# Homework #4: Insurance Claim Prediction

Data 621 Business Analytics and Data Mining

*Aadi Kalloo, Nathan Lim, Asher Meyers, Daniel Smilowitz, Logan Thomson*

*Due July 10, 2016*

## Contents

# Final Project: Loan Approval

Data 621 Business Analytics and Data Mining

*Aadi Kalloo, Nathan Lim, Asher Meyers, Daniel Smilowitz, Logan Thomson*

*Due July 21, 2016*

## Contents



### Abstract

Regression and classification techniques are used to predict loan approval status of customers of a fictional bank given fourteen predictors. Data was obtained from a "data challenge", and the given variables were supplemented with transformations. Models were trained on a dataset of 614 cases, and subsequently used to predict the binary outcome of 367 test cases. Techniques utilized include logistic regression, ridge regression, the Adaboost algorithm, and a neural network. Analyses were performed to determine the optimal number of iterations and optimal number of nodes for the Adaboost algorithm and neural network, respectively. Cross validation results were not linearly co-variant with accuracy scores on the test data set. Using tools supplied by the data challenge, accuracy scores were used to compare models and the Adaboost algorithm was found to produce the best results. The optimal model created yielded an accuracy of 0.8056.

**Keywords: loan prediction, classification, logistic regression, adaboost**

## Introduction

Between 2009-2014, approximately 95 million mortgage loan applications were filed in the United States alone (Home Mortgage Disclosure Act, 2015). When further loan types, including education and personal loans, are considered, it becomes even clearer that the ability to accurately identify reliable borrowers is a high priority for banks and other financial institutions today. Likewise, consumers are significantly affected as well. Surveys and analyses performed by Lusardi & Scheresberg (2013) has shown that individuals of poor financial literacy, and subsequently poor credit history, are declined by banks for loans and mortgages, and turn to high-cost borrowing (i.e. payday loans). Thus, it becomes clear that the factors that determine which individuals are dependable borrowers is information important to both financial institutions and consumers alike.

The dataset of interest contains information about customers of a hypothetical bank and was obtained from a data challenge hosted at AnalyticsVidhya. This data set and challenge were chosen due to its relevance to the current loan-centric economy outlined above, and to address the need for more transparent analyses of optimal methods for financially-related prediction models.

## Literature Review

Although predictors can vary by loan or income level, studies investigating loan approval predictors have found that factors such as Marital Status, Loan Duration, and Number of Dependents can serve as statistically significant predictors of loan approval status (Agbemava et al., 2016). However, as loan approval criteria can vary greatly between financial institutions there is a significant lack of published materials investigating the effects that various factors may play on loan approval status. Furthermore, the majority of financial institutions use proprietary algorithms and statistical methods in order to determine loan approvals.

It should be noted that ratio of training set cases to test set cases is 1.67:1. This naturally leads to markedly decreased algorithm training than the 4:1 ratio that is normally used in prediction tasks (Dobin & Simon, 2011).

Of significant interest to us were the concepts of additive logistic regression or "boosting" as outlined by Friedman et al. (2012). The original concept of boosting was introduced by Schapire (1990) and describes a method of increasing the performance of "weak" classifiers by combining many of them into a more powerful model (Friedman et al., 2012). The authors of this paper introduce the Adaboost algorithm, which is used and analyzed in detail for this project. The Adaboost algorithm is characterized by the use of weights on the target variable, increasing the weight on each iteration for misclassified values (Friedman et al., 2012).

Given that this project was centered around a data challenge, it was also of particular significance to us to research data mining and machine learning methods that could potentially provide greater accuracy scores, despite being outside the scope of IS621 Data Mining. The primary method of prediction in this arena is the neural network. While the foundational ideas of artificial neural network date as far back as the 1940s (Warren & Pitts, 1943), the procedures used by the `neuralnet` package are based on Riedmiller and Braun (1993). As a brief overview, the `neuralnet` package calculates the following function as described by Gunther & Fritsch (2010):

$$o(x) = f(w_0 + \sum_{j=1}^{J} w_j \cdot f(w_{0j} + \sum_{i=1}^{n} w_{ij} x_i))$$

where $w_0$ is the intercept of the output neuron, $w_{0j}$ is the intercept of the $j^{th}$ hidden neuron, and $w_j$ is the synaptic weight.This mathematical model can be expressed visually, and is depicted in the following figure:

Given the fact that loan prediction has traditionally been a field of interest for those who can benefit from financial gain (i.e. big banks, as opposed to academia), research and methodological studies in this area is sparse. Most of the current literature in this area has been focused on financial solvency issues, namely bankruptcy. It has been found previously that logistic regression, neural networks, and classification trees are the three most commonly used data mining methods in the prediction of bankruptcy (Olson et al., 2012). As a means of extending this work to the arena of personal loans, this project serves to compare some of the most commonly used methods of prediction in loan outcomes. We believe that the information and methods investigated here can trickle down to a

smaller scale and be beneficial to smaller economies such as microlending, where loans tend to be oriented around civilian populations instead of financial institutions (Conlin, 1999).

## Methodology

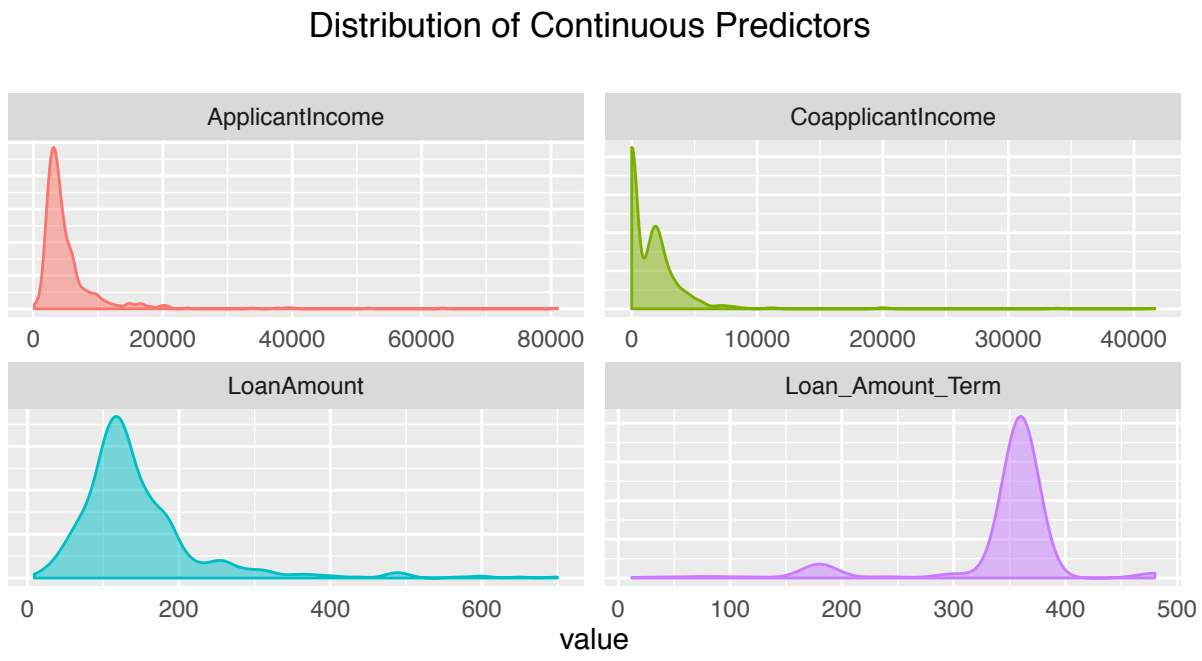
The dataset has 614 rows (each representing a customer) and 11 predictor variables. Furthermore, there is 1 identification variable, and 1 response variable: `Loan_Status`, a binary categorical variable representing whether each customer has been approved for a loan given their credentials. The goal of the data challenge and of this investigation is to predict the approval of loan applications based on the predictors.

### Data Exploration

The classes of the predictor and response variables are presented below:

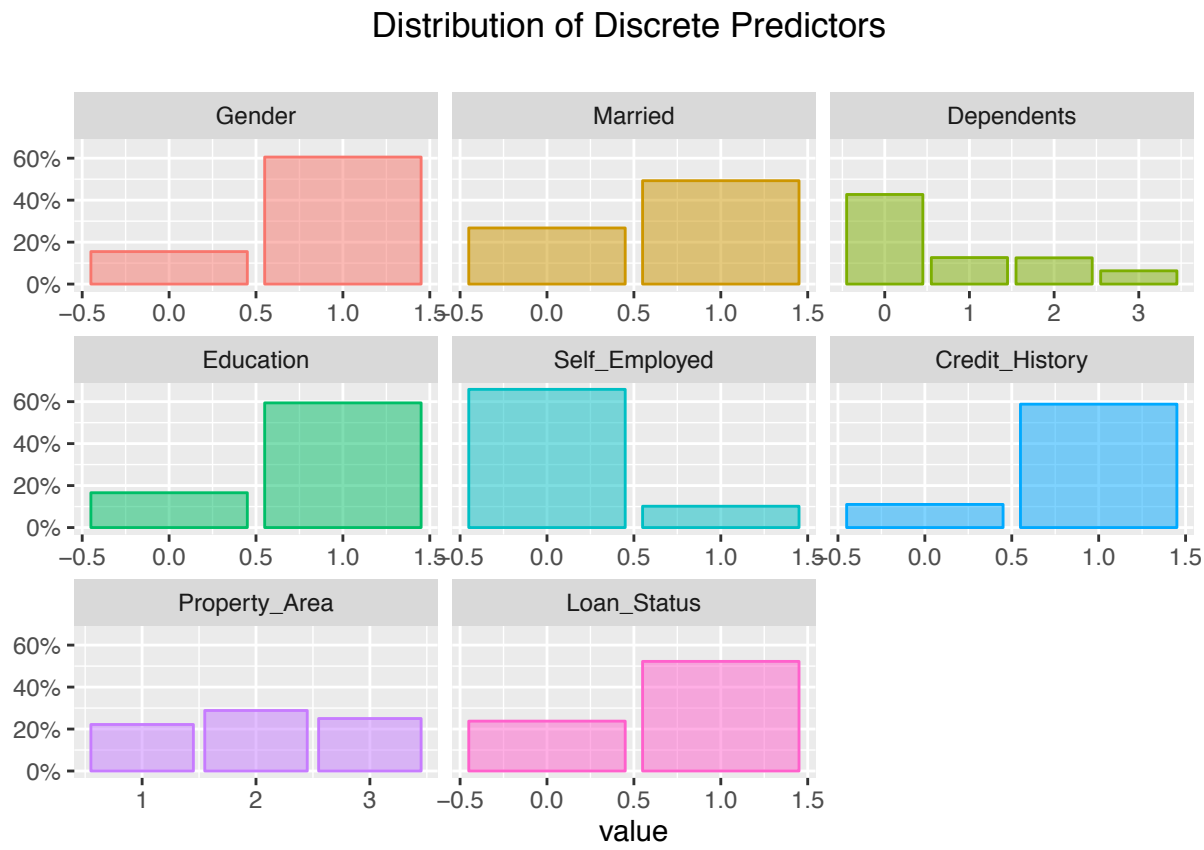
| | Class | Levels/Range |
|---|---|---|
| **Gender** | numeric | 0 - 1 |
| **Married** | numeric | 0 - 3 |
| **Dependents** | numeric | 0 - 1 |
| **Education** | numeric | 0 - 1 |
| **Self_Employed** | integer | 150 - 81000 |
| **ApplicantIncome** | numeric | 0 - 41667 |
| **CoapplicantIncome** | integer | 9 - 700 |
| **LoanAmount** | integer | 12 - 480 |
| **Loan_Amount_Term** | integer | 0 - 1 |
| **Credit_History** | factor | 3 |
| **Property_Area** | numeric | 0 - 1 |
| **Loan_Status** | numeric | 0 - 1 |

It can be seen that four of the predictors are continuous variables; the remaining 7 predictors, as well as the response, are categorical variables, with all but one of these variables (`Dependents`) being binary. The distributions of the continuous predictors are presented below:



It is clear that each of the continuous variables exhibits a strong level of skewness – `ApplicantIncome`, `CoapplicantIncome`, and `LoanAmount` are right-skewed, while `Loan_Amount_Term` is left-skewed, It can be seen that the peak in the distribution of `Loan_Amount_Term` is located at 360 – this corresponds to 30 years, the typical length of a mortgage.

The distributions of the discrete predictors, as well as the target, are presented below:




The last figure in the chart shows that roughly 75% of loan applicants are approved for a loan.

With the exception of `Property_Area`, each of the discrete predictors exhibits a dominant category. The distributions indicate that, with all other variables held constant, typical applicants:

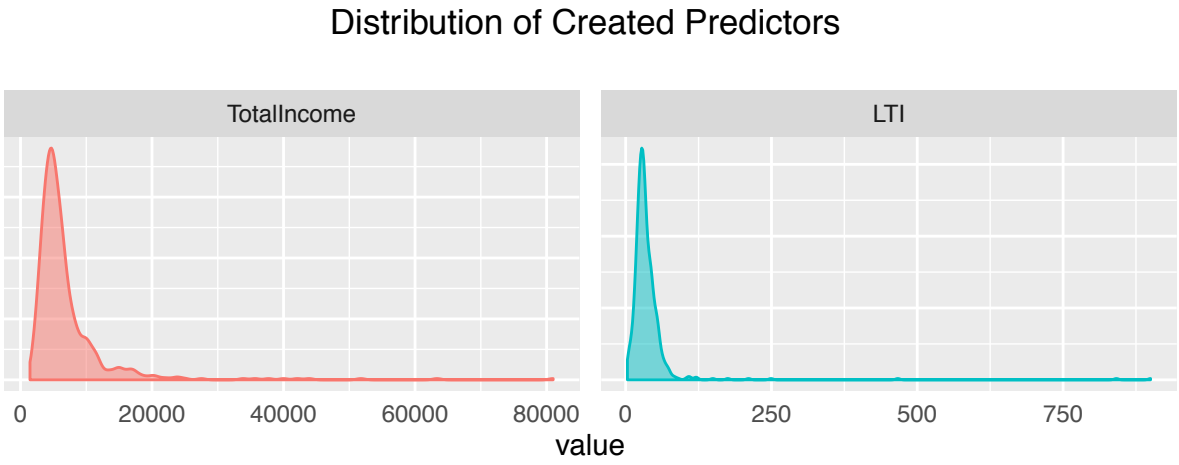
- Are male
- Are married
- Have no children
- Are graduates
- Are not self-employed
- Meet the firm's credit requirements

Inspection of the raw data set reveals that there are a number of missing values; these are addressed prior to the development of predictive models.

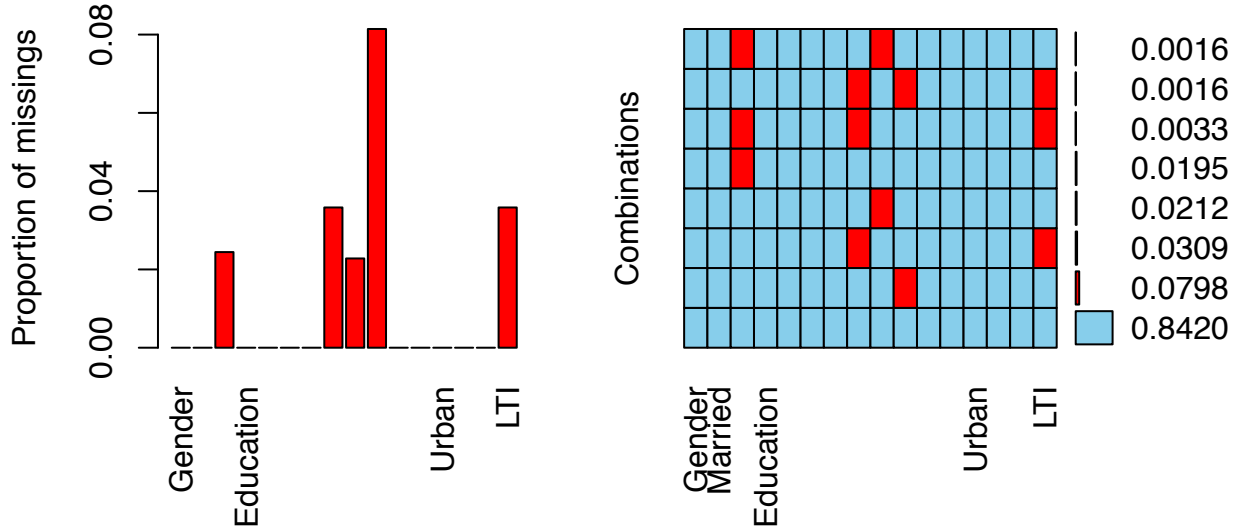
### Data Manipulation

The supplied training data set originally has 11 predictor variables, six of which are character vectors which needed to be converted to binary categorical variables. The response variable, stated as a simple "Y/N" was also converted into the 0 or 1 format for modelling. As it was deemed not significant and also not chosen by any automated model, `Rural` is excluded from the `Property_Area` predictor, leaving only `Urban` and `SemiUrban`, which are turned into separate binary categorical predictors. `Property_Area` is then excluded from the training dataset.

All other variables in the data set were numeric, or already in a binary format. Given the included variables, two new features were created. `TotalIncome` is the sum of applicant and co-applicant incomes, and `LTI` (Loan to Income Ratio) is the loan amount ($\times$ 1000) divided by the `ApplicantIncome` variable. The distributions of thse created variables are shown below; they exhibit a similar degree of right-skewness as the three continuous predictors used to create them.




Missing values were also present in the data, either in the form of empty characters (`""`), or `NA`s. Four of the predictors, `Gender`, `Married`, `Dependents`, and `Self_Employed` contained empty characters. Being only a small proportion of the cases ($<$ 1% to 8%), these were in a way imputed when the predictors were converted into binary categorical variables. Four other variables, `LoanAmount`, `Loan_Amount_Term`, `Credit_History`, and `LTI` all contained small numbers of `NA`s. These are either dropped from the dataset, or imputed with the median, depending on the model. Alternative imputation methods, such as filling with a weighted "coin toss" value yielded less accurate results in the models. A representation of the proportion of missing values is presented below:



**Model Creation**

In total, six models were created and used for prediction of the target variable. A tool was supplied by AnalyticsVidhya that reported accuracy (% of cases correctly predicted) upon submission of the test data predictions.

Logistic regression is a very common method of classifying data (Thomas, 2000). In logistic regression, a linear regression is used where the dependent variable is a non-linear function of the probability of the event occurring. In this project, the purpose of the logistic regression method is to classify cases as either "Approved for loan" or "Denied". The general logistic regression model can be represented as:

$$\ln \frac{p_i}{1-p_i} = \beta_0 + \beta_1 X_1 + ... + \beta_n X_n + \varepsilon$$

A few different logistic regression models were developed for this project. The first was a full model using all variables. This model made use of the `glm` function. This model was created as a basic point of comparison. This model produced an AIC of 935.

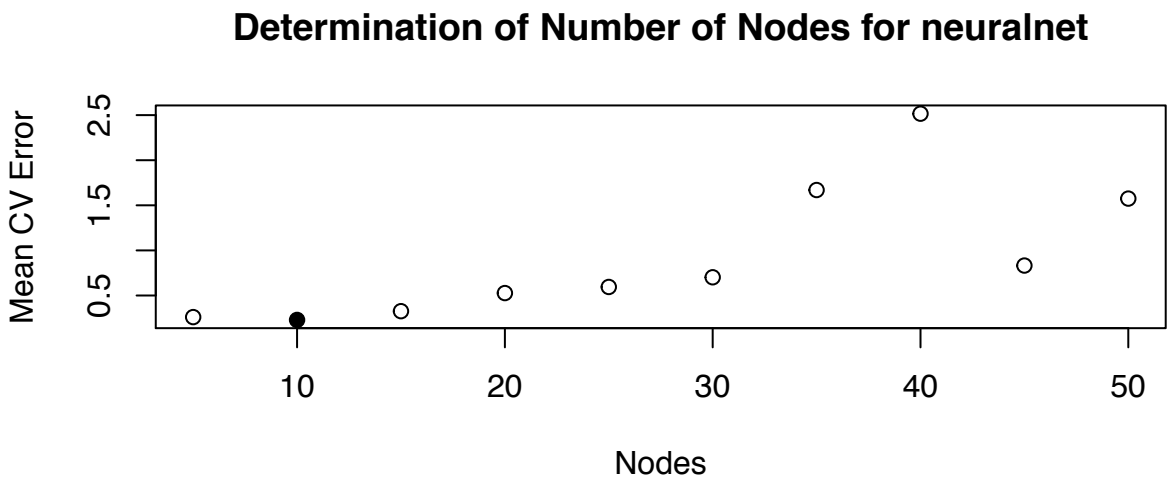
The second model created utilized both the `glm` function and the `leaps` package in order to find the best combination/subset of variables that minimizes the Bayesian Information Criterion (BIC). The "best subsets" approach is a method of automated variable selection that seeks to find the best predictors of the target variable. The `regsubsets` function as part of the `leaps` package found the `Married`, `Credit_History` and `SemiUrban` variables to be the best subset of predictors for this data set. The AIC reported for this model was 518.

Ridge regression methods were combined with logistic regression to create a "Logistic Ridge" model, the third of this project, using the `ridge` library. As this library is no longer actively in development, it was manually obtained and installed. Like Ridge Regression, Logistic Ridge Regression introduces a tuning parameter as a penalty to avoid overfitting (Cule & De Iorio, 2012). This is useful as the ridge regression method aims to reduce variance and produce lower mean squared error for higher-dimensional data such as that used in the full model for these data. The ridge penalty used in the `ridge` package as outlined by Cule & De Iorio (2012) is given as:

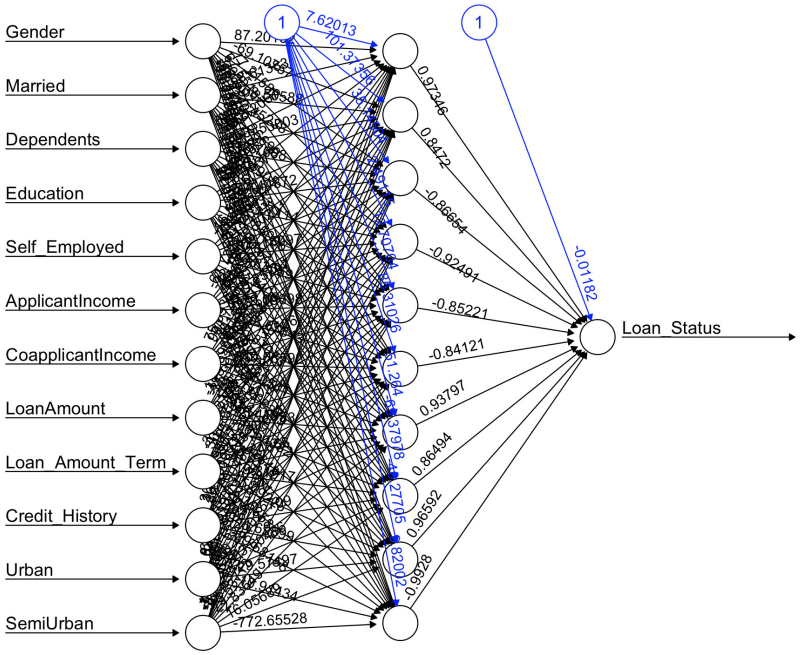
$$k = \frac{p}{\hat{\beta}'\hat{\beta}}$$

The Adaboost algorithm was used for this project in two forms. The first was through the use of the `ada` package. For this second implementation, the effect of the number of iterations of the algorithm on the final accuracy of the test set was analyzed with iterations in the range of 50-525 with steps of 25. It is hypothesized that accuracy will gradually increase with iterations to a peak, then gradually decrease as overfitting starts to become a prominent issue. The second was a manually coded implementation based on the algorithmic outline provided by Friedman et al. (2000). A manual approach was used for the purposes of comparing the academically published to the packaged algorithm. Given that these algorithms work by combining basic models, it is believed that they will perform better than a single logistic regression model alone.

A simple neural network method was pursued. The neural network approach was performed using normalized data with a single hidden layer. The number of nodes to use in the hidden layer was determined using a 10-fold Cross Validation Approach. A plot of the mean CV error against the number of hidden nodes is shown below:




From the above data, a network of 10 nodes was chosen due to its minimal cross validation error. This resulted in a network where the input variables were each sent to the 10 hidden nodes where weights were applied based on the variable's signifiance, and then sent to an output node that used its own weight to output the target variable classification. The illustration of the network used for prediction of the target variable is included here:



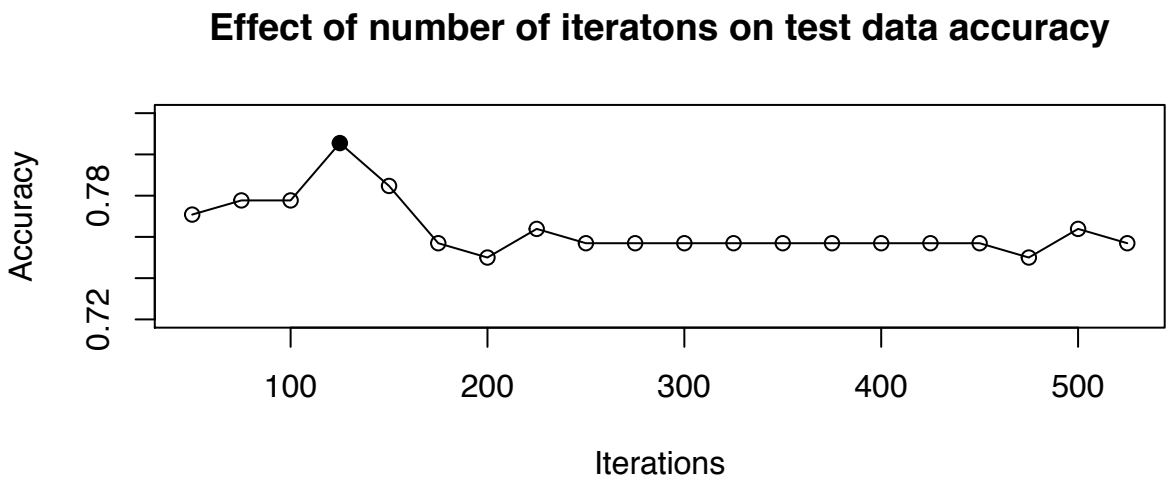

## Results

### Accuracy

The results obtained are summarized as follows:

| | Model | Accuracy |
|---|---|---|
| 1 | ada package | 0.8055 |
| 2 | Best Subsets | 0.7778 |
| 3 | Logistic Ridge | 0.7708 |
| 5 | Full Model, Poisson | 0.7708 |
| 4 | Full Model, Binomial | 0.7638 |
| 6 | Neural Network (neuralnet package) | 0.7431 |
| 7 | Manually implemented adaboost algorithm | 0.7153 |

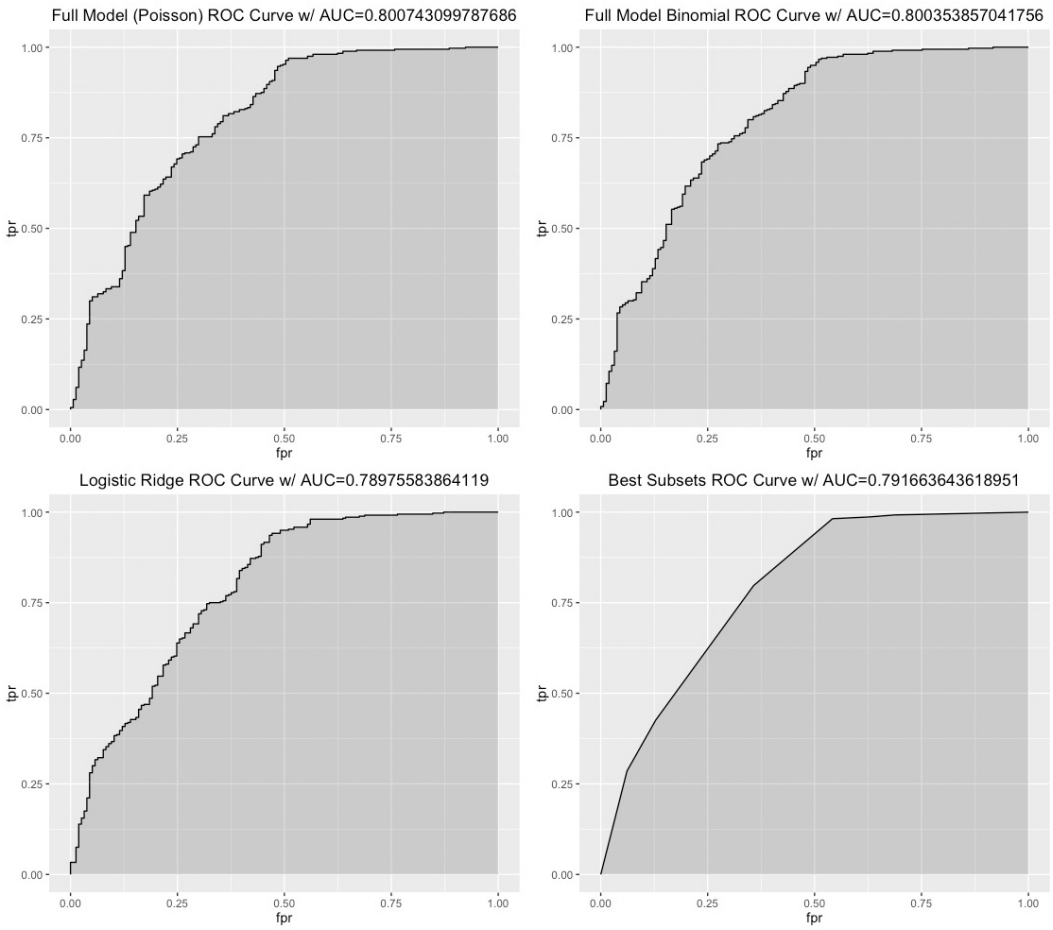
It can be seen that the best results were achieved through use of the `ada` package. The following figure illustrates the results of an analysis of test data accuracy against iterations of the `ada` function:




Optimal accuracy was attained at 125 iterations, with a steady decline and plateau thereafter. These results were in keeping with our hypothesis, however it was expected that the peak accuracy would occur at a higher iteration.

**ROC Curves**

ROC curves for the linear regression models were created and are displayed here.



The ROC curves shown here illustrate that the performance of the four binary classifiers represented do not vary greatly from one another – all classifiers show an Area Under the Curve (AUC) value of approximately 0.8. However, as shown in the table above, the accuracy of these classifiers vary greatly on the test data with a maximum and minimum of the four listed here being 0.7778 and 0.7638, respectively. While this may seem to be a difference of "only" 0.014 on test data accuracy, this translates to a differences of approximately 80 places in the data challenge rankings. This is a potent example of the importance of seemingly miniscule improvements in prediction models.

### 10-Fold Cross-Validation

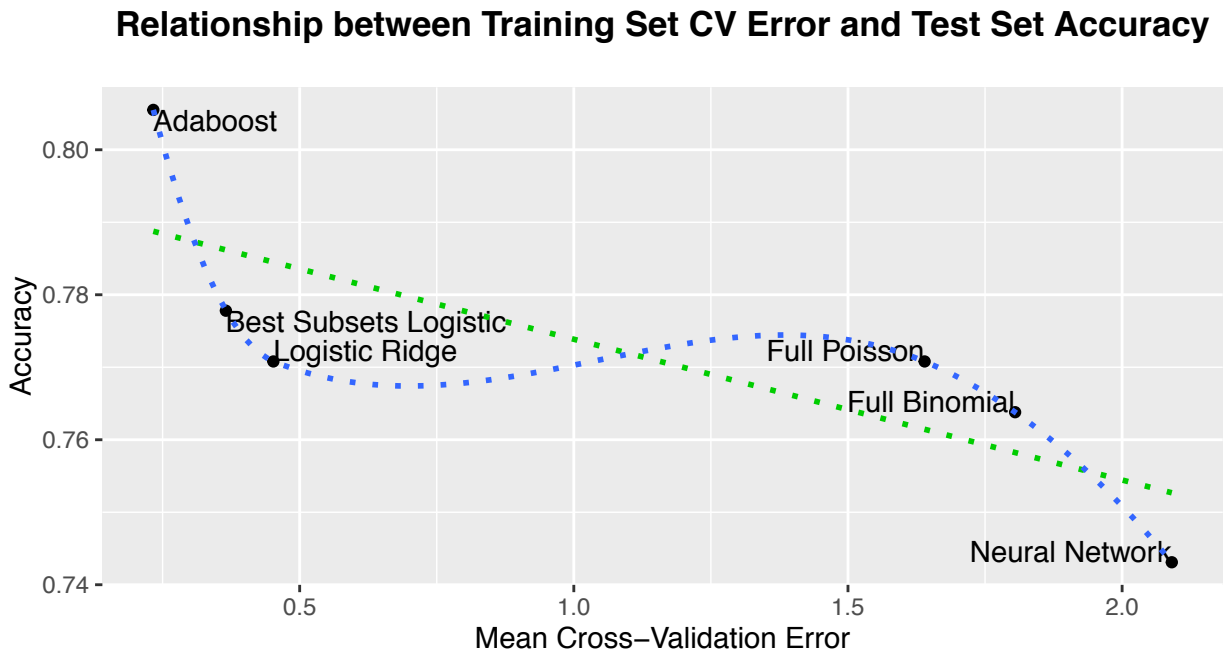
The 10-Fold Cross-Validation approach was used again to determine which model produced the lowest error on training data. This method was chosen as it allows all observations to be used for both training and validation, while also allowing each model to be tested on unseen data. These results are compared in the Discussion section below to the accuracy on test set data. A summary of the mean CV error for each model is shown here:

| Model | Mean CV Error |
|---|---|
| Logistic Ridge Regression Model | 0.2331 |
| Neural Network | 0.3654 |
| Additive (Boosted) Logistic Regression Model | 0.452 |
| Best Subsets Logistic Regression Model | 1.64 |
| Full Logistic Regression Model | 1.805 |
| Full Regression Model, Poisson | 2.091 |

## Discussion

Many data mining methods were explored for this project. It was determined that the additive logistic regression algorithm (`ada` package) produced the best results, with the best subsets logistic regression coming in second. It is believed that the advantage of the `ada` package is due to its ability to combine a number of simpler models. Our hypothesis regarding the manually implemented Adaboost algorithm was incorrect, as it resulted in lower accuracy than the best subsets regression model.

Here the relationship between the Cross Validation error on training data and accuracy on the test data is shown:



While this plot does indicate a general negative relationship between mean CV error and test data accuracy (i.e. as CV error increases, test accuracy decreases), the sample is too small to determine whether or not this data is linear or follows a higher order relationship. That said, the associated regression data for the linear best fit in the plot above is shown here:

| | Estimate | Std. Error | t value | Pr(>\|t\|) |
|---|---|---|---|---|
| **CV** | -0.01942 | 0.007334 | -2.647 | 0.05714 |
| **(Intercept)** | 0.7933 | 0.009799 | 80.96 | 1.395e-07 |

Table 5: Fitting linear model: Accuracy ~ CV

| Observations | Residual Std. Error | $R^2$ | Adjusted $R^2$ |
|---|---|---|---|
| 6 | 0.01368 | 0.6366 | 0.5458 |

This model shows that the slope given by CV, while not quite significant yet (p = 0.057), does have the potential to become significant upon the growth of the sample.

The performance of the best subsets logistic regression model was surprising given its reliance on only three variables (`Married`, `Credit_History` and `SemiUrban`) for prediction of the target variable, `Loan_Status`. Despite this strong performance, the best subsets logistic regression model alone resulted in an overall leaderboard rank of 161. The most significant predictors found here differs from those found by Agbemava et al. (2016). This is most likely due to the population studied – higher income individuals in this project as opposed to lower income cases.

Using our best model, the typical applicant approved for a loan was a married male with no dependents, not self-employed, earning approximately $4000 per month, and requested a loan of about $125000. These results are similar to that seen in the training data set and leans towards higher earners with a lower debt-to-income ratio when compared to the average applicant.

Limitations of this project include the requirement to work with only the data supplied by the data challenge – the given 14 predictor variables. It is most likely that stronger models could be built upon the acquisition of more data. However, the data provided reflects the realistic scenario of having limited data from a loan application, particularly in the situation where an individual may be a new customer of the bank. In subsequent trials and analyses, this project would benefit greatly from further experimentation with more advanced additive logistic regression models and with neural network architectures.

At the time of writing this report, we are currently tied for $8^{th}$ place out of a total of 268 participants. Our results show that even foundational data mining and machine learning techniques can produce powerful prediction models and quite accurate results.

## References


Agbemava, E., Nyarko, I. K., Adade, T. C., & Bediako, A. K. (2016). Logistic Regression Analysis Of Predictors Of Loan Defaults By Customers Of Non-Traditional Banks In Ghana. European Scientific Journal, 12(1).

Conlin, M. (1999). Peer group micro-lending programs in Canada and the United States. Journal of Development Economics, 60(1), 249-269.

Cule, E., & De Iorio, M. (2012). A semi-automatic method to guide the choice of ridge parameter in ridge regression. arXiv preprint arXiv:1205.0686.

Dobbin, K. K., & Simon, R. M. (2011). Optimally splitting cases for training and testing high dimensional classifiers. BMC medical genomics, 4(1), 1.

Günther, F., & Fritsch, S. (2010). neuralnet: Training of neural networks. The R journal, 2(1), 30-38.

"Home Mortgage Disclosure Act." Consumer Financial Protection Bureau: An Official Website of the United States Government. Consumer Financial Protection Bureau, n.d. Web. 17 July 2016. http://www.consumerfinance.gov/data-research/hmda/.

Lusardi, A., & Scheresberg, C. D. B. (2013). Financial literacy and high-cost borrowing in the United States (No. w18969). National Bureau of Economic Research.

Olson, D. L., Delen, D., & Meng, Y. (2012). Comparative analysis of data mining methods for bankruptcy prediction. Decision Support Systems, 52(2), 464-473.

Thomas, L. C. (2000). A survey of credit and behavioural scoring: forecasting financial risk of lending to consumers. International journal of forecasting, 16(2), 149-172.

