Data Science

Fall 2022

Assignment 1

Group 07

Dibya Kumari

Marco De Luca

Andreas Heule

James McGhee

Contents

[1. Raw Data 3](#_Toc123106951)

[a. Parameters Not Relevant to Analysis 3](#_Toc123106952)

[b. Parameters Requiring Processing to Use 4](#_Toc123106953)

[c. Parameters Currently Usable and Relevant 6](#_Toc123106954)

[2. Apply Pre-Processing to Data 7](#_Toc123106955)

[a. Pre-Processing Categorical Variables 7](#_Toc123106956)

[b. Dealing with NA Values 7](#_Toc123106957)

[i. Dropping Parameters with Mostly NA 7](#_Toc123106958)

[ii. Building Models with Observations Containing Complete Data 7](#_Toc123106959)

[iii. Imputing Missing Data for NA 8](#_Toc123106960)

[c. Handling Joint Applications 8](#_Toc123106961)

[3. Prepared Data 9](#_Toc123106962)

[a. No Adjustment to Data 9](#_Toc123106963)

[b. Transforming Parameters 9](#_Toc123106964)

[4. Applying Learning Algorithm to Data 9](#_Toc123106965)

[a. Linear Models 9](#_Toc123106966)

[i. Feature Selection 9](#_Toc123106967)

[ii. Increasing Flexibility 10](#_Toc123106968)

[iii. Ridge Regression and LASSO 10](#_Toc123106969)

[b. Multivariate Adaptive Regression Spline (MARS) 10](#_Toc123106970)

[c. Gradient-Boosted Trees 10](#_Toc123106971)

[d. Extreme Gradient Boosting 10](#_Toc123106972)

[5. Candidate Models 11](#_Toc123106973)

[6. Deploy Chosen Model 11](#_Toc123106974)

# Raw Data

The first challenge was loading the data into R. We had to deal with the header and the separator being a semicolon instead of a comma.

## Parameters Not Relevant to Analysis

We determined the following variables would not be relevant to analysis and should be dropped:

1. Id: This is to identify the loan. It is not relevant for decision-making.
2. member\_id: This is probably not relevant, but it could be the case that a member has obtained prior loans, and then that could be very relevant for the interest rate on this loan. After checking the data, this never actually happened, so we dropped the column.
3. emp\_title: This is a string entered by the loan applicant. While perhaps some natural language analysis could yield interesting results, we determined that there are already so many parameters to consider in this dataset that work to analyze it would not be a productive use of time.
4. loan\_status: This is for after the loan is issued, so we did not believe it will be useful in predicting interest rate and dropped it. After this analysis, Gwen informed the class that this parameter would not appear in the secret data.
5. pymnt\_plan: Indicates if a payment plan has been put in place for the loan. This applies only to current, defaulted loans and would not be relevant for new loans. It should be deleted. After this analysis, Gwen informed the class that this parameter would not be a part of the secret data.
6. url: This is the URL of the loan application. It is not relevant to decision-making, and we dropped this parameter.
7. desc: This is a string entered by the loan applicant. While perhaps some natural language analysis could yield interesting results, we determined that there are already so many parameters to consider in this dataset that work to analyze it would not be a productive use of time.
8. title: This is a string entered by the loan applicant to title the application. While perhaps some natural language analysis could yield interesting results, we determined that there are already so many parameters to consider in this dataset that work to analyze it would not be a productive use of time.
9. zip\_code: This information would indicate where in the country the borrower is located. It might be a relevant variable, but it would at best be an approximation of a borrower's income. Since the first three (of five) digits in a zip code would only indicate in which city (for a 500,000+ inhabitant city) or in what part of the state (for rural areas) the borrower lives, we considered that this would not provide useful information that could not be obtained directly from the income information and dropped this parameter.
10. addr\_state: This information would indicate where in the country the borrower is located. It might be a relevant variable, but it would at best be an approximation of a borrower's income. We did not believe it provided useful information and dropped the parameter.
11. earliest\_cr\_line: This is the earliest reported date on the borrower’s credit report. It is likely correlated strongly to a borrower’s age and could be a good placeholder for this data. We speculated that if this date is old, the person is old, so has a higher interest rate, and if young, low income, so higher interest rate. But this would simply be because of our expectations of income by age, and this data does not directly tell us anything not already available from income information.
12. total\_acc: This parameter tracks all accounts an applicant has, but it does not distinguish between open and closed accounts, nor does it include any payment history. In general, we would not expect it to be a strong predictor.
13. out\_prncp: This value is only relevant after a loan has been issued and should be dropped.
14. out\_prncp\_inv: This value is only relevant after a loan has been issued and should be dropped.
15. total\_pymnt: This value is only relevant after a loan has been issued and should be dropped.
16. total\_pymnt\_inv: This value is only relevant after a loan has been issued and should be dropped.
17. collection\_recovery\_fee: This value is only relevant after a loan has been issued and should be dropped.
18. last\_pymnt\_amnt: This value is only relevant after a loan has been issued and should be dropped.
19. last\_pymnt\_d: This value is only relevant after a loan has been issued and should be dropped.
20. loan\_status: This value is only relevant after a loan has been issued and should be dropped.
21. next\_pymnt\_d: This value is only relevant after a loan has been issued and should be dropped.
22. recoveries: This value is only relevant after a loan has been issued and should be dropped.
23. total\_rec\_int: This value is only relevant after a loan has been issued and should be dropped.
24. total\_rec\_late\_fee: This value is only relevant after a loan has been issued and should be dropped.
25. total\_rec\_prncp: This value is only relevant after a loan has been issued and should be dropped.
26. last\_credit\_pull\_d: This value represents the last time the credit report was checked by Lending Club. Since we are not using data parameters, it is not relevant for analysis.
27. policy\_code: Every value in this column is equal to 1. Therefore, it is not helpful to analysis.

