Recitation Exercises:

1.1 Chapter 5

1.1.1 Exercise 2

a. The probability of 1 sample being selected out of n samples is 1/n. Thus, the probability of **not** selecting that 1 (first) sample is $\mathbf{1 - \frac{1}{n}}$.

b. The probability of 1 sample being selected out of n samples is 1/n. Thus, the probability of **not** selecting that 1 (second) sample is $\mathbf{1 - \frac{1}{n}}$. As bootstrapping means sampling **with** replacement, hence selection of an additional sample does not depend on the already selected sample.

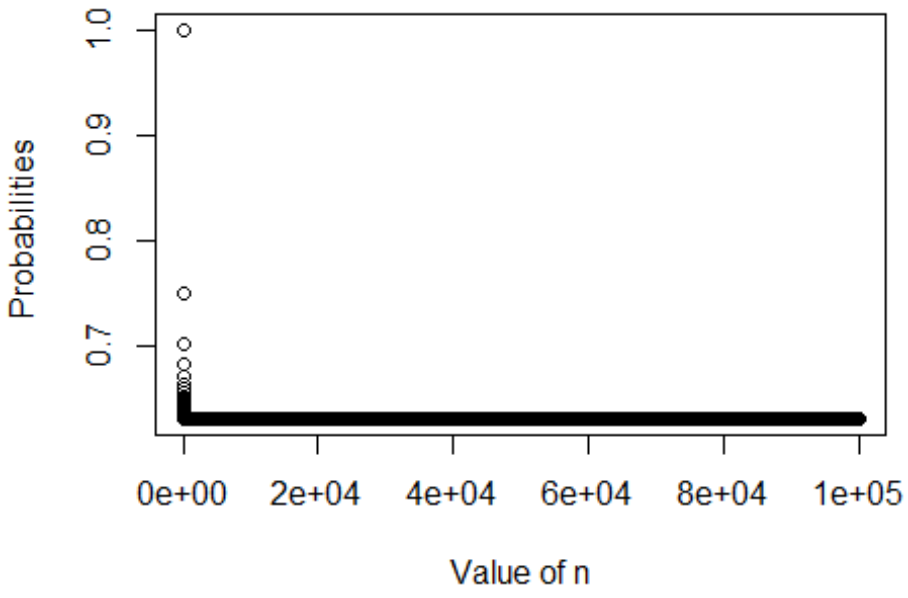
c. Since the probability of not selecting an observation is 1 – 1/n, the probability that the j^{th} observation is not in the bootstrap sample, is nothing but the product of the probabilities that each observation is not the j^{th} observation. We consider the product because the probabilities are independent of each other.
Thus, (1 – 1/n) × (1 – 1/n) × (1 – 1/n) × … up to n times = $\mathbf{(1 - 1/n)^n}$.

d. From the above step we have, probability that the j^{th} observation is **not** in the bootstrap sample is $\mathbf{(1 - 1/n)^n}$, similarly, probability that the j^{th} observation **is in** the bootstrap sample is $\mathbf{1 - (1 - 1/n)^n}$.
Thus, for n = 5, we have $1 - (1 - 1/5)^5 = \mathbf{0.67232}$.

e. Probability that the j^{th} observation **is in** the bootstrap sample is $\mathbf{1 - (1 - 1/n)^n}$.
Thus, for n = 100, we have $1 - (1 - 1/100)^{100} = \mathbf{0.63396}$.

f. Probability that the j^{th} observation **is in** the bootstrap sample is $\mathbf{1 - (1 - 1/n)^n}$.
Thus, for n = 10000, we have $1 - (1 - 1/10000)^{10000} = \mathbf{0.63214}$.

g.
```
plot(1:100000, 1-(1-1/1:100000)^(1:100000), xlab= "Value of n",ylab = "Probabilities ")
```



Probability quickly reaches to **0.632** and remains constant as $n \to \infty$.

h. On executing the give code, we get mean = 0.6329. So, we know that,

$$\lim_{n\to\infty}\left(1+\frac{x}{n}\right)^n = e^x$$

Therefore, if we apply this fact to our case, we get that, the probability that a bootstrap of size n containing the j[th] observation will converge to 1 – 1 / e = 0.632 as $n \to \infty$

1.1.2 Exercise 3

a. In k-fold Cross-Validation, the set of observations is randomly divided into k groups, or folds, of approximately equal size. The first fold is treated as a validation set, and the method is fit on the remaining $k - 1$ folds. The mean squared error, MSE_1, is then computed on the observations in the held-out fold. This procedure is repeated k times; each time, a different group of observations is treated as a validation set. This process results in k estimates of the test error, MSE_1, MSE_2, . . . , MSE_k. The k-fold CV estimate is computed by averaging these values,

$$\boldsymbol{CV_{(k)} = \frac{1}{k}\sum_{i=1}^{k} MSE_i}$$ …(For Quantitative Response)

b. Advantages and Disadvantages of k-fold cross-validation relative to:

i. Validation set approach: The validation set approach has two main drawbacks compared to k-fold cross-validation. First, the validation estimate of the test error rate can be highly variable – depending on precisely which observations are included in the training set and which observations are included in the validation set. Second, only a subset of the observations is used to fit the model. Since statistical methods tend to perform worse when trained on fewer observations, this suggests that the validation set error rate may tend to overestimate the test error rate for the model fit on the entire data set. On the other hand, the only advantage of validation set approach is the computational cost. The validation set approach has a clear benefit in this case, since the model only needs be learnt from the data once, and tested once, unlike the k-fold where the model runs for k times.

