Final Project Report Part1

Group 24: Haijian Wu, Yanyijie Zhou, Rui Qiu

11/25/2021

Prepare data for fitting models

Calculate 10-min forward return of Asset1

```
############calculate 10-min forward return of Asset1########
h = 10 # 10 min
maxt = nrow(df)  #set maximum t

df$rf_1_10[1:(maxt-h)] = round((df[(1+h):maxt, 2] - df[1:(maxt-h), 2])/df[1:(maxt-h), 2
],4)
df$rf_1_10[(maxt-h+1):maxt] = round((df[maxt,2] - df[(maxt-h+1):maxt, 2])/df[(maxt-h+1):
maxt, 2],4)
```

Create tranining and test data set

```
# Features used for predition comes from bret.csv
new_df <- read.csv("bret.csv")
# add response variable into the data frame
new_df$rf_1_10 <- df$rf_1_10

index <- 1:floor(0.7*nrow(new_df))
training <- new_df[index,]
test <- new_df[-index,]
```

Linear model

fit linear model

```
ols <- lm(rf_1_10~., data = training)
summary(ols)
```

```
##
## Call:
## lm(formula = rf_1_10 ~ ., data = training)
##
## Residuals:
##       Min        1Q    Median        3Q       Max
## -0.147290 -0.000917  0.000010  0.000928  0.085510
##
## Coefficients:
##                   Estimate Std. Error t value Pr(>|t|)
## (Intercept)     -1.040e-05  4.758e-06  -2.186 0.028790 *
## Asset_1_BRet_3   4.068e-02  4.076e-03   9.981  < 2e-16 ***
## Asset_1_BRet_10  1.705e-02  2.537e-03   6.721 1.81e-11 ***
## Asset_1_BRet_30  6.274e-03  1.261e-03   4.977 6.48e-07 ***
## Asset_2_BRet_3   2.596e-02  2.228e-03  11.650  < 2e-16 ***
## Asset_2_BRet_10 -5.389e-03  1.435e-03  -3.756 0.000173 ***
## Asset_2_BRet_30  8.863e-03  7.609e-04  11.648  < 2e-16 ***
## Asset_3_BRet_3   1.832e-02  2.247e-03   8.155 3.50e-16 ***
## Asset_3_BRet_10  3.431e-03  1.441e-03   2.381 0.017282 *
## Asset_3_BRet_30 -9.649e-04  7.296e-04  -1.323 0.185984
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.002882 on 366902 degrees of freedom
## Multiple R-squared:  0.004603,   Adjusted R-squared:  0.004578
## F-statistic: 188.5 on 9 and 366902 DF,  p-value: < 2.2e-16
```

we can see from the above summary that all the 3-min, 10-min and 30-min backward return of Asset_1 and Asset_2 are significant in the linear model, while only the 3-min backward return of Asset_3 is highly significant ($p < 0.001$). The 10-min backward return of Asset_3 is slightly significant ($p < 0.05$) and the 30-min backward return of Asset_3 is not significant.

