University of California

Los Angeles

Kaggle Classification Project Report

By

Mayerli Cordero-Cortez

Eric Gallardo

Avanthika Panchapekesan

Shayan Saadat

Luke Villanueva

Stats 101C:

Introduction to Statistical Models and Data Mining

Professor Miles Chen

15 September 2023

Contents

1 Introduction

Loan applications come from a diverse range of applicants, each generally comprising unique financial profiles and credit histories. Institutions often face tough decisions about whether to approve or deny a loan application since this can impact their profits and risk management. However, the decision taken during the loan application process can also have both negative and positive impacts for the applicant. For example, a loan approval can enable someone to start a business or pursue higher education. However, if a loan is denied, it can mean that future borrowing for someone becomes more challenging or an individual is forced to defer personal ambitions.

This paper aims to build a predictive model for classifying the outcome of mortgage loan applications. The data for this project contains information about mortgage loans from the Wells Fargo National Bank for the year 2019. Our response variable is `action_taken`, where `1` represents loan originated or approved, while `3` denotes loan rejected or denied. The dataset also contains numerous attributes or variables that may explain loan outcomes. To mention a few, potential explanatory variables of loan outcome, includes, but are not limited to quantitative attributes, such as loan amount, term, property value, and income. To illustrate, we believe that individuals with high disposable income and properties to offer as collateral are more likely to get approved compared to their counterparts. Categorical factors, such as `loan_type`, purpose, and socio-demographic factors, such as age group, location, ethnicity, and sex may also explain loan outcomes, at least statistically (Munell). This analysis aims to uncover attributes that are significant in explaining loan outcomes, which are then exploited to build a loan predictive model that will facilitate more informed lending decisions.

2 Exploratory Data Analysis

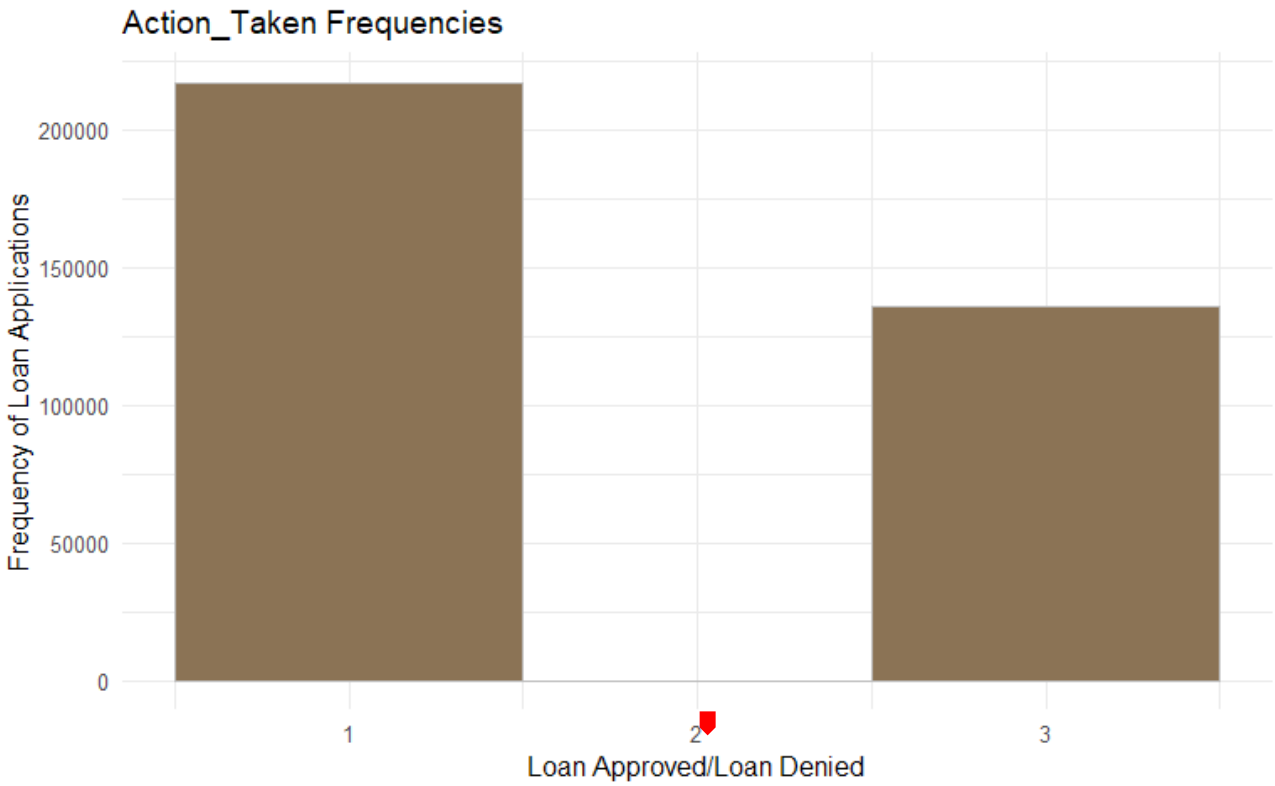


Figure 1: This figure shows the count of applicants that were approved of a loan (1) and rejected for a loan (3). This information implies that stratifying cross-validation by action_taken could be useful since there are nearly twice as many acceptances compared to rejections.

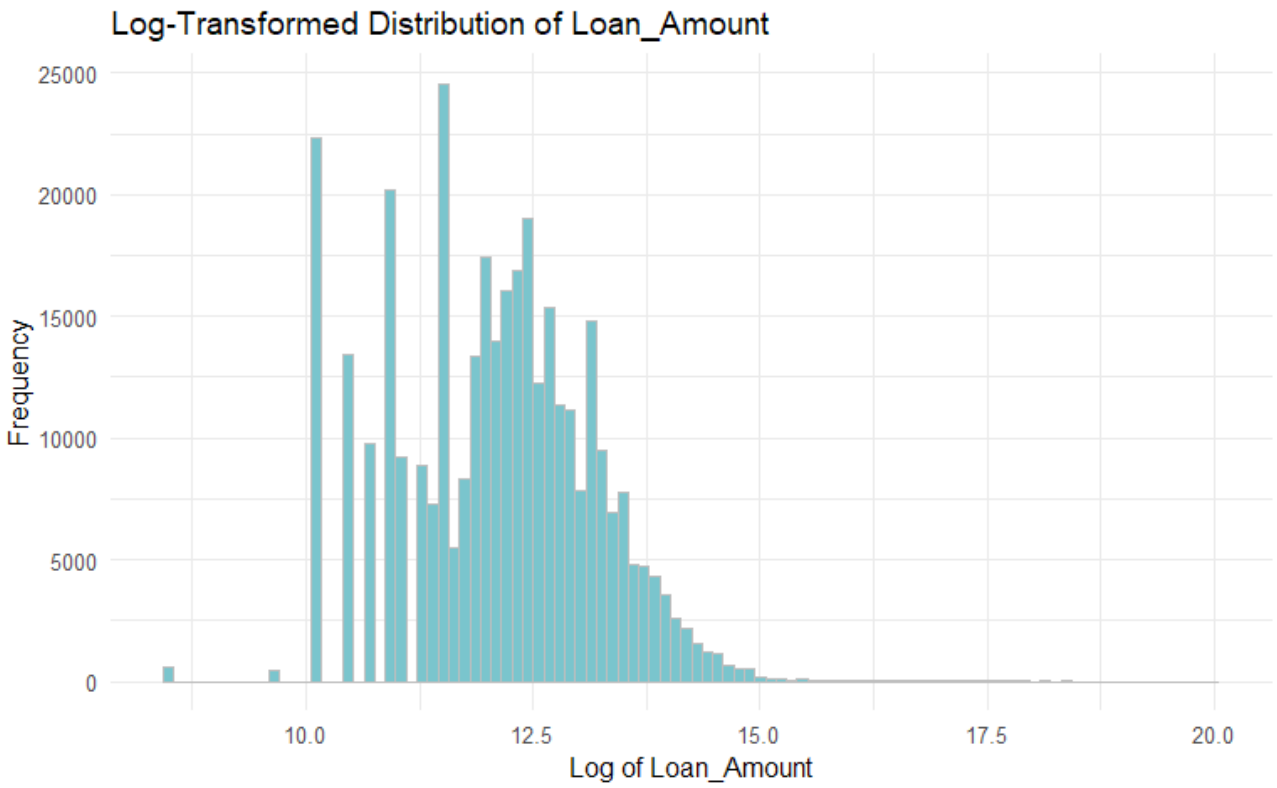


Figure 2: This figure shows a log transformation on the numerical `loan_amount` predictor. Since the data was heavily right-skewed to begin with, the transformation attempts to correct for that skew. Note that the data is still slightly right skewed even after the transformation.