ii. LOOCV: This approach has a larger (or the same, in the case k = n) computational cost as compared to the k-fold CV since the model needs to be trained and tested n times, instead of k. Furthermore, LOOCV suffers from a higher variance in result, since they are typically highly correlated. And performing k-fold CV for, say, $k = 5$ or $k = 10$ will lead to an intermediate level of bias, since each training set contains $(k - 1)n/k$ observations fewer than in the LOOCV approach, but substantially more than in the validation set approach. Therefore, from the perspective of bias reduction, it is clear that LOOCV is to be preferred to k-fold CV.

1.2 Chapter 6

1.2.1 Exercise 1

a. The **Best Subset** selection will have the smallest training RSS as it will consider all the possible models unlike the others which has a greedy approach.

b. Can not be determined, as the given information is insufficient. Best Subset selection may have the lowest training RSS, as it considers all the models, however this may overfit the model resulting in low test RSS causing the other approaches to outperform.

c. True/False:
   i. True
   ii. True
   iii. False (Not Necessary, sometimes it may be true but not always)
   iv. False (Not Necessary, sometimes it may be true but not always)
   v. False (Best subset selection may drop previously chosen predictors when a new one is added, since all possible subsets are considered.)

1.2.2 Exercise 2

a. Option **(iii)** is correct. As Lasso performs feature selection, it substantially generates a sparse model, hence, less flexible. Also, it reduces the variance at the cost of an increase in bias. Thus, in order to improve prediction accuracy, its increase in bias should be less than its decrease in variance.

b. Option **(iii)** is correct. As Ridge performs shrinkage, it shrinks predictors that don't have as a strong relationship with the target variable, hence, less flexible. Also, it reduces the variance at the cost of an increase in bias. Thus, in order to improve prediction accuracy, its increase in bias should be less than its decrease in variance.

c. Option **(ii)** is correct. As non-linear model more tightly follows the observation as compared to the least squares, hence, more flexible. Also, it reduces the bias at the cost of an increase in variance. Thus, in order to improve prediction accuracy, its increase in variance should be less than its decrease in bias.

1.2.3 Exercise 3

a. Option (**iv**) is correct. As we increase **s**, the model becomes more and more flexible as the restriction on β is reducing, thus the coefficients increase from 0 to their least square estimate values. Thus, resulting in decreased RSS.

b. Option (**ii**) is correct. As the model is becoming more and more flexible the test RSS will reduce first and then start increasing when overfitting will start.

c. Option (**iii**) is correct. Variance steadily increase with the increase in model flexibility.

d. Option (**iv**) is correct. Bias decreases with the increase in the model flexibility.

e. Option (**v**) is correct. Irreducible error is model independent and does not depend on s.

1.2.4 Exercise 4

a. Option (**iii**) is correct. As we increase λ, the model becomes less and less flexible as the restriction on β is increasing , thus the coefficients come close to 0 from their least square estimate values. Thus, resulting in increased RSS.

b. Option (**ii**) is correct. As the model is becoming less and less flexible the test RSS will reduce first and then start increasing when overfitting will start.

c. Option (**iv**) is correct. Variance steadily decreases with the decrease in model flexibility.

d. Option (**iii**) is correct. Bias increases with the decrease in the model flexibility.

e. Option (**v**) is correct. Irreducible error is model independent and does not depend on λ.

1.2.5 Exercise 5

a. Ridge: $(n=p=2)$ $\hat{\beta}_0 = 0$ ∴,

$$\min\left[(y_1 - \hat{\beta}_1 x_{11} - \hat{\beta}_2 x_{12})^2 + (y_2 - \hat{\beta}_1 x_{21} - \hat{\beta}_2 x_{22})^2\right] + \lambda(\hat{\beta}_1^2 + \hat{\beta}_2^2)$$

b.

from a, ✍

$$= \left(y_1^2 + \hat{\beta}_1^2 x_{11}^2 + \hat{\beta}_2^2 x_{12}^2 - 2\hat{\beta}_2 x_{12} y_1 - 2\hat{\beta}_1 x_{11} y_1 + 2\hat{\beta}_1 \hat{\beta}_2 x_{11} x_{12}\right) +$$
$$\left(y_2^2 + \hat{\beta}_1^2 x_{21}^2 + \hat{\beta}_2^2 x_{22}^2 - 2\hat{\beta}_2 x_{22} y_1 - 2\hat{\beta}_1 x_{21} y_1 + 2\hat{\beta}_1 \hat{\beta}_2 x_{21} x_{22}\right) +$$
$$\lambda(\hat{\beta}_1^2 + \hat{\beta}_2^2)$$

