Report for MF850 Final Project

Project Participants: Yaofei Li, Liwen Xie, Yutong Wei, Jingtian Zhang

In this project, our first objective is to forecast monthly stock return of a company based on the fundamental data of the company and characteristics of the market; the second objective is to construct a stock trend (up or down) predictor based on the same data set. Noticing the industry names for each company, we decided to categorize companies into sectors based on GICS (Global Industry Classification Standard) and according to their principal business activities. After classification, we split the data into 11 sectors of business.

1. Linear Models - Lasso and Ridge:

We classified companies into 11 sectors based on GICS, such that companies in the same sector have similar performance. First of all, we did linear regression for each single sector to test the viability of linear regression.

Based on the lecture, we used Lasso and Ridge to do variable selection and linear regression, and this is because Lasso and Ridge are the most comprehensive types for linear regression. If these two methods would not work for this dataset, we may infer that all linear regression methods are not feasible. From lecture slides, Lasso can be defined as $\min \sum_{i=1}^{n}(y_i - \alpha^T X_i)^2 + \lambda \sum_{j=1}^{d} |\alpha_j|$, and Ridge can be expressed as $\min \sum_{i=1}^{n}(y_i - \alpha^T X_i)^2 + \lambda \sum_{j=1}^{d} \alpha_j^2$. That is to say, Lasso computes $\hat{\alpha}$ by solving constrained problem $\min \sum_{i=1}^{n}(y_i - \alpha^T X_i)^2$ subject to $\sum_{j=1}^{d} |\alpha_j|$, whereas, Ridge calculates $\hat{\alpha}$ via solving $\min \sum_{i=1}^{n}(y_i - \alpha^T X_i)^2$ subject to $\sum_{j=1}^{d} \alpha_j^2$. To do this, we identify the optimal λ by k-fold cross-validation. Then we use this optimal λ in our further stage of regression.

After running R code of Lasso and Ridge, we have the list of out-of-sample test R squares given as the following:

```
[1] "---------- 1 Consumer Discretionary ----------"
[1] "Optimal Lambda for Lasso:  0.000293549482139983"
[1] "Lasso Test R2:  0.0166910209647872"
[1] "Optimal Lambda for Ridge:  0.00489670050719301"
[1] "Ridge Test R2:  0.0143508459446068"
[1] "---------- 2 Industrials ----------"
[1] "Optimal Lambda for Lasso:  0.00091599254165631"
[1] "Lasso Test R2:  0.000375560175839344"
```

[1] "Optimal Lambda for Ridge: 0.0512114044275262"
[1] "Ridge Test R2: 0.000235658064559278"
[1] "---------- 3 Materials ----------"
[1] "Optimal Lambda for Lasso: 0.0005596655298194"
[1] "Lasso Test R2: 0.0255633532010187"
[1] "Optimal Lambda for Ridge: 0.138633425298098"
[1] "Ridge Test R2: 0.0106746858680688"
[1] "---------- 4 Financials ----------"
[1] "Optimal Lambda for Lasso: 0.00143754247602349"
[1] "Lasso Test R2: 0.0430705753567993"
[1] "Optimal Lambda for Ridge: 0.0553961105282914"
[1] "Ridge Test R2: 0.00219725937605221"
[1] "---------- 5 Consumer Staples ----------"
[1] "Optimal Lambda for Lasso: 0.00239102992701775"
[1] "Lasso Test R2: 0.015520984068461"
[1] "Optimal Lambda for Ridge: 0.233606005195695"
[1] "Ridge Test R2: 0.00764406114174614"
[1] "---------- 6 Health Care ----------"
[1] "Optimal Lambda for Lasso: 0.000324223692350467"
[1] "Lasso Test R2: 0.0206266085948029"
[1] "Optimal Lambda for Ridge: 0.00492791185280789"
[1] "Ridge Test R2: 0.0170264381725872"
[1] "---------- 7 Energy ----------"
[1] "Optimal Lambda for Lasso: 0.00027861037601604"
[1] "Lasso Test R2: 0.000251151920204791"
[1] "Optimal Lambda for Ridge: 0.00201179232499085"
[1] "Ridge Test R2: 8.39685826870649e-06"
[1] "---------- 8 Information Technology ----------"
[1] "Optimal Lambda for Lasso: 0.000501379552798645"
[1] "Lasso Test R2: 0.0118034577134259"
[1] "Optimal Lambda for Ridge: 0.010073883479603"
[1] "Ridge Test R2: 0.0132796832045328"
[1] "---------- 9 Real Estate ----------"
[1] "Optimal Lambda for Lasso: 0.000402884804489608"
[1] "Lasso Test R2: 0.0463507101902958"
[1] "Optimal Lambda for Ridge: 0.95261685192506"
[1] "Ridge Test R2: 0.0148753839149304"
[1] "---------- 10 Utilities ----------"
[1] "Optimal Lambda for Lasso: 0.00516528372351294"
[1] "Lasso Test R2: 0.00894438725743496"
[1] "Optimal Lambda for Ridge: 0.0785077200857387"
[1] "Ridge Test R2: 0.0240681854698092"
[1] "---------- 11 Communication Services ----------"
[1] "Optimal Lambda for Lasso: 0.00426680947834005"