## Appendix – R Code

```
##### setup #####
library(knitr)
opts_chunk$set(echo = FALSE, warning = FALSE, message = FALSE,
    comment = NA, fig.align = "center", error = TRUE)

library(glmnet)
library(leaps)
library(pROC)
library(car)
library(MASS)
library(ROCR)
library(ggplot2)
library(stringr)
library(dplyr)
library(reshape2)
library(vcd)
library(VIM)
library(pander)
library(tidyr)
library(e1071)


train <- read.csv("https://github.com/dsmilo/DATA621/raw/master/Final-Project/loan_train.csv",
    stringsAsFactors = FALSE)
test = read.csv("https://github.com/dsmilo/DATA621/raw/master/Final-Project/loan_test.csv",
    stringsAsFactors = FALSE)
test2 = test

predict.regsubsets <- function(object, newdata, id, ...) {
    form <- as.formula(object$call[[2]])
    mat <- model.matrix(form, newdata)
    coefi <- coef(object, id = id)
    xvars <- names(coefi)
    mat[, xvars] %*% coefi
}

plotROC <- function(model, ndata, gtruth, name = "") {
    prob <- predict(model, newdata = ndata, type = "response")
    pred <- prediction(prob, gtruth)
    perf <- performance(pred, measure = "tpr", x.measure = "fpr")
    auc <- performance(pred, measure = "auc")
    auc <- auc@y.values[[1]]
    roc.data <- data.frame(fpr = unlist(perf@x.values), tpr = unlist(perf@y.values),
        model = "GLM")
    ggplot(roc.data, aes(x = fpr, ymin = 0, ymax = tpr)) + geom_ribbon(alpha = 0.2) +
        geom_line(aes(y = tpr)) + ggtitle(paste0(name, " ROC Curve w/ AUC=",
        (auc)))
}

fillwithmedian <- function(x) {
```

```
    median_val = median(x, na.rm = TRUE)
    x[is.na(x)] = median_val
    return(x)
}
###################################


##### literature review #####
library(ReadImages)  #Had to install manually
library(grid)
download.file("https://cs231n.github.io/assets/nn1/neural_net2.jpeg",
    "nn1.jpeg")
img <- read.jpeg("nn1.jpeg")
grid.raster(img)
###################################


##### data exploration and manipulation #####
train3 = train
test2 = test

train3$Gender = ifelse(train$Gender == "Male", 1, 0)
train3$Married = ifelse(train$Married == "Yes", 1, 0)
train3$Dependents[train3$Dependents == "3+"] = 3
train3$Dependents = as.numeric(train3$Dependents)
train3$Education = ifelse(train$Education == "Graduate", 1, 0)
train3$Self_Employed = ifelse(train$Self_Employed == "Yes", 1,
    0)

train3$Urban = ifelse(train$Property_Area == "Urban", 1, 0)
train3$SemiUrban = ifelse(train$Property_Area == "Semiurban",
    1, 0)
train3$TotalIncome = train3$ApplicantIncome + train3$CoapplicantIncome
train3$LTI = (train3$LoanAmount * 1000)/train3$ApplicantIncome  #Loan to income ratio

train3$Loan_Status = ifelse(train$Loan_Status == "Y", 1, 0)


test2$Gender = ifelse(test$Gender == "Male", 1, 0)
test2$Married = ifelse(test$Married == "Yes", 1, 0)
test2$Dependents[test2$Dependents == "3+"] = 3
test2$Dependents = as.numeric(test2$Dependents)
test2$Education = ifelse(test$Education == "Graduate", 1, 0)
test2$Self_Employed = ifelse(test$Self_Employed == "Yes", 1,
    0)

test2$Urban = ifelse(test$Property_Area == "Urban", 1, 0)
test2$SemiUrban = ifelse(test$Property_Area == "Semiurban", 1,
    0)
test2$TotalIncome = test2$ApplicantIncome + test2$CoapplicantIncome
test2$LTI = (test2$LoanAmount * 1000)/test2$ApplicantIncome  #Loan to income ratio
```

```r
train4 = train3[complete.cases(train3), ]

train = train3[, -1]
train2 = train4
test = test2


train_var = train
train_var$Property_Area = as.factor(train_var$Property_Area)
var_class <- data.frame(Class = rep(NA, ncol(train_var) - 4),
    Levels = rep(NA, ncol(train_var) - 4), stringsAsFactors = FALSE,
    check.names = FALSE, row.names = names(train_var)[1:(ncol(train_var) -
        4)])
names(var_class) = c("Class", "Levels/Range")

for (i in 1:(ncol(train_var) - 3)) {
    var_class[i - 1, 1] <- class(train_var[, i])
    val = length(levels(train_var[, i]))
    if (val == 0) {
        val = paste0(floor(min(train_var[, i], na.rm = TRUE)),
            " - ", max(train_var[, i], na.rm = TRUE))
    }
    var_class[i - 1, 2] <- val
}
pander(var_class)

# continuous variables
cont_vars <- train_var %>% dplyr::select(c(ApplicantIncome, CoapplicantIncome,
    LoanAmount, Loan_Amount_Term))

melted <- melt(cont_vars)

ggplot(melted, aes(value)) + geom_density(aes(fill = variable,
    col = variable), alpha = 0.5, show.legend = FALSE) + facet_wrap(~variable,
    scales = "free") + scale_y_continuous("", labels = NULL) +
    ggtitle("Distribution of Continuous Predictors \n") + theme(axis.ticks = element_blank())

# discrete variables
train_var$Property_Area <- as.numeric(train_var$Property_Area)
poisson_vars <- train_var %>% dplyr::select(-c(ApplicantIncome,
    CoapplicantIncome, LoanAmount, Loan_Amount_Term, TotalIncome,
    LTI, Urban, SemiUrban))

melted <- melt(poisson_vars)

library(scales)
ggplot(melted, aes(value, y = 6 * ..count../sum(..count..))) +
    geom_bar(aes(fill = variable, col = variable), alpha = 0.5,
        show.legend = FALSE) + facet_wrap(~variable, scales = "free_x") +
    scale_y_continuous("", labels = percent) + ggtitle("Distribution of Discrete Predictors \n")

# created variables
```

```
created_vars <- train_var %>% dplyr::select(c(TotalIncome, LTI))

melted <- melt(created_vars)

ggplot(melted, aes(value)) + geom_density(aes(fill = variable,
    col = variable), alpha = 0.5, show.legend = FALSE) + facet_wrap(~variable,
    scales = "free") + scale_y_continuous("", labels = NULL) +
    ggtitle("Distribution of Created Predictors \n") + theme(axis.ticks = element_blank())

# missing values
aggr(train_var, plot = TRUE, sortVars = FALSE, numbers = TRUE,
    cex.lab = TRUE)
###################################


##### Model Creation ##### Full Model
glmflagfull <- glm(Loan_Status ~ . - Property_Area, data = train,
    family = poisson, control = list(maxit = 100))
summary(glmflagfull)
vif(glmflagfull)  #Check for collinearity, VIF > 10
predglmflagfull <- data.frame(class = train$Loan_Status, logit = predict(glmflagfull,
    train))

plotROC(model = glmflagfull, ndata = train, gtruth = train$Loan_Status,
    name = "Full Model (Poisson)")

a = predict(glmflagfull, test, type = "response")
testdf = data.frame(Loan_ID = test$Loan_ID, Loan_Status = a)
testdf$Loan_Status[is.na(testdf$Loan_Status)] = median(testdf$Loan_Status,
    na.rm = TRUE)
testdf$Loan_Status = ifelse(testdf$Loan_Status > 0.5, "Y", "N")

write.csv(testdf, paste0("finalresults-", as.numeric(Sys.time()),
    ".csv"))
###############

# Best Subsets
regfit.full <- regsubsets(Loan_Status ~ . - Property_Area - Loan_ID,
    data = train3, nvmax = 17)
summary(regfit.full)
par(mar = c(1, 1, 1, 1))
par(mfrow = c(1, 2))
plot(regfit.full, scale = "bic", main = "Predictor Variables vs. BIC")
reg.summary <- summary(regfit.full)
reg.summary$bic
plot(reg.summary$bic, xlab = "Number of Predictors", ylab = "BIC",
    type = "l", main = "Best Subset Selection Using BIC")
minbic <- which.min(reg.summary$bic)
points(minbic, reg.summary$bic[minbic], col = "brown", cex = 2,
    pch = 20)
coef(regfit.full, minbic)
var_names = names(coef(regfit.full, minbic))[2:length(names(coef(regfit.full,
```

```
    minbic)))]
length(var_names)

Model_toEval = paste0("glm(Loan_Status ~ ", paste(var_names,
    collapse = " + "), ", data = train, family = binomial(link='logit'))")

bestsubset21 = eval(parse(text = Model_toEval))

summary(bestsubset21)

plotROC(model = bestsubset21, ndata = train, gtruth = train$Loan_Status,
    name = "Best Subsets")

a = predict(bestsubset21, test, type = "response")
testdf = data.frame(Loan_ID = test$Loan_ID, Loan_Status = a)
testdf$Loan_Status[is.na(testdf$Loan_Status)] = median(testdf$Loan_Status,
    na.rm = TRUE)
testdf$Loan_Status = ifelse(testdf$Loan_Status > 0.5, "Y", "N")
###############

# Logistic Ridge
library(ridge)  #installed from source
train$Loan_Status = as.numeric(as.character(train$Loan_Status))
train_lr <- data.frame(lapply(train[, -11], fillwithmedian))
train_lr$Property_Area = train$Property_Area

lr2 = logisticRidge(Loan_Status ~ . - Property_Area, data = train_lr)

plotROC(model = lr2, ndata = train, gtruth = train$Loan_Status,
    name = "Logistic Ridge")

a = predict(lr2, test2, type = "response")
testdf = data.frame(Loan_ID = test$Loan_ID, Loan_Status = a)
testdf$Loan_Status[is.na(testdf$Loan_Status)] = median(testdf$Loan_Status,
    na.rm = TRUE)
testdf$Loan_Status = ifelse(testdf$Loan_Status > 0.5, "Y", "N")
write.csv(testdf, paste0("finalresults-", as.numeric(Sys.time()),
    ".csv"))
###############

# Full model binomial
glmflagfull <- glm(Loan_Status ~ . - Property_Area, data = train,
    family = binomial, control = list(maxit = 100))
summary(glmflagfull)
vif(glmflagfull)  #Check for collinearity, VIF > 10
predglmflagfull <- data.frame(class = train$Loan_Status, logit = predict(glmflagfull,
    train))

plotROC(model = glmflagfull, ndata = train, gtruth = train$Loan_Status,
    name = "Full Model Binomial")

a = predict(glmflagfull, test, type = "response")
```

```
testdf = data.frame(Loan_ID = test$Loan_ID, Loan_Status = a)
testdf$Loan_Status[is.na(testdf$Loan_Status)] = median(testdf$Loan_Status,
    na.rm = TRUE)
testdf$Loan_Status = ifelse(testdf$Loan_Status > 0.5, "Y", "N")
write.csv(testdf, paste0("finalresults-", as.numeric(Sys.time()),
    ".csv"))
###############

# Neural Net Determination of nodes for neuralnet
k = 10
set.seed(1306)
folds = sample(1:k, nrow(train), replace = TRUE)
cv.errors1 = matrix(NA, k, 10, dimnames = list(NULL, paste(1:10)))
cv.errors2 = matrix(NA, k, 10, dimnames = list(NULL, paste(1:10)))
cv.errors3 = matrix(NA, k, 10, dimnames = list(NULL, paste(1:10)))
cv.errors4 = matrix(NA, k, 10, dimnames = list(NULL, paste(1:10)))
cv.errors5 = matrix(NA, k, 10, dimnames = list(NULL, paste(1:10)))
cv.errors6 = matrix(NA, k, 10, dimnames = list(NULL, paste(1:10)))
cv.errors7 = matrix(NA, k, 10, dimnames = list(NULL, paste(1:10)))
cv.errors8 = matrix(NA, k, 10, dimnames = list(NULL, paste(1:10)))

train_2 <- data.frame(lapply(train[, -11], fillwithmedian))
train_2$Property_Area = train$Property_Area
scaled = data.frame(lapply(scaled, fillwithmedian))

num_layers = 0

f1 <- as.formula(paste("Loan_Status ~", paste(n[!n %in% "Loan_Status"],
    collapse = " + ")))
for (j in 1:k) {

    num_layers = num_layers + 5
    print(num_layers)
    dat = scaled[folds != j, ]
    dat = dat[complete.cases(dat), ]
    model5 <- neuralnet(f1, data = dat, hidden = c(num_layers),
        linear.output = TRUE, stepmax = 1e+07)

    for (i in 1:10) {
        print(i)
        f = train_2[folds == j, ]
        f = f[complete.cases(f), ]
        dat = scaled[folds == j, ]
        dat2 = dat[complete.cases(dat), -11]
        pred5 = neuralnet::compute(model5, dat2)
        netresult = pred5$net.result[, 1]
        sc = scaled$Loan_Status[folds == j]
        sc = sc[complete.cases(sc)]
        cv.errors5[j, i] = mean((sc - netresult)^2, na.rm = TRUE)
    }

}
```

```
cv_means = rowMeans(cv.errors5)
node_counts = seq(5, 50, 5)
nn_df = data.frame(Nodes = node_counts, CV_Error = cv_means)
names(nn_df) = c("Nodes", "Mean CV Error")
write.csv(nn_df, "~/Downloads/nn_df.csv", row.names = FALSE)

nn_df = read.csv("https://raw.githubusercontent.com/aadikalloo/AadiMSDA/master/IS621-Data-Mining/nn_d
plot(x = nn_df$Nodes, y = nn_df$Mean.CV.Error, xlab = "Nodes",
    ylab = "Mean CV Error", main = "Determination of Number of Nodes for neuralnet",
    pch = ifelse(nn_df$Nodes == 10, 19, 1))
##########

## Neural network
train4 = train3
train4 = train4[complete.cases(train4), ]
train4 = train4[, -c(1, 12)]
maxs <- apply(train4, 2, max)
mins <- apply(train4, 2, min)
scaled <- as.data.frame(scale(train4, center = mins, scale = maxs -
    mins))
n <- names(scaled)
f <- as.formula(paste("Loan_Status ~", paste(n[!n %in% "Loan_Status"],
    collapse = " + ")))
nn = neuralnet(f, data = scaled, hidden = c(10), linear.output = TRUE,
    stepmax = 1e+06)
plot(nn)

max2 <- function(x) {
    return(max(x, na.rm = TRUE))
}

min2 <- function(x) {
    return(min(x, na.rm = TRUE))
}

test <- data.frame(lapply(test, fillwithmedian))
test = test2
test = test[, -c(1, 12)]
test <- data.frame(lapply(test, fillwithmedian))

maxs.t <- apply(test, 2, max2)
mins.t <- apply(test, 2, min2)
scaled.t <- as.data.frame(scale(test, center = mins.t, scale = maxs.t -
    mins.t))
scaled.t = as.data.frame(lapply(scaled.t, fillwithmedian))
pr.test <- neuralnet::compute(nn, scaled.t)
pr.t_ <- pr.test$net.result * (max(train4$Loan_Status) - min(train4$Loan_Status)) +
    min(train4$Loan_Status)
pr_nn.t = ifelse(pr.t_ > 0.5, "Y", "N")

test1 = read.csv("https://github.com/dsmilo/DATA621/raw/master/Final-Project/loan_test.csv",
    stringsAsFactors = FALSE)
```

```
df = data.frame(Loan_ID = test1$Loan_ID, Loan_Status = as.data.frame(as.vector(pr_nn.t)))
names(df) = c("Loan_ID", "Loan_Status")
sum(is.na(df[, 2]))  #hopefully 0
write.csv(df, paste0("nn-", as.numeric(Sys.time()), ".csv"),
    row.names = FALSE)

download.file("https://github.com/aadikalloo/AadiMSDA/raw/master/IS621-Data-Mining/nn2.jpg",
    "nn2.jpeg")
img <- read.jpeg("nn2.jpeg")
grid.raster(img)
##########


# ada package analysis
model1 = list()

for (i in 1:20) {
    iters = 25 + 25 * i
    model1[[i]] = ada(formula = Loan_Status ~ ., data = train_x1,
        iter = iters, loss = "logistic")
    results = as.numeric(as.character(predict(model1[[i]], test)))
    results = ifelse(results == 1, "Y", "N")
    testdf = data.frame(Loan_ID = test2$Loan_ID, Loan_Status = results)
    write.csv(testdf, paste0("finalresults-", as.numeric(Sys.time()),
        " - ", iters, ".csv"), row.names = FALSE)
}

ada_results = read.csv("https://raw.githubusercontent.com/aadikalloo/AadiMSDA/master/IS621-Data-Minir
    header = FALSE)
names(ada_results) = c("Iterations", "Accuracy")
###############


# Adaboost manual implementation
wts = list()

models = list()
alpha = list()
error_rate = list()
final = 0
predictions = list()
total = 12

train_x <- data.frame(lapply(train[-c(1, 2), -11], fillwithmedian))
test_x = data.frame(lapply(test2[, -c(1, 12)], fillwithmedian))

k = total
set.seed(1306)
folds = sample(1:k, nrow(train_x), replace = TRUE)
folds = rep(1:12, nrow(train_x)/12)

wts[[1]] = rep(1/length(train_x1[, 1]), length(train_x1[, 1]))
```

```
Y_hat_run = list()
Y_hat_run[[1]] = rep(1, length(train_x1[, 1]))

for (i in 1:total) {
    train_x1 = train_x

    models[[i]] = glm(Loan_Status ~ ., data = train_x1, family = binomial,
        weights = wts[[i]])
    Y_hat = predict(models[[i]], train_x1, type = "response")
    Y_hat_run = Y_hat_run[[i]] * Y_hat
    Y_hat = ifelse(Y_hat > 0.5, 1, 0)
    error_rate[[i]] = sum(wts[[i]] * (train_x1$Loan_Status !=
        Y_hat))/length(Y_hat)
    alpha[[i]] = (1/2) * log((1 - error_rate[[i]])/error_rate[[i]])
    loan_status = ifelse(train_x1$Loan_Status > 0, 1, -1)
    Y_hat = ifelse(Y_hat == 1, 1, -1)
    wts[[i + 1]] = wts[[i]] * exp(-alpha[[i]] * loan_status *
        Y_hat)
    wts[[i + 1]] = wts[[i + 1]]/sum(wts[[i + 1]])
}

for (i in 1:total) {
    predictions[[i]] = predict(models[[i]], test_x, type = "response")
    predictions[[i]] = alpha[[i]] * predictions[[i]]
}

toEval = "predictions[[1]]"

for (i in 2:4) {
    toEval = paste0(toEval, " * predictions[[", i, "]]")
}

prop = table(train$Loan_Status)[[1]]/table(train$Loan_Status)[[2]]
new = Y_hat_run[order(Y_hat_run)]
pred2 = ifelse((rank(Y_hat_run)) > prop * length(test[, 1]),
    1, 0)

results = eval(parse(text = toEval))
results = ifelse(results < error_rate[[1]], "N", "Y")

testdf = data.frame(Loan_ID = test2$Loan_ID, Loan_Status = results)

write.csv(testdf, paste0("finalresults-", as.numeric(Sys.time()),
    ".csv"), row.names = FALSE)
###############


##### Results #####
data_results = data.frame(Model = c("ada package", "Best Subsets",
    "Logistic Ridge", "Full Model, Binomial", "Full Model, Poisson",
    "Neural Network (neuralnet package)", "Manually implemented adaboost algorithm"),
    Accuracy = c("0.8055", "0.7778", "0.7708", "0.7638", "0.7708",
```

```
        "0.7431", "0.7153"), stringsAsFactors = FALSE)
data_results2 = data_results[order(data_results$Accuracy, decreasing = TRUE),
    ]
kable(data_results2)

ada_results = read.csv("https://raw.githubusercontent.com/aadikalloo/AadiMSDA/master/IS621-Data-Minin
    header = FALSE)
names(ada_results) = c("Iterations", "Accuracy")
plot(x = ada_results$Iterations, y = ada_results$Accuracy, xlab = "Iterations",
    ylab = "Accuracy", ylim = c(0.72, 0.82), main = "Effect of number of iteratons on test data accur
    pch = ifelse(ada_results$Accuracy == 0.8055, 19, 1))
lines(smooth.spline(x = ada_results$Iterations, y = ada_results$Accuracy))

download.file("https://raw.githubusercontent.com/aadikalloo/AadiMSDA/master/IS621-Data-Mining/roc2.jp
    "roc.jpeg")
img <- read.jpeg("roc.jpeg")
grid.raster(img)

# 10-fold CV
k = 10
set.seed(1306)
folds = sample(1:k, nrow(train), replace = TRUE)
cv.errors1 = matrix(NA, k, 10, dimnames = list(NULL, paste(1:10)))
cv.errors2 = matrix(NA, k, 10, dimnames = list(NULL, paste(1:10)))
cv.errors3 = matrix(NA, k, 10, dimnames = list(NULL, paste(1:10)))
cv.errors4 = matrix(NA, k, 10, dimnames = list(NULL, paste(1:10)))
cv.errors5 = matrix(NA, k, 10, dimnames = list(NULL, paste(1:10)))
cv.errors6 = matrix(NA, k, 10, dimnames = list(NULL, paste(1:10)))
cv.errors7 = matrix(NA, k, 10, dimnames = list(NULL, paste(1:10)))
cv.errors8 = matrix(NA, k, 10, dimnames = list(NULL, paste(1:10)))

train_2 <- data.frame(lapply(train[, -11], fillwithmedian))
train_2$Property_Area = train$Property_Area
scaled = data.frame(lapply(scaled, fillwithmedian))

f1 <- as.formula(paste("Loan_Status ~", paste(n[!n %in% "Loan_Status"],
    collapse = " + ")))
for (j in 1:k) {
    print(j)

    model1 <- glm(Loan_Status ~ . - Property_Area, data = train_2[folds !=
        j, ], family = poisson, control = list(maxit = 100))
    model2 <- glm(formula = Loan_Status ~ Married + Credit_History +
        SemiUrban, family = binomial(link = "logit"), data = train_2[folds !=
        j, ])
    model3 <- logisticRidge(Loan_Status ~ . - Property_Area,
        data = train_2[folds != j, ])
    model4 <- glm(Loan_Status ~ . - Property_Area, data = train_2[folds !=
        j, ], family = binomial, control = list(maxit = 100))
    dat = scaled[folds != j, ]
    dat = dat[complete.cases(dat), ]
    model5 <- neuralnet(f1, data = dat, hidden = c(5, 3), linear.output = TRUE,
```

```
        stepmax = 1e+07)
    model6 <- ada(formula = Loan_Status ~ ., data = train_x1[folds !=
        j, ], iter = 125, loss = "logistic")

    for (i in 1:10) {
        print(i)
        f = train_2[folds == j, ]
        f = f[complete.cases(f), ]

        pred1 = predict(model1, f, id = i)
        cv.errors1[j, i] = mean((train_2$Loan_Status[folds ==
            j] - pred1)^2, na.rm = TRUE)

        pred2 = predict(model2, f, id = i)
        cv.errors2[j, i] = mean((train_2$Loan_Status[folds ==
            j] - pred2)^2, na.rm = TRUE)

        pred3 = predict(model3, f, id = i)
        cv.errors3[j, i] = mean((train_2$Loan_Status[folds ==
            j] - pred3)^2, na.rm = TRUE)

        pred4 = predict(model4, f, id = i)
        cv.errors4[j, i] = mean((train_2$Loan_Status[folds ==
            j] - pred4)^2, na.rm = TRUE)

        dat = scaled[folds == j, ]
        dat = dat[complete.cases(dat), -11]
        pred5 = neuralnet::compute(model5, dat2)
        netresult = pred5$net.result[, 1]
        sc = scaled$Loan_Status[folds == j]
        sc = sc[complete.cases(sc)]
        cv.errors5[j, i] = mean((sc - netresult)^2, na.rm = TRUE)

        pred6 = predict(model6, train_x1[folds == j, ], id = i,
            type = "probs")
        cv.errors6[j, i] = mean((train_2$Loan_Status[folds ==
            j] - pred6)^2, na.rm = TRUE)

    }

}

mean.cv.errors1 <- apply(cv.errors1, 2, mean)
mean.cv.errors2 <- apply(cv.errors2, 2, mean)
mean.cv.errors3 <- apply(cv.errors3, 2, mean)
mean.cv.errors4 <- apply(cv.errors4, 2, mean)
mean.cv.errors5 <- apply(cv.errors5, 2, mean)
mean.cv.errors6 <- apply(cv.errors6, 2, mean)

all.cv.error = data.frame(mean(mean.cv.errors1), mean(mean.cv.errors2),
    mean(mean.cv.errors3), mean(mean.cv.errors4), mean(mean.cv.errors5),
    mean(mean.cv.errors6))
```

```
names(all.cv.error) = c("Full Regression Model, Poisson", "Best Subsets Logistic Regression Model",
    "Logistic Ridge Regression Model", "Full Logistic Regression Model",
    "Neural Network", "Additive (Boosted) Logistic Regression Model")
all.cv.error = t(all.cv.error)
names(all.cv.error) = c("Model", "Mean CV Error")

mean_cv_results = read.csv("https://github.com/aadikalloo/AadiMSDA/raw/master/IS621-Data-Mining/allcv
    check.names = FALSE)
mean_cv = mean_cv_results[order(mean_cv_results$"Mean CV Error"),
    ]
row.names(mean_cv) = NULL
pander(mean_cv)
###################################


##### discussion #####
acc_cv <- data.frame(Accuracy = as.numeric(data_results2$Accuracy[1:6]),
    CV = mean_cv$"Mean CV Error")
rownames(acc_cv) <- c("Adaboost", "Best Subsets Logistic", "Logistic Ridge",
    "Full Poisson", "Full Binomial", "Neural Network")

ggplot(acc_cv, aes(x = CV, y = Accuracy)) + geom_point() + geom_text(aes(label = rownames(acc_cv)),
    vjust = "inward", hjust = "inward", check_overlap = TRUE) +
    geom_smooth(alpha = 0.25, method = glm, fill = 3, col = 3,
        se = FALSE, lty = 3) + scale_x_continuous("Mean Cross-Validation Error") +
    ggtitle("Relationship between Training Set CV Error and Test Set Accuracy\n") +
    theme(plot.title = element_text(lineheight = 0.8, face = "bold")) +
    geom_smooth(alpha = 0.25, method = "loess", se = FALSE, lwd = 1,
        lty = 3)

alm = lm(Accuracy ~ CV, data = acc_cv)
pander(summary(alm))
###################################
```

# Final Project: Loan Approval

Data 621 Business Analytics and Data Mining

*Aadi Kalloo, Nathan Lim, Asher Meyers, Daniel Smilowitz, Logan Thomson*

*Due July 21, 2016*

## Contents



### Abstract

Regression and classification techniques are used to predict loan approval status of customers of a fictional bank given fourteen predictors. Data was obtained from a "data challenge", and the given variables were supplemented with transformations. Models were trained on a dataset of 614 cases, and subsequently used to predict the binary outcome of 367 test cases. Techniques utilized include logistic regression, ridge regression, the Adaboost algorithm, and a neural network. Analyses were performed to determine the optimal number of iterations and optimal number of nodes for the Adaboost algorithm and neural network, respectively. Cross validation results were not linearly co-variant with accuracy scores on the test data set. Using tools supplied by the data challenge, accuracy scores were used to compare models and the Adaboost algorithm was found to produce the best results. The optimal model created yielded an accuracy of 0.8056.

**Keywords: loan prediction, classification, logistic regression, adaboost**

## Introduction

Between 2009-2014, approximately 95 million mortgage loan applications were filed in the United States alone (Home Mortgage Disclosure Act, 2015). When further loan types, including education and personal loans, are considered, it becomes even clearer that the ability to accurately identify reliable borrowers is a high priority for banks and other financial institutions today. Likewise, consumers are significantly affected as well. Surveys and analyses performed by Lusardi & Scheresberg (2013) has shown that individuals of poor financial literacy, and subsequently poor credit history, are declined by banks for loans and mortgages, and turn to high-cost borrowing (i.e. payday loans). Thus, it becomes clear that the factors that determine which individuals are dependable borrowers is information important to both financial institutions and consumers alike.

The dataset of interest contains information about customers of a hypothetical bank and was obtained from a data challenge hosted at AnalyticsVidhya. This data set and challenge were chosen due to its relevance to the current loan-centric economy outlined above, and to address the need for more transparent analyses of optimal methods for financially-related prediction models.

## Literature Review

Although predictors can vary by loan or income level, studies investigating loan approval predictors have found that factors such as Marital Status, Loan Duration, and Number of Dependents can serve as statistically significant predictors of loan approval status (Agbemava et al., 2016). However, as loan approval criteria can vary greatly between financial institutions there is a significant lack of published materials investigating the effects that various factors may play on loan approval status. Furthermore, the majority of financial institutions use proprietary algorithms and statistical methods in order to determine loan approvals.

It should be noted that ratio of training set cases to test set cases is 1.67:1. This naturally leads to markedly decreased algorithm training than the 4:1 ratio that is normally used in prediction tasks (Dobin & Simon, 2011).

Of significant interest to us were the concepts of additive logistic regression or "boosting" as outlined by Friedman et al. (2012). The original concept of boosting was introduced by Schapire (1990) and describes a method of increasing the performance of "weak" classifiers by combining many of them into a more powerful model (Friedman et al., 2012). The authors of this paper introduce the Adaboost algorithm, which is used and analyzed in detail for this project. The Adaboost algorithm is characterized by the use of weights on the target variable, increasing the weight on each iteration for misclassified values (Friedman et al., 2012).

Given that this project was centered around a data challenge, it was also of particular significance to us to research data mining and machine learning methods that could potentially provide greater accuracy scores, despite being outside the scope of IS621 Data Mining. The primary method of prediction in this arena is the neural network. While the foundational ideas of artificial neural network date as far back as the 1940s (Warren & Pitts, 1943), the procedures used by the `neuralnet` package are based on Riedmiller and Braun (1993). As a brief overview, the `neuralnet` package calculates the following function as described by Gunther & Fritsch (2010):

$$o(x) = f(w_0 + \sum_{j=1}^{J} w_j \cdot f(w_{0j} + \sum_{i=1}^{n} w_{ij} x_i))$$

where $w_0$ is the intercept of the output neuron, $w_{0j}$ is the intercept of the $j^{th}$ hidden neuron, and $w_j$ is the synaptic weight.This mathematical model can be expressed visually, and is depicted in the following figure:

Given the fact that loan prediction has traditionally been a field of interest for those who can benefit from financial gain (i.e. big banks, as opposed to academia), research and methodological studies in this area is sparse. Most of the current literature in this area has been focused on financial solvency issues, namely bankruptcy. It has been found previously that logistic regression, neural networks, and classification trees are the three most commonly used data mining methods in the prediction of bankruptcy (Olson et al., 2012). As a means of extending this work to the arena of personal loans, this project serves to compare some of the most commonly used methods of prediction in loan outcomes. We believe that the information and methods investigated here can trickle down to a

smaller scale and be beneficial to smaller economies such as microlending, where loans tend to be oriented around civilian populations instead of financial institutions (Conlin, 1999).

## Methodology

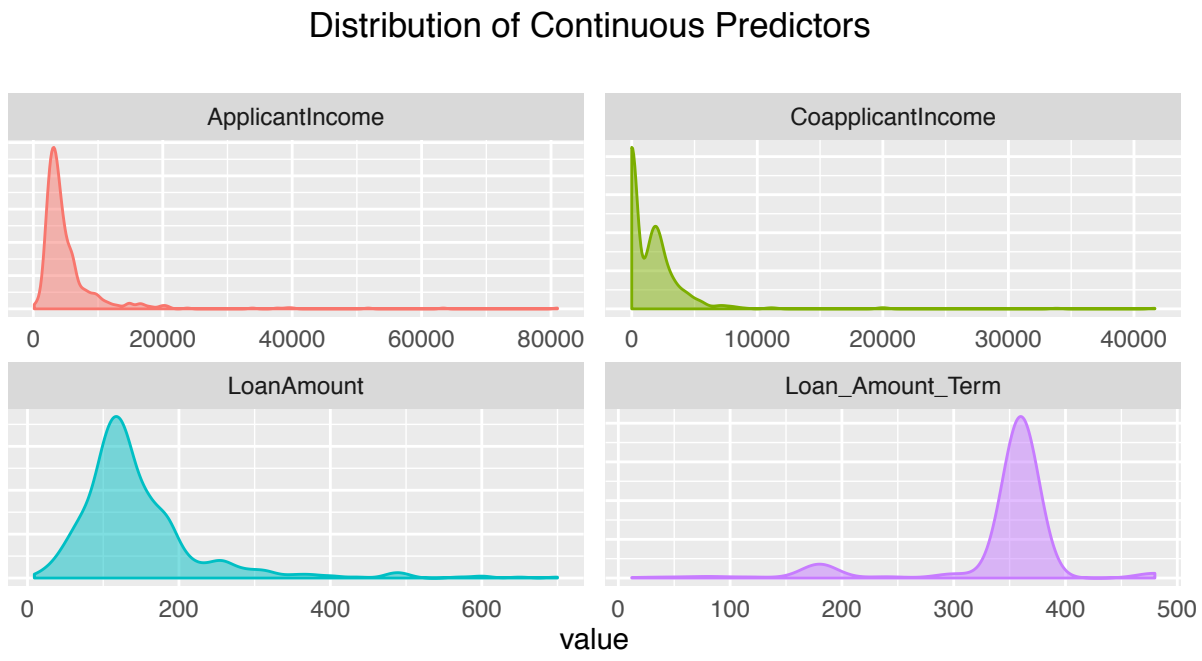
The dataset has 614 rows (each representing a customer) and 11 predictor variables. Furthermore, there is 1 identification variable, and 1 response variable: `Loan_Status`, a binary categorical variable representing whether each customer has been approved for a loan given their credentials. The goal of the data challenge and of this investigation is to predict the approval of loan applications based on the predictors.

### Data Exploration

The classes of the predictor and response variables are presented below:

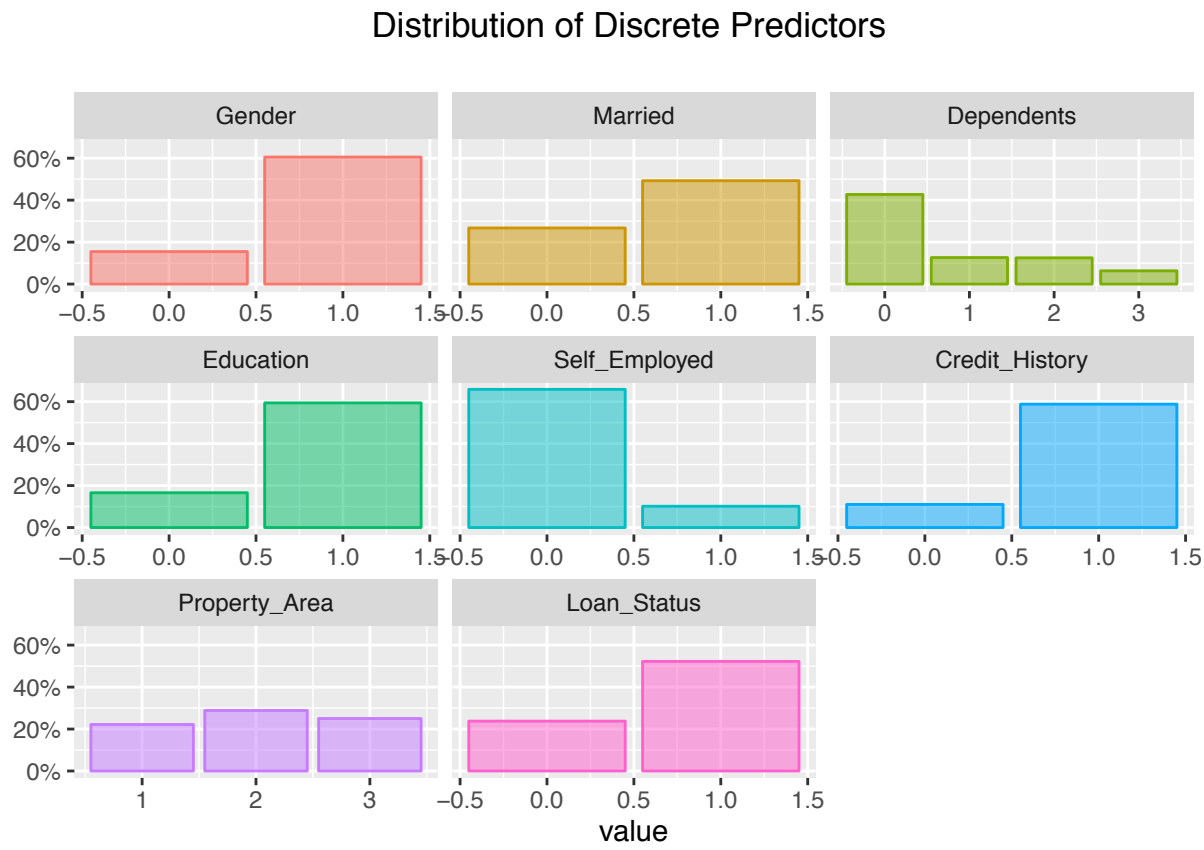
| | Class | Levels/Range |
|---|---|---|
| **Gender** | numeric | 0 - 1 |
| **Married** | numeric | 0 - 3 |
| **Dependents** | numeric | 0 - 1 |
| **Education** | numeric | 0 - 1 |
| **Self_Employed** | integer | 150 - 81000 |
| **ApplicantIncome** | numeric | 0 - 41667 |
| **CoapplicantIncome** | integer | 9 - 700 |
| **LoanAmount** | integer | 12 - 480 |
| **Loan_Amount_Term** | integer | 0 - 1 |
| **Credit_History** | factor | 3 |
| **Property_Area** | numeric | 0 - 1 |
| **Loan_Status** | numeric | 0 - 1 |

It can be seen that four of the predictors are continuous variables; the remaining 7 predictors, as well as the response, are categorical variables, with all but one of these variables (`Dependents`) being binary. The distributions of the continuous predictors are presented below:



It is clear that each of the continuous variables exhibits a strong level of skewness – `ApplicantIncome`, `CoapplicantIncome`, and `LoanAmount` are right-skewed, while `Loan_Amount_Term` is left-skewed, It can be seen that the peak in the distribution of `Loan_Amount_Term` is located at 360 – this corresponds to 30 years, the typical length of a mortgage.

The distributions of the discrete predictors, as well as the target, are presented below:




The last figure in the chart shows that roughly 75% of loan applicants are approved for a loan.

With the exception of `Property_Area`, each of the discrete predictors exhibits a dominant category. The distributions indicate that, with all other variables held constant, typical applicants:

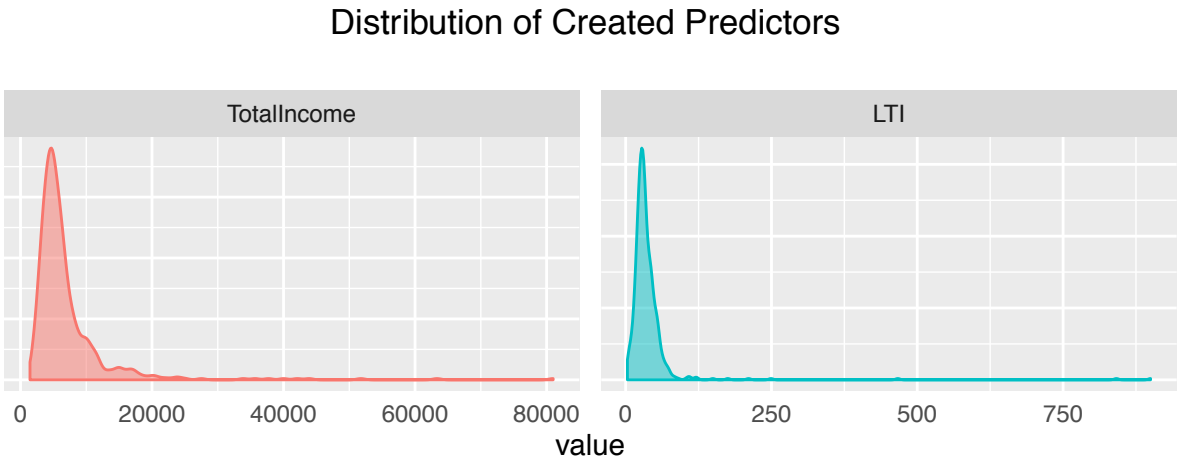
- Are male
- Are married
- Have no children
- Are graduates
- Are not self-employed
- Meet the firm's credit requirements

Inspection of the raw data set reveals that there are a number of missing values; these are addressed prior to the development of predictive models.

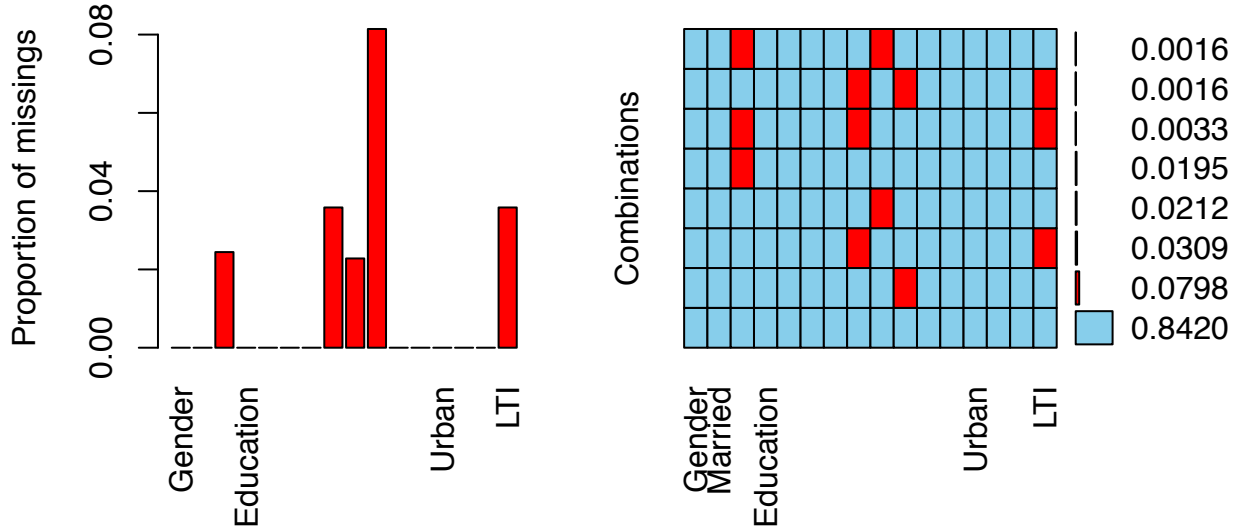
### Data Manipulation

The supplied training data set originally has 11 predictor variables, six of which are character vectors which needed to be converted to binary categorical variables. The response variable, stated as a simple "Y/N" was also converted into the 0 or 1 format for modelling. As it was deemed not significant and also not chosen by any automated model, `Rural` is excluded from the `Property_Area` predictor, leaving only `Urban` and `SemiUrban`, which are turned into separate binary categorical predictors. `Property_Area` is then excluded from the training dataset.

All other variables in the data set were numeric, or already in a binary format. Given the included variables, two new features were created. `TotalIncome` is the sum of applicant and co-applicant incomes, and `LTI` (Loan to Income Ratio) is the loan amount ($\times$ 1000) divided by the `ApplicantIncome` variable. The distributions of thse created variables are shown below; they exhibit a similar degree of right-skewness as the three continuous predictors used to create them.




Missing values were also present in the data, either in the form of empty characters (`""`), or `NA`s. Four of the predictors, `Gender`, `Married`, `Dependents`, and `Self_Employed` contained empty characters. Being only a small proportion of the cases ($<$ 1% to 8%), these were in a way imputed when the predictors were converted into binary categorical variables. Four other variables, `LoanAmount`, `Loan_Amount_Term`, `Credit_History`, and `LTI` all contained small numbers of `NA`s. These are either dropped from the dataset, or imputed with the median, depending on the model. Alternative imputation methods, such as filling with a weighted "coin toss" value yielded less accurate results in the models. A representation of the proportion of missing values is presented below:



**Model Creation**

In total, six models were created and used for prediction of the target variable. A tool was supplied by AnalyticsVidhya that reported accuracy (% of cases correctly predicted) upon submission of the test data predictions.

Logistic regression is a very common method of classifying data (Thomas, 2000). In logistic regression, a linear regression is used where the dependent variable is a non-linear function of the probability of the event occurring. In this project, the purpose of the logistic regression method is to classify cases as either "Approved for loan" or "Denied". The general logistic regression model can be represented as:

$$\ln \frac{p_i}{1-p_i} = \beta_0 + \beta_1 X_1 + ... + \beta_n X_n + \varepsilon$$

A few different logistic regression models were developed for this project. The first was a full model using all variables. This model made use of the `glm` function. This model was created as a basic point of comparison. This model produced an AIC of 935.