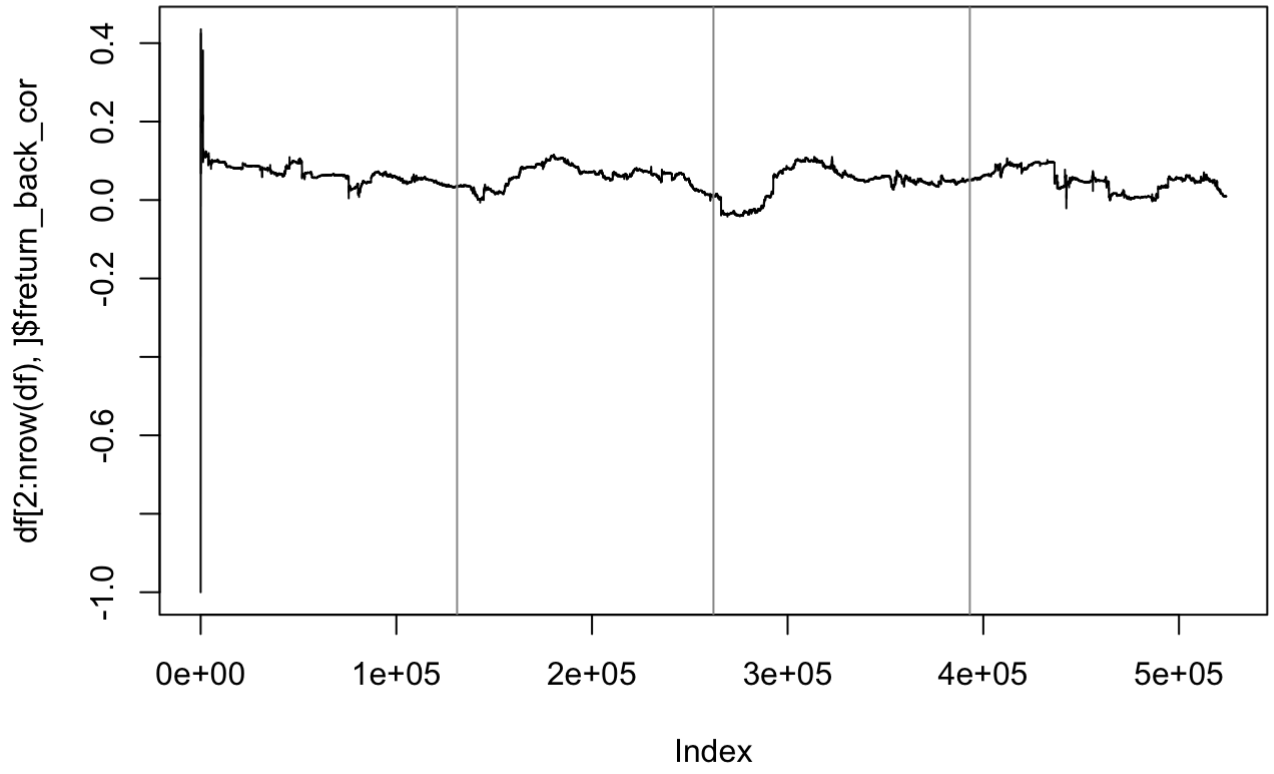
Calculate calculate correlation coefficients

```
# calculate predic value
training$pred <- predict.lm(ols,training)
test$pred <- predict.lm(ols,test)
# calculate correlation
corr_in <- cor(training$rf_1_10, training$pred)
corr_out <- cor(test$rf_1_10, test$pred)
```

Therefore the in-sample correlation of predicted 10-min forward return and real 10-min forward return is 0.0678441. The out-sample correlation of predicted 10-min forward return and real 10-min forward return is 0.0406406

3-week backward rolling cor

```
w = 21*24*60 # three week

df <- rbind(training, test)
idx <- ncol(df)

for(t in 1:length(df[,1])){
    if(t>=w+1){
        start = t-w
        end = t
        sub_1_1 <- df[start:end,idx-1]
        sub_2_1 <- df[start:end,idx]
        freturn_back_cor <- round(cor(sub_1_1,sub_2_1),4)
        df$freturn_back_cor[t] <- freturn_back_cor
    }
    else{
        start = 1
        end = t
        sub_1_2 <- df[start:end,idx - 1]
        sub_2_2 <- df[start:end,idx]
        freturn_back_cor <- round(cor(sub_1_2,sub_2_2),4)
        df$freturn_back_cor[t] <- freturn_back_cor
    }
}
plot(df[2:nrow(df),]$freturn_back_cor,type = "l")
abline(v = 91*24*60, col = "gray60")
abline(v = 2*91*24*60, col = "gray60")
abline(v = 3*91*24*60, col = "gray60")
```



