Econ 573 Assignment 3

Harvey Duperier

2022-10-04

Part I

Question 2

2a). The lasso, relative to least squares, is: *iii.* Less flexible and hence will give improved prediction accuracy when its increase in bias is less than its decrease in variance. This is because the lasso can yield a reduction in variance when compared to least squares in exchange for a small increase in bias, consistently generating more accurate predictions, also making it easier to interpret, making *iii* the correct choice.

2b). The ridge regression, relative to least squares, is: *iii.* Less flexible and hence will give improved prediction accuracy when its increase in bias is less than its decrease in variance. This is because, similarly to lasso, the ridge regression can yield a reduction in variance, when compared to least squares, in exchange for a small increase in bias. The relationship between λ and variance and bias is important: when it increases, the flexibility of the ridge regression decreases which causes decreased variance, but increased bias, making *iii* the correct choice again.

2c). Non-linear methods, relative to least squares, is: *ii.* More flexible and hence will give improved prediction accuracy when its increase in variance is less than its decrease in bias. Contrasting to ridge and lasso methods, non-linear methods work in the opposite way, giving increased prediction accuracy when a decrease in bias gives way to an increase in variance, making *ii* the correct choice.

Question 3

3a). As we increase s from 0, the training RSS will: *iv.* Steadily decrease. As we begin to increase s from 0, all β's will increase from 0 to their least square estimate values. The training RSS for β's at 0 will be the maximum and trend downward to the original least squares RSS; therefore, *iv* is the correct choice.

3b). As we increase s from 0, the test RSS will: *ii.* Decrease initially, and then eventually start increasing in a U shape. When s=0 and all β's are 0, the model is extremely simple and because of that, has a high test RSS. Beginning to increase s, β s will begin to assume non-zero values and the model begins to fit better, so test RSS originally decreases. Eventually, β s will approach their OLS values, and as they begin to over fit the training data, test RSS will begin to increase again, forming a U shape and making *ii* the correct choice.

3c). As we increase s from 0, the variance will: *iii.* Steadily Increase. When s=0, the model basically predicts a constant and has almost no variance, but as we increase s, the model includes more β's, and their values will begin to increase. As the values of βs become increasingly more dependent on training data, the variance will steadily increase, making *iii* the correct choice.

3d). As we increase s from 0, the (squared) bias will: *iv.* Steadily Decrease. As we stated in the previous example, when s=0, the model basically predicts a constant, so the prediction is far from the actual value, and (squared) bias is high. As we increase s from 0 though, more β's become non-zero, and the model continues to fit the training data better, thus making bias steadily decrease, and proving *iv* to be the correct choice.

3e). As we increase s from 0, the irreducible error will: v. Remain Constant. Irreducible error is model dependent and therefore increasing s from 0 will not change it, making it remain constant and proving v to be the best choice.

Question 10

10a).

```
require(tidyverse)
```

```
## Loading required package: tidyverse
```

```
## -- Attaching packages --------------------------------------- tidyverse 1.3.2 --
## v tibble  3.1.8      v dplyr   1.0.10
## v tidyr   1.2.1      v stringr 1.4.1
## v readr   2.1.3      v forcats 0.5.2
## v purrr   0.3.4
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x tidyr::expand() masks Matrix::expand()
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
## x tidyr::pack()   masks Matrix::pack()
## x dplyr::select() masks MASS::select()
## x tidyr::unpack() masks Matrix::unpack()
```

```
set.seed(1)
df <- data.frame(replicate(20, rnorm(n = 1000)))

df %>%
    reduce(function(y, x) y + ifelse(runif(1) < 0.5,rnorm(1, mean = 5, sd = 1), 0)*x + rnorm(1000)) -> 
```

10b).

```
require(caret)
```

```
## Loading required package: caret
```

```
## Loading required package: lattice
```

```
##
## Attaching package: 'caret'
```

```
## The following object is masked from 'package:purrr':
##
##     lift
```

```
## The following object is masked from 'package:pls':
##
##     R2
```

```r
inTrain <- createDataPartition(df$Y, p = 0.1, list = F)

x_train <- df[inTrain, -21]
y_train <- df[inTrain, 21]
x_test <- df[-inTrain, -21]
y_test <- df[-inTrain, 21]
```

10c).

```r
require(leaps); require(ggplot2); require(dplyr); require(ggthemes)
```

```
## Loading required package: ggthemes
```