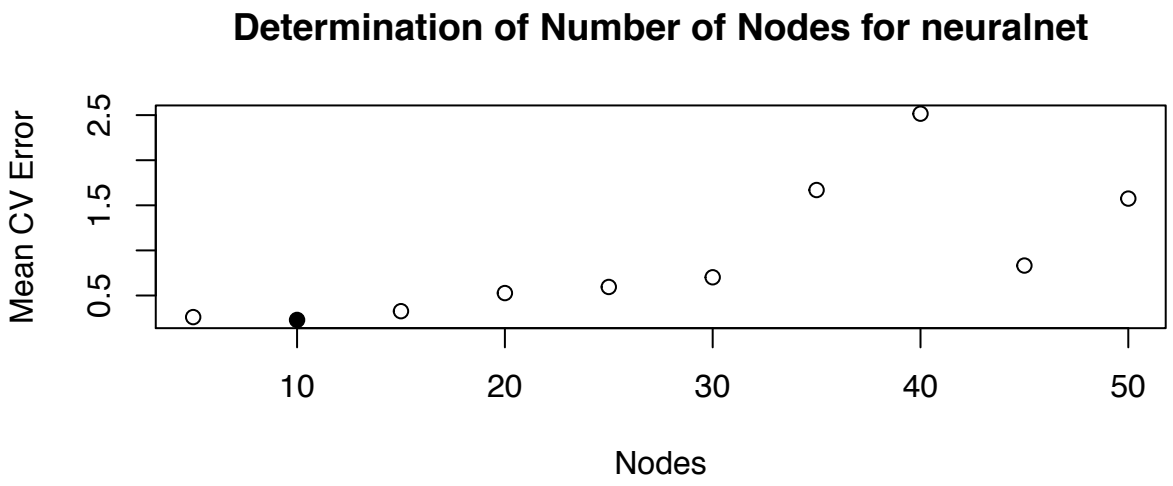
The second model created utilized both the `glm` function and the `leaps` package in order to find the best combination/subset of variables that minimizes the Bayesian Information Criterion (BIC). The "best subsets" approach is a method of automated variable selection that seeks to find the best predictors of the target variable. The `regsubsets` function as part of the `leaps` package found the `Married`, `Credit_History` and `SemiUrban` variables to be the best subset of predictors for this data set. The AIC reported for this model was 518.

Ridge regression methods were combined with logistic regression to create a "Logistic Ridge" model, the third of this project, using the `ridge` library. As this library is no longer actively in development, it was manually obtained and installed. Like Ridge Regression, Logistic Ridge Regression introduces a tuning parameter as a penalty to avoid overfitting (Cule & De Iorio, 2012). This is useful as the ridge regression method aims to reduce variance and produce lower mean squared error for higher-dimensional data such as that used in the full model for these data. The ridge penalty used in the `ridge` package as outlined by Cule & De Iorio (2012) is given as:

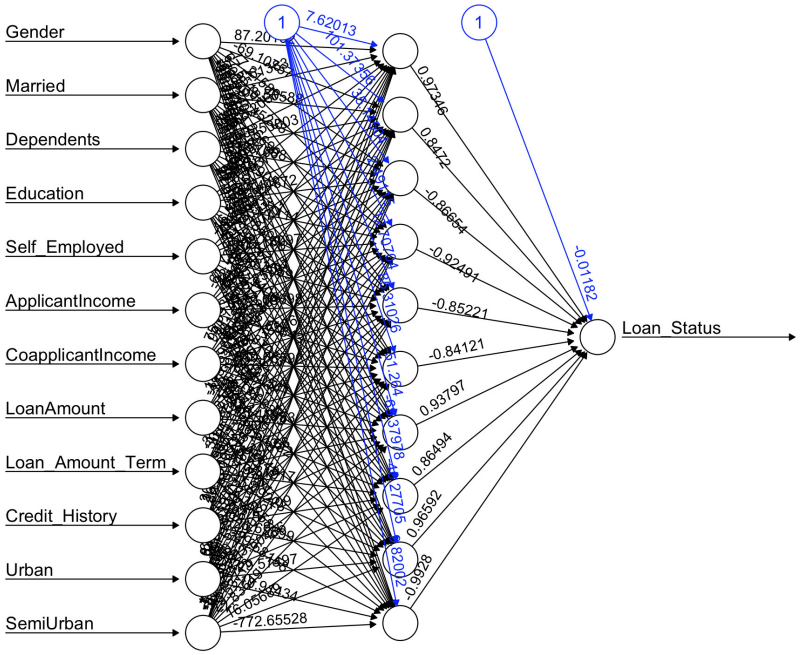
$$k = \frac{p}{\hat{\beta}'\hat{\beta}}$$

The Adaboost algorithm was used for this project in two forms. The first was through the use of the `ada` package. For this second implementation, the effect of the number of iterations of the algorithm on the final accuracy of the test set was analyzed with iterations in the range of 50-525 with steps of 25. It is hypothesized that accuracy will gradually increase with iterations to a peak, then gradually decrease as overfitting starts to become a prominent issue. The second was a manually coded implementation based on the algorithmic outline provided by Friedman et al. (2000). A manual approach was used for the purposes of comparing the academically published to the packaged algorithm. Given that these algorithms work by combining basic models, it is believed that they will perform better than a single logistic regression model alone.

A simple neural network method was pursued. The neural network approach was performed using normalized data with a single hidden layer. The number of nodes to use in the hidden layer was determined using a 10-fold Cross Validation Approach. A plot of the mean CV error against the number of hidden nodes is shown below:




From the above data, a network of 10 nodes was chosen due to its minimal cross validation error. This resulted in a network where the input variables were each sent to the 10 hidden nodes where weights were applied based on the variable's signifiance, and then sent to an output node that used its own weight to output the target variable classification. The illustration of the network used for prediction of the target variable is included here:



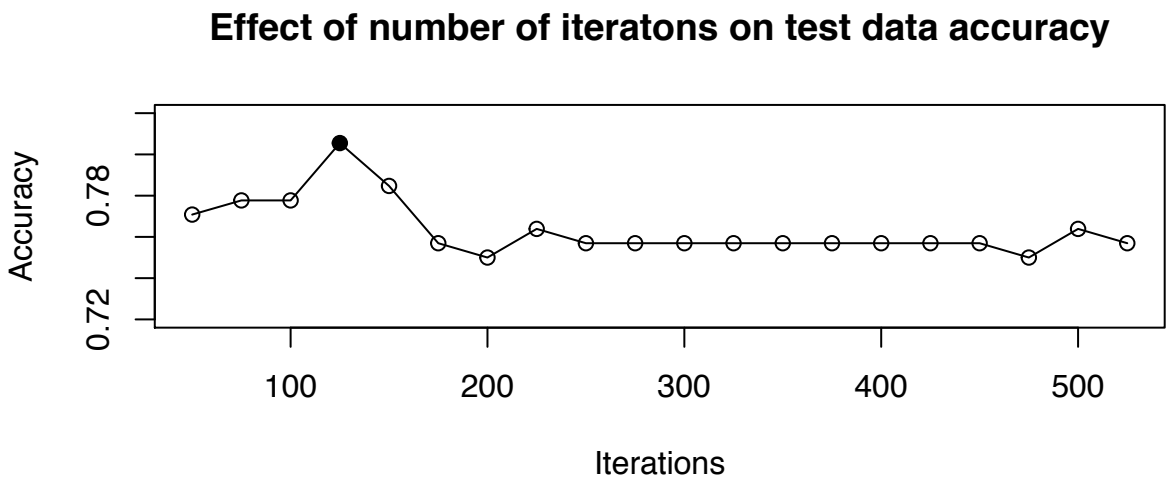

## Results

### Accuracy

The results obtained are summarized as follows:

| | Model | Accuracy |
|---|---|---|
| 1 | ada package | 0.8055 |
| 2 | Best Subsets | 0.7778 |
| 3 | Logistic Ridge | 0.7708 |
| 5 | Full Model, Poisson | 0.7708 |
| 4 | Full Model, Binomial | 0.7638 |
| 6 | Neural Network (neuralnet package) | 0.7431 |
| 7 | Manually implemented adaboost algorithm | 0.7153 |

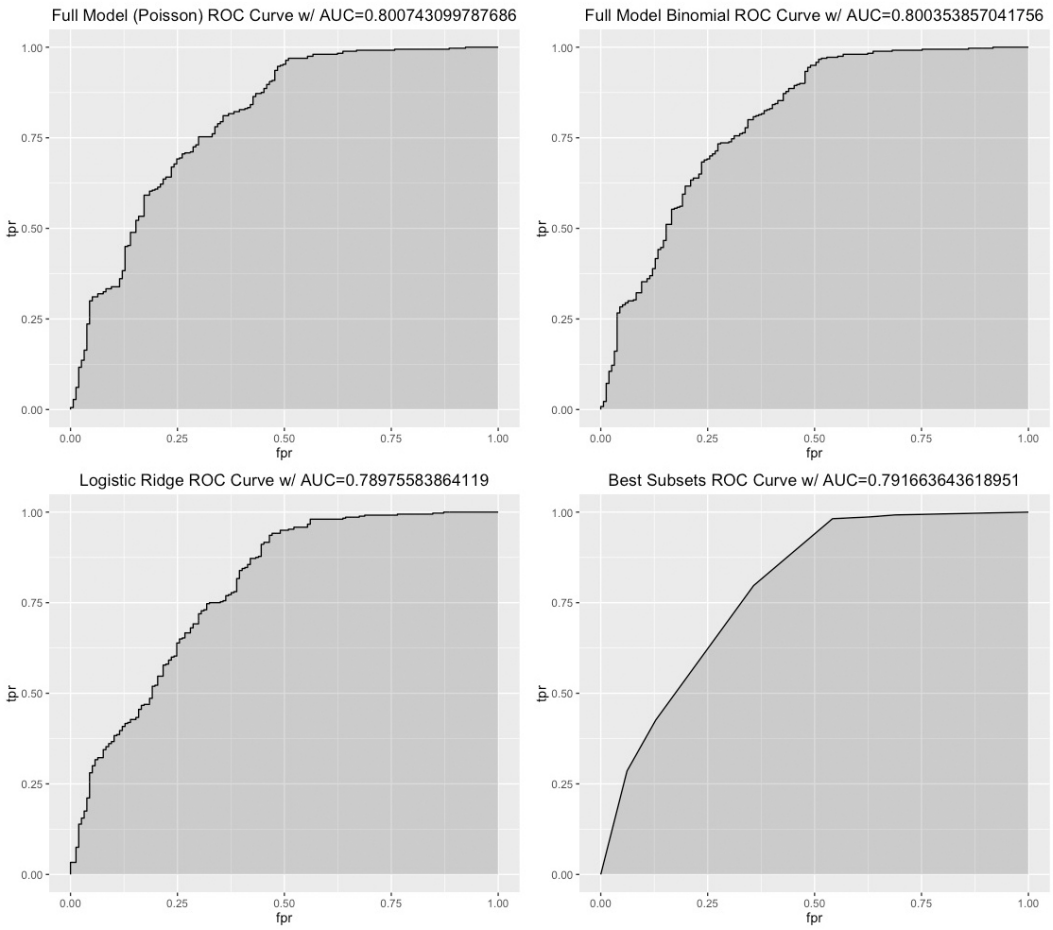
It can be seen that the best results were achieved through use of the `ada` package. The following figure illustrates the results of an analysis of test data accuracy against iterations of the `ada` function:




Optimal accuracy was attained at 125 iterations, with a steady decline and plateau thereafter. These results were in keeping with our hypothesis, however it was expected that the peak accuracy would occur at a higher iteration.

**ROC Curves**

ROC curves for the linear regression models were created and are displayed here.



The ROC curves shown here illustrate that the performance of the four binary classifiers represented do not vary greatly from one another – all classifiers show an Area Under the Curve (AUC) value of approximately 0.8. However, as shown in the table above, the accuracy of these classifiers vary greatly on the test data with a maximum and minimum of the four listed here being 0.7778 and 0.7638, respectively. While this may seem to be a difference of "only" 0.014 on test data accuracy, this translates to a differences of approximately 80 places in the data challenge rankings. This is a potent example of the importance of seemingly miniscule improvements in prediction models.

### 10-Fold Cross-Validation

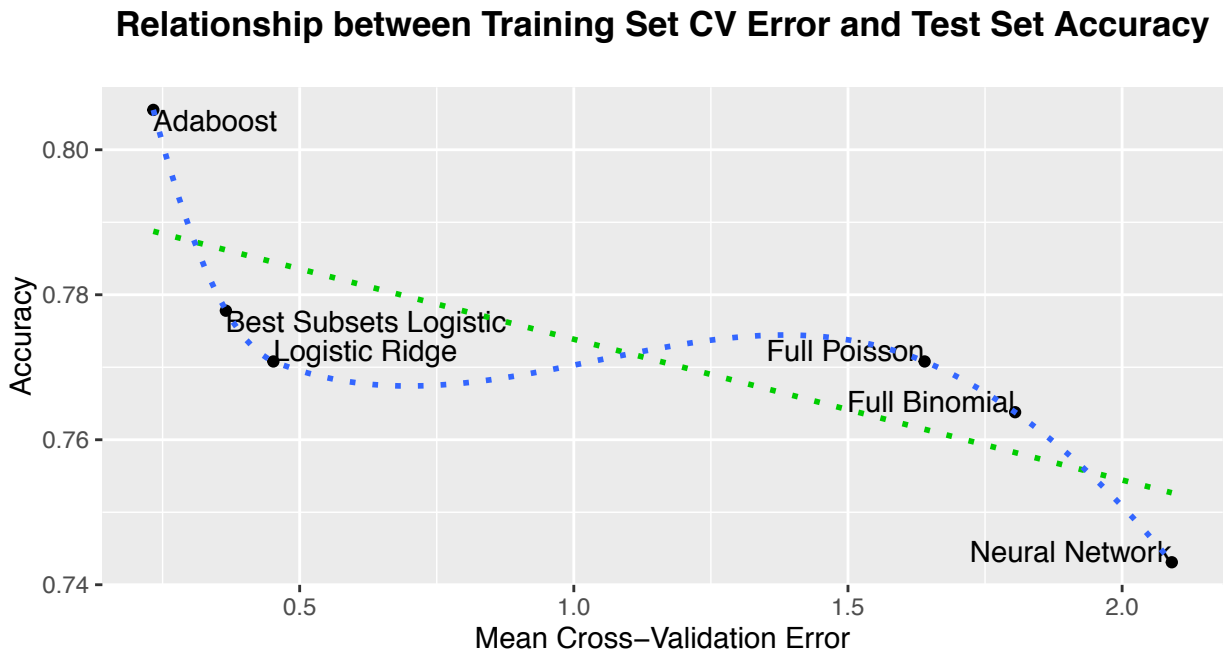
The 10-Fold Cross-Validation approach was used again to determine which model produced the lowest error on training data. This method was chosen as it allows all observations to be used for both training and validation, while also allowing each model to be tested on unseen data. These results are compared in the Discussion section below to the accuracy on test set data. A summary of the mean CV error for each model is shown here:

| Model | Mean CV Error |
|---|---|
| Logistic Ridge Regression Model | 0.2331 |
| Neural Network | 0.3654 |
| Additive (Boosted) Logistic Regression Model | 0.452 |
| Best Subsets Logistic Regression Model | 1.64 |
| Full Logistic Regression Model | 1.805 |
| Full Regression Model, Poisson | 2.091 |

## Discussion

Many data mining methods were explored for this project. It was determined that the additive logistic regression algorithm (`ada` package) produced the best results, with the best subsets logistic regression coming in second. It is believed that the advantage of the `ada` package is due to its ability to combine a number of simpler models. Our hypothesis regarding the manually implemented Adaboost algorithm was incorrect, as it resulted in lower accuracy than the best subsets regression model.

Here the relationship between the Cross Validation error on training data and accuracy on the test data is shown:



While this plot does indicate a general negative relationship between mean CV error and test data accuracy (i.e. as CV error increases, test accuracy decreases), the sample is too small to determine whether or not this data is linear or follows a higher order relationship. That said, the associated regression data for the linear best fit in the plot above is shown here:

| | Estimate | Std. Error | t value | Pr(>\|t\|) |
|---|---|---|---|---|
| **CV** | -0.01942 | 0.007334 | -2.647 | 0.05714 |
| **(Intercept)** | 0.7933 | 0.009799 | 80.96 | 1.395e-07 |

Table 5: Fitting linear model: Accuracy ~ CV

| Observations | Residual Std. Error | $R^2$ | Adjusted $R^2$ |
|---|---|---|---|
| 6 | 0.01368 | 0.6366 | 0.5458 |

This model shows that the slope given by CV, while not quite significant yet (p = 0.057), does have the potential to become significant upon the growth of the sample.

The performance of the best subsets logistic regression model was surprising given its reliance on only three variables (`Married`, `Credit_History` and `SemiUrban`) for prediction of the target variable, `Loan_Status`. Despite this strong performance, the best subsets logistic regression model alone resulted in an overall leaderboard rank of 161. The most significant predictors found here differs from those found by Agbemava et al. (2016). This is most likely due to the population studied – higher income individuals in this project as opposed to lower income cases.

Using our best model, the typical applicant approved for a loan was a married male with no dependents, not self-employed, earning approximately $4000 per month, and requested a loan of about $125000. These results are similar to that seen in the training data set and leans towards higher earners with a lower debt-to-income ratio when compared to the average applicant.

Limitations of this project include the requirement to work with only the data supplied by the data challenge – the given 14 predictor variables. It is most likely that stronger models could be built upon the acquisition of more data. However, the data provided reflects the realistic scenario of having limited data from a loan application, particularly in the situation where an individual may be a new customer of the bank. In subsequent trials and analyses, this project would benefit greatly from further experimentation with more advanced additive logistic regression models and with neural network architectures.

At the time of writing this report, we are currently tied for $8^{th}$ place out of a total of 268 participants. Our results show that even foundational data mining and machine learning techniques can produce powerful prediction models and quite accurate results.

## References


Agbemava, E., Nyarko, I. K., Adade, T. C., & Bediako, A. K. (2016). Logistic Regression Analysis Of Predictors Of Loan Defaults By Customers Of Non-Traditional Banks In Ghana. European Scientific Journal, 12(1).

Conlin, M. (1999). Peer group micro-lending programs in Canada and the United States. Journal of Development Economics, 60(1), 249-269.

Cule, E., & De Iorio, M. (2012). A semi-automatic method to guide the choice of ridge parameter in ridge regression. arXiv preprint arXiv:1205.0686.

Dobbin, K. K., & Simon, R. M. (2011). Optimally splitting cases for training and testing high dimensional classifiers. BMC medical genomics, 4(1), 1.

Günther, F., & Fritsch, S. (2010). neuralnet: Training of neural networks. The R journal, 2(1), 30-38.

"Home Mortgage Disclosure Act." Consumer Financial Protection Bureau: An Official Website of the United States Government. Consumer Financial Protection Bureau, n.d. Web. 17 July 2016. http://www.consumerfinance.gov/data-research/hmda/.

Lusardi, A., & Scheresberg, C. D. B. (2013). Financial literacy and high-cost borrowing in the United States (No. w18969). National Bureau of Economic Research.

Olson, D. L., Delen, D., & Meng, Y. (2012). Comparative analysis of data mining methods for bankruptcy prediction. Decision Support Systems, 52(2), 464-473.

Thomas, L. C. (2000). A survey of credit and behavioural scoring: forecasting financial risk of lending to consumers. International journal of forecasting, 16(2), 149-172.

## Appendix – R Code

```
##### setup #####
library(knitr)
opts_chunk$set(echo = FALSE, warning = FALSE, message = FALSE,
    comment = NA, fig.align = "center", error = TRUE)

library(glmnet)
library(leaps)
library(pROC)
library(car)
library(MASS)
library(ROCR)
library(ggplot2)
library(stringr)
library(dplyr)
library(reshape2)
library(vcd)
library(VIM)
library(pander)
library(tidyr)
library(e1071)


train <- read.csv("https://github.com/dsmilo/DATA621/raw/master/Final-Project/loan_train.csv",
    stringsAsFactors = FALSE)
test = read.csv("https://github.com/dsmilo/DATA621/raw/master/Final-Project/loan_test.csv",
    stringsAsFactors = FALSE)
test2 = test

predict.regsubsets <- function(object, newdata, id, ...) {
    form <- as.formula(object$call[[2]])
    mat <- model.matrix(form, newdata)
    coefi <- coef(object, id = id)
    xvars <- names(coefi)
    mat[, xvars] %*% coefi
}

plotROC <- function(model, ndata, gtruth, name = "") {
    prob <- predict(model, newdata = ndata, type = "response")
    pred <- prediction(prob, gtruth)
    perf <- performance(pred, measure = "tpr", x.measure = "fpr")
    auc <- performance(pred, measure = "auc")
    auc <- auc@y.values[[1]]
    roc.data <- data.frame(fpr = unlist(perf@x.values), tpr = unlist(perf@y.values),
        model = "GLM")
    ggplot(roc.data, aes(x = fpr, ymin = 0, ymax = tpr)) + geom_ribbon(alpha = 0.2) +
        geom_line(aes(y = tpr)) + ggtitle(paste0(name, " ROC Curve w/ AUC=",
        (auc)))
}

fillwithmedian <- function(x) {
```

```
    median_val = median(x, na.rm = TRUE)
    x[is.na(x)] = median_val
    return(x)
}
###################################


##### literature review #####
library(ReadImages)  #Had to install manually
library(grid)
download.file("https://cs231n.github.io/assets/nn1/neural_net2.jpeg",
    "nn1.jpeg")
img <- read.jpeg("nn1.jpeg")
grid.raster(img)
###################################


##### data exploration and manipulation #####
train3 = train
test2 = test

train3$Gender = ifelse(train$Gender == "Male", 1, 0)
train3$Married = ifelse(train$Married == "Yes", 1, 0)
train3$Dependents[train3$Dependents == "3+"] = 3
train3$Dependents = as.numeric(train3$Dependents)
train3$Education = ifelse(train$Education == "Graduate", 1, 0)
train3$Self_Employed = ifelse(train$Self_Employed == "Yes", 1,
    0)

train3$Urban = ifelse(train$Property_Area == "Urban", 1, 0)
train3$SemiUrban = ifelse(train$Property_Area == "Semiurban",
    1, 0)
train3$TotalIncome = train3$ApplicantIncome + train3$CoapplicantIncome
train3$LTI = (train3$LoanAmount * 1000)/train3$ApplicantIncome  #Loan to income ratio

train3$Loan_Status = ifelse(train$Loan_Status == "Y", 1, 0)


test2$Gender = ifelse(test$Gender == "Male", 1, 0)
test2$Married = ifelse(test$Married == "Yes", 1, 0)
test2$Dependents[test2$Dependents == "3+"] = 3
test2$Dependents = as.numeric(test2$Dependents)
test2$Education = ifelse(test$Education == "Graduate", 1, 0)
test2$Self_Employed = ifelse(test$Self_Employed == "Yes", 1,
    0)

test2$Urban = ifelse(test$Property_Area == "Urban", 1, 0)
test2$SemiUrban = ifelse(test$Property_Area == "Semiurban", 1,
    0)
test2$TotalIncome = test2$ApplicantIncome + test2$CoapplicantIncome
test2$LTI = (test2$LoanAmount * 1000)/test2$ApplicantIncome  #Loan to income ratio
```

```r
train4 = train3[complete.cases(train3), ]

train = train3[, -1]
train2 = train4
test = test2


train_var = train
train_var$Property_Area = as.factor(train_var$Property_Area)
var_class <- data.frame(Class = rep(NA, ncol(train_var) - 4),
    Levels = rep(NA, ncol(train_var) - 4), stringsAsFactors = FALSE,
    check.names = FALSE, row.names = names(train_var)[1:(ncol(train_var) -
        4)])
names(var_class) = c("Class", "Levels/Range")

for (i in 1:(ncol(train_var) - 3)) {
    var_class[i - 1, 1] <- class(train_var[, i])
    val = length(levels(train_var[, i]))
    if (val == 0) {
        val = paste0(floor(min(train_var[, i], na.rm = TRUE)),
            " - ", max(train_var[, i], na.rm = TRUE))
    }
    var_class[i - 1, 2] <- val
}
pander(var_class)

# continuous variables
cont_vars <- train_var %>% dplyr::select(c(ApplicantIncome, CoapplicantIncome,
    LoanAmount, Loan_Amount_Term))

melted <- melt(cont_vars)

ggplot(melted, aes(value)) + geom_density(aes(fill = variable,
    col = variable), alpha = 0.5, show.legend = FALSE) + facet_wrap(~variable,
    scales = "free") + scale_y_continuous("", labels = NULL) +
    ggtitle("Distribution of Continuous Predictors \n") + theme(axis.ticks = element_blank())

# discrete variables
train_var$Property_Area <- as.numeric(train_var$Property_Area)
poisson_vars <- train_var %>% dplyr::select(-c(ApplicantIncome,
    CoapplicantIncome, LoanAmount, Loan_Amount_Term, TotalIncome,
    LTI, Urban, SemiUrban))

melted <- melt(poisson_vars)

library(scales)
ggplot(melted, aes(value, y = 6 * ..count../sum(..count..))) +
    geom_bar(aes(fill = variable, col = variable), alpha = 0.5,
        show.legend = FALSE) + facet_wrap(~variable, scales = "free_x") +
    scale_y_continuous("", labels = percent) + ggtitle("Distribution of Discrete Predictors \n")

# created variables
```

```
created_vars <- train_var %>% dplyr::select(c(TotalIncome, LTI))

melted <- melt(created_vars)

ggplot(melted, aes(value)) + geom_density(aes(fill = variable,
    col = variable), alpha = 0.5, show.legend = FALSE) + facet_wrap(~variable,
    scales = "free") + scale_y_continuous("", labels = NULL) +
    ggtitle("Distribution of Created Predictors \n") + theme(axis.ticks = element_blank())

# missing values
aggr(train_var, plot = TRUE, sortVars = FALSE, numbers = TRUE,
    cex.lab = TRUE)
###################################


##### Model Creation ##### Full Model
glmflagfull <- glm(Loan_Status ~ . - Property_Area, data = train,
    family = poisson, control = list(maxit = 100))
summary(glmflagfull)
vif(glmflagfull)  #Check for collinearity, VIF > 10
predglmflagfull <- data.frame(class = train$Loan_Status, logit = predict(glmflagfull,
    train))

plotROC(model = glmflagfull, ndata = train, gtruth = train$Loan_Status,
    name = "Full Model (Poisson)")

a = predict(glmflagfull, test, type = "response")
testdf = data.frame(Loan_ID = test$Loan_ID, Loan_Status = a)
testdf$Loan_Status[is.na(testdf$Loan_Status)] = median(testdf$Loan_Status,
    na.rm = TRUE)
testdf$Loan_Status = ifelse(testdf$Loan_Status > 0.5, "Y", "N")

write.csv(testdf, paste0("finalresults-", as.numeric(Sys.time()),
    ".csv"))
###############

# Best Subsets
regfit.full <- regsubsets(Loan_Status ~ . - Property_Area - Loan_ID,
    data = train3, nvmax = 17)
summary(regfit.full)
par(mar = c(1, 1, 1, 1))
par(mfrow = c(1, 2))
plot(regfit.full, scale = "bic", main = "Predictor Variables vs. BIC")
reg.summary <- summary(regfit.full)
reg.summary$bic
plot(reg.summary$bic, xlab = "Number of Predictors", ylab = "BIC",
    type = "l", main = "Best Subset Selection Using BIC")
minbic <- which.min(reg.summary$bic)
points(minbic, reg.summary$bic[minbic], col = "brown", cex = 2,
    pch = 20)
coef(regfit.full, minbic)
var_names = names(coef(regfit.full, minbic))[2:length(names(coef(regfit.full,
```

```
    minbic)))]
length(var_names)

Model_toEval = paste0("glm(Loan_Status ~ ", paste(var_names,
    collapse = " + "), ", data = train, family = binomial(link='logit'))")

bestsubset21 = eval(parse(text = Model_toEval))

summary(bestsubset21)

plotROC(model = bestsubset21, ndata = train, gtruth = train$Loan_Status,
    name = "Best Subsets")

a = predict(bestsubset21, test, type = "response")
testdf = data.frame(Loan_ID = test$Loan_ID, Loan_Status = a)
testdf$Loan_Status[is.na(testdf$Loan_Status)] = median(testdf$Loan_Status,
    na.rm = TRUE)
testdf$Loan_Status = ifelse(testdf$Loan_Status > 0.5, "Y", "N")
###############

# Logistic Ridge
library(ridge)  #installed from source
train$Loan_Status = as.numeric(as.character(train$Loan_Status))
train_lr <- data.frame(lapply(train[, -11], fillwithmedian))
train_lr$Property_Area = train$Property_Area

lr2 = logisticRidge(Loan_Status ~ . - Property_Area, data = train_lr)

plotROC(model = lr2, ndata = train, gtruth = train$Loan_Status,
    name = "Logistic Ridge")

a = predict(lr2, test2, type = "response")
testdf = data.frame(Loan_ID = test$Loan_ID, Loan_Status = a)
testdf$Loan_Status[is.na(testdf$Loan_Status)] = median(testdf$Loan_Status,
    na.rm = TRUE)
testdf$Loan_Status = ifelse(testdf$Loan_Status > 0.5, "Y", "N")
write.csv(testdf, paste0("finalresults-", as.numeric(Sys.time()),
    ".csv"))
###############

# Full model binomial
glmflagfull <- glm(Loan_Status ~ . - Property_Area, data = train,
    family = binomial, control = list(maxit = 100))
summary(glmflagfull)
vif(glmflagfull)  #Check for collinearity, VIF > 10
predglmflagfull <- data.frame(class = train$Loan_Status, logit = predict(glmflagfull,
    train))

plotROC(model = glmflagfull, ndata = train, gtruth = train$Loan_Status,
    name = "Full Model Binomial")

a = predict(glmflagfull, test, type = "response")
```

```
testdf = data.frame(Loan_ID = test$Loan_ID, Loan_Status = a)
testdf$Loan_Status[is.na(testdf$Loan_Status)] = median(testdf$Loan_Status,
    na.rm = TRUE)
testdf$Loan_Status = ifelse(testdf$Loan_Status > 0.5, "Y", "N")
write.csv(testdf, paste0("finalresults-", as.numeric(Sys.time()),
    ".csv"))
###############

# Neural Net Determination of nodes for neuralnet
k = 10
set.seed(1306)
folds = sample(1:k, nrow(train), replace = TRUE)
cv.errors1 = matrix(NA, k, 10, dimnames = list(NULL, paste(1:10)))
cv.errors2 = matrix(NA, k, 10, dimnames = list(NULL, paste(1:10)))
cv.errors3 = matrix(NA, k, 10, dimnames = list(NULL, paste(1:10)))
cv.errors4 = matrix(NA, k, 10, dimnames = list(NULL, paste(1:10)))
cv.errors5 = matrix(NA, k, 10, dimnames = list(NULL, paste(1:10)))
cv.errors6 = matrix(NA, k, 10, dimnames = list(NULL, paste(1:10)))
cv.errors7 = matrix(NA, k, 10, dimnames = list(NULL, paste(1:10)))
cv.errors8 = matrix(NA, k, 10, dimnames = list(NULL, paste(1:10)))

train_2 <- data.frame(lapply(train[, -11], fillwithmedian))
train_2$Property_Area = train$Property_Area
scaled = data.frame(lapply(scaled, fillwithmedian))

num_layers = 0

f1 <- as.formula(paste("Loan_Status ~", paste(n[!n %in% "Loan_Status"],
    collapse = " + ")))
for (j in 1:k) {

    num_layers = num_layers + 5
    print(num_layers)
    dat = scaled[folds != j, ]
    dat = dat[complete.cases(dat), ]
    model5 <- neuralnet(f1, data = dat, hidden = c(num_layers),
        linear.output = TRUE, stepmax = 1e+07)

    for (i in 1:10) {
        print(i)
        f = train_2[folds == j, ]
        f = f[complete.cases(f), ]
        dat = scaled[folds == j, ]
        dat2 = dat[complete.cases(dat), -11]
        pred5 = neuralnet::compute(model5, dat2)
        netresult = pred5$net.result[, 1]
        sc = scaled$Loan_Status[folds == j]
        sc = sc[complete.cases(sc)]
        cv.errors5[j, i] = mean((sc - netresult)^2, na.rm = TRUE)
    }

}
```

```
cv_means = rowMeans(cv.errors5)
node_counts = seq(5, 50, 5)
nn_df = data.frame(Nodes = node_counts, CV_Error = cv_means)
names(nn_df) = c("Nodes", "Mean CV Error")
write.csv(nn_df, "~/Downloads/nn_df.csv", row.names = FALSE)

nn_df = read.csv("https://raw.githubusercontent.com/aadikalloo/AadiMSDA/master/IS621-Data-Mining/nn_d
plot(x = nn_df$Nodes, y = nn_df$Mean.CV.Error, xlab = "Nodes",
    ylab = "Mean CV Error", main = "Determination of Number of Nodes for neuralnet",
    pch = ifelse(nn_df$Nodes == 10, 19, 1))
##########

## Neural network
train4 = train3
train4 = train4[complete.cases(train4), ]
train4 = train4[, -c(1, 12)]
maxs <- apply(train4, 2, max)
mins <- apply(train4, 2, min)
scaled <- as.data.frame(scale(train4, center = mins, scale = maxs -
    mins))
n <- names(scaled)
f <- as.formula(paste("Loan_Status ~", paste(n[!n %in% "Loan_Status"],
    collapse = " + ")))
nn = neuralnet(f, data = scaled, hidden = c(10), linear.output = TRUE,
    stepmax = 1e+06)
plot(nn)

max2 <- function(x) {
    return(max(x, na.rm = TRUE))
}

min2 <- function(x) {
    return(min(x, na.rm = TRUE))
}

test <- data.frame(lapply(test, fillwithmedian))
test = test2
test = test[, -c(1, 12)]
test <- data.frame(lapply(test, fillwithmedian))

maxs.t <- apply(test, 2, max2)
mins.t <- apply(test, 2, min2)
scaled.t <- as.data.frame(scale(test, center = mins.t, scale = maxs.t -
    mins.t))
scaled.t = as.data.frame(lapply(scaled.t, fillwithmedian))
pr.test <- neuralnet::compute(nn, scaled.t)
pr.t_ <- pr.test$net.result * (max(train4$Loan_Status) - min(train4$Loan_Status)) +
    min(train4$Loan_Status)
pr_nn.t = ifelse(pr.t_ > 0.5, "Y", "N")

test1 = read.csv("https://github.com/dsmilo/DATA621/raw/master/Final-Project/loan_test.csv",
    stringsAsFactors = FALSE)
```

```
df = data.frame(Loan_ID = test1$Loan_ID, Loan_Status = as.data.frame(as.vector(pr_nn.t)))
names(df) = c("Loan_ID", "Loan_Status")
sum(is.na(df[, 2]))  #hopefully 0
write.csv(df, paste0("nn-", as.numeric(Sys.time()), ".csv"),
    row.names = FALSE)

download.file("https://github.com/aadikalloo/AadiMSDA/raw/master/IS621-Data-Mining/nn2.jpg",
    "nn2.jpeg")
img <- read.jpeg("nn2.jpeg")
grid.raster(img)
##########


# ada package analysis
model1 = list()

for (i in 1:20) {
    iters = 25 + 25 * i
    model1[[i]] = ada(formula = Loan_Status ~ ., data = train_x1,
        iter = iters, loss = "logistic")
    results = as.numeric(as.character(predict(model1[[i]], test)))
    results = ifelse(results == 1, "Y", "N")
    testdf = data.frame(Loan_ID = test2$Loan_ID, Loan_Status = results)
    write.csv(testdf, paste0("finalresults-", as.numeric(Sys.time()),
        " - ", iters, ".csv"), row.names = FALSE)
}

ada_results = read.csv("https://raw.githubusercontent.com/aadikalloo/AadiMSDA/master/IS621-Data-Minir
    header = FALSE)
names(ada_results) = c("Iterations", "Accuracy")
###############


# Adaboost manual implementation
wts = list()

models = list()
alpha = list()
error_rate = list()
final = 0
predictions = list()
total = 12

train_x <- data.frame(lapply(train[-c(1, 2), -11], fillwithmedian))
test_x = data.frame(lapply(test2[, -c(1, 12)], fillwithmedian))

k = total
set.seed(1306)
folds = sample(1:k, nrow(train_x), replace = TRUE)
folds = rep(1:12, nrow(train_x)/12)

wts[[1]] = rep(1/length(train_x1[, 1]), length(train_x1[, 1]))
```

```
Y_hat_run = list()
Y_hat_run[[1]] = rep(1, length(train_x1[, 1]))

for (i in 1:total) {
    train_x1 = train_x

    models[[i]] = glm(Loan_Status ~ ., data = train_x1, family = binomial,
        weights = wts[[i]])
    Y_hat = predict(models[[i]], train_x1, type = "response")
    Y_hat_run = Y_hat_run[[i]] * Y_hat
    Y_hat = ifelse(Y_hat > 0.5, 1, 0)
    error_rate[[i]] = sum(wts[[i]] * (train_x1$Loan_Status !=
        Y_hat))/length(Y_hat)
    alpha[[i]] = (1/2) * log((1 - error_rate[[i]])/error_rate[[i]])
    loan_status = ifelse(train_x1$Loan_Status > 0, 1, -1)
    Y_hat = ifelse(Y_hat == 1, 1, -1)
    wts[[i + 1]] = wts[[i]] * exp(-alpha[[i]] * loan_status *
        Y_hat)
    wts[[i + 1]] = wts[[i + 1]]/sum(wts[[i + 1]])
}

for (i in 1:total) {
    predictions[[i]] = predict(models[[i]], test_x, type = "response")
    predictions[[i]] = alpha[[i]] * predictions[[i]]
}

toEval = "predictions[[1]]"

for (i in 2:4) {
    toEval = paste0(toEval, " * predictions[[", i, "]]")
}

prop = table(train$Loan_Status)[[1]]/table(train$Loan_Status)[[2]]
new = Y_hat_run[order(Y_hat_run)]
pred2 = ifelse((rank(Y_hat_run)) > prop * length(test[, 1]),
    1, 0)

results = eval(parse(text = toEval))
results = ifelse(results < error_rate[[1]], "N", "Y")

testdf = data.frame(Loan_ID = test2$Loan_ID, Loan_Status = results)

write.csv(testdf, paste0("finalresults-", as.numeric(Sys.time()),
    ".csv"), row.names = FALSE)
###############


##### Results #####
data_results = data.frame(Model = c("ada package", "Best Subsets",
    "Logistic Ridge", "Full Model, Binomial", "Full Model, Poisson",
    "Neural Network (neuralnet package)", "Manually implemented adaboost algorithm"),
    Accuracy = c("0.8055", "0.7778", "0.7708", "0.7638", "0.7708",
```

```
        "0.7431", "0.7153"), stringsAsFactors = FALSE)
data_results2 = data_results[order(data_results$Accuracy, decreasing = TRUE),
    ]
kable(data_results2)

ada_results = read.csv("https://raw.githubusercontent.com/aadikalloo/AadiMSDA/master/IS621-Data-Minin
    header = FALSE)
names(ada_results) = c("Iterations", "Accuracy")
plot(x = ada_results$Iterations, y = ada_results$Accuracy, xlab = "Iterations",
    ylab = "Accuracy", ylim = c(0.72, 0.82), main = "Effect of number of iteratons on test data accur
    pch = ifelse(ada_results$Accuracy == 0.8055, 19, 1))
lines(smooth.spline(x = ada_results$Iterations, y = ada_results$Accuracy))

download.file("https://raw.githubusercontent.com/aadikalloo/AadiMSDA/master/IS621-Data-Mining/roc2.jp
    "roc.jpeg")
img <- read.jpeg("roc.jpeg")
grid.raster(img)

# 10-fold CV
k = 10
set.seed(1306)
folds = sample(1:k, nrow(train), replace = TRUE)
cv.errors1 = matrix(NA, k, 10, dimnames = list(NULL, paste(1:10)))
cv.errors2 = matrix(NA, k, 10, dimnames = list(NULL, paste(1:10)))
cv.errors3 = matrix(NA, k, 10, dimnames = list(NULL, paste(1:10)))
cv.errors4 = matrix(NA, k, 10, dimnames = list(NULL, paste(1:10)))
cv.errors5 = matrix(NA, k, 10, dimnames = list(NULL, paste(1:10)))
cv.errors6 = matrix(NA, k, 10, dimnames = list(NULL, paste(1:10)))
cv.errors7 = matrix(NA, k, 10, dimnames = list(NULL, paste(1:10)))
cv.errors8 = matrix(NA, k, 10, dimnames = list(NULL, paste(1:10)))

train_2 <- data.frame(lapply(train[, -11], fillwithmedian))
train_2$Property_Area = train$Property_Area
scaled = data.frame(lapply(scaled, fillwithmedian))

f1 <- as.formula(paste("Loan_Status ~", paste(n[!n %in% "Loan_Status"],
    collapse = " + ")))
for (j in 1:k) {
    print(j)

    model1 <- glm(Loan_Status ~ . - Property_Area, data = train_2[folds !=
        j, ], family = poisson, control = list(maxit = 100))
    model2 <- glm(formula = Loan_Status ~ Married + Credit_History +
        SemiUrban, family = binomial(link = "logit"), data = train_2[folds !=
        j, ])
    model3 <- logisticRidge(Loan_Status ~ . - Property_Area,
        data = train_2[folds != j, ])
    model4 <- glm(Loan_Status ~ . - Property_Area, data = train_2[folds !=
        j, ], family = binomial, control = list(maxit = 100))
    dat = scaled[folds != j, ]
    dat = dat[complete.cases(dat), ]
    model5 <- neuralnet(f1, data = dat, hidden = c(5, 3), linear.output = TRUE,
```

```
        stepmax = 1e+07)
    model6 <- ada(formula = Loan_Status ~ ., data = train_x1[folds !=
        j, ], iter = 125, loss = "logistic")

    for (i in 1:10) {
        print(i)
        f = train_2[folds == j, ]
        f = f[complete.cases(f), ]

        pred1 = predict(model1, f, id = i)
        cv.errors1[j, i] = mean((train_2$Loan_Status[folds ==
            j] - pred1)^2, na.rm = TRUE)

        pred2 = predict(model2, f, id = i)
        cv.errors2[j, i] = mean((train_2$Loan_Status[folds ==
            j] - pred2)^2, na.rm = TRUE)

        pred3 = predict(model3, f, id = i)
        cv.errors3[j, i] = mean((train_2$Loan_Status[folds ==
            j] - pred3)^2, na.rm = TRUE)

        pred4 = predict(model4, f, id = i)
        cv.errors4[j, i] = mean((train_2$Loan_Status[folds ==
            j] - pred4)^2, na.rm = TRUE)

        dat = scaled[folds == j, ]
        dat = dat[complete.cases(dat), -11]
        pred5 = neuralnet::compute(model5, dat2)
        netresult = pred5$net.result[, 1]
        sc = scaled$Loan_Status[folds == j]
        sc = sc[complete.cases(sc)]
        cv.errors5[j, i] = mean((sc - netresult)^2, na.rm = TRUE)

        pred6 = predict(model6, train_x1[folds == j, ], id = i,
            type = "probs")
        cv.errors6[j, i] = mean((train_2$Loan_Status[folds ==
            j] - pred6)^2, na.rm = TRUE)

    }

}

mean.cv.errors1 <- apply(cv.errors1, 2, mean)
mean.cv.errors2 <- apply(cv.errors2, 2, mean)
mean.cv.errors3 <- apply(cv.errors3, 2, mean)
mean.cv.errors4 <- apply(cv.errors4, 2, mean)
mean.cv.errors5 <- apply(cv.errors5, 2, mean)
mean.cv.errors6 <- apply(cv.errors6, 2, mean)

all.cv.error = data.frame(mean(mean.cv.errors1), mean(mean.cv.errors2),
    mean(mean.cv.errors3), mean(mean.cv.errors4), mean(mean.cv.errors5),
    mean(mean.cv.errors6))
```

```
names(all.cv.error) = c("Full Regression Model, Poisson", "Best Subsets Logistic Regression Model",
    "Logistic Ridge Regression Model", "Full Logistic Regression Model",
    "Neural Network", "Additive (Boosted) Logistic Regression Model")
all.cv.error = t(all.cv.error)
names(all.cv.error) = c("Model", "Mean CV Error")

mean_cv_results = read.csv("https://github.com/aadikalloo/AadiMSDA/raw/master/IS621-Data-Mining/allcv
    check.names = FALSE)
mean_cv = mean_cv_results[order(mean_cv_results$"Mean CV Error"),
    ]
row.names(mean_cv) = NULL
pander(mean_cv)
###################################


##### discussion #####
acc_cv <- data.frame(Accuracy = as.numeric(data_results2$Accuracy[1:6]),
    CV = mean_cv$"Mean CV Error")
rownames(acc_cv) <- c("Adaboost", "Best Subsets Logistic", "Logistic Ridge",
    "Full Poisson", "Full Binomial", "Neural Network")

ggplot(acc_cv, aes(x = CV, y = Accuracy)) + geom_point() + geom_text(aes(label = rownames(acc_cv)),
    vjust = "inward", hjust = "inward", check_overlap = TRUE) +
    geom_smooth(alpha = 0.25, method = glm, fill = 3, col = 3,
        se = FALSE, lty = 3) + scale_x_continuous("Mean Cross-Validation Error") +
    ggtitle("Relationship between Training Set CV Error and Test Set Accuracy\n") +
    theme(plot.title = element_text(lineheight = 0.8, face = "bold")) +
    geom_smooth(alpha = 0.25, method = "loess", se = FALSE, lwd = 1,
        lty = 3)

alm = lm(Accuracy ~ CV, data = acc_cv)
pander(summary(alm))
###################################
```

# 1 Data Exploration

The dataset of interest contains information about customers of an auto insurance company. The dataset has 8161 rows (each representing a customer) and 25 variables. There are 23 predictor variables and 2 response variables: `TARGET_FLAG`, a binary categorical variable representing whether each customer has been in an accident; and `TARGET_AMT`, a numerical variable indicating the cost of a crash that a customer was in. The class of variables read in from the dataset is presented below:

| | Class | Levels |
|---|---|---|
| **TARGET_FLAG** | integer | - |
| **TARGET_AMT** | numeric | - |
| **KIDSDRIV** | integer | - |
| **AGE** | integer | - |
| **HOMEKIDS** | integer | - |
| **YOJ** | integer | - |
| **INCOME** | factor | 6613 |
| **PARENT1** | factor | 2 |
| **HOME_VAL** | factor | 5107 |
| **MSTATUS** | factor | 2 |
| **SEX** | factor | 2 |
| **EDUCATION** | factor | 5 |
| **JOB** | factor | 9 |
| **TRAVTIME** | integer | - |
| **CAR_USE** | factor | 2 |
| **BLUEBOOK** | factor | 2789 |
| **TIF** | integer | - |
| **CAR_TYPE** | factor | 6 |
| **RED_CAR** | factor | 2 |
| **OLDCLAIM** | factor | 2857 |
| **CLM_FREQ** | integer | - |
| **REVOKED** | factor | 2 |
| **MVR_PTS** | integer | - |
| **CAR_AGE** | integer | - |
| **URBANICITY** | factor | 2 |

The very high number of levels for four of the variables (`INCOME`, `HOME_VAL`, `BLUEBOOK`, and `OLDCLAIM`) indicates that these variables are not in fact factors; investigation of the dataset indicates that these are dollar values interpreted as strings due to the presence of dollar signs and commas. The numerical values are extracted for these variables.