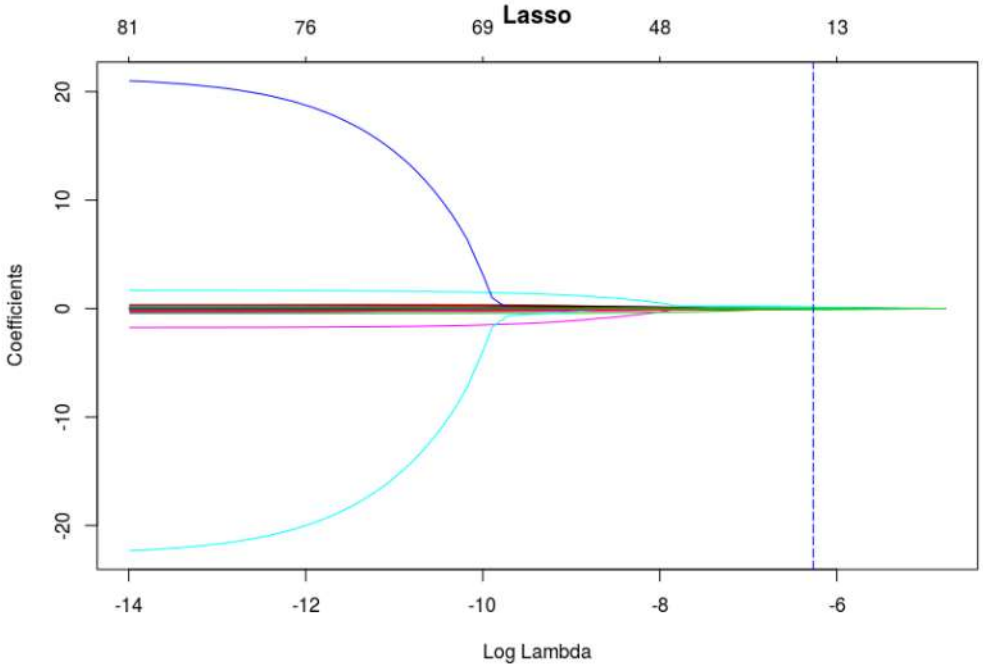
[1] "Lasso Test R2: 0.0233978937166187"
[1] "Optimal Lambda for Ridge: 0.149815862185489"
[1] "Ridge Test R2: 0.00744782078578271"

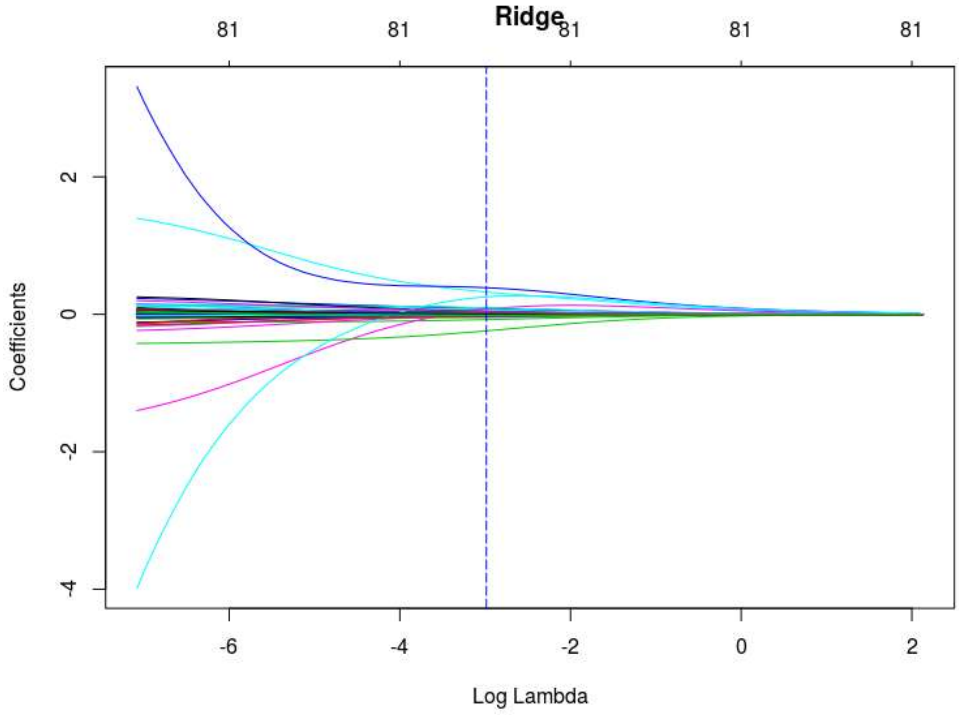
By construction, Ridge is supposed to maintain more variables than Lasso. However, from our output, Ridge somehow kept all variables in the sample dataset. In fact, we observe that none of the input parameter has a dominant effect on the output or considerably significant than the others. In this case, Ridge model is incapable to reject any of these input parameters. Based on this, we exclude directly Ridge model from our model candidate list. Based on the Lasso results we got above, it is obvious that financial sector has the best R square (0.0430705753567993) and industrial sector has the lowest R square (0.000375560175839344). Subsequently, we fitted the linear model of these 2 sectors based on Lasso and Ridge to obtain a general impression on the performance for the best and worst cases, and then have following results (note: since Ridge would not be considered, we excluded its linear model summary from this report but kept its graphs):

