FE590. Assignment #4.

Enter Your Name Here, or "Anonymous" if you want to remain anonymous..

2019-11-24

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:Tingyi Lu

CWID:10440683

Date:11/20/2019

Instructions

When you have completed the assignment, knit the document into a PDF file, and upload *both* the .pdf and .Rmd files to Canvas.

Note that you must have LaTeX installed in order to knit the equations below. If you do not have it installed, simply delete the questions below.

```
CWID = 10440683 #Place here your Campus wide ID number, this will personalize
#your results, but still maintain the reproduceable 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)
```

Question 1:

In this assignment, you will be required to find a set of data to run regression on. This data set should be financial in nature, and of a type that will work with the models we have discussed this semester (hint: we didn't look at time series) You may not use any of the data sets in the ISLR package that we have been looking at all semester. Your data set that you choose should have both qualitative and quantitative variables. (or has variables that you can transform)

Provide a description of the data below, where you obtained it, what the variable names are and what it is describing.

Objective: This project aims to determine the relationship between Korea Stock Exchange and stock markets in many other countries by analyzing their Stock Indexes.

Data source: The dataset was downloaded from https://finance.yahoo.com, ranging from 11/19/2009 - 11/19/2019.

Load in data

```
data <- read.csv("data.csv")
head(data)
```

```
##       Date shanghai        AXJO       GDAXI        HSI        JKSE        KS11
## 1 11/19/09 3320.612 4749.200195 5702.180176 22643.1602 2468.714844 1620.599976
```

```
## 2 11/20/09 3308.346 4685.799805 5663.149902 22455.8398  2487.29126 1619.050049
## 3 11/23/09 3338.663        4717  5801.47998 22771.3906 2481.342285 1606.420044
## 4 11/24/09 3223.526        4685 5769.310059 22423.1406 2471.810547 1611.880005
## 5 11/25/09 3290.165 4722.200195  5803.02002 22611.8008 2461.455078  1599.52002
## 6 11/26/09 3170.979 4708.600098 5614.169922 22210.4102 2393.447998      1524.5
##          MERV         MXX       N100        N255 S.P.500
## 1 2240.030029 30817.66992  655.23999 9549.469727 1094.90
## 2      2231.5 30666.50977 650.549988 9497.679688 1091.38
## 3 2251.379883 31126.16992 663.330017 9401.580078 1106.24
## 4 2247.870117 30961.99023 659.340027 9441.639648 1105.65
## 5 2253.449951 31364.03906 663.400024 9383.240234 1110.63
## 6  2156.98999 30447.83008 641.659973 9081.519531 1091.49
```

```
#transform the raw data
for(i in 2:12){
  data[,i] <- as.numeric(data[,i])
}
```

1 Response variable: KS11: KOSPI Composite Index, Korea Stock Exchange

10 Explanatory Variables: shanghai: SSE Composite Index, ShangHai Stock Exchange, China AXJO: S&P/ASX 200 index, Australian Securities Exchange GDAXI: German stock index HSI: Hang Seng Index, Hong Kong JKSE: Jakarta Stock Exchange Composite Index MERV: Buenos Aires MXX: Mexio IPC Index N100: EURONEXT 100 Index, Europe N255: Nikkei index, Tokyo S.P.500: S&P 500 Index, America

Other variable: The date on which price was recorded.

Data prepariation: Since all the values in the dataset are the actural price. I will first satandardize the data by calculating the continous compounded return, which is log return of each indexes.

Calculating log return

```
for(i in 2:12){
a <- diff(log(as.numeric(data[,i])))
a <- c(0,a)
data[,i] <- a
}
```

```
data <- data[-1,]
data[,1] <- as.Date(data[,1], format="%m/%d/%y")
head(data)
```

```
##         Date     shanghai         AXJO       GDAXI         HSI       JKSE
## 2 2009-11-20 -0.003700771 -0.13058321 -0.01393427 -0.08299692  0.9444616
## 3 2009-11-23  0.009122103  0.06995859  0.03943656  0.13752980 -0.3254224
## 4 2009-11-24 -0.035094699 -0.07177183 -0.01358101 -0.14935154 -0.3677248
## 5 2009-11-25  0.020461973  0.08352679  0.01537796  0.08294908 -0.5877867
## 6 2009-11-26 -0.036897353 -0.02707441 -0.05252243 -0.16047244 -1.6094379
## 7 2009-11-27 -0.023843862 -0.27914231  0.02059998 -0.47939337  0.6931472
##          KS11         MERV        MXX         N100         N255       S.P.500
## 2 -0.03077166 -0.017442303 -0.4054651 -0.05175799 -0.012903405 -0.0032201017
## 3 -0.37469345  0.030327234  0.9162907  0.12820515 -0.017619611  0.0135239136
## 4  0.12783337 -0.004276557 -0.2231436 -0.03688699  0.005907765 -0.0005334497
## 5 -0.38566248  0.008534902  0.5355182  0.03922071 -0.009103386  0.0044940074
## 6 -2.83321334 -0.159357792 -2.6149598 -0.19074857 -0.033003228 -0.0173837003
## 7  1.09861229  0.056512210  1.6739764  0.03465350  0.025660377  0.0037858188
```

Ttransform the data into df.

```r
df<- data.frame(data)
head(df)
```

```
##         Date     shanghai        AXJO       GDAXI         HSI       JKSE
## 2 2009-11-20 -0.003700771 -0.13058321 -0.01393427 -0.08299692  0.9444616
## 3 2009-11-23  0.009122103  0.06995859  0.03943656  0.13752980 -0.3254224
## 4 2009-11-24 -0.035094699 -0.07177183 -0.01358101 -0.14935154 -0.3677248
## 5 2009-11-25  0.020461973  0.08352679  0.01537796  0.08294908 -0.5877867
## 6 2009-11-26 -0.036897353 -0.02707441 -0.05252243 -0.16047244 -1.6094379
## 7 2009-11-27 -0.023843862 -0.27914231  0.02059998 -0.47939337  0.6931472
##          KS11         MERV        MXX         N100         N255       S.P.500
## 2 -0.03077166 -0.017442303 -0.4054651 -0.05175799 -0.012903405 -0.0032201017
## 3 -0.37469345  0.030327234  0.9162907  0.12820515 -0.017619611  0.0135239136
## 4  0.12783337 -0.004276557 -0.2231436 -0.03688699  0.005907765 -0.0005334497
## 5 -0.38566248  0.008534902  0.5355182  0.03922071 -0.009103386  0.0044940074
## 6 -2.83321334 -0.159357792 -2.6149598 -0.19074857 -0.033003228 -0.0173837003
## 7  1.09861229  0.056512210  1.6739764  0.03465350  0.025660377  0.0037858188
```

```r
nrow(df)
```

```
## [1] 2430
```

```r
dim(df)
```

```
## [1] 2430   12
```

Create a qualitative variable by categorizing KS11 as Bullish for possible return and Bearish for negative return.

```r
ksTrend <- rep("Bullish",2430)
ksTrend[data$KS11<0] <- "Bearish"
df$ksTrend <- ksTrend
df$ksTrend <- as.factor(df$ksTrend)
summary(df)
```

```
##       Date              shanghai              AXJO          
##  Min.   :2009-11-20   Min.   :-8.873e-02   Min.   :-2.876169  
##  1st Qu.:2012-05-23   1st Qu.:-5.966e-03   1st Qu.:-0.025897  
##  Median :2014-11-22   Median : 5.037e-04   Median : 0.002205  
##  Mean   :2014-11-19   Mean   :-5.094e-05   Mean   : 0.000534  
##  3rd Qu.:2017-05-21   3rd Qu.: 6.367e-03   3rd Qu.: 0.028012  
##  Max.   :2019-11-19   Max.   : 5.604e-02   Max.   : 3.200408  
##      GDAXI                HSI                  JKSE          
##  Min.   :-7.775696   Min.   :-4.055092   Min.   :-2.079442  
##  1st Qu.:-0.016276   1st Qu.:-0.029483   1st Qu.:-0.015597  
##  Median : 0.002086   Median : 0.001298   Median : 0.002986  
##  Mean   :-0.000158   Mean   : 0.000256   Mean   : 0.002369  
##  3rd Qu.: 0.018761   3rd Qu.: 0.033629   3rd Qu.: 0.021277  
##  Max.   : 7.790282   Max.   : 4.181844   Max.   : 1.098612  
##       KS11                 MERV                MXX          
##  Min.   :-2.939373   Min.   :-6.683778   Min.   :-3.258097  
##  1st Qu.:-0.035305   1st Qu.:-0.017086   1st Qu.:-0.027833  
##  Median : 0.001582   Median : 0.001566   Median : 0.001325  
##  Mean   : 0.001600   Mean   : 0.000407   Mean   : 0.001548  
##  3rd Qu.: 0.044640   3rd Qu.: 0.021569   3rd Qu.: 0.031151  
##  Max.   : 3.113727   Max.   : 7.776115   Max.   : 2.397895  
##       N100                 N255              S.P.500              ksTrend    
```

```
##  Min.   :-7.755767   Min.   :-7.772753   Min.   :-0.0689584   Bearish:1163
##  1st Qu.:-0.019610   1st Qu.:-0.016833   1st Qu.:-0.0032917   Bullish:1267
##  Median : 0.002139   Median : 0.001812   Median : 0.0005870
##  Mean   :-0.000533   Mean   :-0.000128   Mean   : 0.0004116
##  3rd Qu.: 0.024706   3rd Qu.: 0.022109   3rd Qu.: 0.0050162
##  Max.   : 7.071998   Max.   : 7.773174   Max.   : 0.0484032
```

Check missing values, simply drop if there is any.

```
df<- na.omit(df)
nrow(df)
```

```
## [1] 2430
```

The data is clean without any missing values, so now I can do further analysis.

