Bernardo Santos

Dr. Fotios Kokkotos

DSE1002 – Foundations of Data Science II – Fall 2021

**Final Project Report**

**Executive Summary**

Given my previous knowledge of financial markets and investment funds, I can say that the results I achieved were better than what I was expecting. It is well-known for me how hard it is to predict anything in the financial markets, as there are way too many variables in place that can influence the future. The data set I used was lengthy, allowing me to pick a few important variables from almost 200 (more information about the data set in the next section). This certainly helped me move through the models and find the right variables that explained the data the better.

There were two parts of this project: (1) the quantitative response variable part; and (2) the qualitative response variable part. I will dive deeper into these parts in the following two paragraphs.

The quantitative response variable part consisted of picking a quantitative response variable and selecting a few other variables to be used as explanatory variables. I wanted to answer the following question: which factors better explain the success of investment funds in a period of 10 years? I had the intention of understanding what takes for a fund to be successful. Because of my experience working in an investment bank, I had close contact with Machine Learning models that are used in production, so I wanted to make something similar here. I intended to create models I could use to explain the success of a fund. But what does explain it? Is it the sectors they allocate more money on? Is it the size of the fund? Is it the investment type? Is it the total amount of cash they have available? Or is it a blend of many different reasons? To answer that, I had to use the *fund\_return\_10years* as my response variable and look for explanatory variables that could reasonably be used to explain the data.

The qualitative response variable part consisted of picking a qualitative response variable and deciding whether to use the same variables I found useful in the previous part to explain the data. I wanted to answer the following question: what leads a fund to have returns over the inflation rate of a 10-years period? I was motivated by the fact that I knew how high the inflation rate increased in the last few years. The approach I had to this was like the one I had for the previous question: what am I going to use to explain the success of a fund? Are the same variables used in the first part going to be useful here as well? To answer that, I had to create a new column called *return\_over\_inflation,* which was a binary variable that had “0” (False) or “1” (True) as values. False means that the fund did not have a return over the inflation rate in the 10 years period, and True means the fund had a return over the inflation rate in the 10 years period.

**Data and Approach**

My data set is composed of 24,281 rows and 173 columns containing specific information about mutual investment funds. Each row corresponds to a mutual fund, whereas each column corresponds to a feature, such as: name of the fund, category of the fund, rating, return of the fund, etc. The data set was found on Kaggle, and I picked it because of my interest in the financial markets and my previous work experience with programming in such industry.

After carefully analyzing the data set, I understood that I would need to work on a few data engineering methods to clean the data and make it as useful as possible to the models I wanted to build. It is important to recall that the quantitative variable I picked was *fund\_return\_10years* (percentage of the return of the fund after 10 years)*,* whereas the qualitative one was ­*return\_over\_inflation­,* which is a binary classification of whether (“1” and “0” for “Yes”, and “No”, respectively) the fund returned a percentage that is higher than the inflation in the given timeframe. Both naturally correlate to the same variables, since the latter is a mutation of the former.

**Firstly**, I decided to look at the missing data. I created a data frame that contained the percent of missing data for each column and decided to remove the ones where the missing data percentage was 40% or more. That resulted in the removal of 24 columns.

**Secondly**, I removed clearly unneeded columns: *fund\_symbol, fund\_extended\_name, fund\_family, \_inception\_date, category, investment\_strategy, currency,* and *top10\_holdings*. These columns are either free text or categorical columns that will not be useful to our predictions by any means.

**Thirdly**, I removed all the columns that represented the return of the fund per quarter (all of them named *fund\_return\_<year>\_q<quarter>,* i.e.: *fund\_return\_2012\_q1) ­*because most of them were already removed from the approach presented in the last paragraph, and it would make no sense to keep a few and remove others.

**Lastly,** I performed normalization and vectorization of the data. The former means that all the numerical values were put in a scale of 0 to 1 (i.e.: a column has 3 possible values: 10, 50, and 100, the 10 will be converted to 0.1, the 50 will be converted to 0.5, and the 100 will be converted to 1.0). The latter means that the categorical variables are encoded (i.e.: a column has 3 possible values: Small, Medium, and Large, so the Small will be converted to 1, the Medium will be converted to 2, and the Large will be converted to 3).

As a starting point to the final project’s myriad of models, I began examining the correlation between the variables in the dataset. This first approach is fundamental to move forward, since the models are built over variables, all of which must be chosen through detailed analysis of their correlation to the response variable. What I was looking for was variables that were highly correlated to the response variable, but that would not lead the model to overfit. Based on the results, I realized that most of the variables that represented the returns of the funds for each year had extremely high correlations to the response variable, so I decided to not select them when building the models. After looking at the correlations table and playing around with a few models, I concluded that the following variables (22 of them) were the ones that would better suit the necessities of the models in the first part:

*return\_rating, price\_earnings\_ratio, price\_book\_ratio, price\_cashflow\_ratio, price\_sales\_ratio, asset\_cash, asset\_stocks, asset\_bonds, sector\_basic\_materials, sector\_basic\_utilities, sector\_communication\_services, sector\_technology, fund\_return\_ytd, fund\_return\_1year, category\_return\_1year, median\_market\_cap, quarters\_up, quartes\_down, size\_type, investment\_type, net\_asset\_value* and *rating.*

Once the variables were selected, the data set was divided into training and testing data with the former compromising 80% of the total, and the latter 20%.

**Detailed Findings of Quantitative Response Variable – *fund\_return\_10years***

***Null Model***

Graphical user interface, text, application

Description automatically generated Figure : Null Model of Quantitative Variable

The first step in the creation of the models based on the **quantitative** response variable was to build the **null model**. Such model is the simplest of them all and corresponds to the mean of the data found in the training set. The result of this model can be seen in the image inserted above, and we can interpret it in the following way: if we were to guess the result of the response variable, we would stick to the mean of the values for that column we found in the training set, which is 0.56. Such approach results in a sum of residuals squared of 21.86, which is high.

The second step in the creation of the models based on the quantitative response variable was to build the **multiple regression models.** Such models are used to analyze the relationship between a single response variable and several explanatory variables. The objective here is to predict the response variable based on the explanatory variables, which are weighted, based on their relative contribution to the overall prediction.

After quite a while playing around with variables and comparing results, I ended up with 2 multiple regression models, which gave me satisfactory results:

***Linear Regression Models***

Table

Description automatically generated

Figure : Multiple Regression Model 1, Training.

Graphical user interface, text, application

Description automatically generated

Figure : Multiple Regression Model 1, Predictive Analysis.

The results above correspond to the 1st model built for **multiple regression**, which used a high number of variables, and resulted in a R2 value of 0.7157, meaning that 71.57% of the data’s variability is explained by this model. A careful analysis of the p-values shows us that most of the variables used in the model have some sort of high statistical significance, except two of them: *sector\_communication\_services,* and *asset\_bonds.* Thus, they were removed from the 2nd model.

Text

Description automatically generated

Figure : Multiple Regression Model 2, Training

Graphical user interface, text, application, Word

Description automatically generated

Figure : Multiple Regression Model 2, Predictive Analysis.

The results above correspond to the 2nd model built for **multiple regression**. The variables picked correspond to the ones used in the previous model, except for the ones that had substantial high p-values (*sector\_communication\_services,* and *asset\_bonds*) and some others that their removals did not generate a substantial negative impact on the model’s accuracy (*price\_earnings\_ratio* and *fund\_return\_ytd*).

All the p-values are far below the 0.05 threshold, the R2 value and the sum of residuals squared are very close to the 1st model’s values, but since this model is less complex, it is the one preferred, due to the *parsimonious principle.*

***Regression Tree Models***

Text

Description automatically generated

Figure : Regression Tree Model 1, Training

Graphical user interface, text, application

Description automatically generated

Figure : Regression Tree Model 1, Predictive Analysis

The results above correspond to the 1st model built for **regression tree**. The variables picked are the ones I have worked with in the previous model and that generated a decent result. With the regression tree, we see a R2 value of 0.7231, meaning that the model can explain around 72.31% of the data variability present in the training data set. Although the R2 value represents a slight improvement from the linear models, we see a notorious worsening in terms of residuals, as the sum of residuals squared for the regression tree model increased to 24.12.

Text

Description automatically generated

Figure : Regression Tree Model 2, Training

Graphical user interface, text, application

Description automatically generated

Figure : Regression Tree Model 2, Predictive Analysis

The results above correspond to the 2nd model built for **regression tree**. The variables used are the ones used in the 1st model, except for a few variables that did not have notorious statistical significances. The removal of such variables led to an evident increase in the R2 value, which moved up to 78.20%, whereas the sum of residuals squared dropped to 19.01. Of course, this is not a game-changer here, but it does represent an improvement after all. Since this model is simpler and that its accuracy measurements are better, this model should be preferred over the 1st model.

***Random Forest Models***

Graphical user interface, text, application

Description automatically generated

Figure : Random Forest Model 1, Training

Graphical user interface, application, Word

Description automatically generated

Figure : Random Forest Model 1, Predictive Analysis

The results above correspond to the 1st model built for **random forest**. Still using the same variables as the previous models, this time the random forest provided outstanding results with parameters *ntree=201* and *mtry=3*. The percentage of the data variability explained by the model is high: 96.27%, putting it above any other model built so far. And to confirm our hypothesis, the mean squared of residuals is 0.0006, meaning the results found here are solid, and the model explains the data exceptionally well.

Graphical user interface, text, application

Description automatically generated

Figure : Random Forest Model 2, Training

Graphical user interface, application, Word

Description automatically generated

Figure : Random Forest Model 2, Predictive Analysis

The results above correspond to the 2nd model built for **random forest**. The same variables were used, but the parameters were changed to *ntree=201* and *mtry=5,* to generate better results. The output leads us to conclude that this change helped improve the model, since the percentage of variability explained is now 96.44% (around 0.17% better than the previous model), and the mean of squared residuals is 0.0005, (around 0.0001 less than the previous one). It is fair to say that the improvement is not exceptional, but it is better than nothing.

Graphical user interface, text, application

Description automatically generated

Figure : Random Forest Model 3, Training

Word

Description automatically generated with low confidence

Figure : Random Forest Model 3, Predictive Analysis

The results above correspond to the 3rd model built for **random forest**. The same variables were used, but the parameters were changed to *ntree=501* and *mtry=3,* to generate better results. The output tells us that the expected improvement simply did not take place, as the percentage of variability of the data decreased a bit to 96.28% (versus 96.44% from the previous one) and the mean squared error increased to 0.0006 (versus 0.0005 from the previous one). Again, the differences are not large, but they do represent a trend here.

