STA 235H - Prediction Project

December 12th, 2021

Nidhish Nerur

Tanay Sethia

Pavan Agrawal

Sanath Govindarajan

I. Data Preparation

Data preparation is arguably the most important part of this project, seeing that there are many covariates that could potentially give us a very high accuracy model, but are reliant on information that is not technically available before a mortgage application, as well as covariates that are irrelevant or have no predictive power and therefore could cause the overall model to underperform (such as various identifiers).

We excluded identification variables and other variables that are not predictors, such as the financial institution's legal entity identifier, the year the data submission covers, the 5-digit MSA/MD code, the state and county code, and the census tract number. Additionally, we excluded loan denial reasons because we cannot use those to predict whether or not a loan will be denied. Finally, we removed *hoepa_status*, which shows whether the covered loan is a high-cost mortgage, because it is only determined after someone has been given the decision that they are going to get an approved loan. For instance, the 1 and 2 categories of *hoepa_status* are for accepted applicants, and 3 represents N/A which is only given to applicants that may be denied. With a balance table[1], we can clearly see that *hoepa_status* is perfectly collinear with the action of the mortgage status, since loans are only given hoepa consideration after the decision has been made to accept them, and it is unlikely that this covariate will actually be useful in a model whose purpose is prediction..

We converted all numerically-coded categorical variables into factors. Some of these variables include *purchaser_type*, which is the type of entity purchasing a covered loan from the institution, *applicant_race*, which is the applicant's race, and *applicant_credit_score_type*, which is the credit scoring model to generate credit scores.

We created a function *filter_mostly_na_cols* that removes all columns with a total count of N/A values over a certain threshold. After some experimentation, we decided to set the threshold at ≥ 80 N/A values because it significantly lowered RMSE in the regression task. This also brought in important predictors such as debt_to_income_ratio into the clean dataset used to predict whether or not someone was denied a loan.

[1] See Appendix - Figure 1

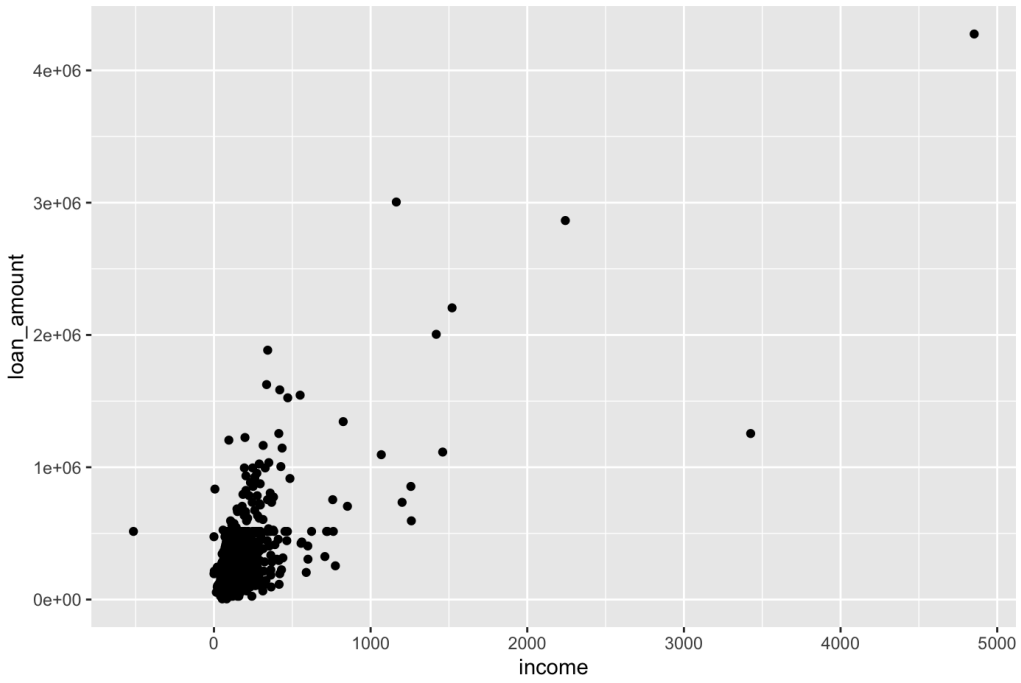
In some columns, there were data points that were interpreted as characters rather than numbers because of the presence of "exempt" values in certain data points. In order to deal with this, in our preprocessing, we chose to instead replace "exempt" with NAs, which were subsequently dropped in future preprocessing steps.

Further, in some specific columns, we noticed an opportunity to make our model even more robust. For example, the debt to income ratio variable is a purely character column that is a mix of exempt, character ranges, and specific percentage number values. In our current model, we were forced to take a factor of this column, seeing as when our model is evaluated against the unseen dataset, we will not be able to transform it into a uniform format. We postulate that it would be much more accurate to not treat this as a factor, but rather average the ranges to create specific numerical values, and utilize that as a continuous numerical covariate to make predictions, seeing as that it is a *critical* covariate in mortgage application decisions - but for the time being, a factor was our best approximation to this ideal solution. After cleaning, we were left with ~780 rows for training.

