489HW4

Nicholas Thompson

10/5/2021

#QUESTION 1  
set.seed(2012021)  
ad.data=read.csv("advertising.csv", header = TRUE)  
str(ad.data)

## 'data.frame': 200 obs. of 5 variables:  
## $ X : int 1 2 3 4 5 6 7 8 9 10 ...  
## $ TV : num 230.1 44.5 17.2 151.5 180.8 ...  
## $ radio : num 37.8 39.3 45.9 41.3 10.8 48.9 32.8 19.6 2.1 2.6 ...  
## $ newspaper: num 69.2 45.1 69.3 58.5 58.4 75 23.5 11.6 1 21.2 ...  
## $ sales : num 22.1 10.4 9.3 18.5 12.9 7.2 11.8 13.2 4.8 10.6 ...

#a  
ad.regmod=glm(sales~TV+radio, data = ad.data)  
summary(ad.regmod)

##   
## Call:  
## glm(formula = sales ~ TV + radio, data = ad.data)  
##   
## Deviance Residuals:   
## Min 1Q Median 3Q Max   
## -8.7977 -0.8752 0.2422 1.1708 2.8328   
##   
## Coefficients:  
## Estimate Std. Error t value Pr(>|t|)   
## (Intercept) 2.92110 0.29449 9.919 <2e-16 \*\*\*  
## TV 0.04575 0.00139 32.909 <2e-16 \*\*\*  
## radio 0.18799 0.00804 23.382 <2e-16 \*\*\*  
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## (Dispersion parameter for gaussian family taken to be 2.826975)  
##   
## Null deviance: 5417.15 on 199 degrees of freedom  
## Residual deviance: 556.91 on 197 degrees of freedom  
## AIC: 780.39  
##   
## Number of Fisher Scoring iterations: 2

#b  
loocv.err=cv.glm(ad.data,ad.regmod)$delta[1]  
loocv.err

## [1] 2.910676

#1st value is test error  
  
#c  
kfoldcv.err=cv.glm(ad.data,ad.regmod,K=10)$delta[1]  
kfoldcv.err

## [1] 2.857398

#test error of leave one out: 2.910676  
#test error of 10 fold: 2.857398  
#10 fold leads to a reductin of about 0.05 in MSE  
  
#d  
ad.kmod=glm(sales~TV+radio+newspaper, data = ad.data)  
kfold\_d.err=cv.glm(ad.data,ad.kmod,K=10)$delta[1]  
kfold\_d.err

## [1] 2.934094

#test error: 2.934094  
#According to the calculation above, the test mse was increased beyond the test mse of the leave one out approach.  
#MSE increased by about 0.2 compared to the LOOCV and about 0.07 compared to the first k-fold

#Question 1e  
#Let's create a few glms  
ad.data=mutate(ad.data,newsq=(ad.data$newspaper)^2)  
ad.data=mutate(ad.data,tvsq=(ad.data$TV)^2)  
ad.data=mutate(ad.data,radiosq=(ad.data$radio)^2)  
  
emod1=glm(sales~TV+radio+newspaper+TV\*newspaper, data = ad.data)  
emod2=glm(sales~TV+radio+newspaper+newspaper\*radio, data = ad.data)  
emod3=glm(sales~TV+radio+newspaper+newsq, data = ad.data)  
emod4=glm(sales~TV+radio+newspaper+tvsq+TV\*radio, data = ad.data)  
  
k1=cv.glm(ad.data,emod1,K=10)$delta[1]  
k2=cv.glm(ad.data,emod2,K=10)$delta[1]  
k3=cv.glm(ad.data,emod3,K=10)$delta[1]  
k4=cv.glm(ad.data,emod4,K=10)$delta[1]  
k1;k2;k3;k4

## [1] 2.793901

## [1] 3.060134

## [1] 3.047494

## [1] 0.4144222

#it doesn't seem that higher level terms produce lower mean squared errors  
#of these 4 models, the one with TV\*newspaper produced the lower mse. Let's see if using this interaction in other models will produce better results  
  
emod5=glm(sales~TV+radio+TV\*newspaper, data = ad.data)  
k5=cv.glm(ad.data,emod5,K=10)$delta[1]  
k5

## [1] 2.846937

#removing newspaper did not reduce MSE  
emod6=glm(sales~TV+radio+newspaper+TV\*newspaper+TV\*radio,data = ad.data)  
k6=cv.glm(ad.data,emod6,K=10)$delta[1]  
k6

## [1] 0.969608

#adding 2 interactions proved effective  
emod7=glm(sales~TV+radio+newspaper+TV\*newspaper+TV\*radio+tvsq, data = ad.data)  
k7=k4=cv.glm(ad.data,emod7,K=10)$delta[1]  
k7

