Give Me Some Credit.

November 12th, 2020

Emma Godfrey

Alexander Bartone

Nathaniel Tolles

Tom Clarke

Introduction

Financial institutions, like banks, provide loans for people or businesses in need of cash in the short term. Loans must be paid back over time, either in a lump sum at maturity compounded by the interest rate or through smaller periodic interest payments, often followed by repayment of the principal at maturity. Banks structure loans such that the loan has a positive net present value from the perspective of the bank; the discounted sum of the future cash flows the bank expects to receive from issuing the loan must be greater than the amount initially loaned to the borrower. These cash flows, payments from the perspective of the borrower, are discounted at the cost of capital, or the expected rate of return the bank could be earning in other projects they could have invested the initial loan into instead. Thus, loans are structured with an interest rate such that it is profitable for the bank to issue loans.

However, borrowers do not always pay back the loans they take out. They might be late on payments, or default and fail to pay back the loan entirely. Thus, banks must weigh the cash flows they expect to receive according to the risk profile of the borrower. Traditionally, if a borrower has a low credit rating, and are judged to be likely to default, they would be charged a higher interest rate, or be denied a loan altogether.

In the early 2000s, bundled home mortgages, called "Mortgage Backed Securities", a form of "Collateralized Debt Obligation", became an increasingly popular derivative. These CDOs were made into different tiers based upon their risk profile, with a higher yield for those deemed more risky. With the housing market apparently booming, many investors wanted a piece of all of the new mortgages being issued. Besides, who doesn't pay their mortgage? Credit rating agencies were shown to have given risky MBS's ratings that understated their riskiness. On top of this, MBS's were 'insured' by 'Credit Default Swaps', where the holder of an MBS would send periodic payments to another party that would reimburse them if the MBS went south. While you can only insure your house a single time, because of incentive problems, an MBS would often be insured by a number of CDS's, such that the value of CDS's came to be far larger than the MBS's themselves. Essentially, financial institutions and investors everywhere doubled, tripled, and quadrupled down on whether the mortgages within the MBS would default. Having a myopic view of risk for a number of years led to the greatest financial crisis since the Great Depression.

In this project, we attempt to build a model that will assist banks in assessing who they should or should not give loans to based upon their risk profile. Using a variety of variables, our aim is to train a learner such that it accurately returns the probability one might default based upon data we have access to. More specifically, we will attempt to predict whether or not people of the test data will be 90 days past due or worse. The predictor variables are defined as follows. 'RevolvingUtilizationOfUnsecuredLines' is the total balance on credit cards and personal lines of credit excluding real estate and installment debt (car loans, etc.) divided by the sum of credit limits. We also have access to the age of the borrower. The

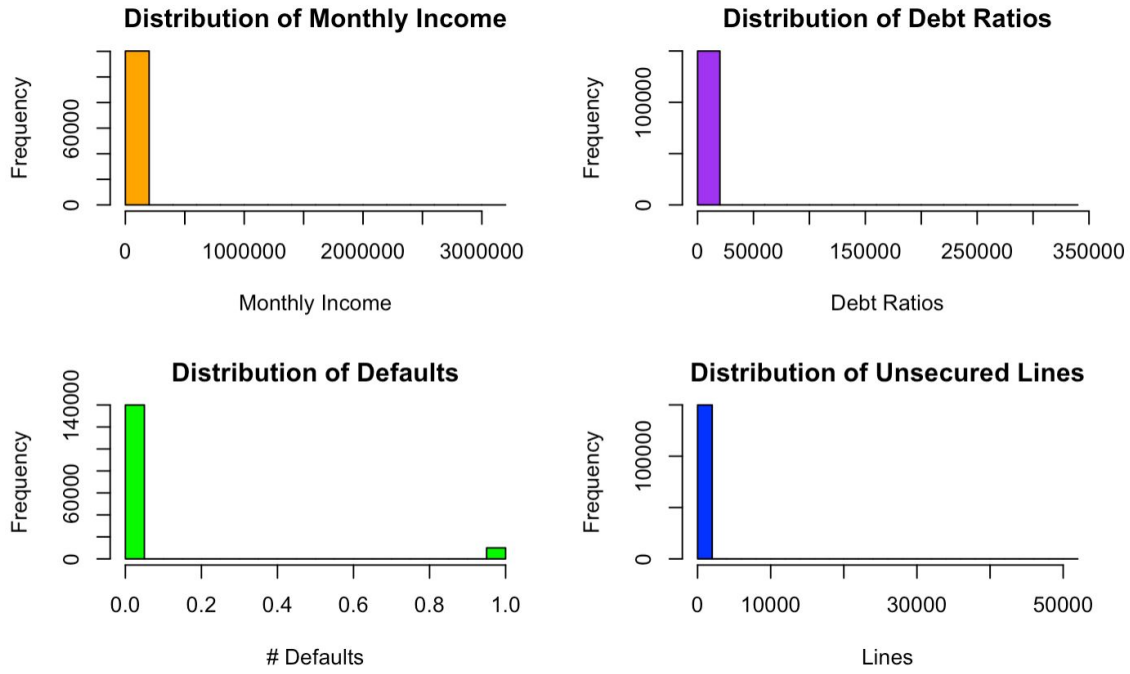
number of times a borrower has been late on a payment by 30-59 days, 60-89 days, and 90 days or more, are expressed in 3 different integer variables. The 'DebtRatio' is calculated by summing monthly debt payments, alimony, and living costs, and dividing the sum by 'MonthlyIncome', which is also a variable included in the data. 'NumberOfOpenCreditLinesAndLoans' refers to the number of open loans and lines of credit an individual has. 'NumberOfRealEstateLoansOrLines' includes the number of mortgages, real estate loans, and the credit lines secured against the value of one's home. 'NumberOfDependents' refers to the number of people in their family that rely on them financially (excluding themselves).