Finally, we used an 80/20 train/test split stratified on *action_taken* for classification and *loan_amount* for regression, with roughly 80% of the clean data used for training and roughly 20% used for testing.

Descriptive Statistics For Relevant Variables

The variable *loan_amount* refers to the amount of the covered loan where the mean loan amount is $340,000. The variable *income* refers to the gross annual income, in thousands of dollars and the individuals had an average income of $178,600. The variable *property_value* represents the value of the property securing the covered loan, with the mean property value being $547,173. *Debt_to_income_ratio* is the ratio, as a percentage, of the applicant's monthly debt to their monthly income, where those with debt-to-income ratios between 20-30% were the group with the most loan originations (197) and those with ratios greater than 60% were the group with the most denials (29).

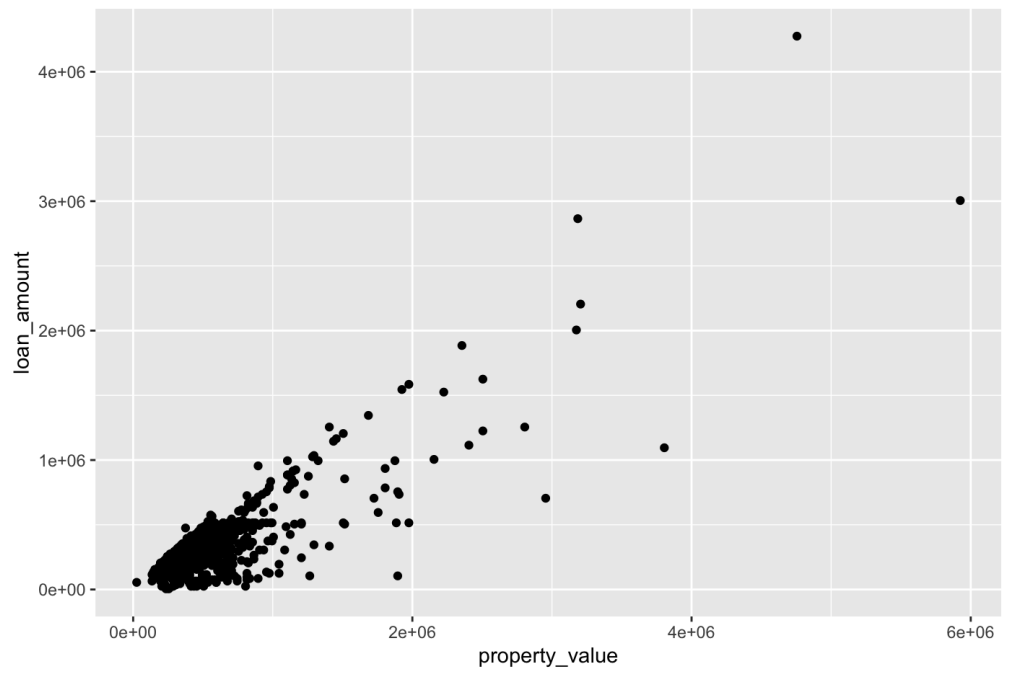


Loan Amount

Min.	5000
1st Qu.	205000
Median	285000
Mean	340688
3rd Qu.	395000
Max.	4275000

Income

Min.	-514.0
1st Qu.	87.0
Median	130.0
Mean	178.6
3rd Qu.	197.0
Max.	4853.0

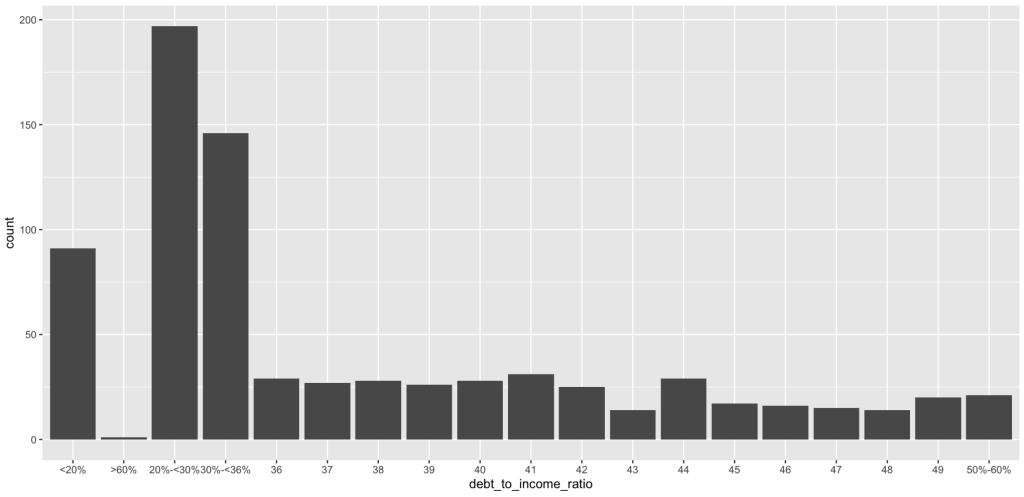


Loan Amount

Min.	5000
1st Qu.	205000
Median	285000
Mean	340688
3rd Qu.	395000
Max.	4275000

Property Value

Min.	25000
1st Qu.	305000
Median	435000
Mean	547173
3rd Qu.	615000
Max.	5925000

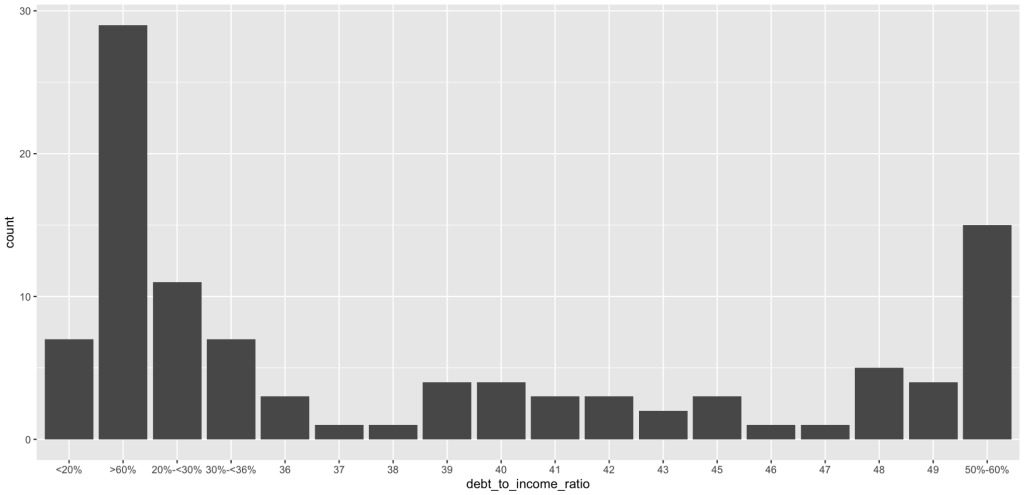


Debt to Income Ratio

20%-<30%	197
30%-<36%	146
<20%	91
41	31
36	29
44	29
(Other)	252

Action Taken

Loan Originated	775

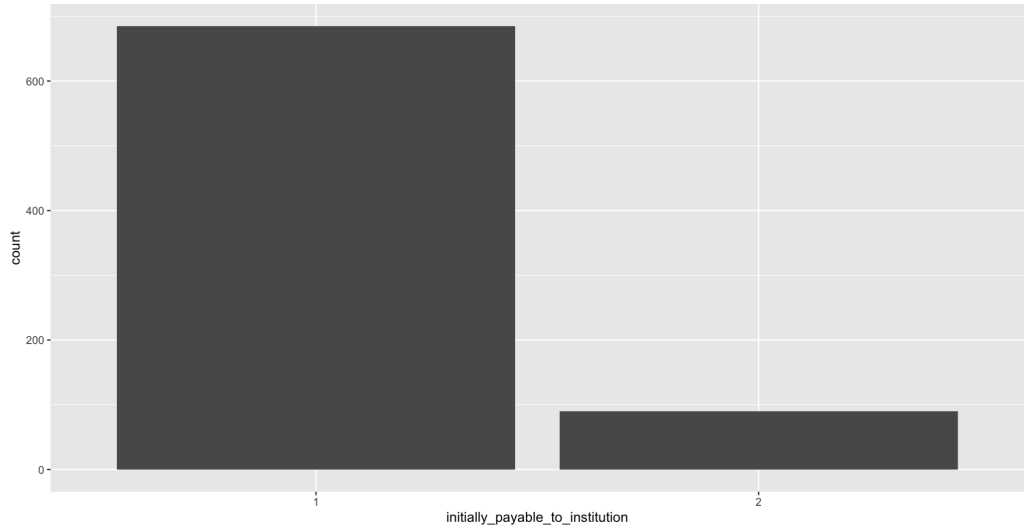


Debt to Income Ratio

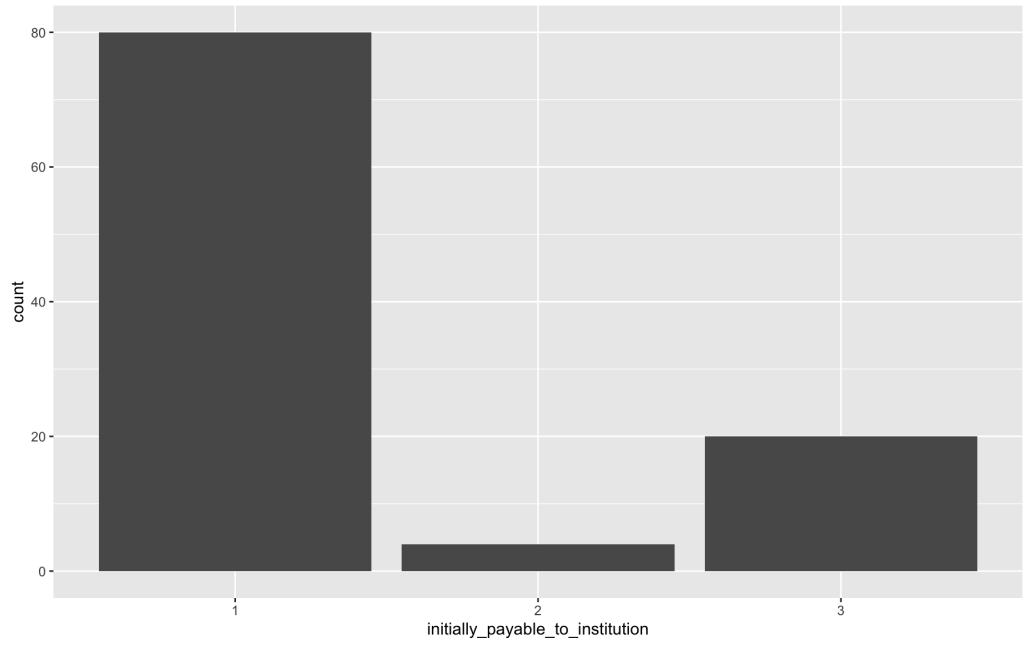
>60%	29
50%-60%	15
20%-<30%	11
<20%	7
30%-<36%	7
48	5
(Other)	30

Action Taken

Application Denied	104



