the response into consideration. Instead, a bootstrap sample is drawn with replacement from the minority class. Afterwards, a sample of the same size is taken with replacement from the majority class. These two samples are then combined into a single bootstrap sample. The combined sample ensures that each class has equal representation. This technique is referred to as downsampling and creates a "balanced random forest" (Chen et al., 2004).

In terms of model assessment, OOB error rate is a popular metric. It allows the user to quantify the classification error of a single prediction through an ensemble that contains information from all other observations. This error rate is calculated by noting the number of times the response is correctly predicted out of all the grown trees (Breiman, 2001). Despite the strengths of the OOB error rate, it is not used as it is sensitive to the class distribution of the response. Instead, the AUC is used to better assess the performance of the RF.

The random forest algorithm for binary classification is given below.

Algorithm 3.4: The Random Forest

1. Split the data into a training and test set randomly.
2. Draw a bootstrap sample with replacement from the training set. Use downsampling to ensure classes are balanced in the sample.

3. Using the bootstrap sample, grow a tree via recursive binary splitting. At each split, randomly sample $m \approx \sqrt{p}$ predictors as potential split candidates. Use the Gini index to determine splits.
4. Stop growing the tree when a predefined minimum node size is met. In this case, $N_m = 1$.
5. Repeat steps 2 to 4 B times. Here, $B = 500$.
6. For test set predictions:
   (a) Drop observations from the test set down all B trees and predict the response.
   (b) For each observation, find the most prevalent predicted class amongst the B trees through 3.7. These are the final test set predictions.

Another important detail to mention is that each time algorithm 3.4 is used, a different test set AUC will be obtained. This is due to the variation amongst trees. To better assess performance, algorithm 3.4 is iterated 30 times. The average test set AUC amongst the iterations is then taken. This is used as the estimate of model performance.

3.5 Random Forests in Credit Risk

A comprehensive review of credit scoring models conducted by Lessmann et al. (2015) considers RFs to be a top performing standard of comparison for new methodologies. This review compared many different models, such as k-nearest-neighbor, logistic regression, stochastic gradient boosting and SVMs with differing kernels. It also includes the data used in this thesis as one of many data sets that were applied. Random forests out performed most other methods, with an average rank of 14.8 out of 41 in terms of the best overall model for prediction (Lessmann et al., 2015). In another comparison, Brown and Mues (2012) employ RFs on imbalanced credit scoring data. Similar to the study done by Lessmann et al. (2015), Brown and Mues (2012) conclude that the random forest classifiers "perform very well in a credit scoring context and are able to cope comparatively well with pronounced class imbalances in these data sets". Furthermore, Wang et al. (2011) employs RFs and uses the German credit data. They achieved an average classification accuracy of 77.05%.

The general consensus behind these studies is that RFs perform well in the context of credit risk data. It should be noted that in Wang et al. (2011), the standard accuracy metric was used

to quantify model performances. For reasons mentioned in section 6.1, metrics like accuracy are sensitive to class imbalances. As a result, the statistics involved here may be a misleading indication of performance. The work done in Lessmann et al. (2015) and Brown and Mues (2012) involve the AUC and yield similar RF performances. All in all, the conclusions of these works seem to establish the RF as a solid performer in the credit risk classification setting.

Chapter 4

Support Vector Machines

The support vector machine (SVM) is a machine learning method suggested by Cortes and Vapnik (1995) that is primarily focused on binary classification. It is accomplishes this through the use of boundaries, support vectors and kernels. SVMs are capable of dealing with complicated predictor-response relationships that other methods would not be able to cope with (Cortes and Vapnik, 1995).

This chapter introduces the SVM starting from defining hyperplanes. It then explains the hyperplane classifier and leads into how they translate into the support vector classifier. Next, the transition into nonlinear decision boundaries is made. Kernel functions and the SVM are then explained. Finally, the use of SVMs in credit risk is investigated in other works.

4.1 Hyperplanes

In $\mathbb{R}^p$ dimensions, a hyperplane is a flat affine subspace of dimension $\mathbb{R}^{p-1}$ where p is the number of predictors (James et al., 2013). For instance, a straight line is a one-dimensional hyperplane in two dimensional space and a plane is a two-dimensional hyperplane in three-dimensional space (James et al., 2013). In $\mathbb{R}^p$ dimensions a hyperplane is defined by the following equation

$$\beta_0 + \beta^T x = 0 \tag{4.1}$$

for parameters $\beta = \beta_1, \ldots, \beta_p$ and predictors $x = x_1, \ldots x_p$ (Friedman et al., 2001). Figure 4.1 below indicates the appearance of a hyperplane in $\mathbb{R}^2$ dimensional space. Here β^* refers to $\frac{\beta}{||\beta||}$, the vector which is normal to the plane.

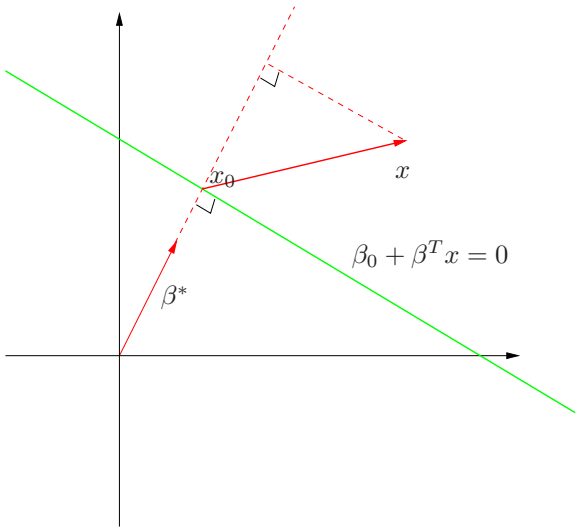


Figure 4.1 – A hyper plane in $\mathbb{R}^2$ dimensional space. Credit for image: Friedman et al. (2001).

4.2 Hyperplanes as Classifiers

The hyperplane can function as a classifier, where points lying on either side of the hyperplane create two different classes. A hyperplane classifier is typically referred to as an optimal separating hyperplane. Cortes and Vapnik (1995) defines this as a hyperplane that separates two classes and looks to maximize the distance to the closest point from either class. These points are known as support vectors and are equidistant points on either side of the maximal margin (optimal) hyperplane (James et al., 2013). If $y_1, \ldots, y_n \in \{-1, 1\}$ represents a binary response with corresponding predictors $x_1, \ldots, x_n \in \mathbb{R}^p$ then the maximal margin hyperplane is the solution to the following optimization problem

$$\begin{aligned} &\text{maximize}_{\beta, \beta_0, ||\beta||=1} \quad M \\ &\text{subject to} \quad y_i(x_i^T \beta + \beta_0) \geq M, \quad i = 1, \ldots, N \end{aligned} \tag{4.2}$$

These constraints ensure that each observation is on the correct side of the hyperplane and at least a distance M from said hyperplane (James et al., 2013). Therefore M represents the margin of the hyperplane and the optimization problem chooses β such that it maximizes M. Figure 4.2 represents maximal margin hyperplane classification for the linearly separable $\mathbb{R}^2$ dimension case.

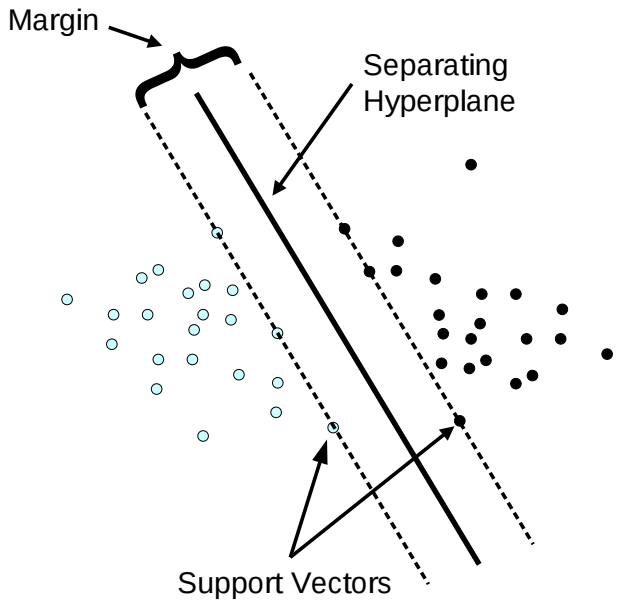


Figure 4.2 – Maximal margin hyperplane classification for a linearly separable case. Credit for image: Meyer (2015).

The $||\beta|| = 1$ condition can be eliminated by redefining terms

$$\begin{aligned} \frac{1}{||\beta||} y_i(x_i^T\beta + \beta_0) &\geq M \\ y_i(x_i^T\beta + \beta_0) &\geq ||\beta||M \end{aligned} \tag{4.3}$$

and setting $||\beta|| = 1/M$. This allows 4.2 to be written as

$$\begin{aligned} &\min_{\beta,\beta_0} \frac{1}{2}||\beta||^2 \\ &\text{subject to} \quad y_i(x_i^T\beta + \beta_0) \geq 1, \quad i = 1, \ldots, N \end{aligned} \tag{4.4}$$

which leads into Lagrange functions (Friedman et al., 2001). The Lagrange function that is minimized with respect to β_0 and β is

$$L_p = \frac{1}{2}||\beta||^2 - \sum_{i=1}^{N} \alpha_i \left[y_i(x_i^T \beta + \beta_0) - 1 \right] \tag{4.5}$$

whose derivatives can be taken and set to zero (Friedman et al., 2001). This yields

$$\beta = \sum_{i=1}^{N} \alpha_i y_i x_i \tag{4.6}$$

$$0 = \sum_{i=1}^{N} \alpha_i y_i \tag{4.7}$$

which can then be substituted into 4.5

$$L = \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i=1}^{N} \sum_{k=1}^{N} \alpha_i \alpha_k y_i y_k x_i^T x_k \tag{4.8}$$
$$\text{subject to} \quad \alpha_i \geq 0$$

resulting in what is called the Wolfe dual (Friedman et al., 2001). Another constraint on the solution is

$$y_i(x_i^T \beta + \beta_0) - 1 = 0 \quad \forall i \tag{4.9}$$

and is a part of the Karush-Kuhn-Tucker conditions along with 4.6 and 4.7 (Friedman et al., 2001). The solution to β in 4.6 is dependent on x_i, which are points subject to $a_i \geq 0$. These are the previously mentioned support vectors. The function

$$\hat{f}(x) = x^T\hat{\beta} + \hat{\beta}_0 \tag{4.10}$$

is the result of the optimization problems. This is used along with

$$\hat{G}(x) = \text{sign}\hat{f}(x) \tag{4.11}$$

to predict the class of new observations (Friedman et al., 2001).

4.3 Support Vector Classifiers

The downfall of using the method discussed in section 4.2 is that it assumes the data is linearly separable. This does not occur very often in practice. An extension of this method is known as the support vector classifier. It attempts to remove the assumption of linearly separable data by introducing slack into the margin. This "slack" allows a certain proportion of observations to appear on the wrong side of the margin.

Let $\epsilon = (\epsilon_1, \epsilon_2, \ldots, \epsilon_N)$ be defined as the slack variables. Then the conditions present in 4.2 can be expressed as

$$y_i(x_i^T\beta + \beta_0) \geq M(1 - \epsilon_i) \tag{4.12}$$

subject to $\forall i$, $\epsilon_i \geq 0$, $\sum_{i=1}^{N} \epsilon_i \leq C$ (Friedman et al., 2001). Here ϵ_i is the proportional amount a prediction is on the wrong side of its margin by and C is a fixed parameter called the "cost". This optimization can be better written as

$$\min \frac{1}{2}||\beta||^2 + C\sum_{i=1}^{N} \epsilon_i \quad \text{subject to} \quad y_i(x_i^T\beta + \beta_0) \geq 1 - \epsilon_i \quad \forall i \quad \text{and} \quad \epsilon_i \geq 0 \tag{4.13}$$

similarly to section 4.2 (Friedman et al., 2001). The Lagrange function for 4.13 is

$$L = \frac{1}{2}||\beta||^2 + C\sum_{i=1}^{N}\epsilon_i - \sum_{i=1}^{N}\alpha_i\Big[y_i(x_i^T\beta + \beta_0) - (1-\epsilon_i)\Big] - \sum_{i=1}^{N}\mu_i\epsilon_i \tag{4.14}$$

which is minimized with respect to β, β_0 and ϵ_i (Friedman et al., 2001). The derivatives can be solved for and set to zero yielding

$$\beta = \sum_{i=1}^{N}\alpha_i y_i x_i \tag{4.15}$$

$$0 = \sum_{i=1}^{N}\alpha_i y_i \tag{4.16}$$

$$\alpha_i = C - \mu_i, \quad \forall i \tag{4.17}$$

subject to α_i, μ_i, $\epsilon_i \geq 0\ \forall i$ (Friedman et al., 2001). Substituting 4.15, 4.16 and 4.17 into 4.14 yields

$$L_D = \sum_{i=1}^{N}\alpha_i y_i - \frac{1}{2}\sum_{i=1}^{N}\sum_{i'=1}^{N}a_i a_{i'} y_i y_{i'} x_i^T x_{i'} \tag{4.18}$$

which is the Wolfe dual (Friedman et al., 2001). This is subject to the constraints

$$\alpha_i\Big[y_i(x_i^T\beta + \beta_0) - (1-\epsilon_i)\Big] = 0 \tag{4.19}$$

$$\mu_i\epsilon_i = 0 \tag{4.20}$$

$$y_i(x_i^T\beta + \beta_0) - (1-\epsilon_i) \geq 0 \tag{4.21}$$

which, in tandem with 4.18, is the specification for the support vector classifier (Friedman et al., 2001). The solution of β has the form

$$\hat{\beta} = \sum_{i=1}^{N} \hat{\alpha}_i y_i x_i \tag{4.22}$$

with nonzero $\hat{\alpha}_i$ only for observations that meet the constraint 4.21. Similarly to section 4.2, these are the support vectors. A proportion of support vectors will lie on the margin where $\hat{\epsilon}_i = 0$. These are characterized by 4.17 and 4.20 where $0 < \hat{\alpha}_i < C$. $\hat{\beta}_0$ can be solved for directly using said points. The remaining support vectors ($\hat{\epsilon}_i > 0$) have $\hat{\alpha}_i = C$. As in section 4.2, 4.11 is used to predict the class of new observations (Friedman et al., 2001).

The tuning parameter, C, controls the number of observations that are allowed to violate the margin. As a result, C also determines the bias-variance trade off of the optimization (Friedman et al., 2001). When C is small, more observations can violate the margin. Variance is therefore minimized at the expense of bias. In contrast, when C is large, fewer observations can violate the margin. Bias is then minimized, however, variance increases (Friedman et al., 2001). Figure 4.3 illustrates support vector classifiers for a nonlinearly separable case with different values of C.

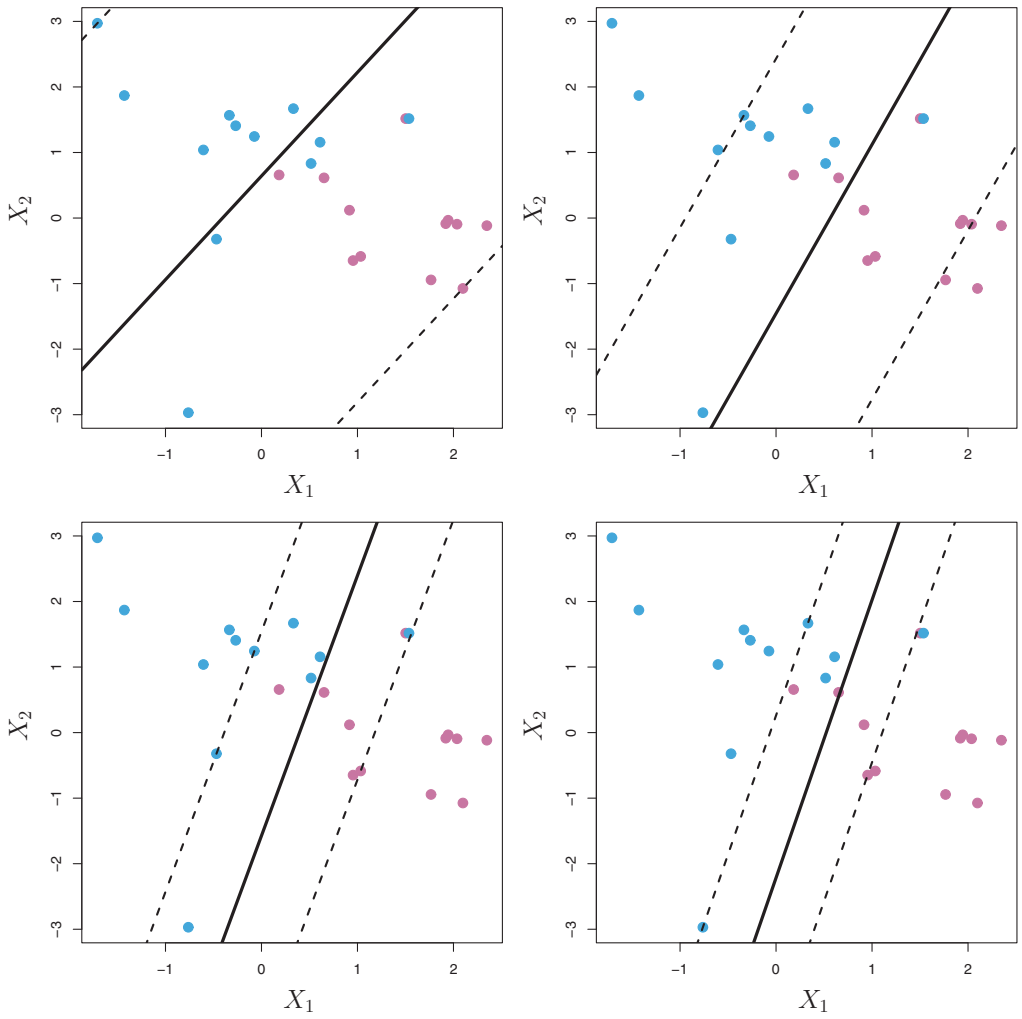


Figure 4.3 – Support vector classifiers for a nonlinearly separable case. Here C is at its largest in the bottom right plot. Credit for image: James et al. (2013).

4.4 Classification with Nonlinear Decision Boundaries

There could be situations where a linear boundary between classes is not adequate. In these situations, a more complicated boundary may be necessary. Nonlinear boundaries are a potential solution to this issue. A popular method of implementation for nonlinear boundaries are called linear basis functions. They operate by changing the vector inputs of X with a transformed version, then use linear models in this new space of derived inputs (Friedman et al., 2001).

