Analysis of U.S.-Based Startup Company Success

Group 11: Hunter Willis and Brandon Coates

MA 5790

February 2022

Abstract:

The realm of startup investing has grown exponentially since the mid-1990's in the U.S. as an ever-increasing amount of capital has flowed into this space. This acceleration of capital is a result of the potential for massive investment returns for these small companies. However, as with any other asset class, the promise for outsized returns does not come without increased risk. The potential return from investing in startups, known as venture capital investing, is among the riskiest investments that can be made due to the large uncertainty of future cash flows and liquidity for an investor. The main goal of this study is to analyze the quantifiable variables of these startups to build a prediction model that accurately categorizes a startup as a success or a failure. To ensure a viable model is built, the predictor variable relationships will be explored both individually and between each other. Transformations that are deemed necessary will be outlined as well as the removal of predictors from the analysis. Based on the transformed predictor space, both linear and nonlinear predictive models will be fit to a training data set using a cross validation technique. Each of these models will be explored for its efficacy of accurate predictions and the top four models built on the training data will be utilized to predict data in the test set. Based on the testing set predictions, the overall best model will be chosen.

Contents

Abstract: 1

Background 3

Variable Introduction and Definitions 4

Preprocessing 4

Correlations 4

Transformations 5

Data Splitting and Resampling 7

Model Fitting 8

Summary 11

Appendix 1: Supplemental Material for Classification Models 12

R Code 23

Background

The U.S. has long been known to be the world's foremost center of innovation which is a product of robust capital markets, relatively easy access to capital, and an entrepreneur-friendly regulatory structure. One of the main levers that has accelerated the trend of startups is the easy access to capital. In addition to this, the amount of initial investment needed to start a business has steadily declined over time as well due to the business-friendly regulatory structure in the U.S. This development picked up steam in the mid-1990's and caused the number of startups entering the market to increase at an exponential rate through today. However, there is a need for an increasing amount of capital as a startup grows and has success, which has caused a new asset class to be born over time to fill that void of capital that these companies require.

Because of the robust, well-oiled capital markets in the U.S., a new asset class evolved to fill this void of growth capital for small, growing startups. This new asset class is known as venture capital investing. Venture capitalists specialize in providing capital to startups that need an investment to keep their growth moving forward. These investors see the innovation that is being developed by the founders of these companies and attempt to capture the massive returns that can be realized as a small business becomes a large enterprise.

As with any other type of investment, venture capital investing does not come without risk. In fact, because of the prospect for some of the highest returns able to be realized in the investing world, this asset class has been one of the most volatile. The reason for this is due to the unproven nature of the companies that are being invested in. These startups often have very little revenue generation, no profits, and may not even have a viable product to sell. For this reason, success in venture capital investing, much more so than any other asset class, often comes down to a binary outcome. Much of the time, the business either fails to produce growth and is closed as provided capital dries up or the company's product or service is wildly successful, and growth is exponential. In addition, the overall success rate of venture capitalists picking successful startups to invest in is extremely low.

The success rate of venture capitalists' investments can often be less than 1% of businesses that produce outsized returns. The need for some percentage of wildly successful investments is required for venture capital funds because of the losses incurred on most of the businesses that are invested in. Venture capitalists seek to increase their ratio of successes to failures to realize higher returns than their competitors in the asset class. In the venture capital space, a success is most often defined by one of their investments going public through an IPO (initial public offering) or getting acquired by a larger firm. These events produce liquidity for venture capitalists and the ability to realize the returns produced from the company's growth. On the other hand, when a company fails to capitalize on their growth strategy, capital usually stops flowing into the company and it is ultimately closed. This usually results in a complete loss for a venture capital firm.

Variable Introduction and Definitions

The dataset that was utilized can be found on Kaggle.com. It contains data on 196,553 startups that have been founded since 1901. The dependent variable for this study is a product of the company's fate. The data contains information on whether the company was acquired, IPO'ed, is still operating, or was closed. Because a measure of success versus failure was one of the goals of the study, the dependent variable was altered to be a success if the company was acquired or went public through an IPO and a failure if the startup was closed. Companies with a status of operating were excluded because they cannot yet be classified as a success or a failure. In addition to the dependent variable, there are 28 predictors. Below is a list of the predictors and their descriptions.

Variable Name	Description
Status	End state of the company: success (acquired, IPO), failure (closed)
Funding Rounds	Type of funding rounds: angel, venture, round A, round B, round C, debt and/or private equity
State Code	Place where company was founded: Maine (MA), Texas (TX), California (CA), New York (NY), other state (other)
Industry Code	Industry the startup operates in: software, web, mobile, enterprise, advertising, video games, ecommerce, consulting, biotech, other
Time to first round	Amount of time between company founding and first round of investment
Time to last round	Amount of time between company founding and last round of investment
Amount raised	Aggregate amount of money the company has raised
Time to first milestone	Amount of time between company founding and hitting first milestone
Time to last milestone	Amount of time between company founding and hitting last milestone
Funding rounds	Overall number of funding rounds the company has gone through
Milestones	Number of milestones the company has achieved

The relationship between the company's status and the predictor variables will be explored. The data will also be preprocessed and analyzed for needed transformations and eliminations of predictor variables. Following this, both linear and nonlinear classification models will be fit and analyzed to determine the best predictive model utilizing the area under the receiver operating characteristic.

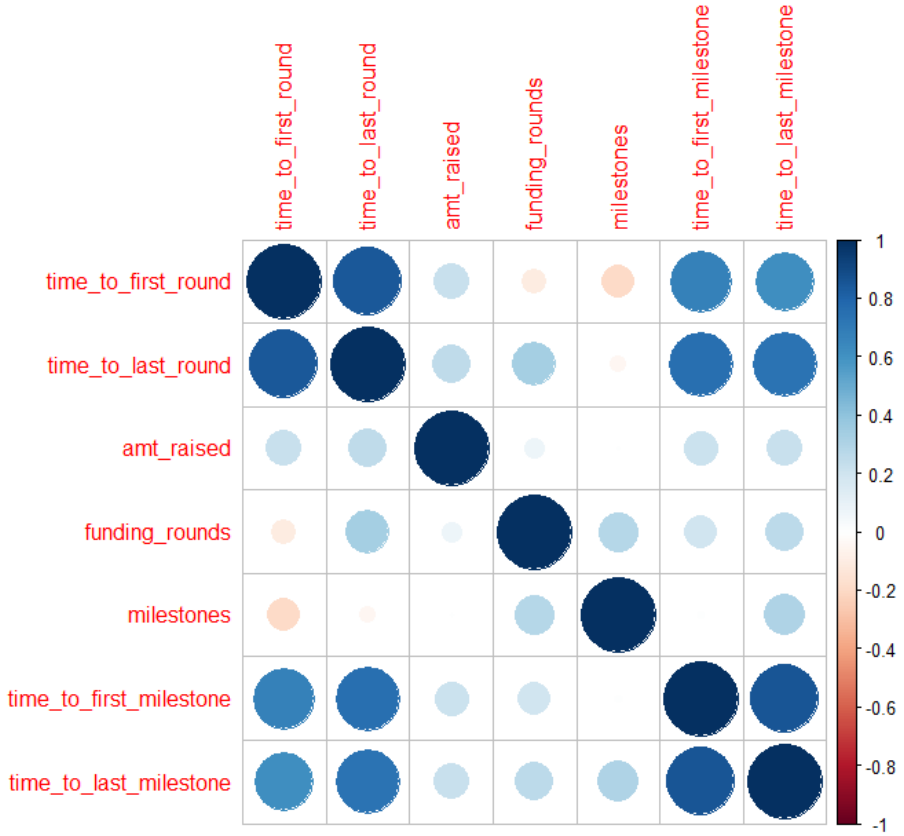
Preprocessing

The first step in the analysis is to preprocess the data. Prior to this, dummy variables needed to be created for the categorical predictors. These predictors include state code, industry code, and type of funding rounds. This created a total of 28 predictor variables. Utilizing a near zero variance procedure, degenerate variables were then found and removed from the data set. This included Texas for the state code category and ecommerce and consulting for the industry code category. Because these categories are still needed to conduct the analysis, they were added back to the other categories for each respective predictor. This left a total of 25 predictors for the correlation analysis and transformations.

Correlations

A correlation plot of the 7 continuous and discrete variables was created to visualize pairwise correlations between these predictors that were large. Below is the correlation plot with the blue representing positive correlations between predictors and red representing negative correlations between predictors. To remove highly correlated predictors, a threshold of 0.7 was utilized to remove

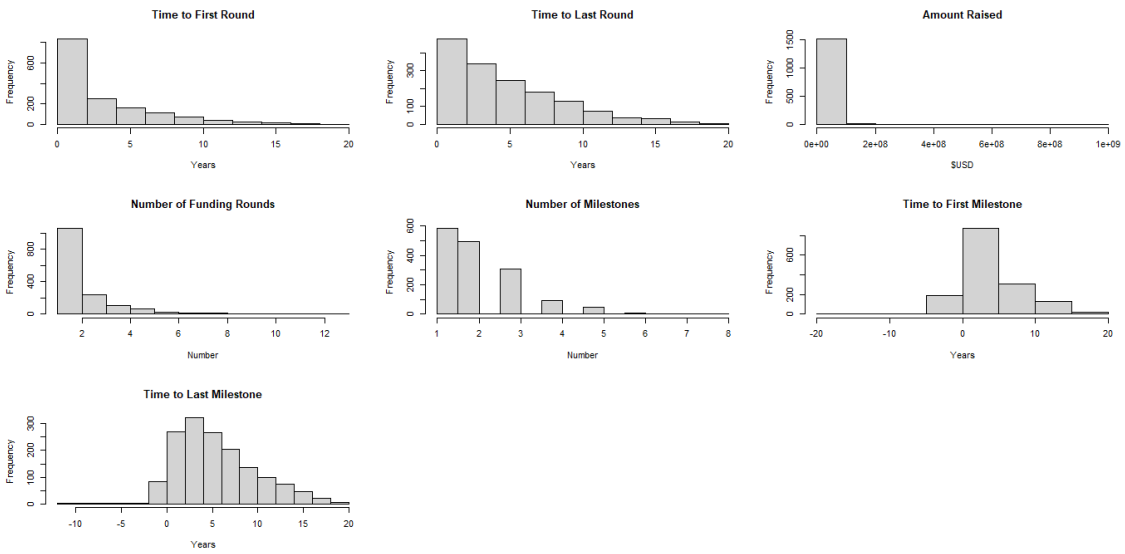
predictors with pairwise correlations higher than this. Time to last milestone and time to last round were the only two predictors that were removed.



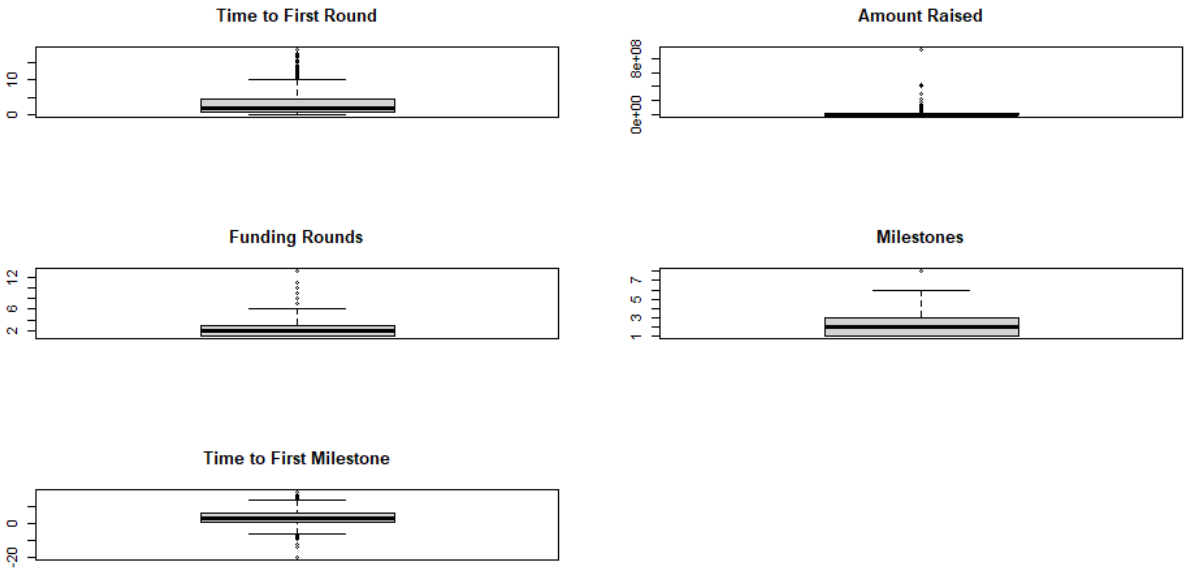
Transformations

After pruning highly correlated variables, the distributions of the continuous and discrete variables were then analyzed. Below are the skewness values, histograms, and boxplots for these predictors before any transformations were applied. From these charts and tables, it is illustrated that most of these predictors are highly skewed and have outliers. The table below shows that time to first round, amount raised, funding rounds, and milestones all exhibit significant right skew.