Additionally, there are 7 variables with only two levels. These are recast as binary variables as follows:

- `PARENT1`, `MSTATUS`, `RED_CAR`, and `REVOKED`: using 1 to indicate Yes
- `SEX`: using 1 to indicate Male
- `CAR_USE`: using 1 to indicate Commercial
- `URBANICITY`: using 1 to indicate Highly Urban/ Urban

Finally, there are three categorical variables – factors with more than two levels. Dummy variables are created for each of these, as follows:

- `EDUCATION`: 5 dummy variables
- `CAR_TYPE`: 6 dummy variables
- `JOB`: 8 dummy variables

A summary of each variable is presented below:

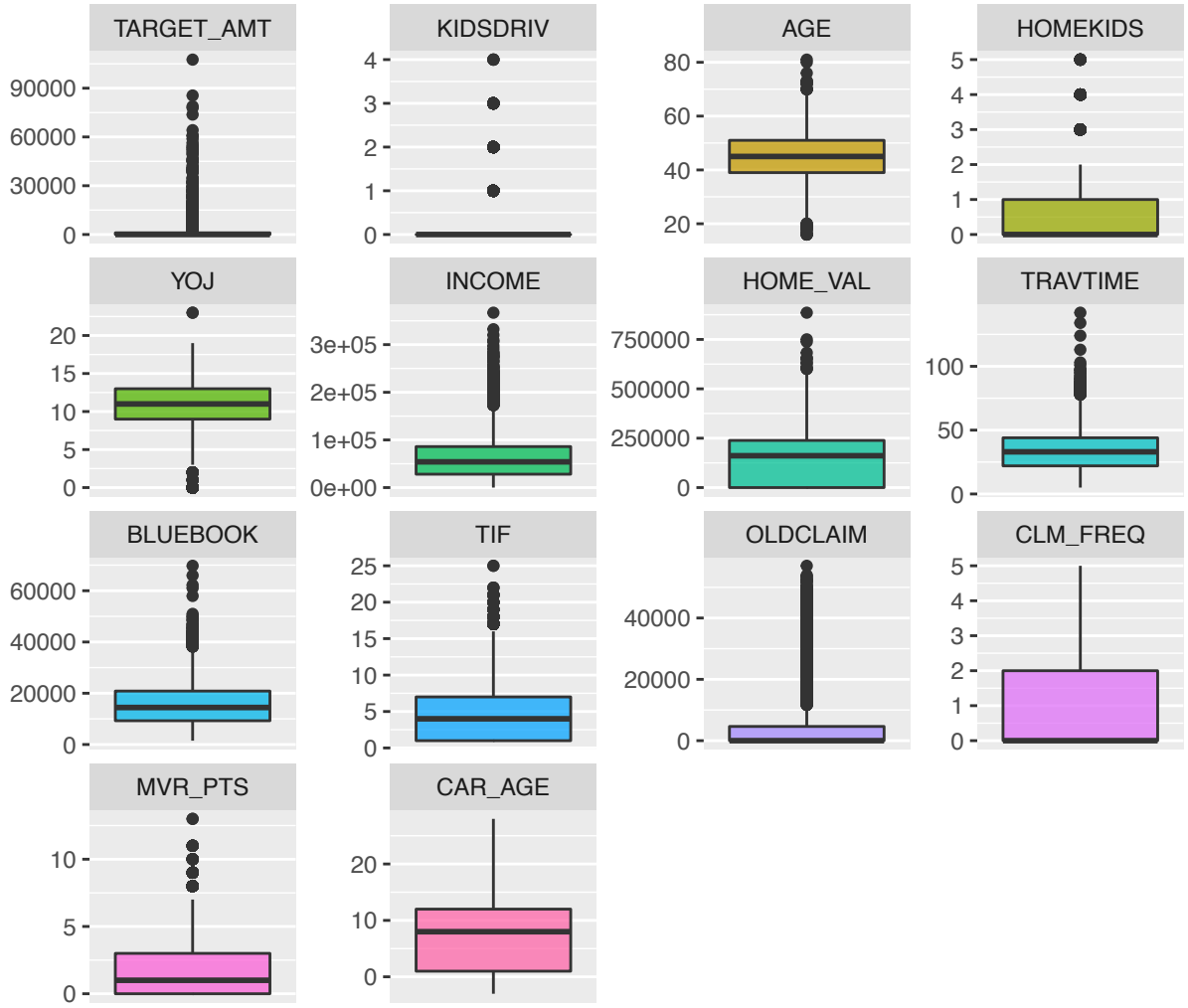
| | MEAN | MEDIAN | IQR | SKEW | $r_{FLAG}$ | $r_{AMT}$ | NAs |
|---|---|---|---|---|---|---|---|
| **TARGET_FLAG** | 0.26 | 0 | 1 | 1.07 | 1 | 0.54 | 0 |
| **TARGET_AMT** | 1504 | 0 | 1036 | 8.71 | 0.54 | 1 | 0 |
| **KIDSDRIV** | 0.17 | 0 | 0 | 3.35 | 0.09 | 0.05 | 0 |
| **AGE** | 44.79 | 45 | 12 | -0.03 | -0.11 | -0.05 | 6 |
| **HOMEKIDS** | 0.72 | 0 | 1 | 1.34 | 0.11 | 0.06 | 0 |
| **YOJ** | 10.5 | 11 | 4 | -1.2 | -0.07 | -0.02 | 454 |
| **INCOME** | 61898 | 54028 | 57889 | 1.19 | -0.14 | -0.06 | 445 |
| **PARENT1** | 0.13 | 0 | 0 | 2.17 | 0.16 | 0.1 | 0 |
| **HOME_VAL** | 154867 | 161160 | 238724 | 0.49 | -0.18 | -0.09 | 464 |
| **MSTATUS** | 0.6 | 1 | 1 | -0.41 | -0.13 | -0.1 | 0 |
| **SEX** | 0.46 | 0 | 1 | 0.14 | -0.02 | 0.01 | 0 |
| **TRAVTIME** | 33.49 | 33 | 22 | 0.45 | 0.05 | 0.03 | 0 |
| **CAR_USE** | 0.37 | 0 | 1 | 0.53 | 0.14 | 0.1 | 0 |
| **BLUEBOOK** | 15710 | 14440 | 11570 | 0.79 | -0.11 | 0 | 0 |
| **TIF** | 5.35 | 4 | 6 | 0.89 | -0.08 | -0.04 | 0 |
| **RED_CAR** | 0.29 | 0 | 1 | 0.92 | -0.02 | 0 | 0 |
| **OLDCLAIM** | 4037 | 0 | 4636 | 3.12 | 0.14 | 0.08 | 0 |
| **CLM_FREQ** | 0.8 | 0 | 2 | 1.21 | 0.22 | 0.12 | 0 |
| **REVOKED** | 0.12 | 0 | 0 | 2.3 | 0.15 | 0.06 | 0 |
| **MVR_PTS** | 1.7 | 1 | 3 | 1.35 | 0.23 | 0.14 | 0 |
| **CAR_AGE** | 8.33 | 8 | 11 | 0.28 | -0.11 | -0.06 | 510 |
| **URBANICITY** | 0.8 | 1 | 0 | -1.46 | 0.22 | 0.12 | 0 |
| **HSDropout** | 0.15 | 0 | 0 | 1.99 | 0.06 | 0.04 | 0 |
| **HS** | 0.29 | 0 | 1 | 0.95 | 0.11 | 0.04 | 0 |
| **Bachelors** | 0.27 | 0 | 1 | 1.01 | -0.05 | -0.02 | 0 |
| **Masters** | 0.2 | 0 | 0 | 1.48 | -0.09 | -0.05 | 0 |
| **PhD** | 0.09 | 0 | 0 | 2.88 | -0.06 | -0.02 | 0 |
| **Minivan** | 0.26 | 0 | 1 | 1.08 | -0.14 | -0.08 | 0 |
| **Panel_Truck** | 0.08 | 0 | 0 | 3.03 | 0 | 0.04 | 0 |
| **Pickup** | 0.17 | 0 | 0 | 1.75 | 0.05 | 0.02 | 0 |
| **Sports_Car** | 0.11 | 0 | 0 | 2.47 | 0.06 | 0.03 | 0 |
| **Van** | 0.09 | 0 | 0 | 2.82 | 0 | 0.01 | 0 |
| **SUV** | 0.28 | 0 | 1 | 0.97 | 0.05 | 0.01 | 0 |
| **Blank_Job** | 0.06 | 0 | 0 | 3.55 | -0.01 | 0.01 | 0 |
| **Professional** | 0.14 | 0 | 0 | 2.11 | -0.04 | 0 | 0 |
| **Blue_Collar** | 0.22 | 0 | 0 | 1.33 | 0.1 | 0.07 | 0 |
| **Clerical** | 0.16 | 0 | 0 | 1.9 | 0.04 | 0 | 0 |
| **Doctor** | 0.03 | 0 | 0 | 5.49 | -0.05 | -0.03 | 0 |
| **Lawyer** | 0.1 | 0 | 0 | 2.62 | -0.06 | -0.03 | 0 |
| **Manager** | 0.12 | 0 | 0 | 2.32 | -0.12 | -0.07 | 0 |
| **Home_Maker** | 0.08 | 0 | 0 | 3.13 | 0.01 | 0 | 0 |
| **Student** | 0.09 | 0 | 0 | 2.92 | 0.07 | 0.02 | 0 |

From the table above, it is clear that there are four variables with missing values, with the proportion of values missing ranging from less than $< 0.1\%$ to roughly 6.2%; these missing values will need to either be imputed or excluded from the dataset before modeling. The variables exhibit varying levels of skewness, with a few extreme values.

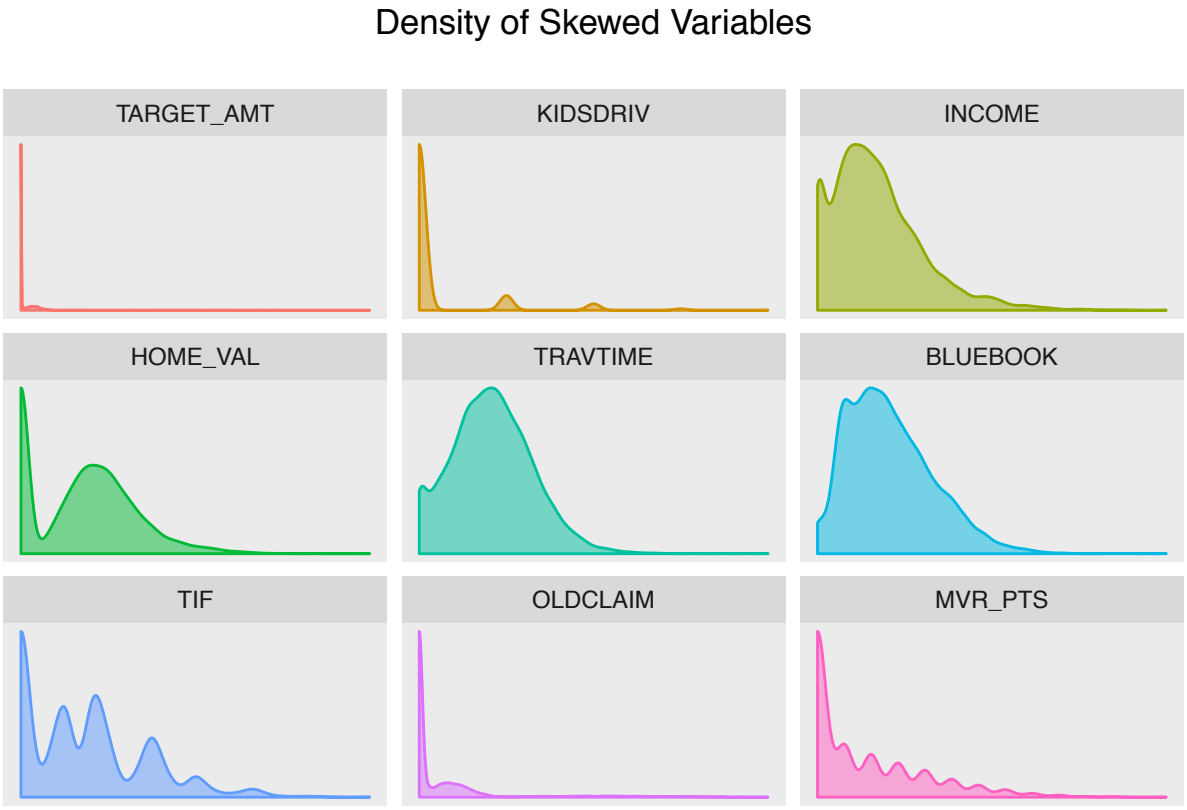
The large number of binary variables in the dataset makes graphical visualization of the distribution of all variables not particularly useful. The proportion of binary variables having a value of 0 or 1 is presented in the table below:

| | 0 | 1 |
|---|---|---|
| **TARGET_FLAG** | 0.74 | 0.26 |
| **PARENT1** | 0.87 | 0.13 |
| **MSTATUS** | 0.4 | 0.6 |
| **SEX** | 0.54 | 0.46 |
| **CAR_USE** | 0.63 | 0.37 |
| **RED_CAR** | 0.71 | 0.29 |
| **REVOKED** | 0.88 | 0.12 |
| **URBANICITY** | 0.2 | 0.8 |
| **HSDropout** | 0.85 | 0.15 |
| **HS** | 0.71 | 0.29 |
| **Bachelors** | 0.73 | 0.27 |
| **Masters** | 0.8 | 0.2 |
| **PhD** | 0.91 | 0.09 |
| **Minivan** | 0.74 | 0.26 |
| **Panel_Truck** | 0.92 | 0.08 |
| **Pickup** | 0.83 | 0.17 |
| **Sports_Car** | 0.89 | 0.11 |
| **Van** | 0.91 | 0.09 |
| **SUV** | 0.72 | 0.28 |
| **Blank_Job** | 0.94 | 0.06 |
| **Professional** | 0.86 | 0.14 |
| **Blue_Collar** | 0.78 | 0.22 |
| **Clerical** | 0.84 | 0.16 |
| **Doctor** | 0.97 | 0.03 |
| **Lawyer** | 0.9 | 0.1 |
| **Manager** | 0.88 | 0.12 |
| **Home_Maker** | 0.92 | 0.08 |
| **Student** | 0.91 | 0.09 |

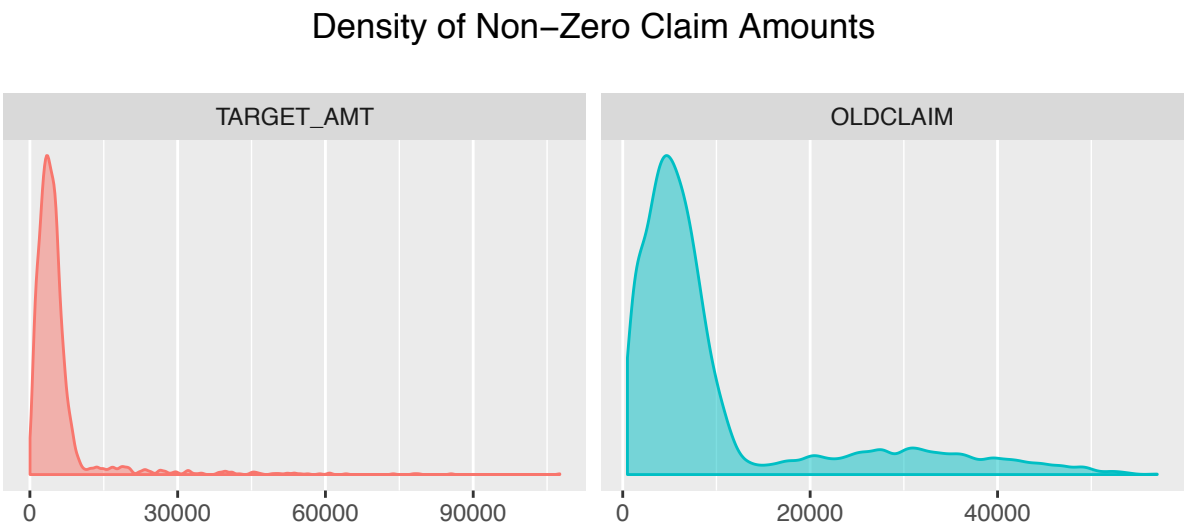
The remaining variables are vizualized below in boxplots:



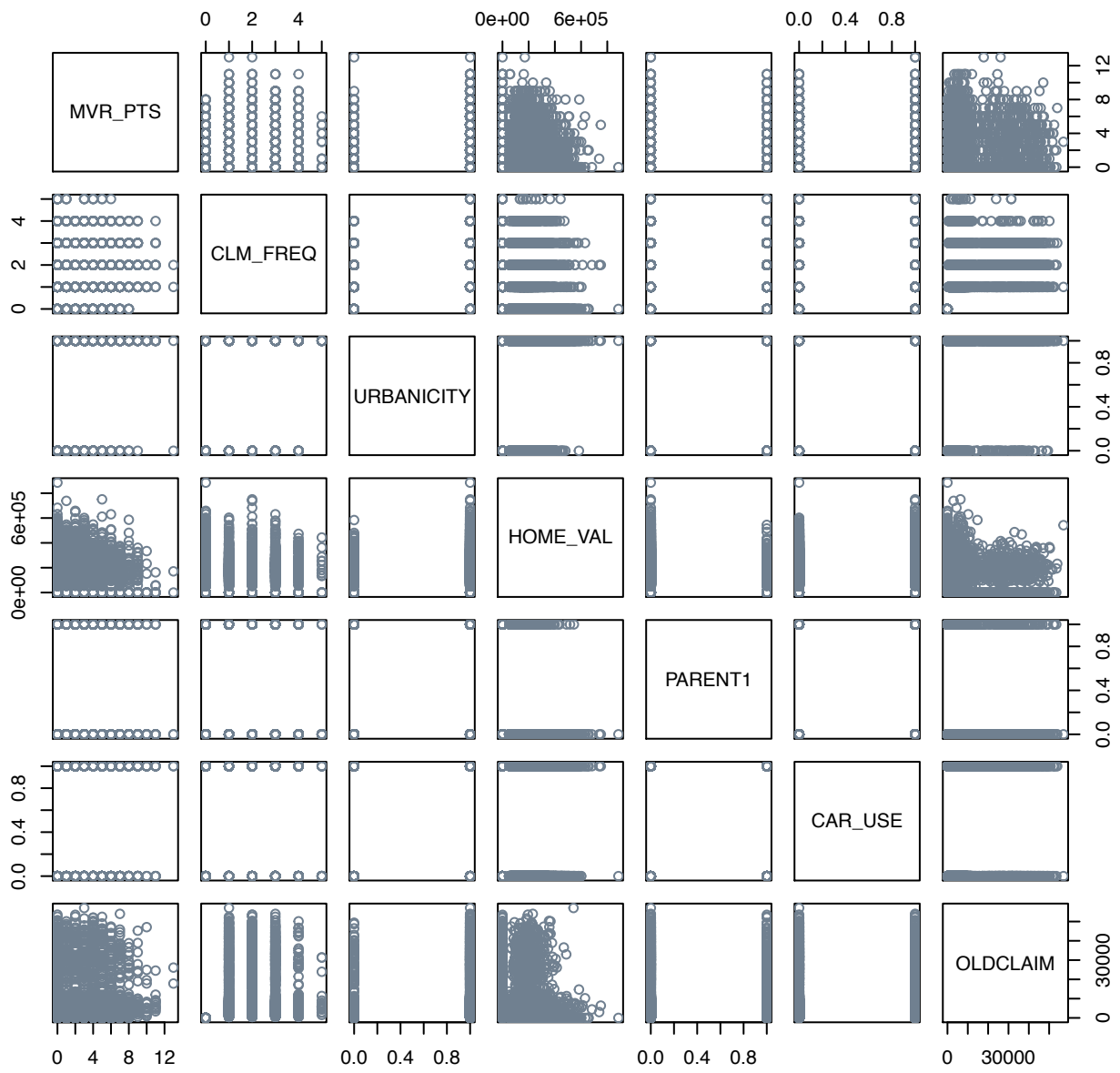

The boxplots illustrate the high skewness of the distributions of the predictors `KIDSDRIV`, `INCOME`, `HOME_VAL`, `TRAVTIME`, `BLUEBOOK`, `TIF`, `OLDCLAIM` and `MVR_PTS`. The target `TARGET_AMT` is also highly skewed – this makes sense, as this value is 0 for any customers without claims. Density plots of these variables are presented below:




Since `TARGET_AMT` and `OLDCLAIM` have such high concentrations at values of zero, separate plots for these two variables are created with values of zero removed:

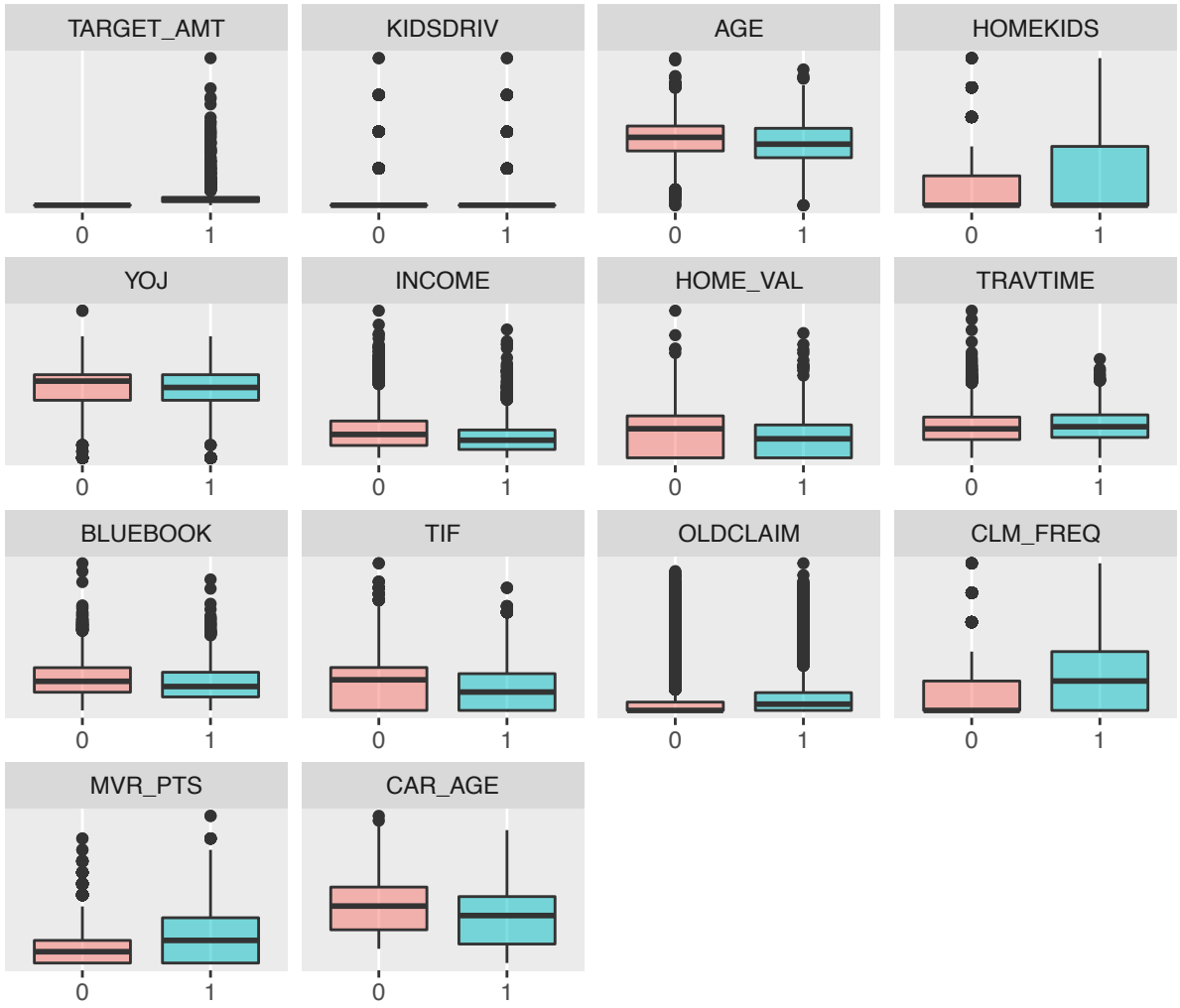



The 8 predictors with the highest correlation to `TARGET_FLAG` and the 8 predictors with the highest correlation to `TARGET_AMT` share 7 predictors. The correlation between these variables is investigated and plotted below:




There appears to be evidence of possible multicollinearity between `HOME_VAL` and `OLDCLAIM`.

Finally, boxplots are also prepared for non-binary variables split by `TARGET_FLAG`:




Interestingly, there are only a few variables with immediately visible difference in median value based on `TARGET_FLAG` values, while the range of predictor values for each flag value differs noticeably.

## 2 Data Preparation

As stated in Part 1, four predictor variables have missing values. Because these missing values represent, in many cases, a non-menial proportion of the dataset at large, they are imputed so the cases containing missing values can be included in modeling. Due to the skewness illustrated by some of the variables with missing data (`INCOME` and `HOME_VAL`), the median is used to avoid any bias introduced into the mean by the skewness of these variables' distribution. For the remaining two variables, the median is also used for imputation for consistency.

Additionally, there is one instance of a vehicle's `CAR_AGE` being -3. Since this variable represents the age of the car in years, this is a nonsensical value. This instance of this variable is removed before imputation.

A summary of the variables with imputation of median values is presented below:

| | MEAN | MEDIAN | IQR | SKEW | $r_{FLAG}$ | $r_{AMT}$ | NAs |
|---|---|---|---|---|---|---|---|
| **TARGET_FLAG** | 0.26 | 0 | 1 | 1.07 | 1 | 0.53 | 0 |
| **TARGET_AMT** | 1504 | 0 | 1036 | 8.71 | 0.53 | 1 | 0 |
| **KIDSDRIV** | 0.17 | 0 | 0 | 3.35 | 0.1 | 0.06 | 0 |
| **AGE** | 44.79 | 45 | 12 | -0.03 | -0.1 | -0.04 | 0 |
| **HOMEKIDS** | 0.72 | 0 | 1 | 1.34 | 0.12 | 0.06 | 0 |
| **YOJ** | 10.53 | 11 | 4 | -1.26 | -0.07 | -0.02 | 0 |
| **INCOME** | 61469 | 54028 | 53597 | 1.24 | -0.14 | -0.06 | 0 |
| **PARENT1** | 0.13 | 0 | 0 | 2.17 | 0.16 | 0.1 | 0 |
| **HOME_VAL** | 155225 | 161160 | 233352 | 0.49 | -0.18 | -0.08 | 0 |
| **MSTATUS** | 0.6 | 1 | 1 | -0.41 | -0.14 | -0.09 | 0 |
| **SEX** | 0.46 | 0 | 1 | 0.14 | -0.02 | 0.01 | 0 |
| **TRAVTIME** | 33.49 | 33 | 22 | 0.45 | 0.05 | 0.03 | 0 |
| **CAR_USE** | 0.37 | 0 | 1 | 0.53 | 0.14 | 0.1 | 0 |
| **BLUEBOOK** | 15710 | 14440 | 11570 | 0.79 | -0.1 | 0 | 0 |
| **TIF** | 5.35 | 4 | 6 | 0.89 | -0.08 | -0.05 | 0 |
| **RED_CAR** | 0.29 | 0 | 1 | 0.92 | -0.01 | 0.01 | 0 |
| **OLDCLAIM** | 4037 | 0 | 4636 | 3.12 | 0.14 | 0.07 | 0 |
| **CLM_FREQ** | 0.8 | 0 | 2 | 1.21 | 0.22 | 0.12 | 0 |
| **REVOKED** | 0.12 | 0 | 0 | 2.3 | 0.15 | 0.06 | 0 |
| **MVR_PTS** | 1.7 | 1 | 3 | 1.35 | 0.22 | 0.14 | 0 |
| **CAR_AGE** | 8.31 | 8 | 8 | 0.3 | -0.1 | -0.06 | 0 |
| **URBANICITY** | 0.8 | 1 | 0 | -1.46 | 0.22 | 0.12 | 0 |
| **HSDropout** | 0.15 | 0 | 0 | 1.99 | 0.05 | 0.03 | 0 |
| **HS** | 0.29 | 0 | 1 | 0.95 | 0.11 | 0.04 | 0 |
| **Bachelors** | 0.27 | 0 | 1 | 1.01 | -0.04 | -0.02 | 0 |
| **Masters** | 0.2 | 0 | 0 | 1.48 | -0.08 | -0.04 | 0 |
| **PhD** | 0.09 | 0 | 0 | 2.88 | -0.07 | -0.02 | 0 |
| **Minivan** | 0.26 | 0 | 1 | 1.08 | -0.14 | -0.08 | 0 |
| **Panel_Truck** | 0.08 | 0 | 0 | 3.03 | 0 | 0.03 | 0 |
| **Pickup** | 0.17 | 0 | 0 | 1.75 | 0.06 | 0.02 | 0 |
| **Sports_Car** | 0.11 | 0 | 0 | 2.47 | 0.06 | 0.02 | 0 |
| **Van** | 0.09 | 0 | 0 | 2.82 | 0 | 0.02 | 0 |
| **SUV** | 0.28 | 0 | 1 | 0.97 | 0.05 | 0.01 | 0 |
| **Blank_Job** | 0.06 | 0 | 0 | 3.55 | 0 | 0.02 | 0 |
| **Professional** | 0.14 | 0 | 0 | 2.11 | -0.04 | 0 | 0 |
| **Blue_Collar** | 0.22 | 0 | 0 | 1.33 | 0.1 | 0.06 | 0 |
| **Clerical** | 0.16 | 0 | 0 | 1.9 | 0.03 | 0.01 | 0 |
| **Doctor** | 0.03 | 0 | 0 | 5.49 | -0.06 | -0.03 | 0 |
| **Lawyer** | 0.1 | 0 | 0 | 2.62 | -0.06 | -0.03 | 0 |
| **Manager** | 0.12 | 0 | 0 | 2.32 | -0.11 | -0.06 | 0 |
| **Home_Maker** | 0.08 | 0 | 0 | 3.13 | 0.01 | -0.01 | 0 |
| **Student** | 0.09 | 0 | 0 | 2.92 | 0.08 | 0.02 | 0 |

The creation of additional or combined predictors did not yield any predictors that would improve models, however due to the skewness of some predictors, log transformations were conducted.

In the table below, we can see the effect of each of the log transformation on the correlations to the two target variables. For many variables, the transformations do not improve the correlation, but the log transformations of `INCOME` and `HOME_VAL` may fit our models better.

| | $r_{FLAG}$ | $\log r_{FLAG}$ | $r_{AMT}$ | $\log r_{AMT}$ |
|---|---|---|---|---|
| **TARGET_FLAG** | 1 | 1 | 0.53 | 0.53 |
| **TARGET_AMT** | 0.53 | 1 | 1 | 0.58 |
| **KIDSDRIV** | 0.1 | 0.11 | 0.06 | 0.06 |
| **AGE** | -0.1 | -0.12 | -0.04 | -0.05 |
| **HOMEKIDS** | 0.12 | 0.13 | 0.06 | 0.07 |
| **YOJ** | -0.07 | -0.08 | -0.02 | -0.02 |
| **INCOME** | -0.14 | -0.11 | -0.06 | -0.03 |
| **PARENT1** | 0.16 | 0.16 | 0.1 | 0.1 |
| **HOME_VAL** | -0.18 | -0.15 | -0.08 | -0.07 |
| **MSTATUS** | -0.14 | -0.14 | -0.09 | -0.09 |
| **SEX** | -0.02 | -0.02 | 0.01 | 0.01 |
| **TRAVTIME** | 0.05 | 0.06 | 0.03 | 0.03 |
| **CAR_USE** | 0.14 | 0.14 | 0.1 | 0.1 |
| **BLUEBOOK** | -0.1 | -0.11 | 0 | 0 |
| **TIF** | -0.08 | -0.08 | -0.05 | -0.05 |
| **RED_CAR** | -0.01 | -0.01 | 0.01 | 0.01 |
| **OLDCLAIM** | 0.14 | 0.24 | 0.07 | 0.13 |
| **CLM_FREQ** | 0.22 | 0.24 | 0.12 | 0.13 |
| **REVOKED** | 0.15 | 0.15 | 0.06 | 0.06 |
| **MVR_PTS** | 0.22 | 0.17 | 0.14 | 0.11 |
| **CAR_AGE** | -0.1 | -0.09 | -0.06 | -0.05 |
| **URBANICITY** | 0.22 | 0.22 | 0.12 | 0.12 |
| **HSDropout** | 0.05 | 0.05 | 0.03 | 0.03 |
| **HS** | 0.11 | 0.11 | 0.04 | 0.04 |
| **Bachelors** | -0.04 | -0.04 | -0.02 | -0.02 |
| **Masters** | -0.08 | -0.08 | -0.04 | -0.04 |
| **PhD** | -0.07 | -0.07 | -0.02 | -0.02 |
| **Minivan** | -0.14 | -0.14 | -0.08 | -0.08 |
| **Panel_Truck** | 0 | 0 | 0.03 | 0.03 |
| **Pickup** | 0.06 | 0.06 | 0.02 | 0.02 |
| **Sports_Car** | 0.06 | 0.06 | 0.02 | 0.02 |
| **Van** | 0 | 0 | 0.02 | 0.02 |
| **SUV** | 0.05 | 0.05 | 0.01 | 0.01 |
| **Blank_Job** | 0 | 0 | 0.02 | 0.02 |
| **Professional** | -0.04 | -0.04 | 0 | 0 |
| **Blue_Collar** | 0.1 | 0.1 | 0.06 | 0.06 |
| **Clerical** | 0.03 | 0.03 | 0.01 | 0.01 |
| **Doctor** | -0.06 | -0.06 | -0.03 | -0.03 |
| **Lawyer** | -0.06 | -0.06 | -0.03 | -0.03 |
| **Manager** | -0.11 | -0.11 | -0.06 | -0.06 |
| **Home_Maker** | 0.01 | 0.01 | -0.01 | -0.01 |
| **Student** | 0.08 | 0.08 | 0.02 | 0.02 |

# 3 Model Creation

## 3.1 Multiple Linear Regression

The second response variable, `TARGET_AMT`, will have a value of 0 for cases that were not flagged as having a car accident (`TARGET_FLAG = 1`), and the cost of the accident otherwise. Since we are not dealing with a two-level categorical response, we will use multiple linear regression models for this variable. The training data has been subset to only include cases that were flagged for a car crash.

### 3.1.1 Full Model

We will start with the full model, including all 38 predictor variables. Median values were imputed into the variables with NAs.

| | Estimate | Std. Error | t value | Pr(>\|t\|) |
|---|---|---|---|---|
| **KIDSDRIV** | -171.9 | 316.6 | -0.5429 | 0.5873 |
| **AGE** | 18.35 | 21.24 | 0.8637 | 0.3879 |
| **HOMEKIDS** | 213.5 | 207.1 | 1.031 | 0.3028 |
| **YOJ** | 19.16 | 49.18 | 0.3896 | 0.6969 |
| **INCOME** | -0.009014 | 0.006742 | -1.337 | 0.1814 |
| **PARENT1** | 277.2 | 587.3 | 0.4721 | 0.6369 |
| **HOME_VAL** | 0.002198 | 0.00202 | 1.088 | 0.2767 |
| **MSTATUS** | -803.6 | 493.5 | -1.628 | 0.1036 |
| **SEX** | 1397 | 656.4 | 2.129 | 0.03338 |
| **TRAVTIME** | 0.7528 | 11.08 | 0.06794 | 0.9458 |
| **CAR_USE** | 438.4 | 521.6 | 0.8405 | 0.4008 |
| **BLUEBOOK** | 0.1245 | 0.03053 | 4.077 | 4.735e-05 |
| **TIF** | -15.74 | 42.52 | -0.3702 | 0.7112 |
| **RED_CAR** | -193.3 | 496.5 | -0.3894 | 0.697 |
| **OLDCLAIM** | 0.02502 | 0.02263 | 1.105 | 0.2691 |
| **CLM_FREQ** | -115.4 | 158 | -0.7305 | 0.4651 |
| **REVOKED** | -1125 | 516.6 | -2.177 | 0.02957 |
| **MVR_PTS** | 110.8 | 68.53 | 1.617 | 0.1061 |
| **CAR_AGE** | -98.42 | 44.06 | -2.233 | 0.02562 |
| **URBANICITY** | 97.78 | 756.2 | 0.1293 | 0.8971 |
| **HSDropout** | -2396 | 1312 | -1.827 | 0.06787 |
| **HS** | -2792 | 1228 | -2.273 | 0.02315 |
| **Bachelors** | -2136 | 1141 | -1.871 | 0.06142 |
| **Masters** | -1202 | 954.1 | -1.26 | 0.2077 |
| **Minivan** | -902.5 | 666.8 | -1.354 | 0.176 |
| **Panel_Truck** | -1543 | 1173 | -1.315 | 0.1885 |
| **Pickup** | -958.9 | 679.5 | -1.411 | 0.1583 |
| **Sports_Car** | 158.2 | 536.4 | 0.2949 | 0.7681 |
| **Van** | -839.2 | 945.3 | -0.8878 | 0.3748 |
| **Blank_Job** | -114.7 | 1286 | -0.0892 | 0.9289 |
| **Professional** | 946.2 | 865.8 | 1.093 | 0.2746 |
| **Blue_Collar** | 405.9 | 706.3 | 0.5747 | 0.5656 |
| **Clerical** | 193.7 | 730.3 | 0.2653 | 0.7908 |
| **Doctor** | -2231 | 1948 | -1.145 | 0.2521 |
| **Lawyer** | 212.5 | 1273 | 0.167 | 0.8674 |
| **Manager** | -894.1 | 1064 | -0.8405 | 0.4007 |
| **Home_Maker** | -138.2 | 830.8 | -0.1664 | 0.8679 |
| **(Intercept)** | 5680 | 1951 | 2.912 | 0.003633 |

(Dispersion parameter for gaussian family taken to be 59138671 )

| | |
|---|---|
| Null deviance: | 1.290e+11 on 2152 degrees of freedom |

| Residual deviance: | 1.251e+11 on 2115 degrees of freedom |
|---|---|

Less than half (15) of the predictor coefficients are significant under a reasonable $\alpha$ , and 4 are between a 0.05 an 0.1 p-value. The coefficients vary in size, though all of them are quite small, 20 of the coefficients have negative values. Most of the slopes (negative or positive) appear to make sense in predicting the payout if the vehicle was in a crash, such as the value of the bluebook value of the car and vehicle type having a positive relationship with the response variable. 3 of the predictors coefficients were not defined, most likely due to the number of predictors, and some slight correlation with another variable. `PhD` for instance, has a higher correlation to the `Doctor` job variable.