Question 2:

Pick a quantitative variable and fit at least four different models in order to predict that variable using the other predictors. Determine which of the models is the best fit. You will need to provide strong reasons as to why the particular model you chose is the best one. You will need to confirm the model you have selected provides the best fit and that you have obtained the best version of that particular model (i.e. subset selection or validation for example). You need to convince the grader that you have chosen the best model.

Split the data into training and testing set.

```
x <- sample(length(df$Date),as.integer(length(df$Date)*80/100))
train <- df[x,]
test <- df[-x,]
nrow(train)
```

```
## [1] 1944
```

```
nrow(test)
```

```
## [1] 486
```

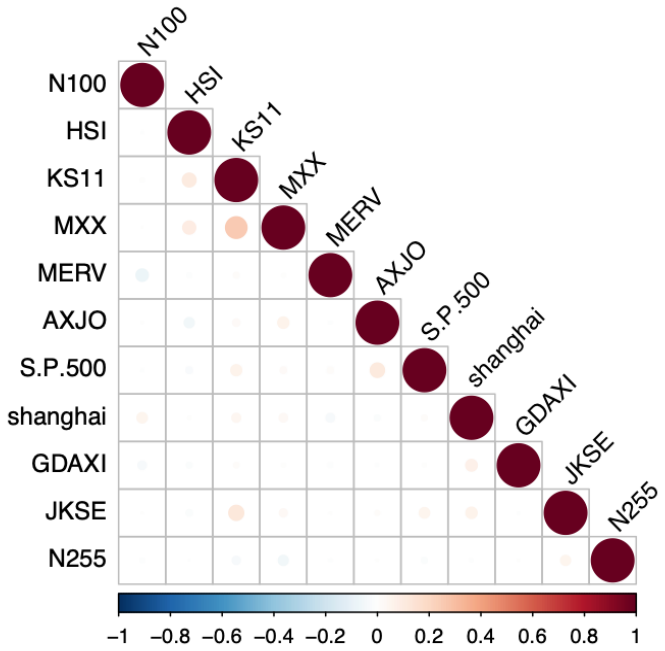
Correlation matrix.

```
source("http://www.sthda.com/upload/rquery_cormat.r")
mydata <- train[, c(2:12)]
require("corrplot")
```

```
## Loading required package: corrplot
```

```
## corrplot 0.84 loaded
```

```
cor <- rquery.cormat(mydata)
```




```
cor$r
```

```
##             N100     HSI   KS11    MXX    MERV    AXJO S.P.500 shanghai   GDAXI
## N100           1
## HSI      -0.0015       1
## KS11      0.0065     0.1      1
## MXX      0.00024    0.09   0.25      1
## MERV      -0.079 -0.0098  0.016 0.0056       1
## AXJO      0.0017  -0.052  0.026  0.066 -0.0026       1
## S.P.500  -0.0026  -0.021  0.063  0.021   0.018    0.11       1
## shanghai   0.055  0.0035   0.04  0.036  -0.037  -0.015   0.015        1
## GDAXI     -0.036  -0.015  0.014 0.0063 -0.0085 -0.0029  0.0052    0.071       1
## JKSE     0.00088  -0.019   0.12  0.032  0.0012   0.011   0.057    0.066 -0.0017
## N255     -0.0088  -2e-04 -0.033 -0.052 -0.0026  0.0015  -0.016  -0.0036  0.0012
##            JKSE N255
## N100
## HSI
## KS11
## MXX
## MERV
## AXJO
## S.P.500
## shanghai
## GDAXI
## JKSE          1
## N255      0.051    1
```

```
cor$p
```

```
##             N100     HSI    KS11    MXX MERV  AXJO S.P.500 shanghai GDAXI  JKSE
## N100           0
## HSI         0.95       0
## KS11        0.78 6.3e-06       0
## MXX         0.99 6.9e-05 4.3e-29      0
## MERV     0.00048    0.66    0.48    0.8    0
## AXJO        0.94   0.021    0.26 0.0034 0.91     0
## S.P.500     0.91    0.35  0.0056   0.37 0.43 8e-07       0
## shanghai   0.015    0.88   0.079   0.11  0.1  0.52     0.5        0
## GDAXI       0.12     0.5    0.54   0.78 0.71   0.9    0.82   0.0018     0
## JKSE        0.97    0.41 1.8e-07   0.15 0.96  0.62   0.012   0.0035  0.94     0
## N255         0.7    0.99    0.14  0.021 0.91  0.95    0.48     0.88  0.96 0.025
##          N255
## N100
## HSI
## KS11
## MXX
## MERV
## AXJO
## S.P.500
## shanghai
## GDAXI
## JKSE
## N255        0
```

It seems this dataset has low dependency among each other, the only relatively highest correlation is between KS11 and MXX.

```
colnames(train)
```

```
##  [1] "Date"     "shanghai" "AXJO"     "GDAXI"    "HSI"      "JKSE"
##  [7] "KS11"     "MERV"     "MXX"      "N100"     "N255"     "S.P.500"
## [13] "ksTrend"
```

(1) Simple Linesr Regression Model: Select the best 1 variable using regsubsets model.

```
library(leaps)
s <- regsubsets(KS11 ~ AXJO + GDAXI + HSI + JKSE + N255 + MERV + MXX  + N100 +
                  shanghai + S.P.500, data= train, method= "exhaustive",nvmax = 10)

summary(s)[7]
```

```
## $outmat
##           AXJO GDAXI HSI JKSE N255 MERV MXX N100 shanghai S.P.500
## 1  ( 1 )  " "  " "   " " " "  " "  " "  "*" " "  " "      " "
## 2  ( 1 )  " "  " "   " " "*"  " "  " "  "*" " "  " "      " "
## 3  ( 1 )  " "  " "   "*" "*"  " "  " "  "*" " "  " "      " "
## 4  ( 1 )  " "  " "   "*" "*"  " "  " "  "*" " "  " "      "*"
## 5  ( 1 )  " "  " "   "*" "*"  "*"  " "  "*" " "  " "      "*"
## 6  ( 1 )  " "  " "   "*" "*"  "*"  " "  "*" " "  "*"      "*"
## 7  ( 1 )  " "  " "   "*" "*"  "*"  "*"  "*" " "  "*"      "*"
## 8  ( 1 )  " "  "*"   "*" "*"  "*"  "*"  "*" " "  "*"      "*"
## 9  ( 1 )  "*"  "*"   "*" "*"  "*"  "*"  "*" " "  "*"      "*"
## 10  ( 1 ) "*"  "*"   "*" "*"  "*"  "*"  "*" "*"  "*"      "*"
```

From the summary, the best 1 variable model is with MXX, which is Mexio IPC Index. This result is the same as what I got from the previous correlation matrix. So, I begin with a simple linear regression model using MXX as variable.

```
model1 <- lm(KS11 ~ MXX, data=train)
summary(model1)
```

```
##
## Call:
## lm(formula = KS11 ~ MXX, data = train)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -2.26968 -0.03806  0.00166  0.04639  2.54583
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept) -0.0002799  0.0042150  -0.066    0.947
## MXX          0.2905899  0.0255367  11.379   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.1858 on 1942 degrees of freedom
## Multiple R-squared:  0.06251,    Adjusted R-squared:  0.06203
## F-statistic: 129.5 on 1 and 1942 DF,  p-value: < 2.2e-16
```

The explanatory variable MXX is significant with a small p-value.

Test the model:

```
p1<- predict(model1,newdata = test)
e1 <- mean((test$KS11 - p1)^2)
e1
```

```
## [1] 0.06368045
```

The MSE of this model is 0.06368045, which is relatively small. So this model might be good.

(2)Multiple Linear Regression. Find the best model with the help of Mallow Cp.