## Parameters Requiring Processing to Use

The following variables were not in the correct form to be processed:

1. funded\_amnt\_inv: This is a string but should be an integer. After conversion, we compared it to the loan\_amt, and saw two distinctly different curves.
2. term: This consists of two values, either “ 36 months” or “ 60 months.” We removed the leading empty space. We determined it should remain as a categorical variable instead of converting it to a number, since the size of the impact of the term on the int\_rate was not immediately obvious, other than 60-month loans would probably have a higher interest rate. We visualized the data and found that it created two distinct curves when mapped against loan amount. We determined that 60-month loans are for more money than 36-month loans. Very few 60-month loans are for under $10,000. We transformed the parameter into dummy columns. After this review, Gwen informed the class that this parameter would not appear in the secret data due to her understanding of the Lending Club loan process. We dropped this parameter.
3. int\_rate: This is a string but should be a float. We visualized the interest rate in various ways, including histograms and by plotting it against other variables.
4. installment: This is a string but should be an float. This should be directly correlated with the loan amount and interest rate and determined by term. It may be useful to use these values instead of those when say, comparing to income, in order to avoid calculation. After this review, Gwen informed the class that this parameter would not be a part of the secret data, and we dropped it.
5. emp\_length: This is a string. This variable represents the length of the employment for the loan applicant. While it has an impact on the interest rate, we were unsure whether it would be best to retain this parameter as a categorical variable, since there is not a linear relationship between the variable and the interest rate. We converted it to integers, with 0 being less than one year, and 10 being 10+ years employment. This decision left us with approximately 40,000 NAs, which seemed like a significant portion of our observations. After review, we assumed that NA would represent an unemployed loan applicant, rather than an applicant with no employment information. We then adjusted the data so that 0 represented unemployed, 1 employment less than 1 year, and 11 represents employment for 10+ years.
6. home\_ownership: This is a categorical variable. We considered converting them to ordered integers, but there was no obvious way whether a mortgage would be “better” than renting, or whether the difference between categories would be approximately equal. Since there were only four possible categories, we decided to leave it as a categorical variable and convert it to dummy columns.
7. annual\_inc: It is a string but should be an integer. There were no negative values, but there are two with 0. There was significant income from a joint applicant in these cases, so we left them as-is. We did discover that there are several applicants that reported high income that both seemed unlikely that such a person would be using this platform and also did not appear to be justified by the reported employment titles. We decided that incomes over $1,000,000 was likely a data error where a person had entered, for example, $10,000.00, but due to the way the input was handled, the period indicating cents was not accepted by the system. We divided incomes over $1 million by 100 to adjust them to be within the reasonable range of data.
8. verification\_status: This is a categorical variable. We converted it to an integer with 0 as not verified to show the effect of verification. After visualization, it appears that that it does have a significant effect on a linear model, but boxplots indicated those with a verified income have a higher interest rate than those with unverified income. This was a surprising result, and we decided to keep it as a categorical variable with dummy columns.
9. issue\_d: This is a string. As interest rates change with time, we believed this would be a very useful parameter. To convert it to a date object, we assumed that each issue date was the first day of the month, as only the month and year were provided. After this review, Gwen informed the class that this parameter would not appear in the secret data, and we dropped it.
10. purpose: This is a string selected by the applicant but appears to be from a dropdown list rather than entered freely. We converted it to dummy columns.
11. dti: This is a string but needed conversion to a float. There were 403 with a dti of 0 and 3266 with dti of less than 1. It seems odd that these borrowers would turn to Lending Club for their loans if this were true. However, considering that these values were such a small percentage of total observations and were not impossible, we did not modify them.
12. revol\_util: This is a string and should be a float. This is a ratio showing how maxed out the borrower's credit cards are. We would expect creditworthy borrowers to have a low ratio. This has a fairly normal distribution.
13. initial\_list\_status: Lending Club reserves a few loans for 12 hours and offers them to the institutional and large retail lenders who want to lend the whole amount for a loan. This may be useful because those loans taken by institutional and large lenders may be the most preferred loans by lenders. A boxplot shows that w loans tend to have a lower interest rate. A simple linear model shows an R^2 of only 0.01, but that this is significant.
14. dti\_joint: This is a string but needed conversion to a float. It is only relevant for joint applications.
15. Il\_util: This is a string but needed conversion to a float. It is relevant for the applicant’s other credit history.
16. all\_util: This is a string but needed conversion to a float. It is relevant for the applicant’s other credit history.
17. annual\_inc\_joint: This is a string but needed conversion to an integer.
18. application\_type: This is a categorical variable that required conversion to dummy columns.
19. verification\_status\_joint: This is a categorical variable that required conversion to dummy columns.