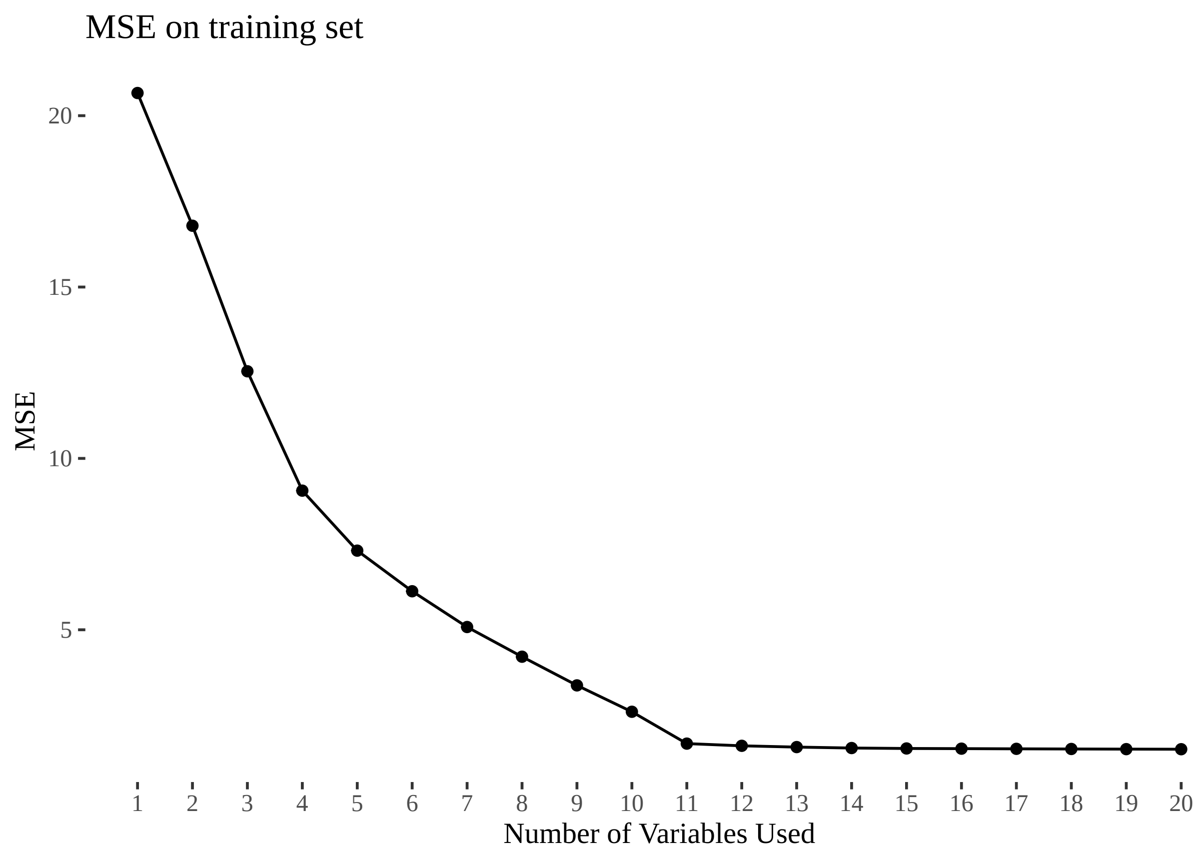
```r
best_set <- regsubsets(x = x_train, y = y_train, nvmax = 20)

best_set_summary <- summary(best_set)

data_frame(MSE = best_set_summary$rss/900) %>%
    mutate(id = row_number()) %>%
    ggplot(aes(id, MSE)) +
    geom_line() + geom_point(type = 9) +
    xlab('Number of Variables Used') +
    ggtitle('MSE on training set') +
    theme_tufte() +
    scale_x_continuous(breaks = 1:20)
```

```
## Warning: `data_frame()` was deprecated in tibble 1.1.0.
## Please use `tibble()` instead.
```

```
## Warning: Ignoring unknown parameters: type
```