Skewness Values	Time to First Round	Amount Raised	Funding Rounds	Milestones	Time to First Milestone
Pre-Transformation	1.591	15.599	2.185	1.133	0.705

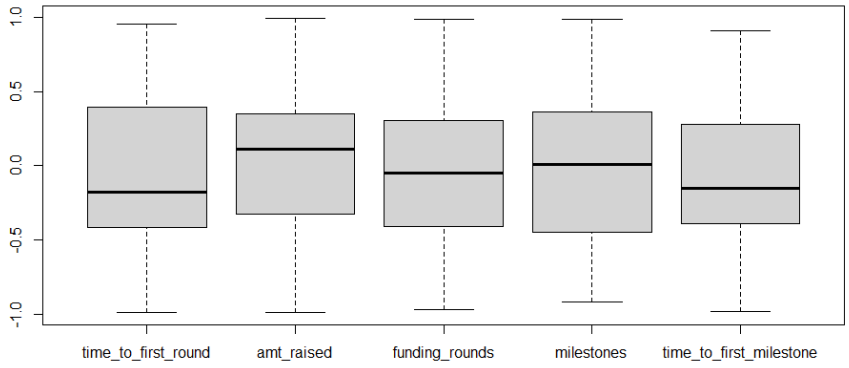


The boxplots for these predictors show that all of them contain significant outliers, specifically amount raised.

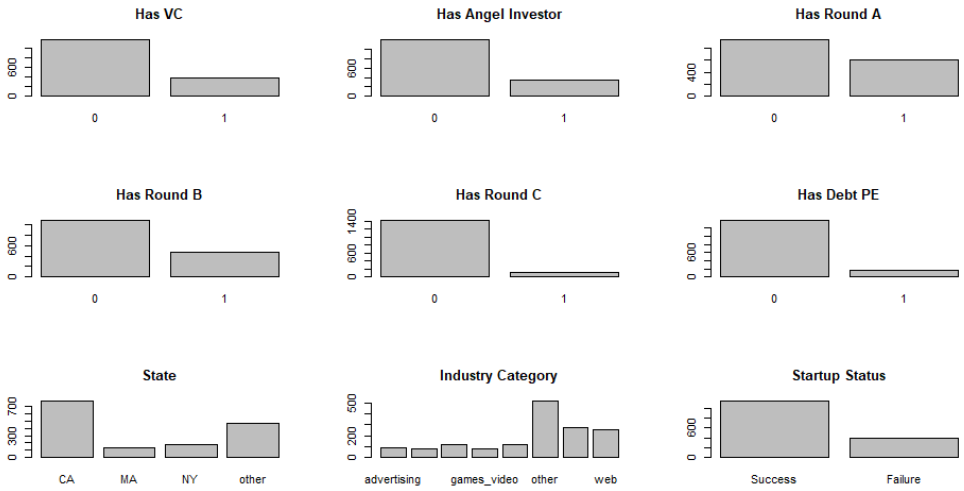


To account for the skewness and differing scales of the continuous and discrete variables, the data was first centered and scaled. A Box Cox transformation was also needed to correct for skewness in the data, however, several of the predictors contained data with negative values. To correct for this, the constant 21 was added to each of these predictors to make them positive values only. After performing the Box Cox transformation, it was found that skewness was still present in the data due to large outliers. The spatial sign transformation was applied to these predictors to correct for the outlier data points. Below is a table of the skewness values and a boxplot containing all the continuous and discrete predictors after centering, scaling, and applying the Box Cox and spatial signs transformations to them. All these predictors do not exhibit any major skewness or outliers after the transformations were applied.

	Time to First Round	Amount Raised	Funding Rounds	Milestones	Time to First Milestone
Pre-Transformation	1.591	15.599	2.185	1.133	0.705
Post-Transformation	0.360	-0.374	0.477	0.257	0.363



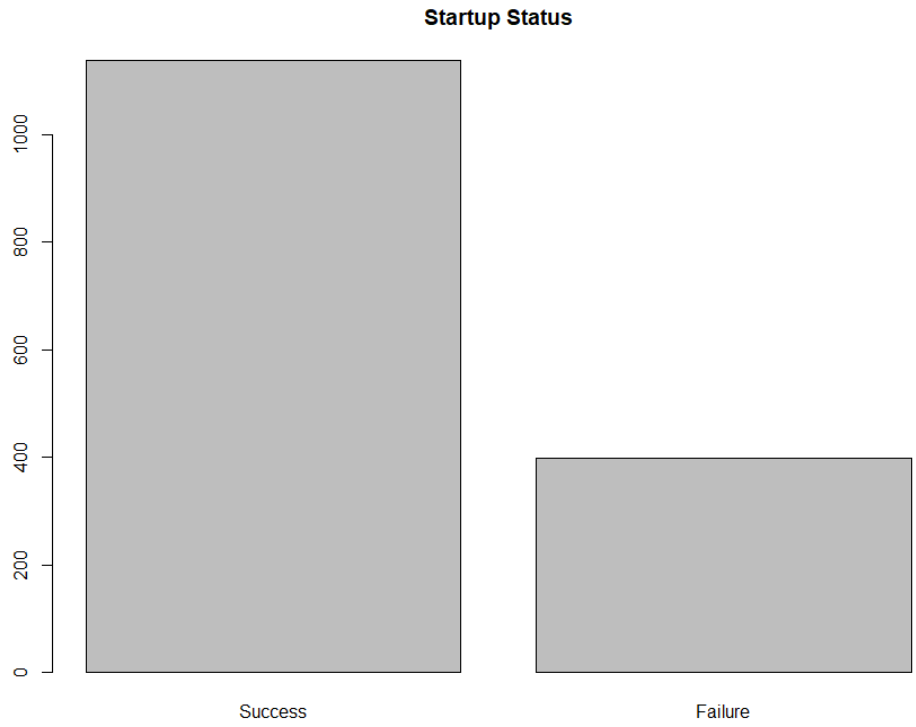
Below are the histograms for each categorical predictor and the response variable, startup status. These plots show that the frequencies are imbalanced for most predictors.



Data Splitting and Resampling

Below is a plot of the startup response variable, startup status. The response variable exhibits a large imbalance between the success and failure categories. For this reason, stratified random sampling was utilized to split the data. This will allow the proportions of the success versus failure classes to remain the same for the testing and training sets relative to the overall dataset. In addition, because of the overall size of the data set (1,535 samples), the data was split with 60% being allocated to the training

set and 40% being allocated to the testing set. This left 922 observations in the training set and 613 in the testing set.



The resampling technique that was chosen to fit the models on the training data was LGOCV (leave group out cross validation). This partitions the training data into 75% of the samples used for trainings and 25% held out to test on the trained model. This is repeated 25 times for each fitted model and performance is averaged over the repeated models.

Model Fitting

Given that the goal of the model was to determine if a startup was a success or failure, classification models were trained on the training data set to determine the top four optimal models that would be used to assess the model with the best predictive capabilities on the test data set. Both linear and non-linear models were trained on the training data set and assessed using the area under the ROC curve. The results from this assessment can be seen in the table below, which lists each of the models that were trained on the training data set and their optimized tuning parameter, area under the curve, sensitivity, specificity, and Kappa values. The tuning parameter plots and corresponding ROC curves can be found in the Appendix.

Linear Classification Models					
Model	Tuning Parameter	AUC	Sensitivity	Specificity	Kappa
Logistic Regression	N/A	0.760	0.902	0.407	0.345
Linear Discriminant Analysis	N/A	0.756	0.897	0.422	0.351
PLS Discriminant Analysis	Number of Components = 8	0.756	0.927	0.363	0.337

Penalized Regression Model	$\alpha = 0;$ $\lambda = 0.178889$	0.770	0.980	0.130	0.149
Non-Linear Classification Models					
Mixture Discriminant Analysis	Subclasses = 1	0.756	0.864	0.449	0.330
Neural Network	Size = 8; Decay = 0.5	0.760	0.886	0.409	0.323
Flexible Discriminant Analysis	Degree = 1; Nprune = 7	0.755	0.905	0.415	0.349
Support Vector Machine	$\sigma = 0.01552198$ C = 0.5	0.744	0.951	0.265	0.268
K-Nearest Neighbors	K = 210	0.769	0.973	0.127	0.136
Naïve Bayes	N/A	0.758	0.959	0.258	0.273

It was found that the optimum model in terms of AUC was the Penalized Regression model, followed by the K-Nearest Neighbor model, the Logistic Regression model, and finally the Neural Network model. It was noted that while the Penalized Regression and K-Nearest Neighbors models had the highest AUC values, they also had low specificity values, which indicated that these models have a bias to predicting startup success. The other two models seem to have less of this bias as well as better Kappa values. These values will be of interest when detailing the final recommendation on the predictive model.

These models were then used to predict the results of the testing data set that was constructed to gauge how well they performed on a new data set. The results of these predictions can be seen in the table below, The ROC curves from these models can be seen in the Appendix. It was found that the Neural Network model had the highest area under the ROC curve value when predicting on the testing data set and also had the highest Kappa value. The Kappa value is also an important metric since Kappa value between 0.3-0.5 is desired to show that there is reasonable agreement between the observed accuracy of the model to the expected accuracy based on the marginal totals of the confusion model. We must also consider the specificity in the models due to the class imbalances of the response variable in the data set, a low specificity value will indicate that the model would be biased towards predicting false positive responses, so for this case, the model will overly predict startup successes. So we found that the models that have proper Kappa values and high specificity values were the Logistic Regression model and the Neural Network model.

