Clifford Huang
Final Project
September 20, 2021
Dr. Block

Financial Investment Company Project

Introduction

Grant's financial investment company offers clients financial investment portfolios. Each portfolio consists of multiple sleeves, which are individually managed by investment firms. Grant wants to know how data can be used to help assess the performance of each sleeve.

Grant has provided a dataset containing 289 anonymous sleeves. For the purposes of this project, the variables were reduced to historic returns of the sleeves on multiple time frames (1yr, 3yr, 5yr, 7yr and 10yr), the weights and weighted scores given to each time frame, and tabulated performance score across all time frames for each sleeve.

In this project, neural networks are used to predict last year's performance on returns based on the 3yr, 5yr, 7yr and 10yr returns. Comparisons are made using the logistic and hyperbolic tangent activation functions. These methods are also compared against a linear model. Parameters from the neural networks are changed from single layer to double layer and the results are tested and compared. Finally, different weight configurations are given to each time frame and the resulting scores are evaluated in their effectiveness in predicting last year's performance on returns.

Preparing the Data

The 1yr returns contained no NAs and since there was only one year's worth of data, the 1yr returns were selected for the target output of the model. The 3yr average returns also had no NAs. The 5yr average returns had 9 NAs; the 7yr average returns had 35 NAs; and the 10yr average returns had 75 NAs. Removing 75 observations would drastically reduce the dataset and so initially these observations were replaced with the column means. However, this drastically reduced the performance of the models and so the NAs were omitted. The final data frame contained 212 observations. 75% of the observations

```
> summary(invest)
    sleeve            minimum              fee              yr1.ret          yr1.rank          yr1.ws          yr3.ret
 Length:287         Length:287         Length:287         Min.   :0.3511   Min.   :0.0000   Min.   :0.0000   Min.   :-0.29000
 Class :character   Class :character   Class :character   1st Qu.:0.6627   1st Qu.:0.2495   1st Qu.:0.4990   1st Qu.: 0.07875
 Mode  :character   Mode  :character   Mode  :character   Median :0.7580   Median :0.4960   Median :0.9920   Median : 0.09670
                                                          Mean   :0.7742   Mean   :0.4987   Mean   :0.9974   Mean   : 0.09021
                                                          3rd Qu.:0.8964   3rd Qu.:0.7495   3rd Qu.:1.4990   3rd Qu.: 0.11445
                                                          Max.   :1.3173   Max.   :1.0000   Max.   :2.0000   Max.   : 0.24570

    yr3.rank         yr3.ws          yr5.ret           yr5.rank         yr5.ws          yr7.ret           yr7.rank
 Min.   :0.0000   Min.   :0.0000   Min.   :-0.1895   Min.   :0.0000   Min.   :0.0000   Min.   :-0.24410   Min.   :0.0000
 1st Qu.:0.2495   1st Qu.:0.4242   1st Qu.: 0.1055   1st Qu.:0.2215   1st Qu.:0.3322   1st Qu.: 0.07362   1st Qu.:0.1450
 Median :0.5000   Median :0.8500   Median : 0.1231   Median :0.4800   Median :0.7200   Median : 0.08720   Median :0.4300
 Mean   :0.4981   Mean   :0.8468   Mean   : 0.1146   Mean   :0.4838   Mean   :0.7257   Mean   : 0.07743   Mean   :0.4439
 3rd Qu.:0.7495   3rd Qu.:1.2741   3rd Qu.: 0.1371   3rd Qu.:0.7400   3rd Qu.:1.1100   3rd Qu.: 0.10222   3rd Qu.:0.7090
 Max.   :1.0000   Max.   :1.7000   Max.   : 0.2144   Max.   :1.0000   Max.   :1.5000   Max.   : 0.15550   Max.   :1.0000
                                   NA's   :9                                           NA's   :35
    yr7.ws          yr10.ret          yr10.rank         yr10.ws           score
 Min.   :0.0000   Min.   :-0.16800   Min.   :0.0000   Min.   :0.0000   Min.   :0.1798
 1st Qu.:0.1885   1st Qu.: 0.08615   1st Qu.:0.0360   1st Qu.:0.0396   1st Qu.:2.1936
 Median :0.5590   Median : 0.09900   Median :0.3220   Median :0.3542   Median :3.7178
 Mean   :0.5770   Mean   : 0.08982   Mean   :0.3773   Mean   :0.4151   Mean   :3.5620
 3rd Qu.:0.9217   3rd Qu.: 0.10952   3rd Qu.:0.6585   3rd Qu.:0.7244   3rd Qu.:4.7683
 Max.   :1.3000   Max.   : 0.15940   Max.   :1.0000   Max.   :1.1000   Max.   :7.6000
                  NA's   :75
```