- Financials:

```
Call:
lm(formula = r_month_train ~ ., data = train[, c("r_month_train",
variable_select)])

Residuals:
Min          1Q      Median          3Q         Max
-0.33210 -0.04556   0.00103   0.04717   0.35929

Coefficients:
Estimate Std. Error t value Pr(>|t|)
(Intercept)          0.003235   0.001798   1.800   0.07205 .
sentiment_bearish    0.030725   0.006595   4.659 3.35e-06 ***
Close                0.034525   0.033285   1.037   0.29972
Adj_Close            0.031121   0.037433   0.831   0.40584
currentratio        -0.021354   0.020109  -1.062   0.28837
divyield            -0.112865   0.036237  -3.115   0.00186 **
ebitdamargin         0.306337   0.392320   0.781   0.43497
ebt                  0.025762   0.070159   0.367   0.71350
epsusd              -0.081602   0.055692  -1.465   0.14298
evebitda             0.135207   0.055292   2.445   0.01454 *
fcfps                0.100396   0.048452   2.072   0.03836 *
grossmargin          0.108263   0.052432   2.065   0.03904 *
pb                   0.157778   0.085812   1.839   0.06608 .
rnd                  0.027735   0.028014   0.990   0.32224
shareswadil          0.002887   0.048294   0.060   0.95233
sps                  0.036010   0.021472   1.677   0.09365 .
```

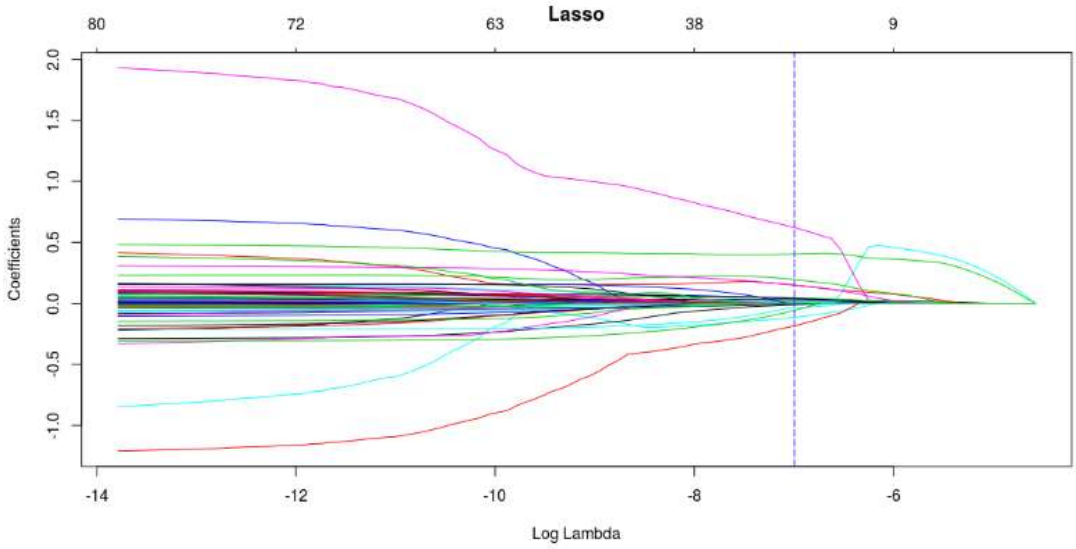
```
tbvps                -0.084466   0.026949   -3.134   0.00174 **
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 0.08918 on 2570 degrees of freedom
Multiple R-squared:  0.03631,   Adjusted R-squared:  0.03032
F-statistic: 6.053 on 16 and 2570 DF,  p-value: 2.586e-13
```