### 3.1.2 Log-Transformed Full Model

As mentioned in the first section, the response variable is quite skewed. Our second multiple linear regression model uses a log transformation on the `TARGET_AMT` variable allows for a better fit of the full model.

| | Estimate | Std. Error | t value | Pr(>\|t\|) |
|---|---|---|---|---|
| **KIDSDRIV** | -0.03288 | 0.03329 | -0.9875 | 0.3235 |
| **AGE** | 0.001866 | 0.002234 | 0.8353 | 0.4036 |
| **HOMEKIDS** | 0.0231 | 0.02178 | 1.061 | 0.289 |
| **YOJ** | -0.001023 | 0.005172 | -0.1979 | 0.8432 |
| **INCOME** | -1.326e-06 | 7.09e-07 | -1.87 | 0.06161 |
| **PARENT1** | 0.02438 | 0.06176 | 0.3947 | 0.6931 |
| **HOME_VAL** | 1.478e-07 | 2.125e-07 | 0.6956 | 0.4867 |
| **MSTATUS** | -0.09766 | 0.0519 | -1.882 | 0.06001 |
| **SEX** | 0.09313 | 0.06904 | 1.349 | 0.1775 |
| **TRAVTIME** | -0.0002604 | 0.001165 | -0.2234 | 0.8232 |
| **CAR_USE** | 0.01333 | 0.05486 | 0.2429 | 0.8081 |
| **BLUEBOOK** | 1.197e-05 | 3.211e-06 | 3.729 | 0.0001973 |
| **TIF** | -0.00197 | 0.004471 | -0.4406 | 0.6596 |
| **RED_CAR** | 0.02164 | 0.05221 | 0.4146 | 0.6785 |
| **OLDCLAIM** | 4.45e-06 | 2.38e-06 | 1.869 | 0.06169 |
| **CLM_FREQ** | -0.03631 | 0.01662 | -2.185 | 0.029 |
| **REVOKED** | -0.0953 | 0.05433 | -1.754 | 0.07957 |
| **MVR_PTS** | 0.01478 | 0.007207 | 2.05 | 0.04045 |
| **CAR_AGE** | -0.002596 | 0.004634 | -0.5603 | 0.5753 |
| **URBANICITY** | 0.05696 | 0.07953 | 0.7162 | 0.474 |
| **HSDropout** | -0.2487 | 0.138 | -1.803 | 0.07153 |
| **HS** | -0.24 | 0.1292 | -1.858 | 0.06334 |
| **Bachelors** | -0.2772 | 0.12 | -2.31 | 0.021 |
| **Masters** | -0.09599 | 0.1003 | -0.9566 | 0.3389 |
| **Minivan** | -0.09253 | 0.07012 | -1.32 | 0.1871 |
| **Panel_Truck** | -0.09075 | 0.1234 | -0.7357 | 0.462 |
| **Pickup** | -0.06364 | 0.07146 | -0.8906 | 0.3732 |
| **Sports_Car** | -0.03615 | 0.05641 | -0.6409 | 0.5217 |
| **Van** | -0.1044 | 0.09941 | -1.05 | 0.2937 |
| **Blank_Job** | -0.01979 | 0.1352 | -0.1464 | 0.8837 |
| **Professional** | 0.08607 | 0.09106 | 0.9452 | 0.3446 |
| **Blue_Collar** | 0.04131 | 0.07428 | 0.5561 | 0.5782 |
| **Clerical** | 0.03384 | 0.0768 | 0.4407 | 0.6595 |
| **Doctor** | -0.05833 | 0.2048 | -0.2848 | 0.7758 |
| **Lawyer** | -0.03244 | 0.1338 | -0.2424 | 0.8085 |
| **Manager** | -0.001392 | 0.1119 | -0.01244 | 0.9901 |
| **Home_Maker** | -0.07093 | 0.08737 | -0.8118 | 0.417 |
| **(Intercept)** | 8.291 | 0.2052 | 40.41 | 4.016e-265 |

(Dispersion parameter for gaussian family taken to be 0.6540844 )

| Null deviance: | 1421 on 2152 degrees of freedom |
|---|---|

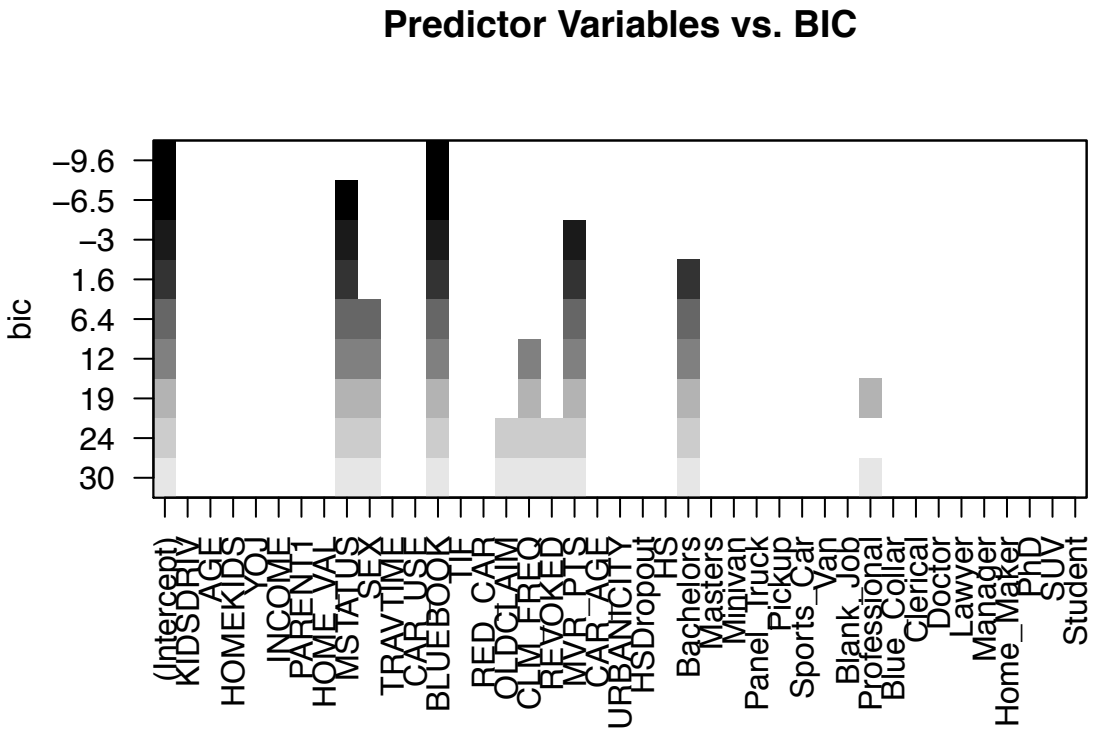
| Residual deviance: | 1383 on 2115 degrees of freedom |
|---|---|

Now, the number of our predictors that are significant (under a .05 $\alpha$) has increased, with 19 coefficients being highly significant. Our AIC has also decreased greatly from 44678 to 5235.6, indicating a better "goodness of fit".

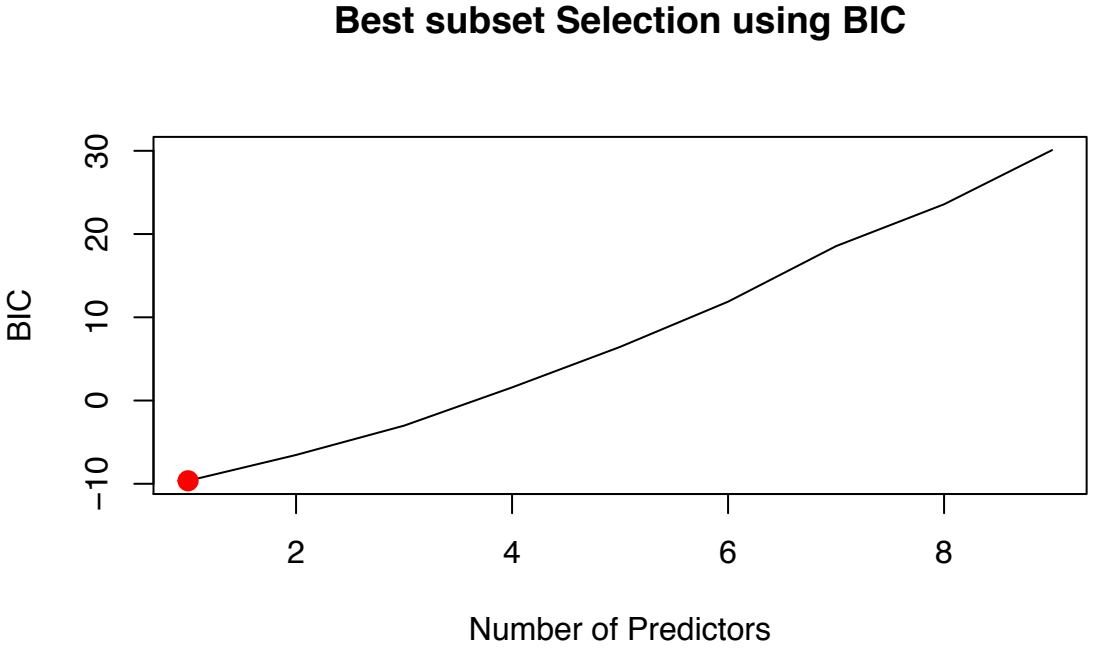
### 3.1.3 Bayesian Information Criteria

Our third linear regression model uses Bayesian Information Criteria (BIC) to determine the quantity of predictors, as well as specifically which ones to use in the model. For this model and going forward, we will continue to use the log transformation of the response variable as part of the model.

```
Reordering variables and trying again:
```




```
[1] 1
```



Examining the first plot, the model that results in the lowest "bic" includes the intercept and only one predictor.

Looking at the second plot, the best subset selection using BIC is the Bluebook value predictor. This is the only predictor chosen by the BIC method for all the different models. This makes sense since the biggest factor in determining the cost of a car accident is generally the value of the vehicle (disregarding personal injury or property damage). In the full model, Bluebook value had the most significant p-value, and a positive coefficient, given the higher the value of the vehicle, the higher the cost of the accident.

| | Estimate | Std. Error | t value | Pr(>\|t\|) |
|---|---|---|---|---|
| **BLUEBOOK** | 1.052e-05 | 2.099e-06 | 5.011 | 5.849e-07 |
| **(Intercept)** | 8.126 | 0.03462 | 234.7 | 0 |

(Dispersion parameter for gaussian family taken to be 0.6529404 )

| | |
|---|---|
| Null deviance: | 1421 on 2152 degrees of freedom |
| Residual deviance: | 1404 on 2151 degrees of freedom |

Compared to our full models, the lone predictor model chosen by the BIC method results in a lower AIC value of 5196.2.

Since `BLUEBOOK` has a good amount of skew, we will apply a log transformation on the predictor to see if we can improve the fit of the model:

| | Estimate | Std. Error | t value | Pr(>\|t\|) |
|---|---|---|---|---|
| **log(BLUEBOOK)** | 0.1598 | 0.02626 | 6.085 | 1.377e-09 |
| **(Intercept)** | 6.778 | 0.2468 | 27.46 | 1.318e-142 |

(Dispersion parameter for gaussian family taken to be 0.6493857 )

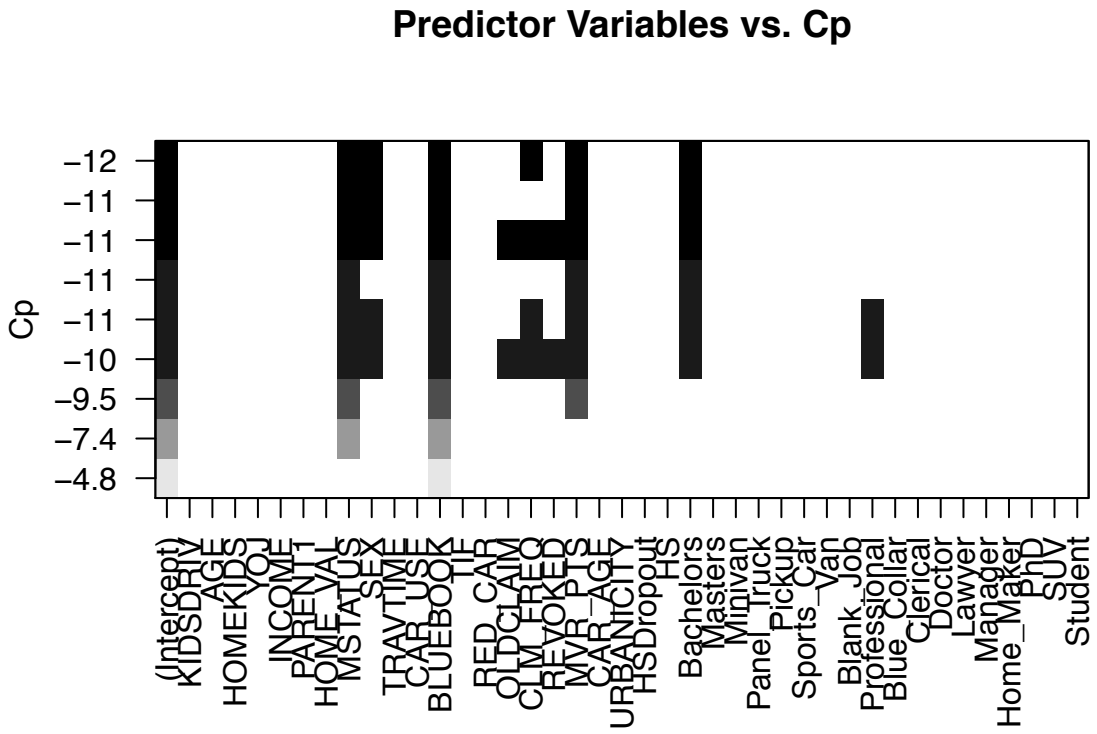
| | |
|---|---|
| Null deviance: | 1421 on 2152 degrees of freedom |
| Residual deviance: | 1397 on 2151 degrees of freedom |

The transformation of the bluebook value data makes for a better fit of our model, and the resulting AIC is 5184.4, the lowest value yet.

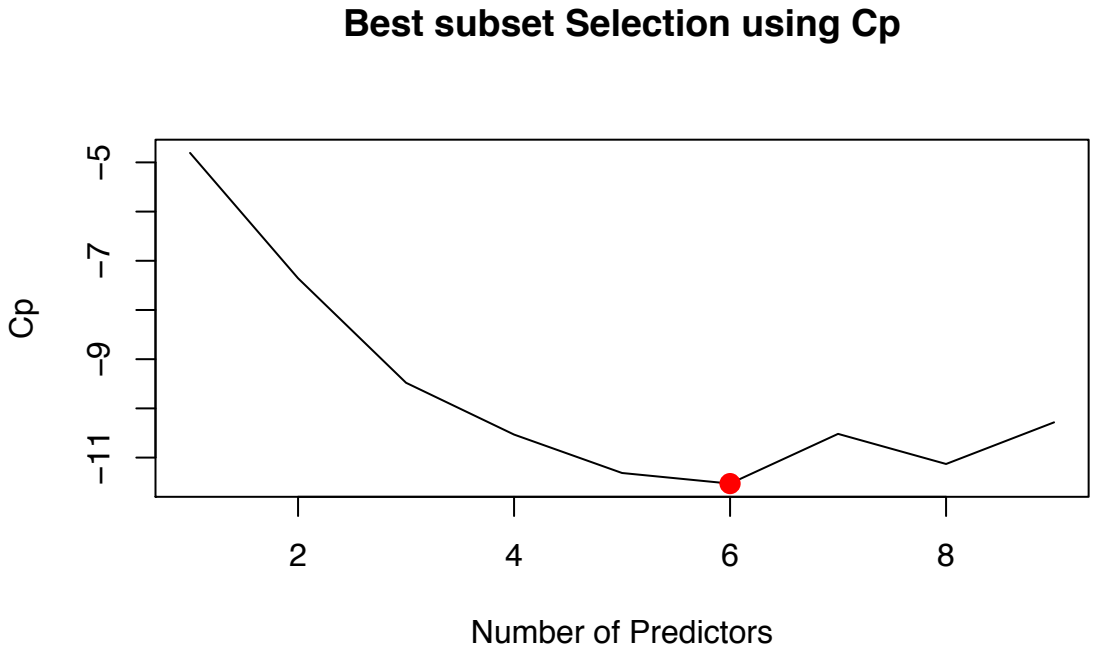
#### 3.1.4 Mallow's $C_p$

Our next multivariate regression model is created utilizing Mallow's $C_p$, instead of BIC, to determine the number of predictors used for a best fit model.

```
Reordering variables and trying again:
```




```
[1] 6
```




Similar to the plots for BIC selection, the smaller $C_p$ values correspond with the best-fit model. Using $C_p$, six predictors result in the lowest "Cp" value, which are `MSTATUS`, `SEX`, `BLUEBOOK`, `CLM_FREQ`, `MVR_PTS`, and `Bachelors`. These predictors selected by using the $C_p$ method correspond with most of the statistically significant predictors from our log transformed response full model, with only the `SEX` predictor over the 0.05 threshold.

| | Estimate | Std. Error | t value | Pr(>\|t\|) |
|---|---|---|---|---|
| **MSTATUS** | -0.07939 | 0.03487 | -2.277 | 0.02291 |
| **SEX** | 0.05837 | 0.03507 | 1.665 | 0.09614 |
| **BLUEBOOK** | 1.038e-05 | 2.105e-06 | 4.932 | 8.773e-07 |
| **CLM_FREQ** | -0.0218 | 0.01458 | -1.495 | 0.1351 |
| **MVR_PTS** | 0.01711 | 0.007059 | 2.423 | 0.01547 |
| **Bachelors** | -0.07107 | 0.0407 | -1.746 | 0.08093 |

| | Estimate | Std. Error | t value | Pr(>\|t\|) |
|---|---|---|---|---|
| **(Intercept)** | 8.142 | 0.04731 | 172.1 | 0 |

(Dispersion parameter for gaussian family taken to be 0.6493582 )

| | |
|---|---|
| Null deviance: | 1421 on 2152 degrees of freedom |
| Residual deviance: | 1394 on 2146 degrees of freedom |

As with the model created with BIC, the coefficient of the `BLUEBOOK` predictor is highly statistically significant. The other variables chosen by the $C_p$ method vary in size and significance of their coefficients. `MSTATUS` and `MVR_PTS` are below the 0.05 $\alpha$, and the remaining predictors are slightly above it. The relationship of marital status and sex to the cost of the accident may have something to do with the type of vehicles single vs. married people, and men vs. women choose to drive.

This model provides the second-lowest AIC of 5189.3, up slightly from the lone-predictor model chosen by BIC method.

Using the same thought-process as our BIC selection method, the most significant predictor, `BLUEBOOK` is log transformed, resulting in another slight reduction in AIC to 5177.5.

| | Estimate | Std. Error | t value | Pr(>\|t\|) |
|---|---|---|---|---|
| **MSTATUS** | -0.07834 | 0.03478 | -2.253 | 0.02438 |
| **SEX** | 0.05497 | 0.03497 | 1.572 | 0.1162 |
| **log(BLUEBOOK)** | 0.1586 | 0.02633 | 6.024 | 2.002e-09 |
| **CLM_FREQ** | -0.02224 | 0.01454 | -1.529 | 0.1264 |
| **MVR_PTS** | 0.01742 | 0.007041 | 2.474 | 0.01346 |
| **Bachelors** | -0.07329 | 0.04059 | -1.806 | 0.07111 |
| **(Intercept)** | 6.804 | 0.2488 | 27.35 | 1.63e-141 |

(Dispersion parameter for gaussian family taken to be 0.6457992 )

| | |
|---|---|
| Null deviance: | 1421 on 2152 degrees of freedom |
| Residual deviance: | 1386 on 2146 degrees of freedom |

#### 3.1.5 Model 5: Manual Selection

Lastly, we'll use some intuition to manually select predictors for our model based on what we know about driving habits, and what may logically affect the predicted costs of a car accident. Keeping with the theme of using the `BLUEBOOK` variable, we will use two of the other predictors that were significant in the full model, that may help in determining the cost of a car accident. `CLM_FREQ` and `MVR_PTS` are both good indicators of a persons driving habits, and more careless or reckless drivers may get into accidents with higher costs than a safer or more responsible driver would. Both of these variables appeared in the $C_p$ selected model, but we will only use these and not the `SEX` or `MSTATUS` predictors for a more parsimonious model.

| | Estimate | Std. Error | t value | Pr(>\|t\|) |
|---|---|---|---|---|
| **log(BLUEBOOK)** | 0.1612 | 0.02624 | 6.145 | 9.497e-10 |
| **CLM_FREQ** | -0.02301 | 0.01456 | -1.58 | 0.1143 |
| **MVR_PTS** | 0.01743 | 0.007052 | 2.472 | 0.01353 |
| **(Intercept)** | 6.749 | 0.248 | 27.21 | 2.313e-140 |

(Dispersion parameter for gaussian family taken to be 0.6479137 )

| | |
|---|---|
| Null deviance: | 1421 on 2152 degrees of freedom |
| Residual deviance: | 1392 on 2149 degrees of freedom |

The result is a model with a slightly higher AIC value of 5181.1, and two of the three predictors being below a 0.05 $\alpha$. The

positive coefficients of BLUEBOOK and MVR_PTS make sense, and the negative slope on claim frequency (less claims, less costly driving habits) makes sense as well.

## 3.2 Binary Logistic Regression

Because we are attempting to predict two response variables, instead of doing this with one model, two separate models will be created to predict each response variable. First, we will create models for the TARGET_FLAG variable, since this will tell us whether a case was involved in a car accident or not. Binary logistic regression is used, as the flag variable is a two-level categorical response (0 or 1).

### 3.2.1 Full Model

We'll start with a full model, using all predictors and no transformations.

| | Estimate | Std. Error | z value | Pr(>\|z\|) |
|---|---|---|---|---|
| **KIDSDRIV** | 0.3862 | 0.06122 | 6.308 | 2.821e-10 |
| **AGE** | -0.001015 | 0.00402 | -0.2525 | 0.8007 |
| **HOMEKIDS** | 0.04965 | 0.03713 | 1.337 | 0.1811 |
| **YOJ** | -0.01105 | 0.008582 | -1.288 | 0.1977 |
| **INCOME** | -3.423e-06 | 1.082e-06 | -3.165 | 0.001551 |
| **PARENT1** | 0.382 | 0.1096 | 3.485 | 0.0004918 |
| **HOME_VAL** | -1.306e-06 | 3.42e-07 | -3.819 | 0.0001338 |
| **MSTATUS** | -0.4938 | 0.08357 | -5.909 | 3.45e-09 |
| **SEX** | 0.08251 | 0.112 | 0.7365 | 0.4614 |
| **TRAVTIME** | 0.01457 | 0.001883 | 7.736 | 1.028e-14 |
| **CAR_USE** | 0.7564 | 0.09172 | 8.247 | 1.626e-16 |
| **BLUEBOOK** | -2.084e-05 | 5.263e-06 | -3.959 | 7.52e-05 |
| **TIF** | -0.05547 | 0.007344 | -7.553 | 4.257e-14 |
| **RED_CAR** | -0.009728 | 0.08636 | -0.1126 | 0.9103 |
| **OLDCLAIM** | -1.389e-05 | 3.91e-06 | -3.554 | 0.00038 |
| **CLM_FREQ** | 0.1959 | 0.02855 | 6.864 | 6.687e-12 |
| **REVOKED** | 0.8874 | 0.09133 | 9.716 | 2.569e-22 |
| **MVR_PTS** | 0.1133 | 0.01361 | 8.324 | 8.53e-17 |
| **CAR_AGE** | -0.0007196 | 0.007549 | -0.09533 | 0.9241 |
| **URBANICITY** | 2.39 | 0.1128 | 21.18 | 1.43e-99 |
| **HSDropout** | 0.1677 | 0.214 | 0.7836 | 0.4333 |
| **HS** | 0.1853 | 0.1962 | 0.9442 | 0.3451 |
| **Bachelors** | -0.2136 | 0.1797 | -1.189 | 0.2345 |
| **Masters** | -0.1227 | 0.1527 | -0.8032 | 0.4218 |
| **Minivan** | -0.7682 | 0.1113 | -6.904 | 5.055e-12 |
| **Panel_Truck** | -0.2074 | 0.2002 | -1.036 | 0.3002 |
| **Pickup** | -0.2142 | 0.1166 | -1.838 | 0.06613 |
| **Sports_Car** | 0.257 | 0.09815 | 2.618 | 0.00884 |
| **Van** | -0.1496 | 0.1589 | -0.9416 | 0.3464 |
| **Blank_Job** | -0.2161 | 0.2145 | -1.007 | 0.3137 |
| **Professional** | -0.05417 | 0.154 | -0.3518 | 0.725 |
| **Blue_Collar** | 0.09451 | 0.1289 | 0.7333 | 0.4634 |
| **Clerical** | 0.1946 | 0.1315 | 1.48 | 0.1388 |
| **Doctor** | -0.6619 | 0.3018 | -2.193 | 0.02829 |
| **Lawyer** | -0.1112 | 0.209 | -0.5321 | 0.5947 |
| **Manager** | -0.7733 | 0.1697 | -4.557 | 5.182e-06 |
| **Home_Maker** | 0.01626 | 0.1507 | 0.1079 | 0.9141 |
| **(Intercept)** | -2.847 | 0.3325 | -8.562 | 1.112e-17 |

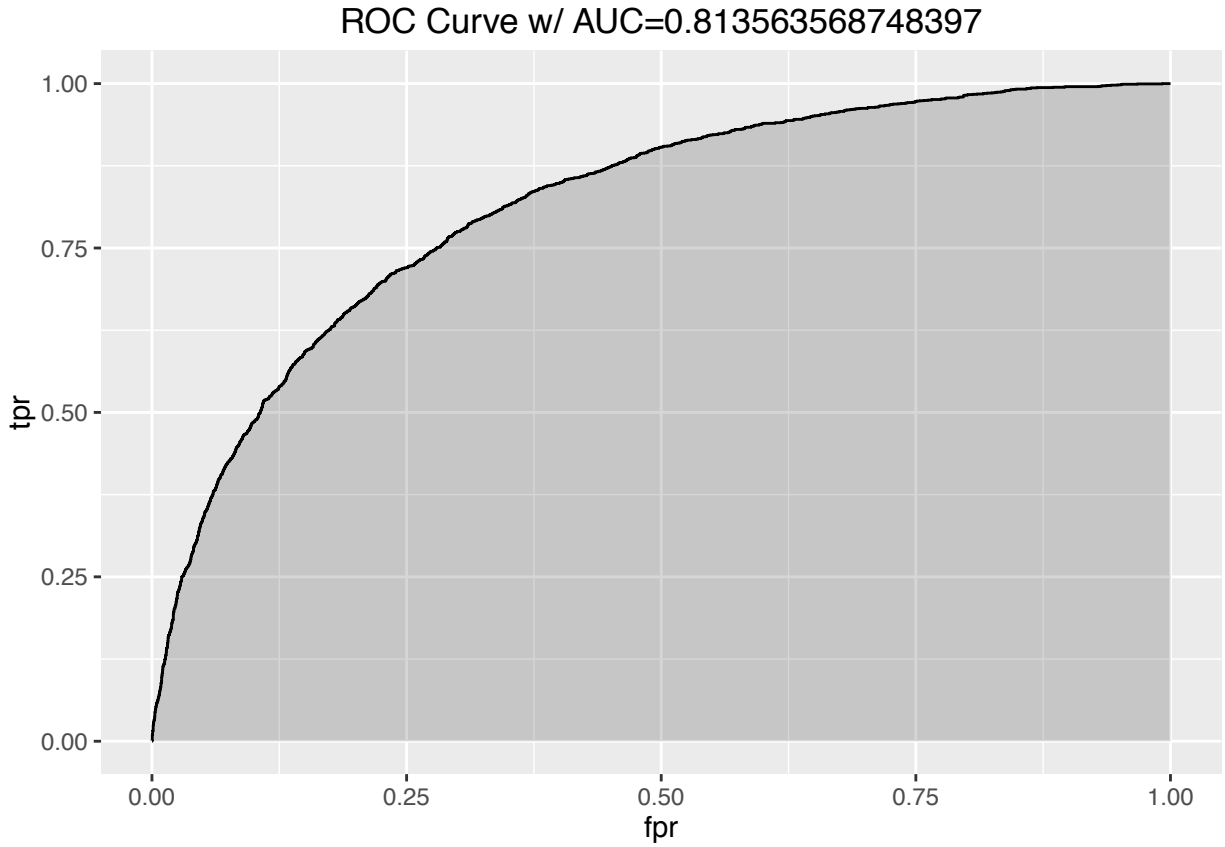
(Dispersion parameter for binomial family taken to be 1 )

| | |
|---|---|
| Null deviance: | 9418 on 8160 degrees of freedom |

| Residual deviance: | 7298 on 8123 degrees of freedom |
|---|---|

```
Error in vif.default(glmflagfull): there are aliased coefficients in the model
```

```
Error in if (tail(stdout, 1) == "") {: argument is of length zero
```




Because we transformed some of the categorical predictors into two-level categorical variables, we have 38 predictors in model. The coefficients vary quite a bit, and we have 13 predictors that are not significant under any reasonable value of $\alpha$. Many of the predictors with high statistical significance seem reasonable when considering the likelihood of an accident. Number of teen drivers, travel time, car use, claim frequency, driver record, and location all have positive slopes, and have significant p-values.

The full model has an area under the curve of 0.8136, and an accuracy of 0.7927

### 3.2.2 Reduced Model

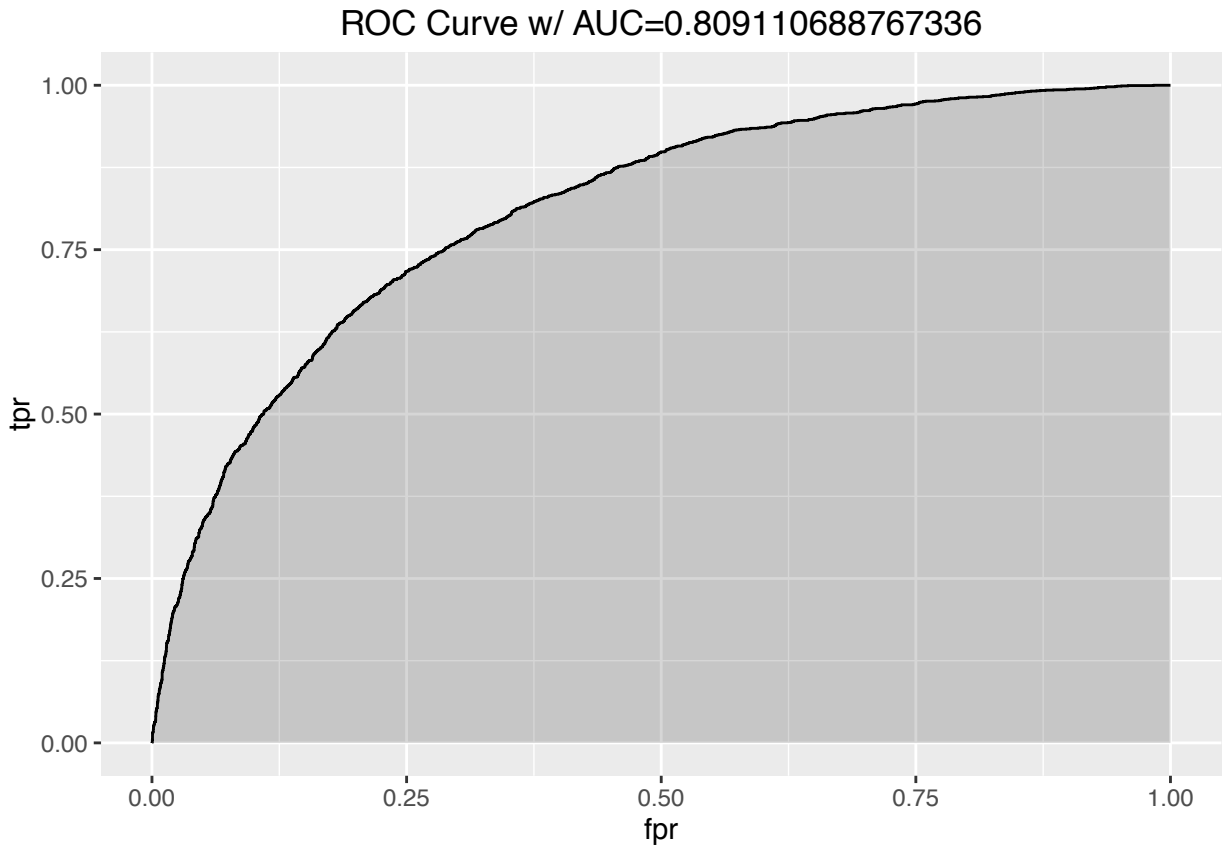
For our next Binary logistic regression model, predictors classified as non-significant have been removed from the full model. The remaining coefficients of predictors are now all significant with regards to their p-values. As with the full model, the predictors relating to the use of the car, performance of the driver, and location have the greatest effect.

| | Estimate | Std. Error | z value | Pr(>\|z\|) |
|---|---|---|---|---|
| **KIDSDRIV** | 0.4283 | 0.05478 | 7.82 | 5.297e-15 |
| **INCOME** | -6.019e-06 | 8.966e-07 | -6.714 | 1.898e-11 |
| **PARENT1** | 0.4929 | 0.09335 | 5.28 | 1.289e-07 |
| **HOME_VAL** | -1.476e-06 | 3.312e-07 | -4.456 | 8.339e-06 |
| **MSTATUS** | -0.4198 | 0.07821 | -5.367 | 7.993e-08 |
| **TRAVTIME** | 0.01426 | 0.001865 | 7.644 | 2.106e-14 |
| **CAR_USE** | 0.7541 | 0.06373 | 11.83 | 2.639e-32 |
| **BLUEBOOK** | -2.584e-05 | 3.98e-06 | -6.492 | 8.494e-11 |
| **TIF** | -0.05476 | 0.007289 | -7.513 | 5.764e-14 |

| | Estimate | Std. Error | z value | Pr(>\|z\|) |
|---|---|---|---|---|
| **OLDCLAIM** | -1.368e-05 | 3.884e-06 | -3.522 | 0.0004289 |
| **CLM_FREQ** | 0.1945 | 0.02831 | 6.871 | 6.385e-12 |
| **REVOKED** | 0.8825 | 0.09064 | 9.736 | 2.114e-22 |
| **MVR_PTS** | 0.1125 | 0.01348 | 8.347 | 6.977e-17 |
| **URBANICITY** | 2.317 | 0.1119 | 20.71 | 3.096e-95 |
| **Minivan** | -0.6336 | 0.07513 | -8.433 | 3.358e-17 |
| **Sports_Car** | 0.3073 | 0.09176 | 3.349 | 0.0008114 |
| **Doctor** | -0.5315 | 0.221 | -2.405 | 0.01618 |
| **Manager** | -0.8527 | 0.1057 | -8.07 | 7.017e-16 |
| **(Intercept)** | -2.76 | 0.1553 | -17.77 | 1.238e-70 |

(Dispersion parameter for binomial family taken to be 1 )

| | |
|---|---|
| Null deviance: | 9418 on 8160 degrees of freedom |
| Residual deviance: | 7361 on 8142 degrees of freedom |

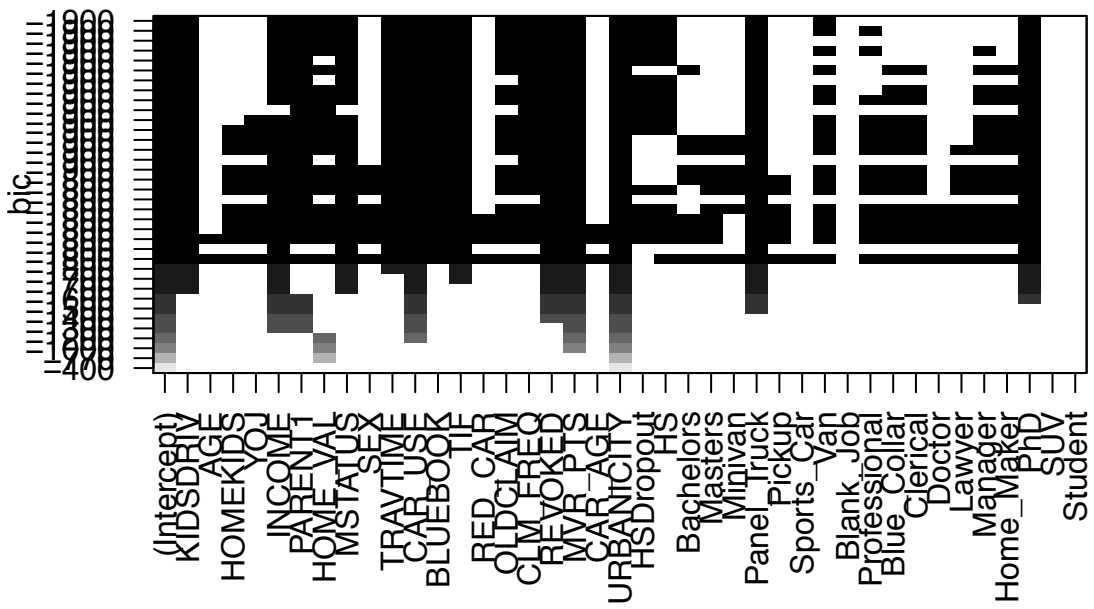


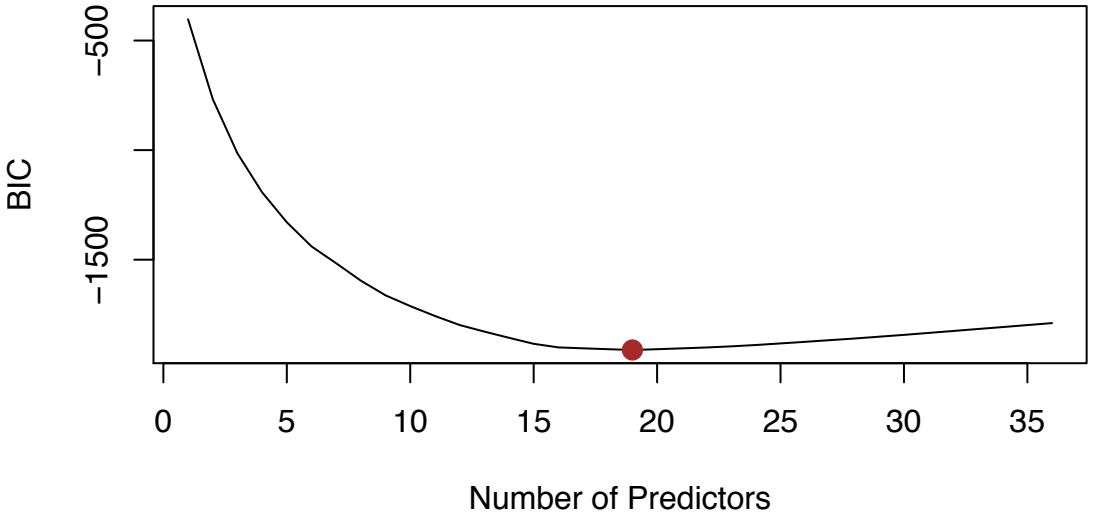

The reduced model has an area under the curve of 0.80911, and an accuracy of 0.7930

#### 3.2.3 Best Subsets

Our last binary logistic regression model will be created using the `regsubsets` function from the `leaps` R package, which performs model selection for us. Bayesian Information Criterion is used for determining the number of predictors, with 19 determined to be the number resulting in the lowest BIC value.

```
Reordering variables and trying again:
```



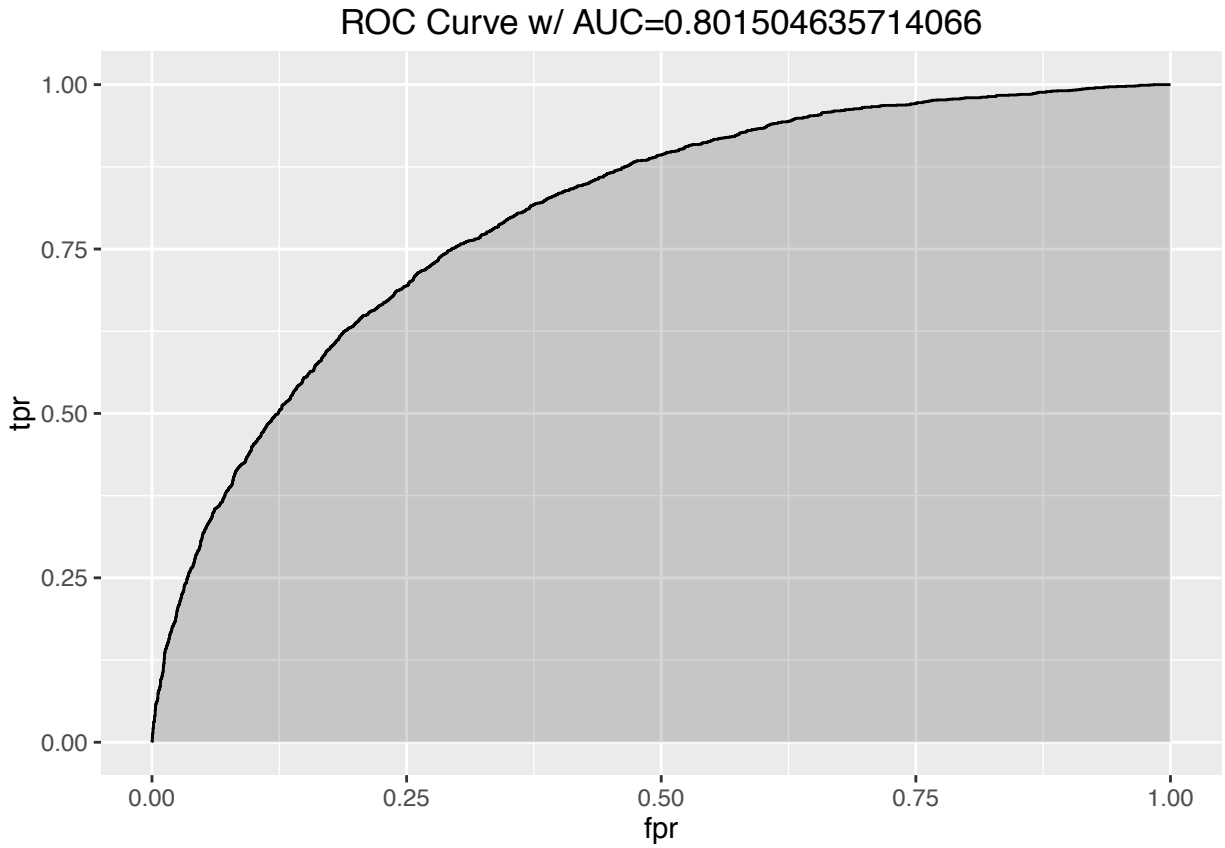




| | Estimate | Std. Error | z value | Pr(>\|z\|) |
|---|---|---|---|---|
| **KIDSDRIV** | 0.4068 | 0.05429 | 7.493 | 6.742e-14 |
| **INCOME** | -5.011e-06 | 9.629e-07 | -5.204 | 1.946e-07 |
| **PARENT1** | 0.485 | 0.09237 | 5.251 | 1.515e-07 |
| **HOME_VAL** | -1.296e-06 | 3.284e-07 | -3.947 | 7.9e-05 |
| **MSTATUS** | -0.4433 | 0.07784 | -5.695 | 1.236e-08 |
| **TRAVTIME** | 0.0148 | 0.001852 | 7.995 | 1.296e-15 |
| **CAR_USE** | 0.8168 | 0.06601 | 12.37 | 3.598e-35 |
| **BLUEBOOK** | -3.196e-05 | 4.611e-06 | -6.93 | 4.198e-12 |
| **TIF** | -0.05354 | 0.007238 | -7.398 | 1.387e-13 |
| **OLDCLAIM** | -1.41e-05 | 3.848e-06 | -3.663 | 0.0002491 |
| **CLM_FREQ** | 0.2016 | 0.02814 | 7.164 | 7.848e-13 |
| **REVOKED** | 0.9073 | 0.0898 | 10.1 | 5.324e-24 |
| **MVR_PTS** | 0.1218 | 0.01344 | 9.063 | 1.265e-19 |

| | Estimate | Std. Error | z value | Pr(>\|z\|) |
|---|---|---|---|---|
| **URBANICITY** | 2.277 | 0.1117 | 20.39 | 2.143e-92 |
| **HSDropout** | 0.5508 | 0.09131 | 6.032 | 1.617e-09 |
| **HS** | 0.5046 | 0.07275 | 6.936 | 4.029e-12 |
| **Panel_Truck** | 0.1874 | 0.1315 | 1.425 | 0.1541 |
| **Van** | 0.1751 | 0.1061 | 1.65 | 0.09894 |
| **PhD** | 0.01237 | 0.1201 | 0.103 | 0.918 |
| **(Intercept)** | -3.253 | 0.1667 | -19.51 | 8.554e-85 |

(Dispersion parameter for binomial family taken to be 1 )

| | |
|---|---|
| Null deviance: | 9418 on 8160 degrees of freedom |
| Residual deviance: | 7471 on 8141 degrees of freedom |




The best subset model has an area under the curve of 0.8015, and an accuracy of 0.7825

| | 0 | 1 |
|---|---|---|
| **0** | 5562 | 446 |
| **1** | 1329 | 824 |

# 4 Model Selection & Prediction

## 4.1 10-fold Cross Validation

The model using only the significant predictors as determined by AIC (Model 1) was selected as the best model for prediction of `TARGET_AMOUNT` in this data set. While the AIC value of this model was not the highest of the models tested, its mean cross-validation error indicates that it is has the best predictive value for unseen data. Additionally, it is a parsimonious

model, and the simplicity lends itself to easier understanding of the model by other users.

