#Lab: Cross-validations and Boostrap

In [1]:
```
library(ISLR)
```

In [2]:
```
set.seed(2)
train=sample(392,196)  #to split teh set of observations
```

In [3]:
```
#?sample
```

In [4]:
```
lm.fit=lm(mpg~horsepower,data=Auto,subset=train)
#subset option in lm() to fit a linear regression using only
#observations corresponding to the training set.
```

In [5]:
```
attach(Auto)
mean((mpg-predict(lm.fit,Auto))[-train]^2)
#predict() to estimate the resp of all 392 observations
#Mean() to find MSE of 196 observations
#-train selects observations are not in traing set
```

25.7265106448139

In [6]:
```
#estimated test MSE is 25.72.
```

In [7]:
```
#poly() to estimate test error for quad and cubic regression
lm.fit2=lm(mpg~poly(horsepower,2),data=Auto,subset=train)
mean((mpg-predict(lm.fit2,Auto))[-train]^2)
```

20.4303642741463

In [8]:
```
lm.fit3=lm(mpg~poly(horsepower ,3),data=Auto,subset=train)
mean((mpg-predict(lm.fit3,Auto))[-train]^2)
```

20.3853268638776

In [9]:
```
#error rates are 20.4 and 20.3
```

In [10]:
```
set.seed(1)
train=sample(392,196)
lmfit=lm(mpg~horsepower,subset=train)
mean((mpg-predict(lm.fit,Auto))[-train]^2)
```

22.4407743741799

In [11]:
```
lm.fit2=lm(mpg~poly(horsepower ,2),data=Auto,subset=train)
mean((mpg-predict(lm.fit2,Auto))[-train]^2)
```

18.7164594933828

In [12]:
```
lm.fit3=lm(mpg~poly(horsepower ,3),data=Auto,subset=train)
mean((mpg-predict(lm.fit3,Auto))[-train]^2)
```

18.7940067973945

In [13]:
```
#if we choose diff training set, we obtain diff errors
#error rates with lm,quad, cubic 22.44,18.71,18.79
```

#Leave-one-out cross validation(LOOCV)

In [14]:
```
library(boot)
```

In [15]:
```
glm.fit=glm(mpg~horsepower ,data=Auto)
cv.err=cv.glm(Auto,glm.fit)
cv.err$delta
#cv.glm() produces a list of components
```

24.2315135179292 24.2311440937562

In [16]:
```
#Our cross-validation estimate for the test error
#is approximately 24.23.

#we can automate this process using for loop
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
```

24.2315135179292 19.2482131244897 19.334984064029 19.4244303104302 19.0332138547041

In [17]:
```
#sharp drop in estimated test MSE b/w linear to quad but not much
#in further higher order polynomials
```

#k-fold cross validation

In [18]:
```
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]
}
```

In [19]:
```
cv.error.10
```

24.2720671232254 19.2690928085129 19.3480535605547 19.2949648229745 19.0319790002896 18.8978121056401 19.1206066690695 19.1466631054789 18.8701307442148 20.9552042280394

In [20]:
```
#computation time is much shorter than LOOCV.
# higher-order polynomial terms leads to lower test error than
#simply using a quadratic fit.
```

#Bootstrap

In [21]:
```
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)))
}
#the alpha fn returns an estimating alpha to the obvervations imde
#by index.
```

In [22]:
```
alpha.fn(Portfolio,1:100) #estimating alpha using all 100 observat

```

0.57583207459283

In [23]:
```
#recording all of the corresponding estimates for α, and computing
#to make this process automate, we use boot()
boot()
boot(Portfolio ,alpha.fn,R=1000)
```

```
Error in NROW(data): argument "data" is missing, with no default
Traceback:

1. boot()
2. NROW(data)
```

In [24]:

```
#The final output shows that using the original data, α̂ = 0.5758,
#and that the bootstrap estimate
#for SE(α̂) is 0.0905.
```

In [25]:

```
#fn takes in the Auto data set as well as a set of indices for the
#and returns the intercept and slope estimates for the linear regr

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

(Intercept)	39.9358610211705
horsepower	-0.157844733353654

In [26]:

```
#alternate way to use boot.fn to create bootstrap estimate
set.seed (1)
boot.fn(Auto,sample(392,392,replace=T))
boot.fn(Auto,sample(392,392,replace=T))
```

(Intercept)	40.3404516830189
horsepower	-0.163486837689938

(Intercept)	40.1186906449022
horsepower	-0.157706320543503

In [27]:

```
boot(Auto ,boot.fn ,1000)
```

```
ORDINARY NONPARAMETRIC BOOTSTRAP


Call:
boot(data = Auto, statistic = boot.fn, R = 1000)


Bootstrap Statistics :
      original        bias    std. error
t1* 39.9358610  0.0544513229 0.841289790
t2* -0.1578447 -0.0006170901 0.007343073
```

In [28]:

```
#This indicates that the bootstrap estimate for SE(β0) is 0.84, ar
#the bootstrap estimate for SE(β1) is 0.0074

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

	Estimate	Std. Error	t value	Pr(>|t|)
(Intercept)	39.9358610	0.717498656	55.65984	1.220362e-187
horsepower	-0.1578447	0.006445501	-24.48914	7.031989e-81

In [29]:

```
#standard error estimates for β0 and β1 obtained are
#0.717 for the intercept and 0.0064 for the slope.
```

In [30]:

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

```
ORDINARY NONPARAMETRIC BOOTSTRAP


Call:
boot(data = Auto, statistic = boot.fn, R = 1000)


Bootstrap Statistics :
        original        bias     std. error
t1* 56.900099702  3.511640e-02 2.0300222526
t2* -0.466189630 -7.080834e-04 0.0324241984
t3*  0.001230536  2.840324e-06 0.0001172164
```