***Neural Network Models***

***Chart

Description automatically generated***

Figure : Neural Network Model 1, Training

Chart, scatter chart

Description automatically generatedGraphical user interface, text, application, Word

Description automatically generated

Figure : Neural Network Model 1, Predictive Analysis

The results above correspond to the 1st model built for **neural network**. After playing around with variables and trying to make the model work in an acceptable time, the variables chosen were some of the ones I have been using in the previous models: *return\_rating, price\_book\_ratio, asset\_cash, asset\_stocks,* and *sector\_technology.* There were used 2 hidden layers: the first with 3 neurons and the second with 2 neurons. Any increase in the number of neurons did not result in any improvement in the accuracy of the model.

The 1st neural network model generated 79.05% of accuracy and a 5.08 as the result of the mean squared error. Satisfactory results, but still a bit far off our best result so far, which was the 2nd random forest model.

Graphical user interface, chart

Description automatically generated

Figure : Neural Network Model 2, Training

Chart, scatter chart

Description automatically generatedGraphical user interface, text, application

Description automatically generated

Figure : Neural Network Model 2, Predictive Analysis

The result above corresponds to the 2nd model built for **neural network**. The same variables were used as the 1st model, but this time 3 hidden layers were used: the first with 5 neurons, the second with 3 neurons, and the third with 2 neurons. This change resulted in a slight increase in the accuracy of the model, moving up to 80.21%, while the mean squared error stayed the same: 5.08.

**Detailed Findings of Qualitative Response Variable – *return\_over\_inflation***

For the qualitative response variable part of the project, I decided to create a new variable that is a bit connected to the quantitative variable used in the first part of the project. I decided to create a variable called *return\_over\_inflation,* which is basically a representation of the *fund\_return\_10years* variable but converted to a binary representation where the “1” (True) represents the value that is equal or above 12.37, and the “0” (False) represents the value that is below such threshold.

The reason why 12.37 was chosen as the threshold is because this is the value of the accumulated inflation rate in the period the data spans. From 2012 to 2020, the accumulated inflation in the United States of America was 12.73%, meaning that the dollar lost its value by such percentage, so U$100.00 in 2012 would represent the same as U$112.73 in 2020, ultimately representing a significant amount of loss of buying power.

Ultimately, the reason why people invest in funds is to make money, plain and simple. Given that after 10 years the inflation rate was of 12.73%, if a fund returns less than that amount after 10 years, it means that the investor lost money, even if the return is not negative. Given that logic, I understand that the best variable I could come up with for the qualitative response is to predict if the fund will make its investment beat the “silent tax”, also known as inflation.

The mutation was performed in the following manner:

Table

Description automatically generated

Figure : Mutation Performed to Create the return\_over\_inflation Variable

The mutation was performed over the data set saved after the data cleaning process done previously was done, so there was no need for normalization and vectorization. It was only necessary to divide the new data set into training and testing data. Once this was done, I moved to build the models, as follows:

***Null Model***

Graphical user interface

Description automatically generated with low confidence

Figure : Null Model of Qualitative Variable

The result of the **null model** shows us that we will have a tough job to overcome it! The null model generates 87.05% of accuracy, since the model can predict the variable to be “0” (or False) all the time. So now let’s see if we can find models that overcome this high accuracy.

***Multiple Logistic Regression Models***

Graphical user interface, text

Description automatically generated

Figure : Multiple Logistic Regression Model 1, Training

Graphical user interface, text, application

Description automatically generated

Figure : Multiple Logistic Regression Model 1, Predictive Analysis

The result above corresponds to the 1st model built for **multiple logistic regression.** I kept on working with the same variables used before, since the qualitative response variable we are using now is derived from the quantitative response variable we used before.

The 1st multiple logistic regression model yielded positive results. Firstly, no variables had a p-value larger than 0.05, meaning that all of them have statistical significance. Besides that, it managed to generate 93.96% of accuracy, overcoming the null model in our first try.

Text

Description automatically generated

Figure : Multiple Logistic Regression Model 2, Training

**Graphical user interface, text, application

Description automatically generated**

Figure : Multiple Logistic Regression Model 2, Predictive Analysis

The results above correspond to the 2nd model built for **multiple logistic regression**. The only difference from the previous model was the removal of the *asset\_cash* variable, which resulted in a slight increase in the accuracy of the model, and of course, made it simpler.

Just like the previous model, no variables had p-values higher than 0.05, meaning that all of them present statistical significance towards the response variable. With a higher accuracy and less variables, this 2nd model is preferred over the 1st.

***Classification Tree Models***

***Text

Description automatically generated***

Figure : Classification Tree Model 1, Training

Graphical user interface, text, application

Description automatically generated

Figure : Classification Tree Model 1, Predictive Analysis

The results above correspond to the 1st model built for **classification tree**. The variables used were the ones used previously, as they have yielded excellent results and appear to explain the data well.

The 1st model resulted in 94.89% of accuracy, overcoming the best model we had so far, which was the 2nd Multiple Logistic Regression model.

Text

Description automatically generated

Figure : Classification Tree Model 2, Training

Graphical user interface, text, application

Description automatically generated

Figure : Classification Tree Model 2, Predictive Analysis

The results above correspond to the 2nd model built for **classification tree**. By removing 1 variable, *size\_type\_encoded,* I managed to generate the same accuracy as the 1st model, which leads us to prefer this 2nd model, as it is simpler. Any other variable removal results in a drop in accuracy of the model.

With a 94.89% of accuracy, this is the best model we have built so far.

***Random Forest Models***

***Graphical user interface, text, application, email

Description automatically generated***

Figure : Random Forest Model 1, Training

Background pattern

Description automatically generated with medium confidence

Figure : Random Forest Model 1, Predictive Analysis

The results above correspond to the 1st model built for **random forest**. As usual, the same variables were used, and the parameters were: *ntree=200,* and *mtry=3.* The model ended up generating 97.97% of accuracy, which is the highest found so far.

Although I am satisfied with this result, we ought to at least attempt to increase the accuracy. Let’s change the parameters values to *ntree=200,* and *mtry=5.*

Graphical user interface, text, application, email

Description automatically generated

Figure : Random Forest Model 2, Training

A picture containing graphical user interface

Description automatically generated

Figure : Random Forest Model 2, Predictive Analysis

The results above correspond to the 2nd model built for **random forest**. The same variables used in the 1st model were used in this one, but the parameters were slightly different: *ntree* was still 200, but *mtry* changed to 5. This change did not result in an increase in accuracy, meaning that the value of 3 for *mtry* is optimal.

Now let’s see how the model behaves if we opt to increase the *ntree* parameter.

Graphical user interface, text, application, email

Description automatically generated

Figure : Random Forest Model 3, Training

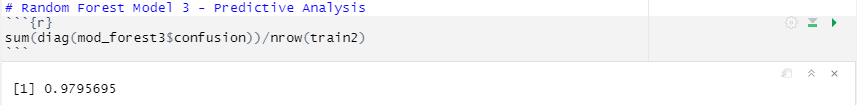


Figure : Random Forest Model 3, Predictive Analysis

The results above correspond to the 3rd model built for **random forest**. The same variables used in the previous models were used, but the *ntree* parameter was altered to be equal to 300. Such change did not result in an increase in accuracy, as it stalled in 97.95%, very similar to the one we found in the 1st model.

Now we can conclude that the parameters used in the 1st model are the optimal ones, since changes to them did not lead to better results. The 1st random forest model is our best one so far.

***KNN Models***

Graphical user interface, text

Description automatically generated

Figure : KNN Model 1, Training and Predictive Analysis

The results above correspond to the 1st model built for **K-Nearest Neighbors** **(KNN).** The same variables used in the previous models were used now, with the ***k*** parameter being set to 2.

With that configuration, the model generated 95.69% of accuracy, which is great, but still a bit below the best model so far, which was the 1st random forest model.

To look for better results, we can see how the model behaves when we change the ***k*** parameter:

Graphical user interface, text, application

Description automatically generated

Figure : KNN Model 2, Training and Predictive Analysis

The results above correspond to the 2nd model built for **KNN.** With the ***k*** parameter now equal to 4, we see that the accuracy of the model has dropped a tiny bit: now it is 95.55%. Of course, this is not a big deal, but with the same variables being used, I would rather keep the 1st model, which yielded 95.69% of accuracy.

***Naïve Bayes Models***

***Text

Description automatically generated***

Figure : Naive Bayes Model 1, Training and Predictive Analysis

The results above correspond to the 1st model built for **naïve bayes**. The same variables we have been using in the previous models were used in this one. The 1st model generated decent results, with the accuracy being 85.22%. Although this would be a good result in most cases, the null model we have is 87%, so this 1st model does not work for us.

However, when we remove the *asset\_cash* and *asset\_stocks* variables, we have a positive surprise:

Text

Description automatically generated

Figure : Naive Bayes Model 2, Training and Predictive Analysis

The results above correspond to the 2nd model built for **naïve bayes**. Now without the *asset\_cash* and *asset\_stocks* variables, the model generated a better result: 91.03% of accuracy, which beats our null model by 4%. This is a decent result for the model, but it is still far off the 1st random forest model, which has 97.97% of accuracy.

**Conclusions**

**Quantitative Variable Models Summary**

|  |  |  |
| --- | --- | --- |
| **Model** | **MSE** | **Accuracy** |
| Random Forest Model | 0.0005 | 96.44% |
| Neural Network Model | 5.08 | 79.05% |
| Linear Regression Model | 6.12 | 71.57% |
| Regression Tree Model | 19.01 | 78.20% |
| Null Model | 21.86 | 56.18% |

**Qualitative Variable Models Summary**

|  |  |
| --- | --- |
| **Model** | **Accuracy** |
| Random Forest Model | 97.97% |
| K-Nearest Neighbors Model | 95.69% |
| Classification Tree Model | 94.89% |
| Multiple Logistic Regression Model | 94.10% |
| Naïve Bayes Model | 91.03% |
| Null Model | 87.05% |

With the results shown above, we can conclude that the Random Forest models outperformed all the other models in both parts: when predicting the quantitative response variable *fund\_return\_10years,* and when classifying the *return\_over\_inflation* variable.

The Random Forest model showed outstanding results when predicting the quantitative variable, with a difference of 17.39% in accuracy to the 2nd place, which was the Neural Network (79.05% of accuracy). Its Mean Squared Error (MSE) is incredibly low, around 0.0005, placing it far away from any other model.

The Random Forest model showed good results when predicting the qualitative variable, with a difference of 1.28% to the 2nd place, KNN (95.69% of accuracy).