#### 4.1.0.0.1 Mean CV Error

| | |
|---|---|
| **MLR Model 1** | 60152520 |
| **MLR Model 2** | 62386094 |
| **MLR Model 3** | 62402465 |
| **MLR Model 4** | 62290747 |
| **MLR Model 5** | 62359927 |
| **Logistic Model 1** | 4.476 |
| **Logistic Model 2** | 4.518 |
| **Logistic Model 3** | 4.304 |

The linear model is applied to a test dataset containing response variables for 2141 cases. A table of the predicted team wins is presented below.

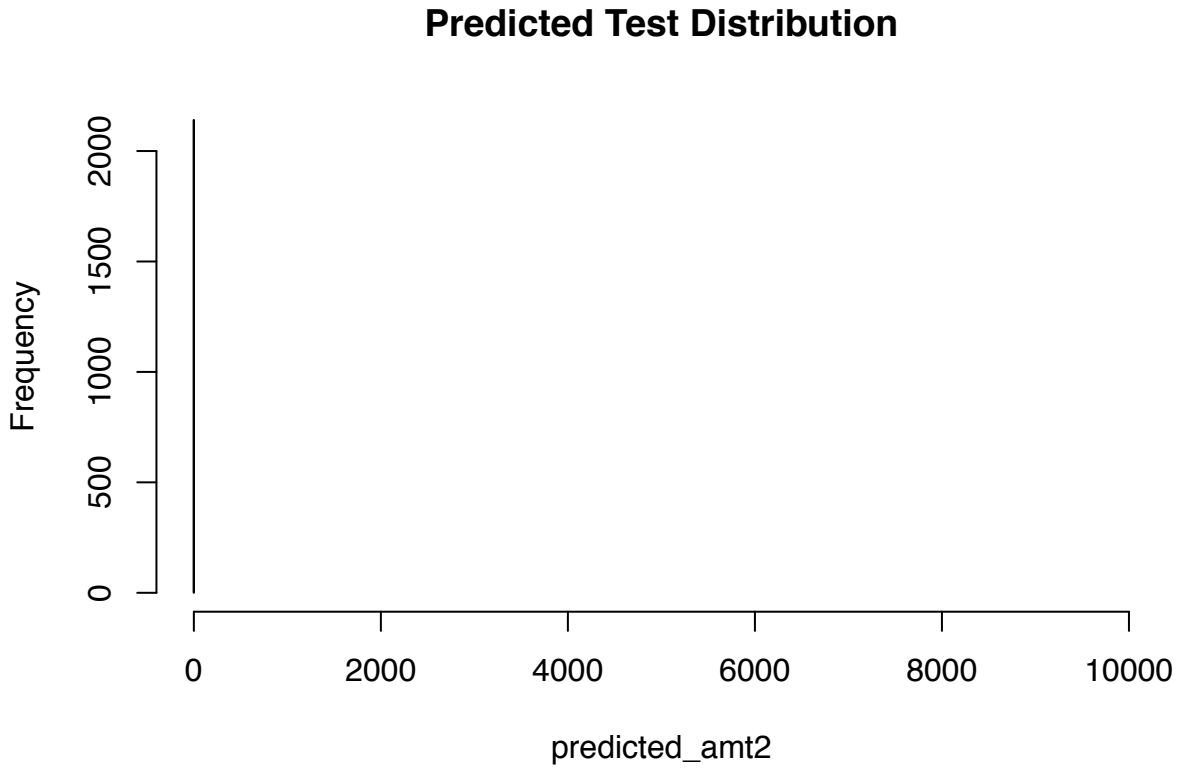
```
Error: object 'predicted_flag_bin' not found
```

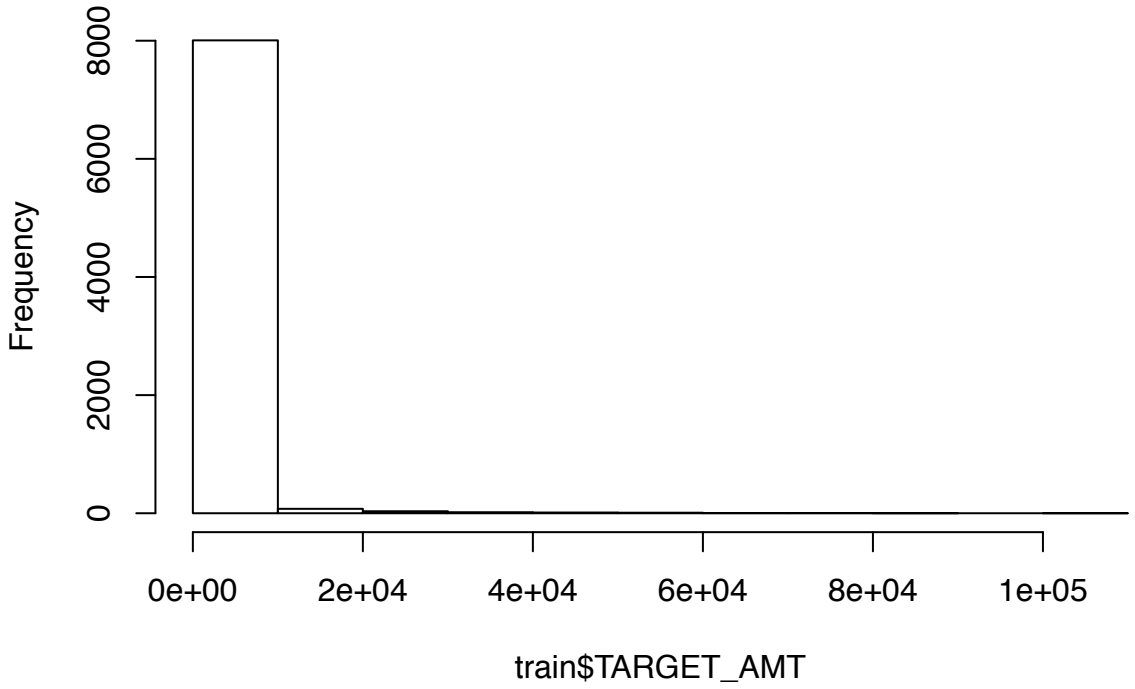
#### 4.1.0.0.2 Predicted Classifications for Test Data, Compared to Training Data

| 0 | 1 |
|---|---|
| 1599 | 313 |

| 0 | 1 |
|---|---|
| 6008 | 2153 |

#### 4.1.0.0.3 Predicted Amounts for Test Data, Compared to Training Data






Similar to the training dataset, the predictions for the test data set predictions are weighted toward zero. It is important to note that Model 1 did not achieve the highest AIC on the training data but, given its cross-validation performance, we believe that Model 1 will produce the greatest generalizability of the models explored here for prediction of `TARGET_AMOUNT`. Likewise, given the Cross Validation results, we believe that Logistic Model 3 would provide the best generalizability for the classifcation task.

A comparison of the full sets of predictions for the evaluation dataset is available in Appendix A.

# Appendix A: Index-wise Results from Predictive Model

| Index | Predicted Probability | Predicted Classifcation | Predicted Cost |
|---|---|---|---|
| 3 | 0.2299 | 0 | 0 |
| 9 | 0.4785 | 0 | 0 |
| 10 | 0.1265 | 0 | 0 |
| 18 | 0.2745 | 0 | 0 |
| 21 | 0.3585 | 0 | 0 |
| 30 | NA | NA | 0 |
| 31 | 0.3362 | 0 | 0 |
| 37 | 0.4575 | 0 | 0 |
| 39 | 0.03772 | 0 | 0 |
| 47 | 0.08512 | 0 | 0 |
| 60 | 0.0454 | 0 | 0 |
| 62 | 0.711 | 1 | 5688 |
| 63 | NA | NA | 0 |
| 64 | 0.155 | 0 | 0 |
| 68 | 0.0564 | 0 | 0 |
| 75 | NA | NA | 0 |
| 76 | 0.6276 | 1 | NA |
| 83 | 0.2066 | 0 | 0 |
| 87 | 0.4715 | 0 | 0 |
| 92 | 0.3028 | 0 | 0 |
| 98 | 0.1526 | 0 | 0 |
| 106 | 0.2729 | 0 | 0 |
| 107 | 0.1046 | 0 | 0 |
| 113 | 0.305 | 0 | 0 |
| 120 | 0.2551 | 0 | 0 |
| 123 | 0.3679 | 0 | 0 |
| 125 | 0.4203 | 0 | 0 |
| 126 | 0.2604 | 0 | 0 |
| 128 | 0.1115 | 0 | 0 |
| 129 | 0.1037 | 0 | 0 |
| 131 | 0.1684 | 0 | 0 |
| 135 | 0.3891 | 0 | 0 |
| 141 | 0.1265 | 0 | 0 |
| 147 | 0.1981 | 0 | 0 |
| 148 | 0.1434 | 0 | 0 |
| 151 | 0.03228 | 0 | 0 |
| 156 | 0.2221 | 0 | 0 |
| 157 | 0.151 | 0 | 0 |
| 174 | 0.09574 | 0 | 0 |
| 186 | 0.5354 | 1 | 5456 |
| 193 | 0.2922 | 0 | 0 |
| 195 | 0.5302 | 1 | 5183 |
| 212 | 0.01951 | 0 | 0 |
| 213 | 0.4097 | 0 | 0 |
| 217 | 0.007808 | 0 | 0 |
| 223 | 0.1942 | 0 | 0 |
| 226 | 0.07365 | 0 | 0 |
| 228 | 0.3156 | 0 | 0 |
| 230 | 0.01214 | 0 | 0 |
| 241 | 0.5801 | 1 | 4134 |
| 243 | 0.1466 | 0 | 0 |
| 249 | 0.2083 | 0 | 0 |
| 281 | 0.7915 | 1 | 5901 |
| 288 | 0.1664 | 0 | 0 |
| 294 | NA | NA | 0 |
| 295 | 0.2786 | 0 | 0 |

| Index | Predicted Probability | Predicted Classifcation | Predicted Cost |
|---|---|---|---|
| 300 | 0.3302 | 0 | 0 |
| 302 | 0.3491 | 0 | 0 |
| 303 | 0.1295 | 0 | 0 |
| 308 | 0.5452 | 1 | 4352 |
| 319 | 0.01326 | 0 | 0 |
| 320 | 0.07282 | 0 | 0 |
| 324 | 0.409 | 0 | 0 |
| 331 | 0.2785 | 0 | 0 |
| 343 | 0.08062 | 0 | 0 |
| 347 | 0.262 | 0 | 0 |
| 348 | 0.8178 | 1 | 4580 |
| 350 | 0.7081 | 1 | 6971 |
| 357 | 0.1501 | 0 | 0 |
| 358 | 0.03717 | 0 | 0 |
| 360 | NA | NA | 0 |
| 366 | 0.2893 | 0 | 0 |
| 367 | 0.7016 | 1 | 5169 |
| 368 | 0.3267 | 0 | 0 |
| 376 | 0.7039 | 1 | 4759 |
| 380 | 0.3274 | 0 | 0 |
| 388 | 0.3139 | 0 | 0 |
| 396 | 0.3736 | 0 | 0 |
| 398 | 0.1981 | 0 | 0 |
| 403 | 0.0443 | 0 | 0 |
| 410 | 0.5396 | 1 | 4331 |
| 412 | 0.3081 | 0 | 0 |
| 420 | 0.3034 | 0 | 0 |
| 434 | 0.04565 | 0 | 0 |
| 440 | 0.4826 | 0 | 0 |
| 450 | 0.6142 | 1 | 7381 |
| 453 | 0.3592 | 0 | 0 |
| 464 | 0.2498 | 0 | 0 |
| 465 | 0.1017 | 0 | 0 |
| 466 | NA | NA | 0 |
| 473 | 0.1063 | 0 | 0 |
| 476 | 0.06112 | 0 | 0 |
| 478 | NA | NA | 0 |
| 479 | 0.1486 | 0 | 0 |
| 493 | 0.04667 | 0 | 0 |
| 497 | 0.2285 | 0 | 0 |
| 503 | 0.01529 | 0 | 0 |
| 504 | 0.2125 | 0 | 0 |
| 505 | 0.3508 | 0 | 0 |
| 507 | 0.2584 | 0 | 0 |
| 513 | 0.3616 | 0 | 0 |
| 519 | NA | NA | 0 |
| 521 | 0.6172 | 1 | 5271 |
| 522 | 0.6996 | 1 | 5741 |
| 545 | NA | NA | 0 |
| 549 | 0.1079 | 0 | 0 |
| 551 | 0.1815 | 0 | 0 |
| 556 | 0.08834 | 0 | 0 |
| 557 | 0.3432 | 0 | 0 |
| 559 | 0.3096 | 0 | 0 |
| 560 | 0.6505 | 1 | 4399 |
| 566 | 0.04631 | 0 | 0 |
| 569 | 0.1906 | 0 | 0 |
| 573 | 0.2677 | 0 | 0 |

| Index | Predicted Probability | Predicted Classifcation | Predicted Cost |
|---|---|---|---|
| 578 | NA | NA | 0 |
| 579 | 0.02793 | 0 | 0 |
| 582 | 0.01888 | 0 | 0 |
| 596 | 0.5925 | 1 | 5386 |
| 598 | 0.6976 | 1 | 2138 |
| 599 | 0.1961 | 0 | 0 |
| 602 | 0.2033 | 0 | 0 |
| 605 | 0.803 | 1 | 4865 |
| 617 | 0.5971 | 1 | 4256 |
| 619 | NA | NA | 0 |
| 630 | 0.227 | 0 | 0 |
| 634 | 0.6112 | 1 | 5120 |
| 643 | 0.3907 | 0 | 0 |
| 645 | 0.1676 | 0 | 0 |
| 647 | 0.3602 | 0 | 0 |
| 649 | 0.1246 | 0 | 0 |
| 656 | 0.0818 | 0 | 0 |
| 657 | 0.1073 | 0 | 0 |
| 658 | 0.1793 | 0 | 0 |
| 667 | 0.07184 | 0 | 0 |
| 692 | 0.1725 | 0 | 0 |
| 693 | NA | NA | 0 |
| 698 | 0.7779 | 1 | 6230 |
| 699 | 0.5132 | 1 | 8366 |
| 700 | 0.04881 | 0 | 0 |
| 704 | 0.101 | 0 | 0 |
| 707 | 0.1084 | 0 | 0 |
| 708 | 0.6346 | 1 | 5018 |
| 709 | 0.1803 | 0 | 0 |
| 713 | 0.1272 | 0 | 0 |
| 714 | 0.09191 | 0 | 0 |
| 716 | 0.604 | 1 | 4182 |
| 718 | 0.1274 | 0 | 0 |
| 722 | NA | NA | 0 |
| 729 | 0.4816 | 0 | 0 |
| 731 | NA | NA | 0 |
| 733 | 0.6503 | 1 | 4956 |
| 746 | NA | NA | 0 |
| 747 | NA | NA | 0 |
| 748 | 0.4175 | 0 | 0 |
| 753 | 0.1897 | 0 | 0 |
| 757 | 0.5713 | 1 | 5630 |
| 763 | NA | NA | 0 |
| 767 | 0.1267 | 0 | 0 |
| 774 | 0.6157 | 1 | 3558 |
| 776 | 0.6023 | 1 | 4794 |
| 788 | 0.1629 | 0 | 0 |
| 794 | 0.1684 | 0 | 0 |
| 799 | 0.2102 | 0 | 0 |
| 803 | 0.2676 | 0 | 0 |
| 806 | 0.6539 | 1 | 7385 |
| 807 | 0.1995 | 0 | 0 |
| 811 | 0.04843 | 0 | 0 |
| 816 | NA | NA | 0 |
| 818 | 0.3013 | 0 | 0 |
| 819 | 0.1061 | 0 | 0 |
| 831 | 0.1178 | 0 | 0 |
| 835 | 0.5386 | 1 | 7133 |

| Index | Predicted Probability | Predicted Classifcation | Predicted Cost |
|---|---|---|---|
| 837 | NA | NA | 0 |
| 841 | 0.826 | 1 | 6469 |
| 846 | 0.2846 | 0 | 0 |
| 856 | 0.3054 | 0 | 0 |
| 861 | NA | NA | 0 |
| 862 | 0.579 | 1 | 5662 |
| 863 | 0.5846 | 1 | 8139 |
| 865 | 0.6753 | 1 | 3994 |
| 871 | 0.6648 | 1 | 6278 |
| 879 | 0.1532 | 0 | 0 |
| 880 | 0.1102 | 0 | 0 |
| 881 | 0.2622 | 0 | 0 |
| 885 | 0.2301 | 0 | 0 |
| 887 | 0.2512 | 0 | 0 |
| 892 | 0.0811 | 0 | 0 |
| 898 | 0.1017 | 0 | 0 |
| 900 | 0.2329 | 0 | 0 |
| 904 | 0.3036 | 0 | 0 |
| 906 | 0.5095 | 1 | 3916 |
| 910 | 0.6261 | 1 | 4745 |
| 912 | NA | NA | 0 |
| 913 | 0.5013 | 1 | 3324 |
| 919 | 0.1174 | 0 | 0 |
| 924 | 0.4228 | 0 | 0 |
| 925 | 0.2741 | 0 | 0 |
| 930 | 0.1957 | 0 | 0 |
| 940 | 0.1756 | 0 | 0 |
| 941 | 0.1202 | 0 | 0 |
| 946 | 0.1635 | 0 | 0 |
| 949 | 0.2732 | 0 | 0 |
| 951 | 0.08371 | 0 | 0 |
| 962 | 0.08464 | 0 | 0 |
| 966 | 0.1027 | 0 | 0 |
| 967 | 0.01183 | 0 | 0 |
| 971 | NA | NA | 0 |
| 981 | 0.02554 | 0 | 0 |
| 982 | 0.09984 | 0 | 0 |
| 983 | 0.09437 | 0 | 0 |
| 984 | 0.03188 | 0 | 0 |
| 989 | 0.2716 | 0 | 0 |
| 990 | 0.5508 | 1 | 4327 |
| 992 | 0.4465 | 0 | 0 |
| 995 | 0.0447 | 0 | 0 |
| 996 | 0.2833 | 0 | 0 |
| 998 | 0.4501 | 0 | 0 |
| 1001 | 0.1027 | 0 | 0 |
| 1007 | 0.1646 | 0 | 0 |
| 1008 | 0.03318 | 0 | 0 |
| 1016 | 0.05257 | 0 | 0 |
| 1022 | 0.06743 | 0 | 0 |
| 1027 | 0.5686 | 1 | 5976 |
| 1032 | 0.2939 | 0 | 0 |
| 1033 | 0.2957 | 0 | 0 |
| 1041 | 0.3579 | 0 | 0 |
| 1065 | 0.7336 | 1 | 5883 |
| 1074 | 0.4697 | 0 | 0 |
| 1075 | 0.3958 | 0 | 0 |
| 1081 | 0.1002 | 0 | 0 |

| Index | Predicted Probability | Predicted Classifcation | Predicted Cost |
|---|---|---|---|
| 1094 | 0.0482 | 0 | 0 |
| 1099 | 0.09574 | 0 | 0 |
| 1105 | 0.5321 | 1 | 8015 |
| 1123 | 0.05724 | 0 | 0 |
| 1135 | 0.005405 | 0 | 0 |
| 1142 | 0.3156 | 0 | 0 |
| 1155 | 0.1181 | 0 | 0 |
| 1169 | 0.0352 | 0 | 0 |
| 1176 | 0.04728 | 0 | 0 |
| 1178 | 0.542 | 1 | 6336 |
| 1180 | 0.04822 | 0 | 0 |
| 1184 | NA | NA | 0 |
| 1185 | 0.6942 | 1 | 6897 |
| 1193 | 0.2719 | 0 | 0 |
| 1196 | 0.1802 | 0 | 0 |
| 1199 | 0.3981 | 0 | 0 |
| 1203 | 0.2796 | 0 | 0 |
| 1205 | 0.3067 | 0 | 0 |
| 1207 | 0.03834 | 0 | 0 |
| 1208 | 0.649 | 1 | 4465 |
| 1212 | NA | NA | 0 |
| 1213 | NA | NA | 0 |
| 1222 | 0.08023 | 0 | 0 |
| 1223 | 0.275 | 0 | 0 |
| 1226 | 0.3298 | 0 | 0 |
| 1227 | 0.2948 | 0 | 0 |
| 1229 | NA | NA | 0 |
| 1230 | 0.2224 | 0 | 0 |
| 1231 | 0.4392 | 0 | 0 |
| 1241 | 0.1034 | 0 | 0 |
| 1243 | 0.04095 | 0 | 0 |
| 1244 | 0.2394 | 0 | 0 |
| 1246 | 0.1607 | 0 | 0 |
| 1248 | NA | NA | 0 |
| 1249 | 0.2583 | 0 | 0 |
| 1252 | 0.1276 | 0 | 0 |
| 1261 | 0.07548 | 0 | 0 |
| 1275 | 0.1549 | 0 | 0 |
| 1281 | 0.8907 | 1 | 6666 |
| 1285 | 0.3488 | 0 | 0 |
| 1288 | 0.481 | 0 | 0 |
| 1290 | 0.07012 | 0 | 0 |
| 1291 | 0.2093 | 0 | 0 |
| 1304 | NA | NA | 0 |
| 1305 | 0.1127 | 0 | 0 |
| 1323 | 0.2519 | 0 | 0 |
| 1342 | NA | NA | 0 |
| 1348 | NA | NA | 0 |
| 1353 | 0.2225 | 0 | 0 |
| 1363 | 0.1995 | 0 | 0 |
| 1371 | 0.2926 | 0 | 0 |
| 1372 | 0.2252 | 0 | 0 |
| 1378 | 0.2055 | 0 | 0 |
| 1381 | 0.3177 | 0 | 0 |
| 1382 | 0.2964 | 0 | 0 |
| 1393 | 0.5309 | 1 | 5471 |
| 1394 | 0.02909 | 0 | 0 |
| 1398 | 0.23 | 0 | 0 |

| Index | Predicted Probability | Predicted Classifcation | Predicted Cost |
|---|---|---|---|
| 1404 | 0.5087 | 1 | 4551 |
| 1405 | 0.7363 | 1 | 4906 |
| 1419 | 0.2155 | 0 | 0 |
| 1421 | 0.1949 | 0 | 0 |
| 1426 | 0.07789 | 0 | 0 |
| 1431 | 0.4544 | 0 | 0 |
| 1435 | 0.03864 | 0 | 0 |
| 1437 | 0.3595 | 0 | 0 |
| 1438 | 0.1717 | 0 | 0 |
| 1442 | 0.4258 | 0 | 0 |
| 1464 | 0.2207 | 0 | 0 |
| 1471 | 0.3437 | 0 | 0 |
| 1473 | 0.2598 | 0 | 0 |
| 1476 | 0.03502 | 0 | 0 |
| 1478 | 0.3105 | 0 | 0 |
| 1479 | 0.4407 | 0 | 0 |
| 1487 | 0.4354 | 0 | 0 |
| 1492 | 0.3792 | 0 | 0 |
| 1496 | 0.1481 | 0 | 0 |
| 1497 | 0.4135 | 0 | 0 |
| 1515 | 0.01855 | 0 | 0 |
| 1519 | 0.2265 | 0 | 0 |
| 1522 | NA | NA | 0 |
| 1526 | NA | NA | 0 |
| 1537 | 0.07045 | 0 | 0 |
| 1538 | 0.9058 | 1 | 5564 |
| 1540 | 0.1602 | 0 | 0 |
| 1543 | 0.1661 | 0 | 0 |
| 1548 | 0.2209 | 0 | 0 |
| 1549 | 0.1649 | 0 | 0 |
| 1556 | 0.6133 | 1 | 5073 |
| 1564 | 0.02741 | 0 | 0 |
| 1570 | 0.2745 | 0 | 0 |
| 1577 | 0.5922 | 1 | 4901 |
| 1585 | 0.2981 | 0 | 0 |
| 1590 | 0.2484 | 0 | 0 |
| 1592 | 0.6388 | 1 | 4856 |
| 1594 | 0.3407 | 0 | 0 |
| 1596 | NA | NA | 0 |
| 1598 | NA | NA | 0 |
| 1603 | NA | NA | 0 |
| 1607 | 0.1504 | 0 | 0 |
| 1612 | NA | NA | 0 |
| 1627 | NA | NA | 0 |
| 1629 | 0.7772 | 1 | 5352 |
| 1630 | 0.1528 | 0 | 0 |
| 1640 | 0.55 | 1 | 4875 |
| 1641 | 0.3781 | 0 | 0 |
| 1646 | NA | NA | 0 |
| 1662 | 0.5494 | 1 | 4518 |
| 1668 | NA | NA | 0 |
| 1671 | 0.04603 | 0 | 0 |
| 1672 | 0.5323 | 1 | 4560 |
| 1673 | 0.6551 | 1 | 6878 |
| 1686 | 0.3641 | 0 | 0 |
| 1688 | 0.5925 | 1 | 4242 |
| 1696 | 0.02242 | 0 | 0 |
| 1701 | 0.0472 | 0 | 0 |

| Index | Predicted Probability | Predicted Classifcation | Predicted Cost |
|---|---|---|---|
| 1707 | 0.1644 | 0 | 0 |
| 1708 | 0.04402 | 0 | 0 |
| 1713 | 0.0422 | 0 | 0 |
| 1715 | 0.1367 | 0 | 0 |
| 1717 | 0.03208 | 0 | 0 |
| 1721 | 0.3099 | 0 | 0 |
| 1724 | NA | NA | 0 |
| 1725 | 0.7121 | 1 | 6872 |
| 1730 | 0.06584 | 0 | 0 |
| 1731 | 0.6001 | 1 | 3634 |
| 1734 | 0.335 | 0 | 0 |
| 1740 | 0.1538 | 0 | 0 |
| 1748 | 0.05553 | 0 | 0 |
| 1749 | 0.05327 | 0 | 0 |
| 1750 | 0.5998 | 1 | 5039 |
| 1763 | 0.1571 | 0 | 0 |
| 1768 | 0.05795 | 0 | 0 |
| 1773 | 0.6105 | 1 | 4236 |
| 1777 | 0.2188 | 0 | 0 |
| 1778 | 0.45 | 0 | 0 |
| 1780 | 0.1093 | 0 | 0 |
| 1782 | 0.25 | 0 | 0 |
| 1784 | NA | NA | 0 |
| 1786 | 0.1525 | 0 | 0 |
| 1787 | 0.09853 | 0 | 0 |
| 1792 | 0.2588 | 0 | 0 |
| 1800 | 0.4474 | 0 | 0 |
| 1801 | 0.2816 | 0 | 0 |
| 1803 | 0.1342 | 0 | 0 |
| 1804 | 0.6527 | 1 | 4107 |
| 1807 | 0.01554 | 0 | 0 |
| 1818 | NA | NA | 0 |
| 1821 | 0.02862 | 0 | 0 |
| 1822 | 0.07757 | 0 | 0 |
| 1828 | 0.0629 | 0 | 0 |
| 1833 | 0.2536 | 0 | 0 |
| 1844 | 0.3419 | 0 | 0 |
| 1847 | 0.3262 | 0 | 0 |
| 1850 | 0.1427 | 0 | 0 |
| 1854 | NA | NA | 0 |
| 1858 | NA | NA | 0 |
| 1864 | 0.1521 | 0 | 0 |
| 1867 | 0.1119 | 0 | 0 |
| 1876 | 0.754 | 1 | 4662 |
| 1880 | 0.2334 | 0 | 0 |
| 1881 | 0.2289 | 0 | 0 |
| 1891 | 0.1488 | 0 | 0 |
| 1894 | 0.1718 | 0 | 0 |
| 1895 | 0.04439 | 0 | 0 |
| 1901 | 0.3171 | 0 | 0 |
| 1905 | 0.03748 | 0 | 0 |
| 1912 | 0.2607 | 0 | 0 |
| 1918 | 0.4066 | 0 | 0 |
| 1921 | NA | NA | 0 |
| 1923 | 0.2135 | 0 | 0 |
| 1924 | NA | NA | 0 |
| 1931 | 0.1317 | 0 | 0 |
| 1941 | 0.09363 | 0 | 0 |

| Index | Predicted Probability | Predicted Classifcation | Predicted Cost |
|---|---|---|---|
| 1950 | 0.0666 | 0 | 0 |
| 1951 | 0.1551 | 0 | 0 |
| 1954 | 0.02677 | 0 | 0 |
| 1961 | 0.4458 | 0 | 0 |
| 1966 | 0.01902 | 0 | 0 |
| 1979 | 0.1004 | 0 | 0 |
| 1982 | NA | NA | 0 |
| 1987 | 0.743 | 1 | 4052 |
| 1997 | 0.4535 | 0 | 0 |
| 2004 | 0.04097 | 0 | 0 |
| 2011 | 0.6041 | 1 | 4646 |
| 2015 | 0.2506 | 0 | 0 |
| 2025 | 0.01711 | 0 | 0 |
| 2033 | 0.2488 | 0 | 0 |
| 2034 | 0.008388 | 0 | 0 |
| 2035 | 0.1198 | 0 | 0 |
| 2036 | 0.4453 | 0 | 0 |
| 2053 | 0.6898 | 1 | 7972 |
| 2059 | 0.7235 | 1 | 6821 |
| 2060 | 0.01933 | 0 | 0 |
| 2073 | NA | NA | 0 |
| 2084 | 0.4465 | 0 | 0 |
| 2089 | 0.07392 | 0 | 0 |
| 2092 | 0.1969 | 0 | 0 |
| 2109 | 0.4512 | 0 | 0 |
| 2129 | 0.2513 | 0 | 0 |
| 2134 | 0.3149 | 0 | 0 |
| 2135 | 0.02947 | 0 | 0 |
| 2148 | 0.02625 | 0 | 0 |
| 2149 | NA | NA | 0 |
| 2150 | 0.3305 | 0 | 0 |
| 2165 | 0.7297 | 1 | 6775 |
| 2166 | 0.04809 | 0 | 0 |
| 2168 | 0.07046 | 0 | 0 |
| 2170 | 0.2554 | 0 | 0 |
| 2171 | 0.1062 | 0 | 0 |
| 2172 | 0.03118 | 0 | 0 |
| 2176 | 0.2208 | 0 | 0 |
| 2182 | 0.0649 | 0 | 0 |
| 2189 | 0.1498 | 0 | 0 |
| 2191 | 0.08038 | 0 | 0 |
| 2197 | 0.02079 | 0 | 0 |
| 2202 | 0.09998 | 0 | 0 |
| 2203 | 0.1312 | 0 | 0 |
| 2204 | 0.5773 | 1 | 5885 |
| 2206 | 0.7187 | 1 | 5982 |
| 2218 | 0.05236 | 0 | 0 |
| 2219 | 0.2112 | 0 | 0 |
| 2221 | 0.4666 | 0 | 0 |
| 2226 | 0.1108 | 0 | 0 |
| 2228 | NA | NA | 0 |
| 2232 | 0.6331 | 1 | 6093 |
| 2236 | 0.3778 | 0 | 0 |
| 2241 | 0.8309 | 1 | 6194 |
| 2245 | 0.09843 | 0 | 0 |
| 2251 | 0.2301 | 0 | 0 |
| 2255 | 0.04828 | 0 | 0 |
| 2256 | 0.0193 | 0 | 0 |

| Index | Predicted Probability | Predicted Classifcation | Predicted Cost |
|---|---|---|---|
| 2259 | 0.03198 | 0 | 0 |
| 2263 | 0.373 | 0 | 0 |
| 2264 | 0.07659 | 0 | 0 |
| 2267 | 0.1332 | 0 | 0 |
| 2273 | 0.7754 | 1 | 5930 |
| 2277 | 0.5577 | 1 | 2901 |
| 2287 | 0.2155 | 0 | 0 |
| 2289 | NA | NA | 0 |
| 2291 | NA | NA | 0 |
| 2296 | 0.8123 | 1 | 4021 |
| 2299 | 0.08981 | 0 | 0 |
| 2306 | 0.04288 | 0 | 0 |
| 2314 | 0.3172 | 0 | 0 |
| 2317 | 0.01167 | 0 | 0 |
| 2318 | 0.6482 | 1 | 6811 |
| 2321 | 0.8607 | 1 | 4785 |
| 2324 | 0.03378 | 0 | 0 |
| 2340 | NA | NA | 0 |
| 2343 | NA | NA | 0 |
| 2349 | 0.3177 | 0 | 0 |
| 2352 | 0.254 | 0 | 0 |
| 2353 | 0.3927 | 0 | 0 |
| 2365 | 0.7447 | 1 | 4408 |
| 2370 | 0.4583 | 0 | 0 |
| 2378 | 0.347 | 0 | 0 |
| 2390 | 0.3579 | 0 | 0 |
| 2399 | 0.2432 | 0 | 0 |
| 2402 | 0.7673 | 1 | 5465 |
| 2403 | 0.4675 | 0 | 0 |
| 2404 | 0.1342 | 0 | 0 |
| 2414 | 0.2642 | 0 | 0 |
| 2422 | 0.29 | 0 | 0 |
| 2424 | 0.1848 | 0 | 0 |
| 2430 | 0.5304 | 1 | 3467 |
| 2435 | NA | NA | 0 |
| 2439 | 0.05404 | 0 | 0 |
| 2442 | NA | NA | 0 |
| 2445 | 0.4158 | 0 | 0 |
| 2449 | 0.2499 | 0 | 0 |
| 2451 | NA | NA | 0 |
| 2461 | 0.7464 | 1 | 5100 |
| 2464 | NA | NA | 0 |
| 2465 | 0.5908 | 1 | 6942 |
| 2472 | 0.07099 | 0 | 0 |
| 2476 | 0.5588 | 1 | 5024 |
| 2482 | 0.1987 | 0 | 0 |
| 2487 | 0.1513 | 0 | 0 |
| 2498 | 0.1856 | 0 | 0 |
| 2501 | 0.09822 | 0 | 0 |
| 2504 | 0.2895 | 0 | 0 |
| 2511 | NA | NA | 0 |
| 2518 | 0.03951 | 0 | 0 |
| 2521 | 0.1298 | 0 | 0 |
| 2530 | 0.2025 | 0 | 0 |
| 2543 | NA | NA | 0 |
| 2545 | 0.3075 | 0 | 0 |
| 2561 | 0.3693 | 0 | 0 |
| 2566 | 0.4983 | 0 | 0 |

| Index | Predicted Probability | Predicted Classifcation | Predicted Cost |
|---|---|---|---|
| 2572 | 0.2654 | 0 | 0 |
| 2577 | 0.1446 | 0 | 0 |
| 2578 | 0.2544 | 0 | 0 |
| 2580 | 0.3266 | 0 | 0 |
| 2581 | 0.293 | 0 | 0 |
| 2582 | 0.1306 | 0 | 0 |
| 2584 | 0.07339 | 0 | 0 |
| 2590 | 0.02218 | 0 | 0 |
| 2598 | 0.01151 | 0 | 0 |
| 2602 | 0.1521 | 0 | 0 |
| 2605 | 0.01511 | 0 | 0 |
| 2616 | 0.1632 | 0 | 0 |
| 2618 | 0.2534 | 0 | 0 |
| 2619 | 0.3259 | 0 | 0 |
| 2624 | 0.06301 | 0 | 0 |
| 2632 | 0.1684 | 0 | 0 |
| 2640 | 0.3495 | 0 | 0 |
| 2646 | 0.01876 | 0 | 0 |
| 2651 | 0.1386 | 0 | 0 |
| 2660 | 0.04379 | 0 | 0 |
| 2661 | 0.02685 | 0 | 0 |
| 2668 | 0.05735 | 0 | 0 |
| 2670 | 0.3152 | 0 | 0 |
| 2680 | 0.4633 | 0 | 0 |
| 2681 | 0.01341 | 0 | 0 |
| 2689 | 0.3767 | 0 | 0 |
| 2694 | NA | NA | 0 |
| 2695 | 0.8022 | 1 | 7500 |
| 2696 | 0.4301 | 0 | 0 |
| 2702 | 0.03872 | 0 | 0 |
| 2704 | 0.1181 | 0 | 0 |
| 2708 | 0.07601 | 0 | 0 |
| 2709 | 0.03237 | 0 | 0 |
| 2714 | 0.4149 | 0 | 0 |
| 2716 | 0.1974 | 0 | 0 |
| 2723 | 0.1139 | 0 | 0 |
| 2725 | 0.3293 | 0 | 0 |
| 2738 | 0.05028 | 0 | 0 |
| 2750 | 0.3981 | 0 | 0 |
| 2756 | 0.2886 | 0 | 0 |
| 2758 | 0.05219 | 0 | 0 |
| 2766 | 0.1756 | 0 | 0 |
| 2767 | 0.2812 | 0 | 0 |
| 2771 | 0.3281 | 0 | 0 |
| 2775 | 0.6042 | 1 | 3579 |
| 2776 | 0.1958 | 0 | 0 |
| 2779 | 0.915 | 1 | 6063 |
| 2780 | 0.3504 | 0 | 0 |
| 2781 | 0.2776 | 0 | 0 |
| 2782 | 0.5208 | 1 | 6675 |
| 2783 | 0.2029 | 0 | 0 |
| 2796 | NA | NA | 0 |
| 2798 | 0.4121 | 0 | 0 |
| 2800 | 0.09902 | 0 | 0 |
| 2803 | 0.1389 | 0 | 0 |
| 2806 | 0.005981 | 0 | 0 |
| 2813 | 0.09904 | 0 | 0 |
| 2818 | 0.08394 | 0 | 0 |

