UCF CFE DATA MINING COMPETITION FINAL REPORT

Team Members: Marc Mailloux & Larry Moralez

Professor: Monsur Chowdhury

Introduction

Our team will be building a statistical model to compete in the CFE 2017 lending analytics competition. This competition provides real, anonymized data from car loan applications. This data is intended to be used to predict future lending decisions (i.e. approve or disapprove). In order to make the most accurate predictions for future decisions, our team will apply data mining techniques learned throughout the course and develop a statistical model that will predict the outcomes of loan lending decisions.

We began the data mining process by doing an initial exploration of the data. We then carried out an initial phase of preprocessing that removed parameters to make the model more interpretable and less computationally expensive to run. We also split the data into training and test observations at an 80/20 split, respectively. We then ran an exhaustive best subset selection. We followed exhaustive subset selection with with a round of validation tests on the testing data for each of the 27 models used. From here we obtained the best performing model for the test data and went on to run predictions using LDA, QDA, Random Forest, AdaBoost, and Support Vector Machine to see which model performed the best.

Data Exploration

In order to clean and prepare the data, we first need to explore the data. The dataset consists of 36 unique parameters. Examples of these parameters include credit score, employment status, monthly income, and vehicle make. These parameters are either numeric (i.e. credit score) or categorical (employment status). The independent variable to be predicted for the competition is Loan Status. This variable describes whether or not the loan application was approved.

The entire dataset contains 109,854 observations, and was divided into a training set and a test set. The training set consists of 87,883 observations and the test set consists of 17,599 observations, for an 80/20 split.

Having applied for loans in the past, we knew how important credit score was to determining the lending decision. Therefore, we plotted credit score with respect to the amount requested by the customer. These results can be seen in Figure 1. It is obvious of the graph that there are indeed outliers for these variables. We would like to mention that we adjusted the y axis so all of the data is not visible in the graph above, yet the majority of it is in the graph. From the top left graph in Figure 1, we can see that more of the customers that have been denied for the loan tend to ask for more money. In the graph below that the threshold for getting approved in terms of credit score is around 600, which tells us if you don't have a credit score of 600 or higher you have a high chance of getting denied. There is some overlap between getting approved and denied for the credit score but it seems the average credit score for those who get denied is around 600 where those who get approved is