Exploratory Data Analysis

To get a sense of what our data looks like, we can explore some basic summary statistics. Below we have both the mean and median values for interesting variables in the original dataset.

	Serious deliquency in 2 years	unsecured revolving lines	age	debt ratio	monthly income	# open lines and loans	real estate loans or lines	# of dependents
mean	0.0668	6.048	52.3	353	6670	8.45	1.02	0.757
median	0	0.154	52	0.367	5400	8	1	0

We can see that for a few of the variables -- serious delinquency in 2 years, value of unsecured revolving lines, debt ratio, and number of dependents -- that the mean and median are extremely different. This is an indication of outliers, or at least extreme values, in the data. We can further see the skewed distributions via the histograms below.

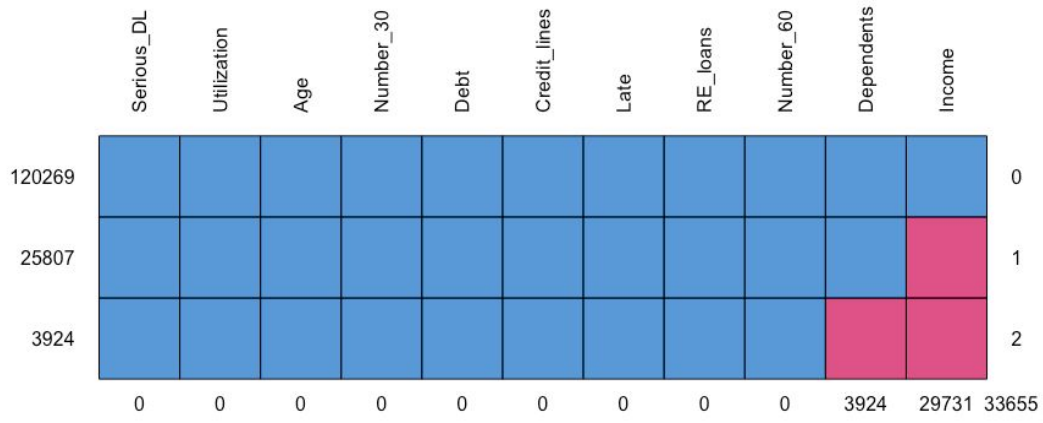


For monthly income, debt ratio, and unsecured lines, the overwhelming majority of the data is concentrated to the lower end of the total range. However, the ranges are extremely large compared to the median values. The median debt ratio is .367, but the histogram indicates a range up to 350,000. These are likely outliers in the data; we shed light on outliers mediation and imputation techniques in the next section. Lastly, we notice that the variable we are interested in predicting -- default or serious delinquency in 2 years, a binary variable -- has mostly zeros. This is a sign of imbalanced data, which is covered in more depth in a later section.