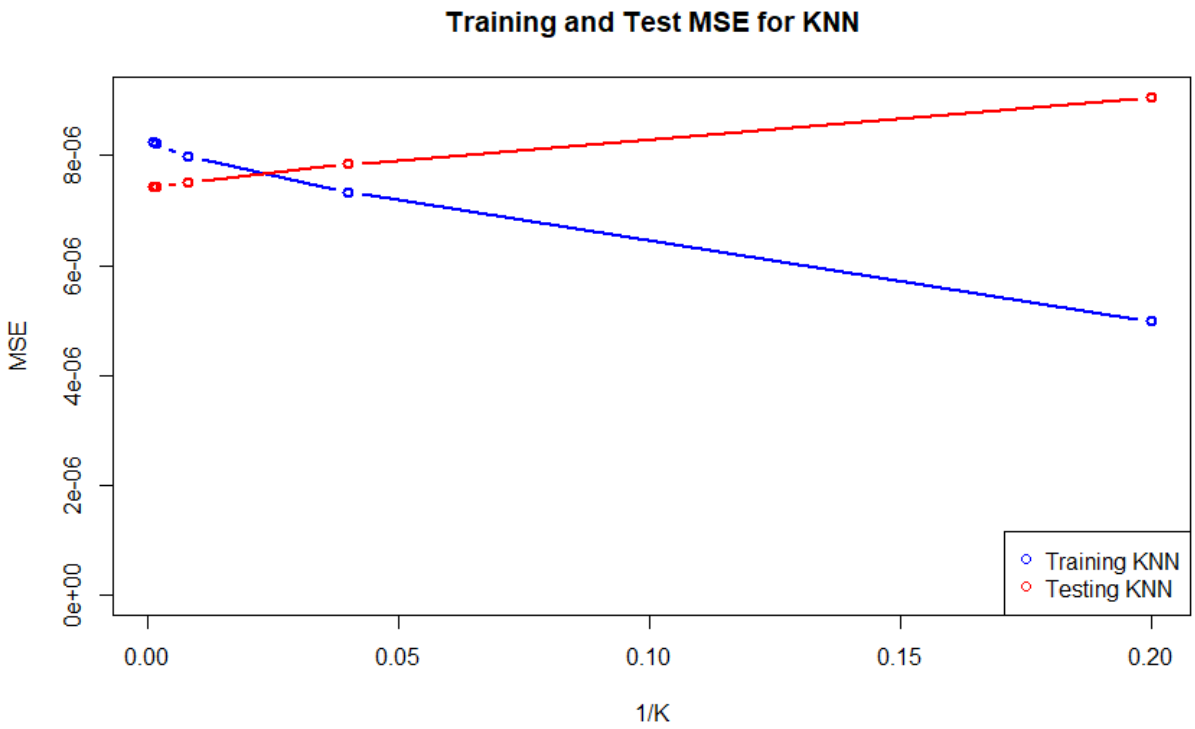
We can see from the plot that the correlation structure is not stationary over the year. The correlation coefficient fluctuates very much at the beginning of the year. In the remaining first quarter, the correlation show a decreasing trend. In the second and the third quarter, the correlation coefficients show a sudden increase from low levels followed by a slow decrease trend. In the forth quarter, the correlation coefficients appear to be particularly volatile in several period, and there is a hump in the curve at the end of the year.

Q 2.4: KNN Regression
In part 2.4, we need to use KNN to predict as what we did in part 2.3. The following is the code.

```
# We have finished calculation for forward return & splitting dataset in Part 2.3

# Fit KNN model
library("FNN")
# K range as the instruction need
k_range <- c(5, 25, 125, 625, 1000)

# get training MSE

trainMSE = c() #creating null vector for training MSE

for(i in 1:length(k_range)){
  knnTrain <- knn.reg(train = train[, -10] , y = train[,10],
                      test = train[, -10],k = k_range[i])
  trainMSE[i] <- mean((train$rf_1_10 - knnTrain$pred)^2)
}

# get testing MSE

testMSE = c() #creating null vector for testing MSE

for(i in 1:length(k_range)){
  knnTest <- knn.reg(train = train[, -10], y = train[,10],
                     test = test[,-10], k = k_range[i])
  testMSE[i] <- mean((test$rf_1_10 - knnTest$pred)^2)
}

# Plot
# Training MSE v.s. 1/K
plot(trainMSE ~ I(1/k_range), type = "b", lwd = 2, col = "blue",
     main = "Training and Test MSE for KNN", xlab = "1/K", ylab = "MSE",
     ylim = c(0,max(max(trainMSE),max(testMSE))))
# Testing MSE v.s. 1/K
lines(testMSE ~ I(1/k_range), type = "b", lwd = 2, col = "red")
# legend
legend("bottomright", legend = c("Training KNN","Testing KNN"), col=c("blue","red"), pch=1)

# We can find the best K based on validation MSE (testing MSE)
bestK = k_range[which.min(testMSE)]

# Calculate the prediction value
knn.model.test <- knn.reg(train = train[, -10] , y = train[,10],
                      test = test[, -10],k = bestK)
knn.model.train <- knn.reg(train = train[, -10] , y = train[,10],
                           test = train[, -10],k = bestK)


# Calculate the correlation between predicted values and the real values
corr_in <- cor(train$rf_1_10, knn.model.train$pred)
corr_out <- cor(test$rf_1_10, knn.model.test$pred)
```

```
# Output
corr_in
corr_out
```

The plot is as shown here. We generate it by running above codes in the form of R.script to save time. We can find that the best K based on test MSE (validation MSE) is 1000



In order to save the time (because the dataset is very large), we do not run the code in RMD. Instead, we utilize R script to get the correlation. Finally, we find that

$$\begin{pmatrix} In-Sample\ Correlation & Out-of-Sample\ Corrleation \\ 0.1183135 & 0.04317027 \end{pmatrix}$$

We can find that the out-of-sample correlation is about 4.3%.