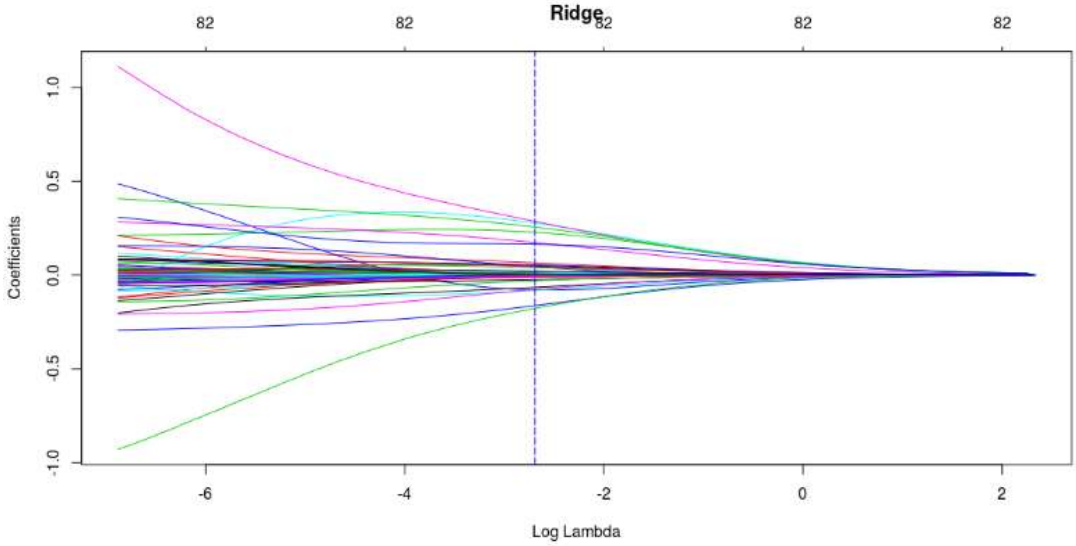




- Industrials:

```
Call:
lm(formula = r_month_train ~ ., data = train[, c("r_month_train",
variable_select)])

Residuals:
Min          1Q      Median          3Q         Max
-0.41079 -0.05931 -0.00007    0.05845    0.40871

Coefficients:
Estimate Std. Error t value Pr(>|t|)
(Intercept)             0.002336     0.001330     1.757    0.07903 .
retmonth_spx            0.034351     0.006301     5.452 5.12e-08 ***
sentiment_bullish -0.002094     0.007056    -0.297    0.76666
sentiment_bearish    0.015389     0.005723     2.689    0.00718 **
Adj_Close               0.850233     0.143851     5.911 3.54e-09 ***
Adj_Volume              0.236179     0.053144     4.444 8.94e-06 ***
capex                   0.014181     0.035428     0.400    0.68896
de                      0.209140     0.104231     2.006    0.04483 *
divyield                0.064190     0.048961     1.311    0.18988
dps                     0.070883     0.069371     1.022    0.30691
ebitdamargin         -0.200040     0.089898    -2.225    0.02609 *
epsusd                  0.381419     0.137115     2.782    0.00542 **
evebitda             -0.036494     0.053190    -0.686    0.49266
fcfps                   0.283070     0.165284     1.713    0.08682 .
grossmargin             0.231800     0.115229     2.012    0.04429 *
investments          -0.013081     0.018888    -0.693    0.48860
marketcap               -0.048132     0.024255    -1.984    0.04724 *
pb                      -0.233742     0.101802    -2.296    0.02170 *
pe1                     0.066082     0.040010     1.652    0.09865 .
prefdivis            -0.271216     0.254129    -1.067    0.28590
tbvps                   -0.390472     0.214253    -1.822    0.06842 .
---
Signif. codes:    0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.1103 on 8451 degrees of freedom
Multiple R-squared:    0.02314,    Adjusted R-squared:    0.02083
F-statistic: 10.01 on 20 and 8451 DF,    p-value: < 2.2e-16
```