Model	AUC	Sensitivity	Specificity	Kappa
Logistic Regression	0.780	0.897	0.447	0.374
Penalized Regression	0.782	0.967	0.264	0.292
Neural Network	0.786	0.903	0.465	0.402
K-Nearest Neighbors	0.771	0.976	0.226	0.262

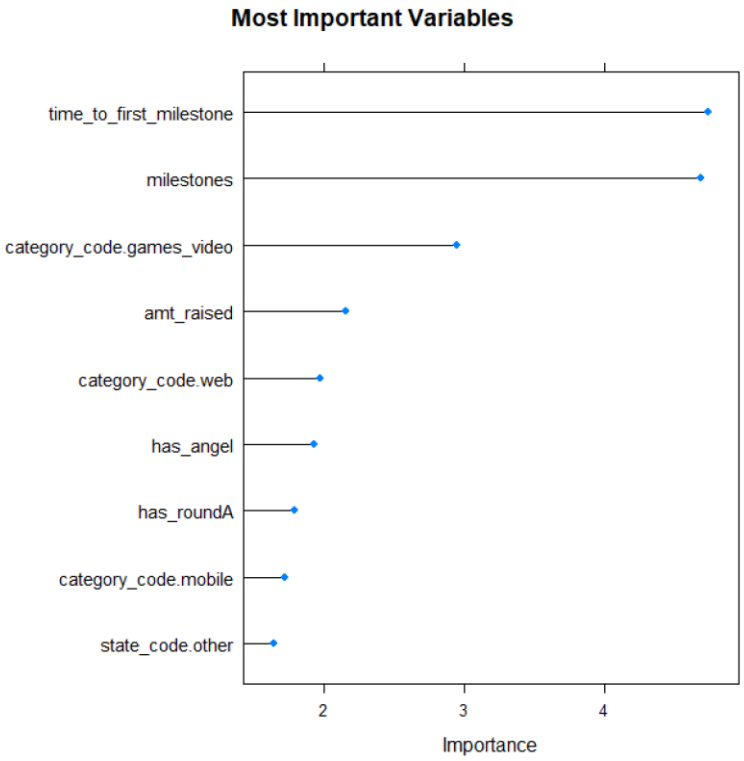
Given the results of the predictions, the optimal model that was selected was the logistic regression model. The reason this model was chosen over the Neural Network model, even though it resulted in better values, was due to the fact that the computational efficiency of this model is better while

resulting in minimal degradation to the assessed accuracy metrics. The Neural Network model only slightly increases the chance of proper failure prediction, but takes much more computational resources to run compared to the simplistic Logistic Regression model. Though it was not the overall goal, but the Logistic Regression model also allows for more interpretability than the Neural Network model, which may allow for a greater understanding of how the predictor variables affect the prediction.

When examining the results of the predictions on the testing data set for the logistic regression model, it was found that the model was able to successfully predict 407 instances of startup successes, while incorrectly predicting successes in 88 instances, which is reflective of the high sensitivity value that was found. The model also correctly predicted 71 instances of startup failure, while mis-predicting 47 successes as failures. Given the class imbalances in the response data, the specificity is relatively high with a value of 0.447, which indicates that the model is likely to mis-predict startup failure half of the time, since the false positive rate would be 0.553.

Logistic Regression		
Prediction	Reference	
	Success	Failure
Success	407	88
Failure	47	71

Given that the Logistic Regression model was chosen as the optimum model, the top five predictor variables were determined using the variable importance factor. A plot of the variable importance factor can be seen below in the plot. It was found that the time to first milestone, number of milestones, is the startup was a video game company, the amount of funding raised, and if the startup fell into the category of providing web services had the largest impact into whether the startup would be successful or not.



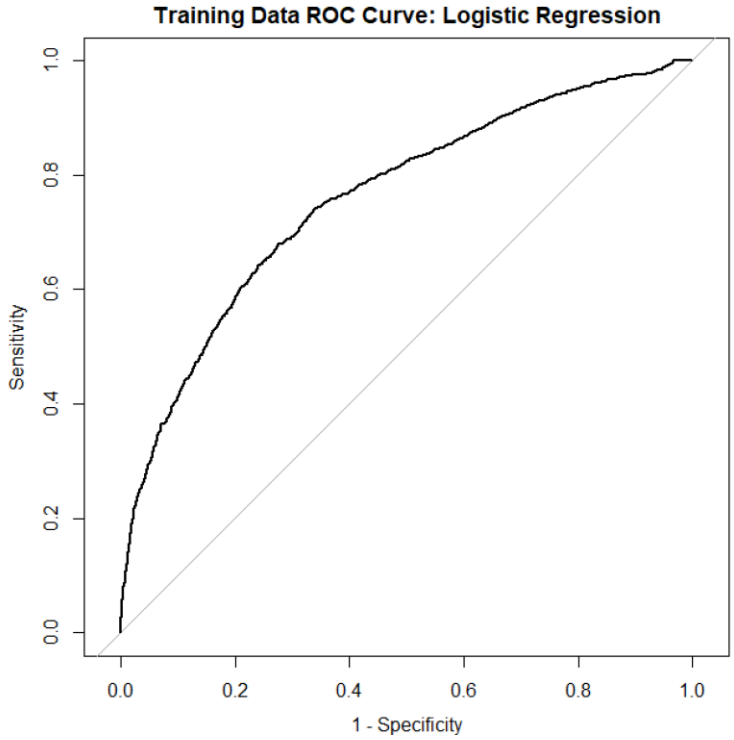
Summary

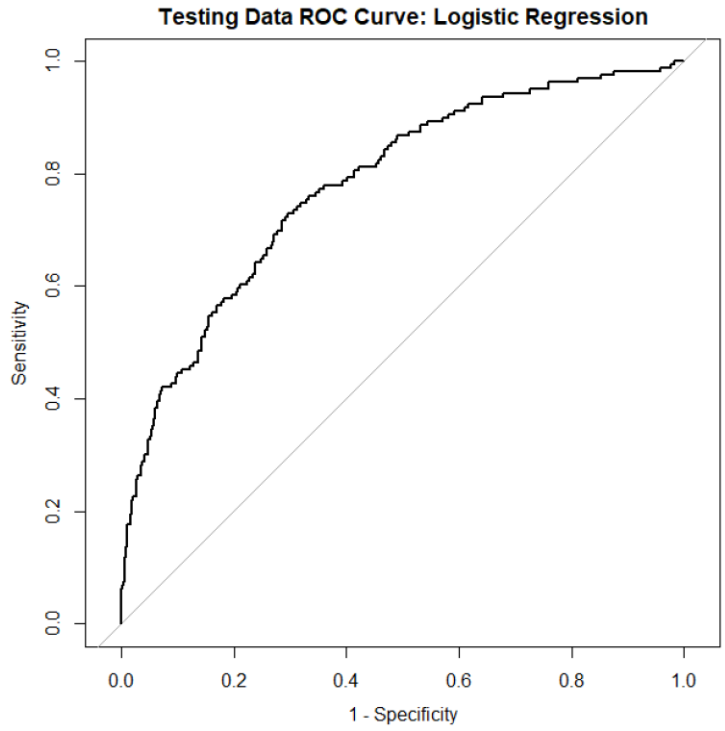
From the work conducted in this project, it was concluded that the best model was the logistic regression model used to predict the success or failure of a startup. This model was chosen over other classification models for its high accuracy and computational efficiency. The resulting area under the curve value was found to be 0.780 with a sensitivity of 0.897 and specificity of 0.447. Given the class imbalances in the response variable of the data set, it was important to choose a model that had a larger specificity rate since the models were biased to responding in false positives and the logistic regression model had one of the highest values in this regard. Though this model was chosen, further data collection and model training would be recommended to attempt to remove the class imbalances to build an even more accurate model.

Appendix 1: Supplemental Material for Classification Models

A.) Logistic Regression Model

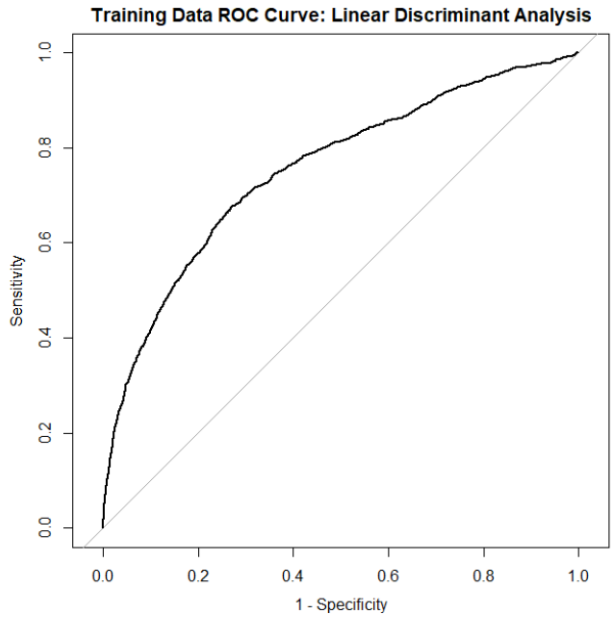
The ROC curves for this model when used on the training set and testing set can be seen below.





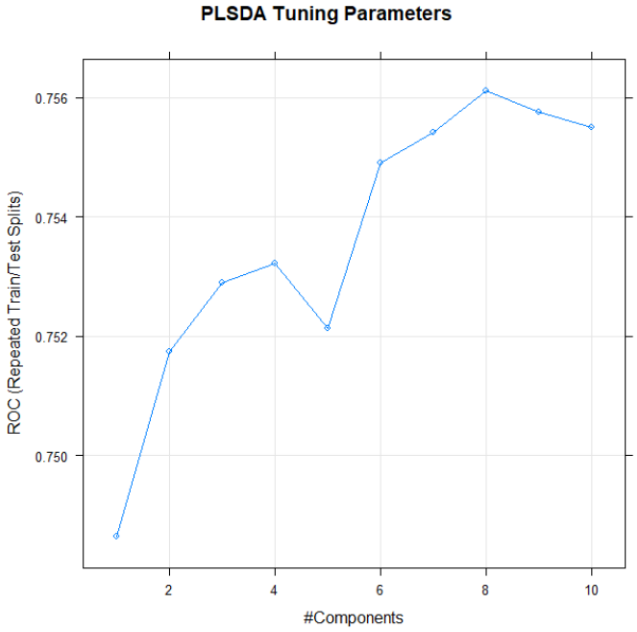
B.) Linear Discriminant Analysis

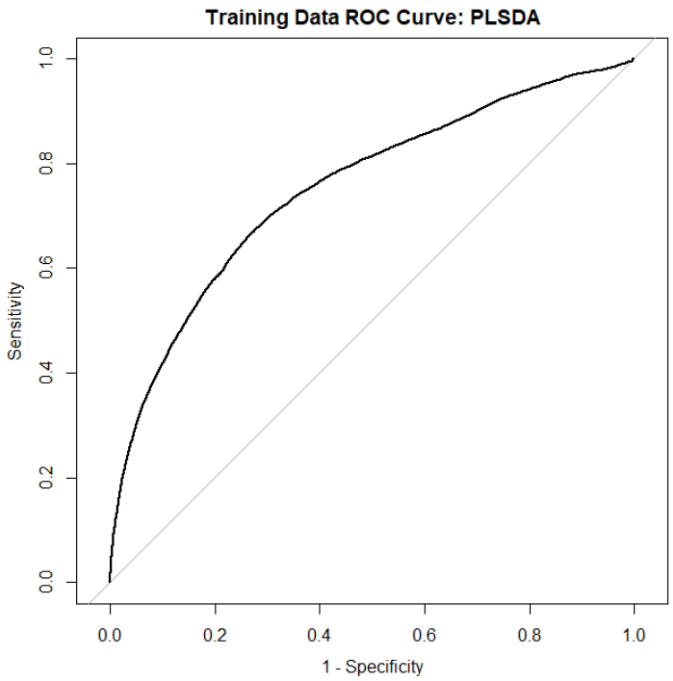
The ROC curves for this model when used on the training set and testing set can be seen below.



C.) Partial Least Squares Discriminant Analysis

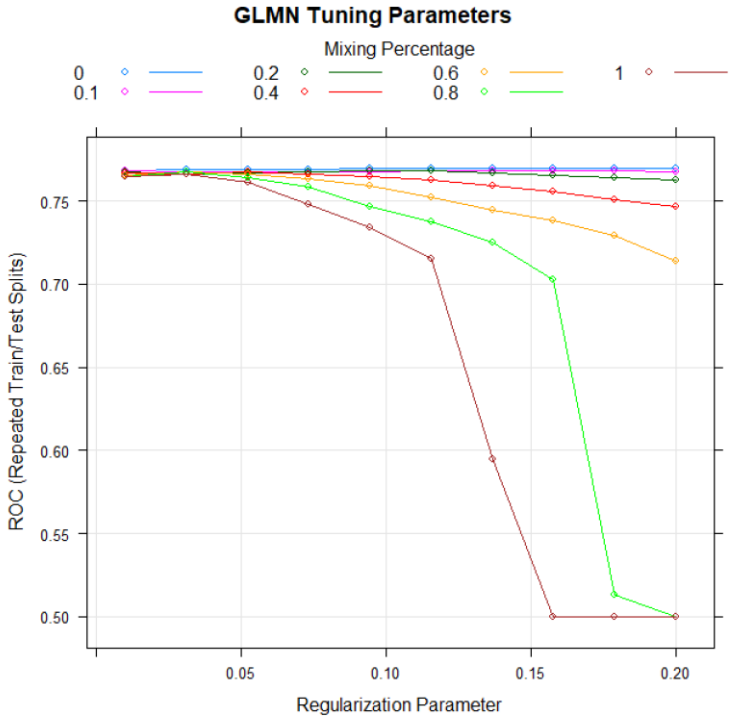
The tuning parameter plot for this model can be seen below, it was found that the optimal tuning parameter was 8 components. The ROC curve for the model training can also be seen below.