## Parameters Currently Usable and Relevant

We reviewed the following variables:

1. loan\_amt: We verified that they were non-null, non-zero, and positive and created visualizations with histograms in various bin sizes.
2. funded\_amt: We verified that they were non-null, non-zero, and positive and also created visualizations to compare to loan\_amt.
3. delinq\_2yrs: This was already an integer.
4. inq\_last\_6mths: This represents how many times a lender has pulled the borrower's credit report. In itself, it does not mean anything, but a higher value is taken as an indication that the borrower is not creditworthy (else, why ask for so much debt?) 0-2 is fine. It has a power law distribution, and a significant effect on the interest rate.
5. inq\_last\_12m: The same issues as inq\_last\_6mths.
6. mths\_since\_last\_delinq: This is the time that has passed since a buyer was delinquent. Approximately half of the entries are NA. However, if the borrower has never been delinquent (that is, always pays on time), it should be NA. NA is a better value than having anything here. But the data shows this is not actually true, and those with NA have a higher interest rate. Possible solution to the NAs: very, very high value to replace NA? But this would not correspond with the data that NA have a higher interest rate. There is a hard cutoff at 84 months in the data because those delinquencies are no longer reported on a credit report (7 years). There are nevertheless very, very few observations beyond that time period, but we are not sure why. They may pre-date a change in the law or loopholes that allow continuing reporting, or voluntary self-reporting to Lending Club.
7. mths\_since\_last\_record: This would be the months since the last public record filing, which means a lawsuit to collect a debt. This is very, very bad, and much better if it is NA (i.e. there are none). They also remain on your credit report for ten years during this time period (this is no longer true, but not relevant). Those with no public records have higher interest rates than those who do, which we did not expect, the same as mths\_since\_last\_delinq.
8. open\_acc: We would expect this to be generally on the high side for users of this platform. A simple linear model shows the interest rate goes down for people with more open accounts and that this is significant, which seems odd.
9. pub\_rec: Histogram shows very, very strongly power law.
10. revol\_balance: This is the revolving debt (credit card, usually) of the borrower. It will be a part of the dti calculation.