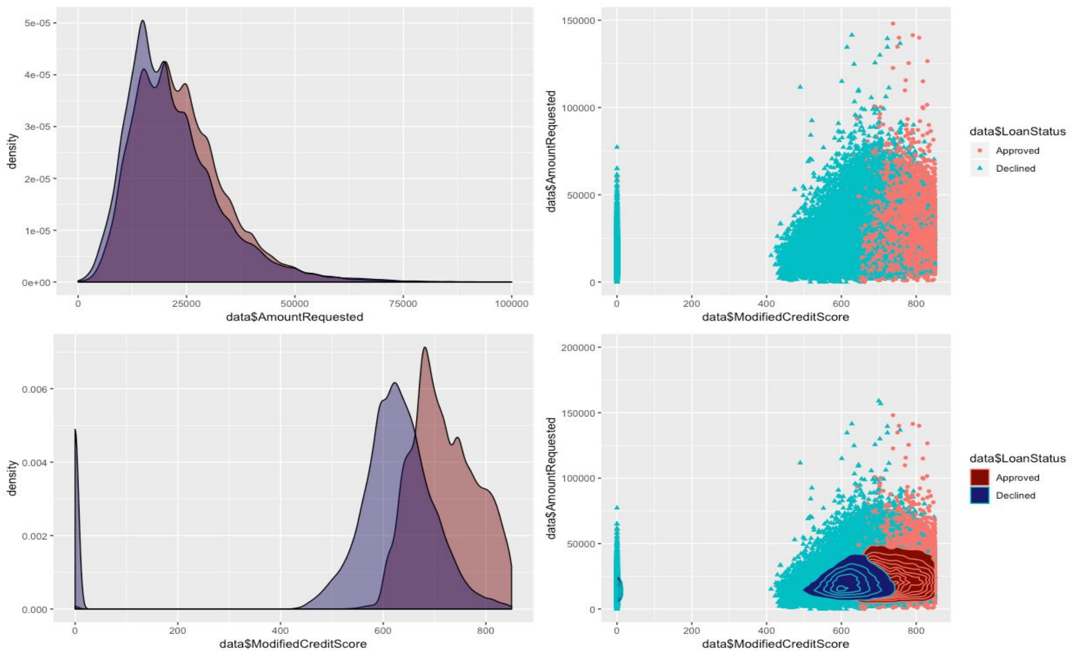


Figure 1: Credit Score vs Amount Request Comparison and Density graphs

closer to 700.

Data Preprocessing

During the the data exploration phase we noticed that some parameters were showing up as factors, some as integers, and some as numerical values. When we attempted to fit a general logistic regression model to the data we noticed that it was taking an extremely long time, in upwards of 8 hours. R would frequently crash and we were never able to obtain results. As we explored some possible reasons for this we noticed that the function treats each categorical variable from a parameter as its own parameter. So if a variable had 100 different categories-or factors, as they are referred to in R- then the logistic regression equation would not simply be modeling 36 parameters but instead 136. We found this to be the key reason why model fitting times were so high. However, one parameter, LTV, has over 1,000 unique categories. This meant that our model was trying to fit well over 1,000 different parameters! Some preprocessing techniques must therefore be applied to generate dummy data for the categorical variables prior to fitting the model. However, we were unsuccessful in our attempt to convert the data to a type that could be used in the model such that it would avoid this problem. Our answer was to remove the parameters that were comprised of multiple factors, with "Source" being the only parameter with factors that we included because it was limited to only 2 factors. The parameters removed include OccupancyStatus, RequestType, CoMonthlyRent, CoMonthlyLiability, LTV, VehicleMake , and AppReceiveDate (because date of application should not have any true correlation with whether or not a loan was approved or not). This helped us to both improve interpretability while at the same time ease computational demands.

After an initial cycle of running subset selection and running the best set of parameters through several different predictive models, we went back and removed outliers before running another cycle and acquiring our final results. We tested each non-factor parameter for outliers using a simple qualitative analysis using boxplots. We found that virtually all parameters contained some outliers. We developed a function to remove these observations and tested the again using the boxplot method. We were successful in removing outliers for several variables. However, some variables removed far too many observations for us to be comfortable with altering the data. For example, our outlier removal function removed over 5,000 observations from ModifiedCreditScore parameter. Given that this parameter is the more important in predicting LoanStatus, we chose not to use outlier removal on it for it would likely lead to much poorer performance on predictions that have we left the outliers in. In all, we only removed outliers from a parameter if the total number of outliers removed was ~1,000 observations, which was roughly 1% of the total data. The parameters that we did not remove outliers from include ModifiedCreditScore, PrevEmployedMonths (4654 observations removed), CoEmployedMonths (5269 observations removed), CoPrevEmployedMonths (>5,000 observations), CototalMonthlyIncome (3537 observations removed), DownPayment (3066 observations removed), and TotalMonthlyIncome (1772 observations removed).

We then split the data into training and testing sets. The training to testing split that we chose was 80/20, respectively. This is a standard convention and since we had over 100,000 observations we chose not to toy with this too much.