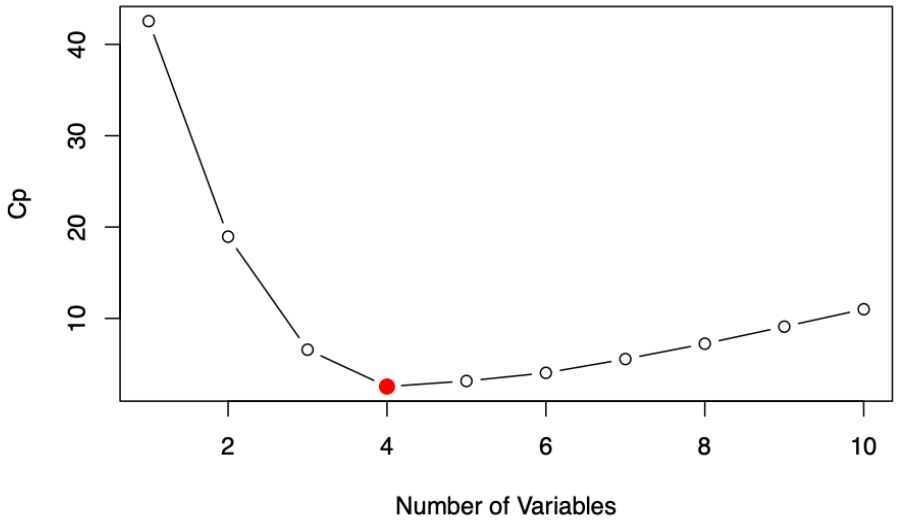
```
c <- summary(s)$cp
c
```

```
##  [1] 42.547102 18.949819  6.573860  2.537514  3.141741  4.033863  5.547056
##  [8]  7.225775  9.097368 11.000000
```

```
#plot the Cp statistics
plot(c ,xlab =" Number of Variables ",ylab="Cp",type='b')

#plot a red dot to indicate the model with minimized Cp statistics

points (which.min(c), c[which.min(c)], col ="red",cex =2, pch =20)
```



Model with 4 variables is the best, as it has the minimum Mallows Cp.

```
summary(s)[7]
```

```
## $outmat
##           AXJO GDAXI HSI JKSE N255 MERV MXX N100 shanghai S.P.500
## 1  ( 1 )  " "  " "   " " " "  " "  " "  "*" " "  " "      " "
## 2  ( 1 )  " "  " "   " " "*"  " "  " "  "*" " "  " "      " "
## 3  ( 1 )  " "  " "   "*" "*"  " "  " "  "*" " "  " "      " "
## 4  ( 1 )  " "  " "   "*" "*"  " "  " "  "*" " "  " "      "*"
## 5  ( 1 )  " "  " "   "*" "*"  "*"  " "  "*" " "  " "      "*"
## 6  ( 1 )  " "  " "   "*" "*"  "*"  " "  "*" " "  "*"      "*"
## 7  ( 1 )  " "  " "   "*" "*"  "*"  "*"  "*" " "  "*"      "*"
## 8  ( 1 )  " "  "*"   "*" "*"  "*"  "*"  "*" " "  "*"      "*"
## 9  ( 1 )  "*"  "*"   "*" "*"  "*"  "*"  "*" " "  "*"      "*"
## 10  ( 1 ) "*"  "*"   "*" "*"  "*"  "*"  "*" "*"  "*"      "*"
```

```
coef(s,4)
```

```
## (Intercept)         HSI        JKSE         MXX     S.P.500
## -0.00065183  0.06962177  0.19851890  0.27640735  1.11458105
```

The 4 selected variables are HSI,JKSE,MXX,S.P.500

Now build a multiple regression model using these 4 variables.

```
model2 <- lm(KS11 ~ HSI+JKSE+MXX+S.P.500,data=train)
summary(model2)
```

```
##
## Call:
## lm(formula = KS11 ~ HSI + JKSE + MXX + S.P.500, data = train)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
```

```
## -2.29787 -0.03988  0.00211  0.04803  2.47313
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept) -0.0006518  0.0041729  -0.156 0.875886
## HSI          0.0696218  0.0180969   3.847 0.000123 ***
## JKSE         0.1985189  0.0397185   4.998 6.31e-07 ***
## MXX          0.2764073  0.0253817  10.890  < 2e-16 ***
## S.P.500      1.1145811  0.4533657   2.458 0.014040 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.1838 on 1939 degrees of freedom
## Multiple R-squared:  0.08427,    Adjusted R-squared:  0.08238
## F-statistic: 44.61 on 4 and 1939 DF,  p-value: < 2.2e-16
```

Since the variable S.P.500 is not so significant, I will drop this variable and run the model again.

```
new_model2 <- lm(KS11 ~ HSI + JKSE  + MXX, data=train)
summary(new_model2)
```

```
##
## Call:
## lm(formula = KS11 ~ HSI + JKSE + MXX, data = train)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -2.32690 -0.04115  0.00219  0.04784  2.48275
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept) -0.0002232  0.0041746  -0.053 0.957372
## HSI          0.0686424  0.0181160   3.789 0.000156 ***
## JKSE         0.2039943  0.0397075   5.137 3.06e-07 ***
## MXX          0.2776961  0.0254093  10.929  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.1841 on 1940 degrees of freedom
## Multiple R-squared:  0.08141,    Adjusted R-squared:  0.07999
## F-statistic: 57.31 on 3 and 1940 DF,  p-value: < 2.2e-16
```

Now all variables are significant, and the Adj R-squared of the model is 0.08, which is higher than model 1. So this model might be better.

Test model2:

```
p2 <- predict(new_model2,newdata = test)
e2 <- mean((test$KS11 - p2)^2)
e2
```

```
## [1] 0.06355534
```

(3)Support Vector Regression using the same selected variables from Msllow Cp.

```
#install.packages("e1071")
library(e1071)
model3 <- svm(KS11 ~ HSI + JKSE + MXX + S.P.500,data=train, type= "eps-regression")
```

```
p3 <- predict(model3, newdata = test)
e3=mean((test$KS11 - p3)^2)
e3
```

```
## [1] 0.0667507
```

(4)Random Forest Regression

```
library(randomForest)
```

```
## randomForest 4.6-14
```

```
## Type rfNews() to see new features/changes/bug fixes.
```

```
model4 <- randomForest(x=train[,c(5,6,9,12)],y=train$KS11,ntree = 500)
```

```
p4 <- predict(model4,newdata = test)
e4=mean((test$KS11 - p4)^2)
e4
```

```
## [1] 0.06410794
```

Comparison table of MSE for all the models:

```
data.frame("MSE"=c("model1"=e1,"model2"=e2,"model3"=e3,"model4"=e4))
```

```
##               MSE
## model1 0.06368045
## model2 0.06355534
## model3 0.06675070
## model4 0.06410794
```

Bsed on the result, model2, which is Multiple Regression Model, is the best with the lowest MSE.

Question 3:

Do the same approach as in question 2, but this time for a qualitative variable.

The same variables selected from Mallow Cp are used to build following models. (1) Logistic Regression Model

```
dirTest <- test$ksTrend

m1 <- glm(ksTrend ~ HSI+JKSE+MXX+S.P.500,data=train,family = binomial)
summary(m1)
```

```
##
## Call:
## glm(formula = ksTrend ~ HSI + JKSE + MXX + S.P.500, family = binomial,
##     data = train)
##
## Deviance Residuals:
##    Min      1Q  Median      3Q     Max
## -1.598  -1.208   1.058   1.145   2.360
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept)  0.07327    0.04556   1.608   0.1078
## HSI          0.51515    0.23532   2.189   0.0286 *
```

```
## JKSE         0.07607    0.44989   0.169   0.8657
## MXX          0.61006    0.31368   1.945   0.0518 .
## S.P.500      3.28540    4.96170   0.662   0.5079
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 2692.3  on 1943  degrees of freedom
## Residual deviance: 2681.4  on 1939  degrees of freedom
## AIC: 2691.4
##
## Number of Fisher Scoring iterations: 3
```

Prediction:

```
prob <- predict(m1, test,type = "response")
contrasts(test$ksTrend)
```

```
##         Bullish
## Bearish       0
## Bullish       1
```

```
predicted.class <- ifelse(prob > 0.5,"Bullish", " Bearlish")
head(predicted.class)
```

```
##         3         7         8         9        10        24
## "Bullish" "Bullish" "Bullish" "Bullish" "Bullish" "Bullish"
```

```
table(predicted.class,dirTest)
```

```
##                dirTest
## predicted.class Bearish Bullish
##        Bearlish      36      33
##       Bullish       191     226
```

```
e11 <- mean(predicted.class != dirTest)
e11
```

```
## [1] 0.5349794
```

(2)LDA model

```
library(MASS)
m2  <-  lda(ksTrend ~  HSI + JKSE + MXX + S.P.500,data=train)
```

```
p22 <- predict(m2,test)$class
```

```
table(p22,dirTest)
```

```
##          dirTest
## p22       Bearish Bullish
##   Bearish      32      28
##   Bullish     195     231
```

```
e22 <- mean(p22 != dirTest)
e22
```

```
## [1] 0.4588477
```

(4) QDA model

```
m3 <- qda(ksTrend ~  HSI + JKSE + MXX + S.P.500,data=train)
```

```
p33 <- predict(m3,test)$class
table(p33,dirTest)
```

```
##          dirTest
## p33       Bearish Bullish
##   Bearish      19      23
##   Bullish     208     236
```

```
e33 <- mean(p33 != dirTest)
e33
```

```
## [1] 0.4753086
```