# Apply Pre-Processing to Data

## Pre-Processing Categorical Variables

Many of the parameters identified above involved categorical variables. For nearly every categorical variable, we created dummy columns to use them. We removed one dummy column from each parameter to avoid introducing collinearity into the model.

Exceptions to this practice was for emp\_length. Here, we ordered the data with NA representing unemployed being replaced with 0, up to 11 for 10+ years of employment. While the relationship between emp\_length and int\_rate is not linear, increasing the dimensionality of the model would make additional dummy variables less significant in the model. By increasing the flexibility of this parameter, we should be able to deal with nonlinearity issues.

## Dealing with NA Values

At the initial stage, NA values were dropped. In particular, those observations with NA for home\_ownership, annual\_inc, delinq\_2yrs, revol\_bal, revol\_util, and collections\_12\_mths\_ex\_med were dropped. This had the effect of removing observations from the data. However, considering the size of the dataset, we did not consider the number significant.

We also dropped NA values from emp\_length in some initial analysis and model building. This resulted in approximately 40,000 observations being dropped. After review, we determined that the NA values were likely unemployed individuals rather than observations with no data for this parameter.

### Dropping Parameters with Mostly NA

In the initial models, we dropped the parameters "mths\_since\_last\_delinq", "mths\_since\_last\_record", "mths\_since\_last\_major\_derog", "tot\_coll\_amt", "tot\_cur\_bal", "open\_acc\_6m", "open\_il\_6m", "open\_il\_12m", "open\_il\_24m", "mths\_since\_rcnt\_il", "total\_bal\_il", "il\_util", "open\_rv\_12m", "open\_rv\_24m", "max\_bal\_bc", "all\_util", "total\_rev\_hi\_lim", "inq\_fi", "total\_cu\_tl", and inq\_last\_12m" due to the high instances of NA values that did not appear to be justified by the category. It was much more likely, for example, that Lending Club had not collected this data for these applicants than for example, mths\_since\_last\_record showing that most applicants had no records. Since the number of NAs in many of these columns was approximately equivalent and such a large proportion of the data, we felt justified in dropping the columns since there was no obvious way to fix the data error.

### Building Models with Observations Containing Complete Data

A possible solution would be to build two models for the dataset, one consisting of the observations without data for these parameters and another for the portion of the dataset with complete data. We created a separate dataset where credit report information was available. We started with whether inq\_last\_12m was not NA, as this seemed to capture all of the observations with complete data.

From this point, we cleaned the data for these parameters. First, if total\_bal\_il was 0, that is, no balance, we made il\_util 0 as well. For any other il\_util NA values, we set them to 0, since we understood this to mean that there were no installment loans active.

Parameters such as mths\_since\_last\_delinq posed a challenge since stronger applicants would in theory have an NA value. When examining the entire dataset, NA values showed a higher interest rate than those with values in this category, which we found strange. When limiting the dataset to only the applicants with credit report information on file, applicants with NA in this column, that is, no delinquencies, have a lower interest rate, which is in line with what we expected. From the data, it was not apparent how to replace the NA values so they could be used. We modified this parameter to a categorical one with 0 for borrowers who had not previously defaulted (NA value) and 1 for borrowers who had previously defaulted (non-NA value).

We did the same processing for other data. For mths\_since\_last\_record, this is similar conceptually to mths\_since\_last\_delinq. In this case, however, it made more sense to set NA applicants to a high number, since the biggest effect is seen 1/3 of the way through the graph when plotting mths\_since\_last\_record against int\_rate. We set NA values equal to 121 (the 10-year cutoff plus one) and kept other values in place.

mths\_since\_last\_major\_derog we made categorical on the same pattern as mths\_since\_last\_delinq since there did not appear to be a clear trend plotting this against int\_rate.

mths\_since\_rcnt\_il showed a good correlation between the parameter and int\_rate, but not much correlation between na and not na. There were too many observations to drop without having a significant impact on the remaining number of observations in this limited section of the dataset. We considered filling NA values with the mean, but since there is a correlation between the value and int\_rate, this may not have been appropriate. We imputed the data using a bagged trees method from the caret library, as described below.

### Imputing Missing Data for NA

The success of imputing mths\_since\_rcnt\_il for the NA values lead us to impute the NA values for all the data in the complete dataset. This was done using the bagged trees method as an alternative to the process described in 2.b.ii., above. The bagged tree method was preferable because it could accept predictors that have missing values themselves. Since many of the parameters had missing values for most of the observations. This method can be used when the parameter to be imputed is categorical or numeric, and our data had both.