Define the mth transformation of X where $m = 1, \ldots, M$ by $h_m(x)$. Then 4.10 can be expressed as

$$\hat{f}(x) = h_m(x)^T \hat{\beta} + \hat{\beta}_0 \tag{4.23}$$

which is now linear in the transformed variables (Friedman et al., 2001). This means that the support vector classifier in section 4.3 can function as described but with a new set of predictors in the form of $h_m(x)$. As before, the class of new observations is predicted using 4.11 (Friedman et al., 2001).

4.5 The Support Vector Machine

There are situations where the support vector classifier is not efficient at enlarging the feature space. This can happen with high degree polynomial basis functions or other complicated bases. An approach known as the support vector machine (SVM) was suggested by Cortes and Vapnik (1995) in order to handle this issue.

First, define

$$\langle x, x' \rangle = \sum_{i=1}^{p} x_i x'_i \tag{4.24}$$

as the inner product of two vectors x and x' both of length p. Now denote the transformed feature

vectors by $h(x_i)$ and $h(x_i')$. The Lagrange (Wolfe) dual function 4.18 now has the form

$$L_D = \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i=1}^{N} \sum_{i'=1}^{N} \alpha_i \alpha_{i'} y_i y_{i'} \langle h(x_i), h(x_{i'}) \rangle \tag{4.25}$$

and note that from the solution 4.15 $f(x)$ can be written as

$$f(x) = h(x)^T \beta + \beta_0 = \sum_{i=1}^{N} \alpha_i y_i \langle h(x), h(x_i) \rangle + \beta_0 \tag{4.26}$$

which involves only the inner products (Friedman et al., 2001). Written in this way, the original transformation $h(x)$ does not need to be known. Only knowledge of kernel function

$$K(x, x') = \langle h(x), h(x') \rangle \tag{4.27}$$

which solves inner products in the transformed space, is needed (Cortes and Vapnik, 1995)(Friedman et al., 2001). These formulations allow the feature space to be enlarged much more efficiently than the support vector classifier. The decision function can be written using kernels as

$$\hat{f}(x) = \sum_{i=1}^{N} \hat{\alpha}_i y_i K(x, x_i) + \hat{\beta}_0$$

where observations are once again assigned to a class by 4.11. This means that the specifics of $h(x)$ do not need to be worked out. One can simply decide on a kernel and implement it directly into the support vector classifier. This thesis uses the following kernel

$$K(x, x') = \exp \Big[-\gamma ||x - x'||^2 \Big] \tag{4.28}$$

which is known as the radial basis function (RBF) kernel. Other kernel types are not considered as the RBF kernel is recommended by studies done in the same context as this work and outperforms them (Huang et al., 2007)(Bellotti and Crook, 2009).

Another important detail of the SVM is that it can not handle categorical inputs naturally due to how the boundaries are constructed. Categorical predictors must first be converted into numerical dummy variables. Dummy variables take the form

$$D_{ij} = \begin{cases} 1 & \text{if level i} \\ 0 & \text{if else} \end{cases} \tag{4.29}$$

where $i = 1, \ldots, p$ are the number of categorical predictors and $j = 1, \ldots, l$ are their respective number of levels. A further technicality is that the SVM is sensitive to the scale of predictors. Predictors that are high in magnitude have a tendency to overpower ones that are not. As a result, all numerical predictors must be centered and scaled before they are trained on.

4.6 Support Vector Machines in Credit Risk

In the comprehensive review of credit scoring models conducted by Lessmann et al. (2015), linear and RBF SVMs have an average rank of 23.4 and 23.2 out of 41 respectively for overall performance. While not the worst classifiers, the SVMs perform similarly to more simplistic methods such as logistic regression. This brings up the question of whether or not SVMs should be considered over these methods. A point to consider is that the SVM is much more flexible. It is capable of handling high-dimensional data and nonlinear relationships. Regression based approaches struggle with these scenarios. Lessmann et al. (2015) notes that "real-world credit data sets are typically large and high-dimensional". As such, the SVM would seem to be a more appropriate method to use on most credit data sets.

Another application of SVMs to credit scoring data conducted by Bellotti and Crook (2009) has more positive results. The authors tested SVMs against traditional methods using data from a large credit card database. They found that SVMs are "competitive and can be used as the basis of a feature selection method to discover those features that are most significant in determining risk of default" (Bellotti and Crook, 2009). In addition, Huang et al. (2007) conduct a similar study

that compares SVMs to neural networks, genetic programming and decision tree classifiers. These authors also make use of the German credit data. They conclude that "experimental results show that SVM is a promising addition to the existing data mining methods" (Huang et al., 2007).

Lessmann et al. (2015) implies that SVMs can achieve similar performances to less complicated models on credit risk data. Although, the study also mentions that credit data is typically high-dimensional. This implies that SVMs may be more appropriate for most credit data sets. Bellotti and Crook (2009) and Huang et al. (2007) coincide in that SVMs are preferred methods to use for credit risk analysis. Based on these observations, there is enough evidence to support the use of SVMs for credit risk classification.

Chapter 5

Missing Data, Imputation and Simulation

Most machine learning techniques are designed based on the assumption that the data has no missing observations. It is not possible to train a model on a predictor that contains missing information. One method of dealing with this is to remove all rows in the data containing missing information. This is known as a complete-case analysis. The downside is a potentially large amount of information loss depending on the proportion of missing data. Another method of dealing with this issue is to impute the missing information using some sort of statistical method (Enders, 2010). This thesis uses an imputation technique suggested by van Buuren and Oudshoorn (1999) called multiple imputation by chained equations (MICE). Its underlying mechanism is more comprehensive than other, more simplistic, techniques (van Buuren and Oudshoorn, 1999)(van Buuren, 2007)(Azur et al., 2011). MICE involves modeling an incomplete variable using the remaining complete variables in the data. The underlying algorithm requires the specification of the conditional distribution of each incomplete variable with regards to the complete. It is therefore convenient to make use of a nonparametric method that is capable of handling all data types. This can greatly reduce algorithm complexity by only needing to specify one model for all incomplete variables. As a result, a method suggested by Rubin (1986) known as predictive mean matching is made use of by this thesis. It is semiparametric and capable of handling ordinal, nominal and numerical variables.

This chapter goes over several mechanisms of missing data and the assumptions made under each. Next, the MICE method is explained. Afterwards, an explanation of PMM is given followed by an outline of the missing data simulation. Finally, a review of other works that involve MI based methods in credit risk analyses are discussed.

5.1 Missing Data Mechanisms

First, consider a data set X that contains $i = 1, \ldots, p$ variables and $j = 1, \ldots, , n$ observations. Define R as a $p \times n$ indicator matrix that describes the missing data mechanism. Let entry $r_{ij} = 0$ if the entry is missing and 1 if it is observed. In addition, let X_{mis} represent the missing information in the data and X_{obs} represent the observed information. Furthermore, let ϕ be a set of unknown nuisance parameters. Finally, define π as the probability of encountering a particular missing data mechanism.

The least restrictive mechanism is known as missing completely at random (MCAR). The probability of encountering the MCAR mechanism can be written as

$$\pi = p(R|\phi) \tag{5.1}$$

which implies that the probability of encountering missing data is completely unrelated to the data. Under this assumption, it is possible to train models using complete-case data. This is not recommended due the aforementioned potential for information loss. Furthermore, incorrect specification of the MCAR assumption can lead to computational issues such as bias. As a result, most imputation methods are not based on the MCAR mechanism.

In contrast, the most restrictive assumption is missing not at random (MNAR). The probability of encountering the MNAR mechanism can be written as

$$\pi = p(R|X_{obs}, X_{mis}, \phi) \tag{5.2}$$

which implies that the probability of encountering missing data is related to the missing data itself (Enders, 2010). The MNAR assumption can be problematic to work with as the factors that influence X_{mis} are difficult to ascertain.

A middle ground between MCAR and MNAR is known as missing at random (MAR). The probability of encountering the MAR mechanism can be written as

$$\pi = p(R|X_{obs}, \phi) \tag{5.3}$$

which implies that the probability of encountering missing data is related to the observed information, but not itself (Enders, 2010). MAR is the most commonly used assumption in imputation methods as it does not carry the risk of MCAR misspecification and the complexity of MNAR (Enders, 2010).

5.2 Multiple Imputation by Chained Equations

Traditional imputation techniques such as the use of the median and mean are often inadequate for many scenarios. They are not capable of handling categorical predictors properly. Furthermore, the correlation among predictors is not taken into consideration by the imputation (Enders, 2010). In many situations, missing data is imputed once, which is referred to as single imputation. With a single imputation, it is not possible to establish error rates or construct appropriate diagnostics (Azur et al., 2011). To combat these problems, multiple imputation (MI) methods have been developed. These involve conducting the imputation independently many times, constructing M unique imputed data sets. MI techniques allow for meaningful error specification and imputation diagnostics.

This thesis makes use of a MI technique known as fully conditional specification or multiple imputation by chained equations (MICE) (van Buuren and Oudshoorn, 1999). For traditional MI methods, the joint distribution of the variables within the imputation model must be predefined. In the presence of multivariate data that contains numerical, ordinal and nominal variables, this is usually inappropriate. The MICE methodology involves a series of regressions where each variable that contains missing data is modeled conditionally with regards to the rest (Azur et al., 2011). This extension to MI allows each variable to be modeled by its respective distribution. Like most imputation techniques, MICE operates under the assumption that the missing data mechanism is MAR. From Azur et al. (2011), the MICE procedure is outlined below in algorithm 5.2 .

Algorithm 5.2: MICE

1. A basic imputation (such as the median) is done for every missing value in the dataset. These are "place holder" imputations.
2. The place holder imputations for a particular variable are set back to missing.
3. The observed values from the variable in step 2 are regressed on the other variables in the imputation model. "Regression" in this context can refer to a nonparametric approach as well. This thesis makes use of PMM, described in section 5.3.
4. The missing values from the variable in question are replaced by the predicted values from the regression model fit in step 3. These predicted values have an error term associated with the model fit added to them. This is done in order to introduce variance into the imputations.
5. Steps 2-4 are repeated for every variable that contains missing data. At the end of this cycle, each missing value is replaced by a prediction that depends on the relationships between the variables in the data.
6. Steps 1-5 are iterated a predefined amount of times. At the end of each iteration, the imputation values are updated. By the end of the iterations, the distribution governing the imputations should converge and become stable. This constitutes the construction of one imputed data set.
7. Steps 2-6 are repeated a predefined number of times in order to construct M imputed data sets.
8. Select one of the M imputed data sets to replace the original.

There is no direct method of choosing M and the number of iterations. Instead, this specification depends primarily on the size of the imputation model. When the number of observations and predictors that contain missing data increases, the algorithm runtime can inflate drastically. Early research implied that an M of 5-10 was appropriate. Later research by Graham et al. (2007) seems to indicate that diminishing returns in imputation performance occur at $M = 40$. Therefore, M is set to 40 and the number of iterations is set to 25. This provides a reasonable number of iterations to reach convergence and an M with maximized performance.

Post-imputation, diagnostics are typically done to asses whether or not the procedure was successful. This thesis makes use of three diagnostics; stream line plots, density plots and fre-

quency plots. Stream line plots are constructed in order to determine if the imputations have reached convergence over the predefined iterations. The mean and standard deviation for the imputed values are plotted against the number of iterations for all M data sets and all variables. Ideally, these M lines should be free of any pattern and intermingle. Furthermore, they should converge to a stationary value (van Buuren and Groothuis-Oudshoorn, 2011)(Azur et al., 2011). Plots are constructed in order to assess the validity of the imputations for continuous variables. These plots superimpose the average imputed density over the observed density. For the imputations to be valid, the average imputed density should follow similar trends compared to the observed density. Similarly, for categorical variables, frequency plots are constructed. The frequency distribution of the observed data is plotted along with the average imputed frequency distribution. Like the density plots, the general trend of the average imputed frequency distribution should be similar to the observed (van Buuren and Groothuis-Oudshoorn, 2011)(Azur et al., 2011).

5.3 Predictive Mean Matching

As discussed in section 5.2, a method needs to be chosen for imputation within the MICE algorithm. The imputation model involved in this work contains numerical, ordinal and nominal data. Instead of specifying a different model for each type of data, one model is used that is capable of handling all three. This semiparametric method is known as predictive mean matching (PMM) and was suggested by Rubin (1986) with later extensions done by Little (1988). Let Y_{obs} be a vector of the observed values for a variable selected for imputation. In the MICE procedure, this would be the variable whose placeholder imputations are set back to missing. Additionally, define Y_{mis} as a vector of the missing values for said variable. Finally, let X be a matrix of the remaining variables which are completely observed. Then from Vink et al. (2014) the PMM algorithm is given below.

Algorithm 5.3: PMM

1. Use a linear regression of Y_{obs} given X to estimate $\hat{\beta}$, $\hat{\sigma}$ and $\hat{\epsilon}$ by means of ordinary least squares. This model can be written in the form $Y_{obs} = X^T\beta + \epsilon$. If Y is a categorical variable, the levels are changed to numerical values.

2. Draw β^* from the posterior predictive distribution of β; $\beta^* \sim N(\hat{\beta}, \hat{\sigma}^2(X^TX)^{-1})$. This is a multivariate normal distribution with mean $\hat{\beta}$ and covariance matrix $\hat{\sigma}^2(X^TX)^{-1}$.

3. Calculate $\hat{Y}_{obs} = X\hat{\beta}$ and $\hat{Y}_{mis} = X_{mis}\beta^*$.

4. For each $\hat{Y}_{mis,i}$, find $\Delta_i = |\hat{Y}_{obs} - \hat{Y}_{mis,i}|$. Here $i = 1, \ldots, l$ refers to the l predictions in $\hat{Y}_{mis}$. If n is the total number of observations in Y, then the length of each vector Δ_i is $n - l$.

5. Randomly sample one value from $\Delta_i' = (\Delta_i^{(1)}, \ldots, \Delta_i^{(k)})$, where Δ_i' are the ordered k smallest elements of Δ_i. Take the Y_{obs} associated with the selected Δ_i' as the imputation. The value k is commonly referred to the amount of "donors" and must be predefined.

The only specification for PMM is the number of donors, k. This value controls the amount of variation in the imputations done by PMM. If $k = 1$, the only variation from one imputation to the next would be the draws from the predictive posterior distribution of β. While there is no direct method of selecting k, using fixed values between 3 and 10 is a popular approach (Morris et al., 2014). The authors of the *mice* R package promote the use of $k = 5$ (van Buuren and Groothuis-Oudshoorn, 2011). Therefore, $k = 5$ is used as a fixed number of donors.

5.4 Missing Data Simulation

The original data involved in this thesis does not contain any missing information. As a result, a missing data simulation is conducted to test MICE in conjunction with PMM. The entire simulation procedure is detailed below.

1. Train models on the unaltered data and note the model performances (pre-simulation).
2. Introduce missing information at a rate of 25% randomly into predictors.
3. Impute the missing information.
4. Retrain models on the new data containing the imputed information and note performances (post-simulation).
5. Assess the validity of imputations and compare the model performances from pre and post-simulation.

This simulates a situation where the missing data mechanism is MCAR and a complete-case

analysis is not possible due to the high proportion of missing values. In reality, it is not likely that 25% of each variable will be missing. Such an extreme simulation is done in order to stress test the imputations. The idea is that if MICE alongside PMM is capable of handling a situation such as this, it will be able to handle data with less missing values. Motivation for this type of simulation is taken from "Flexible Imputation of Missing Data" (van Buuren, 2012).

5.5 Multiple Imputation in Credit Risk

A study done by Florez-Lopez (2010) was conducted on missing credit data. Its purpose was to "analyse the performance of several methods for dealing with missing data such as likewise deletion, simple imputation methods, MLE models and advanced multiple imputation (MI) alternatives based on MarkovChain-MonteCarlo and re-sampling methods" (Florez-Lopez, 2010). The Australian credit approval data set is used in this research (Lichman, 2013). It is very similar in nature to the German credit data set in the sense that the response is a binary variable corresponding to credible or non-credible and predictors are a mixture of continuous and categorical socioeconomic variables. After comparing methodologies, Florez-Lopez (2010) concludes "MI models are found to provide very valuable solutions with regard to credit risk missing data". These models include expectation maximization (EM), EM with importance resampling, EM with bootstrapping and the imputation-posterior method. Florez-Lopez (2010) mentions but does not make use of MICE and PMM.

Beyond Florez-Lopez (2010), there does not appear to be any comprehensive studies that involve credit risk data and multiple imputation methods. With that in mind, the results presented by Florez-Lopez (2010) are promising due to the similarities between the data involved in that work and this thesis. In tandem with the overall strengths of MI based procedures, this gives enough evidence to use MICE and PMM in the context of credit risk. The lack of other studies provides a unique avenue of research.

Chapter 6

Parameter Tuning and Feature Selection

Parameter tuning and feature selection are important components of the machine learning process. This chapter first highlights the parameter tuning procedure and issues stemming from the data. Proceeding this, feature selection is outlined in addition to imputation and model specific issues.

6.1 Parameter Tuning

Parameter tuning refers to a procedure that attempts to find values of the parameters that would maximize model performance. Before this is done, however, the performance metric that parameters are optimized with regards to must be selected. As mentioned in section 2.1, the German credit data suffers from a class imbalance. Maximizing measures such as overall accuracy or the RFs OOB error rate will perform poorly in this scenario. The training procedure will sometimes yield a model that will achieve poor performance when classifying individuals from the under-represented class. This would imply that nearly all non-credible individuals would be classified as credible, which is far from desirable. A confusion matrix is defined below in table 6.1.

	cr	ncr
cr	True Positives (TP)	False Negatives (FN)
ncr	False Positives (FP)	True Negatives (TN)

Table 6.1 – A confusion matrix where cr is credible and ncr is non-credible. The row labels are the actual class while the column labels are predicted class.

The class distribution is the proportion of positive to negative instances. As such, any metric that depends directly on values from both of the rows in table 6.1 will be sensitive to class imbalance. In order to account for this problem, a metric that is insensitive to class distributions needs to be used. One of these metrics is known as the area under curve (AUC) (Fawcett, 2006). In particular, it refers to the area under the receiver operation characteristic curve (ROC). This will be explained before delving any further into parameter tuning.

A discrete classifier, such as the SVM and RF, will produce as single confusion matrix. Define the true positive rate (TPR) as $\frac{TP}{TP+FN}$. Similarly, define the false positive rate (FPR) as $\frac{FP}{FP+TN}$. Plotting the FPR against the TPR would constitute one point in the ROC space (Fawcett, 2006). In order to obtain a curve, additional points must be calculated. This involves changing model predictions from discrete to probabilistic. In other words, the model will output the probability an observation belongs to the positive class. This probability is then compared to a threshold value in order to determine the class. Consider a threshold of 0.70. This would imply that if an observation has a probability of belonging to the positive class greater than or equal to 0.70, it is set to the positive class. Otherwise, it is set to the negative class. These probabilities can also be thought of as ranks, where a higher probability constitutes a higher rank. SVMs have their predictions converted to a probabilistic output through the method suggested by Platt (1999). RFs have a built in voting system for classification. Therefore, probabilistic outputs are readily available through $\frac{\text{positive votes}}{\text{total votes}}$. With these outputs, the ROC can be constructed. This process is detailed below.

