**به نام خدا**

****

**Statistical learning**

**Assignment 2**

**Mohammad hasan shammakhi**

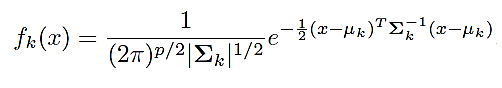
**محمد حسن شماخی**

**93123053**

**‹The Elements of Statistical Learning›**

**Chapter4 Exercise2:**

**a)**

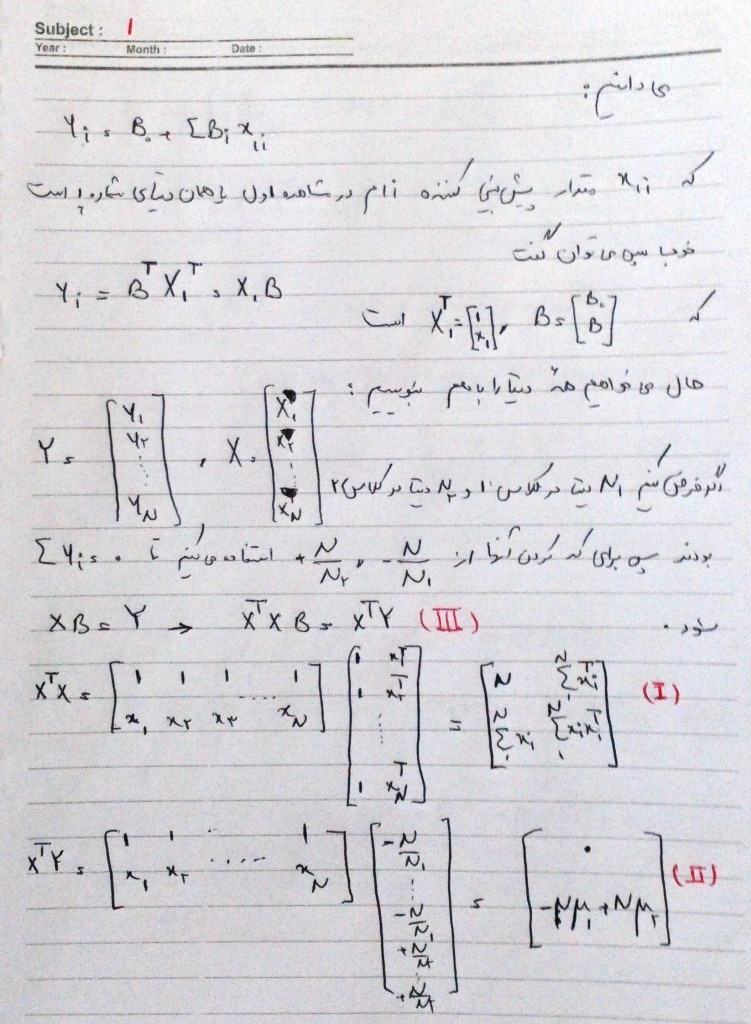
****

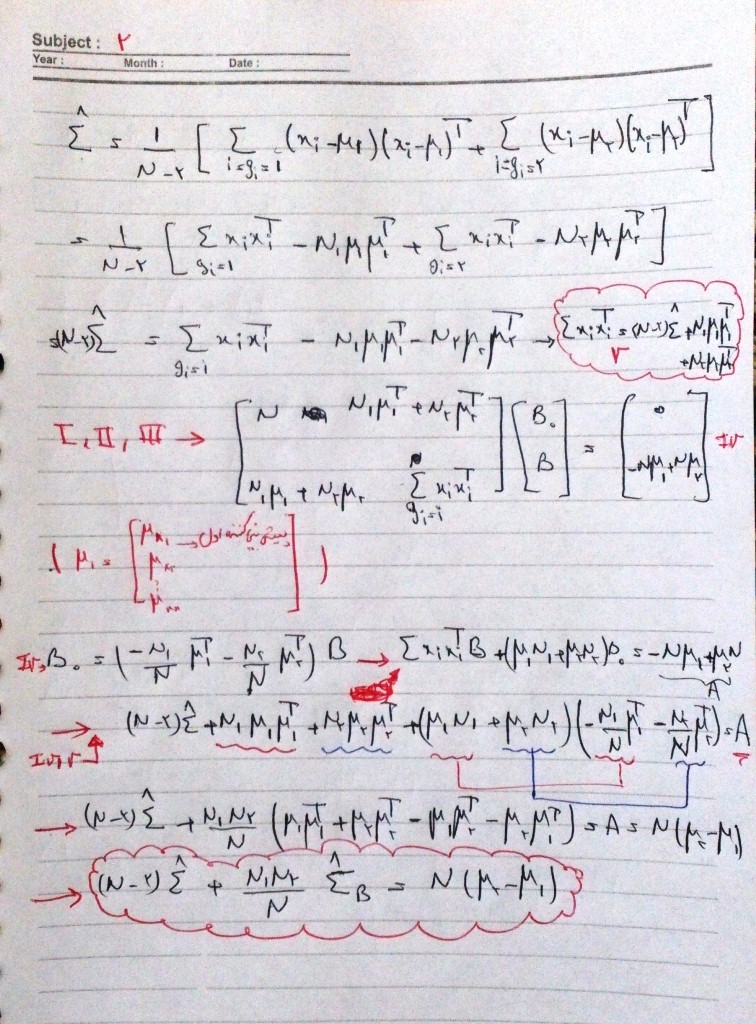
A

**اون معادله با اسم معادله 3 از این جا میاد که اگر A ماتریسی 1\*1 باشه یعنی عدد باشه پس مطمئنا ترانهاده آن با خودش برابر هست.**