Subset Selection

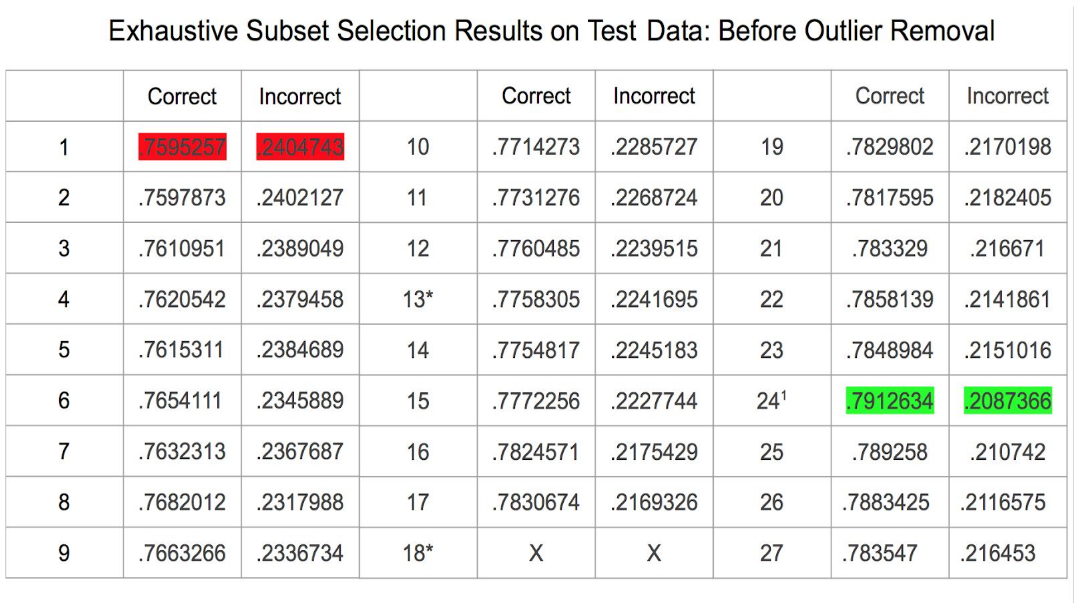
When it came time to run subset selection, we encountered extremely long wait times before R would ultimately crash. This is when we notices that the function was treating each category for a parameter as a factor that would be included as a parameter in the model selection. If we were to have done exhaustive subset selection with this many factors being treated as parameters, we would have had $2^{>1,500}$ different models that would need to be tested and compared. Given our computational limitations, this likely would have taken until the heat death of the universe. Even forward and backwards selection was too costly. This motivated our decision to simply remove the variables that we discussed in the previous section.

With the aforementioned parameters removed, we were able to run exhaustive subset selection on the training data with ease. This gave us the best model for each number of variables, 1 through 27. To gauge which of these models was best, we calculated the predictions accuracy of the model on the testing data. Because we had a testing data set of close to 18,000 observations, we chose not to use cross validation. Furthermore, since the predictor models would be tested on the testing data as well, we felt it was best to select the model that performed best on this data rather than the training data. If we had tested on the training data, with over

80,000 observations, it is possible that our accuracy would have been too high due to overfitting and would have performed poorly on the test data.

One variable, “Source,” was comprised of two factors (Gateway and Lender). Since it was only two factors, we included it in our subset selection. As mentioned previously in the variable date preprocessing, each factor was treated as variable by the function and are therefore included in the subset selection. Rather than remove each of the two factors individual, we removed the full variable the first time it appears (we removed the variables because we worked backwards from the largest model to the smallest, Figure 3). Therefore, we removed the “Source” variable at the 18 variable model when the first of the two factors is removed from the. This means there was nothing to remove when we got to the second “Source” factor in the model at variable 12. (When testing the scores prior to removing outliers, figure 2, we proceed by adding variables to the smallest model. This is opposite of how we proceeded with after outlier remove, hence why the Xs are in different spots in Figure 2 and Figure 3.)