Algorithm 6.1.1: ROC Construction

1. Randomly separate the data into training and test sets. Train a particular model on the training set.
2. Calculate probabilistic outputs for each observation in the test set.
3. Generate a large number of probability thresholds ranging between 0 and 1.
4. For each threshold, apply the results from step 2 and create a confusion matrix.
5. Plot the TPR against the FPR for each threshold.

Afterwards, the AUC is calculated by using trapezoids under the curve. Since the AUC is derived from the TPR and FPR, it is insensitive to class imbalances as no interaction between classes is used to compute it (Fawcett, 2006). The AUC can theoretically range between 0 and 1. Here 0

corresponds the worst classification performance whereas 1 corresponds to the best. An AUC of 0.50 refers to performance that is no better than random guessing. This is represented by a straight line and is typically added to the plot for the sake of comparison. Generally speaking, the AUC can be interpreted as "the probability that the classifier will rank a randomly chosen positive instance higher than a randomly chosen negative instance" (Fawcett, 2006). By maximizing the AUC, it in turn maximizes the prediction performance of both classes.

Parameter tuning does not usually need to be done on RFs. The only parameters that can be tuned are the number of trees grown and the split candidate sample size. For the majority of scenarios these can be kept as the default specifications per section 3.4. In contrast, SVMs require each parameter involved to be tuned. If this is not done, performance will be heavily jeopardized. Tuning is done with a grid search procedure which is defined below.

Algorithm 6.1.2: Grid Search

1. For all parameters involved with the model, select a range of possible values.
2. Partition the training data randomly into K folds. By the end of the algorithm, each fold should be used as a validation set once. Begin by using the first fold and cycle through to the Kth fold. In this case, $K = 10$.
3. For each fold:
   (a) For each parameter combination, train the model on the fold data.
   (b) Predict the outcome of the observations in the validation set. Calculate the AUC.
4. Average the AUC associated with each parameter combination across the K folds.
5. Select the parameter combination associated with the highest average AUC as the optimal parameter combination.

6.2 Feature Selection

Feature selection is a procedure by which a statistical model is simplified through removing predictors that are irrelevant or provide no information to model performance. This is important for both RFs and SVMs. It allows the former to have less complicated trees and the later to be expressed

in a lower dimension. There are many different methods of feature selection, however, this thesis makes use of recursive feature elimination (RFE). It begins with the model including all predictors and systematically removes predictors that contribute the least to the AUC. The procedure continues until only one predictor is left in the model. Afterwards, the optimal number of predictors to retain in the final model is determined graphically or by a predefined metric. In the context of this thesis, RFE can be defined by algorithm 6.2.

Algorithm 6.2: Recursive Feature Elimination

1. Randomly partition the data into training and test sets.
2. Partition the training data randomly into K folds. By the end of the algorithm, each fold should be used as a validation set once. Begin by using the first fold and cycle through to the Kth fold. In this case, $K = 10$.
3. For each fold:
   (a) Tune and train the model on the fold data using the remaining predictors.
   (b) Predict the outcome of the observations in the validation set. Calculate the AUC.
   (c) Determine variable importance or rankings. Variables are ranked in importance based on how much the AUC drops when they are removed from the prediction model.
   (d) Remove the least important variable from the model.
   (e) Repeat steps (a)-(d) until only one variable remains in the model.
4. Average the AUC associated with each number of variables remaining across the K folds.
5. For SVMs:
   (a) Plot the average AUC against the number of variables remaining in the model. Determine the point at which the AUC plateaus. Select the number of variables corresponding to this point as the optimal number of predictors. To find which variables are optimal predictors, average their rank across the K folds.
6. For RFs:
   (a) Repeat steps 2-4 R times. In this case, $R = 5$. Take the average of the result from step 4 across the R repeats.

(b) Plot the average AUC from (a) against the number of variables remaining in the model. Determine the point at which the AUC plateaus. Select the number of variables corresponding to this point as the optimal number of predictors. To find which variables are optimal predictors, average their rank across the K folds and then across the R repeats.

The downside of RFE is that it can become computationally inefficient when exposed to a large number of predictors. Only 20 predictors and 1000 observations are contained within the German credit data set. This dimensionality is not an issue for RFE. The grid search procedure for SVMs is not included in the RFE to reduce computation time. Instead, broad parameter selections are specified in the tuning step and the models are tuned on the reduced predictor set after feature selection. The SVM tuning within the RFE procedure uses 5-fold cross validation.

As indicated in section 4.5, dummy variables need to be created for the categorical predictors before SVMs are trained on them. Consequentially, each level is a different variable within the model. Certain levels can contain more relevant information about the response. As a result, the RFE procedure may only find specific levels of a variable to be important. When this occurs, the entire variable is taken into the model, even if there is only one level from it present in the optimal predictor set. Variable importance ranking is determined in this way. Categorical variables are ranked according to their most important level by way of algorithm 6.2. RFs do not have this problem and variables are ranked according to their specific importance.

Like SVMs, RFs have some issues associated with feature selection. Simulation studies conducted by Strobl et al. (2007) imply that "when random forest variable importance measures are used with data of varying types, the results are misleading because sub-optimal predictor variables may be artificially preferred in variable selection". This is problematic as the German credit data fits this description. Unlike SVMs, however, there does not appear to be an immediate way to rectify the problem. Strobl et al. (2007) maintain that the "two mechanisms underlying this deficiency are biased variable selection in the individual classification trees used to build the random forest on one hand, and effects induced by bootstrap sampling with replacement on the other hand". This implies that the cause of the compromised feature selection is due to the RF methodology itself and not the feature selection procedure. In light of this, the result of the RFE procedure is compared between RFs and SVMs in order to highlight the differences in variable importance ranking. Another issue is that the cross validation procedure experiences additional variance. This is due to the variation amongst trees. The repeated cross validation step in algorithm 6.2 is implemented to account for this.

Yet another roadblock for feature selection is the presence of missing values post-simulation. MICE involves predicting missing observations in a particular variable using the information from other variables. A natural consequence is that the newly imputed information is correlated with the observed. As mentioned in section 5.2, only one imputed data set is taken at the end of the procedure. Feature selection techniques applied to this data can often result in the incorrect predictors being selected (Wood et al., 2008). A popular method of dealing with this is to use the feature selection procedure on the complete-case data and then apply the reduction to the imputed data (Wood et al., 2008). This is not a possibility due to the simulation involved with this work, as each predictor contains 25% missing information. With only 1000 observations, the sample size of the complete-case data would be too small for training procedures. An alternative is to use the "stacked" imputed data. This takes all of the M imputed data sets and merges their results together into one massive data set (Wood et al., 2008). Despite being feasible, from the specifications in section 5.2 where $M = 40$, this would yield a training sample size of 32,000. This is especially problematic for SVMs as the RFE algorithm would have its run time largely inflated. Instead, RFE is done pre-simulation and the optimal predictors are applied in both the pre and post-simulation. Since RFE is done before the missing data simulation, the chance of imputed information affecting the procedure is eliminated. It should be noted that in a non-simulation setting this approach is not possible and an alternative solution would have to be explored.

Chapter 7

Results

In this chapter the results are displayed. They are presented in the order that the analysis is done in; pre-simulation, simulation, post-simulation and comparisons. Discussion of the results is left to chapter 8.

7.1 Pre-simulation

Table 7.1 shows the training and test set that is obtained through sampling. Subsections 7.1.1 and 7.1.2 contain the results of the RFs and SVMs respectively.

Set	Sample Size	Credible	Non-credible
Training	800	560	240
Test	200	140	60

Table 7.1 – The results of the sampling conducted on the German credit data set.

7.1.1 Random Forests

The results of the full model RF are given in table 7.2. Table 7.3 contains the outcome of the RF RFE procedure. This is followed by figure 7.1, which contains the respective plot. Finally, table 7.4 yields the results of the reduced model RF.

Number of Trees:	500
Number of Variables:	20
m:	4
Average AUC:	0.781

Table 7.2 – The classification performance of the full model RF on the pre-simulation test data. An average AUC of 0.781 means that, on average, the RF will rank a randomly chosen credible individual higher than a randomly chosen non-credible individual 78.1% of the time.

Variables	Average AUC	AUC SD
1	0.716	0.050
2	0.744	0.051
3	0.755	0.051
4	0.769	0.052
5	0.781	0.047
6	0.769	0.048
7	0.777	0.046
8	0.771	0.049
9	0.773	0.049
10	0.775	0.048
11	0.778	0.050
12	0.779	0.046
13	0.783	0.043
14	**0.791**	**0.041**
15	0.787	0.046
16	0.786	0.045
17	0.792	0.042
18	0.782	0.043
19	0.791	0.042
20	0.789	0.042

Importance Rank	Variable Name
1	checking_acc_status
2	cred_amnt
3	purpose
4	age_years
5	duration_month
6	cred_history
7	emplmnt_length
8	savings_acc_bonds
9	property
10	inst_rate
11	marital_status_sex
12	current_residence_dur
13	job
14	**inst_plans**
15	housing
16	num_exist_credits
17	debtors_or_guarantors
18	liable_maintenance
19	telephone
20	foreign_worker

Table 7.3 – Results of the pre-simulation RF RFE procedure on the training set. The table on the left contains the average AUC associated with the number of variables remaining in the model. Bolded values correspond to the optimal predictor set. The table on the right contains the importance ranking of each variable where the bolded variable is the last included in the optimal predictor set.

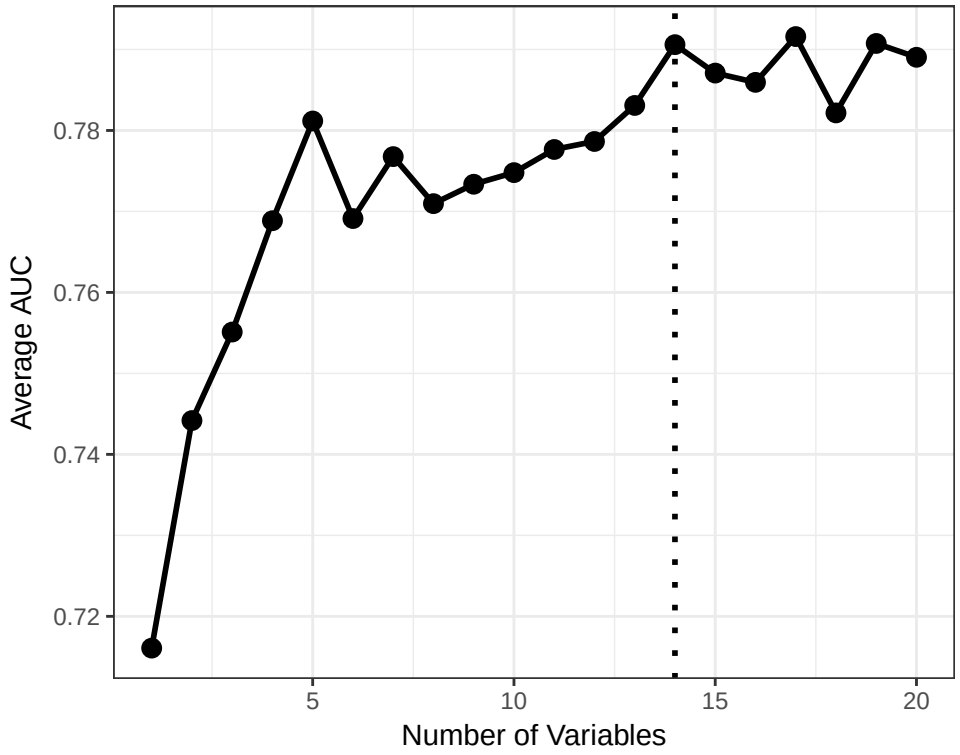


Figure 7.1 – A plot of the average AUC associated with the number of variables left in the model for the RF RFE procedure. This was applied to the pre-simulation training data. The dotted line represents the optimal predictor set.

Number of Trees:	500
Number of Variables:	14
m:	3
Average AUC:	0.780

Table 7.4 – The classification performance of the reduced RF model selected by the RFE procedure on the pre-simulation test data. An average AUC of 0.780 means that, on average, the RF will rank a randomly chosen credible individual higher than a randomly chosen non-credible individual 78.0% of the time.

7.1.2 Support Vector Machines

The results of the tuning procedure for the SVM are given in table 7.5. Respective model fits for optimal parameters can be found in table 7.6. The outcome of the RFE procedure can then be examined in table 7.7 with the associated plot present in figure 7.2. Table 7.8 contains the tuning procedure for reduced model SVM. Finally, table 7.9 includes the respective model fit.

C	γ	Average AUC	AUC SD
100.000	0.050	0.670	0.042
10.000	0.050	0.698	0.049
100.000	0.005	0.726	0.035
0.100	0.050	0.754	0.026
100.000	0.002	0.766	0.034
0.100	0.001	0.770	0.040
0.100	0.002	0.771	0.039
0.100	0.005	0.772	0.038
100.000	0.001	0.772	0.040
10.000	0.005	0.774	0.035
10.000	0.001	0.774	0.039
10.000	**0.005**	**0.776**	**0.047**

Table 7.5 – The tuning procedure for the full model SVM on the pre-simulation training data. Bolded values correspond to the optimal parameter combination.

C	γ	Variables	AUC
10	0.005	20	0.752

Table 7.6 – The classification performance of the full model SVM on the pre-simulation test data. An AUC of 0.752 means that the SVM will rank a randomly chosen credible individual higher than a randomly chosen non-credible individual 75.2% of the time.

Variables	Average AUC	AUC SD
3	0.566	0.198
4	0.596	0.171
1	0.641	0.067
2	0.709	0.070
6	0.711	0.102
⋮	⋮	⋮
42	0.783	0.062
21	**0.784**	**0.075**
30	0.784	0.064
⋮	⋮	⋮
37	0.789	0.066
47	0.789	0.058
61	0.790	0.062
38	0.790	0.068
52	0.790	0.058
36	0.791	0.067
49	0.791	0.056
34	0.792	0.068
51	0.792	0.060
35	0.793	0.069

Importance Rank	Variable Name
1	checking_acc_status
2	duration_month
3	cred_history
4	savings_acc_bonds
5	age_years
6	housing
7	marital_status_sex
8	purpose
9	property
10	inst_plans
11	cred_amnt
12	**emplmnt_length**
13	inst_rate
14	num_exist_credits
15	telephone
16	foreign_worker
17	current_residence_dur
18	debtors_or_guarantors
19	liable_maintenance
20	job

Table 7.7 – Results of the pre-simulation SVM RFE procedure on the training set. The table on the left contains the average AUC associated with the number of variables remaining in the model. The bolded values correspond to the optimal predictor set. Note that based on what was discussed in section 6.2, this includes dummy variables for each level of the categorical predictors. See that section for how variables were ranked. The table on the right contains the importance ranking of each variable where the bolded variable is the last included in the optimal predictor set.

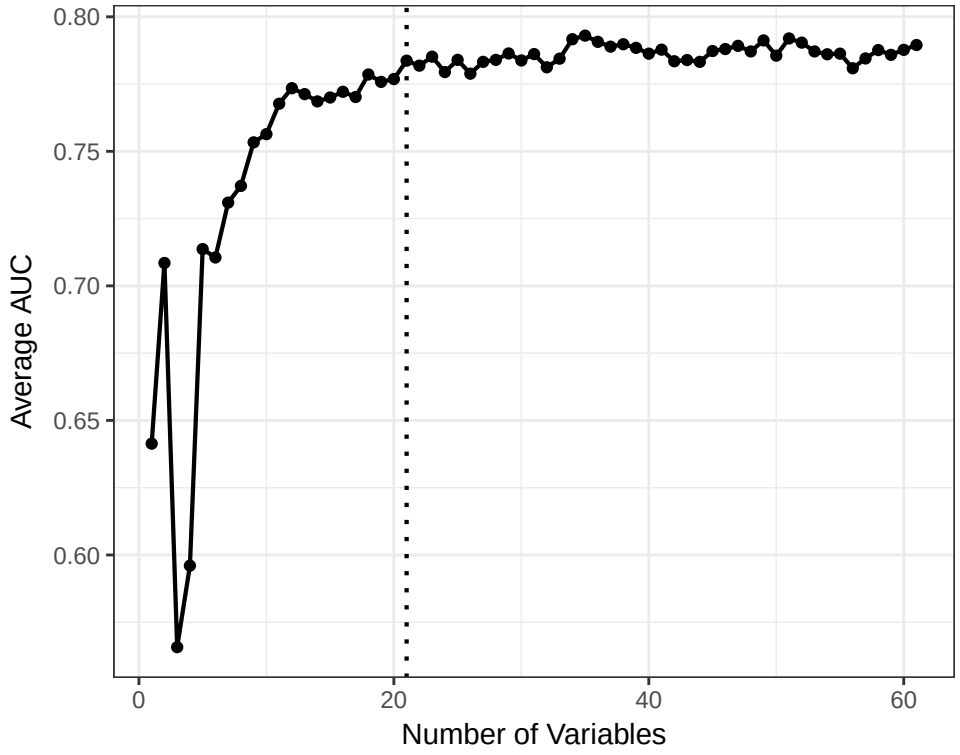


Figure 7.2 – A plot of the average AUC associated with the number of variables left in the model for the SVM RFE procedure. This was applied to the pre-simulation training data. The dotted line represents the optimal predictor set.

C	γ	Average AUC	AUC SD
100.000	0.005	0.773	0.025
0.100	0.001	0.776	0.053
0.100	0.002	0.778	0.052
0.100	0.005	0.780	0.047
10.000	0.001	0.786	0.043
10.000	0.002	0.789	0.035
100.000	0.001	0.798	0.024
100.000	0.002	0.800	0.019
10.000	**0.005**	**0.801**	**0.064**

Table 7.8 – The tuning procedure for the reduced model SVM on pre-simulation training data. Bolded values correspond to the optimal parameter combination.

C	γ	Variables	AUC
10	0.005	12	0.748

Table 7.9 – The classification performance of the reduced SVM model selected by the RFE procedure on the pre-simulation test data. An AUC of 0.748 means that the SVM will rank a randomly chosen credible individual higher than a randomly chosen non-credible individual 74.8% of the time.

7.2 Simulation, Imputation and Diagnostics

The results of the simulation can be found in figure 7.3 in the form of a missing value plot. One example of the stream line, density and frequency plots are given thereafter in figures 7.4, 7.5 and 7.6 respectively. The remaining plots can be found in their designated appendices, A, B and C.