Initially Payable to Institution	
Initially Payable	685
Not Initially Payable	90
Not Applicable	0
Exempt	0
Action Taken	
Loan Originated	775

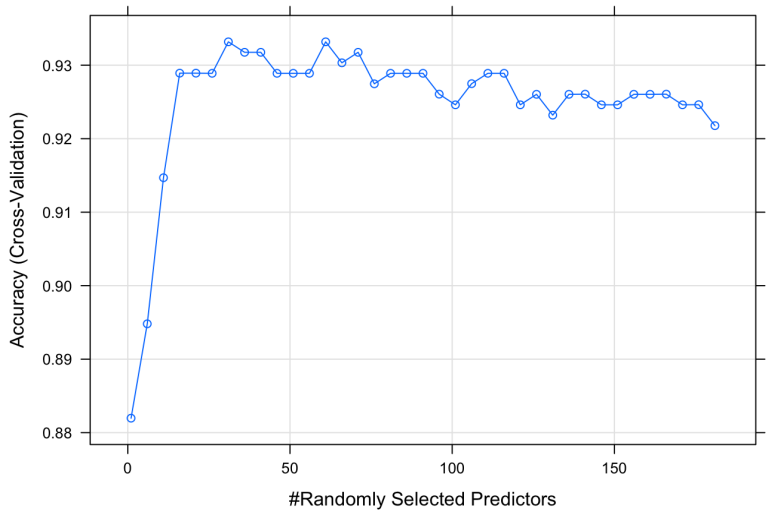


Initially Payable to Institution	
Initially Payable	80
Not Initially Payable	4
Not Applicable	20
Exempt	0
Action Taken	
3	104

II. <u>Classification Task</u>

Preferred Model Description

Our preferred model is Random Forest (ranger) because it creates decision trees using bootstrapped training samples from the clean dataset, like bagging. However, each split in a decision tree is picked from a randomly selected subset of predictors instead of the full set of predictors. This process decorrelates the trees, minimizing gini impurity without introducing significant bias, which improves accuracy.



With 10-fold cross-validation, the optimal hyper-parameter is 31 randomly selected predictors, as it maximizes accuracy.

