Business Analytics: Homework 1

Qingyuan Dong (UNI: qd2145)

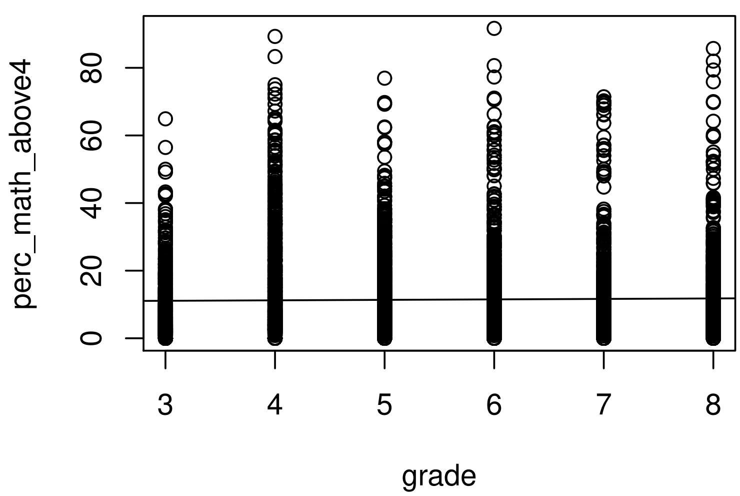
09/25/2018

1 Question 1 (Linear Regression)

(a) Perform a regression of `perc_math_above4` against the student's grade. Is `grade` best treated as a continuous or categorical variable? Why?

I performed a simple linear regression in R as follows:

```
> school <- read.csv('school_data.csv',sep=',',header=TRUE)
> sch <- school[c(2,3,12)]
> attach(sch)
> grade.lr <- lm(perc_math_above4 ~ grade)
> par(mfrow = c(1,1))
> plot(grade, perc_math_above4)
> abline(grade.lr)
```



We can tell that `grade`$\in \{3,4,5,6,7,8\}$. We best treat it as a categorical variable, because it has only 6 possible values. And we cannot see in this model what's the impact of `grade` on `perc_math_above4`.

(b) Explain the values of each of the coefficients in the regression based on the data.

This time, I treated `grade` as categorical variable and rerun the regression. I got this summary of the new regression:

```
> library(psych)
> sch = cbind(sch, dummy.code(sch$grade))
> sch <- sch[-c(1,4)]
> grade_reg = lm(perc_math_above4 ~ . - community_school, data = sch)
> summary(grade_reg)

Call:
lm(formula = perc_math_above4 ~ . - community_school, data = sch)

Residuals:
    Min      1Q  Median      3Q     Max
-15.994  -7.947  -4.150   4.219  79.625

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)   7.6389     0.4376  17.456  < 2e-16 ***
`4`           8.3549     0.6202  13.472  < 2e-16 ***
`5`           3.3290     0.6217   5.355 9.02e-08 ***
`6`           4.4024     0.6890   6.389 1.85e-10 ***
`7`           2.0962     0.6923   3.028  0.00248 **
`8`           4.1494     0.7015   5.915 3.58e-09 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘’. 0.1 ‘’  1

Residual standard error: 12.73 on 4184 degrees of freedom
Multiple R-squared:  0.04468,   Adjusted R-squared:  0.04354
F-statistic: 39.14 on 5 and 4184 DF,  p-value: < 2.2e-16
```

As we know, our regression model is:

$$perc\_math\_above4 = \beta_0 + \beta_1 grade_4 + \beta_2 grade_5 + \beta_3 grade_6 + \beta_4 grade_7 + \beta_5 grade_8$$

In this linear regression model:

- $\hat{\beta}_0 = 7.64$, which means we estimated that the mean percentage of students in Grade 3 attained level 4 in math is 7.64%
- $\hat{\beta}_1 = 8.35$, which means we estimated that the mean percentage of students in Grade 4 attained level 4 in math, compared to Grade 3 students, is 8.35% higher. In other words, the mean percentage of Grade 4 student satisfying level 4 math is $7.64\% + 8.35\% = 15.99\%$.
- $\hat{\beta}_2 = 3.33$, which means we estimated that the mean percentage of students in Grade 5 attained level 4 in math, compared to Grade 3 students, is 3.33% higher, or $7.64\% + 3.33\% = 10.97\%$.
- $\hat{\beta}_3 = 4.40$, which means we estimated that the mean percentage of students in Grade 6 attained level 4 in math, compared to Grade 3 students, is 4.40% higher, or $7.64\% + 4.40\% = 12.04\%$.
- $\hat{\beta}_4 = 2.10$, which means we estimated that the mean percentage of students in Grade 6 attained level 4 in math, compared to Grade 3 students, is 2.10% higher, or $7.64\% + 2.10\% = 9.74\%$.