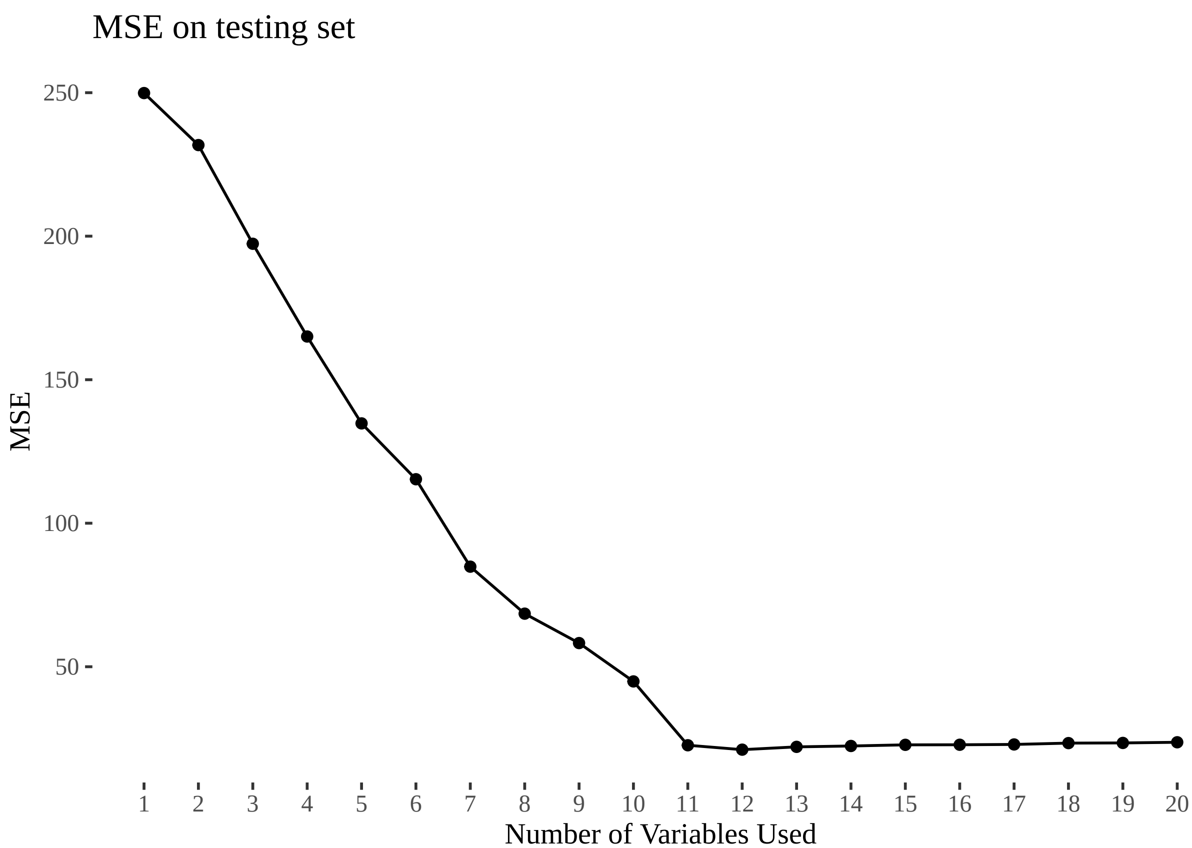


10d).

```
test_errors = rep(NA,19)
test.mat <- model.matrix(Y ~ ., data = df[-inTrain,])
for (i in 1:20){
        coefs = coef(best_set, id=i)
        pred = test.mat[,names(coefs)]%*%coefs
        test_errors[i] = mean((y_test-pred)^2)
}


data_frame(MSE = test_errors) %>%
    mutate(id = row_number()) %>%
    ggplot(aes(id, MSE)) +
    geom_line() + geom_point(type = 9) +
    xlab('Number of Variables Used') +
    ggtitle('MSE on testing set') +
    theme_tufte() +
    scale_x_continuous(breaks = 1:20)
```

```
## Warning: Ignoring unknown parameters: type
```



Ran out of time on 10 and couldn't get a function working.

Question 11

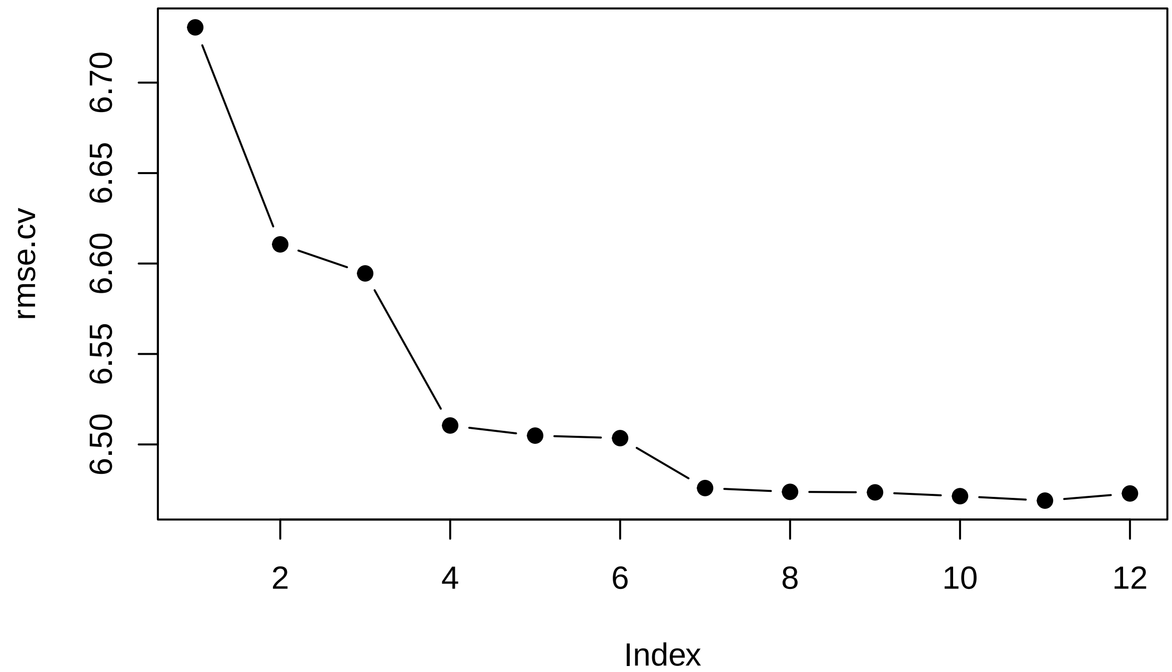
11a).

Best Subset Selection

```
predict.regsubsets = function(object, newdata, id, ...) {
  formTest = as.formula(object$call[[2]])
  mat = model.matrix(formTest, newdata)
  coefi = coef(object, id = id)
  mat[, names(coefi)] %*% coefi
}

k = 10
p = ncol(Boston) - 1
folds = sample(rep(1:k, length = nrow(Boston)))
cv.errors = matrix(NA, k, p)
for (i in 1:k) {
  bestFit = regsubsets(crim ~ ., data = Boston[folds != i, ], nvmax = p)
  for (j in 1:p) {
    pred = predict(bestFit, Boston[folds == i, ], id = j)
    cv.errors[i, j] = mean((Boston$crim[folds == i] - pred)^2)
```

```r
  }
}
rmse.cv = sqrt(apply(cv.errors, 2, mean))
plot(rmse.cv, pch = 19, type = "b")
```




```r
summary(bestFit)
```

```
## Subset selection object
## Call: regsubsets.formula(crim ~ ., data = Boston[folds != i, ], nvmax = p)
## 12 Variables  (and intercept)
##         Forced in Forced out
## zn          FALSE      FALSE
## indus       FALSE      FALSE
## chas        FALSE      FALSE
## nox         FALSE      FALSE
## rm          FALSE      FALSE
## age         FALSE      FALSE
## dis         FALSE      FALSE
## rad         FALSE      FALSE
## tax         FALSE      FALSE
## ptratio     FALSE      FALSE
## lstat       FALSE      FALSE
## medv        FALSE      FALSE
## 1 subsets of each size up to 12
## Selection Algorithm: exhaustive
```

```
##            zn  indus chas nox rm  age dis rad tax ptratio lstat medv
## 1  ( 1 )  " " " "   " "  " " " " " " " " "*" " " " "     " "   " "
## 2  ( 1 )  " " " "   " "  " " " " " " " " "*" " " " "     "*"   " "
## 3  ( 1 )  " " " "   " "  " " " " " " " " "*" " " " "     "*"   "*"
## 4  ( 1 )  "*" " "   " "  " " " " " " "*" "*" " " " "     " "   "*"
## 5  ( 1 )  "*" "*"   " "  " " " " " " "*" "*" " " " "     " "   "*"
## 6  ( 1 )  "*" "*"   " "  " " " " " " "*" "*" " " " "     "*"   "*"
## 7  ( 1 )  "*" " "   " "  "*" " " " " "*" "*" " " "*"     "*"   "*"
## 8  ( 1 )  "*" "*"   " "  "*" " " " " "*" "*" " " "*"     "*"   "*"
## 9  ( 1 )  "*" "*"   " "  "*" "*" " " "*" "*" " " "*"     "*"   "*"
## 10  ( 1 ) "*" "*"   "*"  "*" "*" " " "*" "*" " " "*"     "*"   "*"
## 11  ( 1 ) "*" "*"   "*"  "*" "*" " " "*" "*" "*" "*"     "*"   "*"
## 12  ( 1 ) "*" "*"   "*"  "*" "*" "*" "*" "*" "*" "*"     "*"   "*"
```

```r
which.min(rmse.cv)
```

```
## [1] 11
```