were used in training the model and 25% were used for testing. These proportions were chosen in order to reduce variation in our training models while still leaving sufficient observations to test the results.

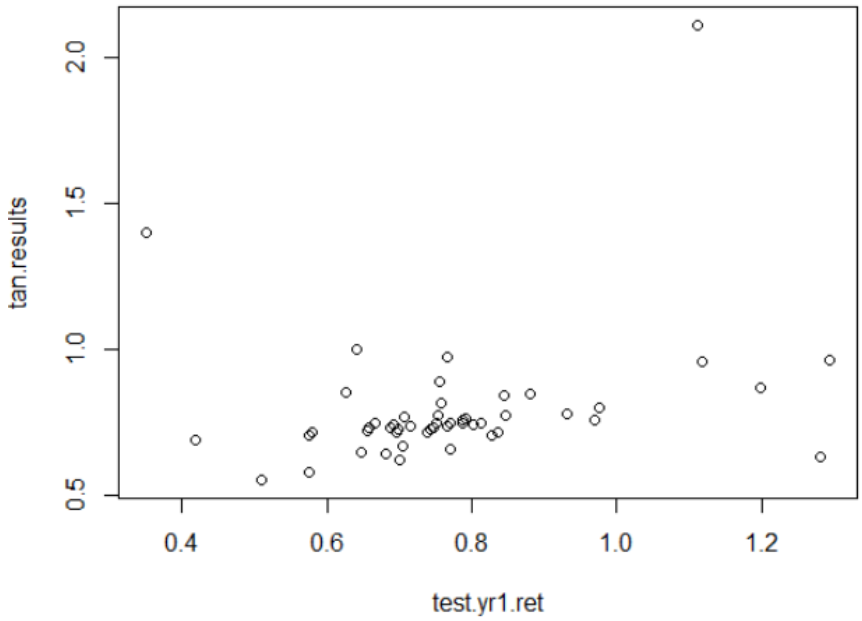
Neural Networks

The neural networks using both the hyperbolic tangent and logistic activation functions proved significantly more effective than the simple linear model. Using a for loop to set multiple seeds, a sampling of results were gathered to confirm the results.