- $\hat{\beta}_5 = 4.15$, which means we estimated that the mean percentage of students in Grade 6 attained level 4 in math, compared to Grade 3 students, is 4.15% higher, or $7.64\% + 4.15\% = 11.79\%$.

(c) Carefully interpret the p-values on each of the coefficients.

p-value means the probability the null hypothesis is true given the observed data. In this case, for each coefficient:

p-value for $\hat{\beta}_0$ is less than 2×10^{-16}, which means $\hat{\beta}_0$ is statistical significant with 0.01 significant level. In other words, we can say that $\beta_0 \neq 0$ with over 99% probability. So the mean percentage of Grade 3 students attaining level-4 math is positive with 99% probability.

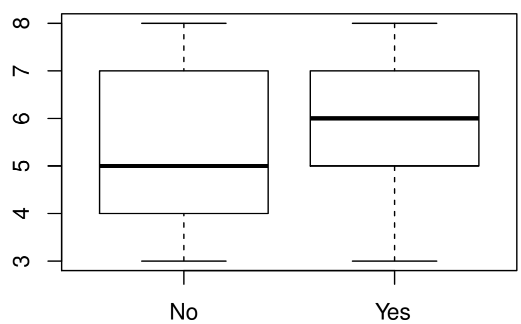
Similarly, $\hat{\beta}_1, \hat{\beta}_2, \hat{\beta}_3, \hat{\beta}_5$ are statistical significant with 0.01 significant level, while $\hat{\beta}_4$ is only significant in a 0.05 level. To interpret it, the percentages of students in Grade 4,5,6,8 are all significantly higher than Grade 3 students with 99% probability, and the same percentage of Grade 7 students is higher than Grade 3 with 95% probability.

(d) Now perform a multivariate regression against the student's grade and whether the school is a community school. Do the p-values change? Carefully explain why, using the data to support your conclusions.

```
> multi_reg = lm(perc_math_above4 ~ ., data = sch)
> summary(multi_reg)

Call:
lm(formula = perc_math_above4 ~ ., data = sch)

Residuals:
    Min      1Q  Median      3Q     Max
-16.363  -8.005  -3.419   3.861  78.682

Coefficients:
                    Estimate Std. Error t value Pr(>|t|)
(Intercept)           8.0050     0.4310  18.572  < 2e-16 ***
community_schoolYes  -9.9891     0.8132 -12.284  < 2e-16 ***
`4`                   8.3580     0.6094  13.716  < 2e-16 ***
`5`                   3.3236     0.6108   5.441 5.60e-08 ***
`6`                   4.9794     0.6786   7.337 2.60e-13 ***
`7`                   2.7060     0.6820   3.968 7.38e-05 ***
`8`                   4.8027     0.6913   6.947 4.30e-12 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' '  1

Residual standard error: 12.51 on 4183 degrees of freedom
Multiple R-squared:  0.07794,   Adjusted R-squared:  0.07662
F-statistic: 58.93 on 6 and 4183 DF,  p-value: < 2.2e-16
```

From the above result, I observed that all the p-values decreased because of adding another dummy variable `community_school`, and all the estimates becomed 3-star significant.

For the reason, I think is because that whether the school is a community school significantly impact the percentage we want to estimate. As shown in above summary, compared to private schools, the percentage of students in community schools attaining level-4 math is 3-star significantly lower. And we also observed from the data (see following figure) that the mean grade of community schools is higher than that of private schools, which indicates the two variables have a positive correlation.



Therefore, if we don't involve this variable in our model, this negative impact will be hidden in other estimates' significance (p-values). This is why when we pull this negative impact out as a new variable, all the other estimates becomes more significant (p-value decreases).