I could reach two different conclusions here: (1) the Random Forest is by far the best model to fit and explain the data I have; and (2) the best predictors to the return of a fund after 10 years are *return\_rating, net\_asset\_value, size\_type, asset\_cash, asset\_stocks, price\_book\_ratio, sector\_utilities, sector\_technology,* and *sector\_basic\_materials.* Here is a breakdown of each predictor:

* *return\_rating:* the rating of the return we are trying to predict from 1 to 5, with 1 being the lowest and 5 being the highest.
* *net\_asset\_value:* it is the company’s total assets minus its total liabilities.
* *size\_type:* the size of the investment fund (small, medium, large).
* *asset\_cash:* percentage of the fund’s assets in cash.
* *asset\_stocks:* percentage of the fund’s assets in stocks.
* *price\_book\_ratio:* it is the market price per share of the company divided by the result of total assets – total liabilities, so: PB Ratio = (market price per share) / (total assets – total liabilities).
* *sector\_utilities:* how much of the fund’s investment is allocated in utilities companies (in percentage).
* *sector\_technology:* how much of the fund’s investment is allocated in technology companies (in percentage).
* *sector\_basic\_materials:* how much of the fund’s investment is allocated in basic materials companies (in percentage).

All in all, I must say I am happy with the results I have obtained. What I intended to achieve was to find a model (or a group of models) that I could use to predict a fund’s return after 10 years based on a few variables. I thought I could use this kind of model at my work, which is an investment bank where we have data about thousands of different investment funds, and we would benefit a lot from correct predictions of these funds’ success (or lack thereof).

**Limitations**

The data set consisted of data from 2012 to 2020, but I think it would be even better if the data set contained data of 2021, up to the 3rd quarter.

Initially, the data set had 24,281 rows and 173 columns, but due to missing data and unformatted data, we ended up having to deal with only 6,852 rows and 28 columns. I believe that the models would benefit from having at least 20,000 rows of data and a few extra variables to choose from. Even though we had great results with Random Forests, Classification Trees, and KNN, some other models faced trouble in explaining the data, and one factor that may have contributed to it is the number of rows we ended up cutting off.

**Appendix: Knitted Code Below**

# Load data set

library(dplyr)

##   
## Attaching package: 'dplyr'

## The following objects are masked from 'package:stats':  
##   
## filter, lag

## The following objects are masked from 'package:base':  
##   
## intersect, setdiff, setequal, union

library(magrittr)  
library(mosaic)

## Registered S3 method overwritten by 'mosaic':  
## method from   
## fortify.SpatialPolygonsDataFrame ggplot2

##   
## The 'mosaic' package masks several functions from core packages in order to add   
## additional features. The original behavior of these functions should not be affected by this.

##   
## Attaching package: 'mosaic'

## The following object is masked from 'package:Matrix':  
##   
## mean

## The following object is masked from 'package:ggplot2':  
##   
## stat

## The following objects are masked from 'package:dplyr':  
##   
## count, do, tally

## The following objects are masked from 'package:stats':  
##   
## binom.test, cor, cor.test, cov, fivenum, IQR, median, prop.test,  
## quantile, sd, t.test, var

## The following objects are masked from 'package:base':  
##   
## max, mean, min, prod, range, sample, sum

library(rpart)  
library(partykit)

## Loading required package: grid

## Loading required package: libcoin

## Loading required package: mvtnorm

library(randomForest)

## randomForest 4.6-14

## Type rfNews() to see new features/changes/bug fixes.

##   
## Attaching package: 'randomForest'

## The following object is masked from 'package:ggplot2':  
##   
## margin

## The following object is masked from 'package:dplyr':  
##   
## combine

library(neuralnet)

##   
## Attaching package: 'neuralnet'

## The following object is masked from 'package:dplyr':  
##   
## compute

library(class)  
library(caret)

##   
## Attaching package: 'caret'

## The following object is masked from 'package:mosaic':  
##   
## dotPlot

library(mdsr)  
library(e1071)  
  
options(scipen = 999)  
  
Funds <- read.csv(file = 'Funds.csv', header = TRUE)

# Check the missing values in the data set

na\_count <- sapply(Funds, function(y) round(sum(length(which(is.na(y))))/nrow(Funds), digits = 2))  
na\_count <- data.frame(na\_count)  
na\_count <- cbind(colName = rownames(na\_count), na\_count)  
rownames(na\_count) <- 1:nrow(na\_count)  
colnames(na\_count) <- c("colName", "rep\_count")  
na\_count[order(-na\_count[,2]),]

## colName rep\_count  
## 39 credit\_us\_government 0.60  
## 40 credit\_aaa 0.60  
## 41 credit\_aa 0.60  
## 42 credit\_a 0.60  
## 43 credit\_bbb 0.60  
## 44 credit\_bb 0.60  
## 45 credit\_b 0.60  
## 46 credit\_below\_b 0.60  
## 47 credit\_other\_ratings 0.60  
## 83 fund\_return\_2010 0.50  
## 126 fund\_return\_2010\_q4 0.50  
## 127 fund\_return\_2010\_q3 0.49  
## 128 fund\_return\_2010\_q2 0.48  
## 84 category\_return\_2010 0.47  
## 129 fund\_return\_2010\_q1 0.47  
## 81 fund\_return\_2011 0.46  
## 122 fund\_return\_2011\_q4 0.46  
## 123 fund\_return\_2011\_q3 0.45  
## 124 fund\_return\_2011\_q2 0.44  
## 82 category\_return\_2011 0.43  
## 125 fund\_return\_2011\_q1 0.43  
## 79 fund\_return\_2012 0.41  
## 118 fund\_return\_2012\_q4 0.41  
## 119 fund\_return\_2012\_q3 0.40  
## 37 bond\_maturity 0.39  
## 120 fund\_return\_2012\_q2 0.39  
## 80 category\_return\_2012 0.38  
## 121 fund\_return\_2012\_q1 0.37  
## 38 bond\_duration 0.35  
## 77 fund\_return\_2013 0.35  
## 114 fund\_return\_2013\_q4 0.35  
## 115 fund\_return\_2013\_q3 0.34  
## 116 fund\_return\_2013\_q2 0.33  
## 78 category\_return\_2013 0.32  
## 117 fund\_return\_2013\_q1 0.32  
## 63 fund\_return\_10years 0.30  
## 75 fund\_return\_2014 0.30  
## 110 fund\_return\_2014\_q4 0.30  
## 136 fund\_alpha\_10years 0.30  
## 142 fund\_beta\_10years 0.30  
## 148 fund\_mean\_annual\_return\_10years 0.30  
## 154 fund\_r\_squared\_10years 0.30  
## 160 fund\_standard\_deviation\_10years 0.30  
## 166 fund\_sharpe\_ratio\_10years 0.30  
## 111 fund\_return\_2014\_q3 0.29  
## 21 price\_earnings\_ratio 0.28  
## 112 fund\_return\_2014\_q2 0.28  
## 22 price\_book\_ratio 0.26  
## 24 price\_cashflow\_ratio 0.26  
## 76 category\_return\_2014 0.26  
## 113 fund\_return\_2014\_q1 0.26  
## 73 fund\_return\_2015 0.24  
## 106 fund\_return\_2015\_q4 0.24  
## 23 price\_sales\_ratio 0.23  
## 26 sector\_basic\_materials 0.23  
## 27 sector\_consumer\_cyclical 0.23  
## 28 sector\_financial\_services 0.23  
## 29 sector\_real\_estate 0.23  
## 30 sector\_consumer\_defensive 0.23  
## 31 sector\_healthcare 0.23  
## 32 sector\_utilities 0.23  
## 33 sector\_communication\_services 0.23  
## 34 sector\_energy 0.23  
## 35 sector\_industrials 0.23  
## 36 sector\_technology 0.23  
## 107 fund\_return\_2015\_q3 0.23  
## 108 fund\_return\_2015\_q2 0.21  
## 74 category\_return\_2015 0.20  
## 109 fund\_return\_2015\_q1 0.19  
## 71 fund\_return\_2016 0.17  
## 102 fund\_return\_2016\_q4 0.17  
## 103 fund\_return\_2016\_q3 0.16  
## 72 category\_return\_2016 0.15  
## 104 fund\_return\_2016\_q2 0.15  
## 105 fund\_return\_2016\_q1 0.13  
## 69 fund\_return\_2017 0.12  
## 98 fund\_return\_2017\_q4 0.12  
## 61 fund\_return\_5years 0.10  
## 86 years\_down 0.10  
## 99 fund\_return\_2017\_q3 0.10  
## 134 fund\_alpha\_5years 0.10  
## 140 fund\_beta\_5years 0.10  
## 146 fund\_mean\_annual\_return\_5years 0.10  
## 152 fund\_r\_squared\_5years 0.10  
## 158 fund\_standard\_deviation\_5years 0.10  
## 164 fund\_sharpe\_ratio\_5years 0.10  
## 170 fund\_treynor\_ratio\_5years 0.10  
## 70 category\_return\_2017 0.08  
## 87 fund\_return\_2020\_q3 0.08  
## 100 fund\_return\_2017\_q2 0.08  
## 101 fund\_return\_2017\_q1 0.07  
## 67 fund\_return\_2018 0.05  
## 88 fund\_return\_2020\_q2 0.05  
## 94 fund\_return\_2018\_q4 0.05  
## 6 rating 0.04  
## 7 return\_rating 0.04  
## 8 risk\_rating 0.04  
## 52 category\_return\_ytd 0.04  
## 59 fund\_return\_3years 0.04  
## 89 fund\_return\_2020\_q1 0.04  
## 95 fund\_return\_2018\_q3 0.04  
## 132 fund\_alpha\_3years 0.04  
## 138 fund\_beta\_3years 0.04  
## 144 fund\_mean\_annual\_return\_3years 0.04  
## 150 fund\_r\_squared\_3years 0.04  
## 156 fund\_standard\_deviation\_3years 0.04  
## 162 fund\_sharpe\_ratio\_3years 0.04  
## 65 fund\_return\_2019 0.03  
## 90 fund\_return\_2019\_q4 0.03  
## 96 fund\_return\_2018\_q2 0.03  
## 64 category\_return\_10years 0.02  
## 68 category\_return\_2018 0.02  
## 97 fund\_return\_2018\_q1 0.02  
## 53 fund\_return\_1month 0.01  
## 54 category\_return\_1month 0.01  
## 55 fund\_return\_3months 0.01  
## 56 category\_return\_3months 0.01  
## 58 category\_return\_1year 0.01  
## 60 category\_return\_3years 0.01  
## 62 category\_return\_5years 0.01  
## 66 category\_return\_2019 0.01  
## 91 fund\_return\_2019\_q3 0.01  
## 1 fund\_symbol 0.00  
## 2 fund\_extended\_name 0.00  
## 3 fund\_family 0.00  
## 4 inception\_date 0.00  
## 5 category 0.00  
## 9 investment\_strategy 0.00  
## 10 investment\_type 0.00  
## 11 size\_type 0.00  
## 12 currency 0.00  
## 13 fund\_net\_annual\_expense\_ratio 0.00  
## 14 category\_net\_annual\_expense\_ratio 0.00  
## 15 asset\_cash 0.00  
## 16 asset\_stocks 0.00  
## 17 asset\_bonds 0.00  
## 18 asset\_others 0.00  
## 19 asset\_preferred 0.00  
## 20 asset\_convertable 0.00  
## 25 median\_market\_cap 0.00  
## 48 net\_asset\_value 0.00  
## 49 fund\_yield 0.00  
## 50 top10\_holdings 0.00  
## 51 fund\_return\_ytd 0.00  
## 57 fund\_return\_1year 0.00  
## 85 years\_up 0.00  
## 92 fund\_return\_2019\_q2 0.00  
## 93 fund\_return\_2019\_q1 0.00  
## 130 quarters\_up 0.00  
## 131 quarters\_down 0.00  
## 133 category\_alpha\_3years 0.00  
## 135 category\_alpha\_5years 0.00  
## 137 category\_alpha\_10years 0.00  
## 139 category\_beta\_3years 0.00  
## 141 category\_beta\_5years 0.00  
## 143 category\_beta\_10years 0.00  
## 145 category\_mean\_annual\_return\_3years 0.00  
## 147 category\_mean\_annual\_return\_5years 0.00  
## 149 category\_mean\_annual\_return\_10years 0.00  
## 151 category\_r\_squared\_3years 0.00  
## 153 category\_r\_squared\_5years 0.00  
## 155 category\_r\_squared\_10years 0.00  
## 157 category\_standard\_deviation\_3years 0.00  
## 159 category\_standard\_deviation\_5years 0.00  
## 161 category\_standard\_deviation\_10years 0.00  
## 163 category\_sharpe\_ratio\_3years 0.00  
## 165 category\_sharpe\_ratio\_5years 0.00  
## 167 category\_sharpe\_ratio\_10years 0.00  
## 168 fund\_treynor\_ratio\_3years 0.00  
## 169 category\_treynor\_ratio\_3years 0.00  
## 171 category\_treynor\_ratio\_5years 0.00  
## 172 fund\_treynor\_ratio\_10years 0.00  
## 173 category\_treynor\_ratio\_10years 0.00