Methods and Results

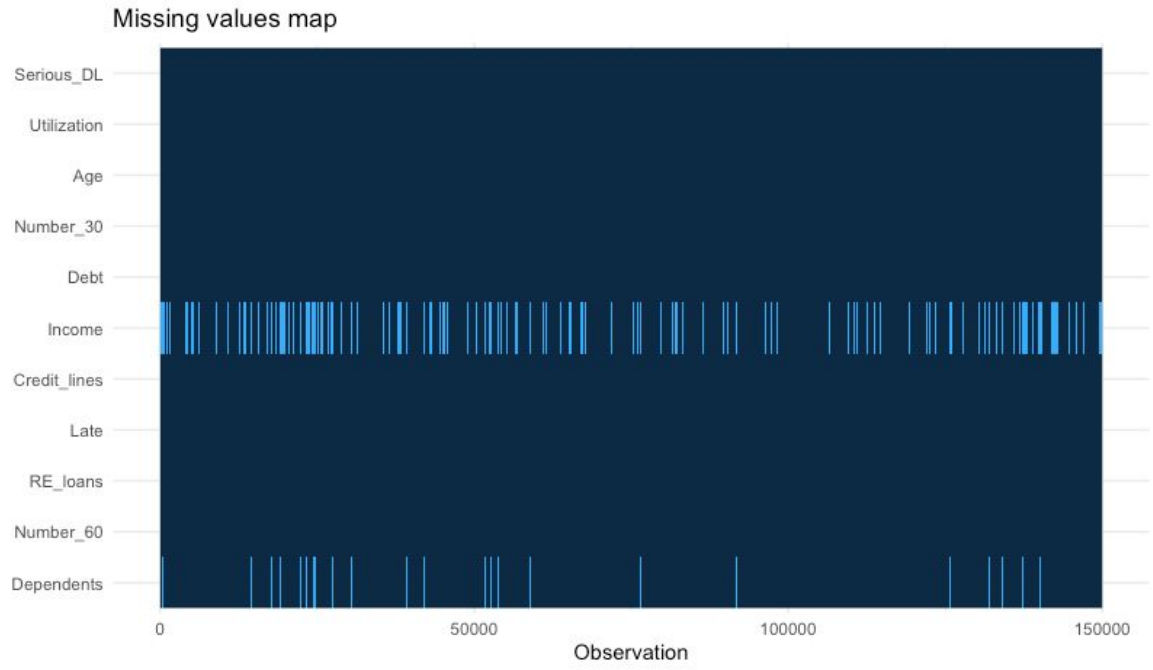
(a) Data Imputation

The training data set contains over 33,000 missing values. More specifically, there are 29,731 observations which are missing monthly income and 3,924 observations missing the number of dependents. Of the 150,000 observations in the data set, 25,807 observations are just missing monthly income. Interestingly, all 3,924 observations which are missing the number of dependents are also missing monthly income. This is summarized in the figure below, which shows missing observations in red.



Not only does missing data present issues when selecting a classification learner, but also the missingness of the data could present meaning in itself. First, we analyze whether the monthly income and number of dependents missing data is missing completely at random (MCAR), missing at random (MAR) or missing not at random (MNAR). If the data is MCAR, then the probability for missingness is the same for all observations. If data is MAR, then the probability of missingness for a variable x is independent of the value of x itself, as it is instead dependent on another explanatory variable in the data set . On the other hand, if the data is MNAR, then there is a relationship between the propensity of a value to be missing

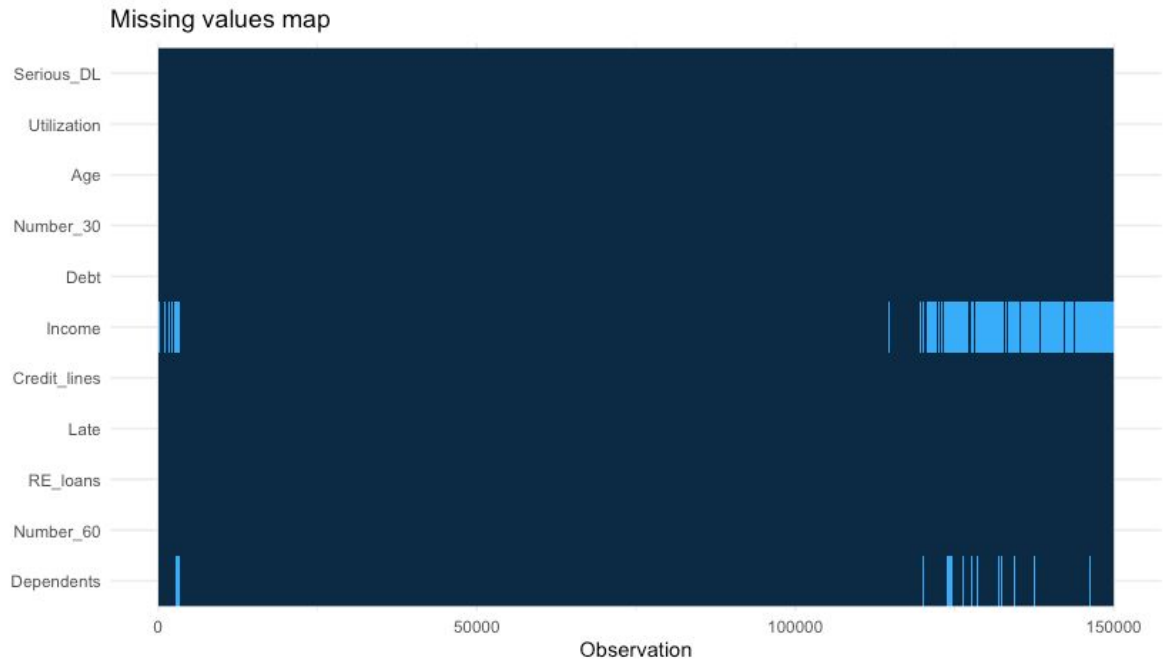
and the actual value itself (which is missing and therefore unknown).[1] While it is difficult to classify data as being either MAR or MNAR, it is better to think of data existing on a continuum of MAR to MNAR. We test for MCAR, MNAR, and MAR by ordering the data by ascending value of each explanatory variable. The 'finalfit' package allows us to create a map of the missing value by observation. Here is the first plot we did, which sorts the data set by ascending order of Revolving Utilization of Unsecured Lines of Credit. The missing values, which are represented by the light blue lines, are scattered throughout. There is no clear pattern with the missingness when ordered by ascending value of Revolving Utilization of Unsecured Lines.