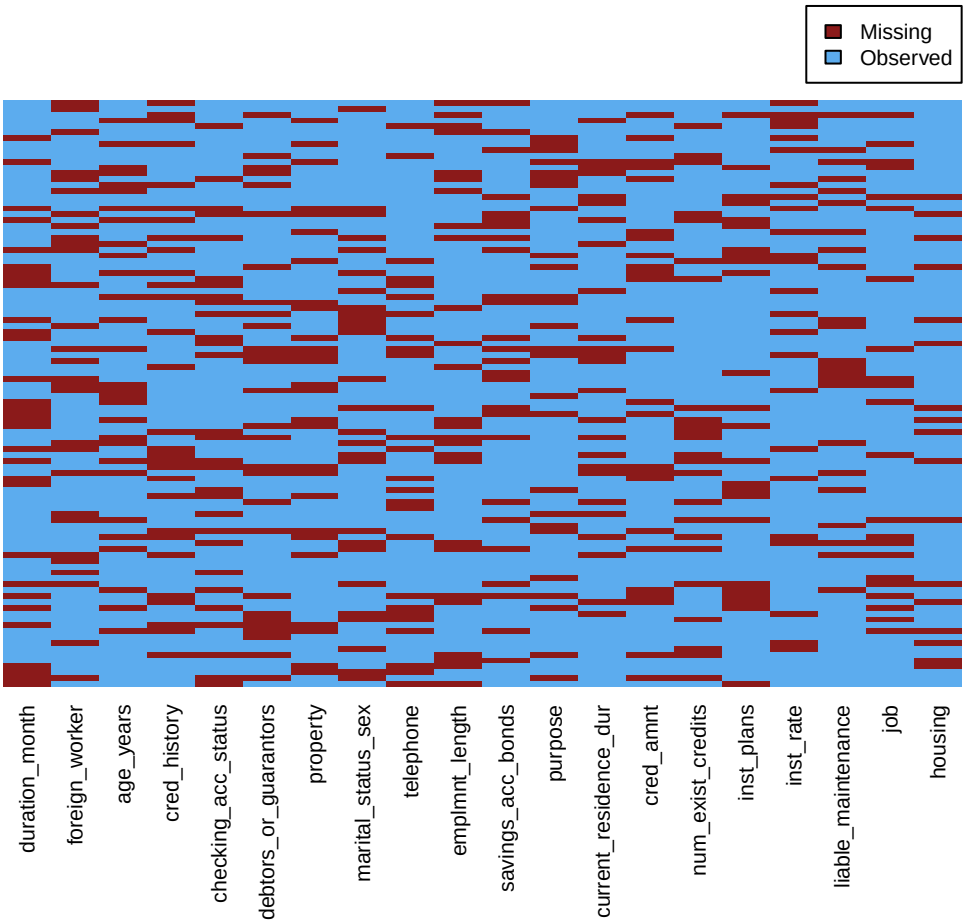


Figure 7.3 – A plot of the missing value pattern for a random sample of size 100. All predictors are included. As previously discussed in section 5.4, each predictor had 25% of their values set to missing. This represents a MCAR mechanism where a complete-case analysis is not possible.

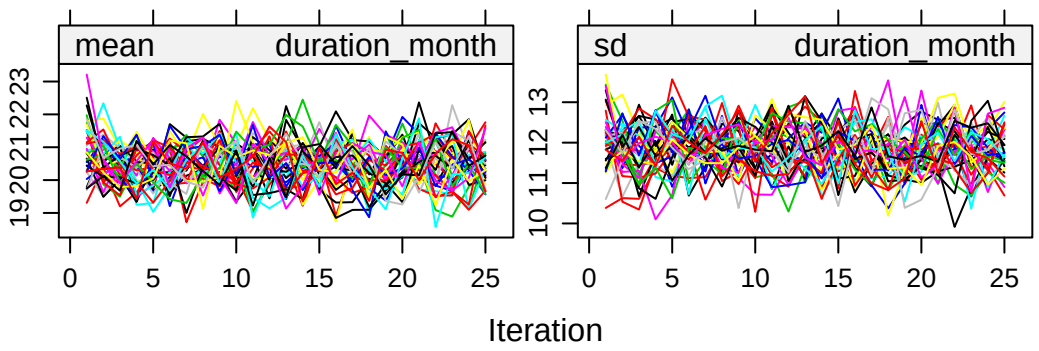


Figure 7.4 – The imputation stream line plots for the mean and standard deviation of the *duration_month* variable. Each line represents a different imputed data set. For both the mean and standard deviation, the imputations converge to stationary values. This implies valid imputations. All other variables converged as well.

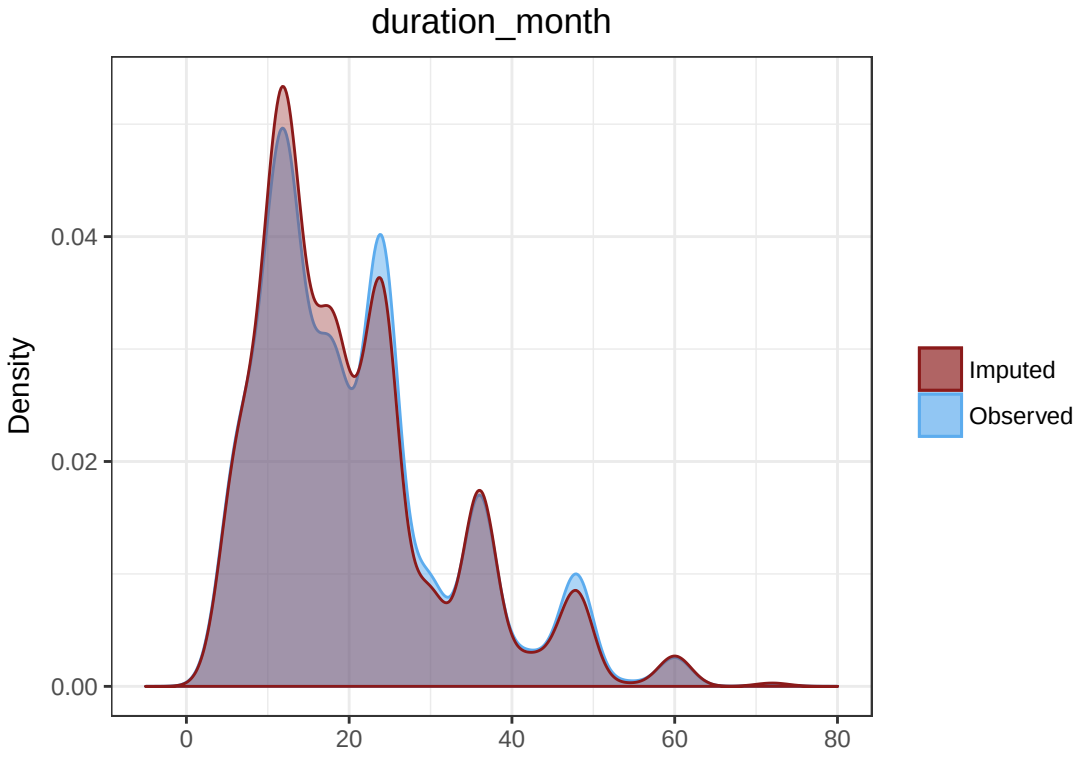


Figure 7.5 – Density plots for the *duration_month* variable. The blue line represents the density for the observed data and the red line is the average imputed density. The average imputed density follows similar trends compared to the observed density. This implies valid imputations. All other continuous variables experience this as well.

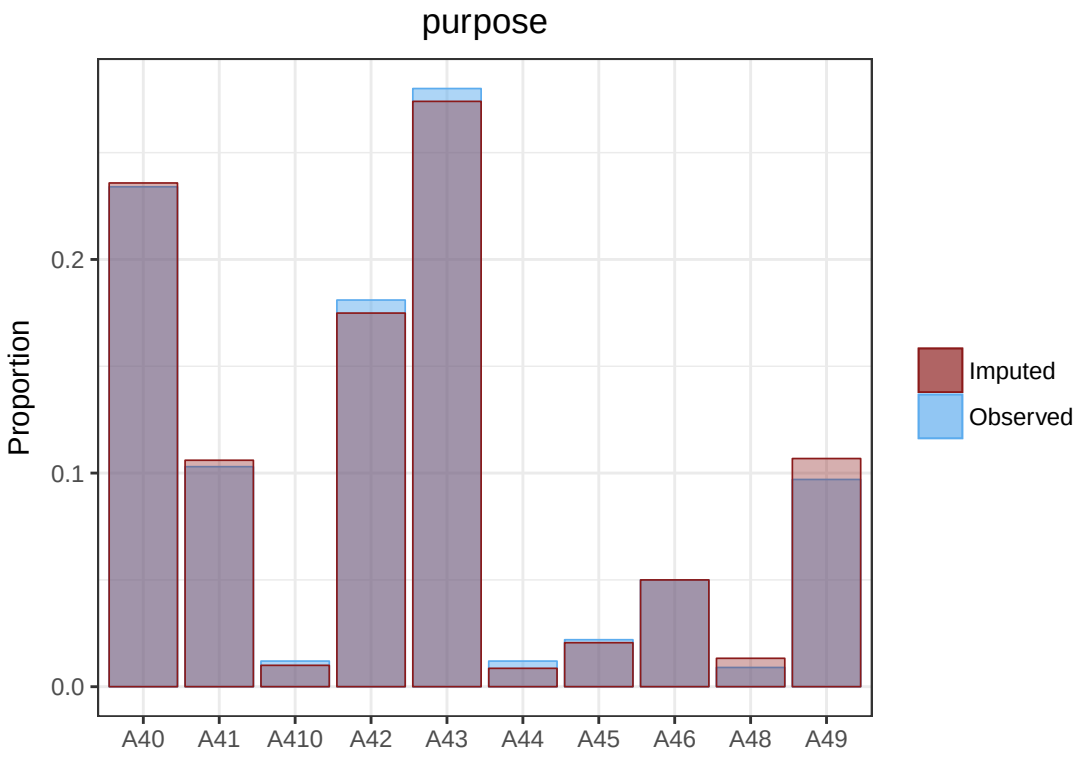


Figure 7.6 – Frequency plots for the *purpose* variable. Blue refers to the observed data while red represents the imputed data. The average imputed frequencies follow similar trends compared to the observed frequencies. This implies valid imputations. All other discrete variables experience this as well.

7.3 Post-simulation

To stay consistent with the pre-simulation data, the same sample is used post-simulation. This means that row indices are identical between pre and post-simulation training and test sets, however, the values may be different if they were removed and imputed. Furthermore, recall from section 6.2 that the dimension reduction from the pre-simulation is applied to the post-simulation models. Therefore, to find the reductions, refer back to section 7.1.

7.3.1 Random Forests

Table 7.10 contains the full and reduced model RF fits.

Full Model		Reduced Model	
Number of Trees:	500	Number of Trees:	500
Number of Variables:	20	Number of Variables:	14
m:	4	m:	3
Average AUC:	0.755	Average AUC:	0.752

Table 7.10 – The classification performance of the reduced and full model RFs on the post-simulation test data. An average AUC of 0.755 means that, on average, the full RF will rank a randomly chosen credible individual higher than a randomly chosen non-credible individual 75.5% of the time. The reduced RF will do so 75.2% of the time.

7.3.2 Support Vector Machines

Tables 7.11 and 7.12 contain the tuning procedure for the full and reduced model SVM respectively. These are followed by table 7.13, which contains the associated fits.

C	γ	Average AUC	AUC SD
100.000	0.005	0.703	0.059
100.000	0.002	0.738	0.055
10.000	0.005	0.756	0.046
100.000	0.001	0.770	0.046
0.100	0.005	0.777	0.028
0.100	0.002	0.777	0.029
0.100	0.001	0.777	0.027
10.000	0.002	0.779	0.038
10.000	**0.001**	**0.781**	**0.033**

Table 7.11 – The tuning procedure for the full model SVM on post-simulation training data. Bolded values correspond to the optimal parameter combination.

C	γ	Average AUC	AUC SD
100.000	0.005	0.733	0.050
100.000	0.002	0.767	0.046
0.100	0.001	0.770	0.053
0.100	0.002	0.771	0.053
0.100	0.005	0.772	0.053
10.000	0.005	0.775	0.052
100.000	0.001	0.777	0.048
10.000	0.002	0.777	0.051
10.000	**0.001**	**0.778**	**0.046**

Table 7.12 – The tuning procedure for the reduced model SVM on post-simulation training data. Bolded values correspond to the optimal parameter combination.

Full Model				Reduced Model			
C	γ	Variables	AUC	C	γ	Variables	AUC
10	0.001	20	0.783	10	0.001	12	0.772

Table 7.13 – The classification performance of the reduced and full model SVMs on the post-simulation test data. An AUC of 0.783 means that the full SVM will rank a randomly chosen credible individual higher than a randomly chosen non-credible individual 78.3% of the time. The reduced SVM will do so 77.2% of the time.

7.4 Comparison

This section compares the model results from the pre and post-simulation data. Direct model comparisons are given in table 7.14. Proceeding this, the variable importance rankings are compared between methods in table 7.15. Finally, the ROC plots for the models in the pre and post-simulation setting are given in figures 7.7 and 7.8 respectively.

Pre-simulation		Post-simulation	
Model	AUC	Model	AUC
Full SVM	0.752	Full SVM	0.783
Reduced SVM	0.748	Reduced SVM	0.773
Full RF	0.781	Full RF	0.755
Reduced RF	0.780	Reduced RF	0.752

Table 7.14 – A comparison of the AUC between the pre and post-simulation models. Between full and reduced models, average RF AUC is 3.6% lower and SVM is 3.7% higher.

RF		SVM	
Importance Rank	Variable Name	Importance Rank	Variable Name
1	checking_acc_status	1	checking_acc_status
2	cred_amnt	2	duration_month
3	purpose	3	cred_history
4	age_years	4	savings_acc_bonds
5	duration_month	5	age_years
6	cred_history	6	housing
7	emplmnt_length	7	marital_status_sex
8	savings_acc_bonds	8	purpose
9	property	9	property
10	inst_rate	10	inst_plans
11	marital_status_sex	11	cred_amnt
12	current_residence_dur	**12**	**emplmnt_length**
13	job	13	inst_rate
14	**inst_plans**	14	num_exist_credits
15	housing	15	telephone
16	num_exist_credits	16	foreign_worker
17	debtors_or_guarantors	17	current_residence_dur
18	liable_maintenance	18	debtors_or_guarantors
19	telephone	19	liable_maintenance
20	foreign_worker	20	job

Table 7.15 – A comparison of the variable importance rankings between methods. Bolded values indicate the last variable included in the optimal predictor set.

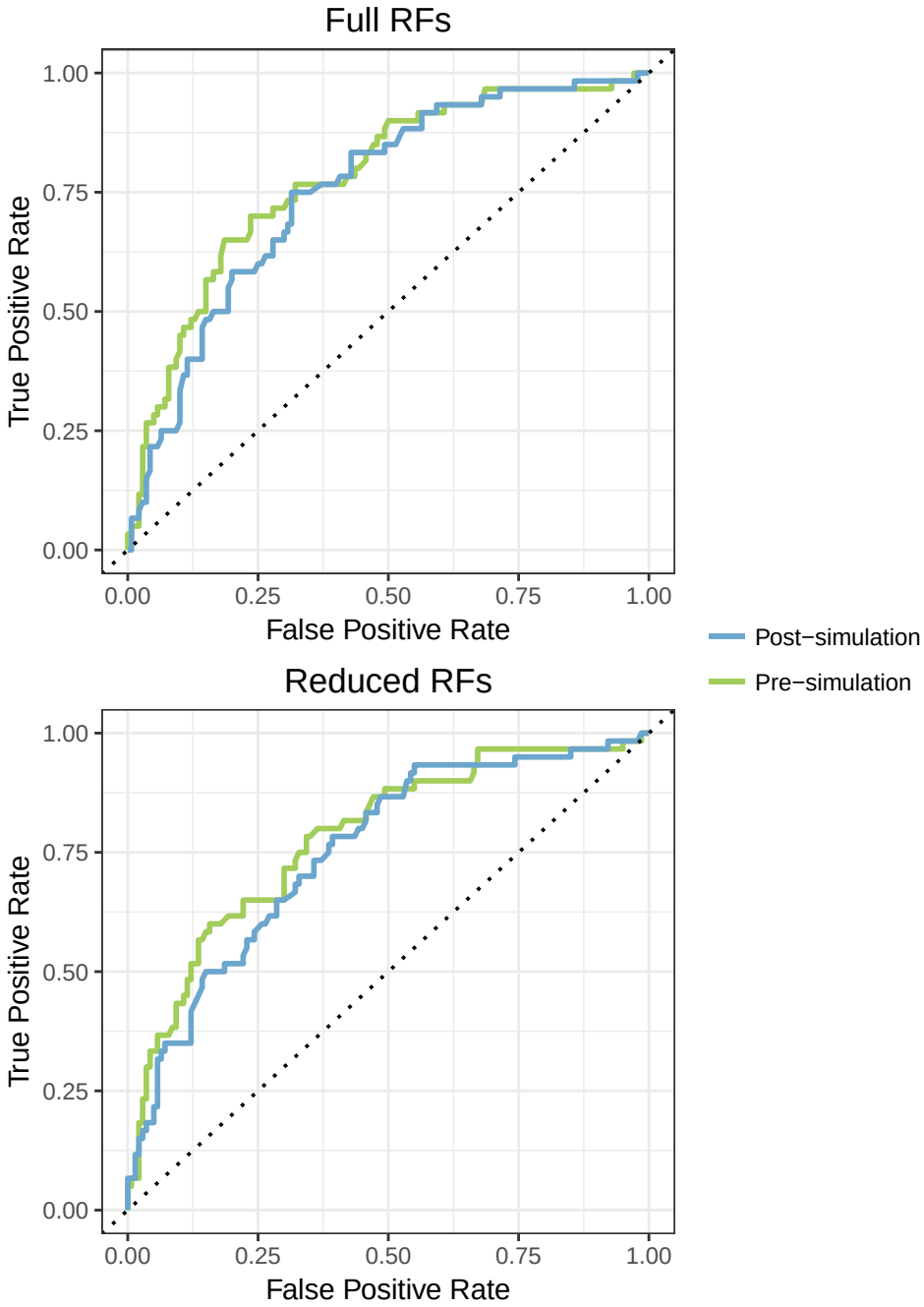


Figure 7.7 – ROC plots for the RFs involved in the pre and post-simulation. The area under the dotted line corresponds to a performance that is no better than random guessing. Note that the ROC for each RF model corresponds to the model that deviated the least from its respective average AUC.