In the bagged tree method, the algorithm constructs regression trees and averages them. Each decision tree would have a low bias but a high variance. Through bagging, the variance is reduced. The downside of this method is that it eliminates the explainability of decision trees, but this drawback was not relevant for our use case.

We used the caret library’s preProcess() function for this imputation. This data was then checked with the Ridge Regression, LASSO, MARS and GBM models.

## Handling Joint Applications

Since few applications were joint applications, how to handle this column of mostly NA values represented a challenge. We created a calculated attribute to represent the sum of the applicant and joint applicant’s income, and then substituted that value in place of the annual\_inc value. This ensured that all applications had the full income considered as one parameter. We performed the same analysis on dti and dti\_joint. Since dti is a ratio of income to debt payments, and debt payments are not in the dataset, we had to perform some algebra to calculate the debt payments for the applicant and joint applicant, sum them, and then calculate the overall dti based on the overall annual income. We then substituted this value for dti. We dropped annual\_inc\_joint and dti\_joint from the dataset, since their information had been incorporated into the other parameters.

# Prepared Data

Our models used the data from the Pre-Processing and made final adjustments to it for use in our models in the following ways.

## No Adjustment to Data

Our first models made no additional adjustments to the data other than those described in 2.a.ii. We used this data to build the first, simple linear models.

## Transforming Parameters

Following the first attempts at a linear model, we attempted to standardize the data. We found that this did not make a significant improvement to the model.

For the next attempts to build a model using the caret library, we scaled and centered the data using the preProcess() function and removed parameters with a near-zero variance.

We also manually kept the int\_rate the same despite the transformation. While scaling and centering the int\_rate would minimize our MSE, this is because the int\_rate would be changed from floats generally between 10 and 20 to floats generally between 1 and -1. This is not predictive, and were this to ever count for reducing MSE, we could reduce it further by further restricting the standard deviation of the transformed data.

# Applying Learning Algorithm to Data

## Linear Models

### Feature Selection

We performed a stepwise forward search to identify the parameters to be included in the model. After dropping the parameters made up of NA or not considered relevant, we had 39 possible predictors. We compared to a backwards search but found the outcome was similar and did not investigate backwards selection further.

From the 25 most simple linear models identified by the search, we built a linear model for each one. We performed a 10-fold cross-validation on each model to compare the CV error as a function of the number of parameters. Continuing to add parameters reduced the CV error. However, this effect was minimal with 25 parameters, showing a 0.01 reduction in CV error in comparison with the 24-parameter model.

This feature selection identified that some parameters were highly correlated. The very highly correlated ones were:

purpose\_credit\_card <-> purpose\_debt\_consolidation

purpose\_credit\_card <-> purpose\_home\_improvement

purpose\_credit\_card <-> purpose\_other

purpose\_debt\_consolidation <-> purpose\_home\_improvement

purpose\_debt\_consolidation <-> purpose\_other

Since these were categorical variables in dummy columns, we determined that these parameters would never both be “1” at the same time and decided to disregard the correlation.

### Increasing Flexibility

We used the 25-parameter model for further tuning as it provided the best CV error and appeared from the trend that additional parameters would provide little explanatory value. To give our model more flexibility to adapt to the data, we ran 10-fold or 5-fold cross-validation tests on the eight top parameters in our model. We allowed the first parameter to adjust as a polynomial between 1 and 10, found the best CV error, and updated the model. If the values produced were close, we preferred the lower order polynomial to minimize the chance of overfitting. After performing this on the first parameter, we conducted the same process for the other parameters in the top 8. At this point, we had reduced the CV error from approximately 13.14 to 12.51. While this was an improvement, it appeared that this process would be labor intensive and not yield substantial gains, since it was unlikely that the parameters of lesser importance would cause a large CV error change when their flexibility increased.

### Ridge Regression and LASSO

We created models using ridge regression and LASSO. They did not provide significantly different MSE from the best linear model we had created. This was the case whether or not we used the preProcess() function of the caret model.