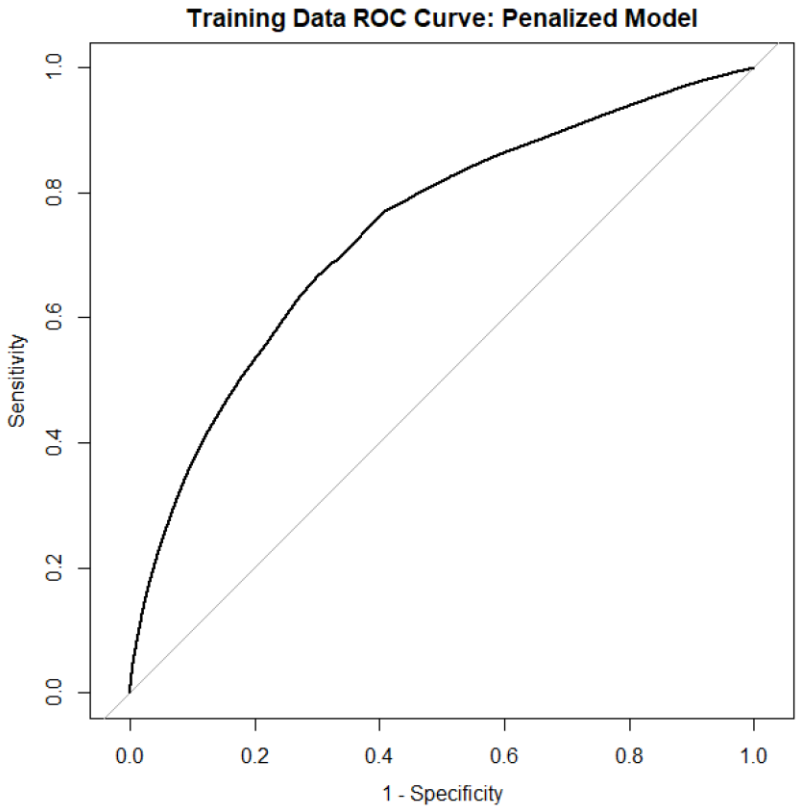


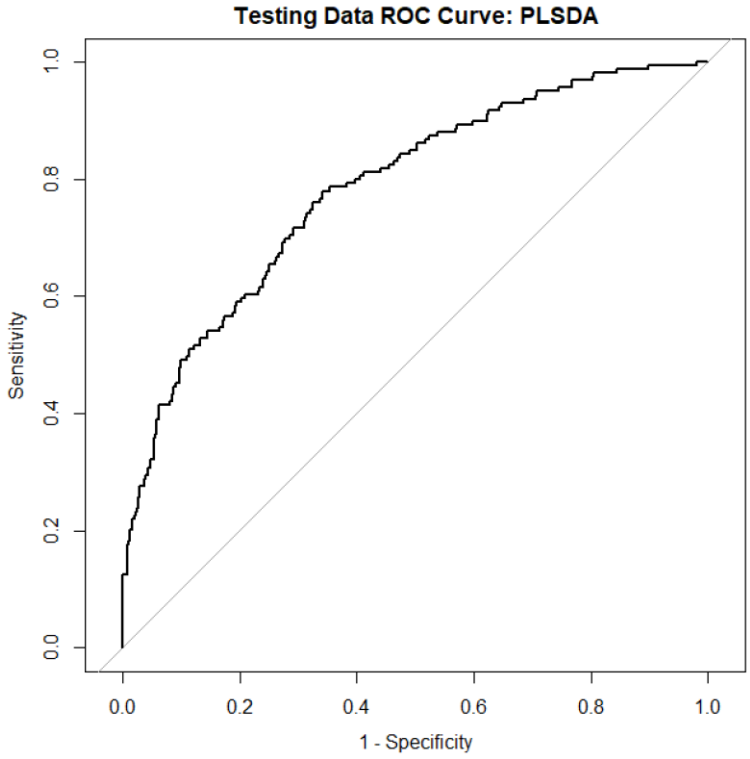


D.) Penalized Logistic Regression Model

The tuning parameter plot for this model can be seen below where it was found that the optimal tuning parameters were $\alpha = 0$ and $\lambda = 0.178889$. The ROC curves for when the model was trained and tested on the testing data set can also be seen.

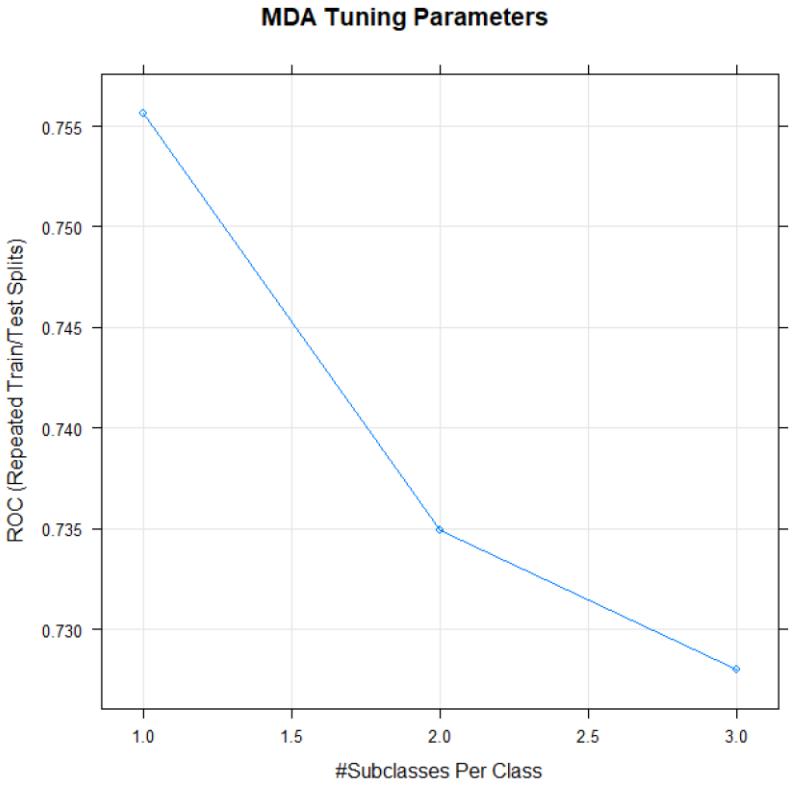


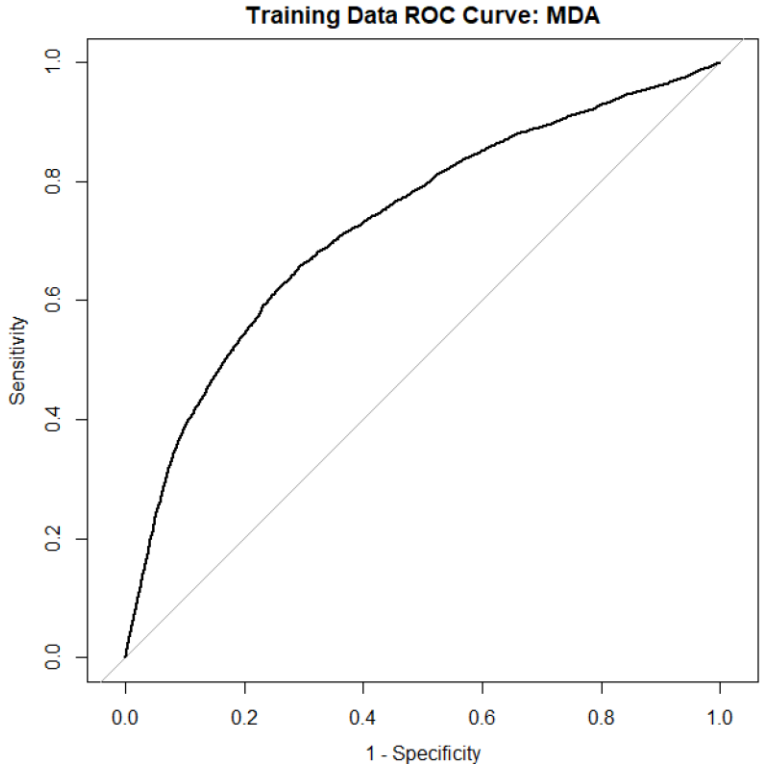




E.) Mixture Discriminant Analysis

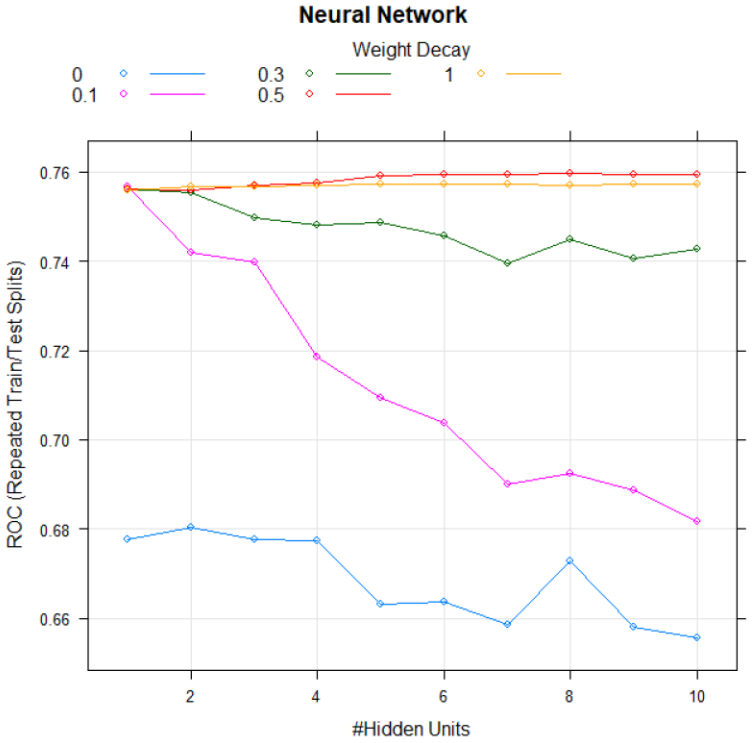
The tuning parameter plot for this model can be seen below, it was found that the optimal tuning parameter for this model was 1 subclass. The ROC curve for when the model was trained can also be seen.