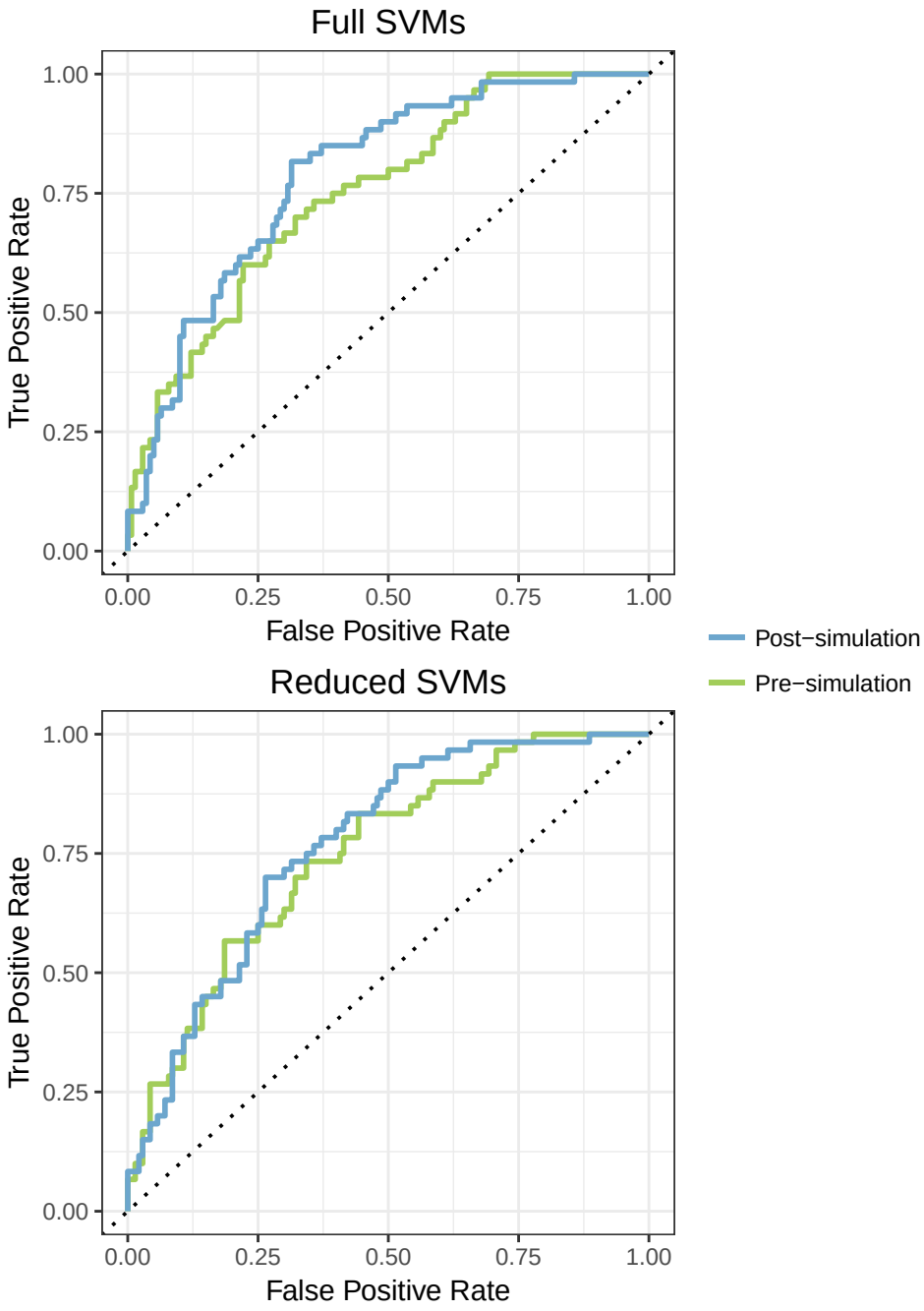


Figure 7.8 – ROC plots for the SVMs involved in the pre and post-simulation. The area under the dotted line corresponds to a performance that is no better than random guessing.

Chapter 8

Discussion

The results of the pre-simulation analysis present in table 7.14 seem to indicate that RFs perform better than SVMs. Both the reduced and full model RF yield a noticeably higher AUC when compared to SVMs. This is consistent with the work done by Lessmann et al. (2015) where RFs also outperformed SVMs in similar settings. The RFE procedure produced reduced models that had a negligible decrease in AUC for RFs and SVMs. Although, from table 7.15 it is apparent that there are some differences between the methods with regards to variable importance ranking. This is expected, as the work done by Strobl et al. (2007) indicated that RF feature selection is biased towards continuous variables and categorical variables with many levels. For example, the RFE procedure identified the continuous variable *cred_amnt* as the second most important variable for the RF model. In contrast, the same variable was ranked 11th out of 20 by the SVM RFE procedure. Despite the compromised feature selection, the highest ranked variables from the SVM RFE procedure are still contained within the reduced RF model. This explains why there was a negligible drop in AUC. In consideration of these factors, the reduced RF model appears to be the top performer in the pre-simulation setting.

The missing value plot present in figure 7.3 reveals the simulation successfully introduced missingness into all variables. As was intended, a complete-case analysis is not viable despite the underlying mechanism being MCAR. Furthermore, the stream line plots contained within appendix A converge to stationary values without any noticeable patterns. In addition, the average imputed densities contained in appendix B follow similar trends compared to the observed densities. Finally, the average imputed frequencies within appendix C behave similarly compared to their observed counterparts. These diagnostics imply that the imputations performed by MICE in

conjunction with PMM are valid.

The post-simulation results present in table 7.14 suggest that there are differences in model performances compared to the pre-simulation. Between full and reduced models, average RF AUC was 3.6% lower and SVM was 3.7% higher. Considering that imputation was done on 25% of the data, these differences are quite minor. Since imputations are valid, these performance differences would likely be negligible in the presence of realistic missingness.

These observations reveal that using MICE in conjunction with PMM can be an optimal method of imputation within the context of credit risk data. This is consistent with Florez-Lopez (2010), where it was found that MI based methodologies performed well in similar settings. As was stated earlier, little research has been done on the use of MI techniques in the credit risk context. While this thesis provides some insight on the subject, there are still many areas to explore. Further research could review alternatives to PMM or examine solutions to MNAR data.

Bibliography

Melissa J. Azur, Elizabeth A. Stuart, Constantine Frangakis, and Philip J. Leaf. Multiple imputation by chained equations: What is it and how does it work? *International Journal of Methods in Psychiatric Research*, 20(1):40–49, 2011.

Dean Baker. The housing bubble and the financial crisis. *Real-World Economics Review*, 2008.

Tony Bellotti and Jonathan Crook. Support vector machines for credit scoring and discovery of significant features. *Expert Systems with Applications*, 36(2):3302–3308, 2009.

Fischer Black and Myron Scholes. The pricing and options and corporate liabilities. *The Journal of Political Economy*, 81(3):637–654, 1973.

Leo Breiman. Bagging predictors. *Machine Learning*, 24(2):123–140, August 1996.

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

Leo Breiman. *Manual On Setting Up, Using, And Understanding Random Forests V3.1*. University of California, Berkeley, 2002.

Iain Brown and Christophe Mues. An experimental comparison of classification algorithms for imbalanced credit scoring data sets. *Expert Systems with Applications*, 39(3):3446–3453, 2012.

Chao Chen, Andy Liaw, and Leo Breiman. Using random forest to learn imbalanced data. Technical report, University of California, Berkeley, 2004.

Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Machine Learning*, 20:273–297, 1995.

Yuliya S. Demyanyk and Otto Van Hemert. Understanding the subprime mortgage crisis. *The Review of Financial Studies*, 24(6):1848–1880, May 2011.

Georges Dionne. Risk management: History, definition, and critique. *Risk Management and Insurance Review*, 16(2):147–166, 2013.

Craig K. Enders. *Applied Missing Data Analysis*. The Guilford Press, 2010.

Tom Fawcett. An introduction to roc analysis. *Pattern Recognition Letters*, 27(8):861–874, 2006.

R. Florez-Lopez. Effects of missing data in credit risk scoring. a comparative analysis of methods to achieve robustness in the absence of sufficient data. *The Journal of the Operational Research Society*, 61(3), 2010.

Jerome H. Friedman, Robert Tibshirani, and Trevor Hastie. *The Elements of Statistical Learning*. Springer Series in Statistics, 2001.

John W Graham, Allison E Olchowski, and Tamika D. Gilreath. How many imputations are really needed? some practical clarifications of multiple imputation theory. *Prevention Science*, 8(3): 206–213, September 2007.

Hans Hofmann. German credit data. Institut für Statistik und Ökonometrie - Universität Hamburg, 1994. URL `https://archive.ics.uci.edu/ml/datasets/Statlog+(German+Credit+Data)`.

Jeff Holt. A summary of the primary causes of the housing bubble and the resulting credit crisis: A non-technical paper. *The Journal of Business Inquiry*, 2009.

Cheng-Lung Huang, Mu-Chen Chen, and Chieh-Jen Wang. Credit scoring with a data mining approach based on support vector machines. *Expert Systems with Applications*, 33(4):847–856, November 2007.

Gareth James, Daniela Witten, Trevor Hastie, and Robert Tibshirani. *An Introduction to Statistical Learning with Applications in R*. Springer, 2013.

Alexandros Karatzoglou, Alex Smola, Kurt Hornik, and Achim Zeileis. kernlab – an S4 package for kernel methods in R. *Journal of Statistical Software*, 11(9), 2004. URL `http://www.jstatsoft.org/v11/i09/`.

Max Kuhn, Jed Wing, Steve Weston, Andre Williams, Chris Keefer, Allan Engelhardt, Tony Cooper, Zachary Mayer, Brenton Kenkel, the R Core Team, Michael Benesty, Reynald Lescarbeau, Andrew Ziem, Luca Scrucca, Yuan Tang, Can Candan, and Tyler Hunt. *caret: Classification and Regression Training*, 2017. URL `{https://CRAN.R-project.org/`

package=caret}and{http://topepo.github.io/caret/index.html}. R package version 6.0-76.

Stefan Lessmann, Bart Baesens, Hsin-Vonn Seow, and Lyn C. Thomas. Benchmarking state-of-the-art classification algorithms for credit scoring: A ten-year update. *European Journal of Operational Research*, 247(1):124–136, 2015.

Andy Liaw and Matthew Wiener. Classification and regression by randomforest. *R News*, 2(3): 18–22, 2002. URL http://CRAN.R-project.org/doc/Rnews/.

M. Lichman. UCI machine learning repository. University of California, Irvine, School of Information and Computer Sciences, 2013. URL http://archive.ics.uci.edu/ml.

Roderick J. A. Little. Missing-data adjustments in large surveys. *Journal of Business & Economic Statistics*, 6(3):287–296, 1988.

Wei-Yin Loh. Classification and regression trees. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 1(1):14–23, 2011.

Wes McKinney. Data structures for statistical computing in python. In *Proceedings of the 9th Python in Science Conference*, pages 51 – 56, 2010.

Robert C. Merton. On the pricing of corporate debt: The risk structure of interest rates. *The Journal of Finance*, 29(2):449–470, May 1974.

David Meyer. Support vector machines, the interface to libsvm in package e1071. FH Technikum Wien, Austria, August 2015.

David Meyer, Evgenia Dimitriadou, Kurt Hornik, Andreas Weingessel, and Friedrich Leisch. *e1071: Misc Functions of the Department of Statistics, Probability Theory Group (Formerly: E1071), TU Wien*, 2017. URL https://CRAN.R-project.org/package=e1071.

Tim P Morris, Ian R White, and Patrick Royston. Tuning multiple imputation by predictive mean matching and local residual draws. *BMC Medical Research Methodology*, 2014.

Oracle Corporation. MySQL 5.7, 2017. URL http://www.mysql.com/.

Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, Jake Vanderplas, Alexandre Passos, David Cournapeau, Matthieu Brucher, Matthieu Perrot, and Édouard

Duchesnay. Scikit-learn: Machine learning in python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

John C. Platt. Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods. In *Advances in Large Margin Classifiers*, pages 61–74. MIT Press, 1999.

Python Software Foundation. Python 3.6.3, 2017. URL `http://www.python.org`.

R Core Team. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria, 2017. URL `https://www.R-project.org/`.

Donald B. Rubin. Statistical matching using file concatenation with adjusted weights and multiple imputations. *Journal of Business & Economic Statistics*, 4(1):87–94, 1986.

Donald B. Rubin. *Multiple Imputation for Nonresponse in Surveys*. John Wiley & Sons, 1987.

T. Sing, O. Sander, N. Beerenwinkel, and T. Lengauer. ROCR: visualizing classifier performance in r. *Bioinformatics*, 21(20), 2005. URL `http://rocr.bioinf.mpi-sb.mpg.de`.

Carolin Strobl, Anne-Laure Boulesteix, Achim Zeileis, and Torsten Hothorn. Bias in random forest variable importance measures: Illustrations, sources and a solution. *BMC Bioinformatics*, 8(1), 2007.

S. Chris Colbert Stéfan van der Walt and Gaël Varoquaux. The numpy array: A structure for efficient numerical computation. *Computing in Science & Engineering*, 13:22–30, 2011.

US Federal Reserve. Z.1 financial accounts of the united states. Federal Reserve Statistical Release, June 2017. URL `https://www.federalreserve.gov/releases/z1/`.

Stef van Buuren. Multiple imputation of discrete and continuous data by fully conditional specification. *Statistical Methods in Medical Research*, 16:219–242, 2007.

Stef van Buuren. *Flexible Imputation of Missing Data*. Chapman and Hall/CRC, 2012.

Stef van Buuren and Karin Groothuis-Oudshoorn. mice: Multivariate imputation by chained equations in r. *Journal of Statistical Software*, 45(3), 2011. URL `http://www.jstatsoft.org/v45/i03/`.

Stef van Buuren and Karin Oudshoorn. Flexible multivariate imputation by mice. TNO Prevention and Health, October 1999.

Gerko Vink, Laurence E. Frank, Jeroen Pannekoek, and Stef van Buuren. Predictive mean matching imputation of semicontinuous variables. *Statistica Neerlandica*, 68(1):61–90, 2014.

Gang Wang, Jian Ma, Lihua Huang, and Kaiquan Xu. Two credit scoring models based on dual strategy ensemble trees. *Knowledge-Based Systems*, 2011.

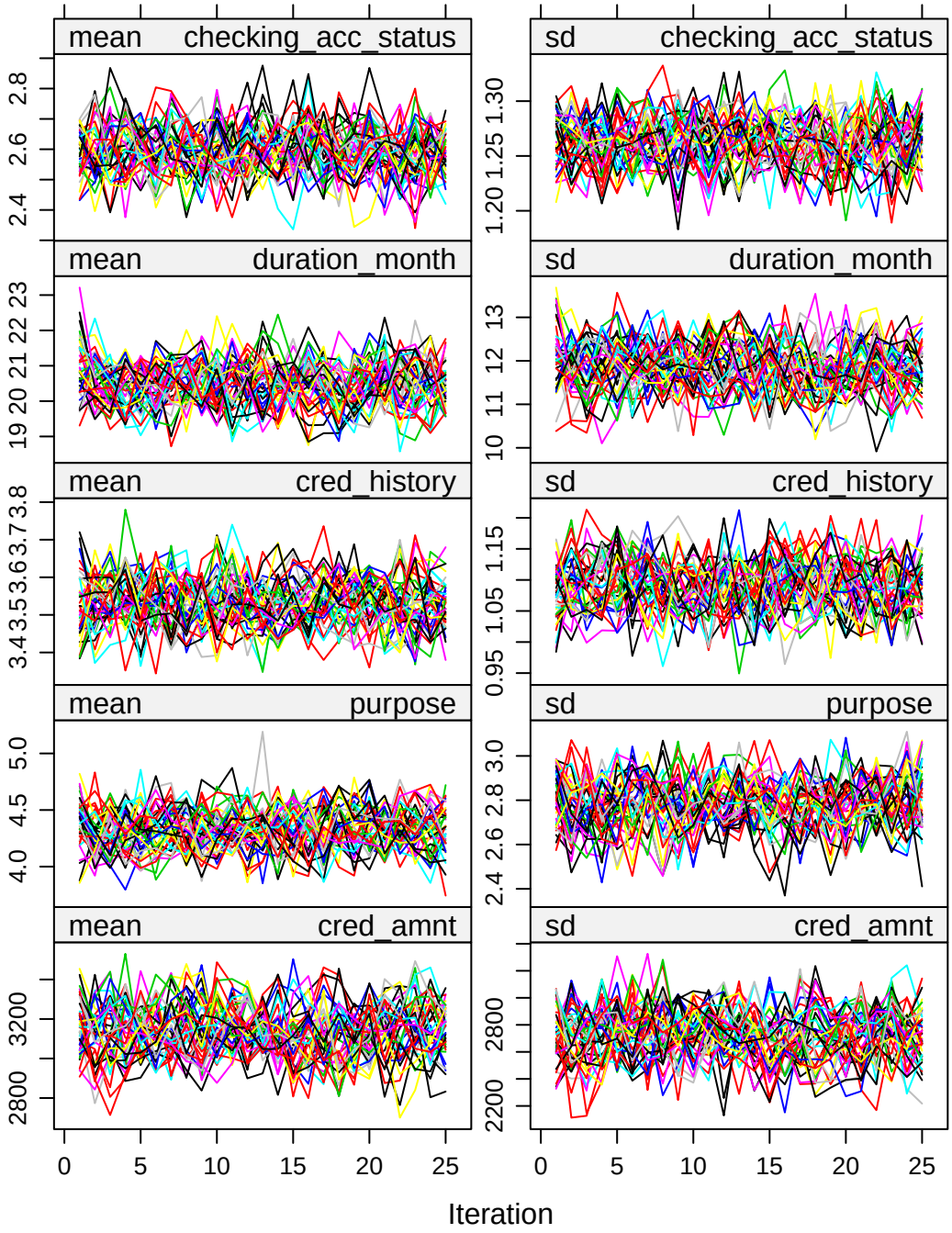
Hadley Wickham. *ggplot2: Elegant Graphics for Data Analysis*. Springer-Verlag New York, 2009.