Most important variables shown (out of 185)

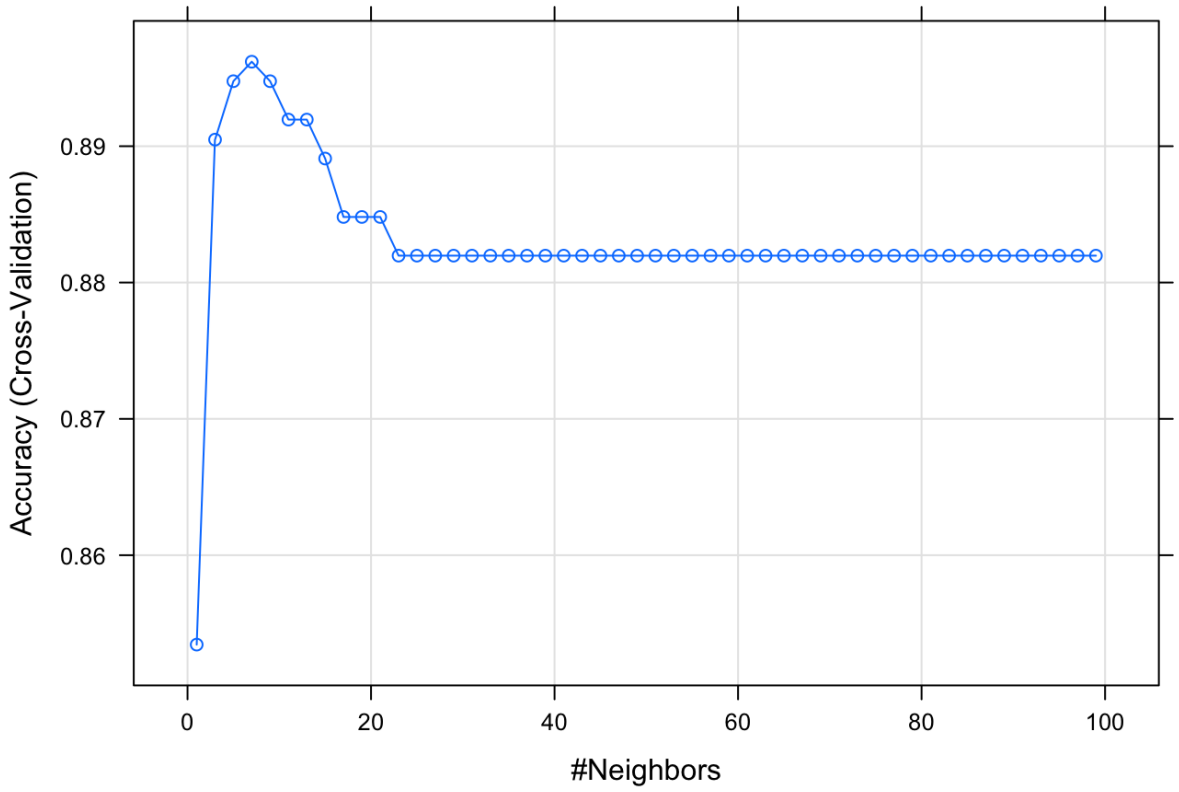
debt_to_income_ratio>60%	100.000
initially_payable_to_institution3	71.724
aus.16	29.380
income	27.091
purchaser_type1	23.136
loan_amount	17.733
debt_to_income_ratio50%-60%	14.422
tract_owner_occupied_units	14.369
aus.15	14.052
tract_one_to_four_family_homes	13.696
purchaser_type71	12.833
tract_population	12.476
property_value	12.461
purchaser_type3	11.148

The most important variable in predicting whether or not the loan was denied is the applicant's total monthly debt to total monthly income (*debt_to_income_ratio*). The second most predictive variable is whether the obligation arising from the loan would have been initially payable to the financial institution with the Not Applicable value (*initially_payable_to_institution3*). The third

most important variable is the automated underwriting system used to evaluate applications with the Not Applicable value (*aus.1*). Finally, the fourth most important variable is the applicant's gross annual income (*income*).

Less Preferred Model Description

We chose to compare Random Forest to the K-nearest neighbors algorithm (KNN). For classification of a novel data point, KNN selects an odd number of neighbors, finds some number K of its *nearest neighbors* in the training data (using a distance metric), and assigns the data point to the category with the most members, out of its K nearest neighbors. Higher K values reduce the chance of overfitting, meaning they decrease variance - however, they increase bias as they reduce the flexibility of the model.



The optimal K value is 7 nearest neighbors because it had the highest accuracy during cross-validation (tested against subsets of the training data set) as seen in the graph.

Model Performance

Our preferred model of Random Forest (ranger) accurately classified 166 data points and inaccurately classified 10 of them, resulting in a prediction accuracy of approximately 94.32% with the clean testing dataset. Based on the confusion matrix below, we find that Random Forest accurately predicts acceptance of the loan (action_taken = 1), and has greater than 50% accuracy when predicting denial of loans (action_taken = 3).

Random Forest Confusion Matrix

	Reference	
Prediction	1	3
1	155	10
3	0	11

The less preferred model is KNN which accurately classified 156 data points and inaccurately classified 20 of them, resulting in a prediction accuracy of approximately 88.64% with the clean testing dataset. Based on the confusion matrix below, we find that KNN accurately predicts acceptance of the loan (action_taken = 1), but it fails to predict denial of loans properly (action_taken = 3).