Taking the partial derivative to $\hat{\beta}_1$ & setting eq^n to 0 to minimize.

$$\frac{\partial}{\partial \hat{\beta}_1}: \left(2\hat{\beta}_1 x_{11}^2 - 2x_{11} y_1 + 2\hat{\beta}_2 x_{11} x_{12}\right) + \left(2\hat{\beta}_1 x_{21}^2 - 2x_{21} y_2 + 2\hat{\beta}_2 x_{21} x_{22}\right) + 2\lambda\hat{\beta}_1 = 0$$

Let $x_{11} = x_{12} = x_1$ and $x_{21} = x_{22} = x_2$ and divide throughout by 2, we get,

$$= \left(\hat{\beta}_1 x_1^2 - x_1 y_1 + \hat{\beta}_2 x_1^2\right) + \left(\hat{\beta}_1 x_2^2 - x_2 y_2 + \hat{\beta}_2 x_2^2\right) + \lambda\hat{\beta}_1 = 0$$

$$= \hat{\beta}_1(x_1^2 + x_2^2) + \hat{\beta}_2(x_1^2 + x_2^2) + \lambda\hat{\beta}_1 = x_1 y_1 + x_2 y_2$$

Add $2\hat{\beta}_1 x_1 x_2$ and $2\hat{\beta}_2 x_1 x_2$ on both sides,

$$= \hat{\beta}_1(x_1^2 + x_2^2)^2 + \hat{\beta}_2(x_1^2 + x_2^2)^2 + \lambda\hat{\beta}_1 = x_1 y_1 + x_2 y_2 + 2\hat{\beta}_1 x_1 x_2 + 2\hat{\beta}_2 x_1 x_2$$

As, $x_1 + x_2 = 0$ ... Given,

$$\therefore \quad \lambda\hat{\beta}_1 = x_1 y_1 + x_2 y_2 + 2\hat{\beta}_1 x_1 x_2 + 2\hat{\beta}_2 x_1 x_2 \quad \text{—(1)}$$

Similarly by taking partial derivative to $\hat{\beta}_2$, we get

$$\lambda\hat{\beta}_2 = x_1 y_1 + x_2 y_2 + 2\hat{\beta}_1 x_1 x_2 + 2\hat{\beta}_2 x_1 x_2 \quad -(2)$$

∴ from (1) and (2) $\boxed{\hat{\beta}_1 = \hat{\beta}_2}$ as $\lambda\hat{\beta}_1 = \lambda\beta_2$

Hence proved.

c. $$\min\left[(y_1 - \hat{\beta}_1 x_{11} - \hat{\beta}_2 x_{12})^2 + (y_2 - \hat{\beta}_1 x_{21} - \hat{\beta}_2 x_{22})^2\right] + \lambda\left(|\hat{\beta}_1| + |\hat{\beta}_2|\right)$$

d. Replacing the penalty term from Ridge Regression, the derivative term to β

$$= \frac{\partial}{\partial\hat{\beta}}(\lambda|\beta|) \therefore \lambda\frac{|\beta|}{\beta}$$

Same like in ridge regression, we get

$$\frac{\lambda|\beta_1|}{\beta_1} = \frac{\lambda|\beta_2|}{\beta_2}$$

... provided β_1 and β_2 are both positive or both negative.

2 Practicum Problems:

2.1 Problem 1

```
# Problem 1
#########################################################################

library(readr)
library(data.table)
library(caret)

## Loading required package: lattice

## Loading required package: ggplot2

hd_URL = "http://archive.ics.uci.edu/ml/machine-learning-
databases/00243/yacht_hydrodynamics.data"
hd_data = fread(hd_URL, header = FALSE)

hd_Header = c("longitudinalPos","prismaticCoef","LDR","BDR","LBR","froudeNo","Residuary")
colnames(hd_data) = hd_Header

#Direct use of attribute names in code
attach(hd_data)

#Creating 80-20 Training Testing Split, createDataPartition() returns the indices
trainIndex = createDataPartition(y = hd_data$Residuary , p = 0.8, list = FALSE)

#Training data
trainData = hd_data[trainIndex,]

#Testing data (note the minus sign)
testData = hd_data[-trainIndex,]

#Training fit for linear model
linearModel1 = lm(hd_data$Residuary~hd_data$longitudinalPos + hd_data$prismaticCoef +
hd_data$LDR + hd_data$BDR + hd_data$BDR + hd_data$LBR +hd_data$froudeNo, data =
trainData)

# Function to compute MSE
MSE = function(yActual, yPred)
{
  return (mean((yActual - yPred)^2))
}
mse1 = MSE(hd_data$Residuary, linearModel1$fitted.values )

# Summarize the results
cat("Training MSE: ", mse1)

## Training MSE:  78.45015

cat("Training RMSE: ", sqrt(mse1))

## Training RMSE:  8.857209

cat("Training R-squared: ",summary(linearModel1)$r.sq)
```

```
## Training R-squared:  0.6575638
```