# Get column names where missing values percentage is greater than 40%

na\_count\_40 <- filter(na\_count, rep\_count >= 0.4)

# Remove the clearly unneeded columns, such as free text and names that do not influence the dataset overall

# Remove quarterly-based returns variables, since we cannot use all of them

cols\_remove <- c("fund\_symbol", "fund\_extended\_name", "fund\_family", "inception\_date", "category", "investment\_strategy", "currency", "top10\_holdings")  
cols\_remove\_join <- c(cols\_remove, na\_count\_40$colName)  
q\_returns <- c()  
for(x in 2012:2020){  
 for(y in 1:4){  
 if(x == 2020 && y == 4){  
   
 } else{  
 name <- paste("fund\_return\_",as.character(x), "\_q",as.character(y), sep="")  
 if(name %in% cols\_remove\_join == FALSE){  
 q\_returns[length(q\_returns) + 1] <- name  
 }  
   
 }  
   
 }  
}  
cols\_remove\_join <- c(cols\_remove\_join, q\_returns)  
new\_funds <- Funds %>%   
 dplyr::select(-cols\_remove\_join)

## Note: Using an external vector in selections is ambiguous.  
## i Use `all\_of(cols\_remove\_join)` instead of `cols\_remove\_join` to silence this message.  
## i See <https://tidyselect.r-lib.org/reference/faq-external-vector.html>.  
## This message is displayed once per session.

# Define target variables

# Main targets as numerical variables: fund\_return\_3months, fund\_return\_1year, fund\_return\_5years, fund\_return\_10years.  
# Main target as categorical variables: return\_rating.

# Perform vectorization

encode\_ordinal <- function(x, order = unique(x)) {  
 x <- as.numeric(factor(x, levels = order, exclude = NULL))  
 x  
}  
  
new\_funds[["investment\_type\_encoded"]] <- encode\_ordinal(new\_funds[["investment\_type"]])  
new\_funds[["size\_type\_encoded"]] <- encode\_ordinal(new\_funds[["size\_type"]])  
  
new\_funds <- new\_funds %>%   
 dplyr::select(-c("investment\_type", "size\_type"))

# Performs normalization

# Remove rows that contain missing data.  
new\_funds <- new\_funds[complete.cases(new\_funds),]  
  
# Convert columns to numeric  
cols <- colnames(new\_funds)  
new\_funds[cols] <- sapply(new\_funds[cols],as.numeric)

## Warning in lapply(X = X, FUN = FUN, ...): NAs introduced by coercion

sapply(new\_funds, class)

## rating return\_rating   
## "numeric" "numeric"   
## risk\_rating fund\_net\_annual\_expense\_ratio   
## "numeric" "numeric"   
## category\_net\_annual\_expense\_ratio asset\_cash   
## "numeric" "numeric"   
## asset\_stocks asset\_bonds   
## "numeric" "numeric"   
## asset\_others asset\_preferred   
## "numeric" "numeric"   
## asset\_convertable price\_earnings\_ratio   
## "numeric" "numeric"   
## price\_book\_ratio price\_sales\_ratio   
## "numeric" "numeric"   
## price\_cashflow\_ratio median\_market\_cap   
## "numeric" "numeric"   
## sector\_basic\_materials sector\_consumer\_cyclical   
## "numeric" "numeric"   
## sector\_financial\_services sector\_real\_estate   
## "numeric" "numeric"   
## sector\_consumer\_defensive sector\_healthcare   
## "numeric" "numeric"   
## sector\_utilities sector\_communication\_services   
## "numeric" "numeric"   
## sector\_energy sector\_industrials   
## "numeric" "numeric"   
## sector\_technology bond\_maturity   
## "numeric" "numeric"   
## bond\_duration net\_asset\_value   
## "numeric" "numeric"   
## fund\_yield fund\_return\_ytd   
## "numeric" "numeric"   
## category\_return\_ytd fund\_return\_1month   
## "numeric" "numeric"   
## category\_return\_1month fund\_return\_3months   
## "numeric" "numeric"   
## category\_return\_3months fund\_return\_1year   
## "numeric" "numeric"   
## category\_return\_1year fund\_return\_3years   
## "numeric" "numeric"   
## category\_return\_3years fund\_return\_5years   
## "numeric" "numeric"   
## category\_return\_5years fund\_return\_10years   
## "numeric" "numeric"   
## category\_return\_10years fund\_return\_2019   
## "numeric" "numeric"   
## category\_return\_2019 fund\_return\_2018   
## "numeric" "numeric"   
## category\_return\_2018 fund\_return\_2017   
## "numeric" "numeric"   
## category\_return\_2017 fund\_return\_2016   
## "numeric" "numeric"   
## category\_return\_2016 fund\_return\_2015   
## "numeric" "numeric"   
## category\_return\_2015 fund\_return\_2014   
## "numeric" "numeric"   
## category\_return\_2014 fund\_return\_2013   
## "numeric" "numeric"   
## category\_return\_2013 category\_return\_2012   
## "numeric" "numeric"   
## years\_up years\_down   
## "numeric" "numeric"   
## quarters\_up quarters\_down   
## "numeric" "numeric"   
## fund\_alpha\_3years category\_alpha\_3years   
## "numeric" "numeric"   
## fund\_alpha\_5years category\_alpha\_5years   
## "numeric" "numeric"   
## fund\_alpha\_10years category\_alpha\_10years   
## "numeric" "numeric"   
## fund\_beta\_3years category\_beta\_3years   
## "numeric" "numeric"   
## fund\_beta\_5years category\_beta\_5years   
## "numeric" "numeric"   
## fund\_beta\_10years category\_beta\_10years   
## "numeric" "numeric"   
## fund\_mean\_annual\_return\_3years category\_mean\_annual\_return\_3years   
## "numeric" "numeric"   
## fund\_mean\_annual\_return\_5years category\_mean\_annual\_return\_5years   
## "numeric" "numeric"   
## fund\_mean\_annual\_return\_10years category\_mean\_annual\_return\_10years   
## "numeric" "numeric"   
## fund\_r\_squared\_3years category\_r\_squared\_3years   
## "numeric" "numeric"   
## fund\_r\_squared\_5years category\_r\_squared\_5years   
## "numeric" "numeric"   
## fund\_r\_squared\_10years category\_r\_squared\_10years   
## "numeric" "numeric"   
## fund\_standard\_deviation\_3years category\_standard\_deviation\_3years   
## "numeric" "numeric"   
## fund\_standard\_deviation\_5years category\_standard\_deviation\_5years   
## "numeric" "numeric"   
## fund\_standard\_deviation\_10years category\_standard\_deviation\_10years   
## "numeric" "numeric"   
## fund\_sharpe\_ratio\_3years category\_sharpe\_ratio\_3years   
## "numeric" "numeric"   
## fund\_sharpe\_ratio\_5years category\_sharpe\_ratio\_5years   
## "numeric" "numeric"   
## fund\_sharpe\_ratio\_10years category\_sharpe\_ratio\_10years   
## "numeric" "numeric"   
## fund\_treynor\_ratio\_3years category\_treynor\_ratio\_3years   
## "numeric" "numeric"   
## fund\_treynor\_ratio\_5years category\_treynor\_ratio\_5years   
## "numeric" "numeric"   
## fund\_treynor\_ratio\_10years category\_treynor\_ratio\_10years   
## "numeric" "numeric"   
## investment\_type\_encoded size\_type\_encoded   
## "numeric" "numeric"

# Max and mins for normalization  
maxs <- apply(new\_funds[1:102], 2, max)   
mins <- apply(new\_funds[1:102], 2, min)  
  
# Perform scaling  
funds\_scaled <- as.data.frame(scale(new\_funds[1:102], center = mins, scale = maxs - mins))  
funds\_scaled$investment\_type\_encoded <- new\_funds[["investment\_type\_encoded"]]  
funds\_scaled$size\_type\_encoded <- new\_funds[["size\_type\_encoded"]]

res <- as.data.frame(cor(funds\_scaled))  
res <- as.data.frame(res["fund\_return\_10years"])  
res <- res %>%   
 filter(abs(res$fund\_return\_10years) > .3) %>%   
 arrange(desc(fund\_return\_10years))

# Given the results from the correlation table from the last chunk,

funds\_scaled <- funds\_scaled %>%   
 dplyr::select(rating, return\_rating, investment\_type\_encoded, size\_type\_encoded, price\_earnings\_ratio, price\_book\_ratio, price\_sales\_ratio, asset\_cash, asset\_stocks, asset\_bonds, sector\_basic\_materials, sector\_utilities, sector\_communication\_services, sector\_technology, fund\_return\_ytd, fund\_return\_1year, category\_return\_1year, fund\_return\_3years, category\_return\_3years, fund\_return\_5years, fund\_return\_10years, category\_return\_5years, fund\_beta\_3years, category\_beta\_3years, fund\_beta\_5years, category\_beta\_5years, fund\_mean\_annual\_return\_3years, fund\_mean\_annual\_return\_3years, net\_asset\_value)

# Divide the data into training and testing

set.seed(200)  
  
n <- nrow(funds\_scaled)  
test\_idx <- sample.int(n, size = round(0.2 \* n))  
train <- funds\_scaled[-test\_idx,]  
test <- funds\_scaled[test\_idx,]  
nrow(funds\_scaled)

## [1] 6852

# Numerical response variable -> fund\_return\_10years

# The following chunks of code are related to the creation of models that are going to predict the numerical response variable 'fund\_return\_10years'.