2 Question 2 (Linear Regression)

(a) Load the file into R. Print a summary of the variables.

```
> egg <- read.csv('egg_production.csv',sep=',',header=TRUE)
> summary(egg)

      eggs            feed         temperature
 Min.   :0.000   Min.   :18.36   Min.   :-12.61
 1st Qu.:1.418   1st Qu.:21.50   1st Qu.: 10.71
 Median :1.782   Median :22.27   Median : 21.76
 Mean   :1.773   Mean   :23.11   Mean   : 19.96
 3rd Qu.:2.174   3rd Qu.:23.30   3rd Qu.: 29.63
 Max.   :3.652   Max.   :32.60   Max.   : 48.12
```

(b) Run a regression of number of eggs on feed and interpret the result. Does it align with your intuition?

```
> attach(egg)
> egg_reg <- lm(eggs ~ feed)
> summary(egg_reg)

Call:
lm(formula = eggs ~ feed)

Residuals:
     Min       1Q   Median       3Q      Max
-1.54185 -0.34831 -0.02782  0.36793  1.81521
```

```
Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept)  3.832768   0.113951   33.63   <2e-16 ***
feed        -0.089108   0.004897  -18.20   <2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' '  1

Residual standard error: 0.5215 on 1550 degrees of freedom
Multiple R-squared:  0.176,     Adjusted R-squared:  0.1755
F-statistic: 331.1 on 1 and 1550 DF,  p-value: < 2.2e-16
```

The result of this regression (significantly) indicates that the more we feed the chicken, the less eggs we will get, which does not make sense according to our common sense.

(c) Now run a regression using *both* variables. Interpret the result. Does this make sense to you?

```
> egg_fullreg <- lm(eggs ~ .,data=egg)
> summary(egg_fullreg)

Call:
lm(formula = eggs ~ ., data = egg)

Residuals:
     Min       1Q   Median       3Q      Max
-1.55172 -0.34901 -0.02884  0.36528  1.81519

Coefficients:
              Estimate Std. Error t value Pr(>|t|)
(Intercept)  3.8448807  0.1160307  33.137   <2e-16 ***
feed        -0.0891043  0.0048985 -18.190   <2e-16 ***
temperature -0.0006112  0.0010969  -0.557    0.577
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' '  1

Residual standard error: 0.5216 on 1549 degrees of freedom
Multiple R-squared:  0.1762,    Adjusted R-squared:  0.1751
F-statistic: 165.6 on 2 and 1549 DF,  p-value: < 2.2e-16
```

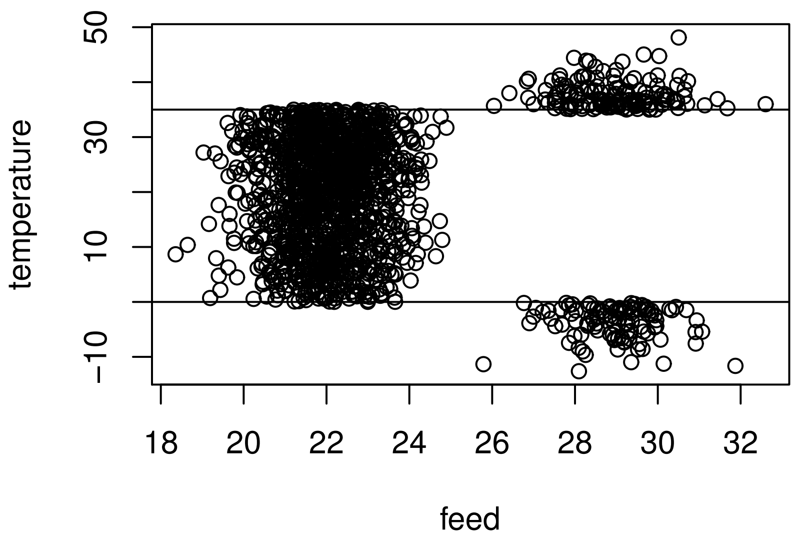
The result tells us:

1. When the temperature keeps stable and the amount of feed increases, the number of eggs will decrease with the probability of 99%
2. the impact of temperature on the number of eggs is insignificant.

The first point still makes no sense. The second conclusion also can be challenged with the argument that a too hot or too cold environment may also cause lower egg productivity.