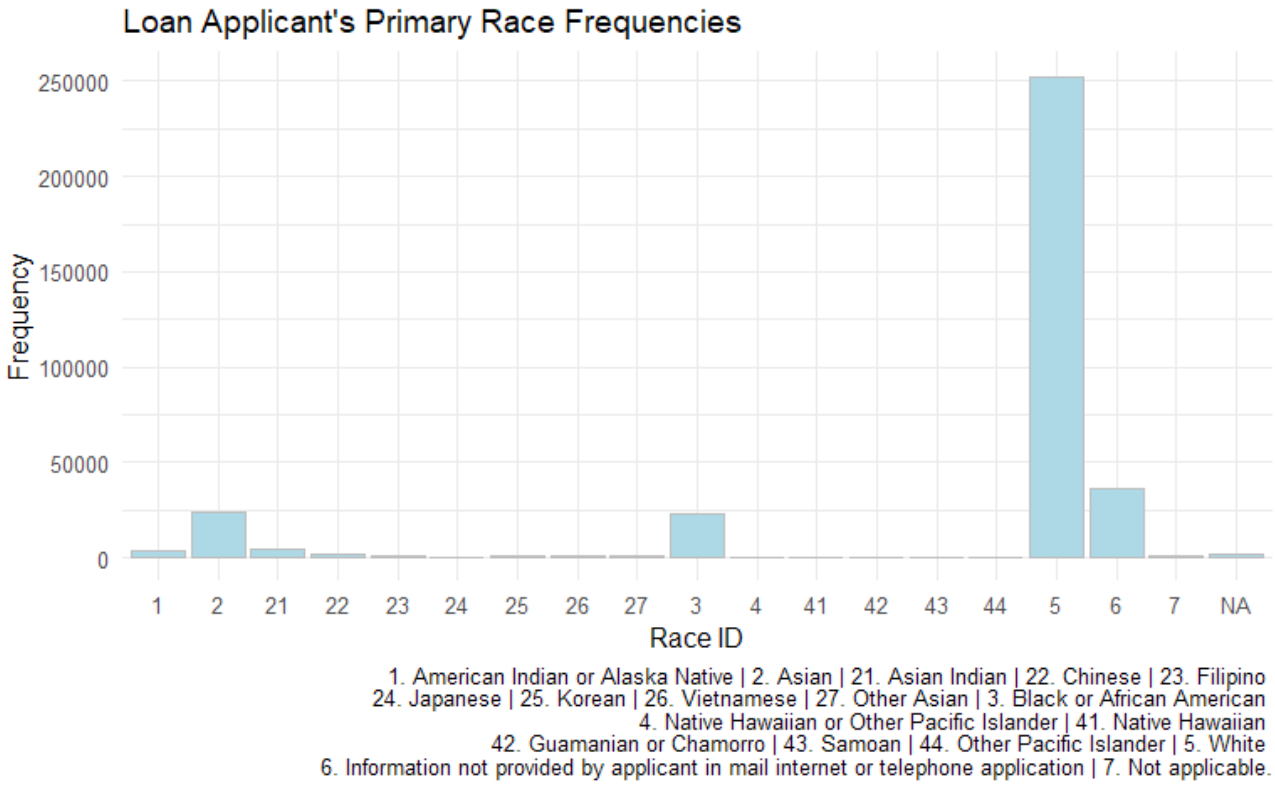


Figure 3: The above figure notes that the data is heavily imbalanced by race and that the majority of loan applicants are white. It's also worth noting that Asian is a broad category of race, but that there are also subcategories of Asian like Indian, Filipino, Japanese, etc… included within the dataset.

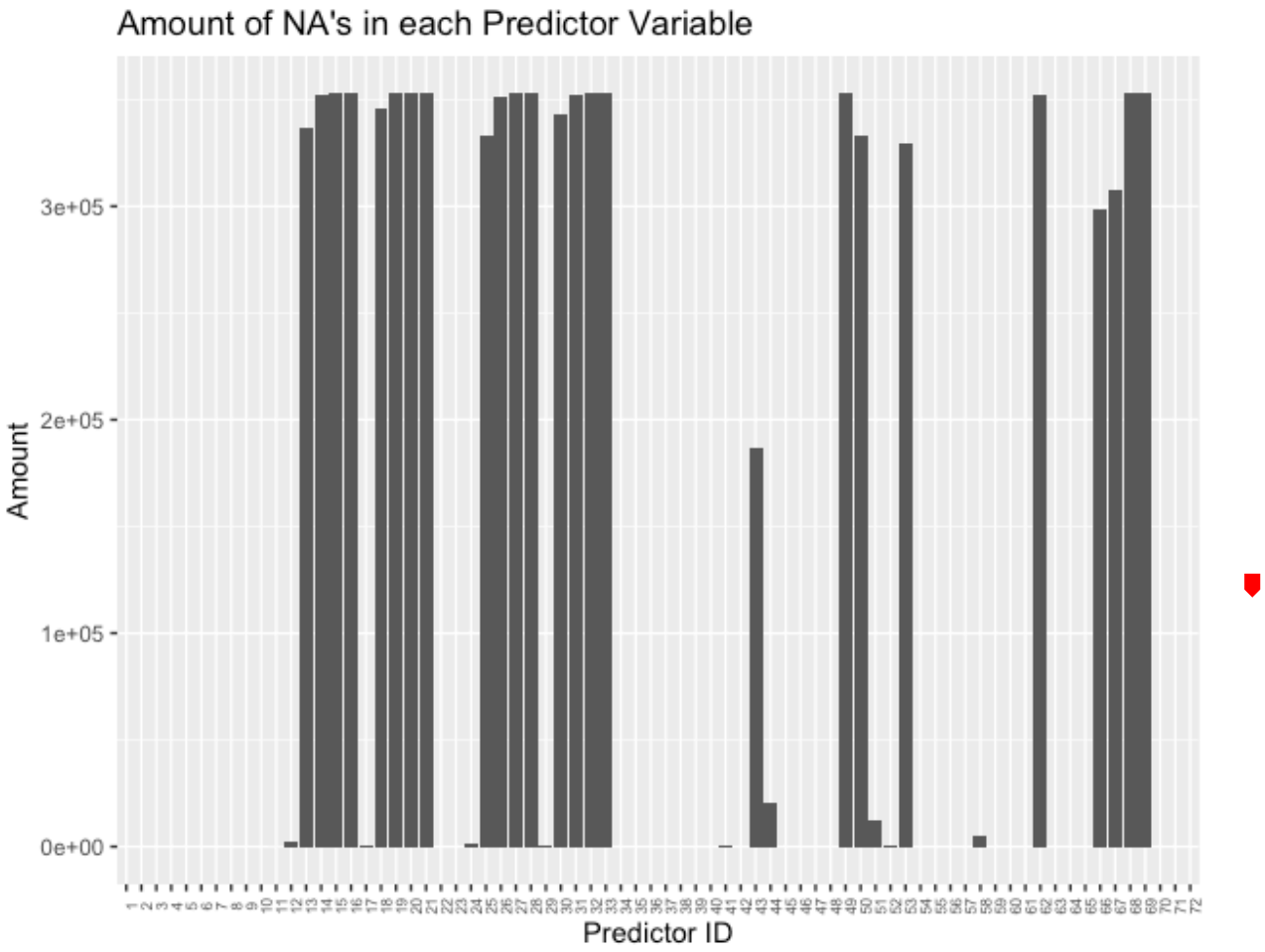


Figure 4: This figure shows all the columns of the training data with ID's representing the column. ID 1 is the column `id`, ID 2 is `action_taken`, and the following IDs are the predictor variables as listed in the order of columns. According to this graph, there are some columns that are predominantly missing values and removing these predictors like the 4th or 5th race or ethnicity of the co-applicant could be useful in reducing model complexity.