**b)**

**با توجه به حجم بالای فرمول های این قسمت بجای تایپ کردن عکس انداختم.**

****

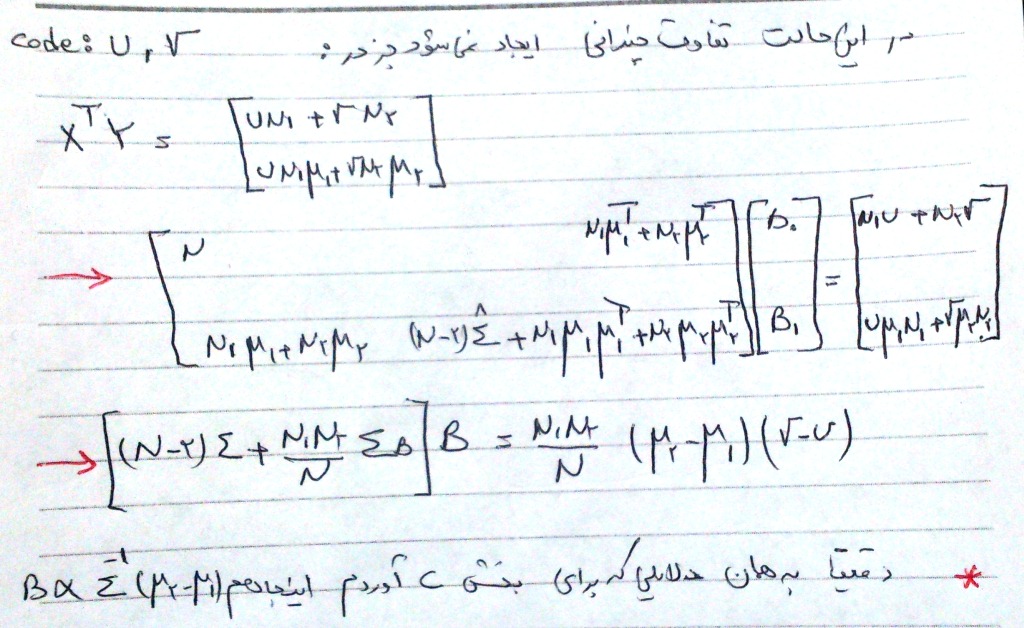
****

**C) با توجه به اینکه داریم:**

**و اینکه ماتریسی 1\*n و ماتریسی n\*1 است پس عدد می باشد لذا هم جهت با بوده و ضریبی از آن می باشد.**

**بنابراین:**

**d)**

****

**Chapter4** **Question 4:**

برای یک پیش بینی کننده فضا در دسترس به این صورت خواهد بود که دقیقا هر میزان از فضای موجود را انتخاب کنیم همان میزان از داده ها در دسترس خواهد بود یعنی مثلا وقتی از بازه 0.5 از داده ها یعنی (0.55-0.45) را به عنوان تست انتخاب کنیم دقیقا بر 10 % از داده ها تست صورت می­گیرد. اما در فضای با دو پیش بینی کننده با انتخاب بازه 0.35 و 0.85 برای دو پیش بینی کننده در واقع 1 % از داده ها مورد استفاده قرار می گیرد. پس:

**a)** با بکارگیری بازه 10 % بدیهی است 10 % از دیتا به عنوان تست انتخاب می شود

**b)** با بکارگیری بازه 10 % از هر پیش بینی کننده 1 % از دیتا به عنوان تست انتخاب می شود.

**c)** با بکارگیری بازه 10 % از هر پیش بینی کننده 10 به توان منفی 98% از دیتا به عنوان تست انتخاب می شود. که عملا هیچ دیتایی را با احتمال بالا شامل نمی شود.

**e)** برای بکارگیری داده به عنوان تست 10% آن ها مناسب می باشد که معادل 10 % از مقادیر ممکن برای عوامل پیش بینی کننده است

**حال اگر دو پیش بینی کننده داشته باشیم و بخواهیم 10 % از دیتا را به عنوان تست استفاده کنیم پس می توان گفت بنابراین پس ازهر پیش بینی کننده 32 % از آنها را انتخاب کرده و به عنوان داده تست در نظر می گیریم.