```
# Rudy Meza
# Machine Learning in R
# Michael Chang
# Spring
# '***Lab and Quiz***'
# '******************Machine Learning********************'

# 5.3 Lab: Cross-Validation and the Bootstrap
library (ISLR)
 set.seed (1)
 train=sample (392 ,196)
?sample
# The sample dataset and size of the sample

lm.fit =lm(mpg~horsepower ,data=Auto ,subset =train )
# We then use the subset option in lm() to fit a linear regression using only
# the observations corresponding to the training set.
attach (Auto)
mean((mpg -predict (lm.fit ,Auto))[-train ]^2)

# We now use the predict() function to estimate the response for all 392
# observations, and we use the mean() function to calculate the MSE of the
# 196 observations in the validation set

# Note: -train command only selects data not in the training set

# The estimated MSE from the linear regression is:
# [1] 26.14142
# The poly() command is for quadratic and cubic regressions.
lm.fit2=lm(mpg~poly(horsepower ,2) ,data=Auto ,subset =train )
mean((mpg -predict (lm.fit2 ,Auto))[-train ]^2)
# [1] 19.82259
lm.fit3=lm(mpg~poly(horsepower ,3) ,data=Auto ,subset =train )
mean((mpg -predict (lm.fit3 ,Auto))[-train ]^2)
# [1] 19.78252

# These are the two different error rates between quadratic and cubic models with the variable
horsepower.
# Different training sets will entail different error results
set.seed (2)
train=sample (392 ,196)
lm.fit =lm(mpg~horsepower ,subset =train)
mean((mpg -predict (lm.fit ,Auto))[-train ]^2)
# [1] 23.29559

lm.fit2=lm(mpg~poly(horsepower ,2) ,data=Auto ,subset =train )
 mean((mpg -predict (lm.fit2 ,Auto))[-train ]^2)
# [1] 18.90124
```

```
lm.fit3=lm(mpg~poly(horsepower ,3) ,data=Auto ,subset =train )
mean((mpg -predict (lm.fit3 ,Auto))[-train ]^2)
# [1] 19.2574

# 5.3.2 Leave-One-Out Cross-Validation
glm.fit=glm(mpg~horsepower ,data=Auto)
coef(glm.fit)
# (Intercept)  horsepower
 # 39.9358610  -0.1578447
lm.fit =lm(mpg~horsepower ,data=Auto)
coef(lm.fit)
# (Intercept)  horsepower
 # 39.9358610  -0.1578447
# Here the glm and lm commands are interchangeable, but the glm also implements the logistic regression with the family commanmd.

# The cv.lm function is part of the boot library.
library (boot)
glm.fit=glm(mpg~horsepower ,data=Auto)
cv.err =cv.glm(Auto ,glm.fit)
cv.err$delta

# [1] 24.23151 24.23114

# These are the Cross-Results from the delta variable.
# These numbers corresond to the LOOCV statistic
cv.error=rep (0,5)
for (i in 1:5){
 glm.fit=glm(mpg~poly(horsepower ,i),data=Auto)
cv.error[i]=cv.glm (Auto ,glm.fit)$delta [1]
 }
 cv.error
 # [1] 24.23151 19.24821 19.33498 19.42443 19.03321
 # The for command allows a for loop to fit polynomials for orders i=1....5

 # 5.3.3 k-Fold Cross-Validation
  # cv.glm() function can also be used to implemenmt k-Fold Cross-Validation
 set.seed (17)
cv.error.10= rep (0 ,10)
 for (i in 1:10) {
 glm.fit=glm(mpg~poly(horsepower ,i),data=Auto)
 cv.error.10[i]=cv.glm (Auto ,glm.fit ,K=10) $delta [1]
 }
cv.error.10
# [1] 24.20520 19.18924 19.30662 19.33799 18.87911 19.02103 18.89609 19.71201 18.95140
# [10] 19.50196

# 5.3.4 The Bootstrap
```

```
library(boot)
alpha.fn=function (data ,index){
X=data$X [index]
Y=data$Y [index]
return ((var(Y)-cov (X,Y))/(var(X)+var(Y) -2* cov(X,Y)))
 }

 alpha.fn(Portfolio ,1:100)
 # [1] 0.5758321

 set.seed (1)
alpha.fn(Portfolio ,sample (100 ,100 , replace =T))
# [1] 0.5963833

boot(Portfolio ,alpha.fn,R=1000)

# The final output shows that using the original data, □a = 0.5758, and that
# the bootstrap estimate for SE(□a) is 0.0886.

boot.fn=function (data ,index )
 return (coef(lm(mpg~horsepower ,data=data ,subset =index)))
boot.fn(Auto ,1:392)

# The boot.fn() function can also be used in order to create bootstrap estimates
# for the intercept and slope terms by randomly sampling from among
# the observations with replacement.

set.seed (1)
boot.fn(Auto ,sample (392 ,392 , replace =T))
boot.fn(Auto ,sample (392 ,392 , replace =T))

boot(Auto ,boot.fn ,1000)

summary (lm(mpg~horsepower ,data=Auto))$coef

boot.fn=function (data ,index )
coefficients(lm(mpg~horsepower +I( horsepower ^2) ,data=data ,
subset =index))
set.seed (1)
boot(Auto ,boot.fn ,1000)

summary (lm(mpg~horsepower +I(horsepower ^2) ,data=Auto))$coef

# ''''''''''''''''''Week 3 Quiz''''''''''''''''''''''''''

# Question 1

RNGkind(sample.kind = "Rounding")
```

```
set.seed (3)
Auto <- na.omit(Auto)
train=sample (392 ,196)
lm.fit =lm(mpg~horsepower ,subset =train)
mean((mpg -predict (lm.fit ,Auto))[-train ]^2)


lm.fit2=lm(mpg~poly(horsepower ,2) ,data=Auto ,subset =train )
 mean((mpg -predict (lm.fit2 ,Auto))[-train ]^2)


lm.fit3=lm(mpg~poly(horsepower ,3) ,data=Auto ,subset =train )
mean((mpg -predict (lm.fit3 ,Auto))[-train ]^2)

# Question 2


lm.fit6=lm(mpg~poly(horsepower ,6) ,data=Auto ,subset =train )
mean((mpg -predict (lm.fit6 ,Auto))[-train ]^2)

glm.fit6=glm(mpg~poly(horsepower,6) ,data=Auto , subset=train)
coef(glm.fit)
mean((mpg -predict (glm.fit6 ,Auto))[-train ]^2)

library (boot)
glm.fit=glm(mpg~poly(horsepower,6) ,data=Auto)
cv.err =cv.glm(Auto ,glm.fit)
cv.err$delta

# Question 3

set.seed (17)

cv.error.10= rep (0 ,10)
 for (i in 1:10) {
 glm.fit=glm(mpg~poly(horsepower ,i),data=Auto)
 cv.error.10[i]=cv.glm (Auto ,glm.fit ,K=20) $delta [1]
 }
cv.error.10

# Question 4

set.seed (2)



boot(Portfolio ,alpha.fn,R=500)
```

```
# Question 5
set.seed (2)
boot(Portfolio ,alpha.fn,R=100)

set.seed (2)
boot(Portfolio ,alpha.fn,R=5000)

# Question 6
set.seed (1)
boot.fn(Auto ,1:1000)

# Question 7
boot.fn=function (data ,index )
coefficients(lm(mpg~cylinders +I( cylinders ^2) ,data=data ,
subset =index))
set.seed (1)
boot.fn(Auto ,1:1000)
boot(Auto ,boot.fn ,1000)
```