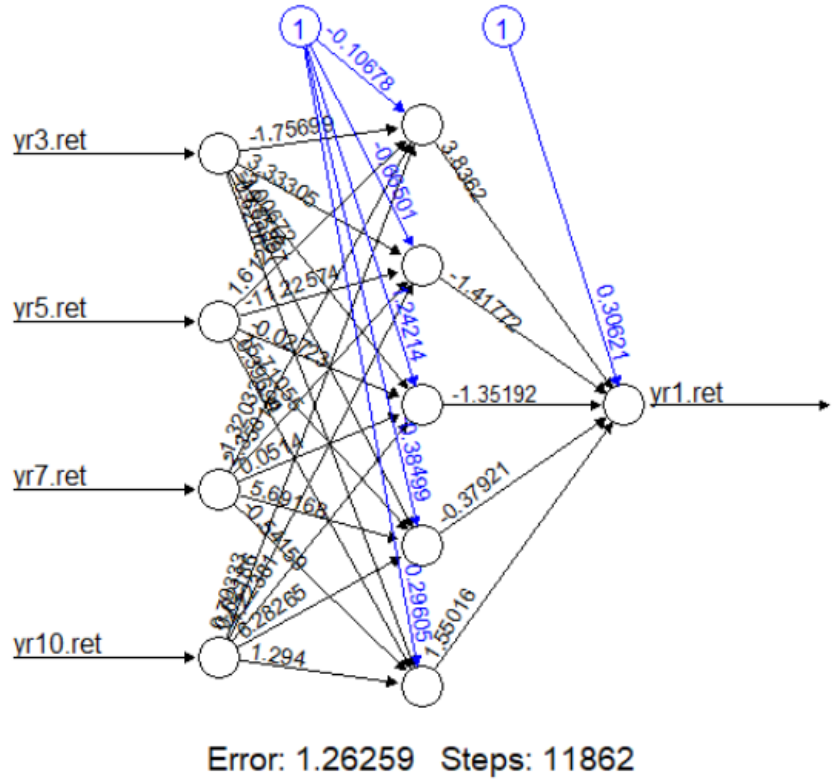
```
for(i in c(1,2,3,5,8,13, 21, 34, 55, 89, 100)){
set.seed(i)

# use 75% for training
# use 25% for testing

train <- slice_sample(invest, prop=0.75)
test <- setdiff(invest, train)

### create neural networks
# train the model to use 10 yr, 7 yr, 5 yr, and 3 yr returns to predict 1 yr returns
# using hyperbolic tangent activation function
tanmodel <- neuralnet(yr1.ret ~ yr3.ret+yr5.ret+yr7.ret+yr10.ret, data=train, hidden=5, act.fct="tanh", linear.output=F)
plot(tanmodel)

# using logistic model activation function
logmodel <- neuralnet(yr1.ret ~ yr3.ret+yr5.ret+yr7.ret+yr10.ret, data=train, hidden=5, act.fct="logistic")
plot(logmodel)
```

In 9 out of 11 iterations, the neural networks clearly outperformed the linear model. However, it was unclear which was better between the hyperbolic tangent or logistic activation methods as both had similarly varying results.

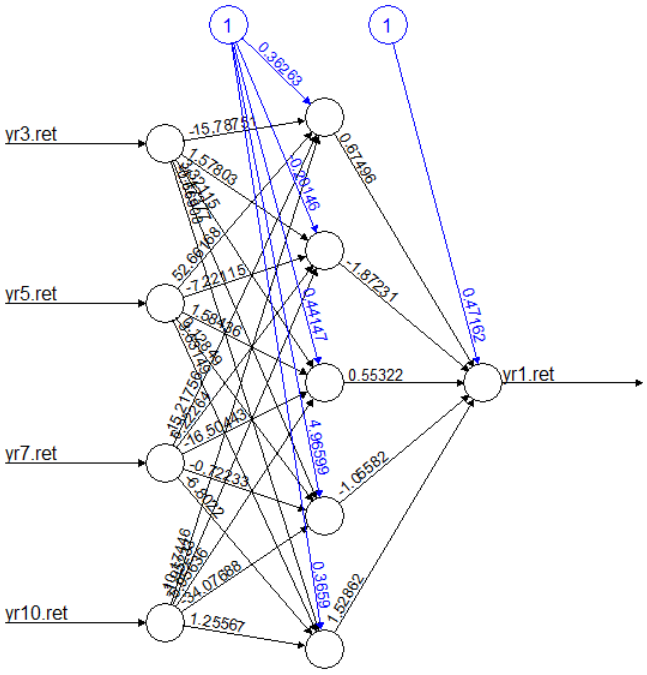
```
[1] "htan seed:  1   0.696537862064238"
[1] "log seed:  1   0.707723249475516"
[1] "line seed:  1   0.659800104674612"
[1] "htan seed:  2   0.694437673316019"
[1] "log seed:  2   0.720186961701067"
[1] "line seed:  2   0.528689625199998"
[1] "htan seed:  3   0.598566177362591"
[1] "log seed:  3   0.548320642388146"
[1] "line seed:  3   0.555687796453383"
[1] "htan seed:  5   0.606426039032913"
[1] "log seed:  5   0.581856047779466"
[1] "line seed:  5   0.390441241486213"
[1] "htan seed:  8   0.548571566757301"
[1] "log seed:  8   0.541811820885406"
[1] "line seed:  8   0.340806247389592"
[1] "htan seed:  13   0.636627806558923"
[1] "log seed:  13   0.597692043600077"
[1] "line seed:  13   0.543372658636936"
[1] "htan seed:  21   0.667921305053576"
[1] "log seed:  21   0.716120038115709"
[1] "line seed:  21   0.559711929152236"
[1] "htan seed:  34   0.712266581780957"
[1] "log seed:  34   0.729858168094746"
[1] "line seed:  34   0.580453203770904"
[1] "htan seed:  55   0.598903867011354"
[1] "log seed:  55   0.667023293131152"
[1] "line seed:  55   0.632863777373887"
[1] "htan seed:  89   0.659027827491049"
[1] "log seed:  89   0.635267481966398"
[1] "line seed:  89   0.520655876340986"
[1] "htan seed:  100   0.509131709556609"
[1] "log seed:  100   0.52970014187825"
[1] "line seed:  100   0.414803934532843"
```



