A3 - Siddharth Iyer

Siddharth Iyer

October 15, 2021

QF301. Homework #3.

2021-10-16

I pledge on my honor that I have not given or received any unauthorized assistance on this assignment/examination. I further pledge that I have not copied any material from a book, article, the Internet or any other source except where I have expressly cited the source.

By filling out the following fields, you are signing this pledge. No assignment will get credit without being pledged.

Name: Siddharth Iyer

CWID: 10447455

Date: 10/13/2021

Instructions

In this assignment, you should use R markdown to answer the questions below. Simply type your R code into embedded chunks as shown above. When you have completed the assignment, knit the document into a PDF file, and upload both the .pdf and .Rmd files to Canvas.

```r
CWID = 10447455 #Place here your Campus wide ID number, this will personalize
#your results, but still maintain the reproducible nature of using seeds.
#If you ever need to reset the seed in this assignment, use this as your seed
#Papers that use -1 as this CWID variable will earn 0's so make sure you change
#this value before you submit your work.
personal = CWID %% 10000
set.seed(personal)#You can reset the seed at any time in your code,
#but please always set it to this seed.
```

Question 1 (20pt)

Question 1.1

Use the quantmod package to obtain the daily adjusted close prices 2 different stocks. You should have at least two years of data for both assets. You should inspect the dates for your data to make sure you are including everything appropriately. Create a data frame of the daily log returns both both stocks along with the lagged returns (2 lags). You may wish to remove the date from your data frame for later analysis. Print the first 6 lines of your data frame. (You may use the same two stocks as in Homework 2.)

Solution:

```
library(quantmod)
stocks = c("AAPL", "AMD")
getSymbols(stocks, from="2018-01-01", to="2021-10-14")
```

```
## [1] "AAPL" "AMD"
```

```
AAPL = AAPL$AAPL.Adjusted
AMD = AMD$AMD.Adjusted

aapl_log_rets = dailyReturn(AAPL, type="log")[-1]
amd_log_rets = dailyReturn(AMD, type="log")[-1]

aapl_l0 = aapl_log_rets[-2:-1]
aapl_l1 = as.numeric(stats::lag(aapl_log_rets,k=1))[-2:-1]
aapl_l2 = as.numeric(stats::lag(aapl_log_rets,k=2))[-2:-1]
amd_l0 = amd_log_rets[-2:-1]
amd_l1 = as.numeric(stats::lag(amd_log_rets,k=1))[-2:-1]
amd_l2 = as.numeric(stats::lag(amd_log_rets,k=2))[-2:-1]

lagged_df = data.frame(aapl_l0, aapl_l1, aapl_l2, amd_l0, amd_l1, amd_l2)
rownames(lagged_df) <- NULL
colnames(lagged_df) <- c("aapl_l0", "aapl_l1", "aapl_l2", "amd_l0", "amd_l1", "amd_l2")
head(lagged_df)
```

```
##         aapl_l0       aapl_l1       aapl_l2       amd_l0      amd_l1
## 1  0.0113209454  0.0046341996 -0.0001740830 -0.020000667  0.04817154
## 2 -0.0037210848  0.0113209454  0.0046341996  0.033115609 -0.02000067
## 3 -0.0001146687 -0.0037210848  0.0113209454 -0.038178911  0.03311561
## 4 -0.0002296884 -0.0001146687 -0.0037210848  0.011774737 -0.03817891
## 5  0.0056641630 -0.0002296884 -0.0001146687  0.014938037  0.01177474
## 6  0.0102733820  0.0056641630 -0.0002296884 -0.009933857  0.01493804
##        amd_l2
## 1  0.05061000
## 2  0.04817154
## 3 -0.02000067
## 4  0.03311561
## 5 -0.03817891
## 6  0.01177474
```

Question 1.2

Split your data into training and testing sets (80% training and 20% test).

Linearly regress one of your stock returns as a function of the lagged returns (2 lags) for both stocks. This should be of the form $r_{1,t} = \beta_0 + \beta_{1,1}r_{1,t-1} + \beta_{1,2}r_{1,t-2} + \beta_{2,1}r_{2,t-1} + \beta_{2,2}r_{2,t-2}$. Evaluate the performance of this model with the mean squared error on the test data.