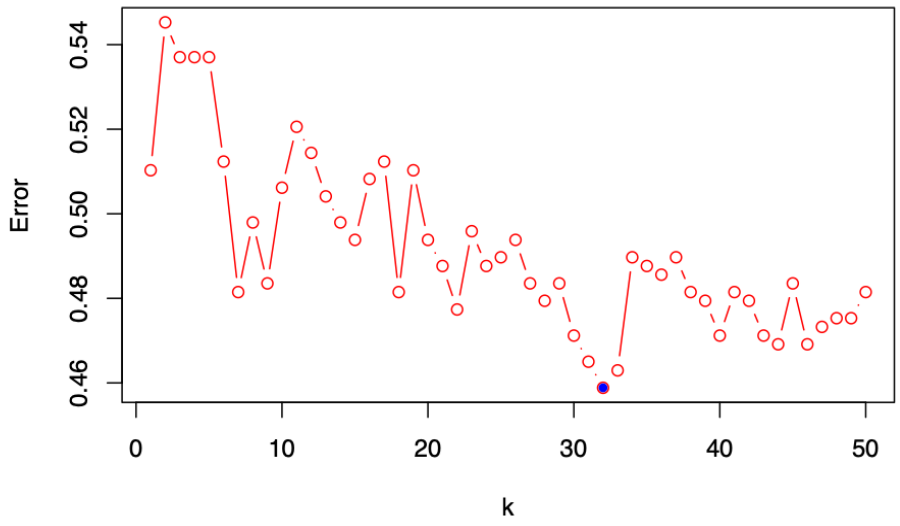
(4)KNN

```
trainKNN  <- as.matrix(data.frame(train$HSI,train$JKSE,train$MXX,train$S.P.500))
testKNN <- as.matrix(data.frame(test$HSI,test$JKSE,test$MXX,test$S.P.500))
dirTrain <- train$ksTrend
```

Try KNN from k=1 to k=50

```
library(class)
e44 <- NULL
for(i in 1:50){
  set.seed(personal)
  pKNN  <- knn(trainKNN, testKNN, dirTrain, k=i)
  e44[i] <- mean(pKNN != dirTest)
}
```

```
plot(e44, type = "b",xlab = "k",ylab = "Error",col="red")
points(which.min(e44),e44[which.min(e44)],pch=20, col="blue")
```




```
which.min(e44)
```

```
## [1] 32
```

I got the minimum misclassification error at k = 32. So I can build a knn model with k=32.

```
m4 <- knn(trainKNN,testKNN, dirTrain, k=which.min(e44))
#confusion matrix
table(m4,dirTest)
```

```
##          dirTest
## m4        Bearish Bullish
##   Bearish      99      97
##   Bullish     128     162
```

```
error4 <- mean(m4 != dirTest)
error4
```

```
## [1] 0.462963
```

(5)Random Forest Tree

```
set.seed(personal)

m5 <- randomForest(ksTrend ~  HSI + JKSE + MXX + S.P.500,data = train, importance=TRUE)
m5
```

```
##
## Call:
##  randomForest(formula = ksTrend ~ HSI + JKSE + MXX + S.P.500,      data = train, importance = TRUE)
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 2
##
##         OOB estimate of  error rate: 49.74%
## Confusion matrix:
##         Bearish Bullish class.error
## Bearish     423     513   0.5480769
## Bullish     454     554   0.4503968
```

```
p55 <- predict(m5,newdata = test,type = "class")
```

```
#confusion matrix
table(p55, dirTest)
```

```
##          dirTest
## p55       Bearish Bullish
##   Bearish     100     115
##   Bullish     127     144
```

```
e55 <- mean(p55 != dirTest)
e55
```

```
## [1] 0.4979424
```

Comparison of the results:

```
table <- data.frame("Methods"=c("Logistic Regression","LDA","QDA","KNN","RandomForest"),
                    "Misclassification Errors"=c(e11,e22,e33,error4,e55))
table
```

```
##               Methods Misclassification.Errors
## 1 Logistic Regression               0.5349794
```

```
## 2                LDA               0.4588477
## 3                QDA               0.4753086
## 4                KNN               0.4629630
## 5       RandomForest               0.4979424
```

From the result, LDA model has the lowest misclassification error of 0.4588477. So LDA model is the best for this qualitive variable.

Question 4:

(Based on ISLR Chapter 9 #7) In this problem, you will use support vector approaches in order to predict whether a given car gets high or low gas mileage based on the Auto data set.

(a)

Create a binary variable that takes on a 1 for cars with gas mileage above the median, and a 0 for cars with gas mileage below the median.

```
library(ISLR)
attach(Auto)
med_gas=median(Auto$mpg)
new_var = ifelse(mpg >med_gas,1,0)
Auto$mpglevel = as.factor(new_var)
```

(b)

Fit a support vector classifier to the data with various values of cost, in order to predict whether a car gets high or low gas mileage. Report the cross-validation errors associated with different values of this parameter. Comment on your results.

```
library(e1071)
set.seed(personal)
tune.out = tune(svm, mpglevel~., data = Auto, kernel="linear",
                ranges = list(cost=seq(1,20),by=1))

summary(tune.out)
```

```
##
## Parameter tuning of 'svm':
##
## - sampling method: 10-fold cross validation
##
## - best parameters:
##  cost by
##     1  1
##
## - best performance: 0.007628205
##
## - Detailed performance results:
##    cost by       error dispersion
## 1     1  1 0.007628205 0.01228382
## 2     2  1 0.007628205 0.01228382
## 3     3  1 0.010192308 0.01315951
## 4     4  1 0.017756410 0.02086164
## 5     5  1 0.020256410 0.02601613
```

```
## 6     6  1 0.017692308 0.02378657
## 7     7  1 0.017692308 0.02378657
## 8     8  1 0.017692308 0.02378657
## 9     9  1 0.017692308 0.02378657
## 10   10  1 0.017692308 0.02378657
## 11   11  1 0.017692308 0.02378657
## 12   12  1 0.017692308 0.02378657
## 13   13  1 0.017692308 0.02378657
## 14   14  1 0.017692308 0.02378657
## 15   15  1 0.022820513 0.02193311
## 16   16  1 0.022820513 0.02193311
## 17   17  1 0.022820513 0.02193311
## 18   18  1 0.025384615 0.02041510
## 19   19  1 0.025384615 0.02041510
## 20   20  1 0.027884615 0.02184352
```

Best cost is 1 with the best performance of 0.007628205.

(c)

Now repeat for (b), this time using SVMs with radial and polynomial basis kernels, with different values of gamma and degree and cost. Comment on your results.

```
set.seed(personal)
tune.out <- tune(svm, mpglevel ~ ., data = Auto, kernel = "polynomial",
                 ranges = list(cost = c( 0.1, 1, 5, 10, 20,25,30,35,50,70,80,100),
                 degree = c(2, 3, 4,5)))
summary(tune.out)
```

```
##
## Parameter tuning of 'svm':
##
## - sampling method: 10-fold cross validation
##
## - best parameters:
##  cost degree
##    80      2
##
## - best performance: 0.3033333
##
## - Detailed performance results:
##     cost degree     error dispersion
## 1    0.1      2 0.5791026 0.03816505
## 2    1.0      2 0.5791026 0.03816505
## 3    5.0      2 0.5791026 0.03816505
## 4   10.0      2 0.5637821 0.03674375
## 5   20.0      2 0.4750000 0.09983733
## 6   25.0      2 0.4415385 0.09802910
## 7   30.0      2 0.4158333 0.08940721
## 8   35.0      2 0.3825000 0.07795137
## 9   50.0      2 0.3493590 0.07326848
## 10  70.0      2 0.3135897 0.06105415
## 11  80.0      2 0.3033333 0.06467960
## 12 100.0      2 0.3033333 0.06354013
## 13   0.1      3 0.5791026 0.03816505
```

```
## 14   1.0      3 0.5791026 0.03816505
## 15   5.0      3 0.5791026 0.03816505
## 16  10.0      3 0.5791026 0.03816505
## 17  20.0      3 0.5791026 0.03816505
## 18  25.0      3 0.5791026 0.03816505
## 19  30.0      3 0.5740385 0.03723747
## 20  35.0      3 0.5586538 0.05130453
## 21  50.0      3 0.4777564 0.11975425
## 22  70.0      3 0.3340385 0.15968972
## 23  80.0      3 0.3647436 0.10457257
## 24 100.0      3 0.3519872 0.09424431
## 25   0.1      4 0.5791026 0.03816505
## 26   1.0      4 0.5791026 0.03816505
## 27   5.0      4 0.5791026 0.03816505
## 28  10.0      4 0.5791026 0.03816505
## 29  20.0      4 0.5791026 0.03816505
## 30  25.0      4 0.5791026 0.03816505
## 31  30.0      4 0.5791026 0.03816505
## 32  35.0      4 0.5791026 0.03816505
## 33  50.0      4 0.5791026 0.03816505
## 34  70.0      4 0.5791026 0.03816505
## 35  80.0      4 0.5791026 0.03816505
## 36 100.0      4 0.5791026 0.03816505
## 37   0.1      5 0.5791026 0.03816505
## 38   1.0      5 0.5791026 0.03816505
## 39   5.0      5 0.5791026 0.03816505
## 40  10.0      5 0.5791026 0.03816505
## 41  20.0      5 0.5791026 0.03816505
## 42  25.0      5 0.5791026 0.03816505
## 43  30.0      5 0.5791026 0.03816505
## 44  35.0      5 0.5791026 0.03816505
## 45  50.0      5 0.5791026 0.03816505
## 46  70.0      5 0.5791026 0.03816505
## 47  80.0      5 0.5791026 0.03816505
## 48 100.0      5 0.5791026 0.03816505
```