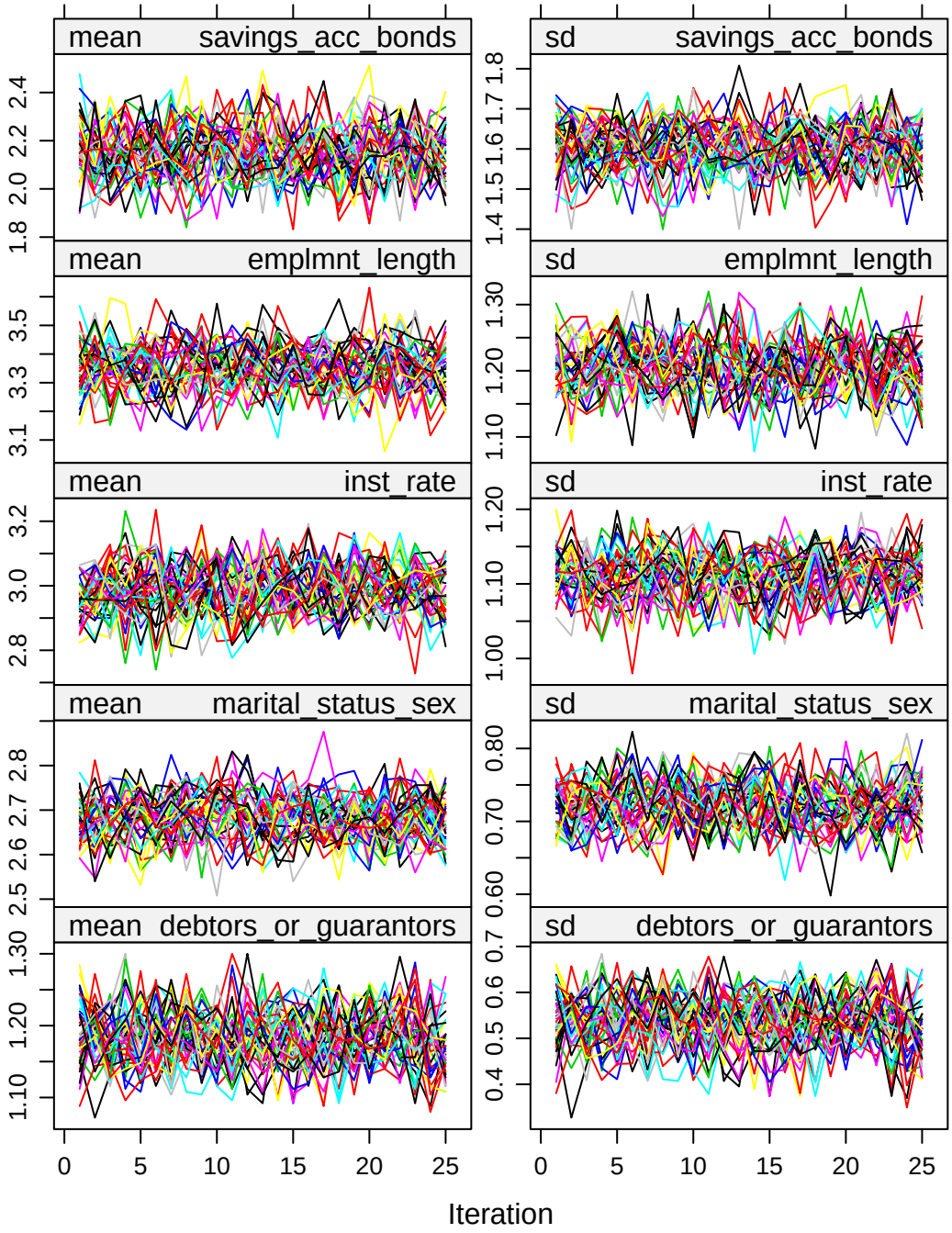
Angela M. Wood, Ian R. White, and Patrick Royston. How should variable selection be performed with multiply imputed data? *Statistics in Medicine*, 27(17):3227–3246, 2008.

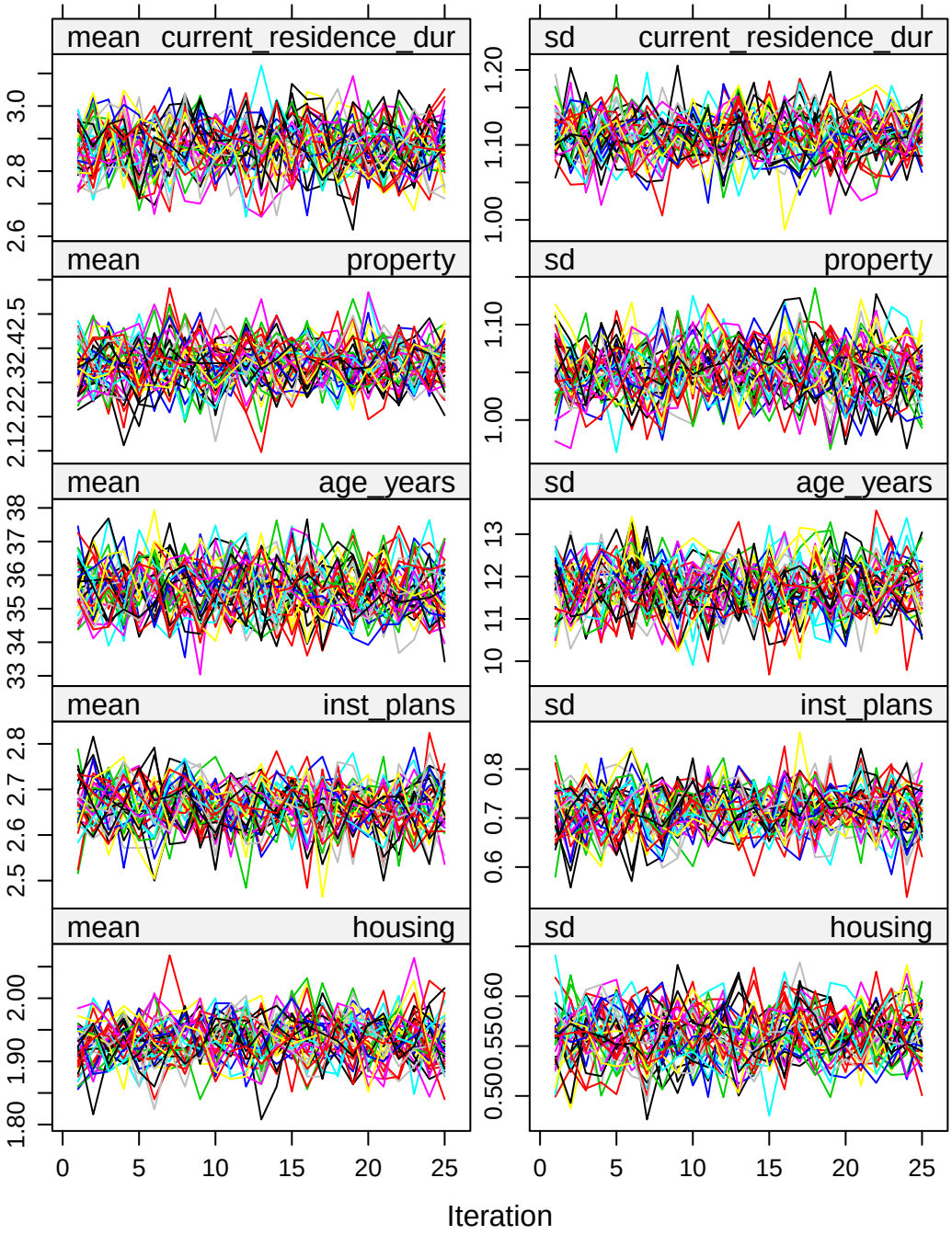
Appendices

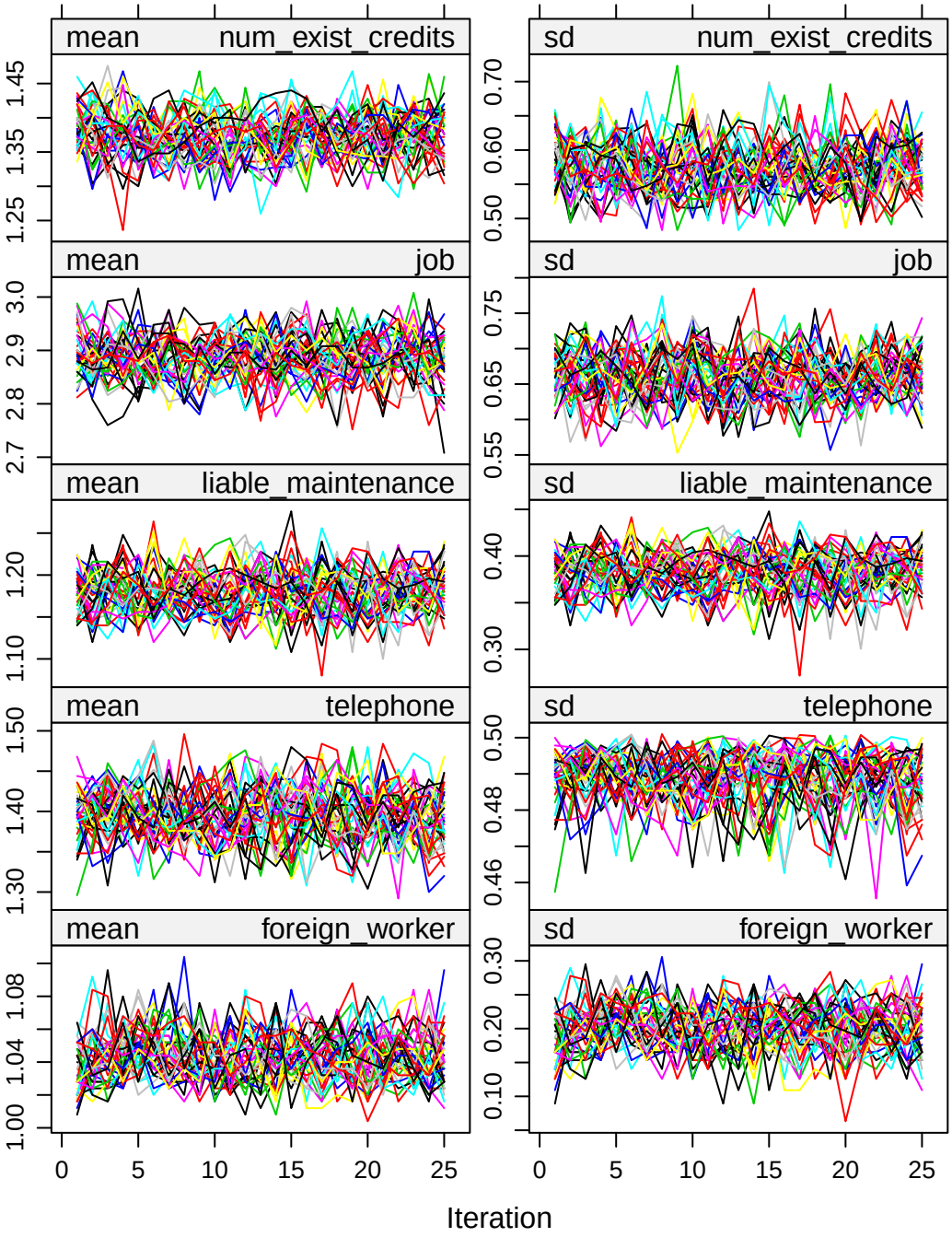
Appendix A

Stream Line Plots


mean checking_acc_status
sd checking_acc_status
mean duration_month
sd duration_month
mean cred_history
sd cred_history
mean purpose
sd purpose
mean cred_amnt
sd cred_amnt
Iteration

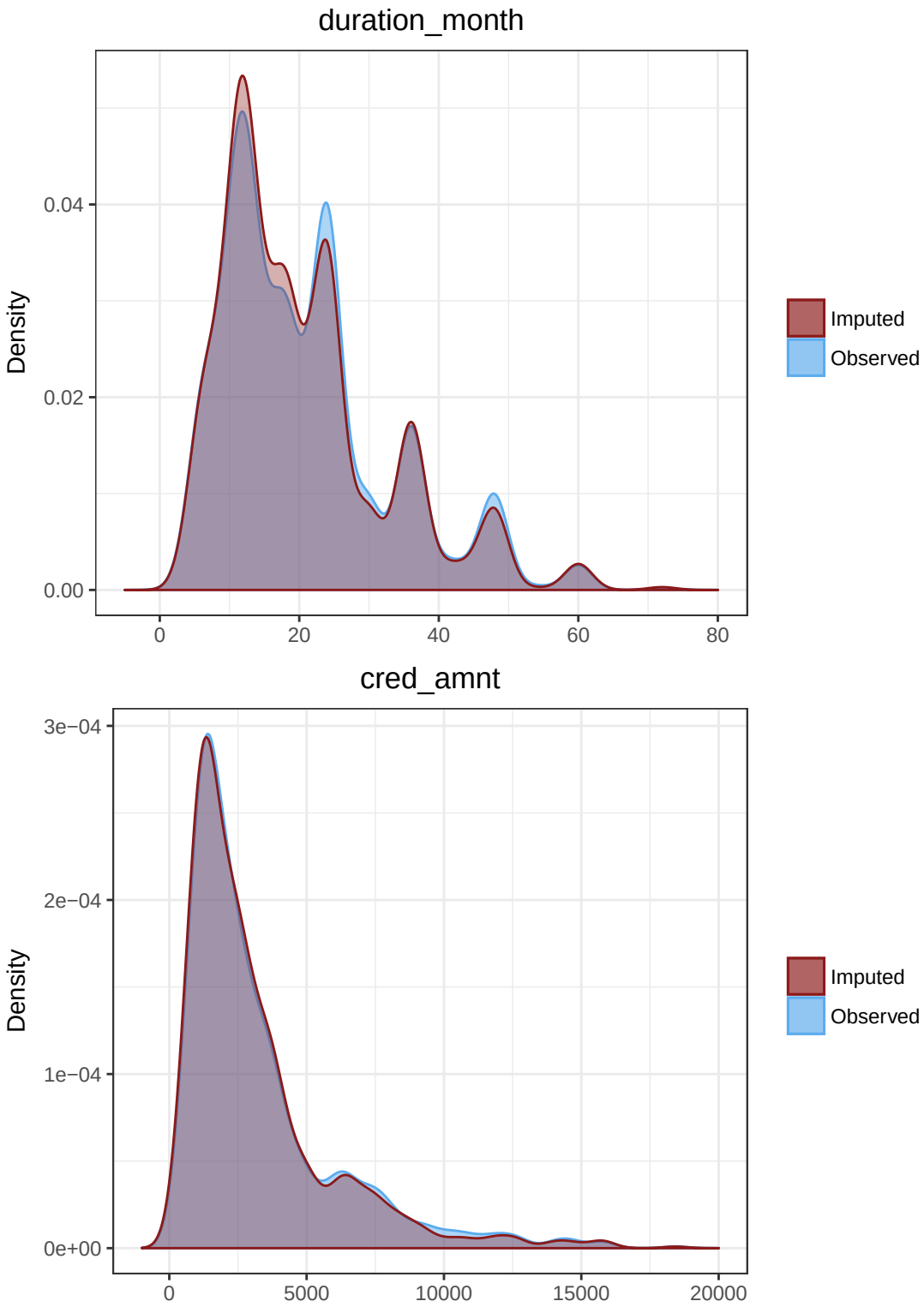

mean savings_acc_bonds
sd savings_acc_bonds
mean emplmnt_length
sd emplmnt_length
mean inst_rate
sd inst_rate
mean marital_status_sex
sd marital_status_sex
mean debtors_or_guarantors
sd debtors_or_guarantors
Iteration

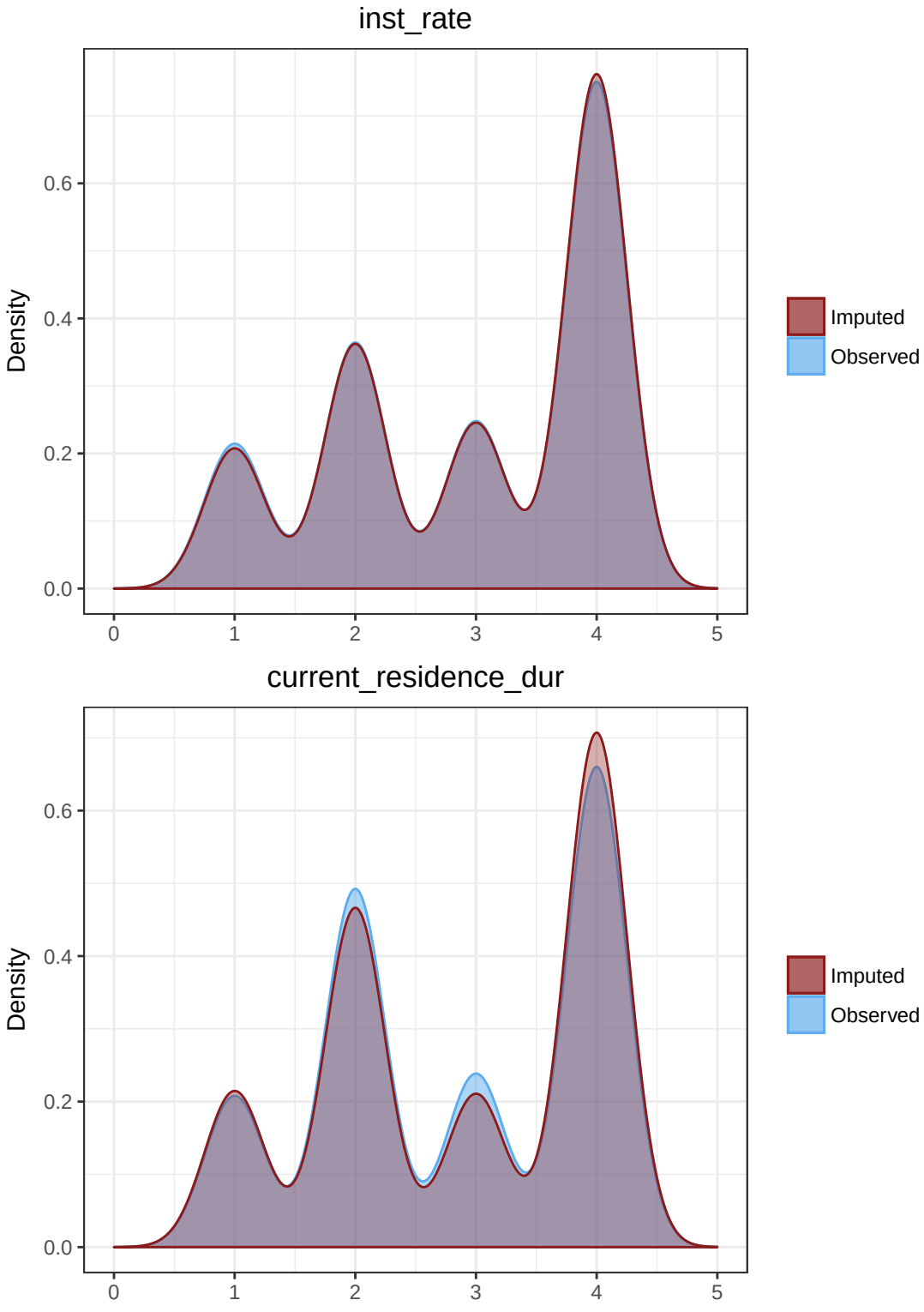

mean current_residence_dur
sd current_residence_dur
mean property
sd property
mean age_years
sd age_years
mean inst_plans
sd inst_plans
mean housing
sd housing
Iteration