(d) You suspect that something fishy is going on, and that the amount of feed given to each chicken depends on the temperature. Investigate this hypothesis, and create a new binary/discrete/categorical variable that captures this phenomenon.

I checked the correlation between `feed` and `temperature`, and found (as shown in the below figure) the amount of feed was clearly separated into three blocks: `temprature` < 0, $0 \leq$ `temprature`



< 35, and `temprature`≥ 35. The amount of feed in the second section ($0 \leq$ `temprature` < 35) is significantly different from the other two sections. Therefore, I created a new binary variable `tem_sec`:

$$tem\_sec = \begin{cases} 1, & \text{if } 0 < temperature < 35 \\ 0, & \text{otherwise} \end{cases}$$

```
tem_sec <- ifelse(temperature < 35 & temperature > 0, 1, 0)
```

(e) Regress number of eggs on feed, temperature, and the new variable you created. Interpret the results.

```
> egg <- cbind(egg,tem_sec)
> egg_newreg <- lm(eggs ~ .,data=egg)
> summary(egg_newreg)

Call:
lm(formula = eggs ~ ., data = egg)

Residuals:
     Min       1Q   Median       3Q      Max 
-1.56444 -0.34099 -0.00796  0.33876  1.74590 

Coefficients:
              Estimate Std. Error t value Pr(>|t|)    
(Intercept)  0.0252558  0.3658299   0.069   0.9450    
feed         0.0387500  0.0125787   3.081   0.0021 ** 
temperature -0.0007344  0.0010570  -0.695   0.4873    
tem_sec      1.0276280  0.0937132  10.966   <2e-16 ***
```

```
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘’. 0.1 ‘’  1

Residual standard error: 0.5026 on 1548 degrees of freedom
Multiple R-squared:  0.2355,    Adjusted R-squared:  0.2341
F-statistic:   159 on 3 and 1548 DF,  p-value: < 2.2e-16
```

Based on the new regression model, we can indicate from the result that:

1. Given the temperature, increasing the amount of feed will improve the number of eggs with the probability of 95%.
2. When the temperature is higher than 0 or lower than 35 (even though the feed amount is significantly lower), the egg productivity is significantly (99%) higher than that in other temperature environments.

(f) Based on all the above, what is the best model to predict egg production based on the data in this dataset?

After eliminating the ineffective variables, the best prediction model looks like this:

$$\widehat{eggs} = \hat{\beta}_0 + \hat{\beta}_1 feed + \hat{\beta}_2 tem\_sec$$

(g) For your best model, what is a 99% confidence interval for the regression coefficients. Interpret the results.

```
> confint(egg_best, level = 0.99)

                   0.5 %     99.5 %
(Intercept) -0.928997065 0.95554640
feed         0.006225925 0.07109294
tem_sec      0.785303276 1.26856920
```

This result indicates that with the probability of 99%, the true value of coefficients:

$$\beta_0 \in [-0.929, 0.956], \beta_1 \in [0.006, 0.071], \beta_2 \in [0.785, 1.269].$$

(h) For your best model, what is a 90% confidence interval for the prediction of the number of eggs that were produced if the feed was 25 and the temperature was -1. Interpret the results.

```
> predict(egg_best, data.frame(feed = 25, tem_sec = 0),
+         interval = "prediction",level=0.9)

        fit       lwr      upr
1 0.9797604 0.1469938 1.812527
```

The result indicates that when feed is 25 and temperature is -1, the number of eggs will lie in the interval $[0.147, 1.813]$ with probability of 90%.