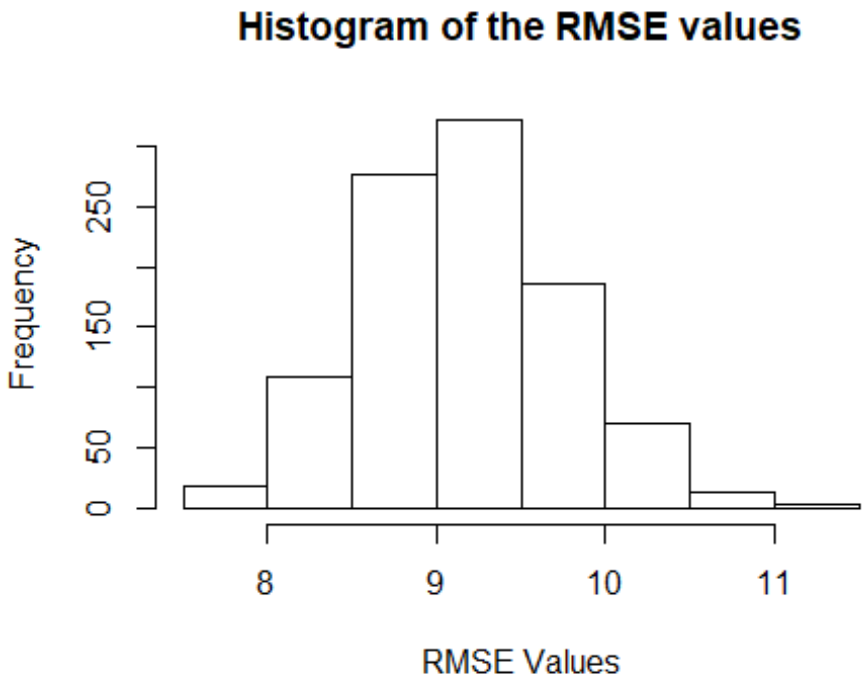
```
# Define training control
train.control =  trainControl(method = "boot", number = 1000)

# Train the model
linearModel2 = train(Residuary~., data = trainData, method = "lm", trControl =
train.control)

# 5 Point Summary for resulting RMSE for each resample
summary(linearModel2$resample$RMSE)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   7.578   8.758   9.137   9.169   9.557  11.315
```

```
# 5 Point Summary for resulting R-Squared for each resample
summary(linearModel2$resample$Rsquared)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  0.5383  0.6175  0.6378  0.6365  0.6560  0.7219
```

```
# Histogram of RMSE values
hist(linearModel2$resample$RMSE, xlab = "RMSE Values", main = "Histogram of the RMSE
values")
```




```
# Calculating the MSE from RMSE
mse2 = mean(linearModel2$resample$RMSE)^2

# Summarize the results
cat("Training Mean MSE (Bootstrap): ", mse2)
```

```
## Training Mean MSE (Bootstrap):  84.06775
```

```
cat("Training Mean RMSE (Bootstrap): ", mean(linearModel2$resample$RMSE))
```

```
## Training Mean RMSE (Bootstrap):  9.168847
```

```
cat("Training Mean R-squared (Bootstrap): ",mean(linearModel2$resample$Rsquared))
```

```
## Training Mean R-squared (Bootstrap):  0.6364541
```

```
predVals_boot = predict(linearModel2,testData)
mse3 = MSE(testData$Residuary, predVals_boot)


RSS = function (yActual, yPred)
{
  return (sum((yActual - yPred)^2))
}

TSS = function (yActual)
{
  return (sum((yActual - mean(yActual))^2))
}

rss = RSS(testData$Residuary, predVals_boot)
tss = TSS(testData$Residuary)

# Summarize the results
cat("Testing MSE (Bootstrap): ", mse3)
```

```
## Testing MSE (Bootstrap):  84.24451
```

```
cat("Testing RMSE (Bootstrap): ", sqrt(mse3))
```

```
## Testing RMSE (Bootstrap):  9.178481
```

```
cat("Testing Mean R-squared (Bootstrap): ",1 - (rss/tss))
```

```
## Testing Mean R-squared (Bootstrap):  0.6546284
```