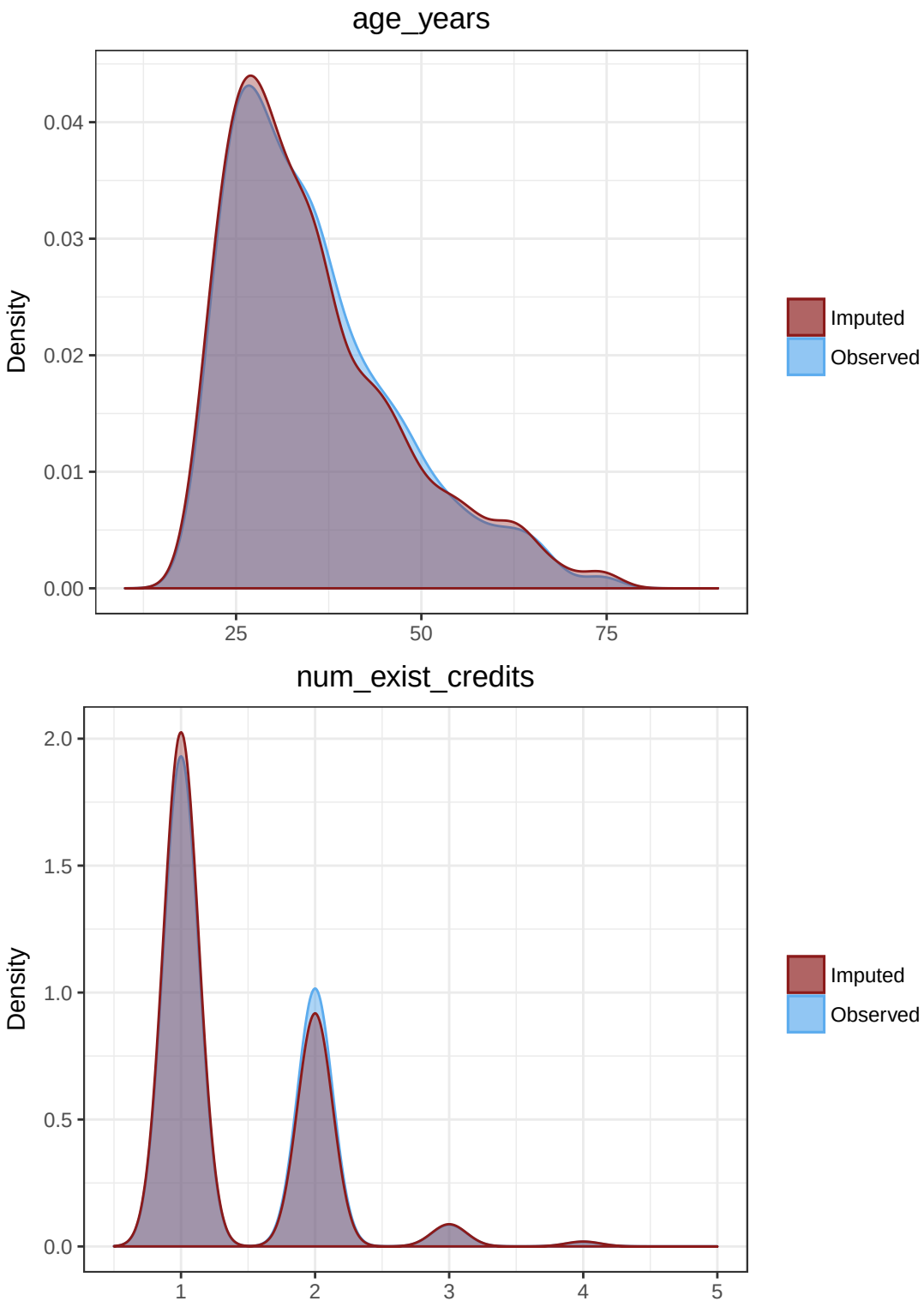

mean num_exist_credits
sd num_exist_credits
mean job
sd job
mean liable_maintenance
sd liable_maintenance
mean telephone
sd telephone
mean foreign_worker
sd foreign_worker
1.25 1.35 1.45
0.50 0.60 0.70
2.7 2.8 2.9 3.0
0.55 0.65 0.75
1.10 1.20
0.30 0.40
1.30 1.40 1.50
0.46 0.48 0.50
1.00 1.04 1.08
0.10 0.20 0.30
0 5 10 15 20 25
Iteration

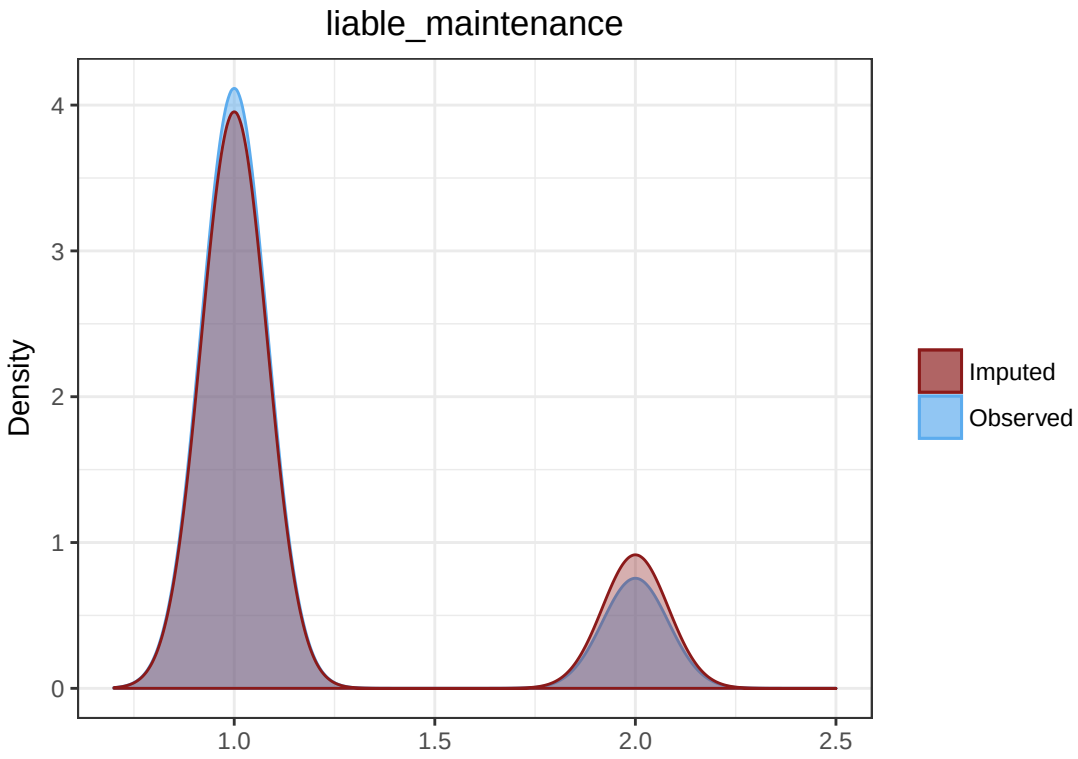
Appendix B

Density Plots


duration_month
Density
0.04
0.02
0.00
0
20
40
60
80
Imputed
Observed
cred_amnt
Density
3e−04
2e−04
1e−04
0e+00
0
5000
10000
15000
20000
Imputed
Observed

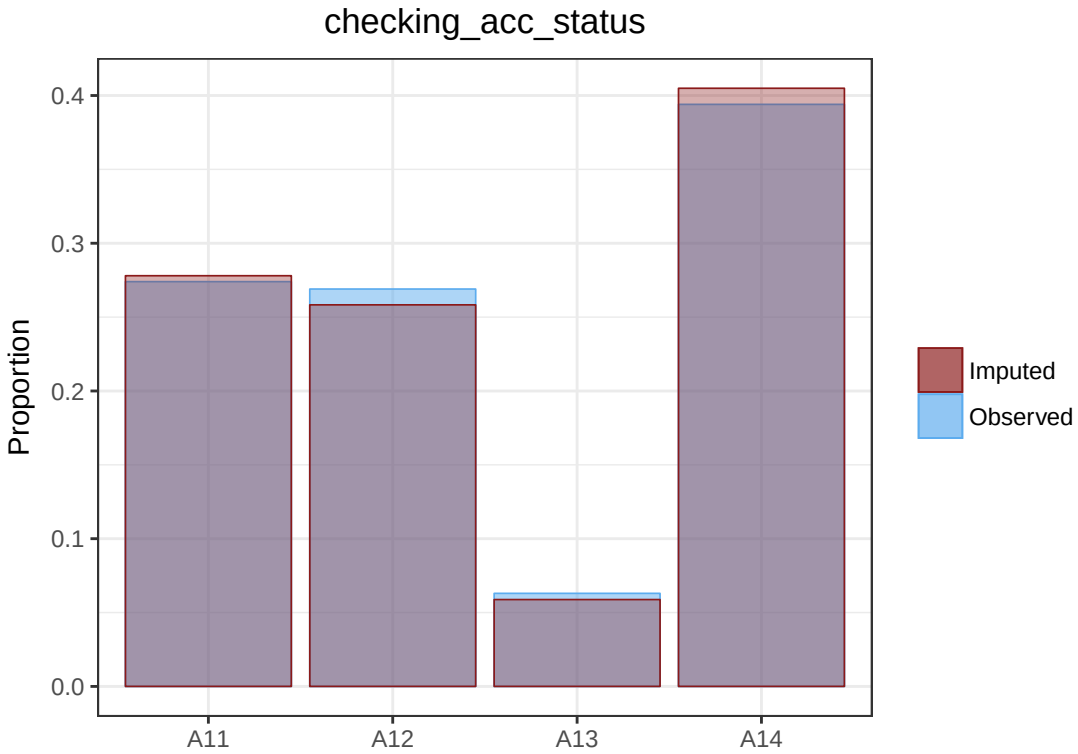

inst_rate
Density
0.0
0.2
0.4
0.6
0
1
2
3
4
5
Imputed
Observed
current_residence_dur
Density
0.0
0.2
0.4
0.6
0
1
2
3
4
5
Imputed
Observed

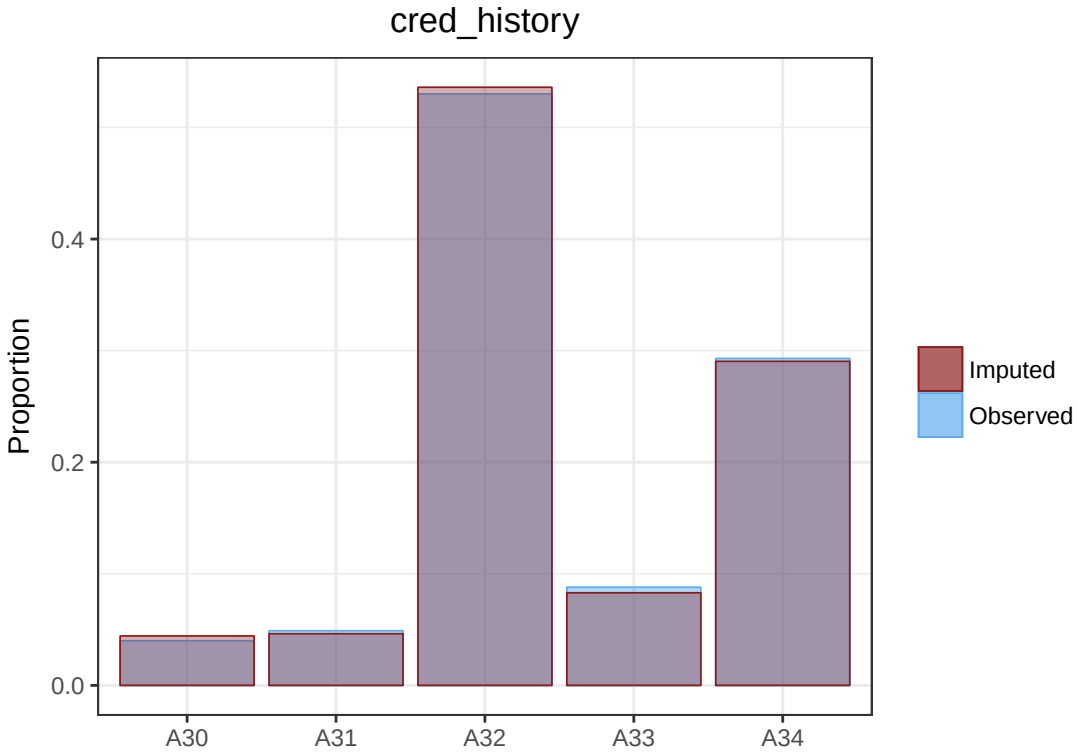

age_years
Density
0.04
0.03
0.02
0.01
0.00
25
50
75
Imputed
Observed
num_exist_credits
Density
2.0
1.5
1.0
0.5
0.0
1
2
3
4
5
Imputed
Observed