3 Question 3 (Linear Model Selection)

(a) Open the file `ibm_return.csv` in R and use the command `summary` to print a summary of the data. Make sure the data is ready for analysis (hint: you might want to look up the `as.Date` function; specifically, try `as.Date('7/2/2010', '%m/%d/%Y')`).

```
> ibm <- read.csv('ibm_return.csv',sep=',',header=TRUE)
> ibm$Date = as.Date(ibm$Date,"%m/%d/%Y")
> summary(ibm)

      Date                Return               X1D                X3D
 Min.   :2012-07-02   Min.   :-8.280000   Min.   :-8.28000   Min.   :-3.54000
 1st Qu.:2012-09-28   1st Qu.:-0.620000   1st Qu.:-0.60000   1st Qu.:-0.31000
 Median :2012-12-31   Median :-0.060000   Median :-0.06000   Median : 0.02000
 Mean   :2012-12-30   Mean   : 0.004618   Mean   : 0.02273   Mean   : 0.02032
 3rd Qu.:2013-04-02   3rd Qu.: 0.660000   3rd Qu.: 0.67000   3rd Qu.: 0.47000
 Max.   :2013-06-28   Max.   : 4.400000   Max.   : 4.40000   Max.   : 1.89000
      X1W                X2W                X3W                X1M
 Min.   :-2.08000   Min.   :-1.04000   Min.   :-0.79000   Min.   :-0.60000
 1st Qu.:-0.24000   1st Qu.:-0.15000   1st Qu.:-0.12000   1st Qu.:-0.10000
 Median : 0.04000   Median : 0.02000   Median : 0.04000   Median : 0.03000
 Mean   : 0.01839   Mean   : 0.02048   Mean   : 0.02478   Mean   : 0.02936
 3rd Qu.: 0.34000   3rd Qu.: 0.22000   3rd Qu.: 0.22000   3rd Qu.: 0.21000
 Max.   : 1.32000   Max.   : 0.78000   Max.   : 0.58000   Max.   : 0.53000
      X6W                X2M                X3M                X4M
 Min.   :-0.40000   Min.   :-0.26000   Min.   :-0.20000   Min.   :-0.14000
 1st Qu.:-0.07000   1st Qu.:-0.06000   1st Qu.:-0.07000   1st Qu.:-0.03000
 Median : 0.07000   Median : 0.04000   Median : 0.02000   Median : 0.01000
 Mean   : 0.03193   Mean   : 0.02863   Mean   : 0.02261   Mean   : 0.02289
 3rd Qu.: 0.16000   3rd Qu.: 0.12000   3rd Qu.: 0.11000   3rd Qu.: 0.07000
 Max.   : 0.34000   Max.   : 0.30000   Max.   : 0.24000   Max.   : 0.20000
      X5M                X6M                X9M                X1Y
 Min.   :-0.06000   Min.   :-0.07000   Min.   :-0.04000   Min.   :-0.01000
 1st Qu.:-0.02000   1st Qu.: 0.01000   1st Qu.: 0.00000   1st Qu.: 0.02000
 Median : 0.01000   Median : 0.02000   Median : 0.03000   Median : 0.03000
 Mean   : 0.02357   Mean   : 0.02546   Mean   : 0.02908   Mean   : 0.03847
 3rd Qu.: 0.06000   3rd Qu.: 0.04000   3rd Qu.: 0.05000   3rd Qu.: 0.05000
 Max.   : 0.15000   Max.   : 0.11000   Max.   : 0.09000   Max.   : 0.11000
```

(b) Divide your data into two parts: a training set (75%) and a test set (25%). Why might it not be a good idea to divide the data randomly in this instance? How else might you divide the data?

Because it's a time-series data. If we randomly separate the data, the time information contained in the data will be destroyed. In this case, I divided my data into two part according to `Date`: the data between the first day and 3rd-quater day will be train data, and the left data will be test data.

```
attach(ibm)
cutoffDate= as.Date("2013-4-2")
train_data = subset(ibm, Date < cutoffDate)
test_data = subset(ibm, Date >= cutoffDate)
```

(c) Create 4 validation tests where you use 4 months of data to fit the model and then measure the performance on the following month.

Here, I referred both codes form the lecture and recitation and combined them to create 4 folds of train-test dataset.

```
train_set = list()
test_set = list()
date_range = range(ibm$Date)

for (i in 1:4){
  offset = (i - 1) * 30
  train_set[[i]] = subset(ibm, ibm$Date >= (date_range[1]+offset)
                          & ibm$Date < (date_range[1]+offset+4*30))
  test_set[[i]] = subset(ibm, ibm$Date >= (date_range[1]+offset+4*30)
                         & ibm$Date < (date_range[1]+offset+5*30))
}
```

Then, I trained a simple one-variable linear model on each fold of train data and examined their performance on the test data.

```
> test_mse = 0
> for (i in 1:4){
+   traReg = lm(Return ~ X1D, data = train_set[[i]] )
+   pred = predict(traReg, test_set[[i]])
+   test_mse = test_mse + mean( ( test_set[[i]]$Return - pred )^2 )
+   print(test_mse)
+ }
[1] 1.053471
[1] 1.484144
[1] 3.1824
[1] 3.82224
> test_mse = test_mse/4
> test_mse
[1] 0.9555599
```

(d) For each, use best subset selection to find the best model. Consider subsets of sizes from 1 to 8. Which subset size is best? What is your final model?