# Null Model

null\_model\_1 <- mean(train$fund\_return\_10years, na.rm=TRUE)  
results\_null\_1 <- data.frame(pred = null\_model\_1, original = test$fund\_return\_10years)  
results\_null\_1 <- results\_null\_1 %>%   
 mutate(resid = round(pred - original, 2))  
cat("Sum of residuals squared of the null model: ", sum(results\_null\_1$resid^2))

## Sum of residuals squared of the null model: 21.8688

cat("\nMean of the null model: ", null\_model\_1)

##   
## Mean of the null model: 0.5618706

# Linear Regression Models - Training

# Linear Model 1 - Training

lm1\_train <- lm(fund\_return\_10years ~ return\_rating + price\_earnings\_ratio + price\_book\_ratio + asset\_cash + asset\_stocks + asset\_bonds + sector\_basic\_materials + sector\_utilities + sector\_communication\_services + sector\_technology + fund\_return\_ytd + category\_return\_1year, data = train)  
  
msummary(lm1\_train)

## Estimate Std. Error t value Pr(>|t|)  
## (Intercept) 0.277059 0.017414 15.910 < 0.0000000000000002  
## return\_rating 0.125362 0.004443 28.218 < 0.0000000000000002  
## price\_earnings\_ratio -0.029623 0.015275 -1.939 0.05251  
## price\_book\_ratio 0.174924 0.016770 10.431 < 0.0000000000000002  
## asset\_cash -0.071266 0.012899 -5.525 0.0000000345  
## asset\_stocks 0.123055 0.014420 8.534 < 0.0000000000000002  
## asset\_bonds -0.035036 0.015623 -2.243 0.02497  
## sector\_basic\_materials -0.463877 0.018569 -24.981 < 0.0000000000000002  
## sector\_utilities 0.073796 0.013795 5.350 0.0000000917  
## sector\_communication\_services -0.034333 0.018944 -1.812 0.06999  
## sector\_technology 0.065521 0.014066 4.658 0.0000032685  
## fund\_return\_ytd 0.079342 0.027993 2.834 0.00461  
## category\_return\_1year 0.185893 0.007982 23.290 < 0.0000000000000002  
##   
## (Intercept) \*\*\*  
## return\_rating \*\*\*  
## price\_earnings\_ratio .   
## price\_book\_ratio \*\*\*  
## asset\_cash \*\*\*  
## asset\_stocks \*\*\*  
## asset\_bonds \*   
## sector\_basic\_materials \*\*\*  
## sector\_utilities \*\*\*  
## sector\_communication\_services .   
## sector\_technology \*\*\*  
## fund\_return\_ytd \*\*   
## category\_return\_1year \*\*\*  
##   
## Residual standard error: 0.06717 on 5469 degrees of freedom  
## Multiple R-squared: 0.7163, Adjusted R-squared: 0.7157   
## F-statistic: 1151 on 12 and 5469 DF, p-value: < 0.00000000000000022

# Linear Model 2 - Training

lm2\_train <- lm(fund\_return\_10years ~ return\_rating + price\_book\_ratio + asset\_cash + asset\_stocks + sector\_basic\_materials + sector\_utilities + sector\_technology + category\_return\_1year, data = train)  
  
msummary(lm2\_train)

## Estimate Std. Error t value Pr(>|t|)   
## (Intercept) 0.263346 0.005454 48.288 < 0.0000000000000002 \*\*\*  
## return\_rating 0.129160 0.003818 33.833 < 0.0000000000000002 \*\*\*  
## price\_book\_ratio 0.172638 0.012227 14.120 < 0.0000000000000002 \*\*\*  
## asset\_cash -0.048544 0.008663 -5.604 0.000000022008 \*\*\*  
## asset\_stocks 0.149961 0.004250 35.286 < 0.0000000000000002 \*\*\*  
## sector\_basic\_materials -0.456677 0.018390 -24.834 < 0.0000000000000002 \*\*\*  
## sector\_utilities 0.082628 0.013353 6.188 0.000000000654 \*\*\*  
## sector\_technology 0.072893 0.013722 5.312 0.000000112659 \*\*\*  
## category\_return\_1year 0.193477 0.006656 29.067 < 0.0000000000000002 \*\*\*  
##   
## Residual standard error: 0.06725 on 5473 degrees of freedom  
## Multiple R-squared: 0.7154, Adjusted R-squared: 0.715   
## F-statistic: 1720 on 8 and 5473 DF, p-value: < 0.00000000000000022

# Predictive Analysis for Linear Model 1

pred\_lm1 <- predict(lm1\_train, newdata=test)  
results\_lm1 <- data.frame(pred = pred\_lm1, original = test$fund\_return\_10years)  
results\_lm1 <- results\_lm1 %>%   
 mutate(resid = round(pred - original, 2))  
cat("Sum of residuals squared for Linear Model 1:", sum(results\_lm1$resid^2))

## Sum of residuals squared for Linear Model 1: 6.1299

# Predictive Analysis for Linear Model 2

pred\_lm2 <- predict(lm2\_train, newdata=test)  
results\_lm2 <- data.frame(pred = pred\_lm2, original = test$fund\_return\_10years)  
results\_lm2 <- results\_lm2 %>%   
 mutate(resid = round(pred - original, 2))  
cat("Sum of residuals squared for Linear Model 2:", sum(results\_lm2$resid^2))

## Sum of residuals squared for Linear Model 2: 6.1711

# Regression Tree Models - Training

# Regression Tree Model 1 - Training

reg\_form\_1 <- as.formula("fund\_return\_10years ~ return\_rating + price\_book\_ratio + asset\_cash + asset\_stocks + sector\_basic\_materials + sector\_utilities + sector\_technology + category\_return\_1year")  
regtree\_1 <- rpart(reg\_form\_1, data=train)  
regtree\_1

## n= 5482   
##   
## node), split, n, deviance, yval  
## \* denotes terminal node  
##   
## 1) root 5482 86.9735500 0.5618706   
## 2) category\_return\_1year< 0.5455275 3989 32.1980500 0.5119676   
## 4) return\_rating< 0.625 2732 18.8272100 0.4891635   
## 8) sector\_basic\_materials>=0.107423 373 3.7332770 0.4148752 \*  
## 9) sector\_basic\_materials< 0.107423 2359 12.7099500 0.5009098   
## 18) asset\_stocks< 0.4181 471 1.1979120 0.4453781 \*  
## 19) asset\_stocks>=0.4181 1888 9.6972400 0.5147633   
## 38) price\_book\_ratio< 0.06241744 122 0.7004108 0.4281776 \*  
## 39) price\_book\_ratio>=0.06241744 1766 8.0189980 0.5207449   
## 78) return\_rating< 0.125 133 1.1794080 0.4428462 \*  
## 79) return\_rating>=0.125 1633 5.9667840 0.5270894   
## 158) category\_return\_1year>=0.2834862 962 3.5523090 0.5109238   
## 316) category\_return\_1year< 0.4450688 175 0.3269928 0.4125941 \*  
## 317) category\_return\_1year>=0.4450688 787 1.1570430 0.5327887 \*  
## 159) category\_return\_1year< 0.2834862 671 1.8026570 0.5502657 \*  
## 5) return\_rating>=0.625 1257 8.8623140 0.5615306   
## 10) asset\_stocks< 0.4953 250 0.7631373 0.4892884 \*  
## 11) asset\_stocks>=0.4953 1007 6.4705260 0.5794656   
## 22) sector\_basic\_materials>=0.1045177 116 0.7053782 0.4689589 \*  
## 23) sector\_basic\_materials< 0.1045177 891 4.1641640 0.5938526 \*  
## 3) category\_return\_1year>=0.5455275 1493 18.3003400 0.6952017   
## 6) price\_book\_ratio< 0.3322325 964 8.0642210 0.6423959   
## 12) sector\_technology< 0.2337756 280 2.6267750 0.5650365 \*  
## 13) sector\_technology>=0.2337756 684 3.0758530 0.6740635 \*  
## 7) price\_book\_ratio>=0.3322325 529 2.6495760 0.7914299 \*

# Regression Tree Model 1 - Predictive Analysis

pred\_tree\_1 = predict(regtree\_1, type="vector")  
rsq\_1 <- function (x, y) cor(x, y) ^ 2  
cat("R-Squared for Regression Tree Model 1:", rsq\_1(train$fund\_return\_10years, pred\_tree\_1))

## R-Squared for Regression Tree Model 1: 0.7231045

results\_tree\_1 <- data.frame(pred = pred\_tree\_1, original = train$fund\_return\_10years)  
results\_tree\_1 <- results\_tree\_1 %>%   
 mutate(resid = round(results\_tree\_1$pred - results\_tree\_1$original, 2))  
cat("\nSum of residuals squared for Regression Tree Model 1:", sum(results\_tree\_1$resid^2))

##   
## Sum of residuals squared for Regression Tree Model 1: 24.127

# Regression Tree Model 2 - Training

reg\_form\_2 <- as.formula("fund\_return\_10years ~ return\_rating + asset\_stocks + sector\_basic\_materials + sector\_technology + category\_return\_1year")  
regtree\_2 <- rpart(reg\_form\_2, data=train)  
regtree\_2

## n= 5482   
##   
## node), split, n, deviance, yval  
## \* denotes terminal node  
##   
## 1) root 5482 86.9735500 0.5618706   
## 2) category\_return\_1year< 0.5455275 3989 32.1980500 0.5119676   
## 4) return\_rating< 0.625 2732 18.8272100 0.4891635   
## 8) sector\_basic\_materials>=0.107423 373 3.7332770 0.4148752 \*  
## 9) sector\_basic\_materials< 0.107423 2359 12.7099500 0.5009098   
## 18) asset\_stocks< 0.4181 471 1.1979120 0.4453781 \*  
## 19) asset\_stocks>=0.4181 1888 9.6972400 0.5147633   
## 38) return\_rating< 0.125 166 1.3412370 0.4415089 \*  
## 39) return\_rating>=0.125 1722 7.3793400 0.5218250   
## 78) category\_return\_1year< 0.141055 167 1.0944270 0.4535814 \*  
## 79) category\_return\_1year>=0.141055 1555 5.4236340 0.5291541   
## 158) category\_return\_1year>=0.2834862 985 3.7445070 0.5094330   
## 316) category\_return\_1year< 0.4450688 189 0.3437251 0.4127205 \*  
## 317) category\_return\_1year>=0.4450688 796 1.2132690 0.5323962 \*  
## 159) category\_return\_1year< 0.2834862 570 0.6340390 0.5632335 \*  
## 5) return\_rating>=0.625 1257 8.8623140 0.5615306   
## 10) asset\_stocks< 0.4953 250 0.7631373 0.4892884 \*  
## 11) asset\_stocks>=0.4953 1007 6.4705260 0.5794656   
## 22) sector\_basic\_materials>=0.1045177 116 0.7053782 0.4689589 \*  
## 23) sector\_basic\_materials< 0.1045177 891 4.1641640 0.5938526 \*  
## 3) category\_return\_1year>=0.5455275 1493 18.3003400 0.6952017   
## 6) category\_return\_1year< 0.9864679 846 7.3540230 0.6349727   
## 12) category\_return\_1year>=0.5518349 209 1.0533420 0.5128828 \*  
## 13) category\_return\_1year< 0.5518349 637 2.1631870 0.6750305   
## 26) return\_rating< 0.375 199 0.5994613 0.6104393 \*  
## 27) return\_rating>=0.375 438 0.3562855 0.7043768 \*  
## 7) category\_return\_1year>=0.9864679 647 3.8646430 0.7739554   
## 14) return\_rating< 0.375 182 0.4958346 0.6827353 \*  
## 15) return\_rating>=0.375 465 1.2616170 0.8096587 \*