## [1] 0.4200533

emod8=glm(sales~TV+radio+newspaper+TV\*newspaper+TV\*radio+tvsq+radiosq, data = ad.data)  
k8=k4=cv.glm(ad.data,emod8,K=10)$delta[1]  
k8

## [1] 0.4124719

#of the models tested, emod8 is the "best" model that produces the lowest MSE of 0.4124719

#QUESTION 2  
set.seed(2102021)  
#a  
str(Boston)

## 'data.frame': 506 obs. of 14 variables:  
## $ crim : num 0.00632 0.02731 0.02729 0.03237 0.06905 ...  
## $ zn : num 18 0 0 0 0 0 12.5 12.5 12.5 12.5 ...  
## $ indus : num 2.31 7.07 7.07 2.18 2.18 2.18 7.87 7.87 7.87 7.87 ...  
## $ chas : int 0 0 0 0 0 0 0 0 0 0 ...  
## $ nox : num 0.538 0.469 0.469 0.458 0.458 0.458 0.524 0.524 0.524 0.524 ...  
## $ rm : num 6.58 6.42 7.18 7 7.15 ...  
## $ age : num 65.2 78.9 61.1 45.8 54.2 58.7 66.6 96.1 100 85.9 ...  
## $ dis : num 4.09 4.97 4.97 6.06 6.06 ...  
## $ rad : int 1 2 2 3 3 3 5 5 5 5 ...  
## $ tax : num 296 242 242 222 222 222 311 311 311 311 ...  
## $ ptratio: num 15.3 17.8 17.8 18.7 18.7 18.7 15.2 15.2 15.2 15.2 ...  
## $ black : num 397 397 393 395 397 ...  
## $ lstat : num 4.98 9.14 4.03 2.94 5.33 ...  
## $ medv : num 24 21.6 34.7 33.4 36.2 28.7 22.9 27.1 16.5 18.9 ...

mean(Boston$medv)

## [1] 22.53281

#22.53281  
  
#b  
sampsd=sd(Boston$medv)  
nrow(Boston)

## [1] 506

sampsd/sqrt(506)

## [1] 0.4088611

#0.4088611  
  
#c  
bootmu <- function(data, indices) {  
 mean(data[indices])  
}  
# bootstrapping with 500 replications  
results <- boot(data=Boston$medv, statistic=bootmu,  
 R=500)  
results

##   
## ORDINARY NONPARAMETRIC BOOTSTRAP  
##   
##   
## Call:  
## boot(data = Boston$medv, statistic = bootmu, R = 500)  
##   
##   
## Bootstrap Statistics :  
## original bias std. error  
## t1\* 22.53281 0.006714625 0.4014805

#standard error from bootstrap 500: 0.4014805  
  
#I want to try a bootstrap of 1000 replications  
boot(data=Boston$medv, statistic=bootmu,  
 R=1000)

##   
## ORDINARY NONPARAMETRIC BOOTSTRAP  
##   
##   
## Call:  
## boot(data = Boston$medv, statistic = bootmu, R = 1000)  
##   
##   
## Bootstrap Statistics :  
## original bias std. error  
## t1\* 22.53281 0.009722332 0.4082346

#0.4082346  
  
#using higher replications obtained a closer match to the result from b  
  
#d  
t.test(Boston$medv)

##   
## One Sample t-test  
##   
## data: Boston$medv  
## t = 55.111, df = 505, p-value < 2.2e-16  
## alternative hypothesis: true mean is not equal to 0  
## 95 percent confidence interval:  
## 21.72953 23.33608  
## sample estimates:  
## mean of x   
## 22.53281

#[21.72953,23.33608]  
#use type norm, use the results from 500 replications  
boot.ci(results, type = "norm")

## BOOTSTRAP CONFIDENCE INTERVAL CALCULATIONS  
## Based on 500 bootstrap replicates  
##   
## CALL :   
## boot.ci(boot.out = results, type = "norm")  
##   
## Intervals :   
## Level Normal   
## 95% (21.74, 23.31 )   
## Calculations and Intervals on Original Scale

#[21.74,23.31]  
#bootstrap confidence interval has a greater value for the lower endpoint but lower value for higher endpoint  
  
#e  
  
median(Boston$medv)

## [1] 21.2

favstats(Boston$medv)

## min Q1 median Q3 max mean sd n missing  
## 5 17.025 21.2 25 50 22.53281 9.197104 506 0

#21.2  
  
#f  
bmed <- function(data, indices) {  
 median(data[indices])  
}  
  
boot.med=boot(data = Boston$medv, statistic = bmed, R=500)  
boot.med

##   
## ORDINARY NONPARAMETRIC BOOTSTRAP  
##   
##   
## Call:  
## boot(data = Boston$medv, statistic = bmed, R = 500)  
##   
##   
## Bootstrap Statistics :  
## original bias std. error  
## t1\* 21.2 -0.0214 0.3749443

#standard error of the sample median: 0.3749443