1Hyperbolic Tangent Test Results

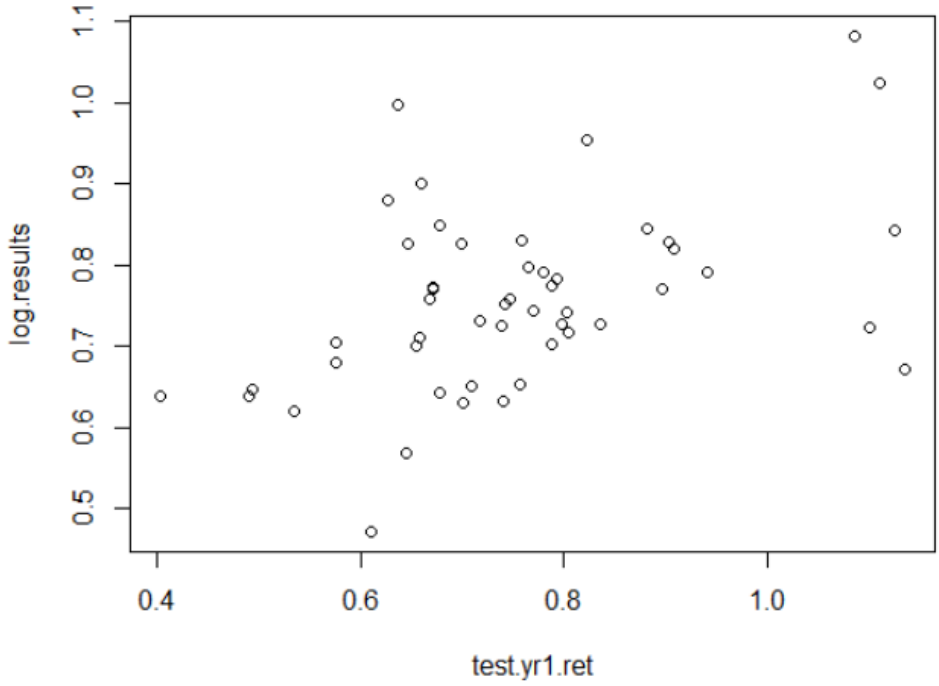


2 Hyperbolic Tangent Neural Network

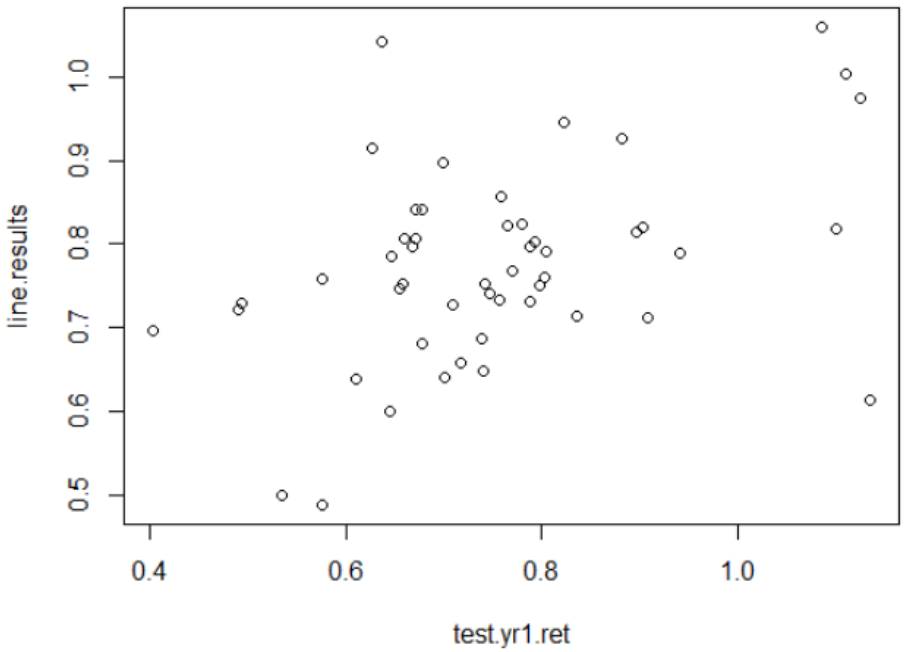


Error: 1.303889 Steps: 5881

3 Logistic Neural Network



4 Logistic Model Test Results



5 Line Model Test Results

Node Optimization

The original number of nodes was arbitrarily chosen to be 5. In order to find a better solution, for loops were used to adjust the number of nodes.

```
### use for loops to identify optimal number of nodes
# single layer neural network
for(nodes in c(1,2,3,4,5,6,7,8,9,10)){
  tanmodel <- neuralnet(yr1.ret ~ yr3.ret+yr5.ret+yr7.ret+yr10.ret, data=train, hidden=nodes, act.fct="tanh", linear.outpu
  tan.results <- predict(tanmodel, test)
  tan.results <- data.frame(test$yr1.ret, tan.results)
  corr<- cor(tan.results$test.yr1.ret, tan.results$tan.results)
  print(paste("Nodes: ", nodes, "   Correlation: ", corr))
}

for(nodes in c(1,2,3,4,5,6,7,8,9,10)){
  logmodel <- neuralnet(yr1.ret ~ yr3.ret+yr5.ret+yr7.ret+yr10.ret, data=train, hidden=nodes)
  log.results <- predict(logmodel, test)
  log.results <- data.frame(test$yr1.ret, log.results)
  corr<- cor(log.results$test.yr1.ret, log.results$log.results)