2.2 Problem 2

```
# Problem 2
#######################################################################
library(readr)
library(data.table)
library(caret)

gcd_URL = "https://archive.ics.uci.edu/ml/machine-learning-
databases/statlog/german/german.data-numeric"
gc_data = fread(gcd_URL, header = FALSE)

#As the response variable need to be numeric, converting categorical V25 to a factor
gc_data$V25 = factor(gc_data$V25)

#Creating 80-20 Training Testing Split, createDataPartition() returns the indices
trainIndex = createDataPartition(y = gc_data$V25 , p = 0.8, list = FALSE)

#Training data
trainData = gc_data[trainIndex,]
```

```r
#Testing data (note the minus sign)
testData = gc_data[-trainIndex,]

# Creating model for y = V25 using glm
logisticModel1 = glm(V25~.,family=binomial,data=trainData)

actualVals = trainData$V25

# Using a 50% cut-off factor i.e probabilities > 0.5 are 2 and rest are 1
fittedVals = ifelse(logisticModel1$fitted.values > 0.5,2,1)
fittedVals = factor(fittedVals)

# Confusion matrix
cm = confusionMatrix(fittedVals, trainData$V25)
cm

## Confusion Matrix and Statistics
##
##           Reference
## Prediction   1   2
##          1 499 117
##          2  61 123
##
##                Accuracy : 0.7775
##                  95% CI : (0.7471, 0.8059)
##     No Information Rate : 0.7
##     P-Value [Acc > NIR] : 5.420e-07
##
##                   Kappa : 0.4324
##
##  Mcnemar's Test P-Value : 3.749e-05
##
##             Sensitivity : 0.8911
##             Specificity : 0.5125
##          Pos Pred Value : 0.8101
##          Neg Pred Value : 0.6685
##              Prevalence : 0.7000
##          Detection Rate : 0.6238
##    Detection Prevalence : 0.7700
##       Balanced Accuracy : 0.7018
##
##        'Positive' Class : 1
##

# Summarize the results
cat("Training Precision: ", cm$byClass[5] * 100, "%")

## Training Precision:  81.00649 %

cat("Training Recall: ", cm$byClass[6] * 100, "%")

## Training Recall:  89.10714 %

cat("Training F1-Score: ", cm$byClass[7] * 100, "%")

## Training F1-Score:  84.86395 %
```

```r
# Predict [By setting the parameter type='response', R will output probabilities in the form of P(y=1|X)]
probs = predict(logisticModel1, testData, type = "response")

#Using a 50% cut-off factor i.e probabilities > 0.5 are Males and rest are Females
fittedVals_test = ifelse(probs > 0.5,2,1)
fittedVals_test = factor(fittedVals_test)

# Confusion matrix
cm_test = confusionMatrix(fittedVals_test, testData$V25)
cm_test
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   1   2
##          1 131  27
##          2   9  33
##
##                Accuracy : 0.82
##                  95% CI : (0.7596, 0.8706)
##     No Information Rate : 0.7
##     P-Value [Acc > NIR] : 7.54e-05
##
##                   Kappa : 0.5312
##
##  Mcnemar's Test P-Value : 0.004607
##
##             Sensitivity : 0.9357
##             Specificity : 0.5500
##          Pos Pred Value : 0.8291
##          Neg Pred Value : 0.7857
##              Prevalence : 0.7000
##          Detection Rate : 0.6550
##    Detection Prevalence : 0.7900
##       Balanced Accuracy : 0.7429
##
##        'Positive' Class : 1
##
```

```r
# Summarize the results
cat("Testing Precision: ", cm_test$byClass[5] * 100, "%")
```

```
## Testing Precision:  82.91139 %
```

```r
cat("Testing Recall: ", cm_test$byClass[6] * 100, "%")
```

```
## Testing Recall:  93.57143 %
```

```r
cat("Testing F1-Score: ", cm_test$byClass[7] * 100, "%")
```

```
## Testing F1-Score:  87.91946 %
```

```r
# Define training control
train.control =  trainControl(method = "cv", number = 10)
```

```r
# Train the model
logisticModel2 = train(V25~., data = trainData, method = "glm", family = "binomial",
trControl = train.control)

fittedVals_cv = ifelse(logisticModel2$finalModel$fitted.values > 0.5,2,1)
fittedVals_cv = factor(fittedVals_cv)

# Confusion matrix
cm_cv = confusionMatrix(fittedVals_cv, trainData$V25)
cm_cv
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   1   2
##          1 499 117
##          2  61 123
##
##                Accuracy : 0.7775
##                  95% CI : (0.7471, 0.8059)
##     No Information Rate : 0.7
##     P-Value [Acc > NIR] : 5.420e-07
##
##                   Kappa : 0.4324
##
##  Mcnemar's Test P-Value : 3.749e-05
##
##             Sensitivity : 0.8911
##             Specificity : 0.5125
##          Pos Pred Value : 0.8101
##          Neg Pred Value : 0.6685
##              Prevalence : 0.7000
##          Detection Rate : 0.6238
##    Detection Prevalence : 0.7700
##       Balanced Accuracy : 0.7018
##
##        'Positive' Class : 1
##
```

```r
# Summarize the results
cat("Training Precision with 10-fold CV: ", cm_cv$byClass[5] * 100, "%")
```

```
## Training Precision with 10-fold CV:  81.00649 %
```

```r
cat("Training Recall with 10-fold CV: ", cm_cv$byClass[6] * 100, "%")
```

```
## Training Recall with 10-fold CV:  89.10714 %
```

```r
cat("Training F1-Score with 10-fold CV: ", cm_cv$byClass[7] * 100, "%")
```

```
## Training F1-Score with 10-fold CV:  84.86395 %
```

```r
# Predict [By setting the parameter type='response', R will output probabilities in the
form of P(y=1|X)]
probs_cv = predict(logisticModel2, testData, type = "prob")
```

```r
#Using a 50% cut-off factor i.e probabilities > 0.5 are Males and rest are Females
fittedVals_cv_test = ifelse(probs > 0.5,2,1)
fittedVals_cv_test = factor(fittedVals_test)

# Confusion matrix
cm_cv_test = confusionMatrix(fittedVals_test, testData$V25)
cm_cv_test
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   1   2
##          1 131  27
##          2   9  33
##
##                Accuracy : 0.82
##                  95% CI : (0.7596, 0.8706)
##     No Information Rate : 0.7
##     P-Value [Acc > NIR] : 7.54e-05
##
##                   Kappa : 0.5312
##
##  Mcnemar's Test P-Value : 0.004607
##
##             Sensitivity : 0.9357
##             Specificity : 0.5500
##          Pos Pred Value : 0.8291
##          Neg Pred Value : 0.7857
##              Prevalence : 0.7000
##          Detection Rate : 0.6550
##    Detection Prevalence : 0.7900
##       Balanced Accuracy : 0.7429
##
##        'Positive' Class : 1
##
```

```r
# Summarize the results
cat("Testing Precision: ", cm_cv_test$byClass[5] * 100, "%")
```

```
## Testing Precision:  82.91139 %
```

```r
cat("Testing Recall: ", cm_cv_test$byClass[6] * 100, "%")
```

```
## Testing Recall:  93.57143 %
```

```r
cat("Testing F1-Score: ", cm_cv_test$byClass[7] * 100, "%")
```

```
## Testing F1-Score:  87.91946 %
```

```r
######################################################################
```