We use this same method to test missingness for the other explanatory variables. Every plot looked similar to the plot above (no distinguishable pattern) except for the plot when ordered by ascending value

[1] Stef van Buuren, *Flexible Imputation of Missing Data* (CRC Press, 2018), https://stefvanbuuren.name/fimd/sec-MCAR.html.

of Debt Ratio. This plot is shown below.



From the above figure, there is clearly a relationship between debt ratio and the missingness of monthly income; when monthly income is missing, we see significantly higher debt ratios. While this relationship may be an indication of data MAR, it is important to note the underlying relationship between monthly income and calculating one's debt ratio. Debt ratio is calculated by dividing one's monthly debt payments, alimony, and living costs by one's monthly income. Thus, missing monthly incomes would cause extraneous debt ratios. In conclusion, while there seems to be a relationship between the debt ratio and the missingness of monthly income, it is likely a result of using the monthly income in calculating the debt ratio.

However, due to the limited knowledge of the dataset, it is still important to create flexibility such that, if monthly income or number of dependents are MNAR, our learner will account for the systematic missingness. We will do this by creating missingness indicator variables for both monthly income and number of dependents. If the observation is missing income, then the indicator variable for missing income will take on a 1 and 0 otherwise. The process is the same for the indicator variable for the missing number of dependents. Missingness indicator variables allow us to impute the missing values while preserving the pattern of missingness should one exist. Now, let us turn our focus to imputation techniques.

While monthly income and number of dependents were the only variables with missing values, many variables have values which were either mis-entered or impractical. A monthly income of $1, for

example, is clearly incorrect. Similarly, the value 50708 is illogical for 'RevolvingUtilizationOfUnsecuredLines', given that this is calculated by dividing total balance on credit cards and personal lines of credit by the sum of credit limits. To account for these values, we initially tried to use the built-in outlier detection function within R, but found that it classified a significant portion of the data as outliers. After realizing this, we decided to use a bit more Bayesian thinking in determining what the cutoffs should be for data to be considered an outlier or entered incorrectly. For instance, 'RevolvingUtilizationOfUnsercuredLines' is a person's outstanding balance on credit cards and personal lines of credit divided by their respective credit limits, so any value above 1 seems improbable. There could be some room for leeway slightly, but upon looking at a histogram of values for 'RevolvingUtilizationOfUnsercuredLines', a threshold of 1.5 seemed to retain the vast majority of possible-seeming data. Anything above 1.5 was given a value of NA, and later imputed.

Regarding the variables 'NumberOfTime30-59DaysPastDueNotWorse', 'NumberOfTime60-89DaysPastDueNotWorse', and 'NumberOfTimes90DaysLate', the vast majority of observations had a value of 0, up to 24 as a maximum for 'NumberOfTime30-59DaysPastDueNotWorse'. However, each of these variables had a small number of observations that had 98 for these three variables, which to have been a default value or some sort of error. Thus, we applied a threshold value of 30 for each of these three, such that values above 30 would be given a value of NA and later imputed. Monthly income of over $50,000 or less than $600, 'NumberOfOpenCreditLinesAndLoans' of greater than 25, and 'NumberOfDependents' of greater than 7 were similarly treated as outliers and imputed.

Imputation is the process by which data scientists deal with missing data. Imputation techniques are seemingly abundant and data set specific. First, since our data set is of size n=150,000, it may seem logical to simply drop the observations with missing data and train our learner on the complete observations. However, our goal is not to best predict the training dataset, but rather to predict whether a person in the test dataset will default. We note that the test dataset contains a similar proportion of missing values; if we were to drop the incomplete observations in the test dataset, our model would not output a default prediction. While only considering complete cases is a valid imputation technique, it is likely the data we seek to classify will be of similar form of the training data and thus will contain missing values. Also, 33,000 missing values from the training data set represents over ⅛ of the total observations. This violates the rule of thumb that it is only acceptable to ignore missing data when the proportion of missing data is below approximately 5%.[2]

Next, let us consider the following median imputation technique. When an observation is missing monthly income, we assign the median monthly income as the previously missing income. We perform

[2] Janus Christian Jakobsen et al., "When and How Should Multiple Imputation Be Used for Handling Missing Data in Randomised Clinical Trials – a Practical Guide with Flowcharts," *BMC Medical Research Methodology* 17, no. 1 (December 6, 2017): 162, https://doi.org/10.1186/s12874-017-0442-1.