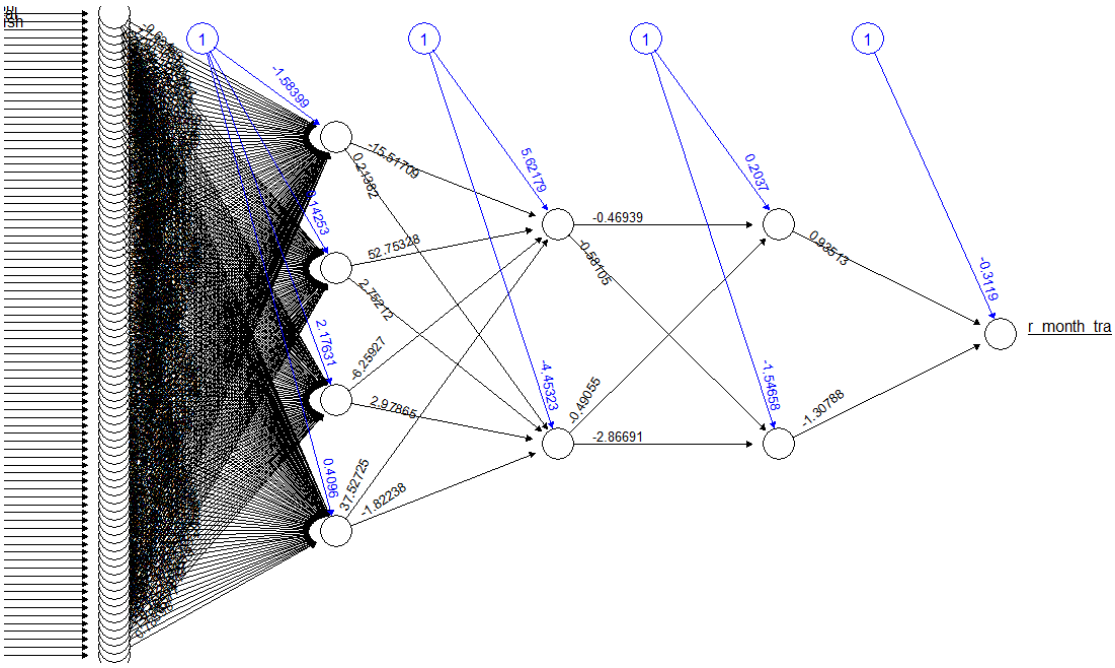




In general speaking, from results above, it is clear that even the financial sector with the best out-of-sample test R square has pretty unideal performance from Lasso regression due to its low R square value. In this stage, we can assume that there is a high non-linearity in our dataset and it comes naturally the idea of applying non-linear model. Our first attempt is neural network.

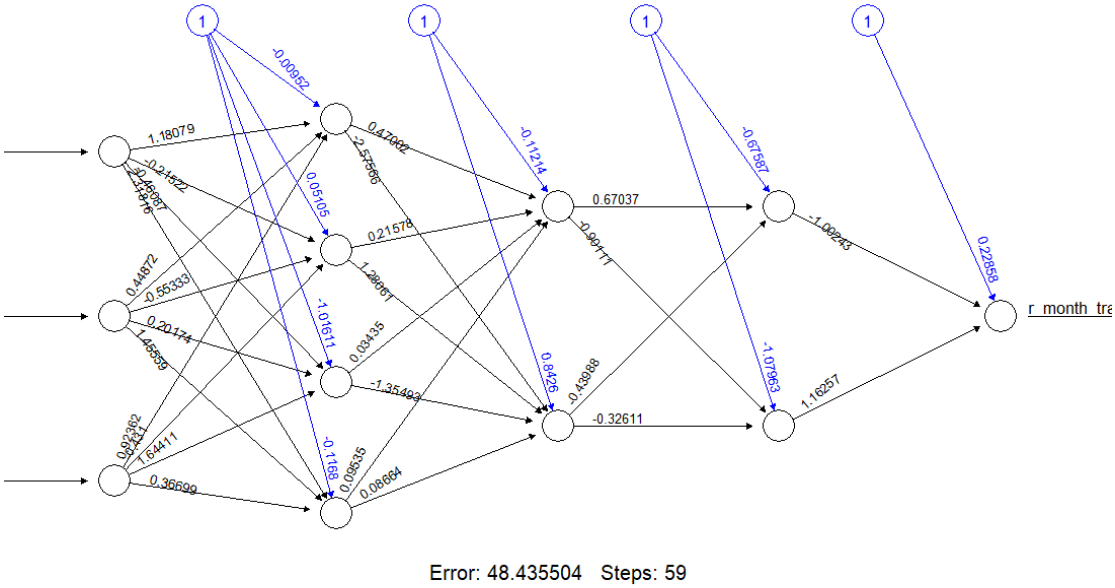
2. Neural Network:

We firstly applied neural network method on all of our sample data by inputting total 85 parameters (with RETMONTH and Date excluded). Neural network's non-linearity mainly comes from hidden layer. As we observed high non-linearity in our data, we chose to apply highest depth we can get from the package (neuralnet) we used. Without optimization, we randomly construct 3 hidden layers. then we get the following plot:



Out-of-Sample R square from Neural network method is even lower than Lasso model for each sector. From our search, R only has packages that constructs at most 3 hidden layers, which is far smaller than our input parameters size. This might be the reason why R square being too low in our case. Hence, dimensionality reduction appears inevitable before any further study.