# Regression Tree Model 2 - Predictive Analysis

pred\_tree\_2 = predict(regtree\_2, type="vector")  
rsq\_2 <- function (x, y) cor(x, y) ^ 2  
cat("R-Squared for Regression Tree Model 2: ", rsq\_2(train$fund\_return\_10years, pred\_tree\_2))

## R-Squared for Regression Tree Model 2: 0.782036

results\_tree\_2 <- data.frame(pred = pred\_tree\_2, original = train$fund\_return\_10years)  
results\_tree\_2 <- results\_tree\_2 %>%   
 mutate(resid = round(results\_tree\_2$pred - results\_tree\_2$original, 2))  
cat("\nSum of residuals squared for Regression Tree Model 2:", sum(results\_tree\_2$resid^2))

##   
## Sum of residuals squared for Regression Tree Model 2: 19.0165

# Random Forest Models - Training

# Random Forest Model 1 - Training

forest\_form\_1 <- as.formula("fund\_return\_10years ~ return\_rating + price\_book\_ratio + asset\_cash + asset\_stocks + sector\_basic\_materials + sector\_utilities + sector\_technology + category\_return\_1year")  
forest\_1 <- randomForest(forest\_form\_1, data=train, ntree=201, mtry=3)  
forest\_1

##   
## Call:  
## randomForest(formula = forest\_form\_1, data = train, ntree = 201, mtry = 3)   
## Type of random forest: regression  
## Number of trees: 201  
## No. of variables tried at each split: 3  
##   
## Mean of squared residuals: 0.000592318  
## % Var explained: 96.27

# Random Forest Model 1 - Predictive Analysis

pred\_forest\_1 <- predict(forest\_1, newdata = test)   
mse\_forest\_1 <- sum((pred\_forest\_1 - test$fund\_return\_10years)^2/nrow(test))  
cat("MSE: ", mse\_forest\_1)

## MSE: 0.0005990188

# Random Forest Model 2 - Training

forest\_form\_2 <- as.formula("fund\_return\_10years ~ return\_rating + price\_book\_ratio + asset\_cash + asset\_stocks + sector\_basic\_materials + sector\_utilities + sector\_technology + category\_return\_1year")  
forest\_2 <- randomForest(forest\_form\_1, data=train, ntree=201, mtry=5)  
forest\_2

##   
## Call:  
## randomForest(formula = forest\_form\_1, data = train, ntree = 201, mtry = 5)   
## Type of random forest: regression  
## Number of trees: 201  
## No. of variables tried at each split: 5  
##   
## Mean of squared residuals: 0.000564895  
## % Var explained: 96.44

# Random Forest Model 2 - Predictive Analysis

pred\_forest\_2 <- predict(forest\_2, newdata = test)   
mse\_forest\_2 <- sum((pred\_forest\_2 - test$fund\_return\_10years)^2/nrow(test))  
cat("MSE: ", mse\_forest\_2)

## MSE: 0.0005767529

# Random Forest Model 3 - Training

forest\_form\_3 <- as.formula("fund\_return\_10years ~ return\_rating + price\_book\_ratio + asset\_cash + asset\_stocks + sector\_basic\_materials + sector\_utilities + sector\_technology + category\_return\_1year")  
forest\_3 <- randomForest(forest\_form\_3, data=train, ntree=501, mtry=3)  
forest\_3

##   
## Call:  
## randomForest(formula = forest\_form\_3, data = train, ntree = 501, mtry = 3)   
## Type of random forest: regression  
## Number of trees: 501  
## No. of variables tried at each split: 3  
##   
## Mean of squared residuals: 0.0005905085  
## % Var explained: 96.28

# Random Forest Model 3 - Predictive Analysis

pred\_forest\_3 <- predict(forest\_3, newdata = test)   
mse\_forest\_3 <- sum((pred\_forest\_3 - test$fund\_return\_10years)^2/nrow(test))  
cat("MSE: ", mse\_forest\_3)

## MSE: 0.0006054386

# Random Forest Model 4 - Training

forest\_form\_4 <- as.formula("fund\_return\_10years ~ price\_book\_ratio + asset\_cash + asset\_stocks + sector\_technology + category\_return\_1year")  
forest\_4 <- randomForest(forest\_form\_4, data=train, ntree=201, mtry=5)  
forest\_4

##   
## Call:  
## randomForest(formula = forest\_form\_4, data = train, ntree = 201, mtry = 5)   
## Type of random forest: regression  
## Number of trees: 201  
## No. of variables tried at each split: 5  
##   
## Mean of squared residuals: 0.0009345795  
## % Var explained: 94.11

pred\_forest\_4 <- predict(forest\_4, newdata = test)   
mse\_forest\_4 <- sum((pred\_forest\_4 - test$fund\_return\_10years)^2/nrow(test))  
cat("MSE: ", mse\_forest\_4)

## MSE: 0.001033797

#rating, return\_rating, investment\_type\_encoded, size\_type\_encoded, price\_earnings\_ratio, price\_book\_ratio, price\_sales\_ratio, asset\_cash, asset\_stocks, asset\_bonds, sector\_basic\_materials, sector\_basic\_utilities, sector\_communication\_services, sector\_technology, fund\_return\_ytd, fund\_return\_1year, category\_return\_1year, fund\_return\_3years, category\_return\_3years, fund\_return\_5years, category\_return\_5years, fund\_beta\_3years, category\_beta\_3years, fund\_beta\_5years, category\_beta\_5years, fund\_mean\_annual\_return\_3years, fund\_mean\_annual\_return\_3years, net\_asset\_value

# Neural Network Models - Training

# Neural Network Model 1 - Training

neural\_form\_1 <- as.formula("fund\_return\_10years ~ return\_rating + price\_book\_ratio + asset\_cash + asset\_stocks + sector\_technology")  
  
nn\_1 <- neuralnet(neural\_form\_1, data = train, hidden=c(3,2), linear.output = FALSE)  
plot(nn\_1)

t <- test %>%   
 dplyr::select(fund\_return\_10years, return\_rating ,price\_book\_ratio , asset\_cash, asset\_stocks,sector\_technology, net\_asset\_value,category\_return\_3years, fund\_return\_5years, category\_return\_5years, investment\_type\_encoded, size\_type\_encoded)

# Neural Network Model 1 - Predictive Analysis

pred\_nn\_1<- compute(nn\_1, t)  
pred\_nn\_result\_1 <- pred\_nn\_1$net.result\*(max(funds\_scaled$fund\_return\_10years) - min(funds\_scaled$fund\_return\_10years))+min(funds\_scaled$fund\_return\_10years)  
  
MSE.nn <- sum((t - pred\_nn\_result\_1)^2)/nrow(t)  
cat("MSE for NN:", MSE.nn, "\n")

## MSE for NN: 5.080336

plot(t$fund\_return\_10years, pred\_nn\_result\_1, col='blue', pch=16, ylab = "predicted rating NN", xlab = "real rating")

Chart, scatter chart

Description automatically generated

#Test the resulting output  
results1 <- data.frame(actual1 = t$fund\_return\_10years, prediction1 = pred\_nn\_1$net.result)  
roundedresults1 <-sapply(results1,round,digits=0)  
roundedresultsdf1 =data.frame(roundedresults1)  
table\_nn1 <- table(roundedresultsdf1$actual1,roundedresultsdf1$prediction1)  
cat("\nAccuracy:", sum(diag(table\_nn1))/nrow(t))

##   
## Accuracy: 0.7905109

# Neural Network Model 2 - Training

neural\_form\_2 <- as.formula("fund\_return\_10years ~ return\_rating + price\_book\_ratio + asset\_cash + asset\_stocks + sector\_technology")  
nn\_2 <- neuralnet(neural\_form\_2, data = train, hidden=c(5,3,2),linear.output = FALSE)  
plot(nn\_2)

# Neural Network Model 2 - Predictive Analysis

pr\_nn\_2 <- compute(nn\_2, t)  
nn\_results\_2 <- pr\_nn\_2$net.result\*(max(funds\_scaled$fund\_return\_10years) - min(funds\_scaled$fund\_return\_10years))+min(funds\_scaled$fund\_return\_10years)  
MSE.nn2 <- sum((t - nn\_results\_2)^2)/nrow(t)  
cat("MSE for NN:", MSE.nn2)

## MSE for NN: 5.079133

plot(t$fund\_return\_10years, nn\_results\_2, col='blue', pch=16, ylab = "predicted rating NN", xlab = "real rating")

Chart, scatter chart

Description automatically generated

#Test the resulting output  
results2 <- data.frame(actual2 = t$fund\_return\_10years, prediction2 = pr\_nn\_2$net.result)  
roundedresults2 <-sapply(results2,round,digits=0)  
roundedresultsdf2 = data.frame(roundedresults2)  
  
table\_nn2 <- table(roundedresultsdf2$actual2,roundedresultsdf2$prediction2)  
cat("\nAccuracy:", sum(diag(table\_nn2))/nrow(t))

##   
## Accuracy: 0.8021898

# Categorical response variable -> mutate -> return\_over\_inflation

Funds2 <- new\_funds %>%   
 dplyr::select(rating, return\_rating, risk\_rating, investment\_type\_encoded, size\_type\_encoded, price\_earnings\_ratio, price\_book\_ratio, price\_sales\_ratio, asset\_cash, asset\_stocks, asset\_bonds, asset\_others, asset\_preferred, asset\_convertable, sector\_basic\_materials, sector\_consumer\_cyclical, sector\_financial\_services, sector\_real\_estate, sector\_consumer\_defensive, sector\_healthcare, sector\_utilities, sector\_communication\_services, sector\_energy, sector\_industrials, sector\_technology, fund\_return\_ytd, fund\_return\_1year, category\_return\_1year, fund\_return\_3years, category\_return\_3years, fund\_return\_5years, category\_return\_5years, fund\_return\_10years, category\_return\_10years, fund\_beta\_3years, category\_beta\_3years, fund\_beta\_5years, category\_beta\_5years, fund\_beta\_10years, category\_beta\_10years, fund\_mean\_annual\_return\_3years, fund\_mean\_annual\_return\_3years, fund\_mean\_annual\_return\_10years, net\_asset\_value)