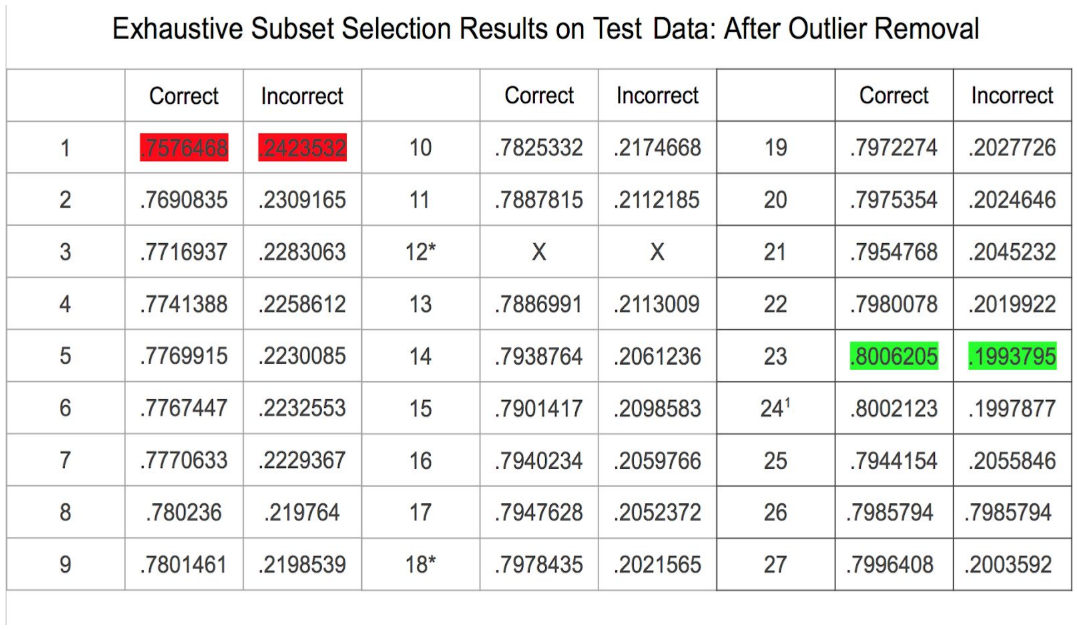
We ran the subset selection and calculated the performance two times: once on the training data prior to outlier removal and once after outlier removal. The results of the subset selection prior to data removal can be seen in Figure 2. For this run, the model with 24 predictors performed the best with a classification accuracy of 79%. The results of the subset selection after outlier removal were much better (Figure 3). The best performing model was the one with 23 predictors and had a classification accuracy of 80%. For both instances of subset selection, the worst performing model was the single predictor model comprised of ModifiedCreditScore. Despite being the worst performing model, it was still able to generate a classification accuracy of ~76% for both instances.



Exhaustive Subset Selection Results on Test Data: Before Outlier Removal

	Correct	Incorrect		Correct	Incorrect		Correct	Incorrect
1	.7595257	.2404743	10	.7714273	.2285727	19	.7829802	.2170198
2	.7597873	.2402127	11	.7731276	.2268724	20	.7817595	.2182405
3	.7610951	.2389049	12	.7760485	.2239515	21	.783329	.216671
4	.7620542	.2379458	13*	.7758305	.2241695	22	.7858139	.2141861
5	.7615311	.2384689	14	.7754817	.2245183	23	.7848984	.2151016
6	.7654111	.2345889	15	.7772256	.2227744	24[1]	.7912634	.2087366
7	.7632313	.2367687	16	.7824571	.2175429	25	.789258	.210742
8	.7682012	.2317988	17	.7830674	.2169326	26	.7883425	.2116575
9	.7663266	.2336734	18*	X	X	27	.783547	.216453

Figure 2: The results of exhaustive subset selection prior to the removal of outliers. The best performing model is highlighted in green with the worst performing model highlighted in red.



Exhaustive Subset Selection Results on Test Data: After Outlier Removal

	Correct	Incorrect		Correct	Incorrect		Correct	Incorrect
1	.7576468	.2423532	10	.7825332	.2174668	19	.7972274	.2027726
2	.7690835	.2309165	11	.7887815	.2112185	20	.7975354	.2024646
3	.7716937	.2283063	12*	X	X	21	.7954768	.2045232
4	.7741388	.2258612	13	.7886991	.2113009	22	.7980078	.2019922
5	.7769915	.2230085	14	.7938764	.2061236	23	.8006205	.1993795
6	.7767447	.2232553	15	.7901417	.2098583	24[1]	.8002123	.1997877
7	.7770633	.2229367	16	.7940234	.2059766	25	.7944154	.2055846
8	.780236	.219764	17	.7947628	.2052372	26	.7985794	.7985794
9	.7801461	.2198539	18*	.7978435	.2021565	27	.7996408	.2003592

Figure 3: The results of exhaustive subset selection after the removal of outliers. The best performing model is highlighted in green with the worst performing model highlighted in red.