2.3 Problem 3

```
#######################################################################
# Problem 3
data("mtcars")

# Creating 80-20 Training Testing Split, createDataPartition() returns the indices
smp_size <- floor(0.8 * nrow(mtcars))

# Setting the seed to make your partition reproducible
set.seed(123)

train_ind <- sample(seq_len(nrow(mtcars)), size = smp_size)

# Training data
trainData = mtcars[train_ind, ]

# Testing data (note the minus sign)
testData = mtcars[-train_ind, ]

# Fitting linear model
linearModel1 = lm(mpg ~ ., trainData)

# Analyze the t-stat and p-values to select relevant features
summary(linearModel1)

##
## Call:
## lm(formula = mpg ~ ., data = trainData)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -3.8774 -1.3957 -0.0511  0.8254  4.5637
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 21.48672   21.70872   0.990   0.3391
## cyl         -1.02280    1.35957  -0.752   0.4643
## disp         0.02220    0.02531   0.877   0.3952
## hp          -0.01921    0.02852  -0.674   0.5114
## drat         0.22504    1.89085   0.119   0.9070
## wt          -4.60044    2.37145  -1.940   0.0728 . (Best Attribute)
## qsec         0.83809    0.80049   1.047   0.3129
## vs           0.76966    2.46407   0.312   0.7594
## am           2.38345    2.51892   0.946   0.3601
## gear        -0.14113    1.77632  -0.079   0.9378
## carb         0.29213    1.01352   0.288   0.7774
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2.85 on 14 degrees of freedom
## Multiple R-squared:  0.8854, Adjusted R-squared:  0.8035
## F-statistic: 10.82 on 10 and 14 DF,  p-value: 5.62e-05
```

```r
# coefficient values for relevant features
linearModel1$coefficients
```

```
## (Intercept)         cyl        disp          hp        drat          wt
## 21.48672070 -1.02279987  0.02220239 -0.01921390  0.22504432 -4.60044246
##        qsec          vs          am        gear        carb
##  0.83808831  0.76966459  2.38344846 -0.14113143  0.29212936
```

```r
# Predict out-of-sample
predicted = predict(linearModel1, testData, type = "response")

# Evaluate error
actual = testData[, "mpg"]
cat("Out-of-Sample test MSE for regular linear model  = ", mean((predicted - actual)^2))
```

```
## Out-of-Sample test MSE for regular linear model  =  6.70686
```

```r
library(glmnet)
```

```
## Loading required package: Matrix
```

```
## Loading required package: foreach
```

```
## Loaded glmnet 2.0-18
```