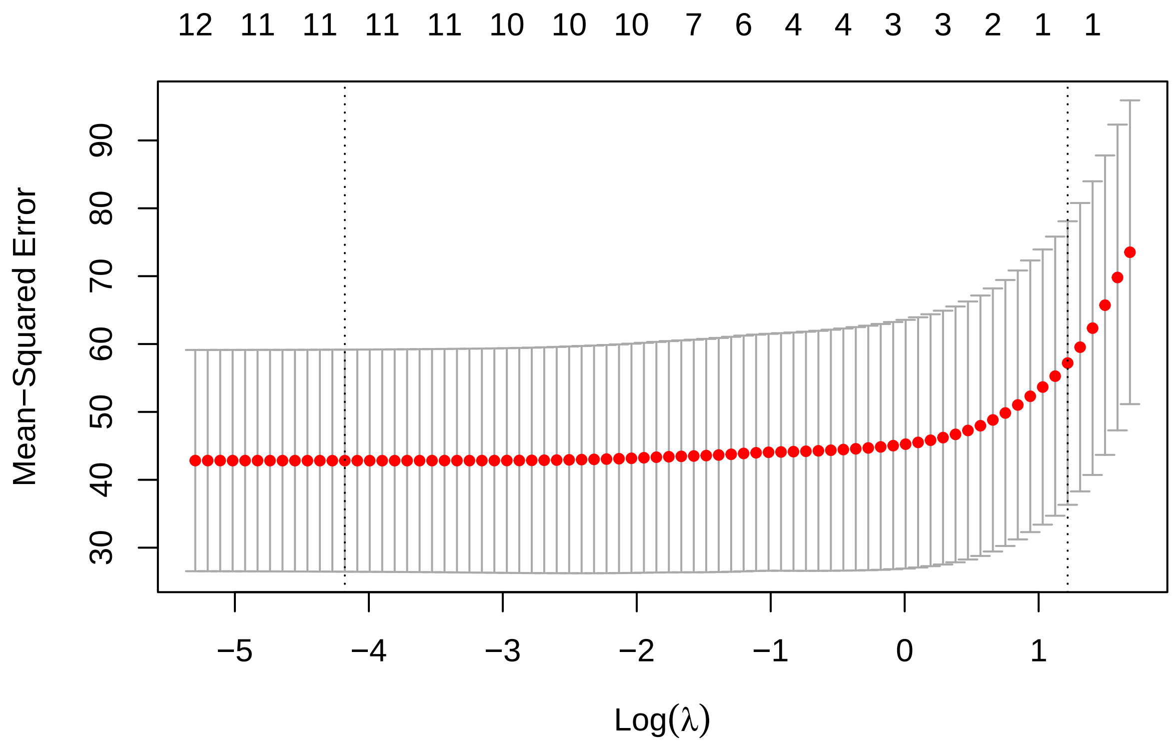
```r
bostonBSMErr=(rmse.cv[which.min(rmse.cv)])^2
bostonBSMErr
```

```
## [1] 41.84723
```

Cross-validation selects a 10-variable model based on the Test MSE. At 9-variables, the CV estimate for the test MSE is 42.82544–the lowest MSE reported.

The Lasso

```r
bostonX=model.matrix(crim~., data=Boston)[,-1]
bostonY=Boston$crim
bostonLasso=cv.glmnet(bostonX, bostonY, alpha=1, type.measure = "mse")
plot(bostonLasso)
```



To predict the training model on the test model, I need to find the lambda that reduces error the most.

```
coef(bostonLasso)
```

```
## 13 x 1 sparse Matrix of class "dgCMatrix"
##                    s1
## (Intercept) 1.4186414
## zn          .
## indus       .
## chas        .
## nox         .
## rm          .
## age         .
## dis         .
## rad         0.2298449
## tax         .
## ptratio     .
## lstat       .
## medv        .
```