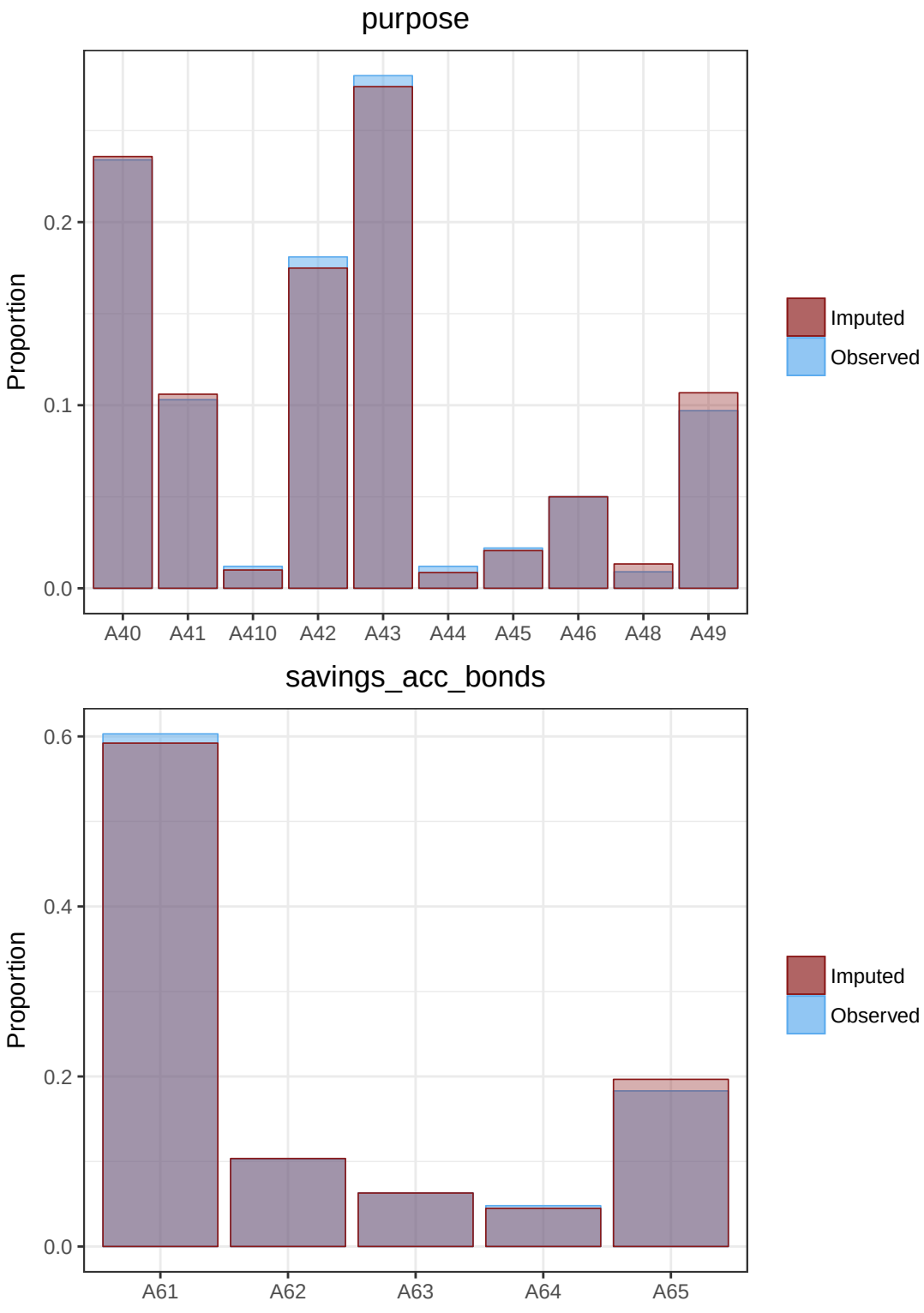

liable_maintenance
Density
0
1
2
3
4
1.0
1.5
2.0
2.5
Imputed
Observed

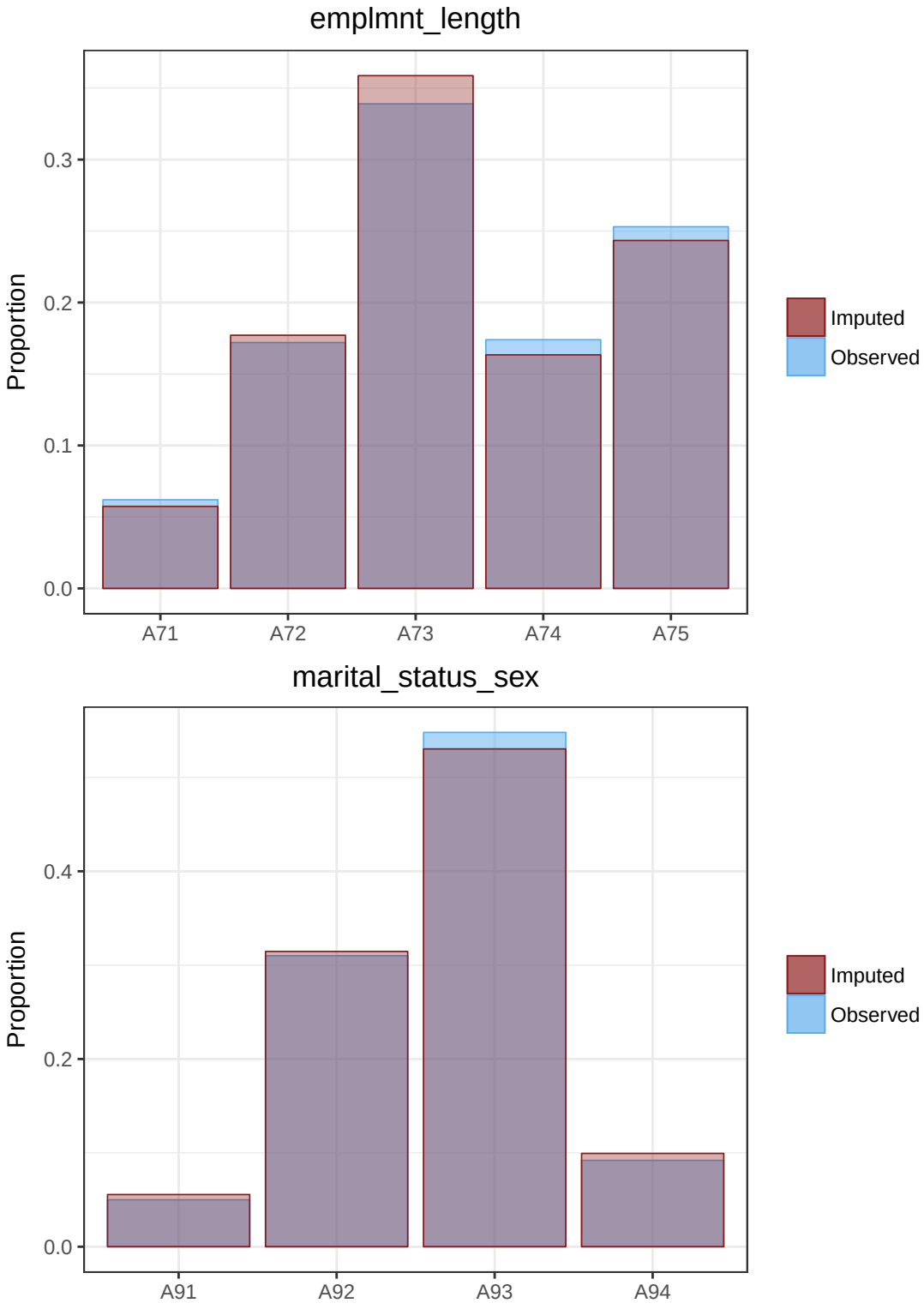
Appendix C

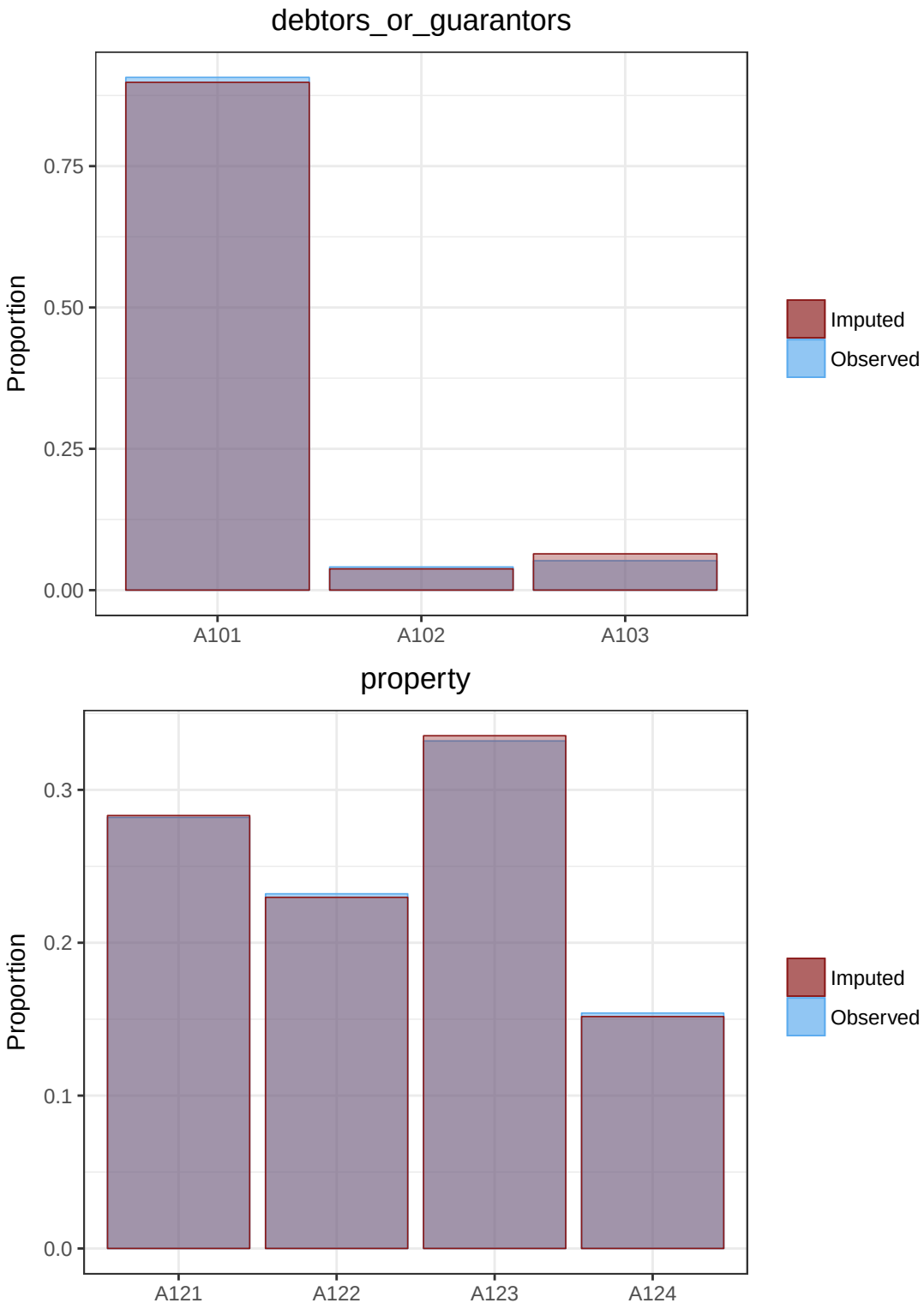
Frequency Plots

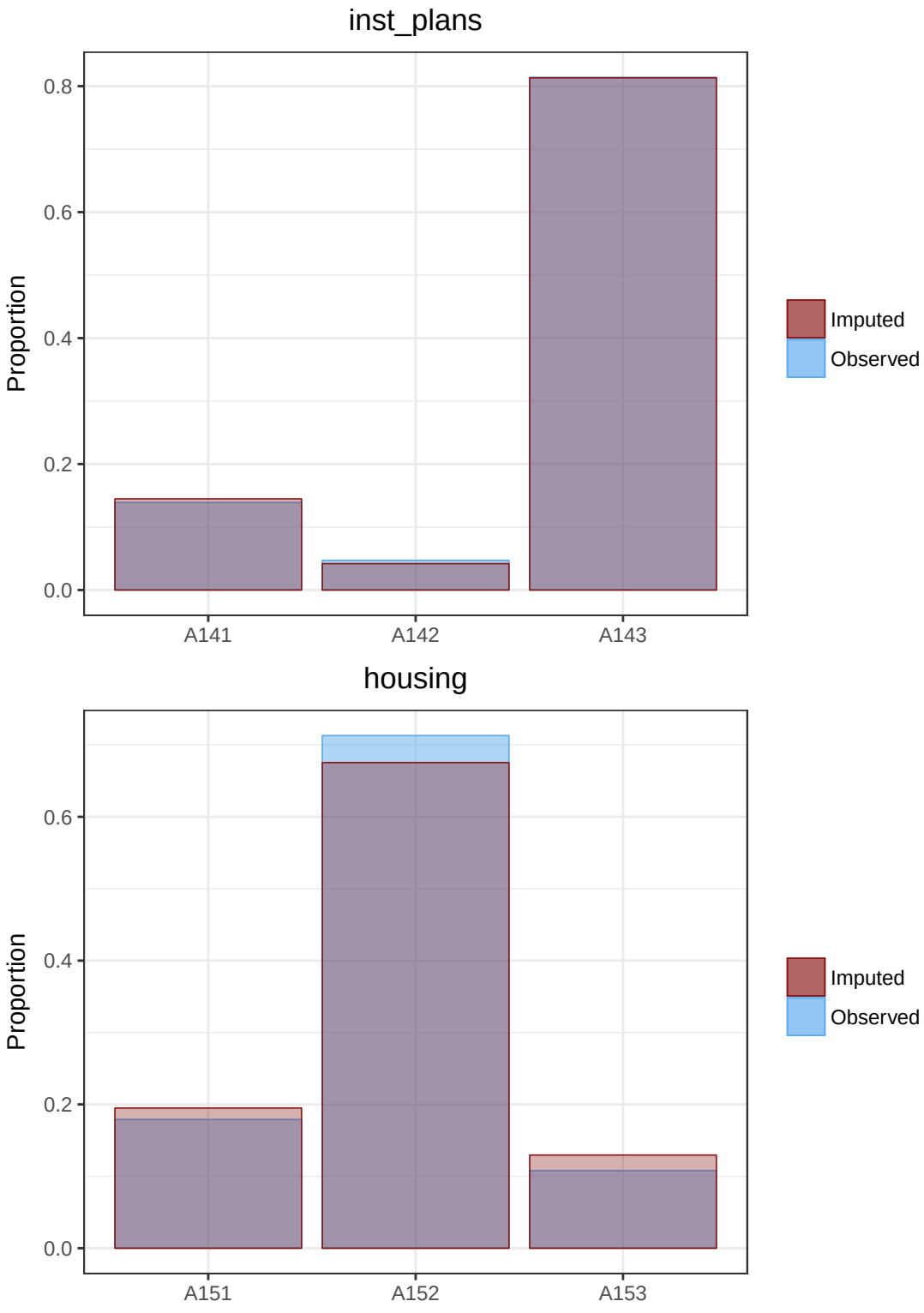

checking_acc_status
Proportion
0.0
0.1
0.2
0.3
0.4
A11
A12
A13
A14
Imputed
Observed

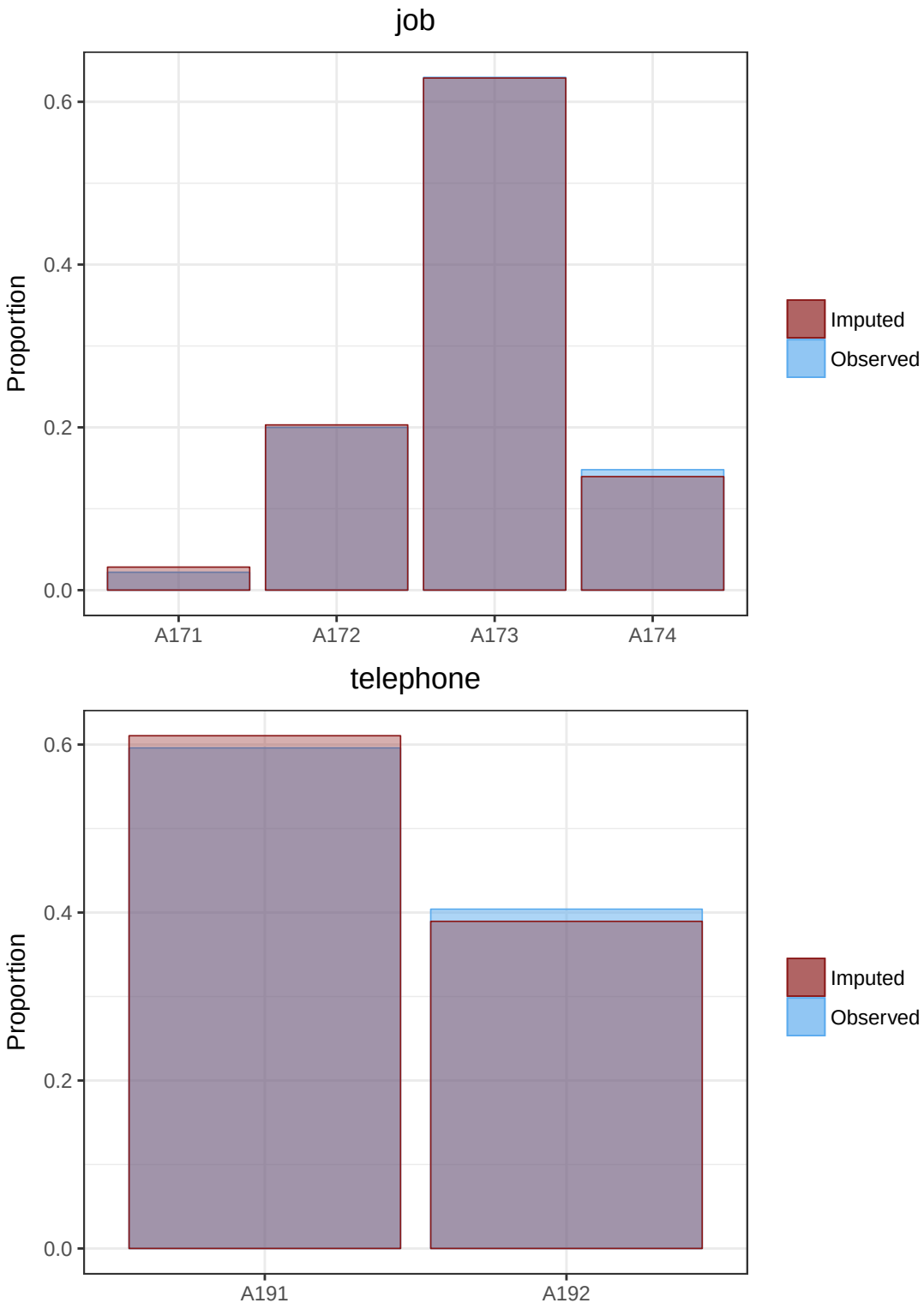

cred_history
Proportion
0.0
0.2
0.4
A30
A31
A32
A33
A34
Imputed
Observed

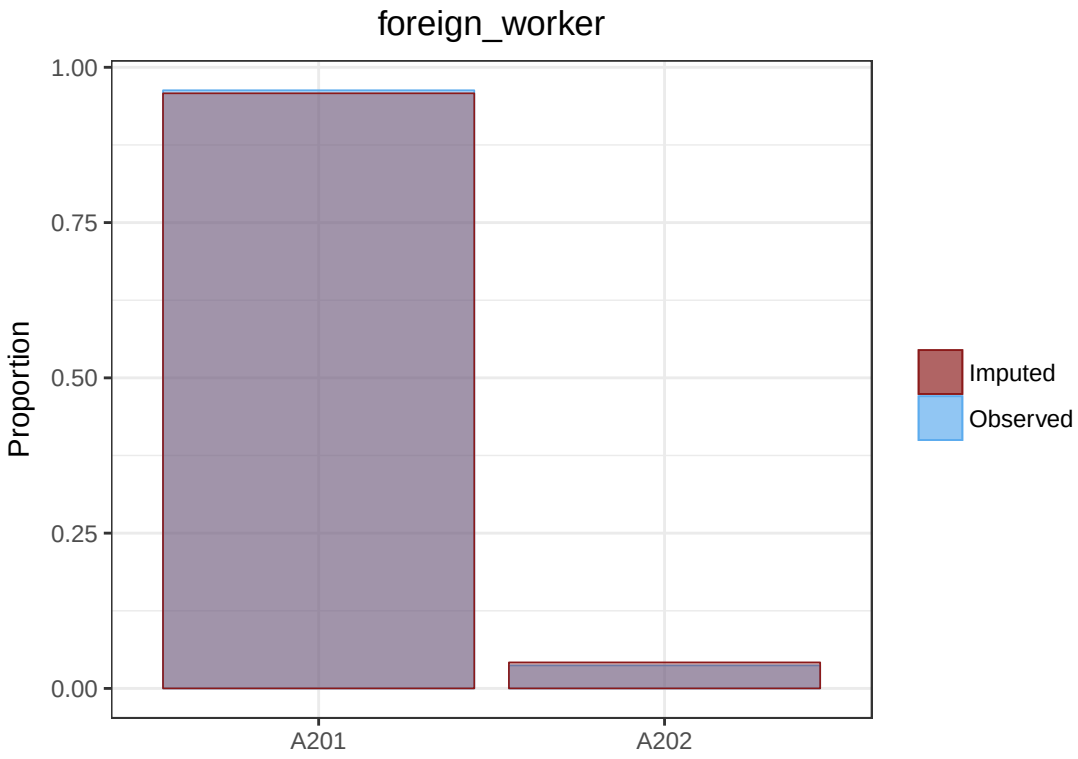

purpose
Proportion
0.0
0.1
0.2
A40
A41
A410
A42
A43
A44
A45
A46
A48
A49
Imputed
Observed
savings_acc_bonds
Proportion
0.0
0.2
0.4
0.6
A61
A62
A63
A64
A65
Imputed
Observed


emplmnt_length
Proportion
0.0
0.1
0.2
0.3
A71
A72
A73
A74
A75
Imputed
Observed
marital_status_sex
Proportion
0.0
0.2
0.4
A91
A92
A93
A94
Imputed
Observed


debtors_or_guarantors
Proportion
0.75
0.50
0.25
0.00
A101
A102
A103
Imputed
Observed
property
Proportion
0.3
0.2
0.1
0.0
A121
A122
A123
A124
Imputed
Observed


inst_plans
Proportion
0.0
0.2
0.4
0.6
0.8
A141
A142
A143
Imputed
Observed
housing
Proportion
0.0
0.2
0.4
0.6
A151
A152
A153
Imputed
Observed


job
Proportion
0.6
0.4
0.2
0.0
A171
A172
A173
A174
Imputed
Observed
telephone
Proportion
0.6
0.4
0.2
0.0
A191
A192
Imputed
Observed


foreign_worker
1.00
0.75
0.50
0.25
0.00
Proportion
A201
A202
Imputed
Observed

Appendix D

Abbreviations

AUC Area under curve

DM Deutsche mark

MAR Missing at random

MCAR Missing completely at random

MI Multiple imputation

MICE Multiple imputation by chained equations

MNAR Missing not at random

PMM Predictive mean matching

RBF Radial basis function

RF Random forest

RFE Recursive feature elimination

ROC Receiver operation characteristic curve

SVM Support vector machine

Appendix E

Software

Data Manipulation MySQL 5.7 (Oracle Corporation, 2017).

General Programming R statistical language and Python (R Core Team, 2017)(Python Software Foundation, 2017).

MICE The *mice* R package (van Buuren and Groothuis-Oudshoorn, 2011).

Model Validation *scikit-learn*, *SciPy*, *NumPy* and *pandas* Python libraries (Pedregosa et al., 2011)(Stéfan van der Walt and Varoquaux, 2011)(McKinney, 2010).

Plots The *ggplot2* R package (Wickham, 2009).

RFs The *randomForest* R package (Liaw and Wiener, 2002).

RFE and Tuning The *caret* R package (Kuhn et al., 2017).

ROC and AUC The *ROCR* R package (Sing et al., 2005).

SVMs The *e1071* and *kernlab* R packages (Meyer et al., 2017)(Karatzoglou et al., 2004).