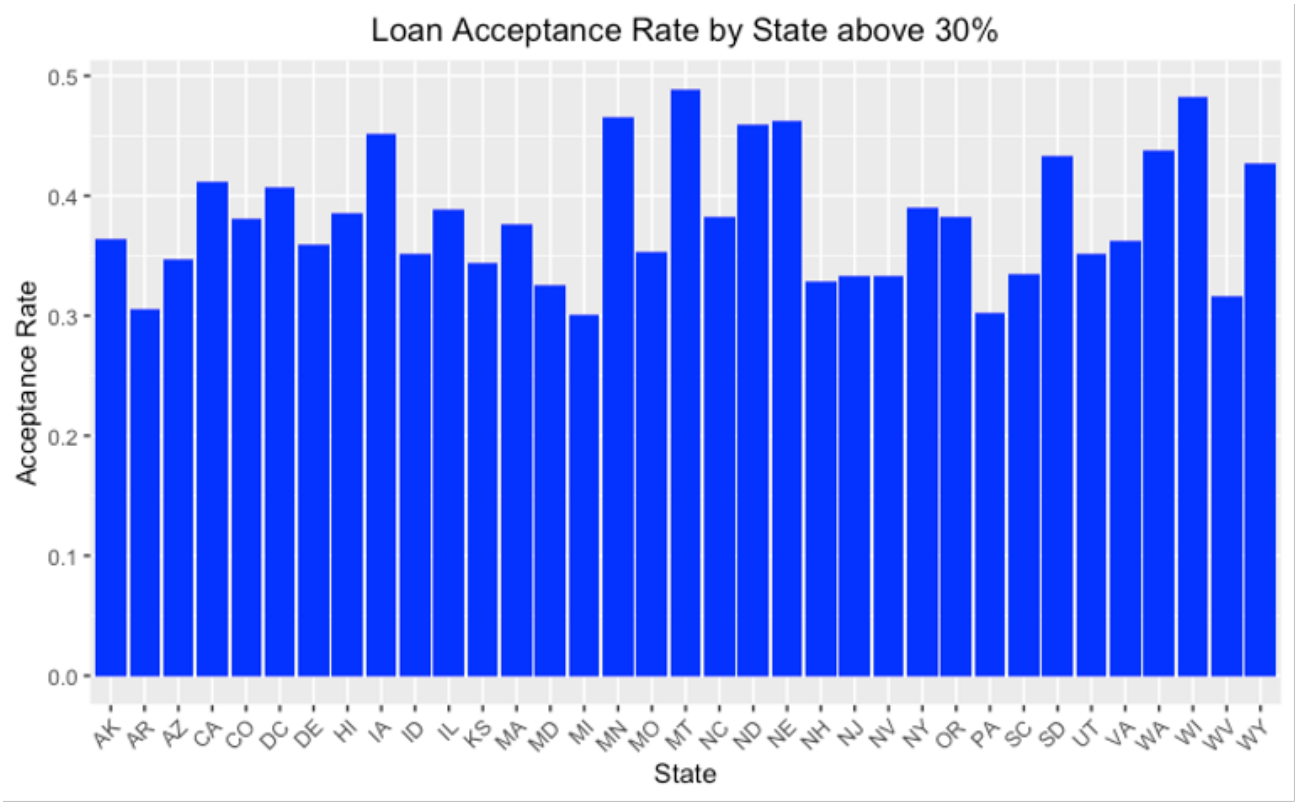


Figure 5: This figure shows the loan acceptance rate by US states. Based on the chart, the loan acceptance rate appears to vary moderately by state, with Montana (MT) recording the highest proportion of loans originating, followed closely by Winsconsin (WI). For Montana and Wisconsin, close to 50% of the applied loans are accepted. This visualization shows how dependent variables vary by state, indicating the need to include it as a feature during model training.