Solution:

```
# split into train and test samples
N <- nrow(lagged_df)
train = sample(N, N*.8, replace = FALSE)

# training multi-linear model
lagged.lin.reg <- lm(aapl_l0~aapl_l1+aapl_l2+amd_l1+amd_l2, lagged_df, subset=train)

# train MSE
cat("Train MSE: ", mean((lagged_df$aapl_l0[train]-predict(lagged.lin.reg, lagged_df[train,]))^2), "\n")
```

```
## Train MSE:  0.0004155334
```

```
# test MSE
cat("Test MSE: ", mean((lagged_df$aapl_l0[-train]-predict(lagged.lin.reg, lagged_df[-train,]))^2), "\n"
```

```
## Test MSE:  0.0004884005
```

Question 2 (35pt)

Question 2.1

Using the same data, train/test split, and consider the same regression problem as in Question 1.2. Create a feed-forward neural network with a single hidden layer (2 hidden nodes) densely connected to the inputs. You may choose any activation functions you wish.

Question 2.1.1

Write the mathematical structure for this neural network.

Solution:

$y = f^{(2)}(f_1^{(1)}(x_1, x_2, x_3, x_4, \beta_1^{(1)}), f_2^{(1)}(x_1, x_2, x_3, x_4, \beta_2^{(1)}), \beta^{(2)})$

Question 2.1.2

Train this neural network on the training data.
Evaluate the performance of this model with the mean squared error on the test data.

Solution:

```
df1 = data.frame(aapl_l0, aapl_l1, aapl_l2, amd_l1, amd_l2)
rownames(df1) <- NULL
colnames(df1) <- c("aapl_l0", "aapl_l1", "aapl_l2", "amd_l1", "amd_l2")
```

```
library(keras)
nn.reg = keras_model_sequential() %>% # Creating a model can sometimes take a bit of time
  layer_flatten(input_shape = c(4)) %>%
  layer_dense(units = 2, activation = "relu") %>%
  layer_dense(1 , activation = "linear")

nn.reg %>% # State the loss function and optimizer (adam is a good choice usually)
  compile(
    loss = "mean_squared_error",
    optimizer = "adam"
  )

nn.reg %>% # Fit the model to training data
  fit(
    x = as.matrix(df1[train,-1]), y = df1[train,1],
    epochs = 1000, # how long to train for
    verbose = 0
  )
```

```
y.nn.pred = predict(nn.reg, as.matrix(df1[-train, -1])) # Predict with the neural network
nn.MSE = mean((df1[-train,1] - y.nn.pred)^2)
cat("Neural Network 1 MSE: ", nn.MSE, "\n")
```

```
## Neural Network 1 MSE:  0.0005056523
```

The NN MSE is very close to the baseline MSE, so I think it's a good model and isn't overfitting.

Question 2.2

Using the same train/test split and consider the same regression problem as in Question 1.2. Train and test another neural network of your own design.

Solution:

```
nn1.reg = keras_model_sequential() %>% # Creating a model can sometimes take a bit of time
  layer_flatten(input_shape = c(4)) %>%
  layer_dense(units = 4, activation = "relu") %>%
  layer_dense(1 , activation = "linear")

nn1.reg %>% # State the loss function and optimizer (adam is a good choice usually)
  compile(
    loss = "mean_squared_error",
    optimizer = "adam"
  )

nn1.reg %>% # Fit the model to training data
  fit(
    x = as.matrix(df1[train,-1]), y = df1[train,1],
    epochs = 1000, # how long to train for
    verbose = 0
  )
```

```
y.nn1.pred = predict(nn1.reg, as.matrix(df1[-train, -1])) # Predict with the neural network
nn1.MSE = mean((df1[-train,1] - y.nn1.pred)^2)
cat("Neural Network 2 MSE: ", nn1.MSE, "\n")
```

```
## Neural Network 2 MSE:  0.0004722278
```

1 hidden layer with 4 hidden nodes.