PCA computes a limited number of principal components to explain a very large fraction of the sample variation. In detail, we set a threshold as 0.99. By PCA, we chose only few eigenvectors with corresponding eigenvalues whose sum exceeds 99% of total sum. We got the following result after implementing PCA method.



Error: 48.435504 Steps: 59

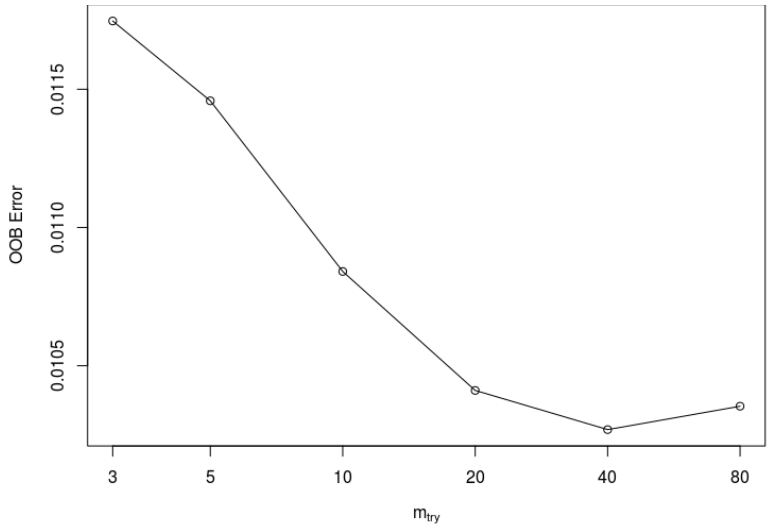
After the dimensionality reduction by PCA, we then train again the Neural Network model. In this case, the out-of-sample test R square is 0.00128484001444595, albeit larger than

original neural network result, still too small to validate this method. Thus, we abandon neural network method and have to turn towards other possibly more suitable method.

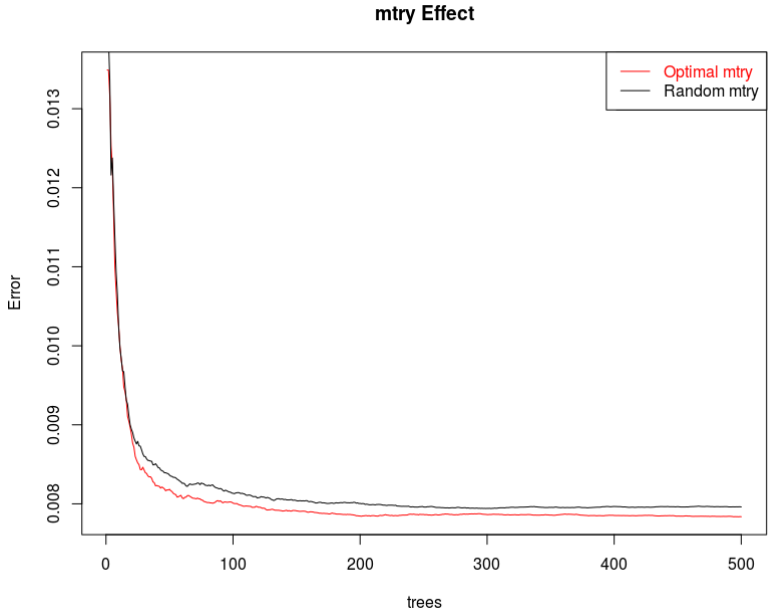
3. Random Forest:

After previous attempts, we found out that even in the same sector where business activities are very similar, companies perform differently. Hence, we need further classification within each sector. This brought us to the method of Random Forest. We repeat the decision tree method on each sector's data in order to narrow the prediction variance; each time, the decision tree randomly splits the data until the approximation error is minimized. However, as the split variables and the split points are selected to maximize information entropy, there will be overlapping or similar split regions and split orders from one decision tree to another. Similarity among the trees would be an issue, because in this case, random forest will not be able to reduce prediction error. To reduce the similarity among the trees as much, we set an upper bound for the number of randomly chosen variables in each split within one decision tree. We note this upper bound as a variable named "mtry". By this, when the training algorithm identifies the optimal split by referring to information entropy, the split variables are actually sampled from different pools. Thus, the decision tree is forced to diversify from one to another.