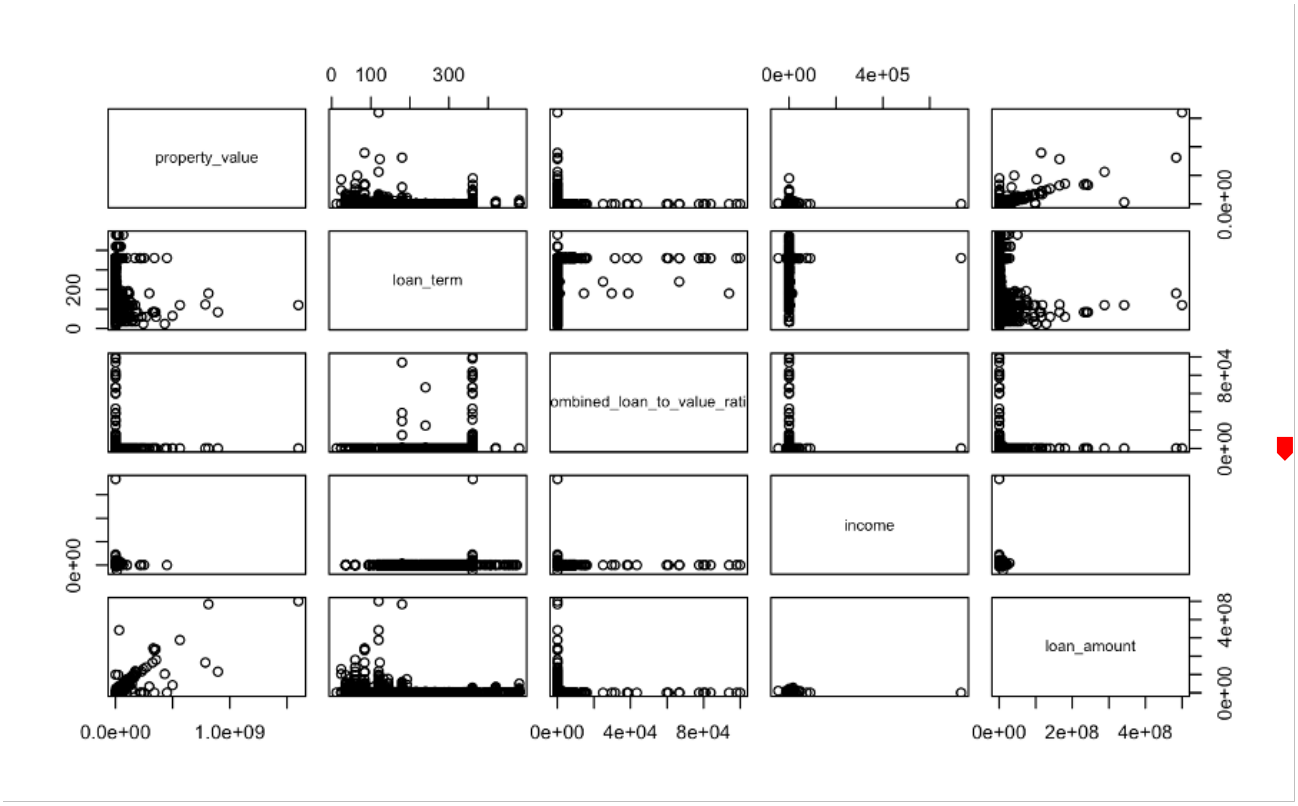


Figure 6: The scatterplots above visualize the relationship between a pair of numeric variables in our dataset, namely, property value, loan term, combined loan-to-value ratio, income, and loan amount. Based on the plots, we cannot discern a clear relationship between the variables. However, there appears to be a moderate positive relationship between property value and loan amount, indicating that loan amount is likely linked to the appraised value or purchase price of the property, which is a common practice in mortgage lending to ascertain credibility (Cooper, 2013).

```
$`1`
 property_value       loan_term   combined_loan_to_value_ratio     income         loan_amount
 Min.   :5.00e+03   Min.   : 12   Min.   :  2.07              Min.   : -73.0   Min.   :     5000
 1st Qu.:2.65e+05   1st Qu.:360   1st Qu.: 58.33              1st Qu.:   79.0   1st Qu.:   145000
 Median :4.15e+05   Median :360   Median : 74.93              Median :  129.0   Median :   255000
 Mean   :7.70e+05   Mean   :331   Mean   : 69.52              Mean   :  199.4   Mean   :   431146
 3rd Qu.:7.85e+05   3rd Qu.:360   3rd Qu.: 80.00              3rd Qu.:  221.0   3rd Qu.:   485000
 Max.   :1.60e+09   Max.   :480   Max.   :183.45              Max.   :46554.0   Max.   :500005000
 NA's   :2789                     NA's   :5242                NA's   :11698

$`3`
 property_value        loan_term     combined_loan_to_value_ratio     income          loan_amount
 Min.   :     5000   Min.   : 36.0   Min.   :     0.02           Min.   :-46378.0   Min.   :    5000
 1st Qu.:   195000   1st Qu.:360.0   1st Qu.:    52.28           1st Qu.:    49.0   1st Qu.:   45000
 Median :   305000   Median :361.0   Median :    71.88           Median :    80.0   Median :  105000
 Mean   :   486421   Mean   :343.2   Mean   :    82.97           Mean   :   124.5   Mean   :  193313
 3rd Qu.:   505000   3rd Qu.:361.0   3rd Qu.:    81.82           3rd Qu.:   132.0   3rd Qu.:  215000
 Max.   :450005000   Max.   :474.0   Max.   :100000.00           Max.   :734000.0   Max.   :57005000
 NA's   :2127        NA's   :365     NA's   :6830                NA's   :8499
```

Figure 7: The table shows the summary statistics of the numeric variables by action taken. The mean property value for loans originated is $770000, compared to $486421 for loans denied. Individuals whose loan applications were rejected also had lower average income and combined loan-to-value ratio compared to their counterparts. We also notice that denied loans also exhibit higher spread in regards to property value and income, as evidenced by the significant range between max and min values. This indicates greater variability in the characteristics of applicants whose loan applications are denied.

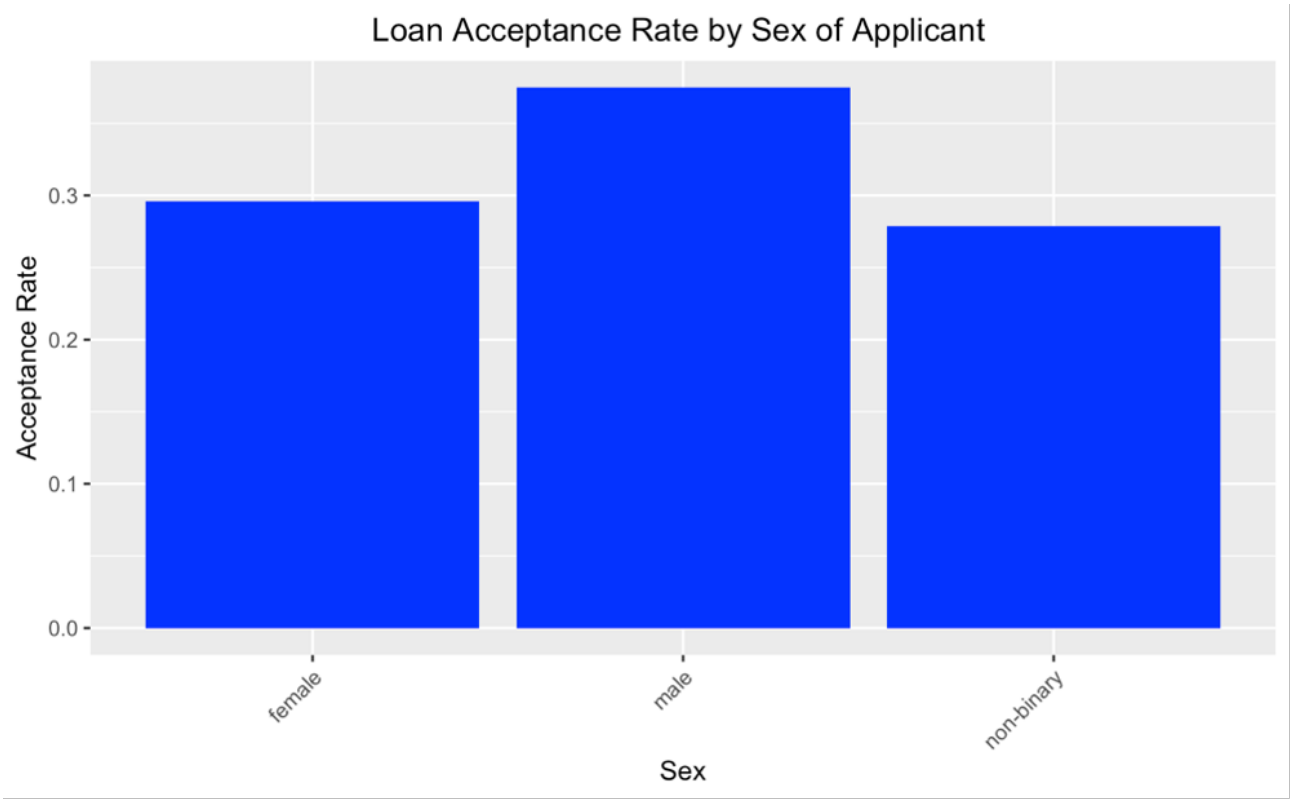


Figure 8: This bar chart of loan acceptance rate by sex shows that male applicants are more likely to get loans approved, compared to female and non-binary applicants, which are below 30%. This visualization is relevant in our analysis as it shows moderate variation in loan acceptance rate by sex of the applicant or borrower - sex is a moderate feature to include in our model for predicting action taken.