Q 2.5: Ridge & Lasso

In part 2.5, we need to do ridge regression and Lasso to predict as what we did before. The following is our code.

Before doing the regression, we need to generate new data.

```
# Read the original data
data_asset <- read.csv("final_project_data.csv")
# Set output matrix & time vec
out_mat <- matrix(rep(0,524160 * 3*14), nrow = 524160)
# time vector includes all h we need
time_vec <- c(3, 10, 30, 60, 120,180,240,360,480,600,720,960,1200,1440)

# calculate the backward return for each dataset and different h
# as we did for Q2.1

# a denotes Asset a (from 1 - 3)
# t_idx refers to the index in time_vec
# h refers to the minutes (for backward counting)
for(a in 1:3){
  for(t_idx in 1:length(time_vec)){
    h = time_vec[t_idx]
    print(h)
    out_mat[1:time_vec[t_idx],(a - 1)*14 + t_idx] =
      (data_asset[1:time_vec[t_idx], 1+a] - data_asset[1, 1+a])/data_asset[1, 1+a]
    out_mat[(time_vec[t_idx] + 1):nrow(data_asset), (a-1)*14 + t_idx] =
      (data_asset[(time_vec[t_idx]+1):nrow(data_asset),1+a] -
               data_asset[1:(nrow(data_asset)-h), 1+a])/(data_asset[1:(nrow(data_asset)-h), 1+a])
  }
}
# store the matrix as dataframe
outmat <- as.data.frame(out_mat)
# Write
write.csv(outmat, "Q2.5input.csv", row.names = FALSE)
```

After generating the new data, we can do prediction.

```
# We have finished calculation for forward return in Part 2.3

#### Splitting the training data and testing data
# Features used for predication comes from Q2.5input csv
new_df <- read.csv("Q2.5input.csv")
# add response variable into the data frame
new_df$rf_1_10 <- df$rf_1_10
# calculate the index for train data
index <- 1:floor(0.7*nrow(new_df))

##### Fit the ridge and lasso
library(glmnet)
# get the grid: vector for lambda
grid = 10^seq(10, -10, length=200)
# Generate data matrix X and response y
X = model.matrix(rf_1_10 ~ ., new_df)[, -1]
y = new_df$rf_1_10

# Ridge regression
```

```
ridge.model = glmnet(X[index,], y[index], alpha=0, lambda=grid)

testMSE = rep(0, length(grid)) #creating null vector for testing MSE
for(i in 1:length(testMSE)){
  ridge.pred_test = predict(ridge.model, s=grid[i], newx=X[-index,])
  testMSE[i] = mean((ridge.pred_test-y[-index])^2)
}

# find the best lambda index for ridge
ridge.lambda_index = which.min(testMSE)

#Lasso
lasso.model = glmnet(X[index,], y[index], alpha=1, lambda=grid)
lassoTestMSE = rep(0, length(grid))
for(i in 1:length(lassoTestMSE)){
  lasso.pred_test = predict(lasso.model, s=grid[i], newx=X[-index,])
  lassoTestMSE[i] = mean((lasso.pred_test-y[-index])^2)
}

# find the best lambda index for lasso
lasso.lambda_index = which.min(lassoTestMSE)

# get prediction
bestRidge.predict = predict(ridge.model, s=grid[ridge.lambda_index], newx=X)
bestLasso.predict = predict(lasso.model, s=grid[lasso.lambda_index], newx=X)

# get in-sample correlation
corr.in.ridge <- cor(new_df$rf_1_10[index], bestRidge.predict[index])
corr.in.lasso <- cor(new_df$rf_1_10[index], bestLasso.predict[index])
corr.in.ridge
corr.in.lasso

# get out-of-sample correlation
corr.out.ridge <- cor(new_df$rf_1_10[-index], bestRidge.predict[-index])
corr.out.lasso <- cor(new_df$rf_1_10[-index], bestLasso.predict[-index])
corr.out.ridge
corr.out.lasso
```

In order to save the time (because the dataset is very large), we do not run the code in RMD. Instead, we utilize R script to get the correlation. Finally, we find that

$$Ridge: \begin{pmatrix} In-Sample\ Correlation & Out-of-Sample\ Corrleation \\ 0.07376389 & 0.0377913 \end{pmatrix}$$

$$Lasso: \begin{pmatrix} In-Sample\ Correlation & Out-of-Sample\ Corrleation \\ 0.06809269 & 0.03918842 \end{pmatrix}$$

We can find that the out-of-sample correlation of ridge regression is about 3.8% and that of lasso is about 3.9%;.