| Index | Predicted Probability | Predicted Classifcation | Predicted Cost |
|---|---|---|---|
| 2821 | NA | NA | 0 |
| 2825 | 0.3064 | 0 | 0 |
| 2829 | 0.04146 | 0 | 0 |
| 2830 | 0.7053 | 1 | 5227 |
| 2833 | 0.06758 | 0 | 0 |
| 2839 | 0.8992 | 1 | 3878 |
| 2843 | 0.08926 | 0 | 0 |
| 2846 | 0.07887 | 0 | 0 |
| 2847 | 0.09807 | 0 | 0 |
| 2848 | 0.08112 | 0 | 0 |
| 2856 | 0.8363 | 1 | 6354 |
| 2863 | 0.3808 | 0 | 0 |
| 2867 | 0.3407 | 0 | 0 |
| 2869 | 0.192 | 0 | 0 |
| 2873 | 0.01128 | 0 | 0 |
| 2874 | 0.2989 | 0 | 0 |
| 2875 | 0.4679 | 0 | 0 |
| 2880 | NA | NA | 0 |
| 2886 | NA | NA | 0 |
| 2887 | 0.2846 | 0 | 0 |
| 2888 | 0.3363 | 0 | 0 |
| 2889 | 0.5934 | 1 | 2228 |
| 2890 | 0.453 | 0 | 0 |
| 2892 | 0.523 | 1 | 4239 |
| 2901 | 0.128 | 0 | 0 |
| 2902 | 0.2172 | 0 | 0 |
| 2905 | 0.3558 | 0 | 0 |
| 2917 | 0.2817 | 0 | 0 |
| 2922 | 0.5455 | 1 | 6764 |
| 2924 | NA | NA | 0 |
| 2930 | 0.1607 | 0 | 0 |
| 2931 | 0.09846 | 0 | 0 |
| 2946 | 0.1258 | 0 | 0 |
| 2955 | 0.4378 | 0 | 0 |
| 2962 | 0.02385 | 0 | 0 |
| 2964 | 0.01923 | 0 | 0 |
| 2965 | 0.3343 | 0 | 0 |
| 2967 | 0.03135 | 0 | 0 |
| 2970 | 0.153 | 0 | 0 |
| 2973 | NA | NA | 0 |
| 2974 | NA | NA | 0 |
| 2976 | 0.6076 | 1 | 7821 |
| 2977 | 0.3302 | 0 | 0 |
| 2978 | 0.1795 | 0 | 0 |
| 2986 | 0.1398 | 0 | 0 |
| 2988 | NA | NA | 0 |
| 2989 | 0.1312 | 0 | 0 |
| 2995 | 0.7242 | 1 | 6564 |
| 3005 | 0.583 | 1 | NA |
| 3011 | 0.1501 | 0 | 0 |
| 3013 | 0.1084 | 0 | 0 |
| 3019 | 0.5346 | 1 | 5501 |
| 3021 | 0.03381 | 0 | 0 |
| 3022 | 0.2668 | 0 | 0 |
| 3029 | NA | NA | 0 |
| 3037 | 0.1782 | 0 | 0 |
| 3042 | 0.1021 | 0 | 0 |
| 3043 | 0.1363 | 0 | 0 |

| Index | Predicted Probability | Predicted Classifcation | Predicted Cost |
|---|---|---|---|
| 3049 | NA | NA | 0 |
| 3050 | 0.5438 | 1 | 6188 |
| 3053 | 0.2299 | 0 | 0 |
| 3058 | 0.1623 | 0 | 0 |
| 3062 | 0.2771 | 0 | 0 |
| 3063 | 0.2682 | 0 | 0 |
| 3065 | 0.04461 | 0 | 0 |
| 3080 | 0.09591 | 0 | 0 |
| 3088 | 0.1703 | 0 | 0 |
| 3093 | 0.3012 | 0 | 0 |
| 3096 | 0.4077 | 0 | 0 |
| 3101 | 0.3429 | 0 | 0 |
| 3103 | 0.5317 | 1 | NA |
| 3107 | 0.2676 | 0 | 0 |
| 3109 | NA | NA | 0 |
| 3111 | 0.1111 | 0 | 0 |
| 3113 | 0.7897 | 1 | 5122 |
| 3116 | 0.007077 | 0 | 0 |
| 3132 | 0.1189 | 0 | 0 |
| 3141 | 0.1614 | 0 | 0 |
| 3153 | 0.3272 | 0 | 0 |
| 3154 | 0.08468 | 0 | 0 |
| 3160 | 0.2657 | 0 | 0 |
| 3167 | 0.07249 | 0 | 0 |
| 3170 | 0.361 | 0 | 0 |
| 3173 | 0.4895 | 0 | 0 |
| 3174 | 0.295 | 0 | 0 |
| 3177 | 0.1705 | 0 | 0 |
| 3179 | 0.2767 | 0 | 0 |
| 3184 | 0.4412 | 0 | 0 |
| 3190 | 0.3591 | 0 | 0 |
| 3193 | NA | NA | 0 |
| 3199 | 0.2279 | 0 | 0 |
| 3201 | NA | NA | 0 |
| 3202 | 0.2493 | 0 | 0 |
| 3203 | 0.5847 | 1 | 2741 |
| 3206 | 0.7322 | 1 | 4518 |
| 3209 | 0.04534 | 0 | 0 |
| 3210 | 0.3781 | 0 | 0 |
| 3217 | 0.2151 | 0 | 0 |
| 3220 | 0.08447 | 0 | 0 |
| 3228 | 0.4469 | 0 | 0 |
| 3232 | 0.03913 | 0 | 0 |
| 3239 | 0.05667 | 0 | 0 |
| 3243 | 0.3465 | 0 | 0 |
| 3245 | 0.1423 | 0 | 0 |
| 3246 | 0.3833 | 0 | 0 |
| 3251 | 0.06941 | 0 | 0 |
| 3253 | 0.4492 | 0 | 0 |
| 3257 | 0.05335 | 0 | 0 |
| 3260 | 0.02819 | 0 | 0 |
| 3261 | 0.2538 | 0 | 0 |
| 3263 | 0.1851 | 0 | 0 |
| 3278 | 0.1407 | 0 | 0 |
| 3281 | NA | NA | 0 |
| 3283 | 0.07939 | 0 | 0 |
| 3290 | 0.01956 | 0 | 0 |
| 3297 | 0.2947 | 0 | 0 |

| Index | Predicted Probability | Predicted Classifcation | Predicted Cost |
|---|---|---|---|
| 3304 | 0.05614 | 0 | 0 |
| 3305 | 0.454 | 0 | 0 |
| 3307 | NA | NA | 0 |
| 3308 | 0.2805 | 0 | 0 |
| 3313 | 0.275 | 0 | 0 |
| 3314 | 0.3831 | 0 | 0 |
| 3317 | 0.1743 | 0 | 0 |
| 3348 | 0.07017 | 0 | 0 |
| 3350 | 0.2277 | 0 | 0 |
| 3359 | 0.03217 | 0 | 0 |
| 3367 | 0.1104 | 0 | 0 |
| 3376 | 0.2008 | 0 | 0 |
| 3378 | 0.2995 | 0 | 0 |
| 3384 | 0.7648 | 1 | 5605 |
| 3386 | 0.1235 | 0 | 0 |
| 3387 | 0.1399 | 0 | 0 |
| 3388 | 0.06509 | 0 | 0 |
| 3390 | 0.03829 | 0 | 0 |
| 3391 | NA | NA | 0 |
| 3396 | 0.3428 | 0 | 0 |
| 3398 | 0.02641 | 0 | 0 |
| 3404 | 0.0666 | 0 | 0 |
| 3406 | 0.02672 | 0 | 0 |
| 3407 | 0.05707 | 0 | 0 |
| 3414 | 0.1083 | 0 | 0 |
| 3419 | NA | NA | 0 |
| 3423 | 0.5022 | 1 | 5395 |
| 3427 | 0.04676 | 0 | 0 |
| 3432 | 0.09246 | 0 | 0 |
| 3434 | 0.1134 | 0 | 0 |
| 3438 | 0.08295 | 0 | 0 |
| 3442 | 0.2844 | 0 | 0 |
| 3443 | 0.08278 | 0 | 0 |
| 3448 | 0.08944 | 0 | 0 |
| 3456 | 0.08536 | 0 | 0 |
| 3464 | 0.1998 | 0 | 0 |
| 3470 | 0.5138 | 1 | 4506 |
| 3475 | 0.4271 | 0 | 0 |
| 3477 | 0.3343 | 0 | 0 |
| 3490 | 0.1119 | 0 | 0 |
| 3493 | 0.3364 | 0 | 0 |
| 3502 | NA | NA | 0 |
| 3508 | 0.02703 | 0 | 0 |
| 3516 | 0.0553 | 0 | 0 |
| 3517 | 0.1864 | 0 | 0 |
| 3525 | 0.2695 | 0 | 0 |
| 3532 | 0.5775 | 1 | 5628 |
| 3535 | 0.235 | 0 | 0 |
| 3536 | 0.7043 | 1 | 4601 |
| 3540 | 0.07429 | 0 | 0 |
| 3547 | 0.2876 | 0 | 0 |
| 3550 | 0.3728 | 0 | 0 |
| 3557 | NA | NA | 0 |
| 3562 | 0.1409 | 0 | 0 |
| 3563 | 0.08331 | 0 | 0 |
| 3564 | 0.2427 | 0 | 0 |
| 3570 | 0.1189 | 0 | 0 |
| 3573 | 0.3985 | 0 | 0 |

| Index | Predicted Probability | Predicted Classifcation | Predicted Cost |
|---|---|---|---|
| 3577 | 0.4778 | 0 | 0 |
| 3579 | 0.5739 | 1 | NA |
| 3581 | NA | NA | 0 |
| 3587 | 0.2651 | 0 | 0 |
| 3602 | 0.3084 | 0 | 0 |
| 3609 | 0.5581 | 1 | 4301 |
| 3612 | 0.2001 | 0 | 0 |
| 3621 | 0.3502 | 0 | 0 |
| 3642 | 0.1941 | 0 | 0 |
| 3647 | 0.7175 | 1 | 6484 |
| 3649 | 0.5499 | 1 | 7868 |
| 3654 | 0.5537 | 1 | 7807 |
| 3660 | 0.5641 | 1 | 3704 |
| 3665 | 0.6805 | 1 | 4920 |
| 3669 | 0.3239 | 0 | 0 |
| 3673 | 0.2078 | 0 | 0 |
| 3675 | 0.5674 | 1 | 8358 |
| 3678 | 0.04528 | 0 | 0 |
| 3680 | 0.3921 | 0 | 0 |
| 3686 | 0.544 | 1 | 6247 |
| 3693 | 0.2518 | 0 | 0 |
| 3710 | NA | NA | 0 |
| 3713 | 0.04243 | 0 | 0 |
| 3718 | 0.3031 | 0 | 0 |
| 3725 | 0.07896 | 0 | 0 |
| 3726 | 0.3307 | 0 | 0 |
| 3747 | 0.1105 | 0 | 0 |
| 3753 | 0.01795 | 0 | 0 |
| 3754 | 0.2575 | 0 | 0 |
| 3760 | 0.8297 | 1 | 6590 |
| 3763 | 0.1092 | 0 | 0 |
| 3765 | 0.3352 | 0 | 0 |
| 3769 | 0.1903 | 0 | 0 |
| 3771 | 0.6109 | 1 | 6040 |
| 3784 | 0.09762 | 0 | 0 |
| 3787 | NA | NA | 0 |
| 3794 | 0.2638 | 0 | 0 |
| 3796 | 0.08333 | 0 | 0 |
| 3798 | 0.0973 | 0 | 0 |
| 3809 | 0.2716 | 0 | 0 |
| 3812 | 0.2933 | 0 | 0 |
| 3819 | 0.2143 | 0 | 0 |
| 3828 | 0.1197 | 0 | 0 |
| 3831 | 0.2327 | 0 | 0 |
| 3833 | 0.1504 | 0 | 0 |
| 3837 | 0.6687 | 1 | 4308 |
| 3839 | 0.7909 | 1 | 5080 |
| 3843 | 0.385 | 0 | 0 |
| 3846 | NA | NA | 0 |
| 3854 | 0.04198 | 0 | 0 |
| 3861 | 0.08067 | 0 | 0 |
| 3864 | 0.4054 | 0 | 0 |
| 3868 | 0.0952 | 0 | 0 |
| 3869 | 0.1128 | 0 | 0 |
| 3870 | 0.2103 | 0 | 0 |
| 3883 | 0.2711 | 0 | 0 |
| 3886 | 0.04617 | 0 | 0 |
| 3889 | 0.2433 | 0 | 0 |

| Index | Predicted Probability | Predicted Classifcation | Predicted Cost |
|---|---|---|---|
| 3894 | 0.4273 | 0 | 0 |
| 3907 | 0.05348 | 0 | 0 |
| 3910 | NA | NA | 0 |
| 3913 | 0.0223 | 0 | 0 |
| 3914 | 0.1486 | 0 | 0 |
| 3921 | 0.2951 | 0 | 0 |
| 3923 | 0.03789 | 0 | 0 |
| 3929 | 0.5008 | 1 | 7715 |
| 3931 | 0.5581 | 1 | 4869 |
| 3932 | 0.2386 | 0 | 0 |
| 3937 | 0.5107 | 1 | 3580 |
| 3943 | 0.3512 | 0 | 0 |
| 3956 | NA | NA | 0 |
| 3957 | 0.367 | 0 | 0 |
| 3961 | 0.5568 | 1 | 4863 |
| 3971 | NA | NA | 0 |
| 4004 | NA | NA | 0 |
| 4005 | 0.0465 | 0 | 0 |
| 4006 | 0.01144 | 0 | 0 |
| 4011 | NA | NA | 0 |
| 4013 | 0.1348 | 0 | 0 |
| 4014 | 0.1363 | 0 | 0 |
| 4016 | 0.6042 | 1 | NA |
| 4017 | 0.02113 | 0 | 0 |
| 4020 | 0.05775 | 0 | 0 |
| 4022 | 0.1217 | 0 | 0 |
| 4026 | NA | NA | 0 |
| 4032 | NA | NA | 0 |
| 4043 | 0.02347 | 0 | 0 |
| 4045 | 0.2303 | 0 | 0 |
| 4048 | 0.2161 | 0 | 0 |
| 4051 | 0.06768 | 0 | 0 |
| 4052 | 0.2628 | 0 | 0 |
| 4056 | 0.04187 | 0 | 0 |
| 4059 | 0.04067 | 0 | 0 |
| 4069 | NA | NA | 0 |
| 4074 | 0.3328 | 0 | 0 |
| 4076 | 0.2837 | 0 | 0 |
| 4077 | 0.677 | 1 | 7240 |
| 4079 | 0.7121 | 1 | 4994 |
| 4081 | 0.4838 | 0 | 0 |
| 4088 | 0.129 | 0 | 0 |
| 4105 | 0.2427 | 0 | 0 |
| 4125 | 0.215 | 0 | 0 |
| 4134 | 0.4783 | 0 | 0 |
| 4139 | NA | NA | 0 |
| 4146 | 0.04168 | 0 | 0 |
| 4149 | 0.1168 | 0 | 0 |
| 4151 | 0.7609 | 1 | 4075 |
| 4155 | 0.213 | 0 | 0 |
| 4157 | 0.1291 | 0 | 0 |
| 4168 | 0.5873 | 1 | 4938 |
| 4170 | 0.2448 | 0 | 0 |
| 4174 | 0.1172 | 0 | 0 |
| 4179 | 0.3014 | 0 | 0 |
| 4185 | 0.09431 | 0 | 0 |
| 4199 | 0.6694 | 1 | 6337 |
| 4205 | 0.08463 | 0 | 0 |

| Index | Predicted Probability | Predicted Classifcation | Predicted Cost |
|---|---|---|---|
| 4208 | 0.02085 | 0 | 0 |
| 4211 | NA | NA | 0 |
| 4212 | 0.08175 | 0 | 0 |
| 4215 | 0.5791 | 1 | 6582 |
| 4217 | NA | NA | 0 |
| 4219 | 0.8092 | 1 | 4929 |
| 4226 | 0.4115 | 0 | 0 |
| 4227 | 0.1805 | 0 | 0 |
| 4229 | 0.02944 | 0 | 0 |
| 4231 | 0.152 | 0 | 0 |
| 4233 | 0.01316 | 0 | 0 |
| 4237 | 0.2933 | 0 | 0 |
| 4243 | 0.3511 | 0 | 0 |
| 4248 | 0.1884 | 0 | 0 |
| 4255 | 0.1316 | 0 | 0 |
| 4262 | 0.07495 | 0 | 0 |
| 4266 | 0.5524 | 1 | 4949 |
| 4268 | 0.3333 | 0 | 0 |
| 4270 | 0.6729 | 1 | 6968 |
| 4273 | 0.1076 | 0 | 0 |
| 4276 | 0.1374 | 0 | 0 |
| 4277 | 0.1508 | 0 | 0 |
| 4279 | 0.3578 | 0 | 0 |
| 4299 | 0.1964 | 0 | 0 |
| 4313 | 0.04026 | 0 | 0 |
| 4322 | 0.05013 | 0 | 0 |
| 4324 | 0.09521 | 0 | 0 |
| 4328 | 0.2688 | 0 | 0 |
| 4331 | 0.4019 | 0 | 0 |
| 4335 | 0.0425 | 0 | 0 |
| 4337 | NA | NA | 0 |
| 4338 | 0.3296 | 0 | 0 |
| 4343 | 0.06121 | 0 | 0 |
| 4347 | 0.1677 | 0 | 0 |
| 4355 | 0.6505 | 1 | 6334 |
| 4357 | 0.003136 | 0 | 0 |
| 4359 | 0.1355 | 0 | 0 |
| 4362 | 0.1122 | 0 | 0 |
| 4368 | 0.4968 | 0 | 0 |
| 4374 | 0.05977 | 0 | 0 |
| 4375 | 0.4623 | 0 | 0 |
| 4378 | 0.3529 | 0 | 0 |
| 4381 | 0.5958 | 1 | 6729 |
| 4387 | 0.1461 | 0 | 0 |
| 4400 | 0.04302 | 0 | 0 |
| 4423 | 0.1543 | 0 | 0 |
| 4424 | 0.03687 | 0 | 0 |
| 4428 | NA | NA | 0 |
| 4433 | 0.7588 | 1 | 6204 |
| 4436 | 0.4273 | 0 | 0 |
| 4437 | 0.2793 | 0 | 0 |
| 4439 | 0.3456 | 0 | 0 |
| 4449 | 0.1499 | 0 | 0 |
| 4456 | 0.05948 | 0 | 0 |
| 4463 | 0.09729 | 0 | 0 |
| 4467 | 0.133 | 0 | 0 |
| 4468 | 0.09895 | 0 | 0 |
| 4469 | 0.1063 | 0 | 0 |

| Index | Predicted Probability | Predicted Classifcation | Predicted Cost |
|---|---|---|---|
| 4472 | 0.2894 | 0 | 0 |
| 4473 | 0.05121 | 0 | 0 |
| 4476 | 0.7027 | 1 | 4506 |
| 4500 | 0.04712 | 0 | 0 |
| 4509 | 0.3207 | 0 | 0 |
| 4513 | 0.8505 | 1 | 5283 |
| 4521 | NA | NA | 0 |
| 4527 | 0.4165 | 0 | 0 |
| 4530 | NA | NA | 0 |
| 4532 | 0.4015 | 0 | 0 |
| 4533 | 0.06814 | 0 | 0 |
| 4535 | NA | NA | 0 |
| 4536 | 0.535 | 1 | 6616 |
| 4542 | 0.3684 | 0 | 0 |
| 4551 | 0.7343 | 1 | 5551 |
| 4554 | 0.06092 | 0 | 0 |
| 4555 | 0.3652 | 0 | 0 |
| 4564 | 0.2369 | 0 | 0 |
| 4572 | 0.3326 | 0 | 0 |
| 4573 | NA | NA | 0 |
| 4577 | 0.1833 | 0 | 0 |
| 4579 | 0.3873 | 0 | 0 |
| 4583 | 0.1045 | 0 | 0 |
| 4584 | 0.489 | 0 | 0 |
| 4596 | 0.05091 | 0 | 0 |
| 4599 | 0.1362 | 0 | 0 |
| 4607 | 0.245 | 0 | 0 |
| 4609 | 0.4018 | 0 | 0 |
| 4610 | 0.1238 | 0 | 0 |
| 4616 | 0.2197 | 0 | 0 |
| 4617 | 0.1714 | 0 | 0 |
| 4633 | 0.194 | 0 | 0 |
| 4638 | 0.3908 | 0 | 0 |
| 4641 | 0.03366 | 0 | 0 |
| 4653 | 0.3794 | 0 | 0 |
| 4655 | 0.3039 | 0 | 0 |
| 4659 | 0.3424 | 0 | 0 |
| 4669 | 0.1037 | 0 | 0 |
| 4678 | 0.1135 | 0 | 0 |
| 4685 | 0.6611 | 1 | 6501 |
| 4686 | 0.1769 | 0 | 0 |
| 4691 | 0.3188 | 0 | 0 |
| 4695 | 0.2453 | 0 | 0 |
| 4698 | NA | NA | 0 |
| 4700 | 0.5515 | 1 | 4914 |
| 4711 | NA | NA | 0 |
| 4722 | 0.03401 | 0 | 0 |
| 4727 | 0.3632 | 0 | 0 |
| 4756 | 0.02201 | 0 | 0 |
| 4762 | 0.4654 | 0 | 0 |
| 4763 | 0.3511 | 0 | 0 |
| 4766 | 0.1098 | 0 | 0 |
| 4770 | 0.1507 | 0 | 0 |
| 4784 | 0.4037 | 0 | 0 |
| 4791 | 0.08122 | 0 | 0 |
| 4795 | 0.107 | 0 | 0 |
| 4799 | 0.5691 | 1 | 5766 |
| 4802 | 0.4023 | 0 | 0 |

| Index | Predicted Probability | Predicted Classifcation | Predicted Cost |
|---|---|---|---|
| 4805 | 0.6919 | 1 | 4706 |
| 4814 | 0.5784 | 1 | 4955 |
| 4816 | NA | NA | 0 |
| 4817 | 0.05815 | 0 | 0 |
| 4822 | 0.4079 | 0 | 0 |
| 4827 | 0.5434 | 1 | NA |
| 4833 | 0.1334 | 0 | 0 |
| 4836 | NA | NA | 0 |
| 4842 | 0.1584 | 0 | 0 |
| 4844 | 0.1395 | 0 | 0 |
| 4845 | 0.4252 | 0 | 0 |
| 4849 | 0.1676 | 0 | 0 |
| 4850 | 0.2508 | 0 | 0 |
| 4860 | 0.03446 | 0 | 0 |
| 4863 | 0.1891 | 0 | 0 |
| 4871 | NA | NA | 0 |
| 4878 | 0.3322 | 0 | 0 |
| 4881 | 0.5706 | 1 | 7248 |
| 4888 | 0.4855 | 0 | 0 |
| 4900 | 0.1154 | 0 | 0 |
| 4906 | 0.3887 | 0 | 0 |
| 4909 | 0.02061 | 0 | 0 |
| 4916 | 0.1288 | 0 | 0 |
| 4918 | NA | NA | 0 |
| 4926 | 0.3068 | 0 | 0 |
| 4928 | 0.1759 | 0 | 0 |
| 4941 | 0.4627 | 0 | 0 |
| 4946 | 0.08544 | 0 | 0 |
| 4949 | 0.1553 | 0 | 0 |
| 4956 | NA | NA | 0 |
| 4966 | 0.1129 | 0 | 0 |
| 4969 | 0.3948 | 0 | 0 |
| 4973 | 0.1207 | 0 | 0 |
| 4978 | 0.3812 | 0 | 0 |
| 4982 | 0.239 | 0 | 0 |
| 4985 | 0.09169 | 0 | 0 |
| 4991 | 0.1651 | 0 | 0 |
| 4998 | 0.06206 | 0 | 0 |
| 5000 | 0.5743 | 1 | 7228 |
| 5004 | 0.5022 | 1 | 5989 |
| 5005 | 0.4427 | 0 | 0 |
| 5011 | 0.6394 | 1 | 5029 |
| 5016 | 0.3421 | 0 | 0 |
| 5018 | 0.06121 | 0 | 0 |
| 5034 | NA | NA | 0 |
| 5038 | 0.03832 | 0 | 0 |
| 5042 | 0.1285 | 0 | 0 |
| 5046 | 0.05871 | 0 | 0 |
| 5051 | 0.2672 | 0 | 0 |
| 5054 | 0.1897 | 0 | 0 |
| 5057 | 0.3328 | 0 | 0 |
| 5062 | 0.04124 | 0 | 0 |
| 5063 | 0.04521 | 0 | 0 |
| 5065 | 0.1158 | 0 | 0 |
| 5066 | 0.1539 | 0 | 0 |
| 5076 | 0.2081 | 0 | 0 |
| 5089 | 0.2369 | 0 | 0 |
| 5092 | 0.32 | 0 | 0 |

| Index | Predicted Probability | Predicted Classifcation | Predicted Cost |
|---|---|---|---|
| 5093 | 0.5211 | 1 | 3305 |
| 5094 | 0.0277 | 0 | 0 |
| 5098 | 0.6807 | 1 | 3942 |
| 5102 | 0.08569 | 0 | 0 |
| 5112 | 0.3141 | 0 | 0 |
| 5117 | 0.4611 | 0 | 0 |
| 5127 | 0.5048 | 1 | 4746 |
| 5130 | NA | NA | 0 |
| 5131 | 0.4345 | 0 | 0 |
| 5132 | 0.5272 | 1 | 7285 |
| 5135 | 0.8486 | 1 | 4665 |
| 5136 | 0.0227 | 0 | 0 |
| 5147 | 0.3969 | 0 | 0 |
| 5157 | 0.1199 | 0 | 0 |
| 5160 | 0.2355 | 0 | 0 |
| 5165 | 0.02781 | 0 | 0 |
| 5166 | 0.4566 | 0 | 0 |
| 5172 | 0.7028 | 1 | 6122 |
| 5173 | 0.1936 | 0 | 0 |
| 5179 | NA | NA | 0 |
| 5184 | 0.4423 | 0 | 0 |
| 5187 | NA | NA | 0 |
| 5191 | 0.1173 | 0 | 0 |
| 5193 | 0.1382 | 0 | 0 |
| 5194 | 0.08933 | 0 | 0 |
| 5199 | 0.2292 | 0 | 0 |
| 5212 | 0.05072 | 0 | 0 |
| 5213 | 0.5752 | 1 | 3515 |
| 5224 | 0.2462 | 0 | 0 |
| 5226 | NA | NA | 0 |
| 5239 | 0.3504 | 0 | 0 |
| 5252 | 0.7401 | 1 | 6780 |
| 5264 | 0.1099 | 0 | 0 |
| 5266 | 0.01282 | 0 | 0 |
| 5271 | 0.01304 | 0 | 0 |
| 5273 | 0.02625 | 0 | 0 |
| 5276 | 0.6687 | 1 | 4465 |
| 5278 | 0.07404 | 0 | 0 |
| 5281 | 0.6665 | 1 | 6406 |
| 5283 | 0.692 | 1 | 7007 |
| 5291 | 0.08633 | 0 | 0 |
| 5294 | 0.2803 | 0 | 0 |
| 5296 | 0.3972 | 0 | 0 |
| 5297 | 0.83 | 1 | 5893 |
| 5313 | 0.02684 | 0 | 0 |
| 5314 | 0.4245 | 0 | 0 |
| 5321 | 0.1992 | 0 | 0 |
| 5325 | 0.02505 | 0 | 0 |
| 5326 | 0.2361 | 0 | 0 |
| 5328 | 0.02331 | 0 | 0 |
| 5334 | 0.1845 | 0 | 0 |
| 5338 | 0.4033 | 0 | 0 |
| 5344 | NA | NA | 0 |
| 5348 | 0.3038 | 0 | 0 |
| 5352 | 0.3064 | 0 | 0 |
| 5353 | 0.08067 | 0 | 0 |
| 5354 | 0.3226 | 0 | 0 |
| 5361 | 0.9063 | 1 | 4109 |

| Index | Predicted Probability | Predicted Classifcation | Predicted Cost |
|---|---|---|---|
| 5364 | 0.05128 | 0 | 0 |
| 5365 | 0.1097 | 0 | 0 |
| 5367 | 0.4252 | 0 | 0 |
| 5379 | 0.3142 | 0 | 0 |
| 5382 | 0.2946 | 0 | 0 |
| 5386 | 0.2572 | 0 | 0 |
| 5395 | 0.1244 | 0 | 0 |
| 5410 | 0.3705 | 0 | 0 |
| 5411 | NA | NA | 0 |
| 5416 | 0.2978 | 0 | 0 |
| 5424 | 0.6798 | 1 | 4093 |
| 5426 | NA | NA | 0 |
| 5428 | 0.09761 | 0 | 0 |
| 5430 | 0.3306 | 0 | 0 |
| 5433 | 0.1709 | 0 | 0 |
| 5437 | 0.003248 | 0 | 0 |
| 5440 | 0.2152 | 0 | 0 |
| 5442 | 0.8528 | 1 | 6362 |
| 5445 | 0.62 | 1 | 5431 |
| 5449 | 0.2348 | 0 | 0 |
| 5452 | 0.3593 | 0 | 0 |
| 5460 | 0.5758 | 1 | 2870 |
| 5461 | 0.05004 | 0 | 0 |
| 5465 | 0.3057 | 0 | 0 |
| 5467 | 0.1095 | 0 | 0 |
| 5471 | 0.2492 | 0 | 0 |
| 5474 | 0.6748 | 1 | 7979 |
| 5475 | 0.04686 | 0 | 0 |
| 5480 | 0.1003 | 0 | 0 |
| 5481 | 0.2834 | 0 | 0 |
| 5484 | NA | NA | 0 |
| 5494 | 0.0622 | 0 | 0 |
| 5495 | 0.823 | 1 | 4643 |
| 5497 | 0.05216 | 0 | 0 |
| 5499 | 0.5334 | 1 | 6052 |
| 5507 | 0.05373 | 0 | 0 |
| 5510 | 0.0991 | 0 | 0 |
| 5515 | 0.189 | 0 | 0 |
| 5516 | 0.0905 | 0 | 0 |
| 5517 | 0.2044 | 0 | 0 |
| 5524 | 0.0908 | 0 | 0 |
| 5530 | 0.1455 | 0 | 0 |
| 5534 | 0.2884 | 0 | 0 |
| 5543 | 0.4106 | 0 | 0 |
| 5545 | 0.634 | 1 | 5305 |
| 5558 | 0.06965 | 0 | 0 |
| 5562 | 0.1857 | 0 | 0 |
| 5573 | 0.7614 | 1 | 7682 |
| 5581 | 0.1152 | 0 | 0 |
| 5583 | 0.5387 | 1 | 7238 |
| 5587 | 0.5814 | 1 | 3679 |
| 5589 | 0.7622 | 1 | 5298 |
| 5591 | 0.07318 | 0 | 0 |
| 5596 | NA | NA | 0 |
| 5606 | 0.7603 | 1 | 5994 |
| 5608 | 0.3715 | 0 | 0 |
| 5611 | 0.06992 | 0 | 0 |
| 5612 | 0.2669 | 0 | 0 |

| Index | Predicted Probability | Predicted Classifcation | Predicted Cost |
|---|---|---|---|
| 5614 | 0.3595 | 0 | 0 |
| 5620 | 0.06002 | 0 | 0 |
| 5623 | 0.1613 | 0 | 0 |
| 5624 | 0.2487 | 0 | 0 |
| 5626 | 0.3201 | 0 | 0 |
| 5633 | 0.1755 | 0 | 0 |
| 5635 | 0.06529 | 0 | 0 |
| 5640 | 0.5631 | 1 | 4184 |
| 5643 | 0.09018 | 0 | 0 |
| 5644 | 0.349 | 0 | 0 |
| 5653 | 0.3405 | 0 | 0 |
| 5663 | 0.02067 | 0 | 0 |
| 5664 | 0.5043 | 1 | 4688 |
| 5667 | NA | NA | 0 |
| 5671 | 0.5464 | 1 | 5815 |
| 5673 | 0.6395 | 1 | 7243 |
| 5676 | 0.05075 | 0 | 0 |
| 5678 | 0.1066 | 0 | 0 |
| 5698 | NA | NA | 0 |
| 5700 | 0.02849 | 0 | 0 |
| 5705 | 0.2386 | 0 | 0 |
| 5706 | 0.6907 | 1 | NA |
| 5711 | NA | NA | 0 |
| 5712 | 0.7802 | 1 | 3797 |
| 5716 | 0.4161 | 0 | 0 |
| 5719 | 0.2428 | 0 | 0 |
| 5725 | 0.8244 | 1 | 3034 |
| 5728 | NA | NA | 0 |
| 5734 | 0.05967 | 0 | 0 |
| 5735 | 0.05367 | 0 | 0 |
| 5743 | NA | NA | 0 |
| 5754 | 0.1954 | 0 | 0 |
| 5755 | 0.2969 | 0 | 0 |
| 5756 | 0.06026 | 0 | 0 |
| 5766 | NA | NA | 0 |
| 5770 | 0.4349 | 0 | 0 |
| 5774 | 0.2638 | 0 | 0 |
| 5775 | 0.02678 | 0 | 0 |
| 5776 | 0.07782 | 0 | 0 |
| 5778 | 0.02868 | 0 | 0 |
| 5786 | 0.4113 | 0 | 0 |
| 5787 | 0.4244 | 0 | 0 |
| 5791 | 0.2429 | 0 | 0 |
| 5794 | 0.2156 | 0 | 0 |
| 5803 | 0.1834 | 0 | 0 |
| 5804 | 0.399 | 0 | 0 |
| 5808 | 0.195 | 0 | 0 |
| 5810 | 0.03952 | 0 | 0 |
| 5813 | NA | NA | 0 |
| 5828 | 0.2418 | 0 | 0 |
| 5839 | 0.2139 | 0 | 0 |
| 5842 | 0.5042 | 1 | NA |
| 5843 | 0.05036 | 0 | 0 |
| 5844 | 0.2785 | 0 | 0 |
| 5847 | 0.6156 | 1 | 7708 |
| 5851 | 0.03762 | 0 | 0 |
| 5854 | 0.02815 | 0 | 0 |
| 5857 | 0.03152 | 0 | 0 |

| Index | Predicted Probability | Predicted Classifcation | Predicted Cost |
|---|---|---|---|
| 5866 | 0.3817 | 0 | 0 |
| 5874 | 0.2821 | 0 | 0 |
| 5886 | 0.07496 | 0 | 0 |
| 5895 | 0.0954 | 0 | 0 |
| 5897 | 0.0227 | 0 | 0 |
| 5898 | 0.2233 | 0 | 0 |
| 5900 | 0.6555 | 1 | 5668 |
| 5902 | 0.3321 | 0 | 0 |
| 5908 | NA | NA | 0 |
| 5909 | 0.05071 | 0 | 0 |
| 5912 | 0.02563 | 0 | 0 |
| 5913 | 0.1404 | 0 | 0 |
| 5917 | 0.3553 | 0 | 0 |
| 5918 | 0.7211 | 1 | 6262 |
| 5921 | 0.06835 | 0 | 0 |
| 5931 | 0.2579 | 0 | 0 |
| 5942 | 0.418 | 0 | 0 |
| 5943 | 0.6797 | 1 | 6297 |
| 5950 | 0.05245 | 0 | 0 |
| 5954 | 0.008763 | 0 | 0 |
| 5983 | 0.03435 | 0 | 0 |
| 5995 | 0.6046 | 1 | 4709 |
| 6002 | 0.08344 | 0 | 0 |
| 6005 | 0.04378 | 0 | 0 |
| 6009 | 0.2565 | 0 | 0 |
| 6011 | 0.00605 | 0 | 0 |
| 6012 | 0.02339 | 0 | 0 |
| 6019 | 0.2088 | 0 | 0 |
| 6021 | 0.2598 | 0 | 0 |
| 6029 | 0.7134 | 1 | 7110 |
| 6036 | 0.4783 | 0 | 0 |
| 6037 | 0.01052 | 0 | 0 |
| 6038 | 0.03844 | 0 | 0 |
| 6043 | 0.06195 | 0 | 0 |
| 6045 | 0.1277 | 0 | 0 |
| 6047 | 0.7159 | 1 | 5251 |
| 6048 | 0.03403 | 0 | 0 |
| 6061 | 0.3363 | 0 | 0 |
| 6063 | 0.1591 | 0 | 0 |
| 6064 | 0.06898 | 0 | 0 |
| 6068 | 0.7316 | 1 | NA |
| 6069 | 0.04758 | 0 | 0 |
| 6070 | 0.4417 | 0 | 0 |
| 6071 | 0.122 | 0 | 0 |
| 6074 | 0.4913 | 0 | 0 |
| 6079 | 0.2214 | 0 | 0 |
| 6082 | 0.06613 | 0 | 0 |
| 6088 | 0.8593 | 1 | NA |
| 6094 | NA | NA | 0 |
| 6095 | 0.3055 | 0 | 0 |
| 6098 | 0.3826 | 0 | 0 |
| 6102 | 0.01543 | 0 | 0 |
| 6105 | 0.3615 | 0 | 0 |
| 6113 | 0.1297 | 0 | 0 |
| 6116 | 0.4341 | 0 | 0 |
| 6120 | 0.4339 | 0 | 0 |
| 6121 | 0.3066 | 0 | 0 |
| 6126 | 0.2844 | 0 | 0 |

| Index | Predicted Probability | Predicted Classifcation | Predicted Cost |
|---|---|---|---|
| 6144 | 0.1433 | 0 | 0 |
| 6145 | 0.02232 | 0 | 0 |
| 6153 | 0.3558 | 0 | 0 |
| 6156 | 0.1249 | 0 | 0 |
| 6159 | 0.3659 | 0 | 0 |
| 6162 | 0.1421 | 0 | 0 |
| 6184 | 0.6431 | 1 | 5577 |
| 6188 | 0.2934 | 0 | 0 |
| 6189 | 0.3517 | 0 | 0 |
| 6191 | NA | NA | 0 |
| 6211 | 0.4383 | 0 | 0 |
| 6216 | 0.1854 | 0 | 0 |
| 6218 | 0.4688 | 0 | 0 |
| 6222 | 0.2554 | 0 | 0 |
| 6235 | 0.2998 | 0 | 0 |
| 6245 | 0.1251 | 0 | 0 |
| 6248 | 0.569 | 1 | 6031 |
| 6253 | NA | NA | 0 |
| 6256 | 0.008421 | 0 | 0 |
| 6257 | 0.5445 | 1 | 5078 |
| 6259 | NA | NA | 0 |
| 6266 | 0.07119 | 0 | 0 |
| 6268 | 0.2739 | 0 | 0 |
| 6275 | NA | NA | 0 |
| 6280 | 0.6166 | 1 | 6730 |
| 6283 | NA | NA | 0 |
| 6288 | 0.06338 | 0 | 0 |
| 6289 | 0.07853 | 0 | 0 |
| 6301 | NA | NA | 0 |
| 6308 | 0.2766 | 0 | 0 |
| 6314 | 0.02575 | 0 | 0 |
| 6315 | 0.09277 | 0 | 0 |
| 6316 | 0.5572 | 1 | NA |
| 6317 | 0.3293 | 0 | 0 |
| 6318 | 0.06813 | 0 | 0 |
| 6323 | 0.6834 | 1 | 4460 |
| 6329 | 0.719 | 1 | 2596 |
| 6336 | 0.2557 | 0 | 0 |
| 6341 | NA | NA | 0 |
| 6348 | 0.2184 | 0 | 0 |
| 6349 | 0.03797 | 0 | 0 |
| 6365 | 0.08989 | 0 | 0 |
| 6372 | 0.1285 | 0 | 0 |
| 6376 | NA | NA | 0 |
| 6378 | 0.07544 | 0 | 0 |
| 6379 | 0.8021 | 1 | NA |
| 6382 | 0.1271 | 0 | 0 |
| 6383 | NA | NA | 0 |
| 6389 | 0.6254 | 1 | 4495 |
| 6390 | 0.05171 | 0 | 0 |
| 6392 | 0.03886 | 0 | 0 |
| 6394 | 0.5036 | 1 | NA |
| 6402 | 0.1031 | 0 | 0 |
| 6404 | 0.3743 | 0 | 0 |
| 6405 | 0.01807 | 0 | 0 |
| 6406 | NA | NA | 0 |
| 6409 | 0.1268 | 0 | 0 |
| 6410 | 0.08248 | 0 | 0 |