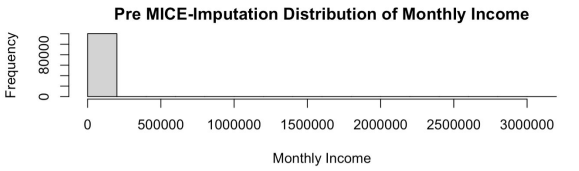
the same procedure for the observations with a missing number of dependents. While this process alleviates the issue of missing data in our testing data set, it does not fully capitalize on the possible information present in the other predictor variables which could more accurately predict the missing value. In addition, it might be applicable under MCAR scenarios but not so for MAR or MNAR data. Additionally, Azur[3] notes that single imputation methods, including median imputation, does not account for the uncertainty in the imputations. Rather, once the median imputation is done, we treat the imputed values as if they are known in the analysis; this is problematic. We motivate a regression based multiple imputation technique in an attempt to use all of the information present in our imputation while accounting for imputation uncertainty.

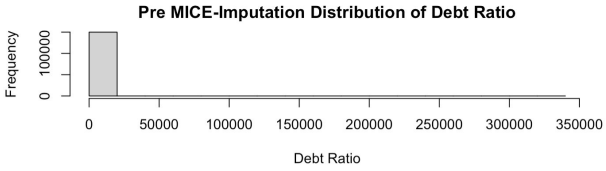
Multivariate imputation by chained equations (MICE) is a method by which we create multiple imputed complete datasets, rather than a single complete data set as in median imputation. The MICE algorithm begins by setting place holders for each missing value using a single imputation approach, such as median imputation. Then, MICE considers one variable. It sets the previously imputed placeholder back to missing for that chosen variable. Now, we have a dataset in which there is only *one* variable with missing values. Next, MICE regresses the chosen variable against the other predictor variables. The missing values of the chosen variable are replaced with the estimates from the regression. The entire process iterates over each variable which initially had missing values. After we regress each variable with missing data on the other predictors, we have a single complete data set generated from this cycle. The cycle is then repeated for a specified number of rounds, each time producing an updated imputed dataset. The final cycle generates the most updated imputed dataset which we will use in our analysis.

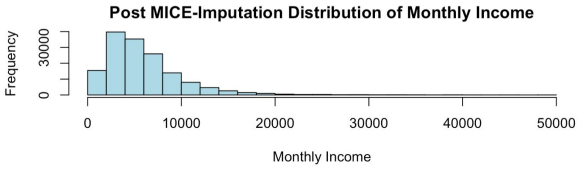
MICE allows us to extract all of the information present in the dataset through the iterative and regression-based process. The training dataset lends itself nicely to MICE implementation as there are most definitely strong indicators of one's income including one's age, number of credit lines, and many other predictors. Our working MAR assumption further confirms that multiple imputation is a logical choice for this data.[4] It would be naive to perform median imputation under the belief that one predictor is independent of another. R has a fantastic package, 'mice', which makes MICE implementation easy. The only interesting choice is the chosen number of cycles. The first time we ran MICE, we naively chose to set the number of cycles to 300. Since there were 5 variables with missing data, this resulted in 1500 iterations; this took over 6 hours to run. After making some adjustments to the cutoff values for variables with impractical entries, we reran MICE with 5 cycles; this took about 1.5 hours. While MICE is effective, it is computationally intensive and small changes in the original dataset results in hours of

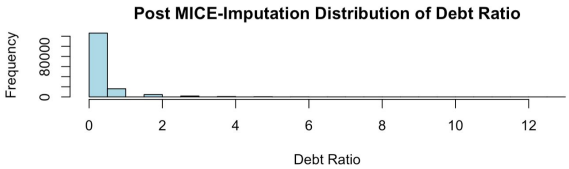
[3] Azur, Melissa J et al. "*Multiple imputation by chained equations: what is it and how does it work?*." International journal of methods in psychiatric research vol. 20,1 (2011): 40-9.
[4] Grigorios Papageorgiou et al., "Statistical Primer: How to Deal with Missing Data in Scientific Research?," *Interactive CardioVascular and Thoracic Surgery* 27, no. 2 (August 1, 2018): 153–58, https://doi.org/10.1093/icvts/ivy102.

imputation. After imputation, we note two significant improvements. First, there are no missing values in the dataset which will allow us to successfully implement a learner. Second, the variables which previously contained entirely impractical values are now distributed as one would expect. Figures showing the improved distributions are included below.









For both monthly income and debt ratio, the result of the imputation is significant. The maximum value for monthly income dropped by a factor of 60 and a factor of ~29000 for debt ratio. The difference between mean and median values for monthly income dropped from 352.6 to .2466, serving as strong evidence of the removal of extreme values.