**

**حال اگر 100 پیش بینی کننده داشته باشیم و بخواهیم 10 % از دیتا را به عنوان تست استفاده کنیم پس می توان گفت بنابراین پس ازهر پیش بینی کننده 98 % از آنها یعنی عملا کل دیتا را باید به عنوان تست انتخاب کنیم که کارامد نمی باشد.**

**Chapter4 Question 5:**

a) با اینکه سیستم مدل خطی دارد اما بخاطر انعطاف بیشتر QDA ممکن است خودش رو بر داده ها تا حدی fit کند لذا بر روی داده های آموزشی ممکن است جواب بهتری بدهد لذا نمی توان برروی داده های آموزشی را با یقین گفت اما بر روی دیتای تست LDA پاسخ بهتری خواهد داد.(فرض: تعداد دیتا به اندازه کافی باشد)

b)برای سیستم های غیر خطی با توجه به دیتا ممکن است LDA یا QDA کارآمد شود .

c) در سیستم های غیر خطی به نظر می آید با افزایش تعداد نمونه ها سیستم واقعی خود را بیشتر نشان دهد و از تصادفی بودن کارآمدی مدل پیشنهاد خارج شود لذا احتمالا پاسخ QDA بهبود خواهد یافت.

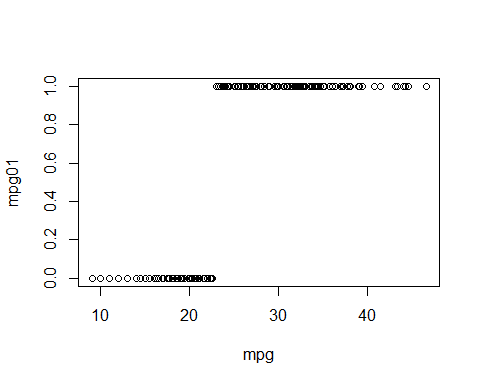
d)برای داده های با تعداد کم می تواند درست باشد اما بتدریج با افزایش تعداد نمونه ها این امکان برای QDA وجود نخواهد داشت

**Chapter4 Question 11:**

rm(list=ls())  
library(ISLR)

## Warning: package 'ISLR' was built under R version 3.1.3

attach(Auto)  
#a   
mpg01=ifelse(mpg<median(mpg),0,1)  
plot(mpg,mpg01)