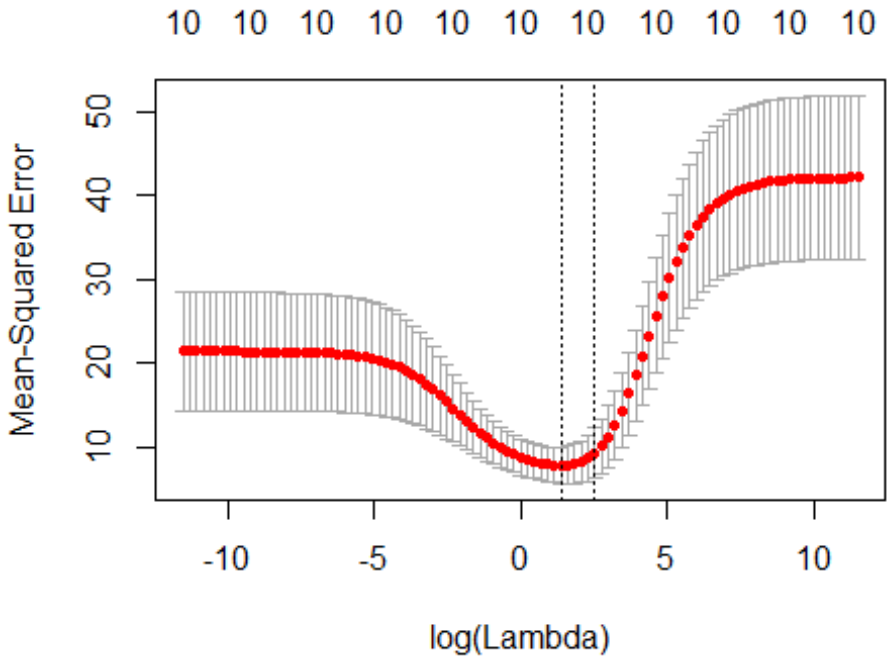
```r
# Lambda vector of 101 elements Ranging from 0 - 100000
lambda_seq = 10^seq(5, -5, by = -.1)

# Extract x and y from training data
y = trainData$mpg
x = model.matrix(mpg~. ,trainData)[,-1]

# Cross-validation to perform minimum lambda
cv_fit = cv.glmnet(x, y, alpha = 0, lambda = lambda_seq)
```

```
## Warning: Option grouped=FALSE enforced in cv.glmnet, since < 3 observations
## per fold
```

```r
opt_lambda = cv_fit$lambda.min
cat("Optimal Lambda = ",opt_lambda)
```

```
## Optimal Lambda =  3.981072
```

```r
# Fitting Ridge Regression with optimal lambda
fit = glmnet(x, y, alpha = 0, lambda = opt_lambda)

# Plot the model
plot(cv_fit)
```




```
# Coeff. of Ridge Regression
coef(fit)
```

```
## 11 x 1 sparse Matrix of class "dgCMatrix"
##                       s0
## (Intercept) 21.127822302
## cyl         -0.454357769
## disp        -0.005948233
## hp          -0.010165020
## drat         0.962887208
## wt          -1.219221002
## qsec         0.213740578
## vs           1.056859029
## am           1.553362139
## gear         0.429302573
## carb        -0.474273201
```

```
x_ridge = model.matrix(mpg~. ,testData)[,-1]

# Predicting on out-of-sample test data
predicted_rdg = predict(fit, s = opt_lambda, newx = x_ridge)


# Evaluate error
actual = testData[, "mpg"]
cat("Out-of-Sample test MSE with Ridge Regression  = ", mean((predicted_rdg - actual)^2))
```

```
## Out-of-Sample test MSE with Ridge Regression  =  4.648872
```

As we can see after the ridge regression the coefficients have shrunk and are more near to zero but none of them are perfect zero. Thus, the MSE for out-of-sample data reduces from 6.70686 to 4.648872 after the Ridge regression. Ridge regression has performed **shrinkage**.

2.4 Problem 4

```
#######################################################################
# Problem 4
data("swiss")

# Creating 80-20 Training Testing Split, createDataPartition() returns the indices
smp_size <- floor(0.8 * nrow(swiss))

# Setting the seed to make your partition reproducible
set.seed(123)

train_ind <- sample(seq_len(nrow(swiss)), size = smp_size)

# Training data
trainData = swiss[train_ind, ]

# Testing data (note the minus sign)
testData = swiss[-train_ind, ]

# Fitting linear model
linearModel1 = lm(Fertility ~ ., trainData)

# Analyze the t-stat and p-values to select relevant features
summary(linearModel1)

##
## Call:
## lm(formula = Fertility ~ ., data = trainData)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -10.8850  -3.0226   0.1069   3.3241  14.3459
##
## Coefficients:
##                  Estimate Std. Error t value Pr(>|t|)
## (Intercept)      67.96084   10.55947   6.436 3.57e-07 ***
## Agriculture      -0.22216    0.08167  -2.720 0.010593 *
## Examination      -0.22362    0.27124  -0.824 0.416003
## Education        -0.89779    0.18977  -4.731 4.64e-05 ***
## Catholic          0.13664    0.03446   3.965 0.000402 ***
## Infant.Mortality  1.13267    0.38546   2.939 0.006177 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 6.54 on 31 degrees of freedom
## Multiple R-squared:  0.7656, Adjusted R-squared:  0.7278
## F-statistic: 20.25 on 5 and 31 DF,  p-value: 6.076e-09

# coefficient values for relevant features
linearModel1$coefficients

##      (Intercept)      Agriculture      Examination        Education
##       67.9608389       -0.2221646       -0.2236157       -0.8977904
```

```
##          Catholic Infant.Mortality
##         0.1366393        1.1326696
```