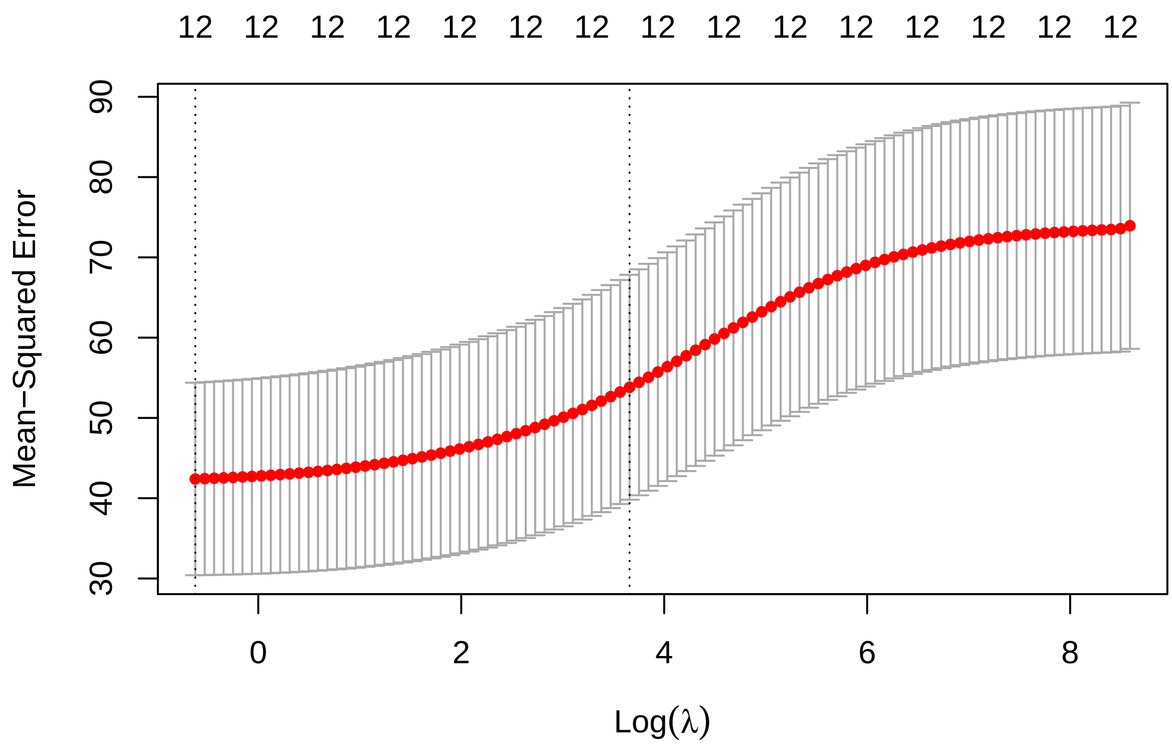
```
bostonLassoErr<-(bostonLasso$cvm[bostonLasso$lambda==bostonLasso$lambda.1se])
bostonLassoErr
```

```
## [1] 57.20346
```

Lasso is only a variable reduction method and because of this, the lasso model that reduces the MSE only includes 1 variable (rad) and has an MSE of 56.87152.

Ridge Regression

```
bostonRidgeReg=cv.glmnet(bostonX, bostonY, type.measure = "mse", alpha=0)
plot(bostonRidgeReg)
```




```
coef(bostonRidgeReg)
```

```
## 13 x 1 sparse Matrix of class "dgCMatrix"
##                        s1
## (Intercept) -1.167059217
## zn          -0.002191230
## indus        0.040017365
## chas        -0.310277874
## nox          2.705525035
## rm          -0.173778611
## age          0.008554200
## dis         -0.139397337
## rad          0.078666548
## tax          0.003411767
## ptratio      0.102335368
## lstat        0.055804532
## medv        -0.036555208
```

```
bostonRidgeErr<-bostonRidgeReg$cvm[bostonRidgeReg$lambda==bostonRidgeReg$lambda.lse]
bostonRidgeErr
```

```
## numeric(0)
```

Ridge Regression attempts to keep all variables unlike the Lasso method. Compared to the Best Subset Selection and the Lasso, the Ridge Regression does not perform well.

PCR

```
bostonPCR = pcr(crim~., data=Boston, scale=TRUE, validation="CV")
summary(bostonPCR)
```

```
## Data:    X dimension: 506 12
##  Y dimension: 506 1
## Fit method: svdpc
## Number of components considered: 12
##
## VALIDATION: RMSEP
## Cross-validated using 10 random segments.
##        (Intercept)  1 comps  2 comps  3 comps  4 comps  5 comps  6 comps
## CV            8.61    7.254    7.247    6.837    6.822    6.770    6.755
## adjCV         8.61    7.252    7.246    6.833    6.820    6.768    6.753
##        7 comps  8 comps  9 comps  10 comps  11 comps  12 comps
## CV       6.624    6.630    6.612     6.604     6.546     6.466
## adjCV    6.616    6.627    6.608     6.600     6.542     6.462
##
## TRAINING: % variance explained
##       1 comps  2 comps  3 comps  4 comps  5 comps  6 comps  7 comps  8 comps
## X       49.93    63.64    72.94    80.21    86.83    90.26    92.79    94.99
## crim    29.39    29.55    37.39    37.85    38.85    39.23    41.73    41.82
##       9 comps  10 comps  11 comps  12 comps
## X       96.78     98.33     99.48    100.00
## crim    42.12     42.43     43.58     44.93
```

The most appropriate PCR model would include 10 components and that would explain 98.33% of the predictors by the model. At 10 components, MSE is 43.58224. Overall, this model works pretty well.