| Index | Predicted Probability | Predicted Classifcation | Predicted Cost |
|---|---|---|---|
| 6411 | 0.1048 | 0 | 0 |
| 6421 | 0.1339 | 0 | 0 |
| 6428 | 0.3066 | 0 | 0 |
| 6429 | NA | NA | 0 |
| 6432 | 0.1144 | 0 | 0 |
| 6436 | 0.05858 | 0 | 0 |
| 6437 | 0.2758 | 0 | 0 |
| 6438 | 0.1546 | 0 | 0 |
| 6445 | 0.1349 | 0 | 0 |
| 6447 | 0.4651 | 0 | 0 |
| 6450 | 0.03273 | 0 | 0 |
| 6462 | 0.2435 | 0 | 0 |
| 6467 | 0.6983 | 1 | NA |
| 6478 | 0.04774 | 0 | 0 |
| 6484 | 0.1633 | 0 | 0 |
| 6492 | 0.3629 | 0 | 0 |
| 6497 | 0.04436 | 0 | 0 |
| 6504 | NA | NA | 0 |
| 6505 | 0.1704 | 0 | 0 |
| 6513 | 0.4569 | 0 | 0 |
| 6525 | 0.2021 | 0 | 0 |
| 6526 | 0.388 | 0 | 0 |
| 6528 | 0.05318 | 0 | 0 |
| 6540 | 0.006759 | 0 | 0 |
| 6542 | 0.1762 | 0 | 0 |
| 6544 | NA | NA | 0 |
| 6548 | 0.1232 | 0 | 0 |
| 6552 | 0.2023 | 0 | 0 |
| 6558 | 0.01365 | 0 | 0 |
| 6567 | 0.09187 | 0 | 0 |
| 6569 | 0.5166 | 1 | 7590 |
| 6572 | 0.05963 | 0 | 0 |
| 6577 | NA | NA | 0 |
| 6581 | 0.3797 | 0 | 0 |
| 6588 | 0.6719 | 1 | 5447 |
| 6591 | 0.5671 | 1 | 3295 |
| 6594 | 0.3088 | 0 | 0 |
| 6600 | 0.4021 | 0 | 0 |
| 6602 | 0.2336 | 0 | 0 |
| 6604 | 0.07878 | 0 | 0 |
| 6605 | 0.0783 | 0 | 0 |
| 6614 | 0.1593 | 0 | 0 |
| 6616 | 0.3072 | 0 | 0 |
| 6621 | 0.3697 | 0 | 0 |
| 6640 | 0.3197 | 0 | 0 |
| 6641 | NA | NA | 0 |
| 6643 | 0.07701 | 0 | 0 |
| 6644 | NA | NA | 0 |
| 6649 | 0.6356 | 1 | 6730 |
| 6650 | 0.5847 | 1 | 5869 |
| 6655 | 0.6256 | 1 | 4819 |
| 6661 | 0.02306 | 0 | 0 |
| 6672 | 0.2541 | 0 | 0 |
| 6677 | 0.04127 | 0 | 0 |
| 6688 | 0.11 | 0 | 0 |
| 6689 | NA | NA | 0 |
| 6691 | 0.03065 | 0 | 0 |
| 6692 | 0.4038 | 0 | 0 |

| Index | Predicted Probability | Predicted Classifcation | Predicted Cost |
|---|---|---|---|
| 6694 | 0.6636 | 1 | 5587 |
| 6702 | 0.6072 | 1 | 5175 |
| 6714 | 0.03663 | 0 | 0 |
| 6716 | 0.3606 | 0 | 0 |
| 6724 | 0.08383 | 0 | 0 |
| 6725 | 0.05098 | 0 | 0 |
| 6730 | 0.2237 | 0 | 0 |
| 6735 | 0.4917 | 0 | 0 |
| 6738 | 0.4072 | 0 | 0 |
| 6739 | 0.3799 | 0 | 0 |
| 6743 | 0.2274 | 0 | 0 |
| 6747 | 0.1142 | 0 | 0 |
| 6750 | 0.7208 | 1 | 5007 |
| 6751 | 0.4674 | 0 | 0 |
| 6753 | NA | NA | 0 |
| 6754 | 0.4136 | 0 | 0 |
| 6755 | 0.172 | 0 | 0 |
| 6762 | 0.1748 | 0 | 0 |
| 6764 | 0.07654 | 0 | 0 |
| 6772 | 0.6741 | 1 | 5234 |
| 6774 | 0.06776 | 0 | 0 |
| 6787 | 0.303 | 0 | 0 |
| 6789 | NA | NA | 0 |
| 6793 | 0.1155 | 0 | 0 |
| 6798 | 0.01331 | 0 | 0 |
| 6799 | 0.02995 | 0 | 0 |
| 6800 | 0.08099 | 0 | 0 |
| 6802 | 0.05184 | 0 | 0 |
| 6808 | 0.5028 | 1 | 2508 |
| 6809 | 0.09605 | 0 | 0 |
| 6812 | 0.02514 | 0 | 0 |
| 6814 | 0.7911 | 1 | 5969 |
| 6816 | 0.6203 | 1 | 4753 |
| 6822 | 0.1307 | 0 | 0 |
| 6829 | 0.4555 | 0 | 0 |
| 6834 | 0.8587 | 1 | 6721 |
| 6836 | 0.04837 | 0 | 0 |
| 6839 | 0.07215 | 0 | 0 |
| 6840 | 0.38 | 0 | 0 |
| 6843 | 0.06241 | 0 | 0 |
| 6846 | 0.5547 | 1 | 2710 |
| 6848 | 0.01952 | 0 | 0 |
| 6852 | 0.09116 | 0 | 0 |
| 6856 | 0.1607 | 0 | 0 |
| 6860 | 0.1267 | 0 | 0 |
| 6866 | 0.3845 | 0 | 0 |
| 6870 | 0.5924 | 1 | 3453 |
| 6878 | 0.4673 | 0 | 0 |
| 6880 | 0.2727 | 0 | 0 |
| 6885 | 0.02612 | 0 | 0 |
| 6897 | 0.07085 | 0 | 0 |
| 6902 | 0.7651 | 1 | 5849 |
| 6904 | 0.5127 | 1 | 6917 |
| 6907 | 0.05124 | 0 | 0 |
| 6909 | NA | NA | 0 |
| 6914 | 0.5012 | 1 | 5155 |
| 6915 | 0.4846 | 0 | 0 |
| 6922 | 0.2736 | 0 | 0 |

| Index | Predicted Probability | Predicted Classifcation | Predicted Cost |
|---|---|---|---|
| 6924 | 0.1831 | 0 | 0 |
| 6933 | 0.06935 | 0 | 0 |
| 6934 | 0.09964 | 0 | 0 |
| 6941 | 0.2205 | 0 | 0 |
| 6957 | 0.3988 | 0 | 0 |
| 6960 | 0.07196 | 0 | 0 |
| 6969 | NA | NA | 0 |
| 6975 | NA | NA | 0 |
| 6980 | 0.7211 | 1 | 6172 |
| 6983 | NA | NA | 0 |
| 6987 | 0.07809 | 0 | 0 |
| 6994 | 0.03858 | 0 | 0 |
| 6997 | 0.01219 | 0 | 0 |
| 7002 | 0.1195 | 0 | 0 |
| 7010 | NA | NA | 0 |
| 7015 | 0.5559 | 1 | 3138 |
| 7019 | 0.3025 | 0 | 0 |
| 7022 | 0.2591 | 0 | 0 |
| 7025 | 0.02192 | 0 | 0 |
| 7029 | 0.07229 | 0 | 0 |
| 7031 | 0.17 | 0 | 0 |
| 7037 | 0.3542 | 0 | 0 |
| 7038 | 0.3134 | 0 | 0 |
| 7043 | 0.1388 | 0 | 0 |
| 7049 | 0.06991 | 0 | 0 |
| 7052 | 0.2715 | 0 | 0 |
| 7053 | 0.3136 | 0 | 0 |
| 7056 | 0.009383 | 0 | 0 |
| 7057 | 0.4901 | 0 | 0 |
| 7080 | 0.1886 | 0 | 0 |
| 7086 | 0.3285 | 0 | 0 |
| 7087 | 0.121 | 0 | 0 |
| 7105 | 0.4724 | 0 | 0 |
| 7108 | NA | NA | 0 |
| 7121 | 0.4042 | 0 | 0 |
| 7122 | 0.2193 | 0 | 0 |
| 7125 | 0.4101 | 0 | 0 |
| 7132 | 0.2744 | 0 | 0 |
| 7134 | 0.1407 | 0 | 0 |
| 7151 | 0.2507 | 0 | 0 |
| 7152 | 0.6994 | 1 | 4520 |
| 7157 | 0.1907 | 0 | 0 |
| 7159 | 0.2921 | 0 | 0 |
| 7166 | 0.6377 | 1 | 3679 |
| 7167 | 0.08439 | 0 | 0 |
| 7177 | 0.04687 | 0 | 0 |
| 7179 | 0.616 | 1 | 5512 |
| 7181 | 0.2821 | 0 | 0 |
| 7183 | 0.1596 | 0 | 0 |
| 7186 | 0.03455 | 0 | 0 |
| 7193 | 0.03031 | 0 | 0 |
| 7205 | 0.03251 | 0 | 0 |
| 7207 | 0.03658 | 0 | 0 |
| 7209 | 0.373 | 0 | 0 |
| 7216 | 0.2822 | 0 | 0 |
| 7232 | 0.7265 | 1 | 6303 |
| 7235 | 0.1085 | 0 | 0 |
| 7238 | NA | NA | 0 |

| Index | Predicted Probability | Predicted Classifcation | Predicted Cost |
|---|---|---|---|
| 7240 | 0.6323 | 1 | 5039 |
| 7243 | NA | NA | 0 |
| 7252 | 0.195 | 0 | 0 |
| 7269 | 0.2055 | 0 | 0 |
| 7275 | 0.03975 | 0 | 0 |
| 7281 | NA | NA | 0 |
| 7283 | 0.05575 | 0 | 0 |
| 7287 | 0.2093 | 0 | 0 |
| 7289 | 0.2778 | 0 | 0 |
| 7291 | 0.3913 | 0 | 0 |
| 7294 | 0.04737 | 0 | 0 |
| 7304 | 0.437 | 0 | 0 |
| 7308 | 0.2702 | 0 | 0 |
| 7313 | 0.05841 | 0 | 0 |
| 7319 | 0.3014 | 0 | 0 |
| 7325 | 0.1703 | 0 | 0 |
| 7326 | 0.1533 | 0 | 0 |
| 7330 | NA | NA | 0 |
| 7332 | NA | NA | 0 |
| 7337 | 0.2657 | 0 | 0 |
| 7341 | 0.2131 | 0 | 0 |
| 7346 | NA | NA | 0 |
| 7353 | 0.5371 | 1 | 6035 |
| 7354 | 0.579 | 1 | 5631 |
| 7361 | 0.3918 | 0 | 0 |
| 7366 | 0.5091 | 1 | 7922 |
| 7368 | 0.03048 | 0 | 0 |
| 7372 | 0.05445 | 0 | 0 |
| 7375 | 0.3875 | 0 | 0 |
| 7377 | 0.3066 | 0 | 0 |
| 7380 | 0.1073 | 0 | 0 |
| 7382 | 0.2623 | 0 | 0 |
| 7385 | 0.6099 | 1 | 7450 |
| 7392 | 0.6307 | 1 | 5912 |
| 7395 | 0.1253 | 0 | 0 |
| 7397 | 0.2113 | 0 | 0 |
| 7403 | NA | NA | 0 |
| 7406 | 0.2555 | 0 | 0 |
| 7409 | NA | NA | 0 |
| 7410 | 0.3106 | 0 | 0 |
| 7412 | 0.07648 | 0 | 0 |
| 7419 | 0.3681 | 0 | 0 |
| 7425 | 0.08251 | 0 | 0 |
| 7435 | NA | NA | 0 |
| 7438 | NA | NA | 0 |
| 7440 | 0.245 | 0 | 0 |
| 7447 | 0.1587 | 0 | 0 |
| 7449 | 0.5959 | 1 | 5272 |
| 7456 | NA | NA | 0 |
| 7464 | 0.1957 | 0 | 0 |
| 7478 | 0.1425 | 0 | 0 |
| 7480 | 0.01682 | 0 | 0 |
| 7481 | 0.4132 | 0 | 0 |
| 7483 | 0.2004 | 0 | 0 |
| 7484 | 0.1978 | 0 | 0 |
| 7491 | 0.5541 | 1 | 8070 |
| 7494 | 0.3817 | 0 | 0 |
| 7501 | 0.3754 | 0 | 0 |

| Index | Predicted Probability | Predicted Classifcation | Predicted Cost |
|---|---|---|---|
| 7503 | 0.8052 | 1 | 7866 |
| 7509 | NA | NA | 0 |
| 7517 | NA | NA | 0 |
| 7518 | 0.15 | 0 | 0 |
| 7519 | 0.3656 | 0 | 0 |
| 7521 | 0.4103 | 0 | 0 |
| 7522 | 0.4702 | 0 | 0 |
| 7536 | 0.06065 | 0 | 0 |
| 7539 | 0.04095 | 0 | 0 |
| 7547 | NA | NA | 0 |
| 7549 | NA | NA | 0 |
| 7552 | 0.577 | 1 | 4469 |
| 7554 | 0.3423 | 0 | 0 |
| 7556 | 0.03725 | 0 | 0 |
| 7564 | 0.1056 | 0 | 0 |
| 7566 | 0.2041 | 0 | 0 |
| 7570 | 0.2367 | 0 | 0 |
| 7571 | 0.0293 | 0 | 0 |
| 7572 | 0.2053 | 0 | 0 |
| 7575 | 0.09879 | 0 | 0 |
| 7586 | 0.1113 | 0 | 0 |
| 7589 | 0.06295 | 0 | 0 |
| 7590 | 0.04135 | 0 | 0 |
| 7597 | 0.2425 | 0 | 0 |
| 7602 | 0.05224 | 0 | 0 |
| 7604 | 0.4658 | 0 | 0 |
| 7605 | 0.3691 | 0 | 0 |
| 7612 | 0.7963 | 1 | NA |
| 7615 | NA | NA | 0 |
| 7617 | NA | NA | 0 |
| 7624 | 0.1142 | 0 | 0 |
| 7632 | 0.1996 | 0 | 0 |
| 7639 | 0.3158 | 0 | 0 |
| 7642 | NA | NA | 0 |
| 7643 | 0.09522 | 0 | 0 |
| 7649 | 0.3671 | 0 | 0 |
| 7650 | 0.3813 | 0 | 0 |
| 7653 | 0.3766 | 0 | 0 |
| 7654 | 0.3051 | 0 | 0 |
| 7657 | 0.4942 | 0 | 0 |
| 7662 | 0.2924 | 0 | 0 |
| 7669 | 0.8235 | 1 | 7237 |
| 7671 | 0.02389 | 0 | 0 |
| 7675 | 0.08791 | 0 | 0 |
| 7678 | 0.2313 | 0 | 0 |
| 7682 | 0.7825 | 1 | 6017 |
| 7688 | 0.3104 | 0 | 0 |
| 7689 | 0.2093 | 0 | 0 |
| 7690 | 0.1319 | 0 | 0 |
| 7692 | NA | NA | 0 |
| 7699 | 0.341 | 0 | 0 |
| 7705 | 0.557 | 1 | 4965 |
| 7712 | 0.1447 | 0 | 0 |
| 7726 | 0.5454 | 1 | 5345 |
| 7728 | 0.1508 | 0 | 0 |
| 7735 | NA | NA | 0 |
| 7737 | 0.6557 | 1 | 4446 |
| 7739 | 0.04481 | 0 | 0 |

| Index | Predicted Probability | Predicted Classifcation | Predicted Cost |
|---|---|---|---|
| 7743 | 0.6116 | 1 | 5531 |
| 7744 | 0.1831 | 0 | 0 |
| 7746 | 0.2833 | 0 | 0 |
| 7749 | 0.3382 | 0 | 0 |
| 7750 | 0.2593 | 0 | 0 |
| 7752 | 0.04092 | 0 | 0 |
| 7755 | 0.1087 | 0 | 0 |
| 7756 | 0.8198 | 1 | 5561 |
| 7762 | 0.1215 | 0 | 0 |
| 7764 | 0.5826 | 1 | 4907 |
| 7769 | 0.1158 | 0 | 0 |
| 7770 | 0.3992 | 0 | 0 |
| 7776 | 0.2632 | 0 | 0 |
| 7778 | 0.4155 | 0 | 0 |
| 7784 | 0.4647 | 0 | 0 |
| 7786 | 0.2194 | 0 | 0 |
| 7789 | 0.1934 | 0 | 0 |
| 7793 | 0.1951 | 0 | 0 |
| 7794 | 0.08699 | 0 | 0 |
| 7804 | 0.231 | 0 | 0 |
| 7811 | NA | NA | 0 |
| 7813 | 0.1699 | 0 | 0 |
| 7815 | 0.3152 | 0 | 0 |
| 7817 | 0.02225 | 0 | 0 |
| 7818 | 0.2217 | 0 | 0 |
| 7821 | 0.2253 | 0 | 0 |
| 7825 | 0.02431 | 0 | 0 |
| 7830 | 0.467 | 0 | 0 |
| 7832 | 0.1691 | 0 | 0 |
| 7835 | 0.0165 | 0 | 0 |
| 7839 | 0.1231 | 0 | 0 |
| 7842 | 0.07586 | 0 | 0 |
| 7849 | 0.5016 | 1 | 7480 |
| 7856 | 0.3876 | 0 | 0 |
| 7857 | 0.004831 | 0 | 0 |
| 7863 | 0.09146 | 0 | 0 |
| 7866 | 0.17 | 0 | 0 |
| 7871 | 0.1494 | 0 | 0 |
| 7875 | NA | NA | 0 |
| 7882 | 0.8144 | 1 | 6435 |
| 7887 | 0.62 | 1 | 5153 |
| 7888 | NA | NA | 0 |
| 7891 | 0.8335 | 1 | 6991 |
| 7895 | 0.01574 | 0 | 0 |
| 7901 | NA | NA | 0 |
| 7906 | 0.2365 | 0 | 0 |
| 7908 | 0.8055 | 1 | 5837 |
| 7917 | NA | NA | 0 |
| 7924 | 0.6438 | 1 | 4583 |
| 7948 | 0.1741 | 0 | 0 |
| 7950 | 0.7433 | 1 | 6009 |
| 7955 | NA | NA | 0 |
| 7957 | 0.07371 | 0 | 0 |
| 7959 | 0.2317 | 0 | 0 |
| 7967 | 0.09708 | 0 | 0 |
| 7969 | 0.05027 | 0 | 0 |
| 7971 | 0.1729 | 0 | 0 |
| 7974 | 0.2578 | 0 | 0 |

| Index | Predicted Probability | Predicted Classifcation | Predicted Cost |
|---|---|---|---|
| 7976 | 0.0894 | 0 | 0 |
| 7986 | 0.7631 | 1 | 4053 |
| 7987 | NA | NA | 0 |
| 7993 | 0.3705 | 0 | 0 |
| 7996 | 0.4322 | 0 | 0 |
| 7998 | 0.283 | 0 | 0 |
| 8018 | 0.05636 | 0 | 0 |
| 8019 | 0.3795 | 0 | 0 |
| 8027 | 0.01575 | 0 | 0 |
| 8036 | 0.1314 | 0 | 0 |
| 8040 | 0.06262 | 0 | 0 |
| 8044 | 0.08932 | 0 | 0 |
| 8050 | 0.04643 | 0 | 0 |
| 8052 | 0.5795 | 1 | NA |
| 8054 | 0.2211 | 0 | 0 |
| 8057 | 0.5018 | 1 | 3782 |
| 8058 | 0.1821 | 0 | 0 |
| 8059 | 0.631 | 1 | 4441 |
| 8066 | 0.7646 | 1 | 4996 |
| 8070 | 0.03201 | 0 | 0 |
| 8072 | 0.349 | 0 | 0 |
| 8078 | 0.03621 | 0 | 0 |
| 8079 | 0.05216 | 0 | 0 |
| 8080 | 0.2261 | 0 | 0 |
| 8081 | 0.1813 | 0 | 0 |
| 8088 | 0.1229 | 0 | 0 |
| 8091 | NA | NA | 0 |
| 8094 | NA | NA | 0 |
| 8095 | 0.6986 | 1 | 5449 |
| 8099 | 0.1037 | 0 | 0 |
| 8101 | 0.2943 | 0 | 0 |
| 8102 | 0.007156 | 0 | 0 |
| 8116 | 0.3713 | 0 | 0 |
| 8125 | 0.4551 | 0 | 0 |
| 8134 | 0.3021 | 0 | 0 |
| 8139 | 0.03433 | 0 | 0 |
| 8141 | 0.06425 | 0 | 0 |
| 8147 | 0.07794 | 0 | 0 |
| 8158 | 0.2018 | 0 | 0 |
| 8160 | NA | NA | 0 |
| 8165 | 0.3245 | 0 | 0 |
| 8187 | 0.3396 | 0 | 0 |
| 8205 | 0.4436 | 0 | 0 |
| 8209 | 0.2999 | 0 | 0 |
| 8211 | 0.3795 | 0 | 0 |
| 8232 | NA | NA | 0 |
| 8236 | NA | NA | 0 |
| 8237 | 0.2556 | 0 | 0 |
| 8238 | 0.7427 | 1 | 5809 |
| 8245 | 0.3455 | 0 | 0 |
| 8256 | 0.2717 | 0 | 0 |
| 8268 | 0.06638 | 0 | 0 |
| 8269 | 0.02077 | 0 | 0 |
| 8270 | 0.4714 | 0 | 0 |
| 8286 | 0.1151 | 0 | 0 |
| 8289 | 0.08858 | 0 | 0 |
| 8301 | 0.4462 | 0 | 0 |
| 8305 | NA | NA | 0 |

| Index | Predicted Probability | Predicted Classifcation | Predicted Cost |
|---|---|---|---|
| 8310 | 0.348 | 0 | 0 |
| 8312 | 0.03615 | 0 | 0 |
| 8318 | 0.8631 | 1 | 5276 |
| 8321 | 0.141 | 0 | 0 |
| 8328 | 0.05982 | 0 | 0 |
| 8331 | 0.02815 | 0 | 0 |
| 8334 | 0.2613 | 0 | 0 |
| 8344 | 0.2755 | 0 | 0 |
| 8345 | 0.1146 | 0 | 0 |
| 8352 | 0.3849 | 0 | 0 |
| 8358 | 0.4372 | 0 | 0 |
| 8359 | 0.4035 | 0 | 0 |
| 8360 | 0.1522 | 0 | 0 |
| 8365 | 0.3022 | 0 | 0 |
| 8366 | 0.1697 | 0 | 0 |
| 8369 | 0.7122 | 1 | 3161 |
| 8373 | 0.03738 | 0 | 0 |
| 8378 | 0.09843 | 0 | 0 |
| 8392 | 0.123 | 0 | 0 |
| 8397 | 0.3753 | 0 | 0 |
| 8399 | 0.1758 | 0 | 0 |
| 8400 | 0.1578 | 0 | 0 |
| 8405 | 0.6483 | 1 | 7243 |
| 8406 | 0.05891 | 0 | 0 |
| 8410 | 0.2018 | 0 | 0 |
| 8413 | 0.09536 | 0 | 0 |
| 8414 | 0.399 | 0 | 0 |
| 8416 | NA | NA | 0 |
| 8426 | 0.0429 | 0 | 0 |
| 8434 | 0.2304 | 0 | 0 |
| 8439 | 0.1348 | 0 | 0 |
| 8440 | 0.1855 | 0 | 0 |
| 8475 | 0.01483 | 0 | 0 |
| 8480 | NA | NA | 0 |
| 8497 | 0.1836 | 0 | 0 |
| 8499 | 0.7658 | 1 | 6847 |
| 8500 | 0.3599 | 0 | 0 |
| 8501 | 0.1202 | 0 | 0 |
| 8502 | NA | NA | 0 |
| 8518 | 0.2708 | 0 | 0 |
| 8520 | 0.537 | 1 | 4351 |
| 8523 | 0.4462 | 0 | 0 |
| 8525 | NA | NA | 0 |
| 8532 | 0.1434 | 0 | 0 |
| 8535 | 0.4381 | 0 | 0 |
| 8543 | 0.1988 | 0 | 0 |
| 8554 | 0.2508 | 0 | 0 |
| 8560 | 0.08541 | 0 | 0 |
| 8561 | 0.2514 | 0 | 0 |
| 8563 | 0.03022 | 0 | 0 |
| 8566 | 0.8024 | 1 | 3257 |
| 8570 | 0.3469 | 0 | 0 |
| 8572 | 0.0684 | 0 | 0 |
| 8582 | 0.1855 | 0 | 0 |
| 8583 | 0.2025 | 0 | 0 |
| 8587 | 0.1492 | 0 | 0 |
| 8592 | NA | NA | 0 |
| 8593 | 0.3812 | 0 | 0 |

| Index | Predicted Probability | Predicted Classifcation | Predicted Cost |
|---|---|---|---|
| 8607 | 0.01679 | 0 | 0 |
| 8609 | 0.2819 | 0 | 0 |
| 8610 | 0.03462 | 0 | 0 |
| 8614 | 0.2976 | 0 | 0 |
| 8616 | 0.4793 | 0 | 0 |
| 8622 | NA | NA | 0 |
| 8623 | 0.1635 | 0 | 0 |
| 8624 | 0.3441 | 0 | 0 |
| 8633 | 0.144 | 0 | 0 |
| 8641 | NA | NA | 0 |
| 8644 | NA | NA | 0 |
| 8649 | 0.6433 | 1 | NA |
| 8653 | 0.1616 | 0 | 0 |
| 8657 | 0.1148 | 0 | 0 |
| 8658 | 0.1252 | 0 | 0 |
| 8663 | 0.06849 | 0 | 0 |
| 8672 | 0.7013 | 1 | 4442 |
| 8680 | 0.5056 | 1 | 3072 |
| 8684 | 0.3838 | 0 | 0 |
| 8687 | 0.1123 | 0 | 0 |
| 8688 | 0.1277 | 0 | 0 |
| 8690 | 0.1254 | 0 | 0 |
| 8712 | 0.3817 | 0 | 0 |
| 8717 | 0.2302 | 0 | 0 |
| 8730 | 0.1237 | 0 | 0 |
| 8739 | 0.1633 | 0 | 0 |
| 8744 | 0.02648 | 0 | 0 |
| 8747 | 0.3095 | 0 | 0 |
| 8748 | 0.2913 | 0 | 0 |
| 8751 | 0.7506 | 1 | NA |
| 8758 | 0.3361 | 0 | 0 |
| 8761 | 0.4553 | 0 | 0 |
| 8763 | 0.0132 | 0 | 0 |
| 8764 | 0.1486 | 0 | 0 |
| 8765 | 0.1241 | 0 | 0 |
| 8773 | 0.09849 | 0 | 0 |
| 8780 | 0.203 | 0 | 0 |
| 8781 | 0.1377 | 0 | 0 |
| 8782 | 0.3932 | 0 | 0 |
| 8785 | 0.1272 | 0 | 0 |
| 8786 | 0.3653 | 0 | 0 |
| 8797 | 0.8042 | 1 | NA |
| 8799 | 0.07846 | 0 | 0 |
| 8807 | 0.7271 | 1 | 4909 |
| 8816 | 0.05086 | 0 | 0 |
| 8817 | 0.1155 | 0 | 0 |
| 8826 | 0.2336 | 0 | 0 |
| 8833 | NA | NA | 0 |
| 8834 | 0.08508 | 0 | 0 |
| 8835 | 0.1423 | 0 | 0 |
| 8840 | 0.1449 | 0 | 0 |
| 8843 | 0.09092 | 0 | 0 |
| 8849 | NA | NA | 0 |
| 8855 | 0.09769 | 0 | 0 |
| 8861 | 0.225 | 0 | 0 |
| 8862 | 0.2776 | 0 | 0 |
| 8865 | 0.2967 | 0 | 0 |
| 8868 | NA | NA | 0 |

| Index | Predicted Probability | Predicted Classifcation | Predicted Cost |
|---|---|---|---|
| 8870 | 0.03048 | 0 | 0 |
| 8880 | 0.2432 | 0 | 0 |
| 8885 | 0.07277 | 0 | 0 |
| 8894 | 0.2391 | 0 | 0 |
| 8895 | 0.1662 | 0 | 0 |
| 8899 | 0.06422 | 0 | 0 |
| 8912 | 0.429 | 0 | 0 |
| 8922 | 0.01717 | 0 | 0 |
| 8924 | 0.1148 | 0 | 0 |
| 8928 | 0.2885 | 0 | 0 |
| 8932 | 0.3277 | 0 | 0 |
| 8943 | 0.1271 | 0 | 0 |
| 8945 | 0.2123 | 0 | 0 |
| 8946 | 0.04513 | 0 | 0 |
| 8954 | 0.3807 | 0 | 0 |
| 8958 | 0.4342 | 0 | 0 |
| 8960 | NA | NA | 0 |
| 8965 | 0.1718 | 0 | 0 |
| 8966 | NA | NA | 0 |
| 8967 | 0.05851 | 0 | 0 |
| 8969 | 0.3727 | 0 | 0 |
| 8980 | 0.1666 | 0 | 0 |
| 8984 | 0.04374 | 0 | 0 |
| 8985 | 0.8273 | 1 | 4801 |
| 8988 | 0.3072 | 0 | 0 |
| 8989 | 0.3352 | 0 | 0 |
| 8995 | NA | NA | 0 |
| 9004 | NA | NA | 0 |
| 9010 | 0.06323 | 0 | 0 |
| 9012 | 0.275 | 0 | 0 |
| 9018 | 0.5613 | 1 | 1839 |
| 9036 | 0.2747 | 0 | 0 |
| 9037 | 0.2999 | 0 | 0 |
| 9040 | 0.1127 | 0 | 0 |
| 9041 | 0.4715 | 0 | 0 |
| 9044 | 0.1972 | 0 | 0 |
| 9045 | 0.1054 | 0 | 0 |
| 9047 | 0.4465 | 0 | 0 |
| 9049 | 0.02392 | 0 | 0 |
| 9061 | 0.02962 | 0 | 0 |
| 9062 | 0.3022 | 0 | 0 |
| 9076 | NA | NA | 0 |
| 9079 | 0.3321 | 0 | 0 |
| 9081 | 0.1813 | 0 | 0 |
| 9082 | 0.1392 | 0 | 0 |
| 9089 | 0.5501 | 1 | 4715 |
| 9092 | 0.166 | 0 | 0 |
| 9094 | 0.324 | 0 | 0 |
| 9115 | 0.02599 | 0 | 0 |
| 9117 | 0.3096 | 0 | 0 |
| 9118 | 0.222 | 0 | 0 |
| 9120 | 0.06577 | 0 | 0 |
| 9124 | 0.015 | 0 | 0 |
| 9128 | 0.2802 | 0 | 0 |
| 9135 | 0.4415 | 0 | 0 |
| 9136 | 0.7532 | 1 | 4784 |
| 9138 | 0.206 | 0 | 0 |
| 9157 | 0.3342 | 0 | 0 |

| Index | Predicted Probability | Predicted Classifcation | Predicted Cost |
|---|---|---|---|
| 9176 | 0.07935 | 0 | 0 |
| 9183 | 0.2604 | 0 | 0 |
| 9187 | 0.4435 | 0 | 0 |
| 9188 | NA | NA | 0 |
| 9190 | 0.08825 | 0 | 0 |
| 9197 | 0.03688 | 0 | 0 |
| 9200 | NA | NA | 0 |
| 9201 | 0.1916 | 0 | 0 |
| 9203 | 0.04329 | 0 | 0 |
| 9212 | 0.3732 | 0 | 0 |
| 9213 | 0.1261 | 0 | 0 |
| 9214 | 0.1053 | 0 | 0 |
| 9217 | 0.2421 | 0 | 0 |
| 9219 | 0.00804 | 0 | 0 |
| 9220 | NA | NA | 0 |
| 9221 | 0.2355 | 0 | 0 |
| 9237 | 0.01137 | 0 | 0 |
| 9240 | 0.2166 | 0 | 0 |
| 9241 | NA | NA | 0 |
| 9248 | 0.3196 | 0 | 0 |
| 9253 | 0.5256 | 1 | 5937 |
| 9259 | 0.653 | 1 | 6516 |
| 9267 | 0.123 | 0 | 0 |
| 9271 | 0.318 | 0 | 0 |
| 9273 | 0.222 | 0 | 0 |
| 9285 | 0.05782 | 0 | 0 |
| 9290 | 0.1697 | 0 | 0 |
| 9291 | 0.09845 | 0 | 0 |
| 9293 | 0.03804 | 0 | 0 |
| 9294 | 0.0713 | 0 | 0 |
| 9301 | NA | NA | 0 |
| 9302 | 0.04199 | 0 | 0 |
| 9312 | 0.01649 | 0 | 0 |
| 9316 | 0.2925 | 0 | 0 |
| 9319 | NA | NA | 0 |
| 9328 | NA | NA | 0 |
| 9331 | 0.637 | 1 | 2924 |
| 9338 | 0.03824 | 0 | 0 |
| 9350 | 0.2803 | 0 | 0 |
| 9356 | 0.14 | 0 | 0 |
| 9359 | 0.4976 | 0 | 0 |
| 9362 | 0.3584 | 0 | 0 |
| 9364 | 0.1696 | 0 | 0 |
| 9370 | NA | NA | 0 |
| 9380 | 0.1156 | 0 | 0 |
| 9386 | 0.08842 | 0 | 0 |
| 9394 | 0.3817 | 0 | 0 |
| 9407 | NA | NA | 0 |
| 9411 | 0.5364 | 1 | 8553 |
| 9422 | NA | NA | 0 |
| 9423 | 0.2331 | 0 | 0 |
| 9429 | 0.2693 | 0 | 0 |
| 9433 | 0.2903 | 0 | 0 |
| 9439 | NA | NA | 0 |
| 9451 | 0.2769 | 0 | 0 |
| 9452 | 0.4304 | 0 | 0 |
| 9453 | 0.02913 | 0 | 0 |
| 9460 | NA | NA | 0 |

| Index | Predicted Probability | Predicted Classifcation | Predicted Cost |
|---|---|---|---|
| 9465 | 0.0508 | 0 | 0 |
| 9470 | 0.05306 | 0 | 0 |
| 9476 | 0.4148 | 0 | 0 |
| 9485 | 0.7606 | 1 | 4836 |
| 9486 | 0.04545 | 0 | 0 |
| 9488 | 0.3253 | 0 | 0 |
| 9507 | 0.009853 | 0 | 0 |
| 9508 | 0.4014 | 0 | 0 |
| 9517 | 0.4361 | 0 | 0 |
| 9521 | NA | NA | 0 |
| 9528 | 0.1095 | 0 | 0 |
| 9532 | 0.4352 | 0 | 0 |
| 9536 | NA | NA | 0 |
| 9540 | 0.05736 | 0 | 0 |
| 9542 | 0.2386 | 0 | 0 |
| 9546 | 0.29 | 0 | 0 |
| 9548 | 0.1185 | 0 | 0 |
| 9549 | 0.1253 | 0 | 0 |
| 9554 | NA | NA | 0 |
| 9555 | 0.4198 | 0 | 0 |
| 9558 | 0.04894 | 0 | 0 |
| 9573 | NA | NA | 0 |
| 9575 | 0.7255 | 1 | 6446 |
| 9584 | 0.6235 | 1 | 6243 |
| 9586 | 0.09117 | 0 | 0 |
| 9588 | 0.119 | 0 | 0 |
| 9591 | 0.345 | 0 | 0 |
| 9592 | 0.7721 | 1 | 6312 |
| 9597 | 0.3097 | 0 | 0 |
| 9600 | NA | NA | 0 |
| 9603 | 0.5389 | 1 | 5006 |
| 9605 | 0.3452 | 0 | 0 |
| 9614 | NA | NA | 0 |
| 9616 | NA | NA | 0 |
| 9622 | 0.5842 | 1 | 4993 |
| 9624 | 0.1553 | 0 | 0 |
| 9629 | 0.5418 | 1 | 8132 |
| 9633 | 0.04865 | 0 | 0 |
| 9640 | 0.1784 | 0 | 0 |
| 9644 | 0.3905 | 0 | 0 |
| 9645 | 0.4841 | 0 | 0 |
| 9646 | 0.1761 | 0 | 0 |
| 9648 | 0.944 | 1 | 4526 |
| 9649 | 0.05664 | 0 | 0 |
| 9660 | 0.1575 | 0 | 0 |
| 9664 | 0.5752 | 1 | 5517 |
| 9675 | 0.09032 | 0 | 0 |
| 9679 | 0.7769 | 1 | 3921 |
| 9680 | 0.5163 | 1 | 6229 |
| 9682 | 0.07569 | 0 | 0 |
| 9697 | 0.02094 | 0 | 0 |
| 9701 | 0.1502 | 0 | 0 |
| 9704 | NA | NA | 0 |
| 9705 | 0.09088 | 0 | 0 |
| 9707 | 0.3823 | 0 | 0 |
| 9714 | 0.05054 | 0 | 0 |
| 9718 | 0.1165 | 0 | 0 |
| 9722 | 0.164 | 0 | 0 |

| Index | Predicted Probability | Predicted Classifcation | Predicted Cost |
|---|---|---|---|
| 9739 | 0.1776 | 0 | 0 |
| 9747 | 0.6033 | 1 | 5284 |
| 9751 | 0.103 | 0 | 0 |
| 9757 | 0.1169 | 0 | 0 |
| 9759 | 0.02218 | 0 | 0 |
| 9760 | 0.05431 | 0 | 0 |
| 9764 | 0.6818 | 1 | 3329 |
| 9776 | 0.3673 | 0 | 0 |
| 9778 | 0.1447 | 0 | 0 |
| 9786 | 0.06316 | 0 | 0 |
| 9803 | 0.4329 | 0 | 0 |
| 9804 | 0.07889 | 0 | 0 |
| 9815 | 0.1113 | 0 | 0 |
| 9824 | 0.0149 | 0 | 0 |
| 9825 | 0.2794 | 0 | 0 |
| 9826 | 0.32 | 0 | 0 |
| 9827 | 0.0168 | 0 | 0 |
| 9833 | 0.1225 | 0 | 0 |
| 9835 | 0.08799 | 0 | 0 |
| 9860 | 0.4045 | 0 | 0 |
| 9865 | 0.2768 | 0 | 0 |
| 9871 | 0.2203 | 0 | 0 |
| 9874 | 0.3189 | 0 | 0 |
| 9880 | 0.2624 | 0 | 0 |
| 9882 | 0.3029 | 0 | 0 |
| 9885 | 0.1066 | 0 | 0 |
| 9888 | NA | NA | 0 |
| 9892 | 0.02425 | 0 | 0 |
| 9893 | 0.2984 | 0 | 0 |
| 9896 | 0.3259 | 0 | 0 |
| 9902 | 0.1768 | 0 | 0 |
| 9906 | 0.06683 | 0 | 0 |
| 9910 | 0.4343 | 0 | 0 |
| 9914 | 0.2443 | 0 | 0 |
| 9918 | 0.4308 | 0 | 0 |
| 9920 | 0.2631 | 0 | 0 |
| 9926 | 0.3784 | 0 | 0 |
| 9931 | 0.09788 | 0 | 0 |
| 9935 | 0.3427 | 0 | 0 |
| 9945 | 0.854 | 1 | 4658 |
| 9953 | 0.2212 | 0 | 0 |
| 9957 | 0.01484 | 0 | 0 |
| 9963 | 0.13 | 0 | 0 |
| 9972 | 0.1778 | 0 | 0 |
| 9976 | NA | NA | 0 |
| 9979 | 0.5026 | 1 | 5954 |
| 9980 | 0.01824 | 0 | 0 |
| 9982 | 0.1512 | 0 | 0 |
| 9991 | 0.6451 | 1 | 5359 |
| 10000 | 0.1839 | 0 | 0 |
| 10003 | 0.167 | 0 | 0 |
| 10005 | 0.9337 | 1 | 4121 |
| 10014 | 0.01445 | 0 | 0 |
| 10032 | 0.5678 | 1 | 5538 |
| 10034 | NA | NA | 0 |
| 10041 | 0.01599 | 0 | 0 |
| 10042 | 0.03011 | 0 | 0 |
| 10044 | 0.0469 | 0 | 0 |

| Index | Predicted Probability | Predicted Classifcation | Predicted Cost |
|---|---|---|---|
| 10045 | 0.2849 | 0 | 0 |
| 10054 | 0.4123 | 0 | 0 |
| 10061 | NA | NA | 0 |
| 10062 | 0.4764 | 0 | 0 |
| 10073 | 0.2322 | 0 | 0 |
| 10081 | 0.02793 | 0 | 0 |
| 10084 | 0.3894 | 0 | 0 |
| 10086 | 0.1118 | 0 | 0 |
| 10093 | 0.3716 | 0 | 0 |
| 10101 | 0.6142 | 1 | 4890 |
| 10105 | 0.411 | 0 | 0 |
| 10110 | 0.3591 | 0 | 0 |
| 10113 | 0.6011 | 1 | 8367 |
| 10115 | 0.5234 | 1 | 3635 |
| 10119 | 0.5791 | 1 | NA |
| 10121 | 0.3985 | 0 | 0 |
| 10124 | 0.8426 | 1 | NA |
| 10126 | 0.2683 | 0 | 0 |
| 10127 | NA | NA | 0 |
| 10145 | 0.1125 | 0 | 0 |
| 10147 | 0.5059 | 1 | 4720 |
| 10148 | NA | NA | 0 |
| 10162 | 0.4572 | 0 | 0 |
| 10163 | 0.03793 | 0 | 0 |
| 10166 | 0.7866 | 1 | 5541 |
| 10172 | 0.1375 | 0 | 0 |
| 10173 | 0.4826 | 0 | 0 |
| 10175 | 0.02112 | 0 | 0 |
| 10180 | 0.05746 | 0 | 0 |
| 10186 | NA | NA | 0 |
| 10192 | 0.3808 | 0 | 0 |
| 10199 | 0.3682 | 0 | 0 |
| 10209 | 0.9409 | 1 | 5701 |
| 10210 | 0.1499 | 0 | 0 |
| 10214 | 0.05019 | 0 | 0 |
| 10215 | 0.3252 | 0 | 0 |
| 10216 | 0.706 | 1 | 5637 |
| 10232 | 0.3777 | 0 | 0 |
| 10239 | 0.3939 | 0 | 0 |
| 10249 | 0.02392 | 0 | 0 |
| 10253 | NA | NA | 0 |
| 10255 | 0.09654 | 0 | 0 |
| 10262 | 0.02873 | 0 | 0 |
| 10264 | 0.03736 | 0 | 0 |
| 10266 | 0.2502 | 0 | 0 |
| 10268 | 0.2075 | 0 | 0 |
| 10271 | 0.174 | 0 | 0 |
| 10272 | 0.1709 | 0 | 0 |
| 10276 | 0.2947 | 0 | 0 |
| 10277 | 0.05329 | 0 | 0 |
| 10279 | 0.2621 | 0 | 0 |
| 10281 | 0.0423 | 0 | 0 |
| 10285 | 0.004129 | 0 | 0 |
| 10294 | 0.2514 | 0 | 0 |
| 10300 | 0.1303 | 0 | 0 |

# Appendix B: R Code