## Multivariate Adaptive Regression Spline (MARS)

We attempted to use a MARS model to improve the performance. This model automatically models nonlinearities and interaction among parameters. The interaction effects we had not modeled well previously, in part because we had so many parameters and exponents for which to control. This was the first real breakthrough, where our MSE reduced from approximately 12 to approximately 8.3.

## Gradient-Boosted Trees

We built a model from gradient-boosted trees, which is a type of decision tree model. Our model we tuned on the parameters interaction.depth of 1 to 3 and trees of 50, 100, and 150, using a 5-fold cross-validation, repeated three times. Shrinkage was held constant at 0.1 and the minimum number in a node constant at 10. Our model revealed the strongest predictive power of any attempted, with a cross-validation MSE of 2.48, but a test MSE of 6.16. The change from cv-MSE versus test MSE was surprising.

We observed that cv-MSE went down as interaction depth increased and the number of trees increased. In order to reduce the test MSE, we reran the model with an interaction depth of 9 and 12 and 1500 trees, keeping other parameters the same. This model took approximately 12 hours to complete but provided a test MSE of 2.31 for the model with interaction depth 12.

## Extreme Gradient Boosting

Having success with gradient-boosted trees, we attempted extreme gradient boosted trees using the xgboost library. Our model was tuned on the number of rounds of boosting, from 50 to 5000, and a maximum depth from 6 to 21. We kept other values constant with an eta (shrinkage) of 0.3, gamma (minimum loss reduction) of 0, colsample\_bytree (subsample ratio of columns) at 1, min\_child\_weight (minimum sum of instance weight) of 5, and subsample percentage of 0.6. We used a 5-fold cross validation to determine the best hyperparameters. We expected increased interaction depth and increased rounds to improve the model as with gradient-boosted trees, but this model began to overfit much sooner. A max depth of 6 at 2000 rounds provided the best model, with a cross-validation MSE of 1.40 and test MSE of 1.89.

# Candidate Models

The Gradient-Boosted Trees became our first candidate model. We attempted to further refine the data by factorizing the binary parameters so that they were not affected by the preProcess() function. This attempt was a complete failure and ruined MSE such that it went back to approximately 8. Because more trees and more interaction depth had resulted in a significantly improved model, we also attempted to tune different tree counts and interaction depths beyond 12 and with more trees. This was unable to complete after four days of model building, and we decided to abandon more complex models.

Our second candidate model was Extreme Gradient Boosting. We attempted to model more hyperparameters by lowering the depth and searching for boosting rounds around 2000 but were unable to devote the computing time necessary to the project and ran into R crashing multiple times after working for several hours.

# Deploy Chosen Model

We kept the Extreme Gradient Boosting model as our final model and deployed it for use. To do this, we exported the model as “xgbModel.rds,” which is submitted with this report. We also attempted to export our preprocess model, which could then be applied on the secret test data. This file did not export successfully, as its size continued to grow far beyond the size recognized in RStudio. For this reason, we are not submitting the preprocessing model.

We are submitting with this report, in addition to the model, the model data before it is split into training and test sets or used in the preProcess() function but after manual preprocessing. This represents those actions described in Sections 1, 2.a., 2.b.i., 2.b.ii., and 2.c. of this report. The final preprocessing to scale, center, impute, and remove near-zero variance parameters will occur on the reviewer’s computer. This will ensure that the same model is used for these transformations of the data on the test set as was used on the training set.

We discovered a strange quirk of our chosen model in the deployment phase. When we used previously split test and training data, our model performed significantly worse and achieved a test MSE of approximately 16. This appeared to be an issue with not having reset the seed prior to calling preProcess() in our original model building process, likely due to the bagged imputation this function performs. We rebuilt the deployed model setting the seed prior to calling preProcess(), but this resulted in a test MSE of approximately 2.3, worse than expected. We decided to keep the splitting into training and test sets as part of the deployment rather than providing a separate training and test set in order to ensure the lower MSE. We also struggled to export and import to and from .csv and achieve the same results. To solve this issue, there are two solutions. First, the data may be imported in chunk 1 as an .rds file, which is provided, split into training and test sets, and preprocessed. Alternatively, you may import the preProcess.rds file, which we will attempt to provide with this report, as shown in chunk 2. This file is 2.4 GB large uncompressed on disk, but 31 MB in RAM after loaded in R.

To use the model, please see model\_testing.Rmd.