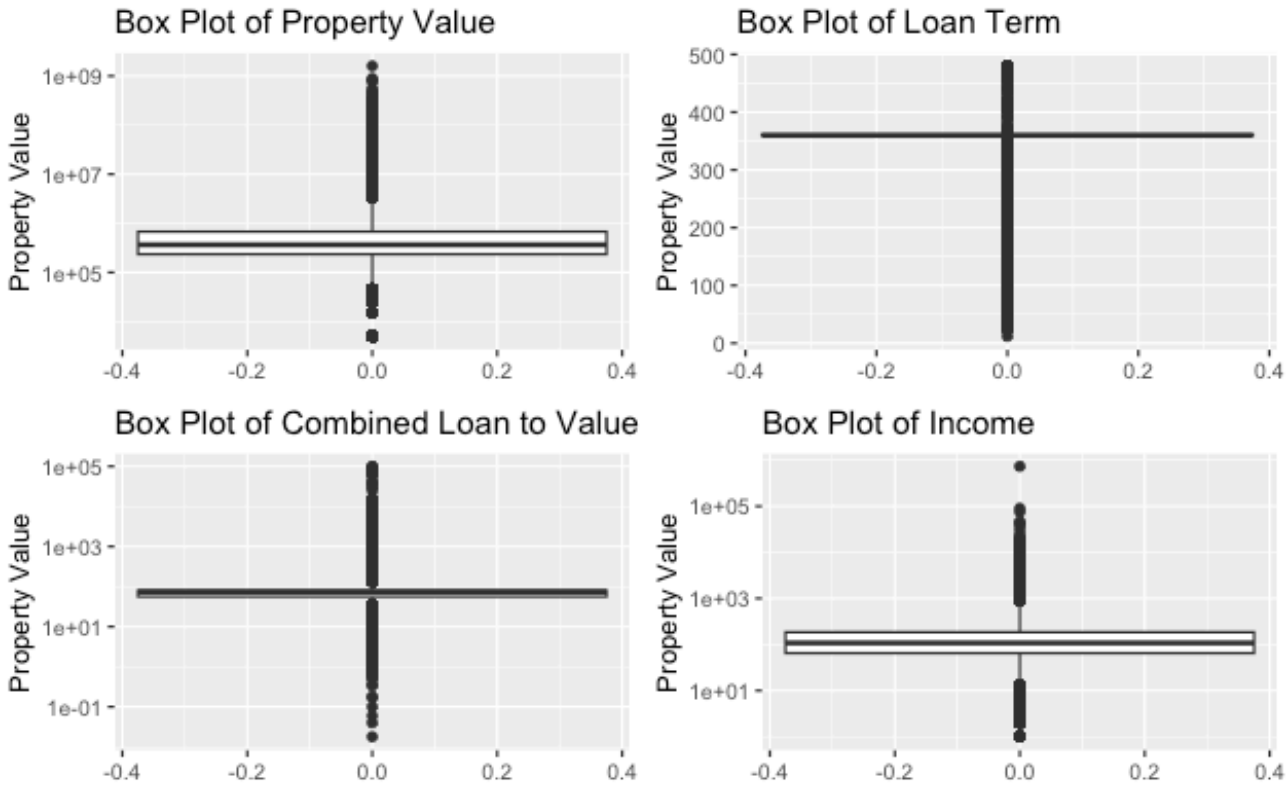


Figure 9: This figure represents a side-to-side boxplot of the numeric variables. There is the presence of numerous outliers, as evidenced by the points outside the upper and lower whiskers. This visualization supports the need to perform further processing, in particular, potentially identifying and removing outliers present in numerical variables and transforming some of the numeric variables.

3 Preprocessing and Recipes

3.1 Variable Selection

Since there are over 300,000 observations, only 10% of the training dataset will be used to test for model evaluation. This downsampled training set was obtained through random sampling.

Based on the exploratory data analysis there was a certain amount of data cleaning that was necessary prior to recipe creation.

The following variables were selected for removal on the basis of having too many missing values in the training and testing sets:

- `id`
- `activity_year`
- `legal_entity_identifier_lei`
- `total_points_and_fees`
- `introductory_rate_period`
- `multifamily_affordable_units`
- `ethnicity_of_co_applicant_or_co_borrower_4`
- `ethnicity_of_co_applicant_or_co_borrower_5`
- `race_of_co_applicant_or_co_borrower_4`
- `race_of_co_applicant_or_co_borrower_5`

Additionally to output classification modeling, `action_taken` in the training set was turned into a `factor` class instead of `numeric`.

3.2 Recipe

```
# SET WHICH RECIPE TO USE
rec <- recipe(action_taken ~ ., data = train_train)

# imputation via knn took forever, so switched to linear regression imputation
rec <- rec %>%
  # filtering NA's from predictors with few NA's
  step_filter(!is.na(loan_term)) %>%
  # mutating in prep for dummy encoding
```

```
  step_mutate_at(starts_with("ethnicity_of"),
                 fn = function(x) if_else(is.na(x), 4, x)) %>%
  step_mutate_at(starts_with("race_of"),
                 fn = function(x) if_else(is.na(x), 7, x)) %>%
  step_mutate_at(starts_with("automated_underwriting_system"),
                 fn = function(x) if_else(is.na(x), 6, x)) %>%
  # change numeric cols into double for imputation
  step_mutate_at(loan_amount, income,
                 combined_loan_to_value_ratio,
                 property_value,
                 fn = function(x) as.double(x)) %>%
  # imputation of numeric variables, rolling works better because linear reg
  # cannot work with NA's, and KNN takes forever
  step_impute_roll(income,
                   combined_loan_to_value_ratio,
                   property_value,
                   window = 13) %>%
  # fix state NA's
  step_mutate_at(state,
                 fn = function(x) if_else(is.na(x), "NA", x)) %>%
  # fix age_of_applicant_62 NA's
  step_mutate_at(age_of_applicant_62,
                 fn = function(x) if_else(is.na(x), "NA", x)) %>%
  # fix age_of_co_applicant_62 NA's
  step_mutate_at(age_of_co_applicant_62,
                 fn = function(x) if_else(is.na(x), "NA", x)) %>%
  # filter out numeric variables that still have NA's
  step_filter(!is.na(income)
              & !is.na(combined_loan_to_value_ratio)
              & !is.na(property_value)) %>%
  # since income can be negative, it cannot log transform correctly,
  # so must shift by minimum of income so lowest would be value of 1
  step_mutate_at(income,
                 fn = function(x) x <- abs(min(x)) + x + 1) %>%
  # transformation of numeric variables
```

```
  step_log(loan_amount,
           income,
           combined_loan_to_value_ratio,
           property_value) %>%
  # change all nominal variables into characters
  step_mutate_at(all_predictors(),
                 -loan_amount,
                 -income,
                 -combined_loan_to_value_ratio,
                 -property_value,
                 fn = function(x) as.factor(x)) %>%
  # remove zero variance variables
  step_zv(all_predictors()) %>%
  # remove near zero variance variables (for LDA to sorta work better)
  step_nzv(all_predictors()) %>%
  # dummy encoding of nominal variables
  step_dummy(all_predictors(),
             -loan_amount,
             -income,
             -combined_loan_to_value_ratio,
             -property_value)
```

After removing the variables in the preprocessing step, the recipe was based on classifying `action_taken` on the remaining variables. In this recipe, all observations with missing `loan_term` values were removed. Categorical predictors with both mixed `NA` values and a numeric value representing `NA` values were merged into one predictor. Missing values in numerical predictors were imputed using the rolling window method and if any missing values were still present, they were converted into the character `NA`. Additionally, since the numeric predictors were heavily right skewed, entries were incremented by one to force non-zero entries in these predictors and the numeric predictor variables were log-transformed and normalized. Predictors that had zero or near-zero variance were removed. Then all nominal predictors were turned into dummy variables for the purpose of classification modeling.

4 Models

4.1 Overview:

- The first model that'll be evaluated is the boosted tree model. The model will use the `xgboost` engine and follow the standard recipe given above. The only hyperparameter tuned for this model is `learn_rate`.

- The second model tested is Decision tree which will use the `rpart` engine with the standard recipe shown prior. The model will tune for the hyperparameter of cost complexity.

- The third model that will be evaluated is the logistic regression model which will follow the `glm` engine using the standard recipe. There will be no hyperparameters set or tuned for this model. Later on, the logistic regression model will use the `glmnet` engine in order to tune for penalty and mixture hyperparameters in hopes of improving logistic regression predictive results.

- The fourth model used is the naive Bayes model using the engine `klaR` following the same recipe. It is tuned for the hyperparameter smoothness.