KNN Confusion Matrix

	Reference	
Prediction	1	3
1	155	20
3	0	1

Why we chose the Random Forest method compared to all other methods

With KNN models, as the value of k (number of nearest neighbors) increases, variance decreases to reduce overfitting. Additionally, points of loan denial and acceptance are closely clustered and our dataset has far more acceptances than denials, so KNN assumes all but 1 person were accepted for a loan. Whereas, Random Forest learns more effectively from the clean dataset, and de-correlates trees by selecting a subset of predictors, thus reducing variance better than KNN. Compared to bagging which uses the full set of predictors, by selecting a random subset of predictors, Random Forest allows other variables to be the strongest predictors, which can make the classifications less variable. Ordinary least squares (OLS) models minimize the sum of squared residuals, Ridge regression shrinks the least important but still retains all covariates, Lasso regression shrinks less important variables but it can eliminate unimportant ones, and a single decision tree is limited to the classification it makes. In contrast to these models, Random

Forests are preferred since they are made using bootstrapped training samples, which averages the classifications made by the trees, resulting in greater accuracy. In general, boosting is effective on models with high bias but low variance because each tree grows based on existing trees. However, our dataset's main problem was high variance, so we chose random forest which decorrelates trees to minimize overfitting. Logistic regression is not as complex and often uses less parameters, resulting in less accurate results compared to random forests.

III. Regression Task

Preferred Model Description

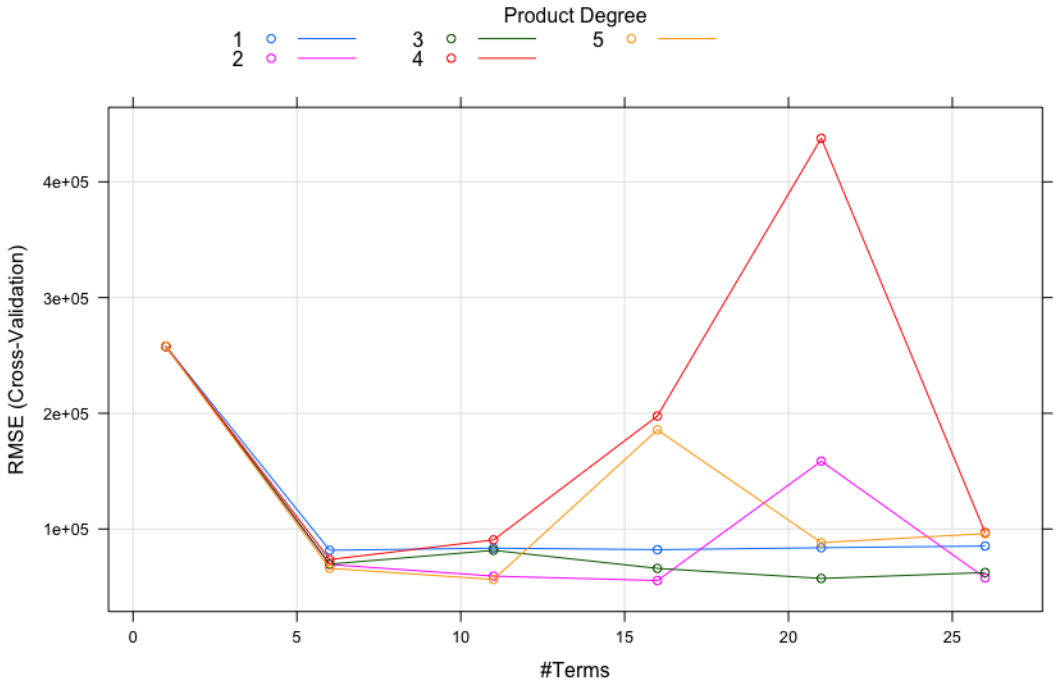
We chose to test several models, out of which the top two were the Bagged MARS (multivariate adaptive regression splines) model and Random Forest (ranger). Based on our testing, we found that Bagged MARS had a RMSE of 47,807, while Random Forest (ranger) had a RMSE of 193,871. Because Bagged MARS had a lower RMSE, we preferred it over Random Forest.

Below, we can see a tuning graph for the hyperparameters of the Bagged MARS model. We tuned both the "degree" and "number of terms" hyperparameters for this model and found that this model performed best when degrees = 2 and terms is between 10 and 20. Unfortunately, due to the runtime complexity of this model, we were unable to tune it further on a larger search grid, since every iteration of this model took around 5 minutes to complete and frequently saw crashes. However, even with these limitations, we were still able to achieve a lower RMSE on this model versus other simpler models.