(b) *Balancing the dataset*

The next issue to address is the imbalanced design of the dataset. Within the training dataset, approximately 93.3% of people did not default and 6.7% did default. Machine learning algorithms tend to exhibit lower accuracy on imbalance datasets for some of the following reasons. First, typical machine learning algorithms try to minimize overall error, of which the defaulting class contributes very little to, and thus the algorithm defaults to predicting the majority class. This phenomenon creates a highly biased model towards the majority class. Additionally, such algorithms assume errors obtained from misclassification have the same cost regardless of the class. In reality, the cost of losing interest payments by not issuing a loan and losing the principal loan amount due to default is drastically different. Because of this, it is of utmost importance to remedy the bias resulting from imbalanced datasets through different sampling techniques.

First, we will consider random oversampling of the minority class. In essence, we replicate observations from the minority class, in this case defaulters, until we reach balanced classes. The main advantage of oversampling the minority class is that there is no information loss; this contrasts undersampling in which we do not use all of the majority class observations in our training dataset.

However, training a model with an oversampled balanced data set is prone to overfitting as it adds identical observations to the dataset.

Next, we will consider an alternative form of oversampling to achieve balanced datasets, namely Synthetic Minority Oversampling Technique (SMOTE). The SMOTE algorithm generates new minority class data which looks similar to the existing minority class data in the feature space. To generate the data, SMOTE uses both bootstrapping and k-nearest neighbors. More precisely, consider an existing minority observation in the feature space and find its nearest neighbor. Take the difference between the observation and its nearest neighbor. Then, multiply the resulting difference by a random number between 0 to 1, and add the result to the original observation. This produces a new minority observation which lives between the original and its nearest neighbor.

In order to implement both random oversampling and SMOTE, we used the package 'ROSE' within R. We performed both oversampling and SMOTE on the imputed dataset and will contrast the results in the next section.

(c) *Classification Trees*

We first consider a tree-based model to predict whether an individual will default. Tree-based models rely upon recursive partitioning, a method by which the model iteratively breaks the dataset up into subgroups based on an optimized variable split point. At each root node of the tree, the model chooses the variable to split on based on an optimization criterion,. In this case, the tree split in such a way that minimizes Gini Impurity, the probability of misclassification. The classification tree continues splitting indefinitely until it reaches a stopping criterion, usually the number of observations in the daughter leaf.

Classification trees have various advantages which motivate their use in our first model. First, decision trees are intuitively accessible to most audiences. In deciding whether to use the model for default detection, a model accessible to non-mathematicians will likely yield higher use. Second, decision trees are a non-parametric learner and thus can easily detect meaningful interactions amongst predictor variables. In contrast, parametric models such as logistic regression require the researcher to specify interactions a priori. Aside from the many advantages, a single classification tree is sensitive to changes in the data and thus inhibiting their predictive accuracy; in the subsequent section we discuss boosting as a method to alleviate this issue.

To generate a classification tree model, we used the function 'rpart' within the 'rpart' library in R. Additionally, to create visualizations of the trees, we used the package 'rpart.plot'. First, we build a classification tree using the imputed unbalanced dataset. As we see below in figure 3, the model first splits on 'NumberOfTimes90DaysLate'. This initial split is in agreement with the importance plot as

'NumberOfTimes90DaysLate' is deemed the most signal bearing variable. Interestingly, neither debt ratio nor monthly income had significant impacts on classification. This is slightly reassuring as we were unsure whether the missingness of monthly income was correlated with debt ratio; even if the two variables were correlated, their seeming unimportance will hopefully not bias the results.

The area under the ROC is a typical performance measurement in classification problems. The ROC is plotted by comparing the true positive rate (TPR) against the false positive rate (FPR). The TRP is defined as (True positive)/(True positive + False Negative). Contrastingly, the FPR is defined as (False Positive)/(True negative + False Positive). Intuitively, a model with strong predictive capability will be able to correctly distinguish the classes and thus yields an AUC closer to 1. A model which yields an AUC of 0.5 implies that the predictive capabilities of the model are no better than chance alone. For the classification tree of the unbalanced but imputed data, the AUC was approximately 0.786; this was obtained by running the models on a subset of data which was not used for training and comparing the predictions with the truth. While the model has predictive capabilities better than chance, there is a significant number of misclassified observations.