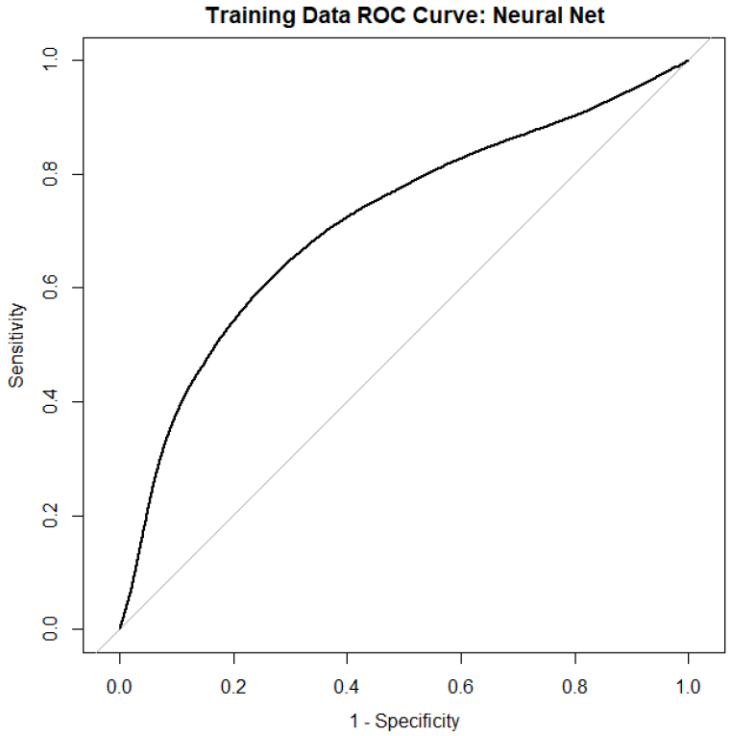


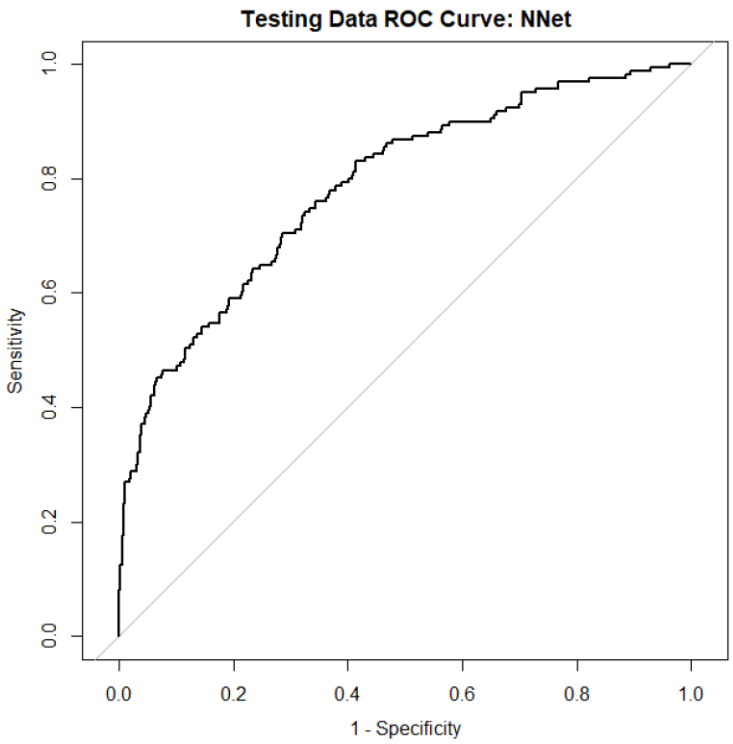


F.) Neural Network

The tuning parameter plot for this model can be seen below where it was found that the optimal tuning parameters were size of 10 and decay of 1. The ROC curves for when the model was trained and tested on the testing data set can also be seen.

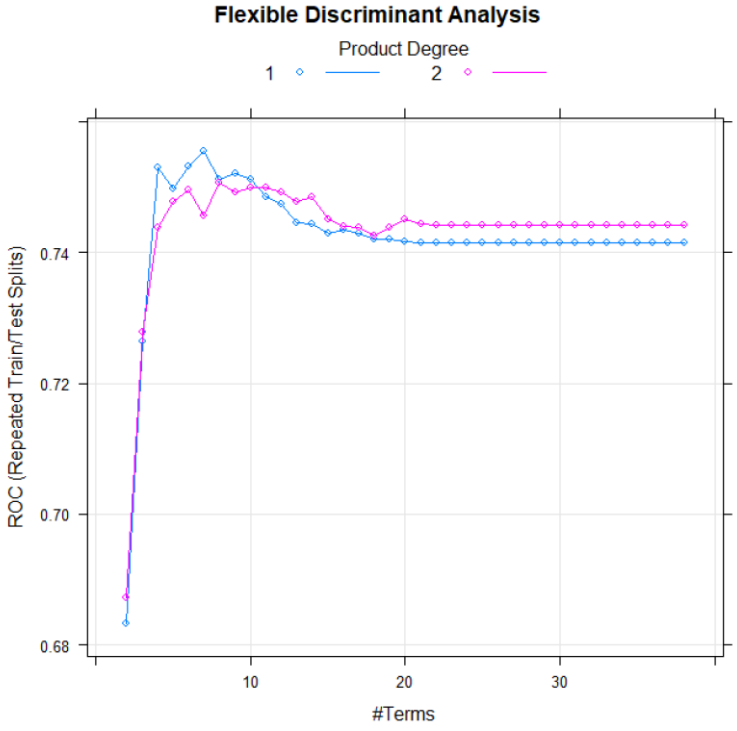


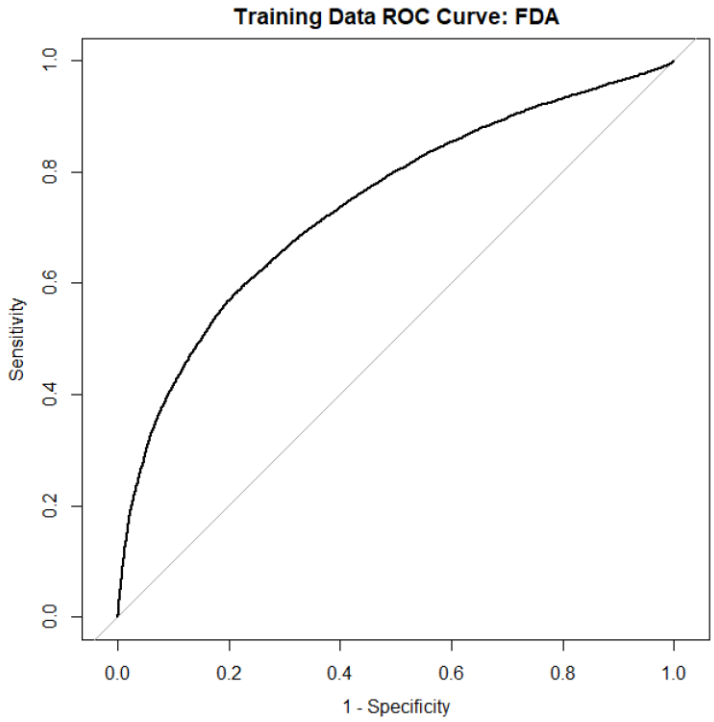




G.) Flexible Discriminant Analysis

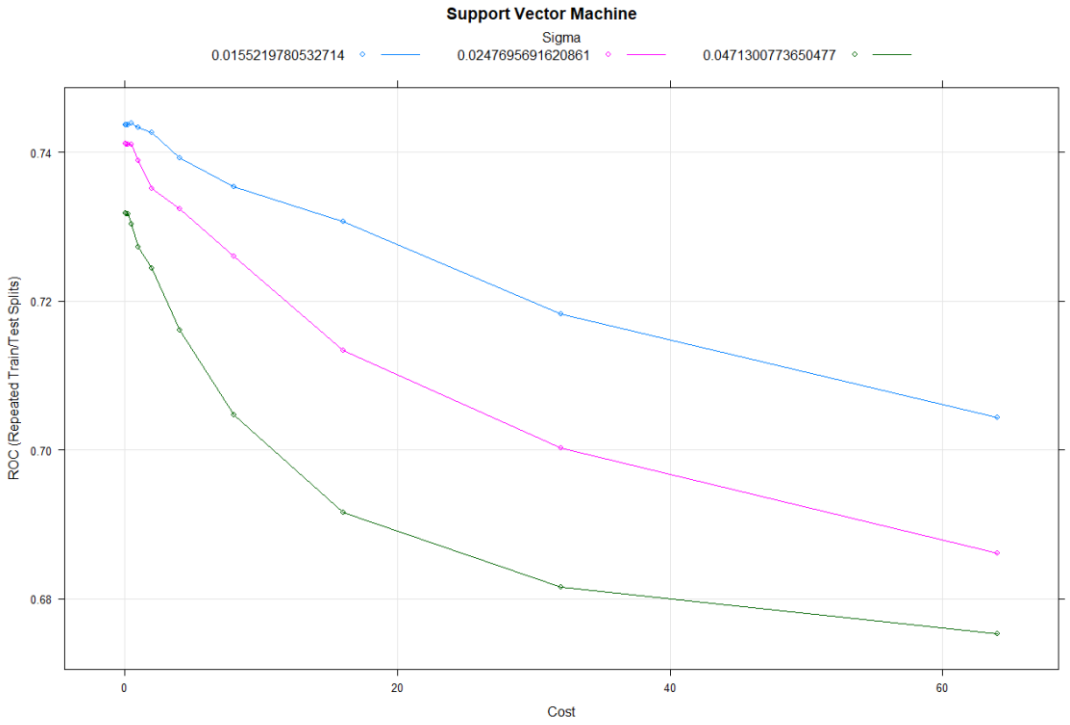
The tuning parameter plot for this model can be seen below, it was found that the optimal tuning parameters for this model was 1 degree and 7 prunes. The ROC curve for when the model was trained can also be seen.

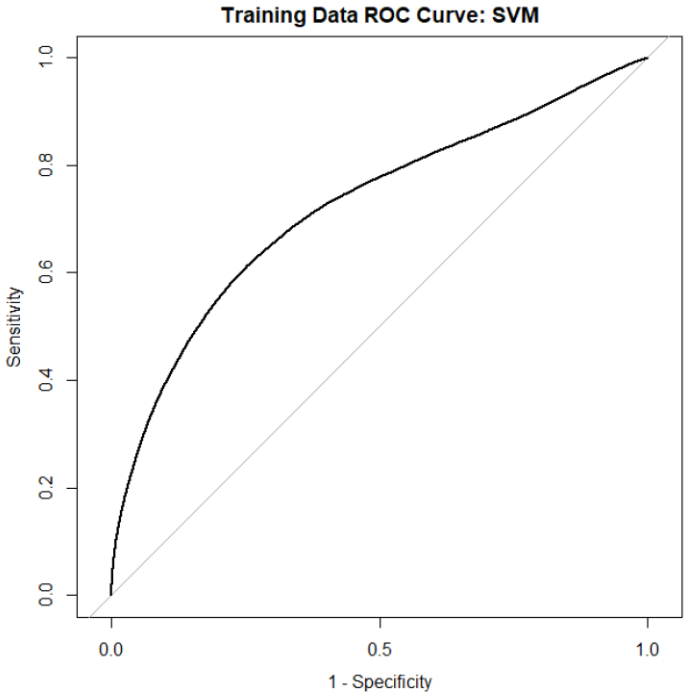




H.) Support Vector Machine

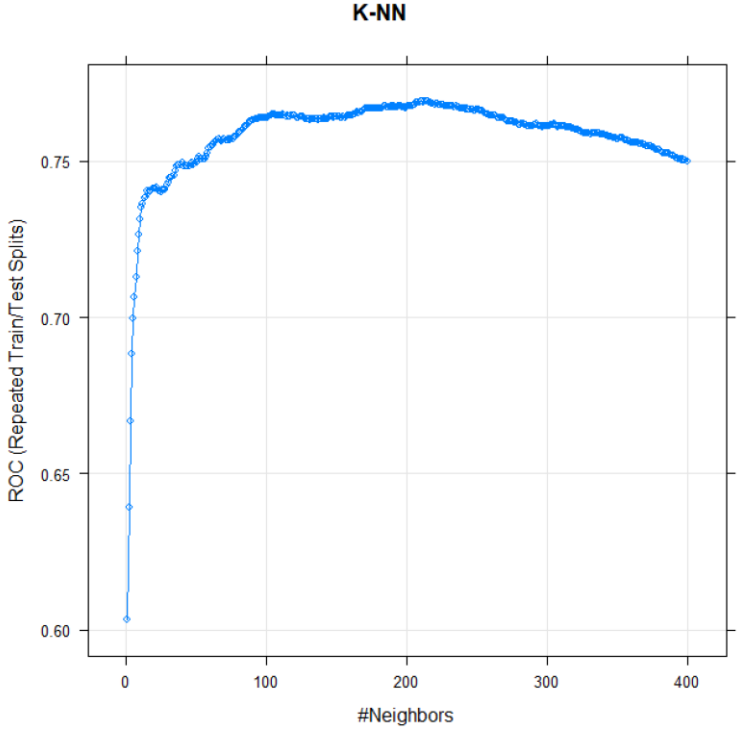
The tuning parameter plot for this model can be seen below, it was found that the optimal tuning parameters for this model was $\sigma = 0.01552198$ and C = 0.5. The ROC curve for when the model was trained can also be seen.