Prediction

For the purpose of analysis we test 3 predictor models against 6 fitting procedure. The predictors models consist of the 23(best) and 24(second best) variable models. The 6 fitting models we used were Linear Discrete Analysis, Quadratic Discrete Analysis, Random Forest, SVM with Radial Kernel, Svm with Linear Kernel, and AdaBoost. All methods were implemented using the Caret Package in R, where repeated cross validation was used to find the best models for each learning model. All models were first fitted on the training data to find the training classification error rate. This can be seen in Figure 3. Once trained, we used the fitted values on the testing data that we split earlier to find the Mean Classification Error Rate. To show these results we plotted the ROC curve on the testing data in Figure 4. From the data seen in Figure 3 we see that AdaBoost, RandomForest, and SVM1 performed extremely well on the training data. Additionally we have the metric Kappa which takes the class imbalance into consideration given a more accurate representation of our learning models. The kappa values fall almost by 10 percent and has a wider interval for each of the learning methods compared to the accuracies. .

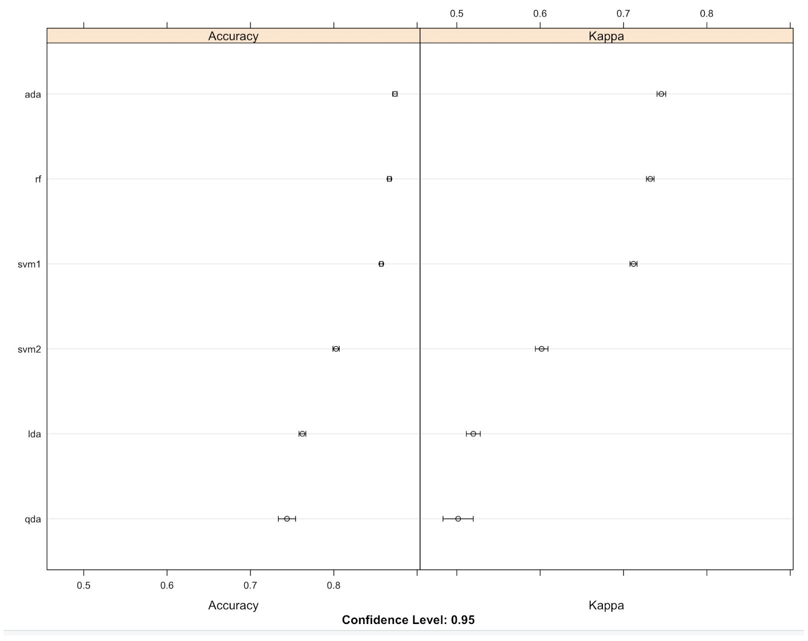


Figure 3: Fitted training Classification Accuracies Comparison

Looking at Figure 4 we see that AdaBoost and Random Forest achieved the best results as expected from our training predictors. R gave us an error that did not allow us to plot the

SVM1 and SVM2 on the ROC graph. A more indepth look at our learning models in Figure 5 we see that through cross validation the number of predictors chosen is indeed 23 and AdaBoost chooses the maximum number of trees as well. Figure 5 also verify our subset selection methods where the best predictors were the 23 we have chosen variables.