I used the cross-validation approach of best subset selection on the four folds as follows:

```
> # find the best model using best subset selection
> library(leaps)
> library(ISLR)
> # define a new function (refer to https://rpubs.com/davoodastaraky/subset)
> predict.regsubsets =function (object ,newdata ,id ,...){
+   form=as.formula(object$call [[2]])
+   mat=model.matrix(form,newdata)
+   coefi=coef(object ,id=id)
+   xvars=names(coefi)
+   mat[,xvars]%*%coefi
+ }

> MSE = matrix(NA, 4, 8, dimnames=list(NULL, paste(1:8)))
> for(j in 1:4){
+   best.subset = regsubsets(Return ~ ., data = train_set[[j]], nvmax = 8)
```

```
+   for(t in 1:8){
+     pred = predict.regsubsets(best.subset, test_set[[j]],id = t)
+     MSE[j,t] = mean( (test_set[[j]]$Return - pred)^2)
+   }
+ }
> mean.MSE = apply(MSE, 2, mean)
> mean.MSE

       1        2        3        4        5        6        7        8
1.029740 1.416415 2.383048 2.297965 2.331657 2.808123 2.464793 2.757122

> best.size = which(mean.MSE == min(mean.MSE))
> reg.best = regsubsets(Return ~ .,data = train_data, nvmax=19)
> coef(reg.best, best.size)

(Intercept)         X5M
 0.09589897 -4.22253309
>
> #evaluate the mse on the test data
> pred = predict.regsubsets(reg.best, test_data,id = best.size)
> test.MSE= mean( (test_data$Return - pred)^2)
> test.MSE

[1] 2.170265
```

As we can seen, according to this cross-validation process, the best subset size is 1, and my final model is:

$$Return = 0.096 - 4.223\ X5M$$

And it has a test MSE=2.17 on the test data (the last three-month data).

(e) On the same 4 validation tests, use lasso regression to find the best model. Consider the values $0, .001, .01, .1, 1, 10, 100, 1000$ for λ. Which choice of λ is the best? What is your final model? Do you think extra values of λ are required in addition to those? Explain your answer, and if you think extra values are required, try those too.

I used `glmnet` function for every λ in each fold as follows:

```
> library(glmnet)
>
> # Set up a grid of Lambda parameters to try
> grid = c(0, 0.001, 0.01, 0.1, 1, 10, 100, 1000)
> # 4-flod validation loop
> lasso.mse <- numeric(4)
> mean.lasso.mse <- numeric(length(grid))
> nzero <- c()
> lambda <- c()
> for(j in 1:length(grid)){
+   for(i in 1:4){
+     X = model.matrix(Return ~ . - 1, data = train_set[[i]])[ ,]
+     y = train_set[[i]]$Return
+     lasso.mod = glmnet(X, y, alpha = 1, lambda = grid[j])
+     nzero = cbind(nzero, lasso.mod$df)
+     lambda = cbind(lambda, lasso.mod$lambda)
+     newX = model.matrix(Return ~ .- 1, data = test_set[[i]])
+     lasso.pred = predict(lasso.mod, newx = newX, s = grid[j])
+     lasso.mse[i] = mean( (test_set[[i]]$Return - lasso.pred)^2)
```

```
+   }
+   mean.lasso.mse[j] = mean(lasso.mse)
+ }
> mean.lasso.mse

[1] 2.9491323 2.7924059 2.0822954 1.0469749 0.9152798
    0.9152798 0.9152798 0.9152798

> best.lambda = grid[which(mean.lasso.mse == min(mean.lasso.mse))]
> best.lambda

[1]    1   10  100 1000
```