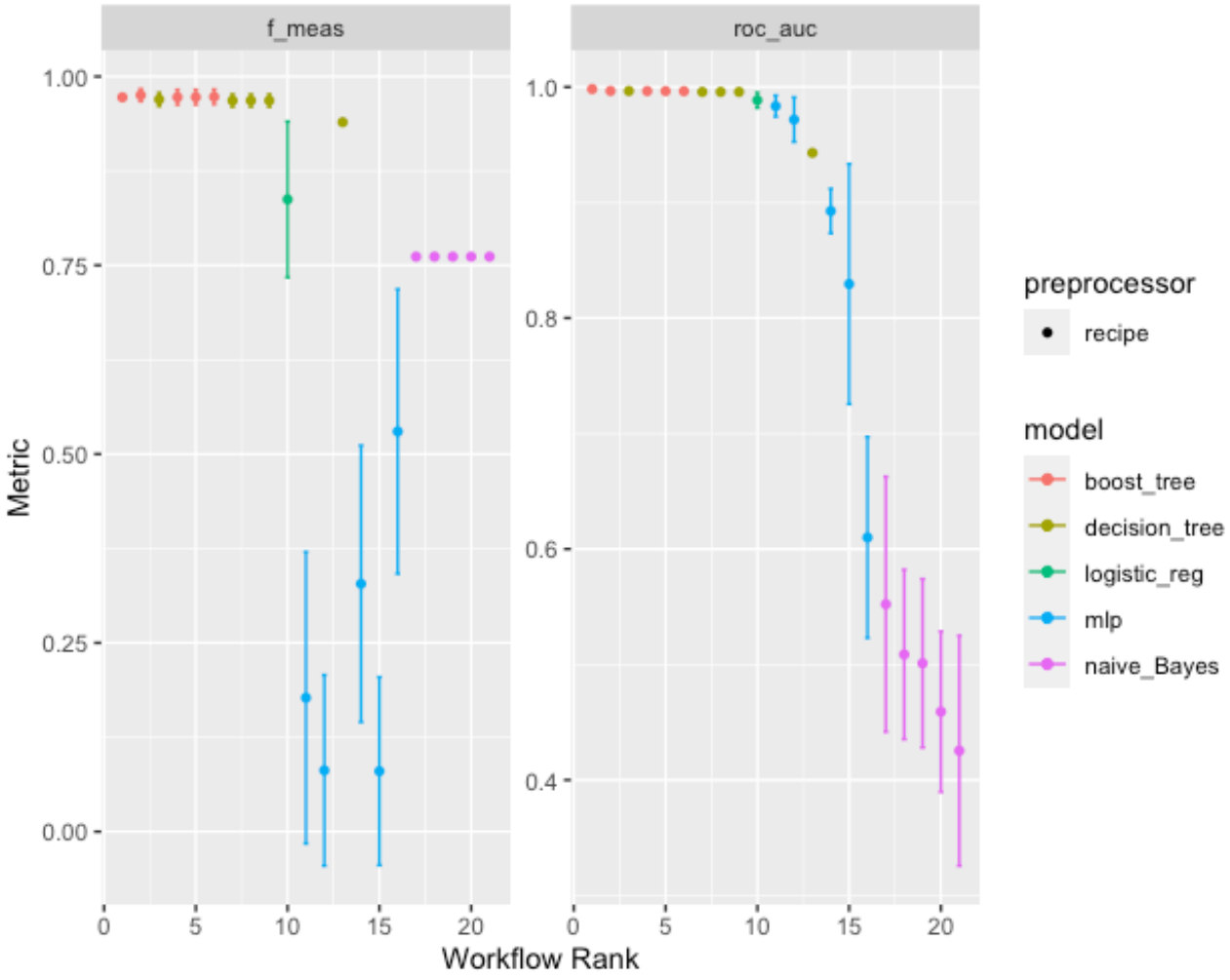
- The last model used is a neural network model running on the `brulee` engine using the same recipe. This model has the hyperparameters hidden units and learn rate which will be tuned for. This model also has the hyperparameter `epochs` which will not be tuned, but set to 500. Additionally the activation for this model is set to the rectified linear unit or `relu` for short.

4.2 Evaluation

ID	Type	Engine	Recipe	Hyperparameters	F-Score	Standard Error
1	`boost_tree`	`xgboost`	`rec`	`learn_rate`	0.9753	0.00497
2	`decision_tree`	`rpart`	`rec`	`cost_complexity`	0.9697	0.00538
3	`log_reg`	`glm`	`rec`	`X`	0.8372	0.06274
4	`naive_Bayes`	`klaR`	`rec`	`smoothness`	0.7618	0.00001
5	`mlp`	`brulee`	`rec`	`hidden_units`, `learn_rate`, `epochs = 500`	0.5299	0.11455

The above table reports the five models used in a workflow set for model evaluation to obtain comparison based F-scores using tuning grids and cross validation on only 10% of the training data.

Note, that the sample training folds used for tuning and cross validation were stratified by `action_taken` due to the lower occurrence of denied loan applications relative to loan origins. Based on the F-score results, boosted tree based models were the most accurate of the tested models, while the neural network model was the worst performing. The results of the tuning grid and cross validation for these 5 models on the same recipe are also reported in the below plot:



The following chart is the second phase of model evaluation. Here, two of the top models in the previous evaluation were used for further tuning. Note that now the logistic regression model is using the `glmnet` engine and is being tuned for hyperparameters.

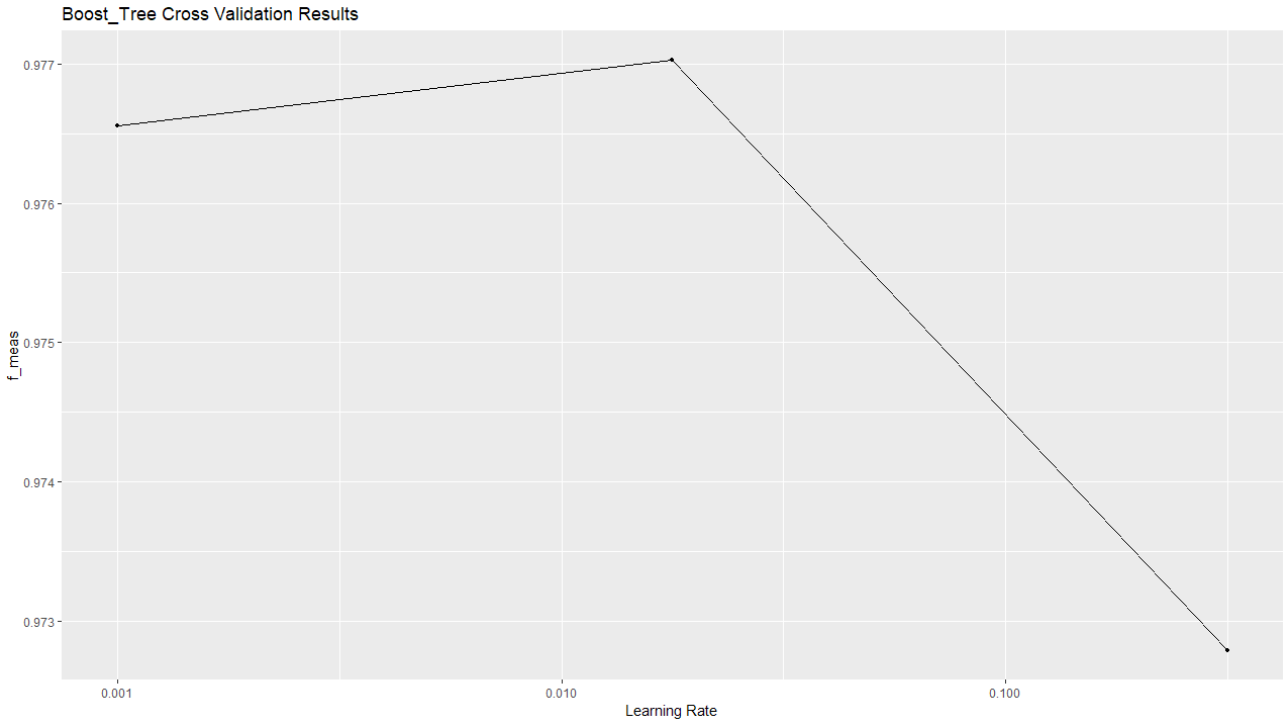
ID	Type	Engine	Recipe	Hyperparameters	F-Score	Standard Error
1	`boost_tree`	`xgboost`	`rec`	`learn_rate`	0.9766	X
2	`boost_tree`	`xgboost`	`rec`	`learn_rate`	0.9771	X
3	`boost_tree`	`xgboost`	`rec`	`learn_rate`	0.9727	X
4	`log_reg`	`glmnet`	`rec`	`penalty`, `mixture`	0.943	X
5	`log_reg`	`glmnet`	`rec`	`penalty`, `mixture`	0.956	X
6	`log_reg`	`glmnet`	`rec`	`penalty`, `mixture`	0.972	X