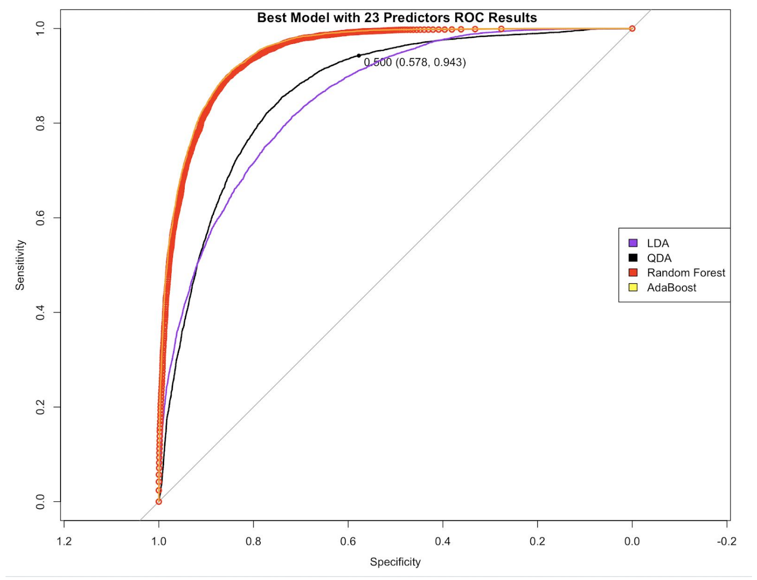


Figure 4: Best Model Result

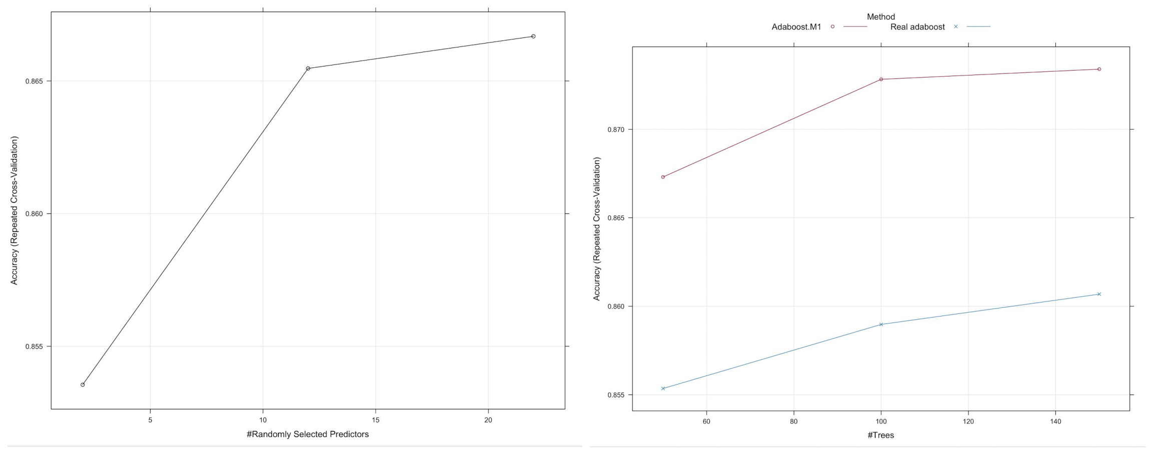
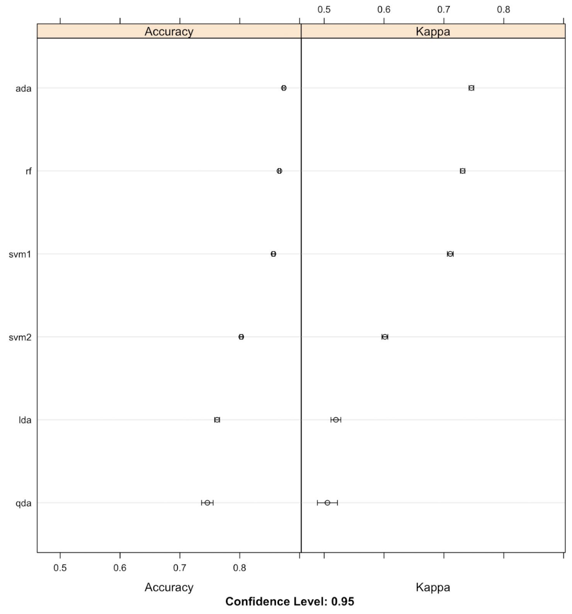


Figure 5: A close look at our results

As previously stated we used also wanted to analysis our second best model which was has 24 predictors. These results can be seen in Figure 6 and 7 showing the training classification error and the test classification ROC. The 24 predictor model actually perform marginally better than the 23 predictor model. The best training classification for the 23 predictor model was

.8734, and the 24 predictor model .8737 where both training methods were AdaBoost.