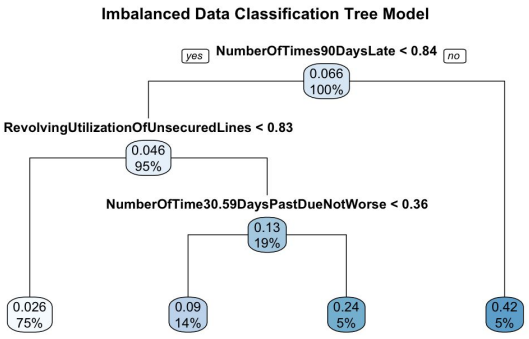


Figure 3: Classification tree generated from unbalanced imputed data

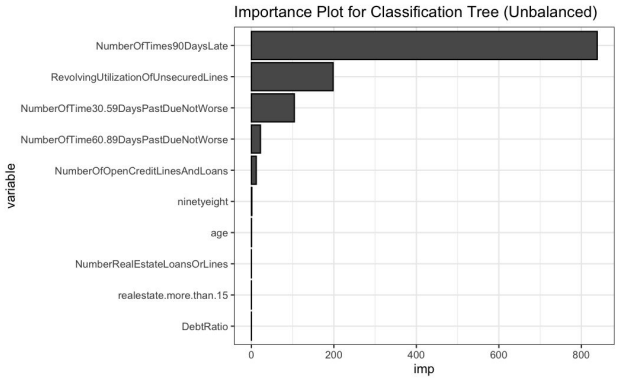
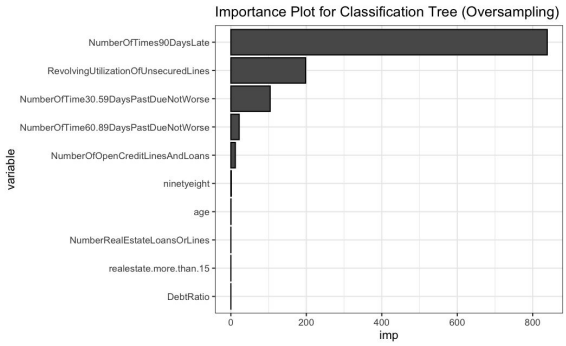
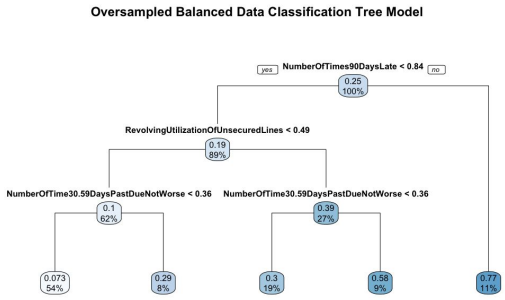


Figure 4: Importance plot for unbalanced tree

We will now consider a classification tree constructed using a balanced dataset. To construct the balanced dataset, we used random oversampling of the minority class. After balancing the dataset, we again fit a single classification tree to the data. Similar to the model with the unbalanced dataset, the number of times an individual is 90 days late constitutes the initial tree split and is the most important variable. The oversampled balanced data tree resulted in an AUC of 0.819; this is a fairly significant improvement. In general, changing the oversampling method did not yield significantly different results. However, using SMOTE to balance the data changed the model is one interesting way. The SMOTE balanced data classification tree splits on the indicator variable 'ninetyeight' which denotes whether the observation had the unlikely value of 98 for the number of times 30-59 days late on payment. While the value 98 is likely a default value if the individual did not give the information, our model suggests that

there are significant differences between the individuals who possibly withheld information and others. Aside from this difference, the oversampling method had minimal effect on the tree construction.





While a single classification tree can yield a fairly high AUC on unseen testing data, we posit that it may be overfitting the training data and thus limiting the testing data results. We now turn our focus to a method which limits overfitting, namely boosting.

(d) Boosted Classification Trees

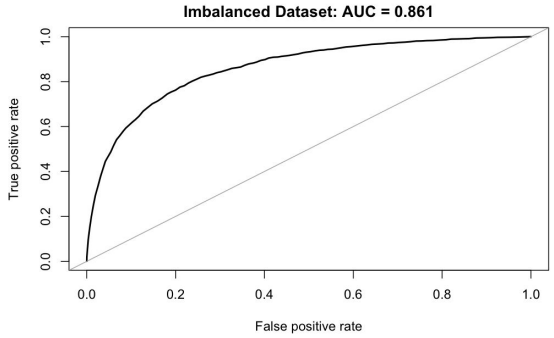
Boosting is a machine learning technique which consists of a combination of many weak learners in an effort to create a strong learner. In this case, our weak learner will be a single classification tree.. The algorithm begins by building a single tree equally weighing the importance of each observation. The algorithm then sequentially builds more classification trees focused on predicting the observations which were previously misclassified by assigning those observations a weight greater than 1. Once the decrease in error rate plateaus, the algorithm produces a final model by adding all of the decision trees together.

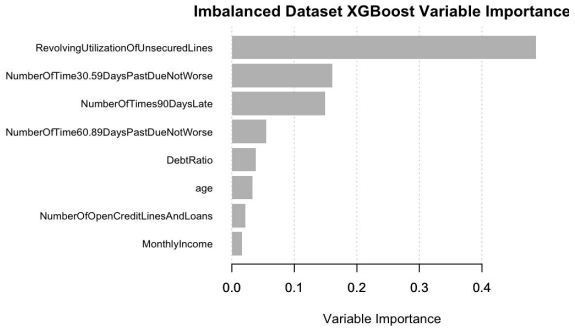
In our attempt to model whether a person will default on a loan or not, we use an algorithm called Extreme Gradient Boosting (XGBoost). XGBoost uses the gradient descent algorithm to minimize error when adding additional trees to the model. The gradient descent algorithm is used to find a local minimum of the loss function and thus improve the overall error rate. Intuitively, we can imagine choosing a point on a traditional cereal bowl and calculate the derivative of our error function, the so-called gradient, evaluated at our chosen point. The gradient is then used to find the direction in which to tune the parameter(s) to maximally reduce the boosted model error. XGBoost employs various hyperparameters to increase the computational speed and accuracy of the general gradient boosting algorithm. Unlike CART, XGBoost employs many behind-the-scenes optimization processes and is difficult to conceptualize. However, XGBoost generally has outstanding predictive accuracy and thus is a good choice for a Kaggle competition.