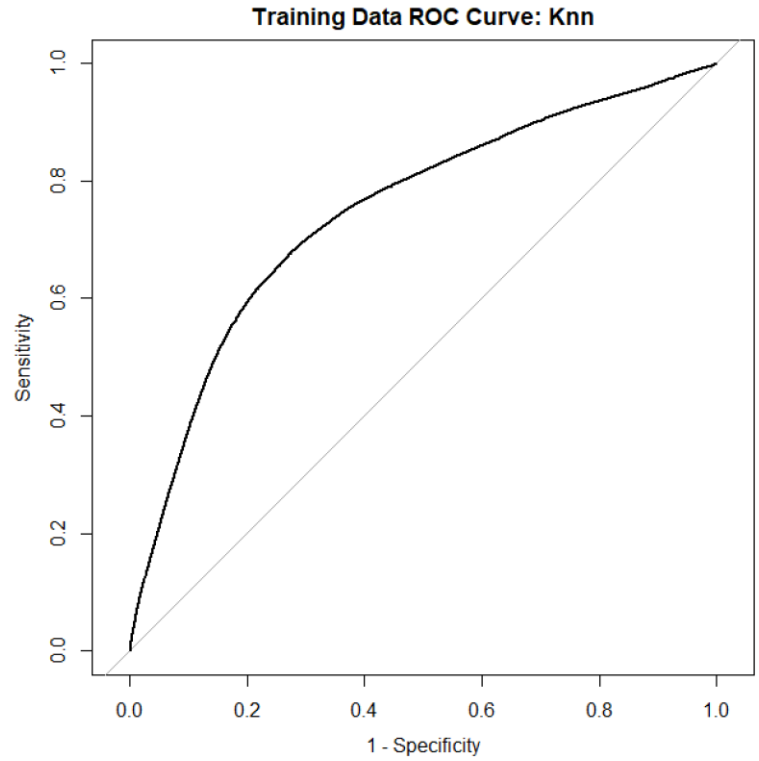


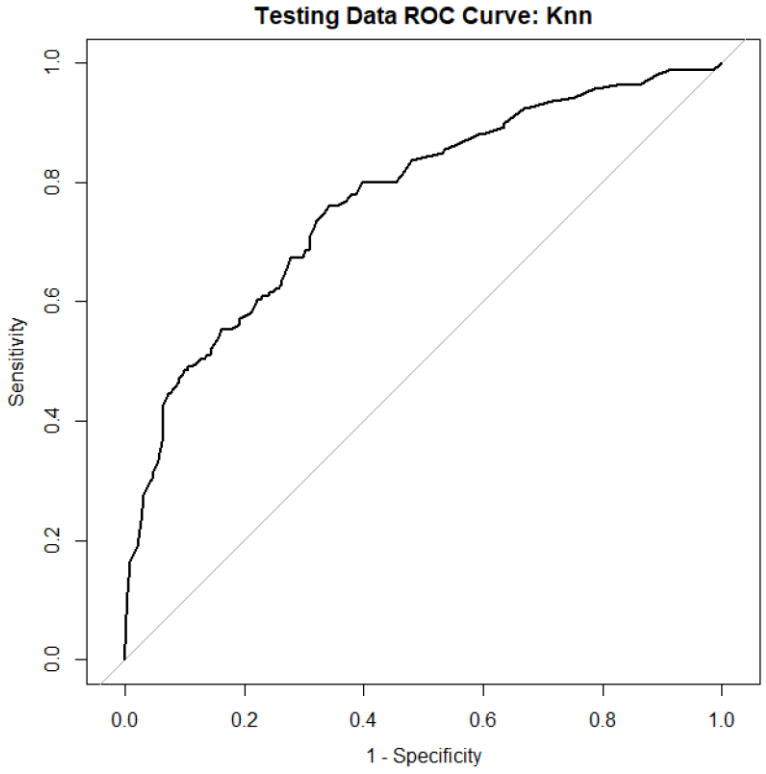


I.) K-Nearest Neighbors

The tuning parameter plot for this model can be seen below where it was found that the optimal tuning parameter was 97 neighbors. The ROC curves for when the model was trained and tested on the testing data set can also be seen.

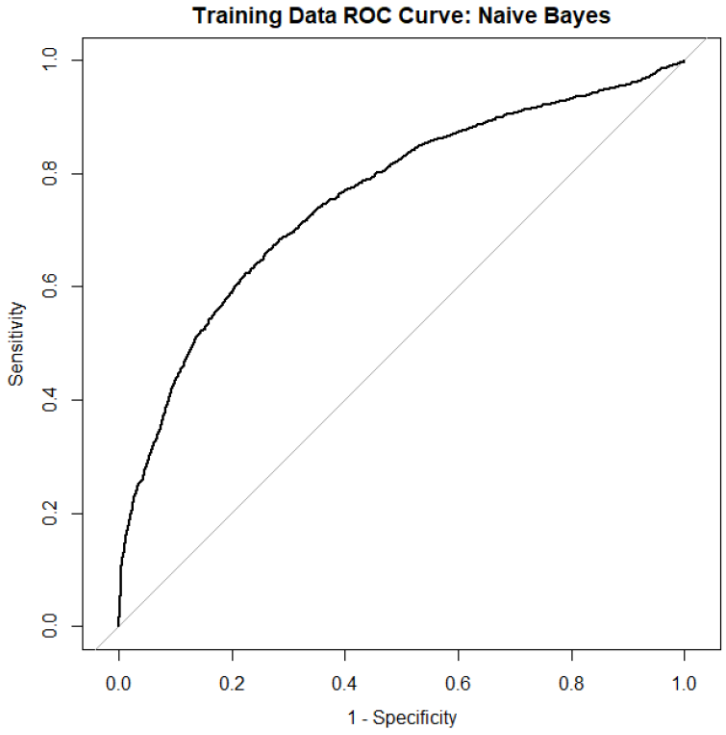






J.) Naïve Bayes

The ROC curve for this model when being trained to the training data set can be seen below.


Training Data ROC Curve: Naive Bayes
Sensitivity
1 - Specificity
0.0
0.2
0.4
0.6
0.8
1.0