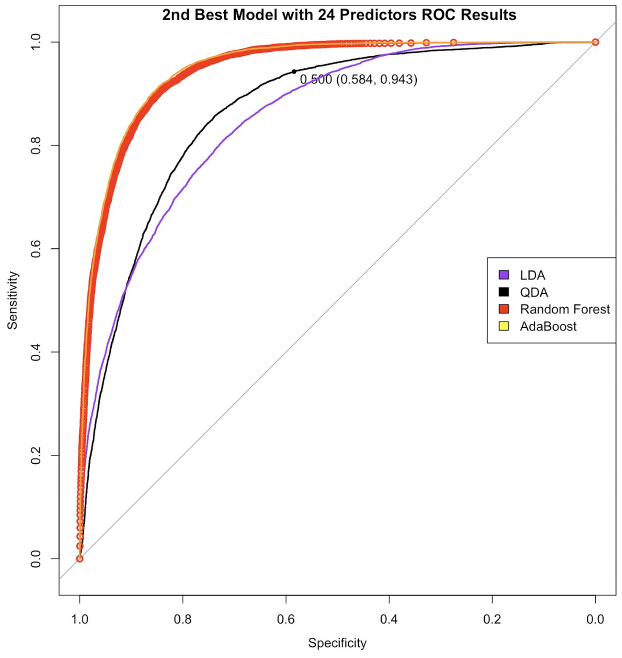


Figure 6

Figure 7

Comparing the test mean classification rate the 23 predictor model achieved a 87.72% accuracy, while the 24 predictor model achieved 87.76%. From these results we would suggest not using the 24 predictor model as the rise in accuracy is extremely minimal. Adding the 24th predictor is not suggested since there is no direct benefit to be gained. This can also be seen by comparing Figures 4 and 7 as there is no significant difference in their responses. Lastly we also expect LDA and QDA to do worse compared to the other models since they explicitly state their decision boundaries. AdaBoost, Random Forest and SVM proved to be the more flexible methods relative to our dataset, where LDA and QDA were too inflexible. The main benefit to the other methods is cross validation to find the optimal hyper parameters, which can be different for each dataset unlike LDA or QDA the decision boundary will also be linear or quadratic.

Table 1: Classification Results

Test Classification Accuracies	23 Predictor Model	24 Predictor Model
LDA	76.28 %	76.29 %
QDA	74.28 %	74.67 %

Random Forest	87.26 %	87.25 %
SVM - Radial	86.13 %	85.96 %
SVM - Linear	80.56 %	80.60 %
AdaBoost	87.72 %	87.76 %

Conclusion

Our methodology allowed us to achieve and accuracy score of 87.76 using the 24 predictor model. Prior to achieving these results we went through through a lengthy process of data preparation and model selection. We began by removing certain predictors to increase interpretability and to save on computation time. We then removed the outliers from some of the predictors, although we left the outliers in for other predictors (i.e. ModifiedCreditScore) in the event that the outlier removal function removed too many observations. In following, we split the data in training and test sets, the former being used to train the model and to run exhaustive subset selection on, the latter to test the results of the subset selection and perform predictions using 6 different models. Once subset selection was complete, we tested each of the 27 models on the testing data to find the top two scoring sets of predictors. The best model had 23 predictors, and the second best had 24. From there we went ahead with the prediction using LDA, QDA, Random Forest, SVM with radial Kernel, SVM with Linear Kernel, and ADA Boost. We found that the best model combined AdaBoost method with the 24 set of predictors. We also achieved similar results with AdaBoost and the set of 23 predictors (87.72% vs 87.76%). We can conclude that for this specific data set and prediction scenario, AdaBoost is most likely to generate the high classification accuracy.