Question 2.6 Report

Prepare data for fitting models

Calculate 10-min forward return of Asset1

```
############calculate 10-min forward return of Asset1########
h = 10 # 10 min
maxt = nrow(df)  #set maximum t

df$rf_1_10[1:(maxt-h)] = round((df[(1+h):maxt, 2] - df[1:(maxt-h), 2])/df[1:(maxt-h), 2],4)
df$rf_1_10[(maxt-h+1):maxt] = round((df[maxt,2] - df[(maxt-h+1):maxt, 2])/df[(maxt-h+1):maxt, 2],4)
```

```
library(pls)
```

```
## Warning: ³Ì¼°ü'pls'ÊÇÓÃR°æ±¾4.1.2 À´½¨Ôìµ
```

```
##
## 载入程辑包：'pls'
```

```
## The following object is masked from 'package:stats':
##
##     loadings
```

```
library(ISLR)
# Read the original data
data_asset <- read.csv("final_project_data.csv")
# Set output matrix & time vec
out_mat <- matrix(rep(0,524160 * 3*14), nrow = 524160)
# time vector includes all h we need
time_vec <- c(3, 10, 30, 60, 120,180,240,360,480,600,720,960,1200,1440)
# time vector includes all h we need
time_vec <- c(3, 10, 30, 60, 120,180,240,360,480,600,720,960,1200,1440)
# calculate the backward return for each dataset and different h
# as we did for Q2.1
# a denotes Asset a (from 1 - 3)
# t_idx refers to the index in time_vec
# h refers to the minutes (for backward counting)
for(a in 1:3){
for(t_idx in 1:length(time_vec)){
h = time_vec[t_idx]
out_mat[1:time_vec[t_idx],(a - 1)*14 + t_idx] =
(data_asset[1:time_vec[t_idx], 1+a] - data_asset[1, 1+a])/data_asset[1, 1+a]
out_mat[(time_vec[t_idx] + 1):nrow(data_asset), (a-1)*14 + t_idx] =
(data_asset[(time_vec[t_idx]+1):nrow(data_asset),1+a] -
data_asset[1:(nrow(data_asset)-h), 1+a])/(data_asset[1:(nrow(data_asset)-h), 1+a])
}
}
# store the matrix as dataframe
outmat <- as.data.frame(out_mat)
# Set output matrix & time vec

# Write
write.csv(outmat, "Q2.6input.csv", row.names = FALSE)
```

```
#After generating the new data, we can do prediction.
# We have finished calculation for forward return in Part 2.3
#### Splitting the training data and testing data
# Features used for predication comes from Q2.5input csv
new_df <- read.csv("Q2.6input.csv")
# add response variable into the data frame
new_df$rf_1_10 <- df$rf_1_10

index <- 1:floor(0.7*nrow(new_df))
trainingpcr <- new_df[index,]
testpcr <- new_df[-index,]
set.seed(1)
PCR <- pcr(rf_1_10 ~ ., data = trainingpcr, scale = TRUE, validation = "none")
testMSEpcr <-rep(0,42)
predictPCR <- predict(PCR,new_df[-index,names(new_df)!='rf_1_10'])
for(i in 1:42){
  testMSEpcr[i]<-mean((predictPCR[,,i]-new_df[-index,43])^2)
}
min_index = which.min(testMSEpcr)
min_index
```

```
## [1] 6
```

```
#From the mse, We see that the optimal pc is 6.
#Use this value to generate prediction
pcr_train_predict <- predict(PCR, trainingpcr[names(trainingpcr)!='rf_1_10'],ncomp = 6)
pcr_test_predict <- predict(PCR, testpcr[names(testpcr)!='rf_1_10'],ncomp = 6)
#get in-sample correclation
corr_in_pcr <- cor(trainingpcr$rf_1_10, pcr_train_predict)
corr_in_pcr
```

```
## [1] 0.06697405
```

```
#get out-sample correclation
corr_out_pcr <- cor(testpcr$rf_1_10, pcr_test_predict)
corr_out_pcr
```

```
## [1] 0.04150374
```