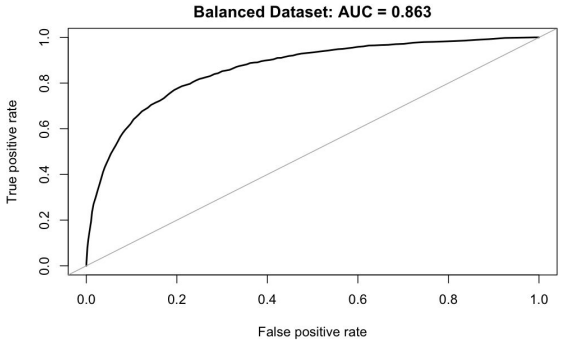
We built an XGBoost model for both the unbalanced and oversampled balanced datasets. Ultimately, the XGBoost model constructed with the oversampled balanced dataset yielded the most optimal results. As we in the figures below, the XGBoost model for the imbalanced dataset extracts the

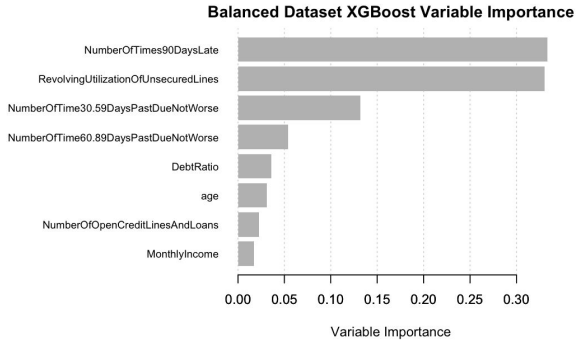
most information from the revolving utilization of unsecured lines. Again, we see monthly income and debt ratio have minimal impact on the model, which was surprising as both variables seem indicative of default status. The XGBoost model for the imbalanced dataset ultimately yielded an AUC of 0.861; this is a significant improvement from the single classification tree model.

Finally, we ran XGBoost on the oversampled balanced dataset. In both XGBoost models, we cross validated to find the optimal number of trees to sequentially add to the model. Unlike the first model where cross-validation deemed 7 trees optimal, the second XGBoost model did not converge on an optimal number of trees to include. We ended up including 400 individual trees within the boosted model for the balanced dataset. Also of interest is the order of variable importance; the number of times 90 days past due seems to carry the most signal in this model. Ultimately the XGBoost model increased the AUC by 0.2%. The AUC of the XGBoost model with the balanced dataset was 0.863. Due to the highest predictive accuracy on the test dataset, we present the XGBoost balanced dataset model as our final model.









Conclusion

To summarize, we first checked for patterns in the missingness of data to determine our imputation methods. Our findings indicate that the data was MAR, which allowed us to add binary indicator variables to account for imputation and to then use MICE to impute missing values and outliers

in the training data set. We then compensated for the imbalanced data set by testing our learners on three data sets: an original data set, a synthetically generated data set, and a data set created by random oversampling. For our learners, we tested multiple standard classification trees and boosted classification trees via extreme gradient boosting. Our highest AUC was achieved when using extreme gradient boosting on the oversampled balanced dataset—this AUC was 0.863. After applying this to the test data set, we received a Kaggle score of AUC = 0.85397. As an ensemble learner, the high predictive accuracy of extreme gradient boosting can be attributed to the boosting algorithm whereby the final model includes many intermediate models which were created to predict the errors of prior models. Despite the high AUC, there are several factors which could be corrected to improve the model. First, the thresholds that constituted outliers for our explanatory variables were created due to contextual knowledge of variables and with help of the 1.5 IQR rule. A more rigorous system for constituting outliers would likely improve results. Second, the ROSE algorithm generates synthetic data using a smoothed-bootstrap approach. Since we have ten explanatory variables, this smoothing will take place in 10-space, which will produce inaccurate results due to the curse of dimensionality. An alternative non-parametric method for data generation might be more appropriate for this problem. Finally, multiple imputation by chained equations relies on multivariate linear regression methods to impute data. Although the MICE algorithm iterates through the data and runs multiple cycles, it rests on fundamental assumptions of linearity. Alternative imputations methods, such as random forest imputation, could better fit our data set. However, despite these limitations, our AUC of 0.863 on the training data is high, reflecting the strong predictive power of balanced extreme gradient boosting.