11b). Since the model that had the lowest cross-val error is the best subset selection model, I would propose this model as I computed above. This model also has an MSE of 42.82544.

11c). The Best Subset Selection Model only includes 10 variable as I explained above. More variation of the response would be included if the model were to include the left out features. Since we are aiming to have low variance and low MSE in the model prediction accuracy, this is the best.

Part II

Question 4

4a). On average,we will use about 10% of the available observations to make the prediction.

4b). On average, we will use about 1% of the available observations to make the prediction.

4c). On average, we will use

$$10^{-98}$$

% of the available observation to make the prediction because

$$0.10^{100}$$

*100 =

$$10^{-98}$$

%.

4d). Observations that are near any given test observation decrease exponentially as p increases linearly based off my observations from parts (a)-(c). So basically, we p nears infinity, the percent of available observations we use to make the predictions approaches 0.

4e). i). p=1, length = 0.10 **ii).** p=2, length =

$$\sqrt{0.10}$$

iii). p=100, length =

$$0.10^{1/100}$$

Question 8

Based off these results we should prefer to use the logistic regression for classification of new observations because the 1-nearest neighbors method would have a test error rate of 36% whereas the logistic regression has a lower test error rate of 30%.

Question 11

i)

$$a_k$$

=log($\frac{\pi_k}{\pi_K}$)

ii)

$$b_{kj}$$

=log($\frac{b_{kj}x_j}{b_K x_j}$)

ii)

$$c_{kjl}$$

=log($\frac{c_{kjl}x_j x_l}{c_{Kjl}x_j x_l}$)

Question 12

Ran out of time to do this.