From the result, it seems that $\lambda = 1, 10, 100, 1000$ are all best λ. And if we look at the printed mean MSE for each λ cross the 4 folds, we can find that the mean decreases when λ increases, and then reaches a stable value (MSE $= 0.9153$) with $\lambda \geq 1$. It seems weird, so I checked the non-zero variables in each loop:

```
> n.l <- rbind(nzero, lambda)
> print(n.l)

     [,1] [,2] [,3] [,4]   [,5]   [,6]   [,7]   [,8]  [,9] [,10] [,11] [,12]
[1,]   15   15   15   15 15.000 14.000 15.000 15.000 12.00 12.00 13.00 13.00
[2,]    0    0    0    0  0.001  0.001  0.001  0.001  0.01  0.01  0.01  0.01
     [,13] [,14] [,15] [,16] [,17] [,18] [,19] [,20] [,21] [,22] [,23] [,24]
[1,]   3.0   4.0   5.0   3.0     0     0     0     0     0     0     0     0
[2,]   0.1   0.1   0.1   0.1     1     1     1     1    10    10    10    10
     [,25] [,26] [,27] [,28] [,29] [,30] [,31] [,32]
[1,]     0     0     0     0     0     0     0     0
[2,]   100   100   100   100  1000  1000  1000  1000
```

As we can see, when λ reached 1, all estimated coefficients are shrunk to 0, which means it's a NULL model. So no more λ are needed in this case, and `best.lambda` should be 0.1. So I retrained the model with $\lambda = 0.1$ with the whole train data:

```
> X.train = model.matrix(Return ~ . - 1, data = train_data)[ ,]
> y.train = train_data$Return
>
> lasso.best = glmnet( X.train, y.train, alpha = 1, lambda = 0.1)
> lasso_coef = predict(
+   glmnet(X.train, y.train, alpha = 1, lambda = 0.1),
+   type = "coefficients" )
> lasso_coef

16 x 1 sparse Matrix of class "dgCMatrix"
                     s0
(Intercept)  0.09599649
Date         .
X1D          0.03086842
X3D          .
X1W          .
X2W          .
X3W          .
X1M          .
X6W          .
X2M          .
X3M          .
```

```
X4M          .
X5M         -1.66975780
X6M          .
X9M          .
X1Y         -0.59180696
```

Therefore, our final model will be:

$$Return = 0.096 + 0.031X1D - 1.670X5M - 0.592X1Y$$

(f) Consider the Lasso model you calculated in the previous part. What is the MSE of your model on the test data? How does that compare to the MSE on the validation tests?

```
> X.test = model.matrix(Return ~ . - 1, data = test_data)[ ,]
> y.test = test_data$Return
>
> final.pred = predict(lasso.best, newx = X.test, s = 0.1)
> lasso.test.mse = mean( (test_data$Return - final.pred)^2)
> lasso.test.mse

[1] 2.158173
```

From the result, we found the test MSE = 2.158, while the mean MSE on the validation tests is 1.047. There's a significantly difference between the two MSE, which means the variance of our model is large.

(g) Create a trading strategy from the model you picked. Start with $1 of investment and every day select to go either long or short according to the prediction of the model. What is the return of your trading strategy on the test data? Based on the results, should you invest using this strategy? How would you test the stability of these results?

I use my final model from lasso regression to generate my trading strategy: according to the predicted return on the next day, I choose to long if `pred_return` $>= 0$, and short if `pred_return` < 0.

```
> strategy = lm(Return~X1D+X5M+X1Y, data = train_data)
> pred_return = predict(strategy, test_data)
>
> # Find investment decisions every day
> portfolio = sign(pred_return)
> perf = prod( 1 + (portfolio * test_data$Return/100) )
> perf

[1] 1.097073

> # Find performance from just holding IBM
> perf_ibm = prod( 1 + (test_data$Return/100) )
> perf_ibm

[1] 0.9043544
```

Based on this trading strategy, my return on the test data is 1.097, while the return of just holding IBM stock (passive strategy) is 0.904. Therefore, I should invest using this strategy. I will test the stability of these results on more historical stock price data, and continuously retrain the model to improve the stability of the model.