R Code

```
#########################################################
#MA 5790 - Final Project: Startup Classification
#########################################################


library(tidyverse)
library(dplyr)
library(mlbench)
library(caret)
library(AppliedPredictiveModeling)
library(e1071)
library(corrplot)
library(pROC)
library(MASS)
library(mda)
library(nnet)
library(earth)
library(kernlab)
library(klaR)
library(svmpath)


#load data
startups <- read.csv("C:/Users/bjc_2/Documents/MA5790/Final Project/StartupData.csv")
attach(startups)


#Pre-Processing
startups <- filter(startups, country_code == "USA", founded_at > as.Date("1994-12-31"),
state_code != "",
                   funding_rounds > 0, category_code != "", amt_raised > 0, time_to_first_milestone !=
"NA",
                   time_to_last_milestone != "NA", time_to_first_round >= 0, time_to_last_round >= 0)
```

```
startups$status <- as.factor(startups$status)

startups <- filter(startups, status != "operating")

levels(startups$status) <- c("Success","Failure","Success","Success")

startups$has_VC <- as.numeric(startups$has_VC)

startups$has_angel <- as.numeric(startups$has_angel)

startups$has_roundA <- as.numeric(startups$has_roundA)

startups$has_roundB <- as.numeric(startups$has_roundB)

startups$has_roundC <- as.numeric(startups$has_roundC)

startups$has_debtPE <- as.numeric(startups$has_debtPE)

startups$time_to_first_milestone <- as.numeric(startups$time_to_first_milestone)

startups$time_to_last_milestone <- as.numeric(startups$time_to_last_milestone)


#state dummy variables

startups$state_code <- as.factor(as.character(ifelse(startups$state_code == 'CA', yes = "CA", no =
ifelse(startups$state_code == 'NY',

                yes = "NY", no = ifelse(startups$state_code == 'MA',

                yes = "MA", no = ifelse(startups$state_code == 'TX', yes = "TX", no = "other"))))))


#industry category dummy variables

startups$category_code <- as.factor(as.character(ifelse(startups$category_code == 'software', yes
= "software",

                    no = ifelse(startups$category_code == 'web', yes = "web",

                    no = ifelse(startups$category_code == 'mobile', yes = "mobile",

                    no = ifelse(startups$category_code == 'enterprise', yes = "enterprise",

                    no = ifelse(startups$category_code == 'advertising', yes = "advertising",

                    no = ifelse(startups$category_code == 'games_video', yes = "games_video",

                    no = ifelse(startups$category_code == 'ecommerce', yes = "ecommerce",

                    no = ifelse(startups$category_code == 'consulting', yes = "consulting",

                    no = ifelse(startups$category_code == 'biotech', yes = "biotech",

                    no = "other"))))))))))
```

```
dmy <- dummyVars(formula = ~ state_code + category_code,
                 data = startups,
                 fullRank = TRUE)
dummy.vars <- data.frame(predict(dmy, newdata = startups))
head(dummy.vars, n = 10)

#Combined Dataset
startups.predictors <- cbind(dummy.vars, startups[12:24])
summary(startups.predictors)
startups.response <- startups[3]
summary(startups)



###################################
# Data Exploration
###################################

#Split data into categorical and quantitative group
startupsColNum <- cbind(startups.predictors[,14:17], startups.predictors[,24:26])
head(startupsColNum)
summary(startupsColNum)

startupsCatVar <- cbind(startups.predictors[,1:13], startups.predictors[,18:23])
head(startupsCatVar)

###################################
# Numerical Predictor Examination
###################################
#Create histograms of numeric variables
par(mfrow = c(3,3))
```

```
hist(startupsColNum$time_to_first_round, main = "Time to First Round", xlab = "Years")
hist(startupsColNum$time_to_last_round, main = "Time to Last Round", xlab = "Years")
hist(startupsColNum$amt_raised, main = "Amount Raised", xlab = "$USD")
hist(startupsColNum$funding_rounds, main = "Number of Funding Rounds", xlab = "Number")
hist(startupsColNum$milestones, main = "Number of Milestones", xlab = "Number")
hist(startupsColNum$time_to_first_milestone, main = "Time to First Milestone", xlab = "Years")
hist(startupsColNum$time_to_last_milestone, main = "Time to Last Milestone", xlab = "Years")

#Create Scatter plot matrix to examine relationships between predictors
panel.cor <- function(x , y, digits = 2, prefix = "", cex.cor, ...) {
  usr <- par("usr")
  on.exit(par(usr))
  par(usr = c(0, 1, 0, 1))
  Cor <- cor(x, y) # Remove abs function if desired
  txt <- paste0(prefix, format(c(Cor, 0.123456789), digits = digits)[1])
  if(missing(cex.cor)) {
    cex.cor <- 0.4 / strwidth(txt)
  }
  text(0.5, 0.5, txt,
       cex = 1 + cex.cor * Cor) # Resize the text by level of correlation
}

pairs(startupsColNum,
      data = startupsColNum,
      upper.panel = panel.cor,
      lower.panel = panel.smooth)

#Calculate correlation between predictors
correlation <- cor(startupsColNum)
correlation
```

```
corrplot(correlation)

#Find high correlation predictors
highcor <- findCorrelation(correlation, 0.7)
colnames(startupsColNum)[highcor]

#Remove highly correlated predictors
startupsColNum.New <- startupsColNum[, -highcor]
dim(startupsColNum.New)
cornew <- cor(startupsColNum.New)
corrplot(cornew)

########################################
# Categorical Exploration
########################################
#recreate factors for funding round dummy variables
startupsCatVar$has_VC <- as.factor(startups$has_VC)
startupsCatVar$has_angel <- as.factor(startups$has_angel)
startupsCatVar$has_roundA <- as.factor(startups$has_roundA)
startupsCatVar$has_roundB <- as.factor(startups$has_roundB)
startupsCatVar$has_roundC <- as.factor(startups$has_roundC)
startupsCatVar$has_debtPE <- as.factor(startups$has_debtPE)

#Create bar plots of categorical data
par(mfrow = c(3,3))
plot(startupsCatVar$has_VC, main = "Has VC")
plot(startupsCatVar$has_angel, main = "Has Angel Investor")
plot(startupsCatVar$has_roundA, main = "Has Round A")
plot(startupsCatVar$has_roundB, main = "Has Round B")
plot(startupsCatVar$has_roundC, main = "Has Round C")
```

```
plot(startupsCatVar$has_debtPE, main = "Has Debt PE")

plot(startups$state_code, main = "State")

plot(startups$category_code, main = "Industry Category")

plot(startups.response, main = "Startup Status")


#Reset to numeric

startupsCatVar$has_VC <- as.numeric(startups$has_VC)

startupsCatVar$has_angel <- as.numeric(startups$has_angel)

startupsCatVar$has_roundA <- as.numeric(startups$has_roundA)

startupsCatVar$has_roundB <- as.numeric(startups$has_roundB)

startupsCatVar$has_roundC <- as.numeric(startups$has_roundC)

startupsCatVar$has_debtPE <- as.numeric(startups$has_debtPE)


#Find Near Zero or Zero Variance Predictors

nzv <- nearZeroVar(startupsCatVar)

nzv


#Predictors is_TX, is_ecommerce, and is_consulting were found to be nzv

#Remove these predictors from data set

startupsCatVar.New <- startupsCatVar[, -nzv]

head(startupsCatVar.New)


#state dummy variables

startups$state_code <- as.factor(as.character(ifelse(startups$state_code == 'CA', yes = "CA", no =
ifelse(startups$state_code == 'NY',

                                yes = "NY", no = ifelse(startups$state_code == 'MA',

                                yes = "MA", no = ifelse(startups$state_code == 'TX', yes = "other", no =
"other"))))))


#industry category dummy variables
```

```
startups$category_code <- as.factor(as.character(ifelse(startups$category_code == 'software', yes = "software",
                        no = ifelse(startups$category_code == 'web', yes = "web",
                        no = ifelse(startups$category_code == 'mobile', yes = "mobile",
                        no = ifelse(startups$category_code == 'enterprise', yes = "enterprise",
                        no = ifelse(startups$category_code == 'advertising', yes = "advertising",
                        no = ifelse(startups$category_code == 'games_video', yes = "games_video",
                        no = ifelse(startups$category_code == 'ecommerce', yes = "other",
                        no = ifelse(startups$category_code == 'consulting', yes = "other",
                        no = ifelse(startups$category_code == 'biotech', yes = "biotech",
                        no = "other"))))))))))


dmy <- dummyVars(formula = ~ state_code + category_code,
            data = startups,
            fullRank = TRUE)
dummy.vars <- data.frame(predict(dmy, newdata = startups))
head(dummy.vars, n = 10)


startupsCatVar.New <- cbind(startupsCatVar.New[,11:16], dummy.vars[,1:10])


################################
# Data Transformations
################################
#First calculate skewness values for untransformed numerical predictors
skewnessval <- apply(startupsColNum.New, 2, skewness)
skewnessval


#Heavy skewness present in each variable except time to first milestone
#Need to center and scale variables first, cannot use Box Cox method due to =< 0 values, must add constant
```

```
#Add constant to all numerical predictors equal to the absolute value of the minimum value + 1
startupsColNum.add <- startupsColNum.New + 21
summary(startupsColNum.add)

startupsColNum.Trans <- preProcess(startupsColNum.add, method = c("center",
"scale","BoxCox"))
startupsColNum.Trans

#Apply Transformations and review new skewness
Transformed <- predict(startupsColNum.Trans, startupsColNum.add)
skewtrans <- apply(Transformed, 2, skewness)
skewtrans

#Apply spatial sign transformation to lessen outlier effect
spatial.startups <- as.data.frame(spatialSign(Transformed))
skewspatial <- apply(spatial.startups, 2, skewness)
skewspatial

boxplot(spatial.startups)

#Join numerical and categorical variables to new predictor set
startups.predictors.clean <- cbind.data.frame(spatial.startups, startupsCatVar.New)
head(startups.predictors.clean)

################################
# Random Sampling
################################

#Use stratified random splits based on classes of response variable
#Set random number seed
```

```
set.seed(20)

#Create training set using 60% of data
trainingRows <- createDataPartition(as.matrix(startups.response), p = 0.6, list = FALSE)
head(trainingRows)
nrow(trainingRows)

trainPredictors <- startups.predictors.clean[trainingRows, ]
trainClasses <- startups.response[trainingRows]
length(trainClasses)
dim(trainPredictors)

traindata <- cbind(trainPredictors, trainClasses)
summary(traindata)

#Create test set of data
testPredictors <- startups.predictors.clean[-trainingRows, ]
testClasses <- startups.response[-trainingRows]

testdata <- cbind(testPredictors, testClasses)

#resampling
ctrl <- trainControl(method = "LGOCV",
                     summaryFunction = twoClassSummary,
                     classProbs = TRUE,
                     savePredictions = TRUE)

########################
#Logistic Regression
########################
```

```
set.seed(200)
lrTrain <- train(traindata[,1:21],
                 y = traindata$trainClasses,
                 method = "glm",
                 metric = "ROC",
                 trControl = ctrl)
lrTrain
head(lrTrain$pred)
length(lrTrain$pred[,1])

#confusion matrix - training data
confusionMatrix(data = lrTrain$pred$pred,
                reference = lrTrain$pred$obs,
                positive = "Success")

#ROC curve - training data
trainROC.lr <- roc(response = lrTrain$pred$obs,
                   predictor = lrTrain$pred$Success,
                   levels = rev(levels(lrTrain$pred$obs)))
par(mfrow = c(1,1))
plot(trainROC.lr, legacy.axes = TRUE, main = "Training Data ROC Curve: Logistic Regression")
auc(trainROC.lr)

#testing data
predict.lr <- predict(lrTrain, testdata[,1:21], type="prob")
testPredictors$lrprob <- predict.lr[,"Success"]
testPredictors$lr.class <- predict(lrTrain, testdata[,1:21])
testPredictors$obs <- as.factor(testClasses)

#confusion matrix - testing data
```

```
confusionMatrix(data = testPredictors$lr.class,
                reference = testPredictors$obs,
                positive = "Success")

#ROC Curve- testing data
testROC.lr <- roc(response = testPredictors$obs,
                  predictor = testPredictors$lrprob,
                  levels = rev(levels(testPredictors$obs)))
plot(testROC.lr, legacy.axes = TRUE, main = "Testing Data ROC Curve: Logistic Regression")
auc(testROC.lr)

#Most important variables
glmnImp <- varImp(lrTrain, scale = FALSE)
glmnImp
plot(glmnImp, top = 9, scales = list(y = list(cex = .95)), main = "Most Important Variables")



###################################
#Linear Discriminant Analysis
###################################
set.seed(200)
LDAtrain <- train(traindata[,1:21],
                  y = traindata$trainClasses,
                  method = "lda",
                  metric = "ROC",
                  trControl = ctrl)
LDAtrain
head(LDAtrain$pred)
length(LDAtrain$pred[,1])
```

```
#confusion matrix - training data
confusionMatrix(data = LDAtrain$pred$pred,
                reference = LDAtrain$pred$obs,
                positive = "Success")

#ROC curve - training data
trainROC.LDA <- roc(response = LDAtrain$pred$obs,
                    predictor = LDAtrain$pred$Success,
                    levels = rev(levels(LDAtrain$pred$obs)))
plot(trainROC.LDA, legacy.axes = TRUE, main = "Training Data ROC Curve: Linear
Discriminant Analysis")
auc(trainROC.LDA)

#testing data
predict.LDA <- predict(LDAtrain, testdata[,1:21], type="prob")
testPredictors$LDAprob <- predict.LDA[,"Success"]
testPredictors$LDA.class <- predict(LDAtrain, testdata[,1:21])
testPredictors$obs <- as.factor(testClasses)

#confusion matrix - testing data
confusionMatrix(data = testPredictors$LDA.class,
                reference = testPredictors$obs,
                positive = "Success")

#ROC Curve - testing data
testROC.LDA <- roc(response = testPredictors$obs,