# Categorical response variable training/testing data sets

Funds2 <- na.omit(Funds2)  
Funds2 <- Funds2 %>%   
 mutate(Funds2, return\_over\_inflation=ifelse(fund\_return\_10years >= 12.37, TRUE, FALSE))

# Performs normalization

# Remove rows that contain missing data.  
Funds2 <- Funds2[complete.cases(Funds2),]  
  
# Convert columns to numeric  
cols <- colnames(Funds2)  
Funds2[cols] <- sapply(Funds2[cols],as.numeric)  
sapply(Funds2, class)

## rating return\_rating   
## "numeric" "numeric"   
## risk\_rating investment\_type\_encoded   
## "numeric" "numeric"   
## size\_type\_encoded price\_earnings\_ratio   
## "numeric" "numeric"   
## price\_book\_ratio price\_sales\_ratio   
## "numeric" "numeric"   
## asset\_cash asset\_stocks   
## "numeric" "numeric"   
## asset\_bonds asset\_others   
## "numeric" "numeric"   
## asset\_preferred asset\_convertable   
## "numeric" "numeric"   
## sector\_basic\_materials sector\_consumer\_cyclical   
## "numeric" "numeric"   
## sector\_financial\_services sector\_real\_estate   
## "numeric" "numeric"   
## sector\_consumer\_defensive sector\_healthcare   
## "numeric" "numeric"   
## sector\_utilities sector\_communication\_services   
## "numeric" "numeric"   
## sector\_energy sector\_industrials   
## "numeric" "numeric"   
## sector\_technology fund\_return\_ytd   
## "numeric" "numeric"   
## fund\_return\_1year category\_return\_1year   
## "numeric" "numeric"   
## fund\_return\_3years category\_return\_3years   
## "numeric" "numeric"   
## fund\_return\_5years category\_return\_5years   
## "numeric" "numeric"   
## fund\_return\_10years category\_return\_10years   
## "numeric" "numeric"   
## fund\_beta\_3years category\_beta\_3years   
## "numeric" "numeric"   
## fund\_beta\_5years category\_beta\_5years   
## "numeric" "numeric"   
## fund\_beta\_10years category\_beta\_10years   
## "numeric" "numeric"   
## fund\_mean\_annual\_return\_3years fund\_mean\_annual\_return\_10years   
## "numeric" "numeric"   
## net\_asset\_value return\_over\_inflation   
## "numeric" "numeric"

# Max and mins for normalization  
maxs <- apply(Funds2, 2, max)   
mins <- apply(Funds2, 2, min)  
  
# Perform scaling  
funds\_scaled\_2 <- as.data.frame(scale(Funds2, center = mins, scale = maxs - mins))

set.seed(200)  
n <- nrow(funds\_scaled\_2)  
test\_idx <- sample.int(n, size=round(0.2 \* n))  
train2 <- funds\_scaled\_2[-test\_idx,]  
test2 <- funds\_scaled\_2[test\_idx,]

# Null Model

null\_model\_2 <- prop.table(table(Funds2$return\_over\_inflation))  
null\_model\_2

##   
## 0 1   
## 0.8705487 0.1294513

# Multiple Regression Models

# Multiple Logistic Regression Model 1 - Training

lg1\_train <- glm(return\_over\_inflation ~ net\_asset\_value + return\_rating + asset\_cash + asset\_stocks + sector\_technology + size\_type\_encoded, data = train2, family = binomial)  
summary(lg1\_train)

##   
## Call:  
## glm(formula = return\_over\_inflation ~ net\_asset\_value + return\_rating +   
## asset\_cash + asset\_stocks + sector\_technology + size\_type\_encoded,   
## family = binomial, data = train2)  
##   
## Deviance Residuals:   
## Min 1Q Median 3Q Max   
## -3.1053 -0.1701 -0.0381 -0.0038 4.7277   
##   
## Coefficients:  
## Estimate Std. Error z value Pr(>|z|)   
## (Intercept) -25.4156 1.8258 -13.921 < 0.0000000000000002 \*\*\*  
## net\_asset\_value 12.5505 1.6708 7.512 0.0000000000000583 \*\*\*  
## return\_rating 3.6859 0.2880 12.798 < 0.0000000000000002 \*\*\*  
## asset\_cash 6.2140 1.9981 3.110 0.00187 \*\*   
## asset\_stocks 15.4424 1.7316 8.918 < 0.0000000000000002 \*\*\*  
## sector\_technology 23.4929 0.9838 23.879 < 0.0000000000000002 \*\*\*  
## size\_type\_encoded -3.3886 0.6826 -4.964 0.0000006897203185 \*\*\*  
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## (Dispersion parameter for binomial family taken to be 1)  
##   
## Null deviance: 4237.6 on 5481 degrees of freedom  
## Residual deviance: 1597.1 on 5475 degrees of freedom  
## AIC: 1611.1  
##   
## Number of Fisher Scoring iterations: 9

# Multiple Logistic Regression Model 1 - Predictive Analysis

# Let's take a look at how the model performs with predictions.  
logistic\_1\_probs = predict(lg1\_train, type="response")  
favstats(logistic\_1\_probs)

## min Q1 median Q3 max mean  
## 0.000000000002446327 0.0001169815 0.003374683 0.08183308 0.9999945 0.130062  
## sd n missing  
## 0.2622391 5482 0

# Build predictions based on probabilities.  
logistic\_1\_pred = rep(FALSE, nrow(train2))  
logistic\_1\_pred[logistic\_1\_probs > 0.5] = TRUE  
  
# Misclassification table and accuracy.  
table1 <- table(logistic\_1\_pred, train2$return\_over\_inflation)  
table1

##   
## logistic\_1\_pred 0 1  
## FALSE 4658 220  
## TRUE 111 493

cat("\nAccuracy: ", sum(diag(table1))/sum(table1))

##   
## Accuracy: 0.9396206

# Multiple Logistic Regression Model 2 - Training

lg2\_train <- glm(return\_over\_inflation ~ net\_asset\_value + return\_rating + asset\_stocks + sector\_technology + size\_type\_encoded, data = train2, family = binomial)  
summary(lg2\_train)

##   
## Call:  
## glm(formula = return\_over\_inflation ~ net\_asset\_value + return\_rating +   
## asset\_stocks + sector\_technology + size\_type\_encoded, family = binomial,   
## data = train2)  
##   
## Deviance Residuals:   
## Min 1Q Median 3Q Max   
## -3.0912 -0.1770 -0.0423 -0.0051 4.5674   
##   
## Coefficients:  
## Estimate Std. Error z value Pr(>|z|)   
## (Intercept) -22.2898 1.2758 -17.472 < 0.0000000000000002 \*\*\*  
## net\_asset\_value 13.4267 1.6894 7.948 0.0000000000000019 \*\*\*  
## return\_rating 3.7134 0.2885 12.871 < 0.0000000000000002 \*\*\*  
## asset\_stocks 12.4573 1.2035 10.351 < 0.0000000000000002 \*\*\*  
## sector\_technology 23.3128 0.9910 23.524 < 0.0000000000000002 \*\*\*  
## size\_type\_encoded -3.3569 0.6794 -4.941 0.0000007763451741 \*\*\*  
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## (Dispersion parameter for binomial family taken to be 1)  
##   
## Null deviance: 4237.6 on 5481 degrees of freedom  
## Residual deviance: 1606.1 on 5476 degrees of freedom  
## AIC: 1618.1  
##   
## Number of Fisher Scoring iterations: 9

# Multiple Logistic Regression Model 2 - Predictive Analysis

# Let's take a look at how the model performs with predictions.  
logistic\_2\_probs = predict(lg2\_train, type="response")  
favstats(logistic\_2\_probs)

## min Q1 median Q3 max mean  
## 0.00000000003131244 0.0001698697 0.003891956 0.08514347 0.9999979 0.130062  
## sd n missing  
## 0.2619738 5482 0

# Build predictions based on probabilities.  
logistic\_2\_pred = rep(FALSE, nrow(train2))  
logistic\_2\_pred[logistic\_2\_probs > 0.5] = TRUE  
  
# Misclassification table and accuracy.  
table2 <- table(logistic\_2\_pred, train2$return\_over\_inflation)  
table2

##   
## logistic\_2\_pred 0 1  
## FALSE 4659 213  
## TRUE 110 500

cat("\nAccuracy: ", sum(diag(table2))/sum(table2))

##   
## Accuracy: 0.9410799

# Classification Tree Models

# Classification Tree Model 1 - Training

class\_tree\_form <- as.formula("return\_over\_inflation ~ net\_asset\_value + return\_rating + asset\_cash + asset\_stocks + sector\_technology + size\_type\_encoded")  
class\_tree\_1 <- rpart(class\_tree\_form, data=train2)  
class\_tree\_1

## n= 5482   
##   
## node), split, n, deviance, yval  
## \* denotes terminal node  
##   
## 1) root 5482 620.265800 0.13006200   
## 2) sector\_technology< 0.2963812 4468 118.668800 0.02730528   
## 4) net\_asset\_value< 0.2084689 4450 103.475100 0.02382022   
## 8) sector\_technology< 0.2834138 4228 57.204350 0.01371807 \*  
## 9) sector\_technology>=0.2834138 222 37.621620 0.21621620   
## 18) return\_rating< 0.625 118 3.864407 0.03389831 \*  
## 19) return\_rating>=0.625 104 25.384620 0.42307690   
## 38) asset\_stocks< 0.6818 35 0.000000 0.00000000 \*  
## 39) asset\_stocks>=0.6818 69 15.942030 0.63768120 \*  
## 5) net\_asset\_value>=0.2084689 18 1.777778 0.88888890 \*  
## 3) sector\_technology>=0.2963812 1014 246.541400 0.58284020   
## 6) asset\_stocks< 0.9101 150 7.573333 0.05333333 \*  
## 7) asset\_stocks>=0.9101 864 189.610000 0.67476850   
## 14) return\_rating< 0.375 141 28.212770 0.27659570 \*  
## 15) return\_rating>=0.375 723 134.683300 0.75242050   
## 30) sector\_technology< 0.3361279 258 63.740310 0.55426360   
## 60) return\_rating< 0.625 110 16.990910 0.19090910 \*  
## 61) return\_rating>=0.625 148 21.432430 0.82432430 \*  
## 31) sector\_technology>=0.3361279 465 55.191400 0.86236560 \*