For a polynomial kernel, the lowest cross-validation error is obtained for a degree of 2 and a cost of 80.

```
set.seed(personal)
tune.out <- tune(svm, mpglevel ~ ., data = Auto, kernel = "radial",
                 ranges = list(cost = c( 0.1, 1, 5, 10, 20,25,30,35,50,70,80,100),
                 gamma=c( 0.1, 1, 5, 10, 20,25,30,35,50,70,80,100)))
summary(tune.out)
```

```
##
## Parameter tuning of 'svm':
##
## - sampling method: 10-fold cross validation
##
## - best parameters:
##  cost gamma
##    10   0.1
##
## - best performance: 0.03307692
##
```

```
## - Detailed performance results:
##       cost gamma      error dispersion
## 1     0.1   0.1 0.07653846 0.03813752
## 2     1.0   0.1 0.05352564 0.03877749
## 3     5.0   0.1 0.03307692 0.03612590
## 4    10.0   0.1 0.03307692 0.03404377
## 5    20.0   0.1 0.03570513 0.04535963
## 6    25.0   0.1 0.03320513 0.04360838
## 7    30.0   0.1 0.03576923 0.04550209
## 8    35.0   0.1 0.03320513 0.04011840
## 9    50.0   0.1 0.03826923 0.03867704
## 10   70.0   0.1 0.03576923 0.03864250
## 11   80.0   0.1 0.03576923 0.03864250
## 12  100.0   0.1 0.03576923 0.03864250
## 13    0.1   1.0 0.57910256 0.03816505
## 14    1.0   1.0 0.06365385 0.04485760
## 15    5.0   1.0 0.05858974 0.04143669
## 16   10.0   1.0 0.05858974 0.04143669
## 17   20.0   1.0 0.05858974 0.04143669
## 18   25.0   1.0 0.05858974 0.04143669
## 19   30.0   1.0 0.05858974 0.04143669
## 20   35.0   1.0 0.05858974 0.04143669
## 21   50.0   1.0 0.05858974 0.04143669
## 22   70.0   1.0 0.05858974 0.04143669
## 23   80.0   1.0 0.05858974 0.04143669
## 24  100.0   1.0 0.05858974 0.04143669
## 25    0.1   5.0 0.57910256 0.03816505
## 26    1.0   5.0 0.52288462 0.06209833
## 27    5.0   5.0 0.52038462 0.06138454
## 28   10.0   5.0 0.52038462 0.06138454
## 29   20.0   5.0 0.52038462 0.06138454
## 30   25.0   5.0 0.52038462 0.06138454
## 31   30.0   5.0 0.52038462 0.06138454
## 32   35.0   5.0 0.52038462 0.06138454
## 33   50.0   5.0 0.52038462 0.06138454
## 34   70.0   5.0 0.52038462 0.06138454
## 35   80.0   5.0 0.52038462 0.06138454
## 36  100.0   5.0 0.52038462 0.06138454
## 37    0.1  10.0 0.57910256 0.03816505
## 38    1.0  10.0 0.54326923 0.05531279
## 39    5.0  10.0 0.53820513 0.05365180
## 40   10.0  10.0 0.53820513 0.05365180
## 41   20.0  10.0 0.53820513 0.05365180
## 42   25.0  10.0 0.53820513 0.05365180
## 43   30.0  10.0 0.53820513 0.05365180
## 44   35.0  10.0 0.53820513 0.05365180
## 45   50.0  10.0 0.53820513 0.05365180
## 46   70.0  10.0 0.53820513 0.05365180
## 47   80.0  10.0 0.53820513 0.05365180
## 48  100.0  10.0 0.53820513 0.05365180
## 49    0.1  20.0 0.57910256 0.03816505
## 50    1.0  20.0 0.56115385 0.04207091
## 51    5.0  20.0 0.55352564 0.04079490
## 52   10.0  20.0 0.55352564 0.04079490
```

```
## 53   20.0  20.0 0.55352564 0.04079490
## 54   25.0  20.0 0.55352564 0.04079490
## 55   30.0  20.0 0.55352564 0.04079490
## 56   35.0  20.0 0.55352564 0.04079490
## 57   50.0  20.0 0.55352564 0.04079490
## 58   70.0  20.0 0.55352564 0.04079490
## 59   80.0  20.0 0.55352564 0.04079490
## 60  100.0  20.0 0.55352564 0.04079490
## 61    0.1  25.0 0.57910256 0.03816505
## 62    1.0  25.0 0.56371795 0.04470584
## 63    5.0  25.0 0.56115385 0.04207091
## 64   10.0  25.0 0.56115385 0.04207091
## 65   20.0  25.0 0.56115385 0.04207091
## 66   25.0  25.0 0.56115385 0.04207091
## 67   30.0  25.0 0.56115385 0.04207091
## 68   35.0  25.0 0.56115385 0.04207091
## 69   50.0  25.0 0.56115385 0.04207091
## 70   70.0  25.0 0.56115385 0.04207091
## 71   80.0  25.0 0.56115385 0.04207091
## 72  100.0  25.0 0.56115385 0.04207091
## 73    0.1  30.0 0.57910256 0.03816505
## 74    1.0  30.0 0.56628205 0.04035148
## 75    5.0  30.0 0.56371795 0.04470584
## 76   10.0  30.0 0.56371795 0.04470584
## 77   20.0  30.0 0.56371795 0.04470584
## 78   25.0  30.0 0.56371795 0.04470584
## 79   30.0  30.0 0.56371795 0.04470584
## 80   35.0  30.0 0.56371795 0.04470584
## 81   50.0  30.0 0.56371795 0.04470584
## 82   70.0  30.0 0.56371795 0.04470584
## 83   80.0  30.0 0.56371795 0.04470584
## 84  100.0  30.0 0.56371795 0.04470584
## 85    0.1  35.0 0.57910256 0.03816505
## 86    1.0  35.0 0.57397436 0.04347983
## 87    5.0  35.0 0.56628205 0.04035148
## 88   10.0  35.0 0.56628205 0.04035148
## 89   20.0  35.0 0.56628205 0.04035148
## 90   25.0  35.0 0.56628205 0.04035148
## 91   30.0  35.0 0.56628205 0.04035148
## 92   35.0  35.0 0.56628205 0.04035148
## 93   50.0  35.0 0.56628205 0.04035148
## 94   70.0  35.0 0.56628205 0.04035148
## 95   80.0  35.0 0.56628205 0.04035148
## 96  100.0  35.0 0.56628205 0.04035148
## 97    0.1  50.0 0.57910256 0.03816505
## 98    1.0  50.0 0.57910256 0.03816505
## 99    5.0  50.0 0.57397436 0.04347983
## 100  10.0  50.0 0.57397436 0.04347983
## 101  20.0  50.0 0.57397436 0.04347983
## 102  25.0  50.0 0.57397436 0.04347983
## 103  30.0  50.0 0.57397436 0.04347983
## 104  35.0  50.0 0.57397436 0.04347983
## 105  50.0  50.0 0.57397436 0.04347983
## 106  70.0  50.0 0.57397436 0.04347983
```

```
## 107  80.0  50.0 0.57397436 0.04347983
## 108 100.0  50.0 0.57397436 0.04347983
## 109   0.1  70.0 0.57910256 0.03816505
## 110   1.0  70.0 0.57910256 0.03816505
## 111   5.0  70.0 0.57910256 0.03816505
## 112  10.0  70.0 0.57910256 0.03816505
## 113  20.0  70.0 0.57910256 0.03816505
## 114  25.0  70.0 0.57910256 0.03816505
## 115  30.0  70.0 0.57910256 0.03816505
## 116  35.0  70.0 0.57910256 0.03816505
## 117  50.0  70.0 0.57910256 0.03816505
## 118  70.0  70.0 0.57910256 0.03816505
## 119  80.0  70.0 0.57910256 0.03816505
## 120 100.0  70.0 0.57910256 0.03816505
## 121   0.1  80.0 0.57910256 0.03816505
## 122   1.0  80.0 0.57910256 0.03816505
## 123   5.0  80.0 0.57910256 0.03816505
## 124  10.0  80.0 0.57910256 0.03816505
## 125  20.0  80.0 0.57910256 0.03816505
## 126  25.0  80.0 0.57910256 0.03816505
## 127  30.0  80.0 0.57910256 0.03816505
## 128  35.0  80.0 0.57910256 0.03816505
## 129  50.0  80.0 0.57910256 0.03816505
## 130  70.0  80.0 0.57910256 0.03816505
## 131  80.0  80.0 0.57910256 0.03816505
## 132 100.0  80.0 0.57910256 0.03816505
## 133   0.1 100.0 0.57910256 0.03816505
## 134   1.0 100.0 0.57910256 0.03816505
## 135   5.0 100.0 0.57910256 0.03816505
## 136  10.0 100.0 0.57910256 0.03816505
## 137  20.0 100.0 0.57910256 0.03816505
## 138  25.0 100.0 0.57910256 0.03816505
## 139  30.0 100.0 0.57910256 0.03816505
## 140  35.0 100.0 0.57910256 0.03816505
## 141  50.0 100.0 0.57910256 0.03816505
## 142  70.0 100.0 0.57910256 0.03816505
## 143  80.0 100.0 0.57910256 0.03816505
## 144 100.0 100.0 0.57910256 0.03816505
```

For a radial kernel, the lowest cross-validation error is obtained for a gamma of 0.1 and a cost of 10.