                   predictor = testPredictors$LDAprob,
                   levels = rev(levels(testPredictors$obs)))
plot(testROC.LDA, legacy.axes = TRUE, main = "Testing Data ROC Curve: Linear Discriminant
Analysis")
auc(testROC.LDA)
```

```
#########################################################
#Partial Least Squares Linear Discriminant Analysis
#########################################################
set.seed(200)
PLSDAtrain <- train(traindata[,1:21],
                    y = traindata$trainClasses,
                 method = "pls",
                 tuneGrid = expand.grid(.ncomp = 1:10),
                 metric = "ROC",
                 trControl = ctrl)
PLSDAtrain
plot(PLSDAtrain, main = "PLSDA Tuning Parameters")
head(PLSDAtrain$pred)
length(PLSDAtrain$pred[,1])

#confusion matrix - training data
confusionMatrix(data = PLSDAtrain$pred$pred,
                reference = PLSDAtrain$pred$obs,
                positive = "Success")

#ROC curve - training data
trainROC.PLSDA <- roc(response = PLSDAtrain$pred$obs,
                      predictor = PLSDAtrain$pred$Success,
                      levels = rev(levels(PLSDAtrain$pred$obs)))
plot(trainROC.PLSDA, legacy.axes = TRUE, main = "Training Data ROC Curve: PLSDA")
auc(trainROC.PLSDA)

#testing data
predict.PLSDA <- predict(PLSDAtrain, testdata[,1:21], type="prob")
```

```
testPredictors$PLSDAprob <- predict.PLSDA[,"Success"]
testPredictors$PLSDA.class <- predict(PLSDAtrain, testdata[,1:21])
testPredictors$obs <- as.factor(testClasses)

#confusion matrix - testing data
confusionMatrix(data = testPredictors$PLSDA.class,
                reference = testPredictors$obs,
                positive = "Success")

#ROC Curve - testing data
testROC.PLSDA <- roc(response = testPredictors$obs,
                     predictor = testPredictors$PLSDAprob,
                     levels = rev(levels(testPredictors$obs)))
plot(testROC.PLSDA, legacy.axes = TRUE, main = "Testing Data ROC Curve: PLSDA")
auc(testROC.PLSDA)

########################
#Penalized Model
########################
set.seed(476)
glmnGrid <- expand.grid(.alpha = c(0, .1, .2, .4, .6, .8, 1),
                        .lambda = seq(.01, .2, length = 10))

glmnTrain <- train(x=traindata[,1:21],
                   y = traindata$trainClasses,
                   method = "glmnet",
                   tuneGrid = glmnGrid,
                   metric = "ROC",
                   trControl = ctrl)
glmnTrain
```

```
plot(glmnTrain, main = "GLMN Tuning Parameters")
head(glmnTrain$pred)
length(glmnTrain$pred[,1])

#confusion matrix - training data
confusionMatrix(data = glmnTrain$pred$pred,
                reference = glmnTrain$pred$obs,
                positive = "Success")

#ROC curve - training data
trainROC.glmn <- roc(response = glmnTrain$pred$obs,
                     predictor = glmnTrain$pred$Success,
                     levels = rev(levels(glmnTrain$pred$obs)))
plot(trainROC.glmn, legacy.axes = TRUE, main = "Training Data ROC Curve: Penalized
Model")
auc(trainROC.glmn)

#testing data
predict.glmn <- predict(glmnTrain, testdata[,1:21], type="prob")
testPredictors$glmnprob <- predict.glmn[,"Success"]
testPredictors$glmn.class <- predict(glmnTrain, testdata[,1:21])
testPredictors$obs <- as.factor(testClasses)

#confusion matrix - testing data
confusionMatrix(data = testPredictors$glmn.class,
                reference = testPredictors$obs,
                positive = "Success")

#ROC Curve - testing data
testROC.glmn <- roc(response = testPredictors$obs,
```

```
                predictor = testPredictors$glmnprob,
                levels = rev(levels(testPredictors$obs)))
plot(testROC.glmn, legacy.axes = TRUE, main = "Testing Data ROC Curve: Penalized Models")
auc(testROC.glmn)

####################################
# Nonlinear Discriminant Analysis
####################################
set.seed(200)
MDAtrain <- train(traindata[,1:21],
            y = traindata$trainClasses,
            method = "mda",
            metric = "ROC",
            tuneGrid = expand.grid(.subclasses = 1:3),
            trControl = ctrl)
MDAtrain
plot(MDAtrain, main = "MDA Tuning Parameters")
head(MDAtrain$pred)
length(MDAtrain$pred[,1])

#confusion matrix - training data
confusionMatrix(data = MDAtrain$pred$pred,
            reference = MDAtrain$pred$obs,
            positive = "Success")

#ROC curve - training data
trainROC.MDA <- roc(response = MDAtrain$pred$obs,
                predictor = MDAtrain$pred$Success,
                levels = rev(levels(MDAtrain$pred$obs)))
plot(trainROC.MDA, legacy.axes = TRUE, main = "Training Data ROC Curve: MDA")
```

```
auc(trainROC.MDA)

#testing data
predict.MDA <- predict(MDAtrain, testdata[,1:21], type="prob")
testPredictors$MDAprob <- predict.MDA[,"Success"]
testPredictors$MDA.class <- predict(MDAtrain, testdata[,1:21])
testPredictors$obs <- as.factor(testClasses)

#confusion matrix - testing data
confusionMatrix(data = testPredictors$MDA.class,
                reference = testPredictors$obs,
                positive = "Success")

#ROC Curve - testing data
testROC.MDA <- roc(response = testPredictors$obs,
                   predictor = testPredictors$MDAprob,
                   levels = rev(levels(testPredictors$obs)))
plot(testROC.MDA, legacy.axes = TRUE, main = "Testing Data ROC Curve: MDA")
auc(testROC.MDA)

##########################
# Neural Network Model
##########################
set.seed(200)
#Set training parameters
nnetGrid <- expand.grid(.size = 1:10, .decay = c(0, 0.1, 0.3, 0.5, 1))
maxSize <- max(nnetGrid$.size)
numWts <- (maxSize * (21 + 1) + (maxSize + 1)*2)

NNettrain <- train(traindata[,1:21],
```

```
                y = traindata$trainClasses,
                method = "nnet",
                metric = "ROC",
                tuneGrid = nnetGrid,
                trace = FALSE,
                maxit = 2000,
                maxNWts = numWts,
                trControl = ctrl)
NNettrain
plot(NNettrain, main = "Neural Network")
head(NNettrain$pred)
length(NNettrain$pred[,1])

#confusion matrix - training data
confusionMatrix(data = NNettrain$pred$pred,
              reference = NNettrain$pred$obs,
              positive = "Success")

#ROC curve - training data
trainROC.NNet <- roc(response = NNettrain$pred$obs,
                  predictor = NNettrain$pred$Success,
                  levels = rev(levels(NNettrain$pred$obs)))
plot(trainROC.NNet, legacy.axes = TRUE, main = "Training Data ROC Curve: Neural Net")
auc(trainROC.NNet)

#testing data
predict.NNet <- predict(NNettrain, testdata[,1:21], type="prob")
testPredictors$NNetprob <- predict.NNet[,"Success"]
testPredictors$NNet.class <- predict(NNettrain, testdata[,1:21])
testPredictors$obs <- as.factor(testClasses)
```

```
#confusion matrix - testing data
confusionMatrix(data = testPredictors$NNet.class,
                reference = testPredictors$obs,
                positive = "Success")

#ROC Curve - testing data
testROC.NNet <- roc(response = testPredictors$obs,
                    predictor = testPredictors$NNetprob,
                    levels = rev(levels(testPredictors$obs)))
plot(testROC.NNet, legacy.axes = TRUE, main = "Testing Data ROC Curve: NNet")
auc(testROC.NNet)

################################
# Flexible Discriminant Analysis
################################
set.seed(200)
#Define training grid
marsGrid <- expand.grid(.degree = 1:2, .nprune = 2:38)

FDAtrain <- train(traindata[,1:21],
                  y = traindata$trainClasses,
                  method = "fda",
                  metric = "ROC",
                  tuneGrid = marsGrid,
                  trControl = ctrl)
FDAtrain
plot(FDAtrain, main = "Flexible Discriminant Analysis")
head(FDAtrain$pred)
length(FDAtrain$pred[,1])
```

```
#confusion matrix - training data
confusionMatrix(data = FDAtrain$pred$pred,
                reference = FDAtrain$pred$obs,
                positive = "Success")

#ROC curve - training data
trainROC.FDA <- roc(response = FDAtrain$pred$obs,
                    predictor = FDAtrain$pred$Success,
                    levels = rev(levels(FDAtrain$pred$obs)))
plot(trainROC.FDA, legacy.axes = TRUE, main = "Training Data ROC Curve: FDA")
auc(trainROC.FDA)

#testing data
predict.FDA <- predict(FDAtrain, testdata[,1:21], type="prob")
testPredictors$FDAprob <- predict.FDA[,"Success"]
testPredictors$FDA.class <- predict(FDAtrain, testdata[,1:21])
testPredictors$obs <- as.factor(testClasses)

#confusion matrix - testing data
confusionMatrix(data = testPredictors$FDA.class,
                reference = testPredictors$obs,
                positive = "Success")

#ROC Curve - testing data
testROC.FDA <- roc(response = testPredictors$obs,
                   predictor = testPredictors$FDAprob,
                   levels = rev(levels(testPredictors$obs)))
plot(testROC.FDA, legacy.axes = TRUE, main = "Testing Data ROC Curve: FDA")
auc(testROC.FDA)
```

```
############################
# Support Vector Machine
############################
set.seed(200)
#Define svmgrid
sigmaRangeReduced <- sigest(as.matrix(traindata[,1:21]))
svmGrid <- expand.grid(.sigma = sigmaRangeReduced,
                       .C = 2^(seq(-4, 6)))

SVMtrain <- train(traindata[,1:21],
                  y = traindata$trainClasses,
                  method = "svmRadial",
                  metric = "ROC",
                  tuneGrid = svmGrid,
                  fit = FALSE,
                  trControl = ctrl)
SVMtrain
plot(SVMtrain,  main = "Support Vector Machine")
head(SVMtrain$pred)
length(SVMtrain$pred[,1])

#confusion matrix - training data
confusionMatrix(data = SVMtrain$pred$pred,
                reference = SVMtrain$pred$obs,
                positive = "Success")

#ROC curve - training data
trainROC.SVM <- roc(response = SVMtrain$pred$obs,
                    predictor = SVMtrain$pred$Success,
```

```
                levels = rev(levels(SVMtrain$pred$obs)))
plot(trainROC.SVM, legacy.axes = TRUE, main = "Training Data ROC Curve: SVM")
auc(trainROC.SVM)

#testing data
predict.SVM <- predict(SVMtrain, testdata[,1:21], type="prob")
testPredictors$SVMprob <- predict.SVM[,"Success"]
testPredictors$SVM.class <- predict(SVMtrain, testdata[,1:21])
testPredictors$obs <- as.factor(testClasses)

#confusion matrix - testing data
confusionMatrix(data = testPredictors$SVM.class,
              reference = testPredictors$obs,
              positive = "Success")

#ROC Curve - testing data
testROC.SVM <- roc(response = testPredictors$obs,
                 predictor = testPredictors$SVMprob,
                 levels = rev(levels(testPredictors$obs)))
plot(testROC.SVM, legacy.axes = TRUE, main = "Testing Data ROC Curve: SVM")
auc(testROC.SVM)

######################
# K-Nearest Neighbors
######################
set.seed(200)
Knntrain <- train(traindata[,1:21],
                y = traindata$trainClasses,
                method = "knn",
                metric = "ROC",
```

```
                tuneGrid = data.frame(.k = 1:400),
                trControl = ctrl)
Knntrain
plot(Knntrain, main = "K-NN")
head(Knntrain$pred)
length(Knntrain$pred[,1])

#confusion matrix - training data
confusionMatrix(data = Knntrain$pred$pred,
              reference = Knntrain$pred$obs,
              positive = "Success")

#ROC curve - training data
trainROC.Knn <- roc(response = Knntrain$pred$obs,
                  predictor = Knntrain$pred$Success,
                  levels = rev(levels(Knntrain$pred$obs)))
plot(trainROC.Knn, legacy.axes = TRUE, main = "Training Data ROC Curve: Knn")
auc(trainROC.Knn)

#testing data
predict.Knn <- predict(Knntrain, testdata[,1:21], type="prob")
testPredictors$Knnprob <- predict.Knn[,"Success"]
testPredictors$Knn.class <- predict(Knntrain, testdata[,1:21])
testPredictors$obs <- as.factor(testClasses)

#confusion matrix - testing data
confusionMatrix(data = testPredictors$Knn.class,
              reference = testPredictors$obs,
              positive = "Success")
```

```
#ROC Curve - testing data
testROC.Knn <- roc(response = testPredictors$obs,
                   predictor = testPredictors$Knnprob,
                   levels = rev(levels(testPredictors$obs)))
plot(testROC.Knn, legacy.axes = TRUE, main = "Testing Data ROC Curve: Knn")
auc(testROC.Knn)

###############
# Naive Bayes
###############
set.seed(200)
nBayestrain <- train(traindata[,1:21],
                     y = traindata$trainClasses,
                     method = "nb",
                     metric = "ROC",
                     tuneGrid = data.frame(.fL = 2, .usekernel = TRUE, .adjust = TRUE),
                     trControl = ctrl)
nBayestrain
head(nBayestrain$pred)
length(nBayestrain$pred[,1])

#confusion matrix - training data
confusionMatrix(data = nBayestrain$pred$pred,
                reference = nBayestrain$pred$obs,
                positive = "Success")

#ROC curve - training data
trainROC.nBayes <- roc(response = nBayestrain$pred$obs,
                       predictor = nBayestrain$pred$Success,
                       levels = rev(levels(nBayestrain$pred$obs)))
```

```
plot(trainROC.nBayes, legacy.axes = TRUE, main = "Training Data ROC Curve: Naive Bayes")
auc(trainROC.nBayes)

#testing data
predict.nBayes <- predict(nBayestrain, testdata[,1:15], type="prob")
testPredictors$nBayesprob <- predict.nBayes[,"Success"]
testPredictors$nBayes.class <- predict(nBayestrain, testdata[,1:15])
testPredictors$obs <- as.factor(testClasses)

#confusion matrix - testing data
confusionMatrix(data = testPredictors$nBayes.class,
                reference = testPredictors$obs,
                positive = "Success")

#ROC Curve - testing data
testROC.nBayes <- roc(response = testPredictors$obs,
                      predictor = testPredictors$nBayesprob,
                      levels = rev(levels(testPredictors$obs)))
plot(testROC.nBayes, legacy.axes = TRUE, main = "Testing Data ROC Curve: Naive Bayes")
auc(testROC.nBayes)
```