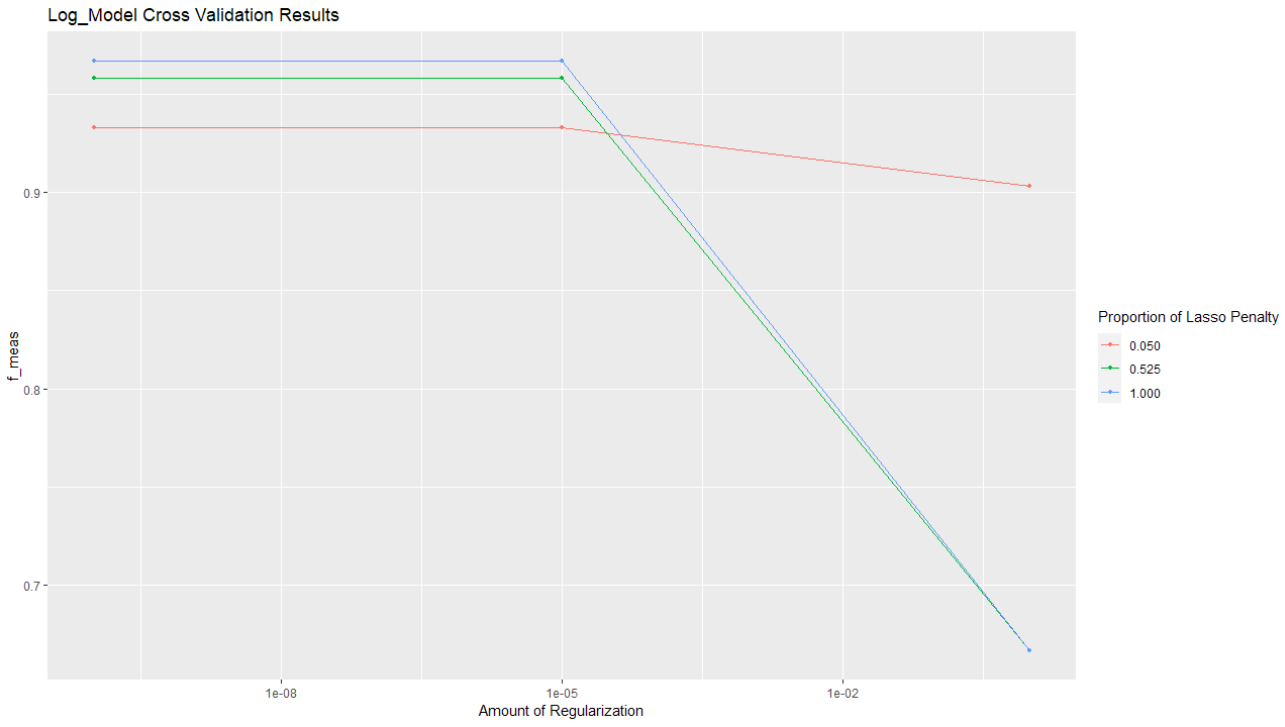
For the `boost_tree` model, the tuning was only set to one hyperparameter — `learn_rate`, while the logistic regression tested two hyperparameters — `penalty` and `mixture`.

Model Descriptions:

1. Boost tree model with `learn_rate` of 0.001
2. Boost tree model with `learn_rate` of 0.019
3. Boost tree model with `learn_rate` of 0.15
4. Logistic regression model with `penalty` of 0.05 and `mixture` of 0
5. Logistic regression model with `penalty` of 0.525 and `mixture` of 0
6. Logistic regression model with `penalty` of 1 and `mixture` of 0

Based on the F measure, the boost tree model was the most consistent model to provide the best accuracy compared to the logistic regression model. The following are the graphs to compare the results of the different hyperparameters for the different models:

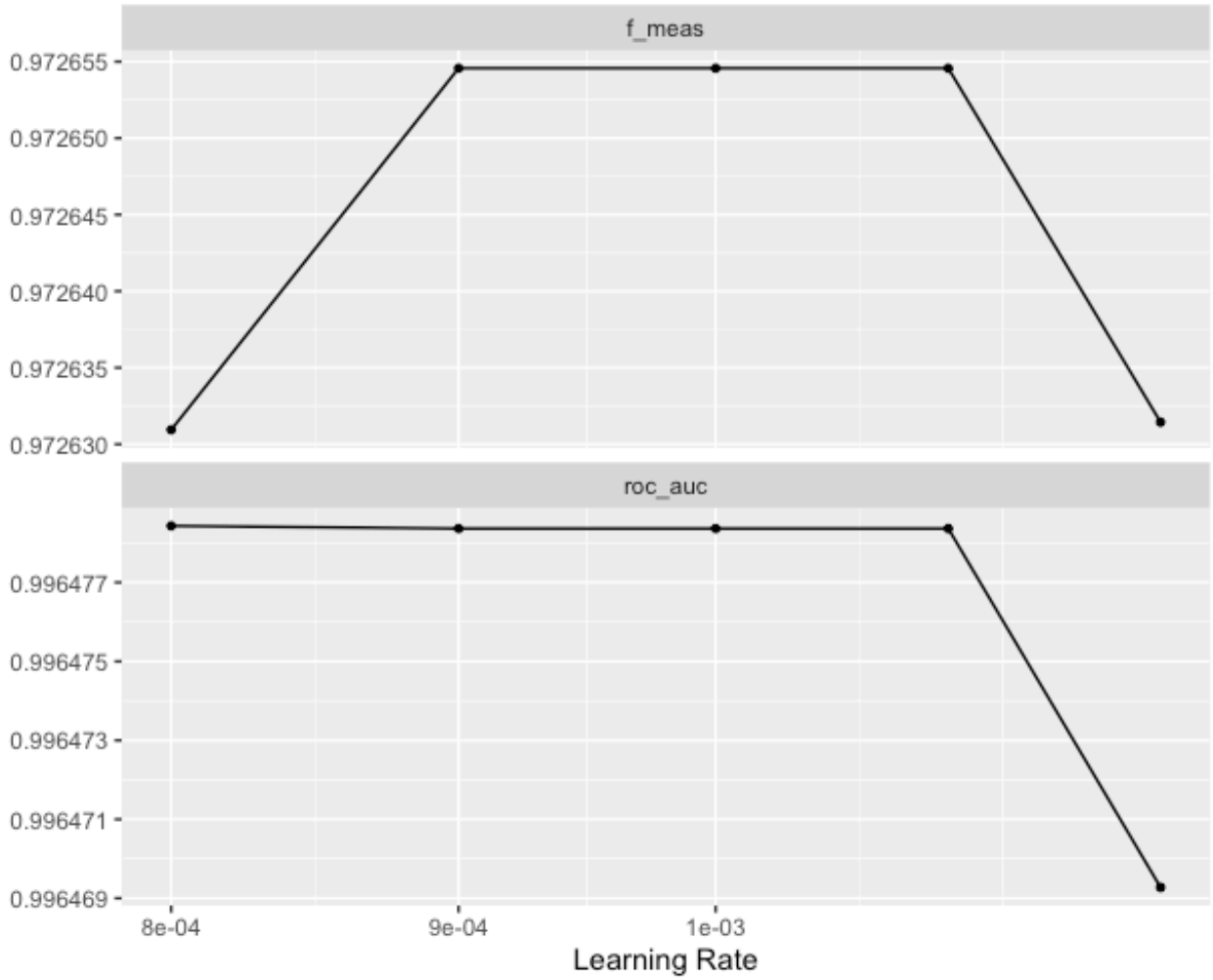




From these graphs, the best parameter setting for the boost tree model is to set the learn rate between 0 and 0.05. The best parameter setting for the logistic regression model is to have the highest penalty with the lowest mixture setting. The final model was decided to be a `boost_tree` model that would be using a small sample of the data for training.

4.3 Tuning

The best performing model as reported in the data was the boosted tree model using 10% of the training data sampled randomly. To obtain optimal tuning hyperparameters for the boosted tree model, the boosted tree model had a tuning grid to test for an optimal value for `learn_rate`. The optimal `learn_rate` value was determined to be around 0.0009 based on the downsampled training data as shown in the plot below:



Note that the learning rates shown here are from values of 0.0008 to 0.0012 in increments of 0.0001. This range was determined based on prior model tuning to narrow down the hyperparameter range for the optimal value.

Based on the results of model evaluation and hyperparameter tuning, the optimal model was the boosted tree model with a `learn_rate` of 0.0009.

Below is a table that shows the 10-fold cross validation results on the boosted tree once the `learn_rate` parameter was set to 0.0009 using the downsampled training set.

ID	Type	Engine	Recipe	Hyperparamters	F-Score	Standard Error
1	`boost_tree`	`xgboost`	`rec`	`learn_rate = 0.0009`	0.9727	0.00602

The F-score and standard error are post-tuning and cross-validation.