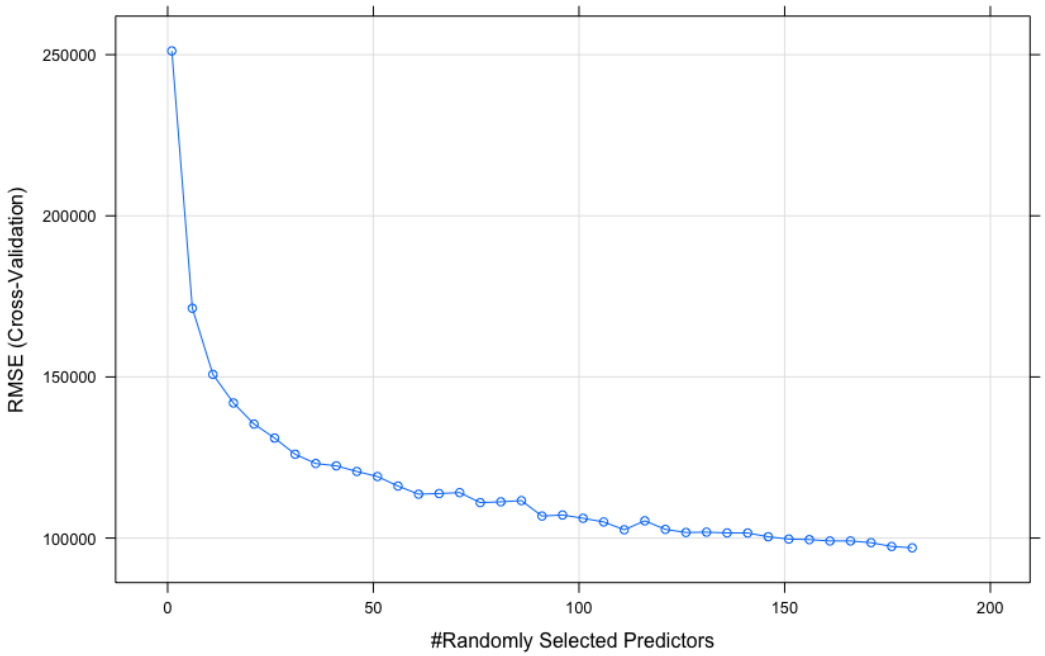
Less Preferred Model Description

Random Forest (ranger) uses bootstrapped training samples from the clean dataset, and each split in a decision tree is picked from a randomly selected subset of predictors instead of the full set of predictors. This process decorrelates the trees, minimizing variance without introducing significant bias, which decreases RMSE.

Model Performance



Comparing this model to the Random Forest (ranger) model, we can see that its RMSE was significantly lower than the RMSE found in the RF model on the testing data set.



Even when looking at the optimal number of randomly selected predictors for the random forest model, the cross-validation RMSE is significantly worse than the cv-RMSE for the bagged

MARS model. Analyzing these results shows definitively that the bagged MARS model makes for a better model for our application, since we achieve significantly better test RMSE and cross validation RMSE on our data.

Why we chose the bagged MARS method compared to all other methods

We tested every regression model reviewed in class as well as a neural network model, bagged MARS model, and gradient boosted trees. We chose bagged MARS because it had the lowest error on our testing data set. For our specific data, since we have lots of covariates, not a large amount of data points, and not a *very* high variance in the overall data, bagged MARS fit well to our specific circumstances. Based on information we found online, the MARS model is like a collection of linear regressions, each of which is used for a subset of the range of the data. It uses splines (a type of piecewise function, the simplest of which is a set of line segments connected at their endpoints) to approximate nonlinear data as a spline. It uses a spline best fit method to find relationships between these nonlinear data points using "knots". Finally, the model is bagged, meaning that multiple models are trained on bootstrapped samples and then aggregated into a final model (in the case of regression, they are probably averaged). In this case, looking at the model's optimizations, we are making a *large* tradeoff for better bias on our training set in order to get lower RMSEs, running the risk of overfitting on our data. However, seeing that when we evaluate this model against our test set, even with this tradeoff, performance on unseen data is fairly good, showing that our tradeoff for better bias works in our favor.

Because of the way the MARS model operates within caret, it is a non-trivial task to rank variables by importance for it, so instead, we derived our variable importance insights from the random forest model. To no surprise, property value was the best predictor of the actual approved loan amount.

property_value	100.0000
conforming_loan_limitNC	42.0208
loan_to_value_ratio	14.3139
income	5.0352

Further, we can see that loan-to-value ratio is a critical predictor of loan amount. These two variables coupled together make a lot of sense, since essentially, *property_value* * *loan_to_value_ratio* is the approximate loan amount that an applicant is applying for. If we naively run a regression of *loan_amount* = .01 * *loan_to_value_ratio* * *property_value*, our RMSE is comparable to the RMSE found by most of the statistical models we tried with this model. We saw that many of the more complex models struggled to pick up the relationship between *property_value* and *loan_to_value_ratio*, while for an average statistician analyzing the

data, it would be very obvious to pick up on this relationship. Even trying to create a linear model with just a few variables, including the interaction term *property_value : loan_to_value* yielded very similar results to our naive formula of calculating the final loan value, with no added value from bringing the individual terms as covariates.