We firstly trained the variable "mtry" in the "Financials" sector, and found the results given below. The optimal "mtry" appears to be 40 here and the similar value is also obtained in other sectors as well.



For a clearer illustration, we then compare the cases with optimal and random "mtry", the comparison result is given as following. As we can see, the optimal "mtry" leadd to a sharper decrease of out-of-bag error convergent to a smaller asymptotic value.



Besides, percent variance explained is 4.41 by using optimal "mtry", whereas the same percent variance explained is 3.08 for the model with randomly chosen "mtry".

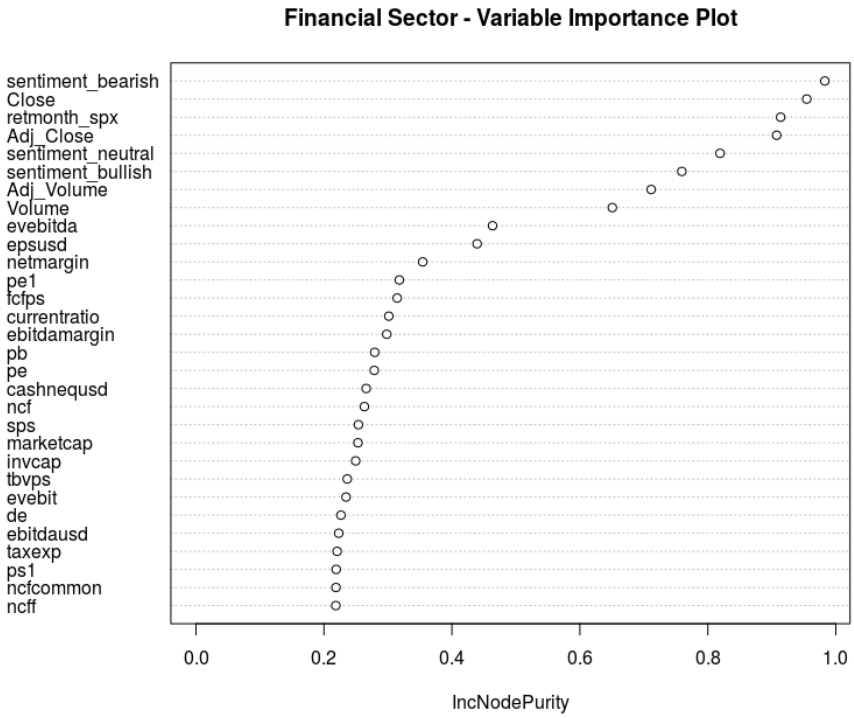
- Result for model with randomly chosen "mtry":

  Call:
  randomForest(formula = r_month_train ~ ., data = traindata, importane = T, proximity = T)
  Type of random forest: regression
  Number of trees: 500
  No. of variables tried at each split: 27
  Mean of squared residuals: 0.007945673
  % Var explained: 3.08

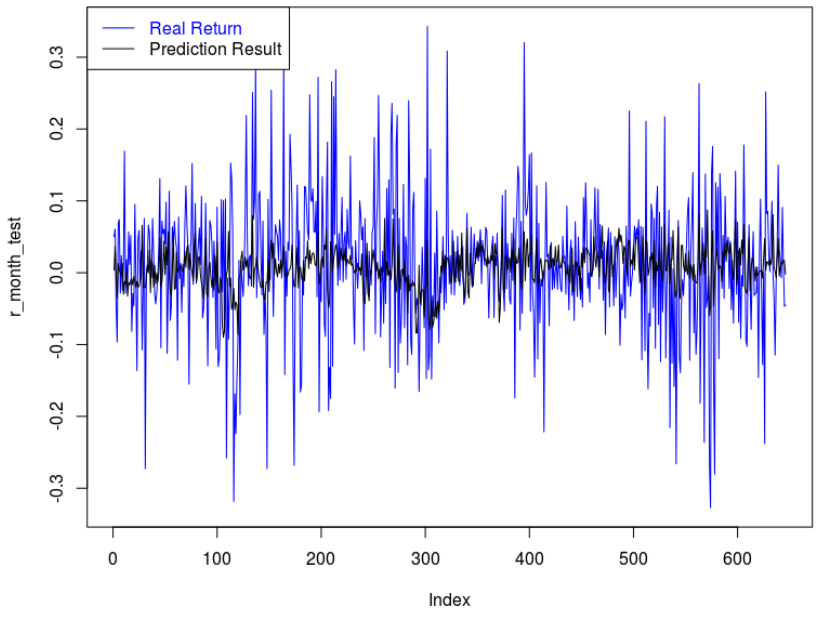
- Result for model with optimal "mtry":