# Classification Tree Model 1 - Predictive Analysis

test21 <- test2 %>%  
 dplyr::mutate(x\_return = ifelse(predict(class\_tree\_1, newdata = test2) >= 0.5, 1, 0))  
confusion <- tally(x\_return ~ return\_over\_inflation, data=test21, format="count")  
cat("Accuracy:", sum(diag(confusion))/nrow(test21))

## Accuracy: 0.9489051

# Classification Tree Model 2 - Training

class\_tree\_form\_2 <- as.formula("return\_over\_inflation ~ net\_asset\_value + return\_rating + asset\_cash + sector\_technology + asset\_stocks")  
class\_tree\_2 <- rpart(class\_tree\_form\_2, data=train2)  
class\_tree\_2

## n= 5482   
##   
## node), split, n, deviance, yval  
## \* denotes terminal node  
##   
## 1) root 5482 620.265800 0.13006200   
## 2) sector\_technology< 0.2963812 4468 118.668800 0.02730528   
## 4) net\_asset\_value< 0.2084689 4450 103.475100 0.02382022   
## 8) sector\_technology< 0.2834138 4228 57.204350 0.01371807 \*  
## 9) sector\_technology>=0.2834138 222 37.621620 0.21621620   
## 18) return\_rating< 0.625 118 3.864407 0.03389831 \*  
## 19) return\_rating>=0.625 104 25.384620 0.42307690   
## 38) asset\_stocks< 0.6818 35 0.000000 0.00000000 \*  
## 39) asset\_stocks>=0.6818 69 15.942030 0.63768120 \*  
## 5) net\_asset\_value>=0.2084689 18 1.777778 0.88888890 \*  
## 3) sector\_technology>=0.2963812 1014 246.541400 0.58284020   
## 6) asset\_stocks< 0.9101 150 7.573333 0.05333333 \*  
## 7) asset\_stocks>=0.9101 864 189.610000 0.67476850   
## 14) return\_rating< 0.375 141 28.212770 0.27659570 \*  
## 15) return\_rating>=0.375 723 134.683300 0.75242050   
## 30) sector\_technology< 0.3361279 258 63.740310 0.55426360   
## 60) return\_rating< 0.625 110 16.990910 0.19090910 \*  
## 61) return\_rating>=0.625 148 21.432430 0.82432430 \*  
## 31) sector\_technology>=0.3361279 465 55.191400 0.86236560 \*

# Classification Tree Model 2 - Predictive Analysis

test22 <- test2 %>%  
 dplyr::mutate(x\_return = ifelse(predict(class\_tree\_2, newdata = test2) >= 0.5, 1, 0))  
confusion2 <- tally(x\_return ~ return\_over\_inflation, data=test22, format="count")  
cat("Accuracy:", sum(diag(confusion2))/nrow(test22))

## Accuracy: 0.9489051

# Random Forest Models

# Random Forest Model 1 - Training

train2$return\_over\_inflation <- as.character(train2$return\_over\_inflation)  
train2$return\_over\_inflation <- as.factor(train2$return\_over\_inflation)  
  
f <- as.formula("return\_over\_inflation ~ net\_asset\_value + return\_rating + asset\_cash + asset\_stocks + sector\_technology + size\_type\_encoded")  
mod\_forest <- randomForest(f, data=train2, ntree=200, mtry=3)  
mod\_forest

##   
## Call:  
## randomForest(formula = f, data = train2, ntree = 200, mtry = 3)   
## Type of random forest: classification  
## Number of trees: 200  
## No. of variables tried at each split: 3  
##   
## OOB estimate of error rate: 2.02%  
## Confusion matrix:  
## 0 1 class.error  
## 0 4720 49 0.01027469  
## 1 62 651 0.08695652

# Random Forest Model 1 - Predictive Analysis

sum(diag(mod\_forest$confusion))/nrow(train2)

## [1] 0.9797519

# Random Forest Model 2 - Training

train2$return\_over\_inflation <- as.character(train2$return\_over\_inflation)  
train2$return\_over\_inflation <- as.factor(train2$return\_over\_inflation)  
f2 <- as.formula("return\_over\_inflation ~ net\_asset\_value + return\_rating + asset\_cash + asset\_stocks + sector\_technology")  
mod\_forest2 <- randomForest(f2, data=train2, ntree=200, mtry=5)  
mod\_forest2

##   
## Call:  
## randomForest(formula = f2, data = train2, ntree = 200, mtry = 5)   
## Type of random forest: classification  
## Number of trees: 200  
## No. of variables tried at each split: 5  
##   
## OOB estimate of error rate: 2.1%  
## Confusion matrix:  
## 0 1 class.error  
## 0 4718 51 0.01069407  
## 1 64 649 0.08976157

# Random Forest Model 2 - Predictive Analysis

sum(diag(mod\_forest2$confusion))/nrow(train2)

## [1] 0.9790223

# Random Forest Model 3 - Training

train2$return\_over\_inflation <- as.character(train2$return\_over\_inflation)  
train2$return\_over\_inflation <- as.factor(train2$return\_over\_inflation)  
f3 <- as.formula("return\_over\_inflation ~ net\_asset\_value + return\_rating + asset\_cash + asset\_stocks + sector\_technology")  
mod\_forest3 <- randomForest(f3, data=train2, ntree=300, mtry=3)  
mod\_forest3

##   
## Call:  
## randomForest(formula = f3, data = train2, ntree = 300, mtry = 3)   
## Type of random forest: classification  
## Number of trees: 300  
## No. of variables tried at each split: 3  
##   
## OOB estimate of error rate: 2.06%  
## Confusion matrix:  
## 0 1 class.error  
## 0 4718 51 0.01069407  
## 1 62 651 0.08695652

# Random Forest Model 3 - Predictive Analysis

sum(diag(mod\_forest3$confusion))/nrow(train2)

## [1] 0.9793871

# KNN Models

# KNN Model 1 - Training and Predictive Analysis

train3 <- train2 %>%   
 dplyr::select(return\_rating , sector\_technology , net\_asset\_value, asset\_cash, asset\_stocks)  
  
test3 <- test2 %>%   
 dplyr::select(return\_rating , sector\_technology , net\_asset\_value, asset\_cash, asset\_stocks)  
  
knn\_1 <- knn(train=train3, test=test3, cl=train2$return\_over\_inflation, k=2)  
confusion\_knn <- table(knn\_1, test2$return\_over\_inflation)  
confusionMatrix(confusion\_knn)

## Confusion Matrix and Statistics  
##   
##   
## knn\_1 0 1  
## 0 1163 24  
## 1 33 150  
##   
## Accuracy : 0.9584   
## 95% CI : (0.9464, 0.9683)   
## No Information Rate : 0.873   
## P-Value [Acc > NIR] : <0.0000000000000002  
##   
## Kappa : 0.8164   
##   
## Mcnemar's Test P-Value : 0.2893   
##   
## Sensitivity : 0.9724   
## Specificity : 0.8621   
## Pos Pred Value : 0.9798   
## Neg Pred Value : 0.8197   
## Prevalence : 0.8730   
## Detection Rate : 0.8489   
## Detection Prevalence : 0.8664   
## Balanced Accuracy : 0.9172   
##   
## 'Positive' Class : 0   
##

# KNN Model 2 - Training and Predictive Analysis

knn\_2 <- knn(train=train3, test = test3, cl=train2$return\_over\_inflation, k=4)  
confusion\_knn2 <- table(knn\_2, test2$return\_over\_inflation)  
confusionMatrix(confusion\_knn2)

## Confusion Matrix and Statistics  
##   
##   
## knn\_2 0 1  
## 0 1159 22  
## 1 37 152  
##   
## Accuracy : 0.9569   
## 95% CI : (0.9448, 0.9671)   
## No Information Rate : 0.873   
## P-Value [Acc > NIR] : < 0.0000000000000002  
##   
## Kappa : 0.8127   
##   
## Mcnemar's Test P-Value : 0.06836   
##   
## Sensitivity : 0.9691   
## Specificity : 0.8736   
## Pos Pred Value : 0.9814   
## Neg Pred Value : 0.8042   
## Prevalence : 0.8730   
## Detection Rate : 0.8460   
## Detection Prevalence : 0.8620   
## Balanced Accuracy : 0.9213   
##   
## 'Positive' Class : 0   
##

# Naive Bayes Models

# Naive Bayes Model 1 - Training and Predictive Analysis

f <- as.formula("return\_over\_inflation ~ net\_asset\_value + return\_rating + asset\_cash + asset\_stocks + sector\_technology + size\_type\_encoded")  
  
nb\_1 <- naiveBayes(f, data=train2)  
nb\_pred\_1 <- predict(nb\_1, newdata=train2)  
confusiontable<- table(nb\_pred\_1, train2$return\_over\_inflation)  
confusionMatrix(confusiontable)

## Confusion Matrix and Statistics  
##   
##   
## nb\_pred\_1 0 1  
## 0 4002 43  
## 1 767 670  
##   
## Accuracy : 0.8522   
## 95% CI : (0.8426, 0.8615)   
## No Information Rate : 0.8699   
## P-Value [Acc > NIR] : 0.9999   
##   
## Kappa : 0.544   
##   
## Mcnemar's Test P-Value : <0.0000000000000002  
##   
## Sensitivity : 0.8392   
## Specificity : 0.9397   
## Pos Pred Value : 0.9894   
## Neg Pred Value : 0.4662   
## Prevalence : 0.8699   
## Detection Rate : 0.7300   
## Detection Prevalence : 0.7379   
## Balanced Accuracy : 0.8894   
##   
## 'Positive' Class : 0   
##

# Naive Bayes Model 2 - Training and Predictive Analysis

f2 <- as.formula("return\_over\_inflation ~ net\_asset\_value + return\_rating + sector\_technology + size\_type\_encoded")  
  
nb\_2 <- naiveBayes(f2, data=train2)  
nb\_pred\_2 <- predict(nb\_2, newdata=train2)  
confusiontable2 <- table(nb\_pred\_2, train2$return\_over\_inflation)  
confusionMatrix(confusiontable2)

## Confusion Matrix and Statistics  
##   
##   
## nb\_pred\_2 0 1  
## 0 4466 189  
## 1 303 524  
##   
## Accuracy : 0.9103   
## 95% CI : (0.9024, 0.9177)   
## No Information Rate : 0.8699   
## P-Value [Acc > NIR] : < 0.00000000000000022  
##   
## Kappa : 0.6286   
##   
## Mcnemar's Test P-Value : 0.0000003498   
##   
## Sensitivity : 0.9365   
## Specificity : 0.7349   
## Pos Pred Value : 0.9594   
## Neg Pred Value : 0.6336   
## Prevalence : 0.8699   
## Detection Rate : 0.8147   
## Detection Prevalence : 0.8491   
## Balanced Accuracy : 0.8357   
##   
## 'Positive' Class : 0   
##