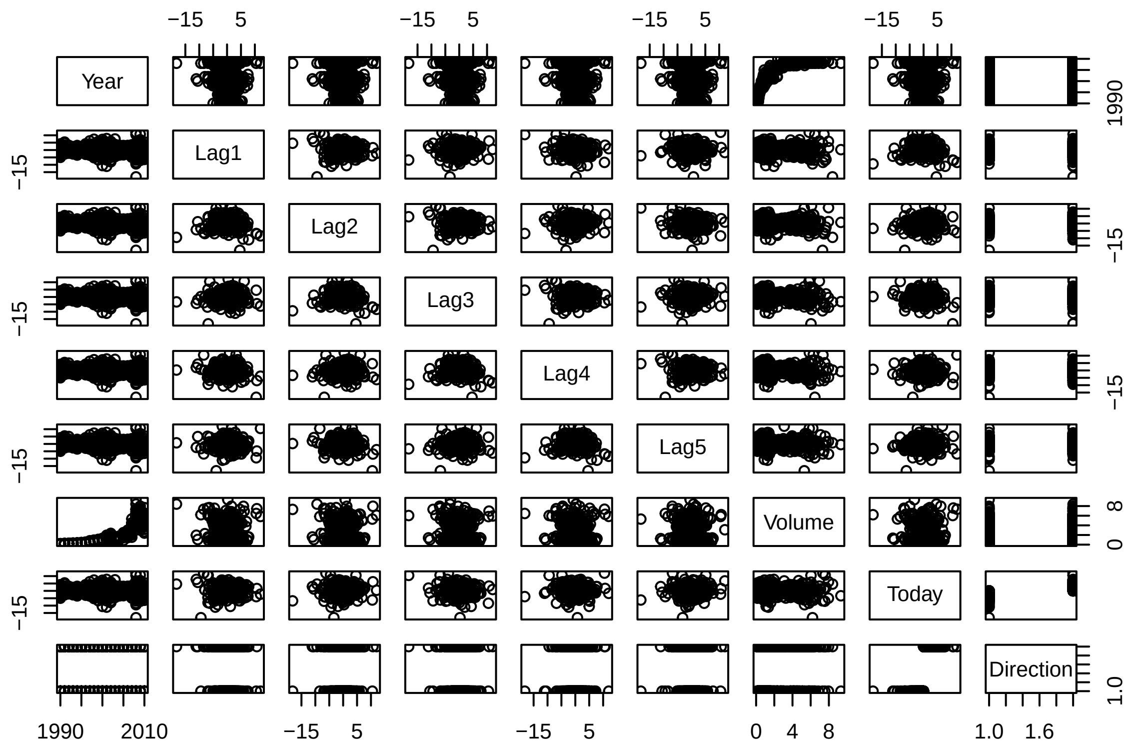
Question 13

a).

```
data(Weekly)
summary(Weekly)
```

```
##       Year           Lag1               Lag2               Lag3         
##  Min.   :1990   Min.   :-18.1950   Min.   :-18.1950   Min.   :-18.1950  
##  1st Qu.:1995   1st Qu.: -1.1540   1st Qu.: -1.1540   1st Qu.: -1.1580  
##  Median :2000   Median :  0.2410   Median :  0.2410   Median :  0.2410  
##  Mean   :2000   Mean   :  0.1506   Mean   :  0.1511   Mean   :  0.1472  
##  3rd Qu.:2005   3rd Qu.:  1.4050   3rd Qu.:  1.4090   3rd Qu.:  1.4090  
##  Max.   :2010   Max.   : 12.0260   Max.   : 12.0260   Max.   : 12.0260  
##       Lag4               Lag5              Volume            Today         
##  Min.   :-18.1950   Min.   :-18.1950   Min.   :0.08747   Min.   :-18.1950  
##  1st Qu.: -1.1580   1st Qu.: -1.1660   1st Qu.:0.33202   1st Qu.: -1.1540  
##  Median :  0.2380   Median :  0.2340   Median :1.00268   Median :  0.2410  
##  Mean   :  0.1458   Mean   :  0.1399   Mean   :1.57462   Mean   :  0.1499  
##  3rd Qu.:  1.4090   3rd Qu.:  1.4050   3rd Qu.:2.05373   3rd Qu.:  1.4050  
##  Max.   : 12.0260   Max.   : 12.0260   Max.   :9.32821   Max.   : 12.0260  
##  Direction 
##  Down:484  
##  Up  :605  
##            
##            
##            
## 
```

```
pairs(Weekly)
```




```
cor(Weekly[ ,-9])
```

```
##              Year         Lag1        Lag2        Lag3         Lag4
## Year    1.00000000 -0.032289274 -0.03339001 -0.03000649 -0.031127923
## Lag1   -0.03228927  1.000000000 -0.07485305  0.05863568 -0.071273876
## Lag2   -0.03339001 -0.074853051  1.00000000 -0.07572091  0.058381535
## Lag3   -0.03000649  0.058635682 -0.07572091  1.00000000 -0.075395865
## Lag4   -0.03112792 -0.071273876  0.05838153 -0.07539587  1.000000000
## Lag5   -0.03051910 -0.008183096 -0.07249948  0.06065717 -0.075675027
## Volume  0.84194162 -0.064951313 -0.08551314 -0.06928771 -0.061074617
## Today  -0.03245989 -0.075031842  0.05916672 -0.07124364 -0.007825873
##                Lag5      Volume        Today
## Year   -0.030519101  0.84194162 -0.032459894
## Lag1   -0.008183096 -0.06495131 -0.075031842
## Lag2   -0.072499482 -0.08551314  0.059166717
## Lag3    0.060657175 -0.06928771 -0.071243639
## Lag4   -0.075675027 -0.06107462 -0.007825873
## Lag5    1.000000000 -0.05851741  0.011012698
## Volume -0.058517414  1.00000000 -0.033077783
## Today   0.011012698 -0.03307778  1.000000000
```

There appears to be a positive correlation between the Voluma and Year variables.

b).

```r
glmf=glm(Direction~Lag1+Lag2+Lag3+Lag4+Lag5+Volume, data = Weekly, family = binomial)
summary (glmf)
```

```
##
## Call:
## glm(formula = Direction ~ Lag1 + Lag2 + Lag3 + Lag4 + Lag5 +
##     Volume, family = binomial, data = Weekly)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.6949  -1.2565   0.9913   1.0849   1.4579
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept)  0.26686    0.08593   3.106   0.0019 **
## Lag1        -0.04127    0.02641  -1.563   0.1181
## Lag2         0.05844    0.02686   2.175   0.0296 *
## Lag3        -0.01606    0.02666  -0.602   0.5469
## Lag4        -0.02779    0.02646  -1.050   0.2937
## Lag5        -0.01447    0.02638  -0.549   0.5833
## Volume      -0.02274    0.03690  -0.616   0.5377
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 1496.2  on 1088  degrees of freedom
## Residual deviance: 1486.4  on 1082  degrees of freedom
## AIC: 1500.4
##
## Number of Fisher Scoring iterations: 4
```

There isnt a variable that shows great significance, but Lag2 shows a little.

c).

```r
glmprob.wk = predict(glmf, type = "response")
glmpred.wk = rep("Down", length(glmprob.wk))
glmpred.wk[glmprob.wk > 0.5] <- "Up"

table(glmpred.wk, Weekly$Direction)
```

```
##
## glmpred.wk Down  Up
##       Down   54  48
##       Up    430 557
```

```r
mean(glmpred.wk == Weekly$Direction)
```

```
## [1] 0.5610652
```

The model is telling us that about 56% of the responses in the market are correctly predicted.

d).

```r
train=(Weekly$Year<2009)
weekly09=Weekly[!train ,]
direction09=Weekly$Direction[!train]
dim(weekly09)
```

```
## [1] 104   9
```

```r
glm_fit=glm(Direction~Lag2, data = Weekly,family=binomial ,subset=train)
glm_probability=predict (glm_fit,weekly09, type="response")
glm_prediction=rep("Down",104)
glm_prediction[glm_probability >.5]=" Up"
table(glm_prediction ,direction09)
```

```
##                direction09
## glm_prediction Down Up
##            Up    34 56
##           Down    9  5
```

This is telling use that we correctly predicted the response of the market about 62.5% of the time.

e).

```r
ldafit=lda(Direction~Lag2 ,data = Weekly ,subset=train)
ldafit
```

```
## Call:
## lda(Direction ~ Lag2, data = Weekly, subset = train)
##
## Prior probabilities of groups:
##      Down        Up
## 0.4477157 0.5522843
##
## Group means:
##             Lag2
## Down -0.03568254
## Up    0.26036581
##
## Coefficients of linear discriminants:
##            LD1
## Lag2 0.4414162
```

```r
lda.prediction=predict(ldafit , weekly09)
names(lda.prediction)
```

```
## [1] "class"     "posterior" "x"
```

```r
ldaclass=lda.prediction$class
table(ldaclass , direction09)
```

```
##         direction09
## ldaclass Down Up
##     Down    9  5
##     Up     34 56
```

Using the LDA method created the same results as the method used in part d.

f).

```
weeklyqda=qda(Direction~Lag2 ,data=Weekly ,subset=train)
weeklyqda
```

```
## Call:
## qda(Direction ~ Lag2, data = Weekly, subset = train)
##
## Prior probabilities of groups:
##      Down        Up
## 0.4477157 0.5522843
##
## Group means:
##             Lag2
## Down -0.03568254
## Up    0.26036581
```

```
classqda=predict(weeklyqda ,weekly09)$class
table(classqda ,direction09)
```

```
##         direction09
## classqda Down Up
##     Down    0  0
##     Up     43 61
```