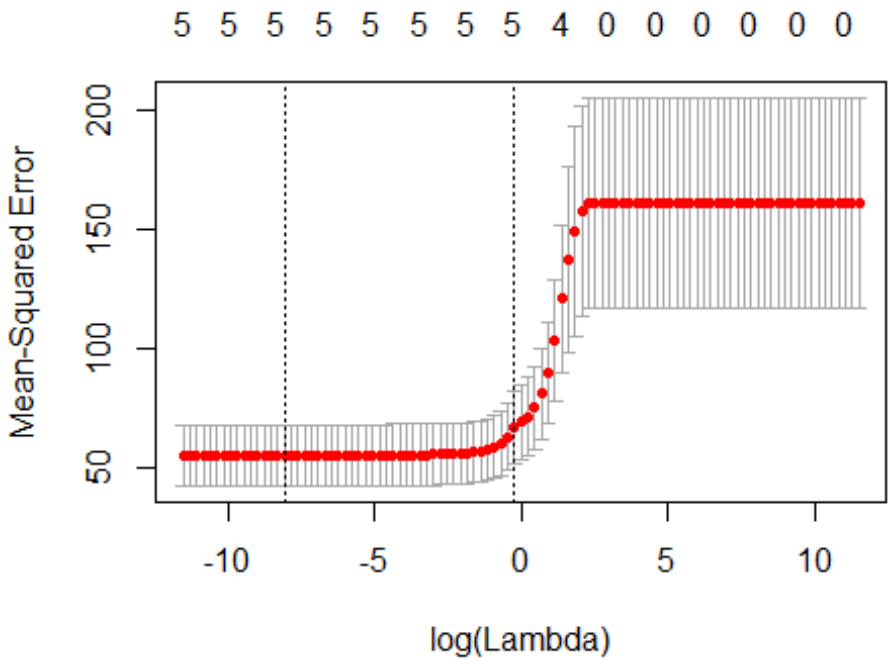
```r
# Predict out-of-sample
predicted = predict(linearModel1, testData, type = "response")

# Evaluate error
actual = testData[, "Fertility"]
cat("Out-of-Sample test MSE for regular linear model  = ", mean((predicted - actual)^2))
```

```
## Out-of-Sample test MSE for regular linear model  =  93.27207
```

```r
library(glmnet)

# Lambda vector of 101 elements Ranging from 0 - 100000
lambda_seq = 10^seq(5, -5, by = -.1)

# Extract x and y from training data
y = trainData$Fertility
x = model.matrix(Fertility~. ,trainData)[,-1]

# Cross-validation to perform minimum lambda
cv_fit = cv.glmnet(x, y, alpha = 1, lambda = lambda_seq)

opt_lambda = cv_fit$lambda.min
cat("Optimal Lambda = ",opt_lambda)
```

```
## Optimal Lambda =  0.0003162278
```

```r
# Fitting Lasso Regression with optimal lambda
fit = glmnet(x, y, alpha = 1, lambda = opt_lambda)

# Plot the model
plot(cv_fit)
```



```r
# Coeff. of Lasso Regression
coef(fit)
```

```
## 6 x 1 sparse Matrix of class "dgCMatrix"
##                              s0
## (Intercept)      67.9556030
## Agriculture      -0.2221546
## Examination      -0.2229399
## Education        -0.8981031
## Catholic          0.1366884
## Infant.Mortality  1.1324147
```

```r
x_lasso = model.matrix(Fertility~. ,testData)[,-1]

# Predicting on out-of-sample test data
predicted_lso = predict(fit, s = opt_lambda, newx = x_lasso)


# Evaluate error
actual = testData[, "Fertility"]
cat("Out-of-Sample test MSE with Lasso Regression  = ", mean((predicted_lso - actual)^2))
```

```
## Out-of-Sample test MSE with Lasso Regression  =  93.27388
```

```r
#########################################################################
```

After the Lasso, we were supposed to get some coefficient perfectly equal to zero, however we aren't getting such results, rather the coefficients have shrunk to some extent and the out-of-sample MSE has raised a little bit from 93.27207 to 93.27388. Lasso usually performs variable selection, but in this case it is performing shrinkage.