Question 2.3

How would you determine if your models are overfitting the data? Do you suspect this is happening in your models?

Solution:

Overfitting happens when the model fits noise in the training dataset. So when applied to testing datasets, there is an increase in MSE relative to training MSE. I don't suspect this is the case for the two models above, but if we increase the epochs and noise in the data, there could be overfitting.

```
# test MSE
cat("Linear Model Test MSE: ", mean((lagged_df$aapl_l0[-train]-predict(lagged.lin.reg, lagged_df[-train
```

```
## Linear Model Test MSE:  0.0004884005
```

```
cat("Neural Network 1 Train MSE: ", mean((df1[train,1] - predict(nn.reg, as.matrix(df1[train, -1])))^2)
```

```
## Neural Network 1 Train MSE:  0.0004032256
```

```
cat("Neural Network 1 Test MSE: ", nn.MSE, "\n")
```

```
## Neural Network 1 Test MSE:  0.0005056523
```

```
cat("Neural Network 2 Train MSE: ", mean((df1[train,1] - predict(nn1.reg, as.matrix(df1[train, -1])))^2
```

```
## Neural Network 2 Train MSE:  0.0004028901
```

```
cat("Neural Network 2 Test MSE: ", nn1.MSE, "\n")
```

```
## Neural Network 2 Test MSE:  0.0004722278
```

Question 3 (35pt)

Question 3.1

Using the same data, train/test split, and consider the same regression problem as in Question 1.2. Train a decision tree on the training data. Evaluate the performance of this model with the mean squared error on the test data.

Solution:

```
library(tree)
library(randomForest)

tree.reg <- tree(aapl_10~., data = df1, subset = train)
y.tree.pred = predict(tree.reg, df1[-train,])
cat("Tree Test MSE: ", mean((df1$aapl_10[-train] - y.tree.pred)^2), "\n")
```

```
## Tree Test MSE:  0.0005102021
```

Question 3.2

Using the same train/test split and consider the same regression problem as in Question 1.2. Train and test a random forest with 250 trees and 2 predictors.

Solution:

```
rf.reg = randomForest(aapl_10~.,data=df1,subset=train,ntree=250,mtry=2,importance=TRUE)

rf.pred = predict(rf.reg, df1[-train,])
rf.MSE = mean((df1$aapl_10[-train] - rf.pred)^2)
cat("Random Forest MSE: ", rf.MSE, "\n")
```

```
## Random Forest MSE:  0.0005094451
```

Question 3.3

How would you determine if your models are overfitting the data? Do you suspect this is happening in either the decision tree or random forest?

Solution:

```
cat("Tree Train MSE: ", mean((df1[train,1] - predict(tree.reg, df1[train, -1]))^2), "\n")
```

```
## Tree Train MSE:  0.0003468481
```

```
cat("Tree Test MSE: ", mean((df1[-train,1] - predict(tree.reg, df1[-train, -1]))^2), "\n")
```

```
## Tree Test MSE:  0.0005102021
```

```
cat("RF Train MSE: ", mean((df1[train,1] - predict(rf.reg, df1[train, -1]))^2), "\n")
```

```
## RF Train MSE:  9.556771e-05
```

```r
cat("RF Test MSE: ", mean((df1[-train,1] - predict(rf.reg, df1[-train, -1]))^2), "\n")
```

```
## RF Test MSE:  0.0005094451
```

Based on results above, I suspect Random Forest may be overfitting because the test MSE shot up 4x the train MSE. That indicates it was fitting noise in the training set.

Question 4 (10pt)

Question 4.1

Consider the same regression problem as in Question 1.2. Of the methods considered in this assignment, which would you recommend in practice? Explain briefly (1 paragraph) why you choose this fit.

Solution:

For this case, I would choose the linear model. To me, it is not apparent that the Random Forest, Tree, and Neural Network models provide a boost in MSE performance that is worth the complexity added.

The linear model provides an equally good estimate while being faster and simpler.