4 Question 5 (KNN)

(a) Load the data into R and create a training and test set. Use the data from your section as training data, and the data from the other section as test data.

```
> cuisine <- read.csv('cuisine2.csv',sep=',',header=TRUE)
> my.cuisine <- cuisine[which(cuisine$X == "Qingyuan␣Dong"), ]
> my.cuisine

               X Mexican Chinese Greek Indian Thai Italian African French Sushi
57 Qingyuan Dong       3       5    NA      3    3       3      NA      4     4
   Steakhouse Vegan Spanish Caribbean Seafood Bar.Food Section
57          3     2       3        NA       3        2       2

> # divide data into train and test
> attach(cuisine)
> train_data = subset(cuisine, Section == my.cuisine$Section)
> test_data = subset(cuisine, Section != my.cuisine$Section)
```

(b) How might you use these rankings to develop a distance between classmates? Use this distance ranking to find the 5 closest students to you in your section.

I used the following formula to define the distance between two classmates (x, y):

$$dists\,(x, y) = \sqrt{\sum_{i=1}^{n} (x_i + y_i)^2}$$

```
> dists = as.matrix(dist(train, diag = TRUE) )
> close.names = names(sort(dists[my.name, ])[1:6])
> close.names

[1] "Xinyi␣Li"              "Qingyuan␣Dong"          "Jiachen␣Liu"
[4] "Justine␣Zhang"         "Zheng␣Peng"             "Laetitia␣De␣Coudenhove"
```

So here, I found 5 closest students to me. But after checking their rankings, I found "Xinyi Li" has NA for every feature, which results in the distance between she and me is 0. To tackle this problem, I will include the 6th nearest person to me instead of taking her into account.

```
> cuisine[close.names, ]

                       Mexican Chinese Greek Indian Thai Italian African French
Xinyi Li                    NA      NA    NA     NA   NA      NA      NA     NA
Qingyuan Dong                3       5    NA      3    3       3      NA      4
Jiachen Liu                  2       5     2      3    3       3      NA     NA
Justine Zhang                3       5     3      2    4       3       2      5
Zheng Peng                   3       5     3      4    4       4       3      3
Laetitia De Coudenhove       4       4     4      4    4       4      NA      5
                       Sushi Steakhouse Vegan Spanish Caribbean Seafood Bar.Food
Xinyi Li                  NA         NA    NA      NA        NA      NA       NA
```

```
Qingyuan Dong              4           3     2        3         NA        3         2
Jiachen Liu                3           3    NA       NA         NA        4         3
Justine Zhang              3           3     3        3          2        4         2
Zheng Peng                 5           3     3        3          3        3         2
Laetitia De Coudenhove     4           3     2        3         NA        3         2
                       Section
Xinyi Li                     2
Qingyuan Dong                2
Jiachen Liu                  2
Justine Zhang                2
Zheng Peng                   2
Laetitia De Coudenhove       2

> close.names = names(sort(dists[my.name, ])[1:7])
> close.names

[1] "Xinyi␣Li"               "Qingyuan␣Dong"          "Jiachen␣Liu"
[4] "Justine␣Zhang"          "Zheng␣Peng"             "Laetitia␣De␣Coudenhove"
[7] "Tao␣Cui"

> dists[my.name, close.names]

              Xinyi Li          Qingyuan Dong            Jiachen Liu
              0.000000               0.000000               2.529822
         Justine Zhang             Zheng Peng Laetitia De Coudenhove
              2.717465               2.717465               2.717465
               Tao Cui
              2.935198
```

Therefore, the 5 closet students to me are:
Jiachen Liu, Justine Zhang, Zheng Peng, Laetitia De Coudenhove, Tao Cui.

(c) Using only the training set, use the 3-NN method to complete the missing rankings in the data in the training set.

```
k = 4

for(i in 1: nrow(train)){
  closest = sort(as.matrix(dist(train, diag = TRUE))[i,-i])
  train_ordered = train[names(closest), ]

  for (j in 1:16){
    if(is.na(train[i,j] == FALSE)){
      closest = closest[!is.na(closest)]
      train[i,j] <- mean( closest[1: min(length(closest), k)] )
    }
  }
}
```

(d) Find the number of neighbors that minimizes the in-training RMSE. Consider number of neighbors from 1 to 20 and plot the RMSE (Root MSE) for each. Do the same thing with the test set RMSE. What do you notice about these RMSEs?

They are not the same. In my codes for this part, something went wrong. And I really need help on this part. I will definitely figure this out.