  print(paste("Nodes: ", nodes, "   Correlation: ", corr))
}
```

```
[1] "Nodes:  1    Correlation:  0.376907650450353"
[1] "Nodes:  2    Correlation:  0.382548884741015"
[1] "Nodes:  3    Correlation:  0.47001776741663"
[1] "Nodes:  4    Correlation:  0.488705527192478"
[1] "Nodes:  5    Correlation:  0.476039259535115"
[1] "Nodes:  6    Correlation:  0.455521036492595"
[1] "Nodes:  7    Correlation:  0.518899024712725"
[1] "Nodes:  8    Correlation:  0.456906198563985"
[1] "Nodes:  9    Correlation:  0.491657826562044"
[1] "Nodes:  10    Correlation:  0.488366333987987"
```

6 Hyperbolic Tangent

```
[1] "Nodes:  1    Correlation:  0.41514814289465"
[1] "Nodes:  2    Correlation:  0.412810481996589"
[1] "Nodes:  3    Correlation:  0.417363500631434"
[1] "Nodes:  4    Correlation:  0.436032112898535"
[1] "Nodes:  5    Correlation:  0.496000700637085"
[1] "Nodes:  6    Correlation:  0.5044700014599014"
[1] "Nodes:  7    Correlation:  0.45473188702883"
[1] "Nodes:  8    Correlation:  0.482887425391005"
[1] "Nodes:  9    Correlation:  0.464320076663667"
[1] "Nodes:  10    Correlation:  0.504431483802667"
```

7 Logistic

Based on the results, choosing 5 nodes was nearly optimal in both hyperbolic tangent and logistic activation functions.