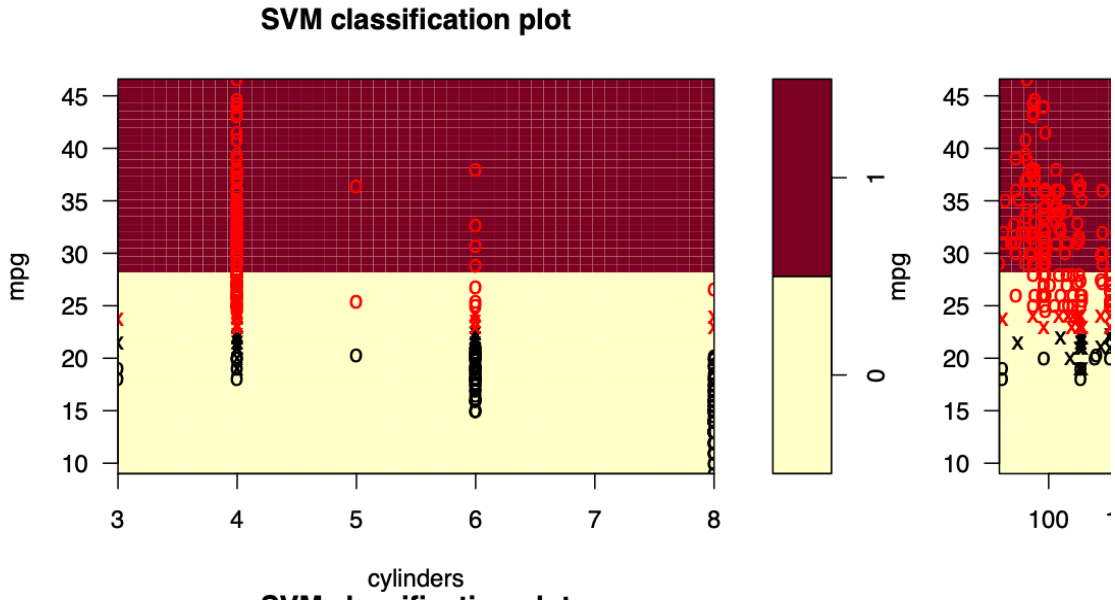
(d)

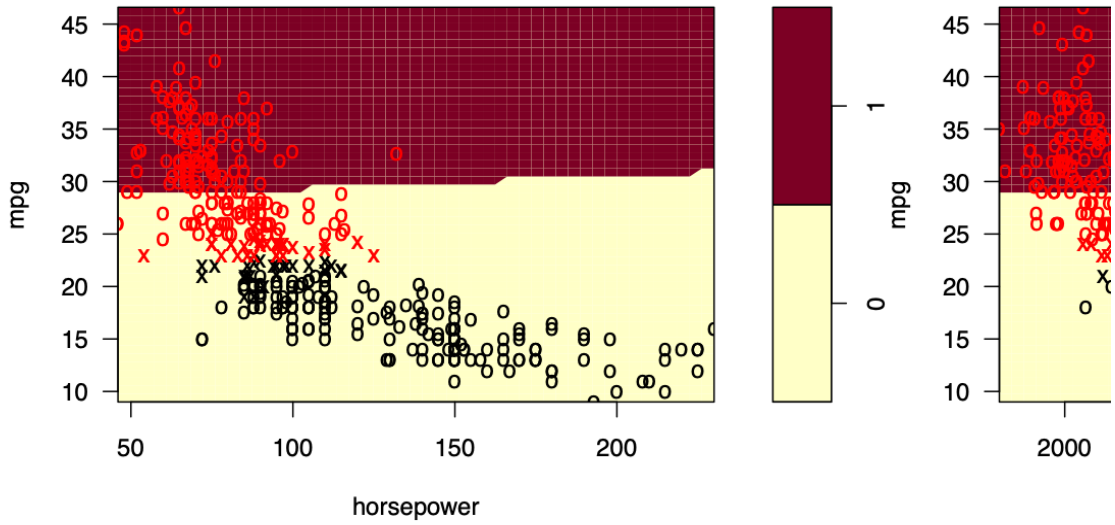
Make some plots to back up your assertions in (b) and (c). Hint: In the lab, we used the plot() function for svm objects only in cases with p=2 When p>2,you can use the plot() function to create plots displaying pairs of variables at a time. Essentially, instead of typing plot(svmfit , dat) where svmfit contains your fitted model and dat is a data frame containing your data, you can type plot(svmfit , dat, x1~x4) in order to plot just the first and fourth variables. However, you must replace x1 and x4 with the correct variable names. To find out more, type ?plot.svm.

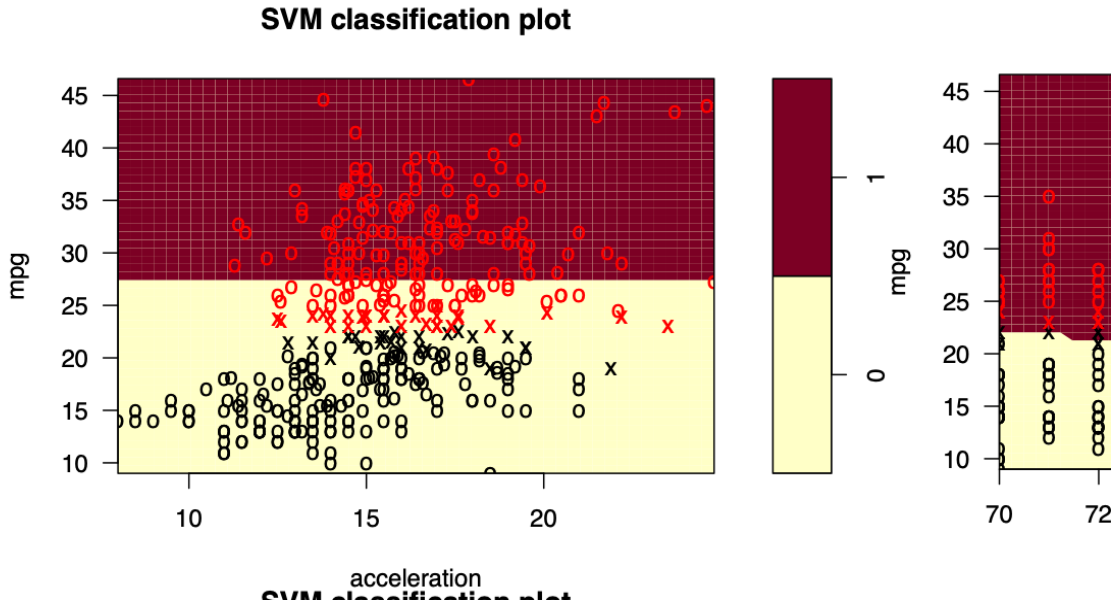
```
svm.linear = svm(mpglevel ~ ., data = Auto, kernel = "linear", cost = 1)
svm.pl = svm(mpglevel ~ ., data = Auto, kernel = "polynomial", cost = 80,degree = 2)
svm.rd = svm(mpglevel ~ ., data = Auto, kernel = "radial", cost = 10, gamma = 0.1)
plots = function(x) {
      for (name in names(Auto)[!(names(Auto) %in% c("mpg", "mpglevel", "name"))]) {
```