Comparing our chosen model bagged MARS between the other models we learned in class, we can see that there are not many that can compare to the level of accuracy we get from bagged MARS. Some of the methods discussed in class are not relevant to this example, such as logistic regression, KNN, and other classification methods. Other models, such as a simple linear model, lasso, and ridge regression, work well as long as we explicitly define the property value and loan to value ratio relationship in the train function. If this relationship is not explicitly defined, the accuracy is very poor and the models are not able to pick up on this relationship. When assessing trees and boosted models, again we have similar levels of performance to the random forest, and the RMSE is reasonable, but in general, they pale in comparison to the bagged MARS model, which is able to pick up on the more intricate relationships in the data.

IV. Brief Conclusions

Overall recommendations

Based on our experience working with this data, it is clear that creating a model around acceptances and denials is possible, but providing more training and testing data to this model could allow for better future results. Based on the previously provided figures, we can see our models learning general relationships between various covariates, such as credit score and payment to income ratio, to create a usable model for prediction, but some of the more specific covariates that could point to mortgage denials do not stand out within the sample, because of the limited number of denials available to train and test our models.

Assessment of our Models' Performance

Looking at the various RMSE and accuracy metrics we had for our models, we are satisfied with the results of our modeling efforts. There are clear, logical, explainable trends found within the dataset, that we can model in various ways. For example, there is a very clear relationship between having a high debt to income ratio and being denied for a mortgage. This is a logical relationship, considering that lenders would consider individuals with a higher debt to income ratio to have a greater risk of default, and would therefore deny their mortgage applications at a higher rate than the average Travis County resident.

Critique of our Methodology

In general, we placed a large emphasis on avoiding the overfitting problem with our finalized models. However, this is easier said than done, since the sheer fact of us hand-selecting a model based on *test* error is a factor in overfitting the trained models on our data selection. In other

words, even though each individual model only sees the training data, our model selection algorithms (highest accuracy and lowest RMSE on the testing data) are highly linked with the testing data set, which could cause an overfitting effect on the final selected model. There are not a lot of ways we can avoid this.

Interesting Future Assessments

Seeing that there is a large trove of data available, it would be interesting to do a more complete analysis on mortgage acceptance/denial rates between different races and ethnicities in the United States. If we want to derive any sort of causal inference from our dataset, it is a good opportunity to use matching in our dataset, to match individuals that have similar application details (i.e. credit score, debt to income ratio, property value, etc) and see if there are statistically significant differences in the acceptance or denial of their applications. With this information, and a deeper analysis into the various balance aspects of the matching process, we could derive interesting insights on mortgage denial/acceptance rates across races and ethnicities.

V. References

Functions:

An Introduction to R,

https://cran.r-project.org/doc/manuals/r-release/R-intro.html#Writing-your-own-functions.

"General Interface for Bagged Mars Models - bag_mars." - *bag_mars • Baguette*,

https://baguette.tidymodels.org/reference/bag_mars.html.

Greenwell, Bradley Boehmke & Brandon. "Hands-on Machine Learning with R." *Chapter 7*

Multivariate Adaptive Regression Splines, 1 Feb. 2020,

https://bradleyboehmke.github.io/HOML/mars.html.

James, G. et al. (2021). "An Introduction to Statistical Learning with Applications in R" (ISLR).

"Mars3.6-Bag-1." *MARS with Bagging*,

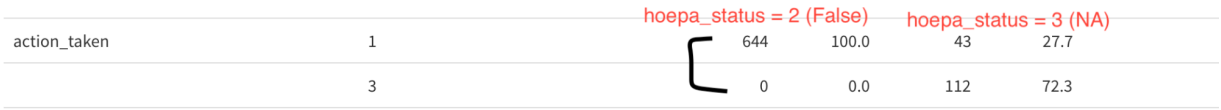
http://www.cs.toronto.edu/~delve/methods/mars3.6-bag-1/mars3.6-bag-1.html.

VI. Appendix

Data Improvements Needed

After working with this dataset, it is quite clear that there are lots of correlations that we are able to pick up between the covariates and the outcomes we tried to predict. However, we recommend much more cleaning of the source data in order to derive better insights. Some of this cleaning must be done by the suppliers of the data, as we found that multiple covariates were vaguely defined or raised questions about whether they could be considered predictors - even the source data codebook did not make it clear where some covariates were stemming from. More insight on the data collection and publishing process from the source dataset would be very helpful when trying to come up with models around these datasets in the future.

Figure 1 - Filtered balance table, split by hoepa_status



We can see that hoepa status is only given once a loan has been accepted.
Some loans are still accepted when hoepa_status is unavailable, this could just be for various logistical reasons.