Using a double layer neural network did not improve the results based on correlation. Once again, for loops were used to try various node combinations from (1, 1) to (5, 5).

```
# double layer neural network
for(nodes1 in c(1,2,3,4,5)){
  for(nodes2 in c(1,2,3,4,5)){
    tanmodel <- neuralnet(yr1.ret ~ yr3.ret+yr5.ret+yr7.ret+yr10.ret, data=train, hidden=c(nodes1, nodes2), act.fct="tanh"
    tan.results <- predict(tanmodel, test)
    tan.results <- data.frame(test$yr1.ret, tan.results)
    corr<- cor(tan.results$test.yr1.ret, tan.results$tan.results)
    print(paste("Nodes: ", nodes1, " ", nodes2, "   Correlation: ", corr))
  }
}

for(nodes1 in c(1,2,3,4,5)){
  for(nodes2 in c(1,2,3,4,5)){
    logmodel <- neuralnet(yr1.ret ~ yr3.ret+yr5.ret+yr7.ret+yr10.ret, data=train, hidden=c(nodes1, nodes2), act.fct="logis
    log.results <- predict(logmodel, test)
    log.results <- data.frame(test$yr1.ret, log.results)
    corr<- cor(log.results$test.yr1.ret, log.results$log.results)
    print(paste("Nodes: ", nodes1, " ", nodes2, "   Correlation: ", corr))
  }
}
```

```
[1] "Nodes:  1  1   Correlation:  0.363182862318245"
[1] "Nodes:  1  2   Correlation:  0.410177875284546"
[1] "Nodes:  1  3   Correlation:  0.40477857356055"
[1] "Nodes:  1  4   Correlation:  0.432982285198053"
[1] "Nodes:  1  5   Correlation:  0.413970099046091"
[1] "Nodes:  2  1   Correlation:  0.493123955399148"
[1] "Nodes:  2  2   Correlation:  -0.106001940053331"
[1] "Nodes:  2  3   Correlation:  0.422618637498801"
[1] "Nodes:  2  4   Correlation:  0.43534556776937"
[1] "Nodes:  2  5   Correlation:  0.443353011869988"
[1] "Nodes:  3  1   Correlation:  0.458920706122178"
[1] "Nodes:  3  2   Correlation:  0.468688324163419"
[1] "Nodes:  3  3   Correlation:  0.457124743632332"
[1] "Nodes:  3  4   Correlation:  0.414927394689759"
[1] "Nodes:  3  5   Correlation:  0.452767891705303"
[1] "Nodes:  4  1   Correlation:  0.455059904329834"
[1] "Nodes:  4  2   Correlation:  0.450160426291277"
[1] "Nodes:  4  3   Correlation:  0.533881595653818"
[1] "Nodes:  4  4   Correlation:  0.411104314442433"
[1] "Nodes:  4  5   Correlation:  0.477978105925249"
[1] "Nodes:  5  1   Correlation:  0.504971614841612"
[1] "Nodes:  5  2   Correlation:  0.52809494330571"
[1] "Nodes:  5  3   Correlation:  0.520744724171833"
[1] "Nodes:  5  4   Correlation:  0.401864064068097"
[1] "Nodes:  5  5   Correlation:  0.503630713353609"
```

8 Hyperbolic Tangent

```
[1] "Nodes:  1   1    Correlation:  0.368197224242469"
[1] "Nodes:  1   2    Correlation:  0.354299197094135"
[1] "Nodes:  1   3    Correlation:  0.362114401652438"
[1] "Nodes:  1   4    Correlation:  0.365194391303815"
[1] "Nodes:  1   5    Correlation:  0.36245521883763"
[1] "Nodes:  2   1    Correlation:  0.36653592793946"
[1] "Nodes:  2   2    Correlation:  0.427556073147982"
[1] "Nodes:  2   3    Correlation:  0.218225897400893"
[1] "Nodes:  2   4    Correlation:  0.476720343809504"
[1] "Nodes:  2   5    Correlation:  0.411670743876319"
[1] "Nodes:  3   1    Correlation:  0.461290720887539"
[1] "Nodes:  3   2    Correlation:  0.403356649184303"
[1] "Nodes:  3   3    Correlation:  0.437730409270529"
[1] "Nodes:  3   4    Correlation:  0.470340020962022"
[1] "Nodes:  3   5    Correlation:  0.474834239177045"
[1] "Nodes:  4   1    Correlation:  0.462188199988033"
[1] "Nodes:  4   2    Correlation:  0.49557859141619"
[1] "Nodes:  4   3    Correlation:  0.43058460089142"
[1] "Nodes:  4   4    Correlation:  0.3767886686182084"
[1] "Nodes:  4   5    Correlation:  0.416383166991639"
[1] "Nodes:  5   1    Correlation:  0.474387093190532"
[1] "Nodes:  5   2    Correlation:  0.526292629820773"
[1] "Nodes:  5   3    Correlation:  0.415949556927619"
[1] "Nodes:  5   4    Correlation:  0.469424035738547"
[1] "Nodes:  5   5    Correlation:  0.494102616751572"
```