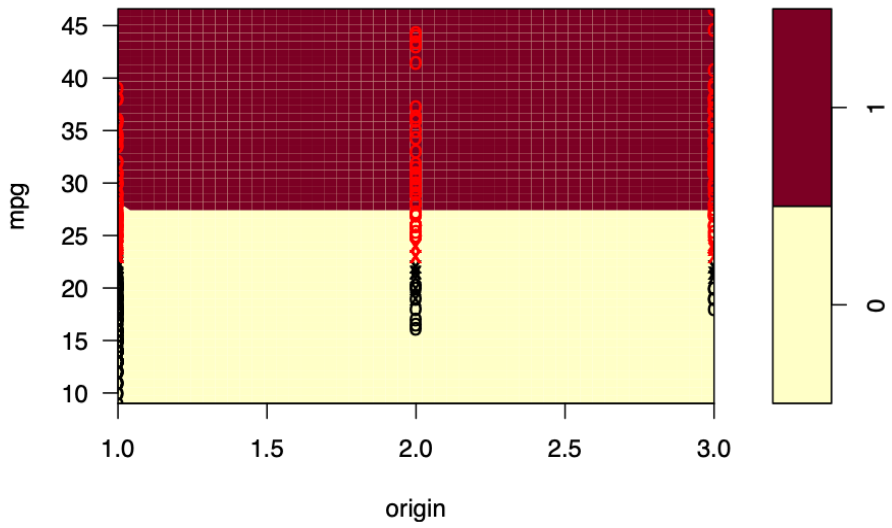
```
            plot(x, Auto, as.formula(paste("mpg~", name, sep = "")))
        }
}
```

```
plots(svm.linear)
```