  Call:
  randomForest(formula = r_month_train ~ ., data = traindata, importane T, proximity = T, do.trace = 50, ntree = 500, mtry = 40)
  Type of random forest: regression
  Number of trees: 500
  No. of variables tried at each split: 40
  Mean of squared residuals: 0.007837141
  % Var explained: 4.41

In this case, the variable importance in our random forest model is given as following.



When plotting the real returns from the test sample and the prediction results given by random forest model, we realize that the trend prediction is generally well and the error comes from the fact that the model cannot replicate the real magnitude of variation. An example in the "Financials" sector is attached below. Motivated by this observation, we simply use the same random forest model to realize the trend prediction for the part (iii). The idea is absolutely straight: whenever the predicted return is positive, we assume there is an uptrend for the stock price and vice versa. By this method, we are capable to correctly predict 64% of the trend in the test sample.



All the analysis above is restricted to the “Financials” sector as a sample study. However, we find similar results through all sectors and they are given below.

[1] "-------- Consumer Discretionary --------"
[1] "Train R2: 0.85440522318204"
[1] "Test R2: 0.10964058282784"
[1] "Test MSE: 0.0100069332867733"
[1] "% Correct Trend Prediction: 61.914984972091"
[1] "-------- Industrials --------"
[1] "Train R2: 0.854922569738804"
[1] "Test R2: 0.18887164819673"
[1] "Test MSE: 0.00964998975294687"
[1] "% Correct Trend Prediction: 66.225791213982"
[1] "-------- Materials --------"
[1] "Train R2: 0.86546877696296"
[1] "Test R2: 0.2522977598167"
[1] "Test MSE: 0.0089764265165706 7"
[1] "% Correct Trend Prediction: 66.7752442996743"
[1] "-------- Financials --------"
[1] "Train R2: 0.835262823227975"
[1] "Test R2: 0.122712620166776"
[1] "Test MSE: 0.00786443625748004"
[1] "% Correct Trend Prediction: 64.7058823529412"
[1] "-------- Consumer Staples --------"
[1] "Train R2: 0.816947073751948"
[1] "Test R2: 0.0478591661764394"
[1] "Test MSE: 0.00611851994758686"
[1] "% Correct Trend Prediction: 58.1436077057793"
[1] "-------- Health Care --------"
[1] "Train R2: 0.865923543981041"
[1] "Test R2: 0.219936673705639"
[1] "Test MSE: 0.0173877741738063"
[1] "% Correct Trend Prediction: 66.4025356576862"
[1] "-------- Energy --------"
[1] "Train R2: 0.892438440162945"
[1] "Test R2: 0.298091645028033"
[1] "Test MSE: 0.0229216337987222"
[1] "% Correct Trend Prediction: 72"
[1] "-------- Information Technology --------"
[1] "Train R2: 0.85267754350871"
[1] "Test R2: 0.1306849244411"
[1] "Test MSE: 0.010125527043738"
[1] "% Correct Trend Prediction: 64.0017101325353"
[1] "-------- Real Estate --------"

[1] "Train R2: 0.80435621427366"
[1] "Test R2: 0.030020874163906"
[1] "Test MSE: 0.00772905063141717"
[1] "% Correct Trend Prediction: 60.7843137254902"
[1] "-------- Utilities --------"
[1] "Train R2: 0.870873898556492"
[1] "Test R2: 0.0778962069701287"
[1] "Test MSE: 0.00681989203596511"
[1] "% Correct Trend Prediction: 73.2876712328767"
[1] "-------- Communication Services --------"
[1] "Train R2: 0.809849373427771"
[1] "Test R2: 0.100532327417118"
[1] "Test MSE: 0.00578033179381805"
[1] "% Correct Trend Prediction: 63.6363636363636"

Despite the prediction R square varies across different sectors, the average level is significantly better than the results given by previous models. Even the worst case, the "Real Estate", have a R square of 0.03 still larger than that in Lasso model.

4. Instruction

The sector classification is recorded in the file "GICS.csv". Please put this file in the same folder of our R script. There are total 12 .rds files, including 11 classified sectors and 1 sector called 'other' in case there are unclassified industries in the test data. Each .rds file contains trained model in that sector. When running the model, please put all .rds files and .r file in one folder.