Using the qda model, we correctly predicted the response about 58.65% of the time.

g).

```
trainX=cbind(Weekly$Lag2)[train ,]
testX=cbind(Weekly$Lag2)[!train ,]
direction.train =Weekly$Direction [train]
dim(trainX)= c(985,1)
dim(testX)=c(104,1)
set.seed(1)
knnprediction=knn(trainX,testX,direction.train ,k=1)
table(knnprediction ,direction09)
```

```
##              direction09
## knnprediction Down Up
##          Down   21 30
##          Up     22 31
```

Using our KNN model, we obtained correct predictions about 50% of the time.

h).

```
nbayes=naive_bayes(Direction~Lag2 ,data=Weekly ,subset=train)
nbayes
```

```
##
## =================================== Naive Bayes ===================================
##
##  Call:
## naive_bayes.formula(formula = Direction ~ Lag2, data = Weekly,
##     subset = train)
##
## ---------------------------------------------------------------------------------
##
## Laplace smoothing: 0
##
## ---------------------------------------------------------------------------------
##
##  A priori probabilities:
##
##      Down        Up
## 0.4477157 0.5522843
##
## ---------------------------------------------------------------------------------
##
##  Tables:
##
## ---------------------------------------------------------------------------------
##  ::: Lag2 (Gaussian)
## ---------------------------------------------------------------------------------
##
## Lag2          Down          Up
##   mean -0.03568254  0.26036581
##   sd    2.19950394  2.31748546
##
## ---------------------------------------------------------------------------------
```

```
nbayes.class=predict(nbayes ,weekly09)
```

```
## Warning: predict.naive_bayes(): more features in the newdata are provided as
## there are probability tables in the object. Calculation is performed based on
## features to be found in the tables.
```

```
table(nbayes.class ,direction09)
```

```
##             direction09
## nbayes.class Down Up
##         Down    0  0
##         Up     43 61
```

Using the native Bayes model, we obtained correct predictions about 58.65% of the time which is the same performance as the qda model.

i). The method that appears the provide the best results on the data is the regression model with 62.5% of correct predictions.

j).

```
glm2=glm(Direction~Lag2:Lag3, data = Weekly,family=binomial ,subset=train)
glmprobability2=predict (glm_fit,weekly09, type="response")
glmprediction2=rep("Down",104)
glmprediction2[glmprobability2 >.5]=" Up"
table(glmprediction2 ,direction09)
```

```
##                direction09
## glmprediction2 Down Up
##             Up    34 56
##            Down    9  5
```

Looking at the relationship between Lag2 and Lag3, the glm model correctly predicted the market about 62.5% of the time.

```
lda2=lda(Direction~Lag2^2 ,data = Weekly ,subset=train)
lda2
```

```
## Call:
## lda(Direction ~ Lag2^2, data = Weekly, subset = train)
##
## Prior probabilities of groups:
##      Down        Up
## 0.4477157 0.5522843
##
## Group means:
##              Lag2
## Down -0.03568254
## Up    0.26036581
##
## Coefficients of linear discriminants:
##            LD1
## Lag2 0.4414162
```

```
ldapred2=predict(lda2 , weekly09)
names(ldapred2)
```

```
## [1] "class"     "posterior" "x"
```

```
lda2class=ldapred2$class
table(lda2class , direction09)
```

```
##          direction09
## lda2class Down Up
##      Down    9  5
##      Up     34 56
```

Squaring Lag2, the LDA model rises to correct predictions about 62.5% of the time as well.

```
qda2=qda(Direction~Lag2:Lag3 ,data=Weekly ,subset=train)
qda2
```

```
## Call:
## qda(Direction ~ Lag2:Lag3, data = Weekly, subset = train)
##
## Prior probabilities of groups:
##      Down        Up
## 0.4477157 0.5522843
##
## Group means:
##       Lag2:Lag3
## Down -0.1937158
## Up   -0.6405132
```

```
classqda2=predict(qda2 ,weekly09)$class
table(classqda2 ,direction09)
```

```
##          direction09
## classqda2 Down Up
##      Down    6  8
##      Up     37 53
```

Using the QDA model to compared Lag2 and Lag3, this rises to correct predictions obtained about 56.73% of the time.

```
Xtrain=cbind(Weekly$Lag2)[train ,]
Xtest=cbind(Weekly$Lag2)[!train ,]
Directiontrain =Weekly$Direction [train]
dim(Xtrain)= c(985,1)
dim(Xtest)=c(104,1)
set.seed(1)
knn2=knn(Xtrain,Xtest,Directiontrain ,k=15)
table(knn2 ,direction09)
```

```
##       direction09
## knn2   Down Up
##   Down   20 20
##   Up     23 41
```

After setting K=15, the KNN model rises to 58.65% of correct predictions

```
Xtrain2=cbind(Weekly$Lag2)[train ,]
Xtest2=cbind(Weekly$Lag2)[!train ,]
Directiontrain2 =Weekly$Direction [train]
dim(Xtrain2)= c(985,1)
dim(Xtest2)=c(104,1)
set.seed(1)
knn3=knn(Xtrain2,Xtest2,Directiontrain2 ,k=25)
table(knn3 ,direction09)
```

```
##       direction09
## knn3   Down Up
##   Down   19 25
##   Up     24 36
```

When setting K=25, the QDA model correctly predicted the market about 52.88% of the time.