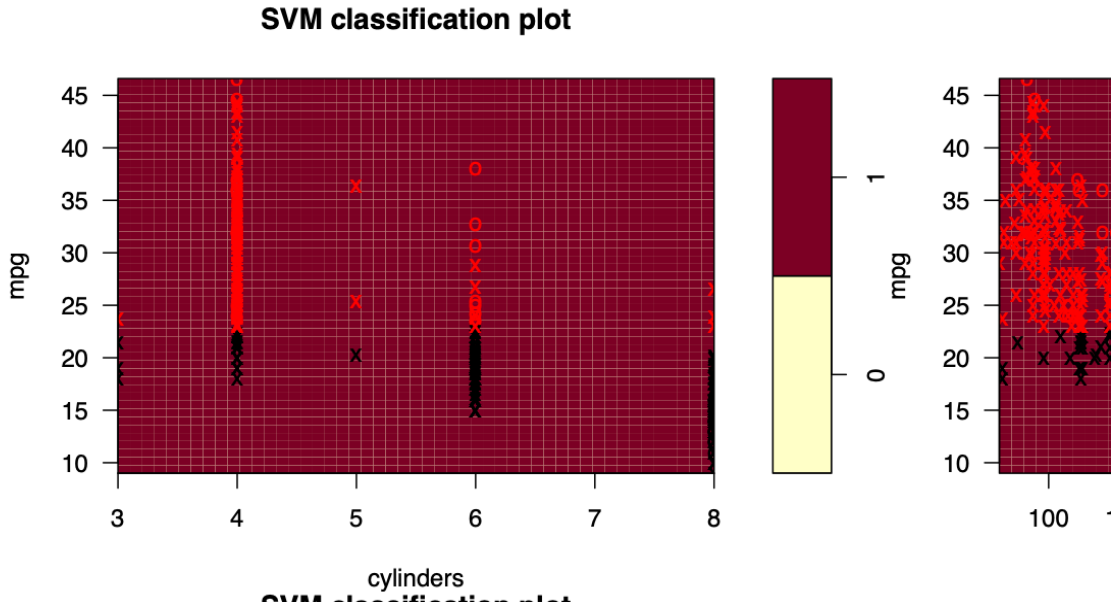


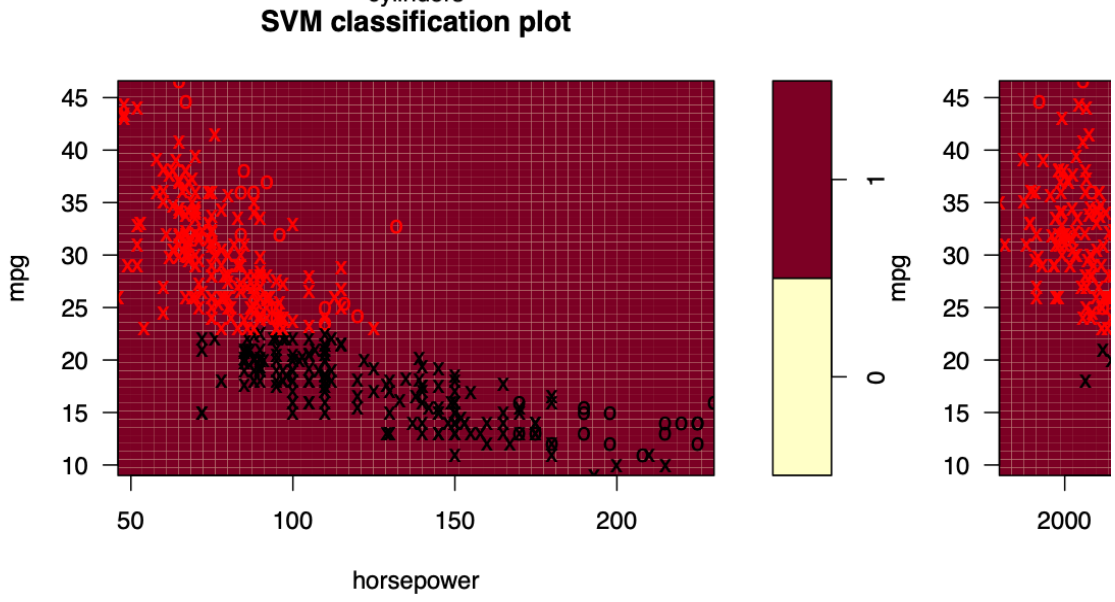


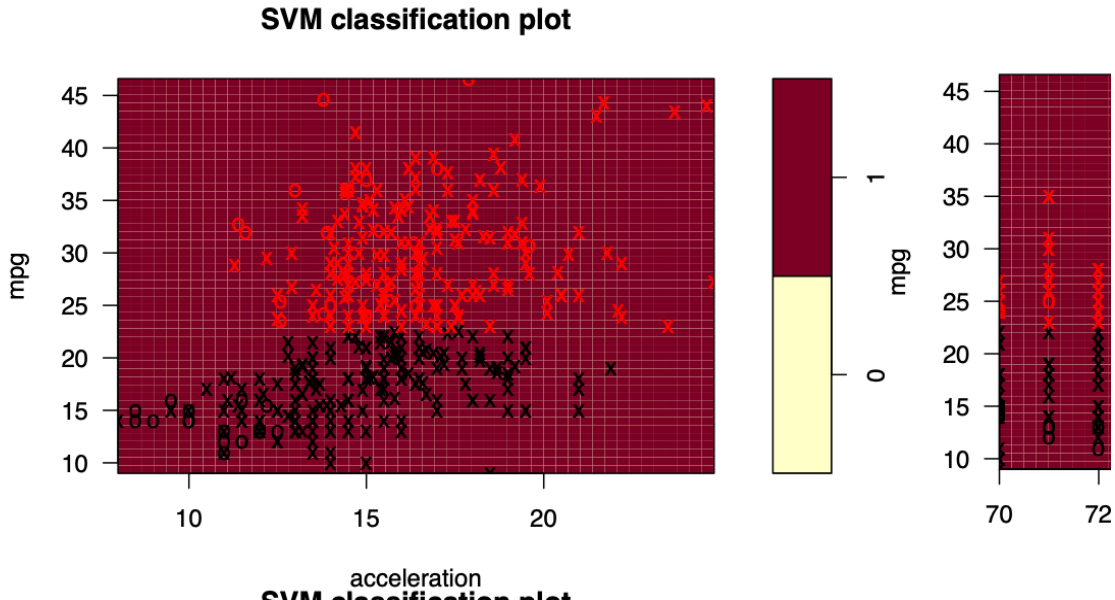

SVM classification plot
mpg
acceleration
mpg

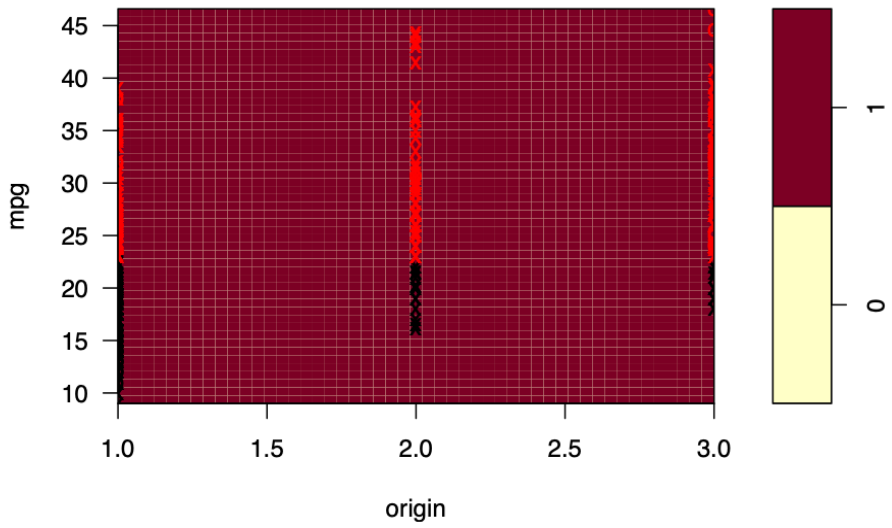

SVM classification plot
mpg
origin

```
plots(svm.pl)
```

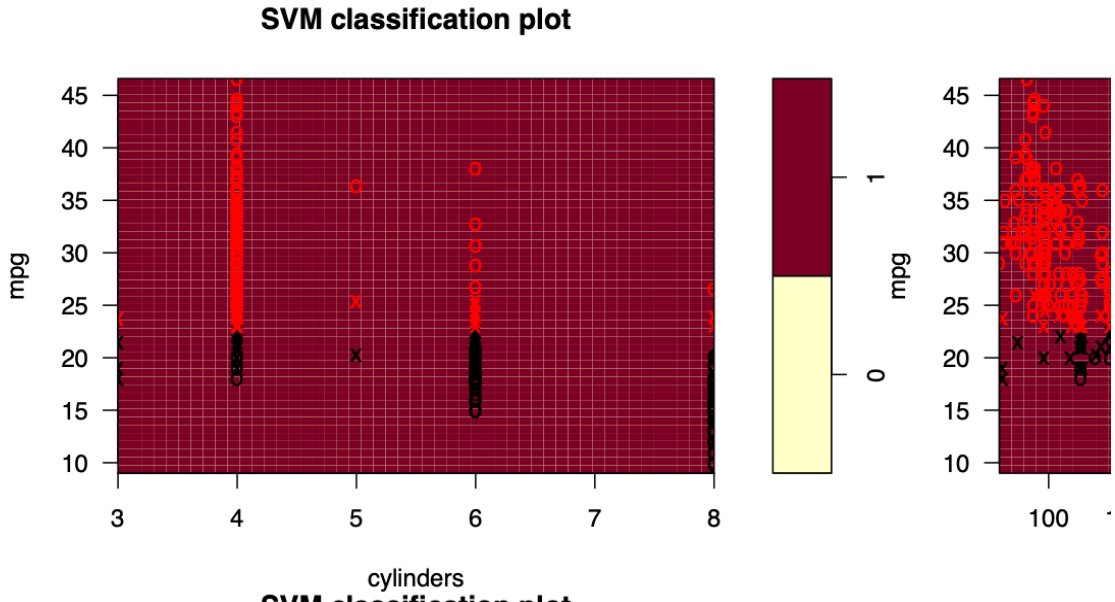

SVM classification plot
mpg
cylinders
mpg
100

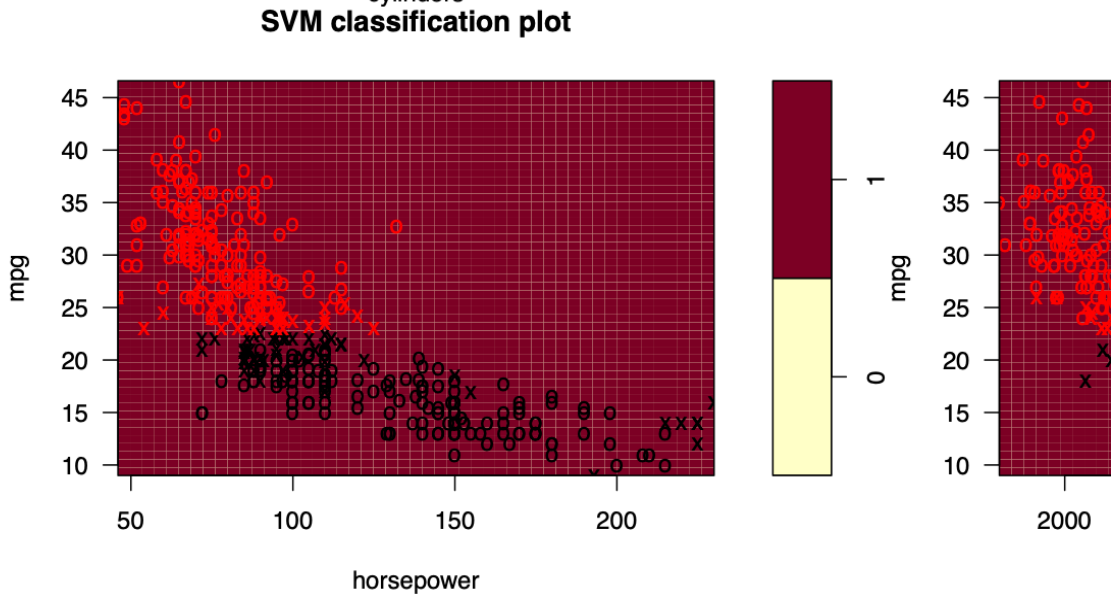

SVM classification plot
mpg
horsepower
mpg
2000

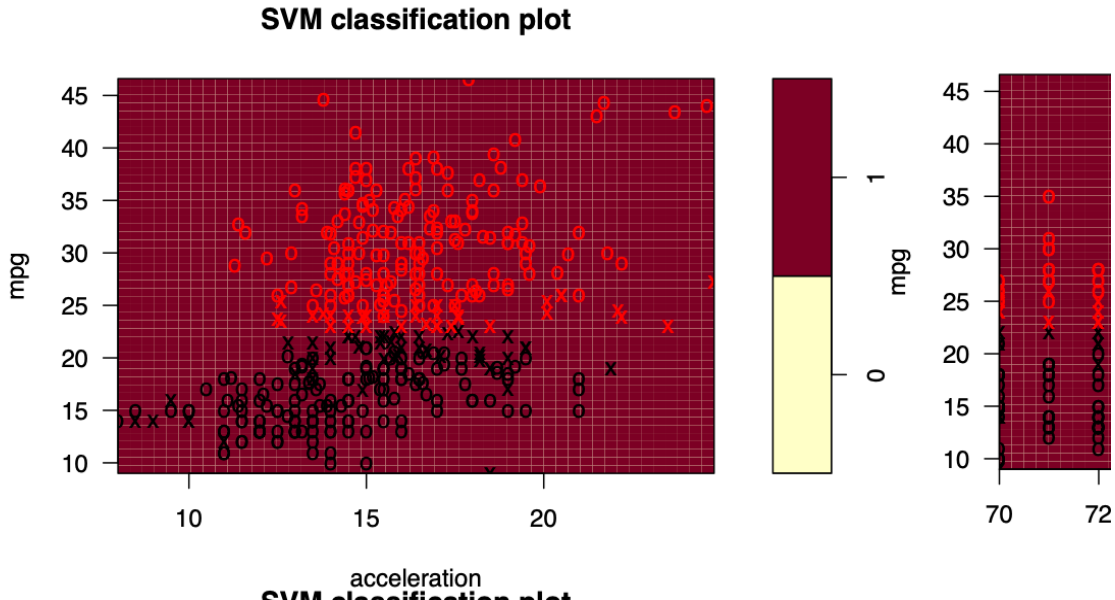

SVM classification plot
mpg
acceleration
mpg

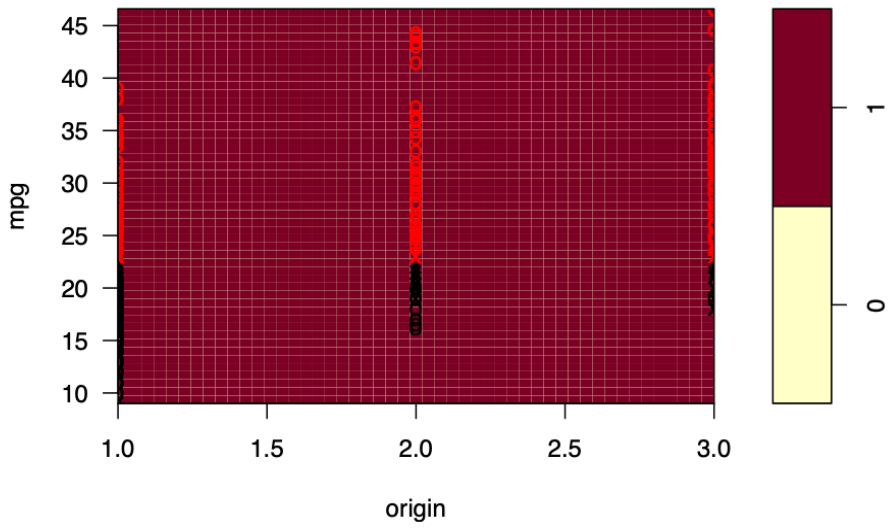

SVM classification plot
mpg
origin

```
plots(svm.rd)
```


SVM classification plot
mpg
cylinders
mpg


SVM classification plot
mpg
horsepower
mpg


SVM classification plot
mpg
acceleration
mpg
45
40
35
30
25
20
15
10
10
15
20
1
0
70
72


SVM classification plot
mpg
origin
45
40
35
30
25
20
15
10
1.0
1.5
2.0
2.5
3.0
1
0