9 Logistic

Calibrating Weights

The method employed by Grant's company to choose sleeves is based on a performance score, which is calculated by summing the returns of each time frame, multiplied by an arbitrarily chosen weight—ranging from 2 for 1yr returns to 1 for 10yr returns. Using for loops to assign different weight combinations, I was able to assess the effectiveness of various weight calibrations using 1yr returns as a performance indicator.

2.1 1.9 1.7 1.5 1.3 1.1 … 0.1

	description						values
9100	wt3:	0.1	wt5:	2.1	wt7:	0.1	wt10: 0.1 0.2727538
9200	wt3:	0.1	wt5:	1.9	wt7:	0.1	wt10: 0.1 0.2719661
9300	wt3:	0.1	wt5:	1.7	wt7:	0.1	wt10: 0.1 0.2710195
9400	wt3:	0.1	wt5:	1.5	wt7:	0.1	wt10: 0.1 0.2698602
8100	wt3:	0.3	wt5:	2.1	wt7:	0.1	wt10: 0.1 0.2695132
9099	wt3:	0.1	wt5:	2.1	wt7:	0.1	wt10: 0.3 0.2691610
8200	wt3:	0.3	wt5:	1.9	wt7:	0.1	wt10: 0.1 0.2685254
9500	wt3:	0.1	wt5:	1.3	wt7:	0.1	wt10: 0.1 0.2684078
9199	wt3:	0.1	wt5:	1.9	wt7:	0.1	wt10: 0.3 0.2681135
8300	wt3:	0.3	wt5:	1.7	wt7:	0.1	wt10: 0.1 0.2673578
9299	wt3:	0.1	wt5:	1.7	wt7:	0.1	wt10: 0.3 0.2668691
7100	wt3:	0.5	wt5:	2.1	wt7:	0.1	wt10: 0.1 0.2667113
8099	wt3:	0.3	wt5:	2.1	wt7:	0.1	wt10: 0.3 0.2664326
8400	wt3:	0.3	wt5:	1.5	wt7:	0.1	wt10: 0.1 0.2659564
9098	wt3:	0.1	wt5:	2.1	wt7:	0.1	wt10: 0.5 0.2659403
7200	wt3:	0.5	wt5:	1.9	wt7:	0.1	wt10: 0.1 0.2655863

Interestingly, the highest correlations between performance scores and 1yr returns were associated with increasing weights to the 5yr returns and decreasing weights on all other time frames. This is likely due to the cyclical nature of markets. Last year there was a huge spike in returns and that number diminishes as the time frame becomes extended. Probably the 5yr returns are most similar to 1yr returns on average. While this may simply be a matter of correlations, it raises the question of whether or not the weights should be scaled as they were.

```
> cor(invest$yr1.ret, invest$yr5.ret)
[1] 0.28134
> cor(invest$yr1.ret, invest$yr3.ret)
[1] 0.2306472
> cor(invest$yr1.ret, invest$yr10.ret)
[1] 0.2098223
```

Conclusion

This project demonstrated how neural networks can be used to make predictions on returns of financial investments. The scope of the project, though limited, showed that neural networks could outperform linear models in making predictions on financial returns. In order to build stronger models and to perform more meaningful tasks such as portfolio optimization, more data is necessary. For

example, data containing decades of annual returns would allow for much stronger back testing, as it would embody the grand cycles of generational market trends. Furthermore, it would allow for portfolio optimization and predictive models such as the ADX indicator. Such models could help Grant and his company go beyond addressing insufficiencies in their current process and make effective decisions based on data.