5 Results and Discussion

5.1 Selection of Final Model

Drawing an explanation from the previous discussion of model evaluation, the boosted tree model was selected as the final model for predicting loan outcome (denied or originated) because of its relatively high accuracy (0.9727 – based on downsampled cross-validation) in predicting loan outcome based on unknown data. With regards to predictions fitted on the full training set, the F-score achieved on the full test set was 0.98022 on the private competition submission.

5.2 Weaknesses of the Model

It should be noted that a final F-score of 0.980 was not that great of a result compared to other groups. Given more time to tune and test hyperparameters for other models like a neural network, predictive performance could've increased. More time would've also been useful for variable selection within the classification model and for fine-tuning of preprocessing and recipes.

5.3 Future Steps and Final Reflections

A plausible approach to further improve the predictive performance is to include more variables in data collection likely to explain loan outcomes. They include, but are not limited to, employment status, whether an individual has ever defaulted or not, creditworthiness, etc. Also, several variables were removed from the training set because of missing values and only one level for categorical variables. Therefore, data completeness is also a viable suggestion to improve model performance as it will reduce bias and improve generalization.

6 Appendix

6.1 Final Annotated Script

```
# Note: The submission for this script is for the results of boostedpred1.csv
# on the competition results. That was the submission that was selected for
# in the competition, but another model was autoselected by kaggle, but those
# results were not reproducible, so boostedtree1.csv was our choice for
# competition submission.

library(tidyverse)
library(tidymodels)
library(stringr)
library(yardstick)
library(xgboost)

train <- read.csv("train2.csv")
test <- read.csv("test2.csv")

colsToRemove <- c("id","activity_year",
                  "legal_entity_identifier_lei",
                  "total_points_and_fees",
                  "introductory_rate_period",
                  "multifamily_affordable_units",
                  "ethnicity_of_co_applicant_or_co_borrower_4",
                  "ethnicity_of_co_applicant_or_co_borrower_5",
                  "race_of_co_applicant_or_co_borrower_4",
                  "race_of_co_applicant_or_co_borrower_5")

# remove unnecessary cols
train <- train %>%
  dplyr::select(-colsToRemove)

# make the output a factor
train$action_taken <- as.factor(train$action_taken)
```

```r
# group numeric cols together in the beginning (income,
# loan_amount, property_value, combined_loan_to_value_ratio)
train <- train %>%
  relocate(income,
           .before = loan_type) %>%
  relocate(loan_amount,
           .before = loan_type) %>%
  relocate(property_value,
           .before = loan_type) %>%
  relocate(combined_loan_to_value_ratio,
           .before = loan_type)

# TESTING DATA
# Remove unnecessary cols
test <- test %>%
  dplyr::select(-colsToRemove)

# Group numeric cols together in the beginning (income, loan_amount,
# property_value, combined_loan_to_value_ratio)
test <- test %>%
  relocate(income,
           .before = loan_type) %>%
  relocate(loan_amount,
           .before = loan_type) %>%
  relocate(property_value,
           .before = loan_type) %>%
  relocate(combined_loan_to_value_ratio,
           .before = loan_type)

# Downsampling Randomly from the Training Set
set.seed(10)
data_split <- initial_split(train,
                            prop = .1,
                            strata = action_taken)
```

```r
train_sample <- training(data_split)

# Creating Model
boost_tree_model <- boost_tree(learn_rate = 0.0009) %>%
  set_engine("xgboost") %>%
  set_mode("classification")

# Creating recipe
rec <- recipe(action_taken ~ .,
              data = train_sample)

rec <- rec %>%
  # filtering NA's from predictors with few NA's
  step_filter(!is.na(loan_term)) %>%
  # mutating in prep for dummy encoding
  step_mutate_at(starts_with("ethnicity_of"),
                 fn = function(x) if_else(is.na(x), 4, x)) %>%
  step_mutate_at(starts_with("race_of"),
                 fn = function(x) if_else(is.na(x), 7, x)) %>%
  step_mutate_at(starts_with("automated_underwriting_system"),
                 fn = function(x) if_else(is.na(x), 6, x)) %>%
  # change numeric cols into double for imputation
  step_mutate_at(loan_amount,
                 income,
                 combined_loan_to_value_ratio,
                 property_value,
                 fn = function(x) as.double(x)) %>%
  # imputation of numeric variables, rolling works better because linear
  # reg cannot work with NA's, and KNN takes forever
  step_impute_roll(income,
                   combined_loan_to_value_ratio,
                   property_value,
                   window = 13) %>%
  # fix state NA's
```

```
  step_mutate_at(state,
                 fn = function(x) if_else(is.na(x), "NA", x)) %>%
  # fix age_of_applicant_62 NA's
  step_mutate_at(age_of_applicant_62,
                 fn = function(x) if_else(is.na(x), "NA", x)) %>%
  # fix age_of_co_applicant_62 NA's
  step_mutate_at(age_of_co_applicant_62,
                 fn = function(x) if_else(is.na(x), "NA", x)) %>%
  # filter out numeric variables that still have NA's
  step_filter(!is.na(income)
              & !is.na(combined_loan_to_value_ratio)
              & !is.na(property_value)) %>%
  # since income can be negative, it cannot log transform correctly,
  # so must shift by minimum of income so lowest would be value of 1
  step_mutate_at(income,
                 fn = function(x) x <- abs(min(x)) + x + 1) %>%
  # transformation of numeric variables
  step_log(loan_amount,
           income,
           combined_loan_to_value_ratio,
           property_value) %>%
  # change all nominal variables into characters
  step_mutate_at(all_predictors(),
                 -loan_amount,
                 -income,
                 -combined_loan_to_value_ratio,
                 -property_value,
                 fn = function(x) as.factor(x)) %>%
  # normalize all numeric predictors
  step_normalize(all_numeric_predictors()) %>%
  # remove zero variance variables
  step_zv(all_predictors()) %>%
  # remove near zero variance variables (for LDA to sorta work better)
  step_nzv(all_predictors()) %>%
  # dummy encoding of nominal variables
```

```
  step_dummy(all_predictors(),
             -loan_amount,
             -income,
             -combined_loan_to_value_ratio,
             -property_value)

# Boosted Tree -- Optimally tuned at: learn_rate = 0.0010)
set.seed(10)

# Creating workflow
wf_bt <- workflow() %>%
  add_model(boost_tree_model) %>%
  add_recipe(rec)

# Fitting model to full training data
final_model_fit <- wf_bt %>% fit(data = train)

# Make predictions on test data
preds <- final_model_fit %>% predict(new_data = test)

# Reading in test data again to reinclude original ID's
test <- read.csv("test2.csv")

# Prediction Results
test_preds <- test %>%
  dplyr::select(id) %>%
  bind_cols(preds) %>%
  rename(action_taken = .pred_class)

# CSV of results
write_csv(test_preds, "boostedpred1.csv")
```

6.2 Team Member Contributions

- *Mayerli Cordero-Cortes*: Data Evaluation, Model Construction
- *Eric Gallardo*: Model Construction, Model Tuning, Data Visualization, Research Write-Up, and Recipe Creation
- *Avanthika Panchapakesan*: Data Evaluation, Data Visualization, Background Research, Research Write-Up
- *Shayan Saadat*: Data Evaluation, Data Visualization, Research Write-Up, Final Drafting of Report
- *Luke Villanueva*: Data Evaluation, Data Visualization, Recipe Creation and Model Construction, Research Write-Up

6.3 Note on Official Script Submission

The submission for this report is based on the submitted results of boostedpred1.csv in the competition. That was the submission that was selected for in the competition, but another model was autoselected by kaggle, but those results were not reproducible, so boostedtree1.csv is our choice for the competition submission and our report.

References

Cooper, Daniel. 2013. "House Price Fluctuations: The Role of Housing Wealth as Borrowing Collateral." *The Review of Economics and Statistics* 95 (4): 1183–97. https://EconPapers.repec.org/RePEc:tpr:restat:v:95:y:2013:i:4:p:1183-1197.

Munnell, Alicia H., Geoffrey M. B. Tootell, Lynn E. Browne, and James McEneaney. 1996. "Mortgage Lending in Boston: Interpreting HMDA Data." *The American Economic Review* 86 (1): 25–53. http://www.jstor.org/stable/2118254.