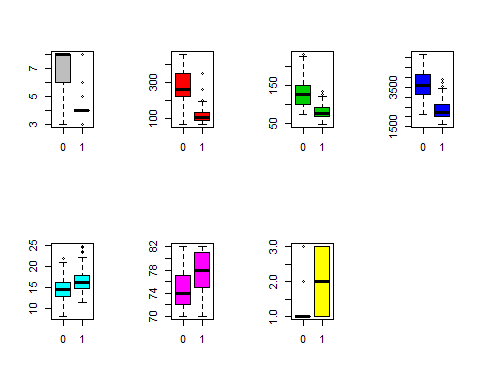
در اینجا می توان دید که آن برش مورد نظر نسبت به میانه زده شده است.

nAuto=data.frame(mpg01,Auto[-1])

حال دیتای جدیدی می سازیم که بجای mpg متغیر mpg01 گذاشته شده است.

#b  
par(mfrow=c(2,4))  
boxplot(cylinders~mpg01, col=8)  
boxplot(displacement~mpg01, col=2)  
boxplot(horsepower~mpg01, col=3)  
boxplot(weight~mpg01, col=4)  
boxplot(acceleration~mpg01, col=5)  
boxplot(year~mpg01, col=6)  
boxplot(origin~mpg01, col=7)

پس از رسم plotbox ها می توان دید که mpg01 بیشتر از بقیه به cylinders *و* displacement *و* horsepower *و* weight *و* year *وابسته می باشد.*



حال داده ها را به دو دسته train و تست تقسیم می کنیم که در این مثال داده های آموزشی را ماشین های تولید قبل سال 80 در نظر گرفته ایم.

#c  
ind=(year<80);  
train=nAuto[ind,]  
test=nAuto[!ind,]  
################  
#d  
library(MASS)  
mod=lda(mpg01~cylinders+displacement+horsepower+weight+year,data = train)  
summary(mod)

## Length Class Mode   
## prior 2 -none- numeric   
## counts 2 -none- numeric   
## means 10 -none- numeric   
## scaling 5 -none- numeric   
## lev 2 -none- character  
## svd 1 -none- numeric   
## N 1 -none- numeric   
## call 3 -none- call   
## terms 3 terms call   
## xlevels 0 -none- list

ابتدا داده های اموزشی را ترین کردیم و می خواهیم بر روی داده های تست حاصل کار خود را ببینیم

pred=predict(mod,test,type = "response")  
head(pred$class)

## [1] 1 1 1 1 1 1  
## Levels: 0 1

حال داده ها را با تقریب به 0 و 1 که معرف کلاس ها هست تقسیم می کنیم؛ پس از آن جدولی تشکیل می دهیم تا ببینیم چقدر به اشتباه 0 ها را 1 و چقدر از 1 ها را 0تشخیص داده است.

class\_pred=pred$class  
table(class\_pred,mpg01[!ind])

##   
## class\_pred 0 1  
## 0 4 4  
## 1 1 76

mean(class\_pred!=mpg01[!ind])

مشاهده می شود که 6 درصد خطا خواهیم داشت.

## [1] 0.05882353

################  
#e

حال به روش QDA می خواهیم عمل کنیم وببینیم چقدر تفاوت ایجاد خواهد شد و آیا از LDA بهتر خواهد بود.

mod=qda(mpg01~cylinders+displacement+horsepower+weight+year,data = train)  
summary(mod)

## Length Class Mode   
## prior 2 -none- numeric   
## counts 2 -none- numeric   
## means 10 -none- numeric   
## scaling 50 -none- numeric   
## ldet 2 -none- numeric   
## lev 2 -none- character  
## N 1 -none- numeric   
## call 3 -none- call   
## terms 3 terms call   
## xlevels 0 -none- list

pred=predict(mod,test,type = "response")  
head(pred$class)

## [1] 1 1 1 1 1 1  
## Levels: 0 1

class\_pred=pred$class  
table(class\_pred,mpg01[!ind])

##   
## class\_pred 0 1  
## 0 5 11  
## 1 0 69

mean(class\_pred!=mpg01[!ind])

## [1] 0.1294118

مشاهده می شود که 13 درصد خطا خواهیم داشت. که یکی از دلایل آن ارتباط خطی موجود بین برخی پیش بینی کننده و mpg01 است که در حالت QDA بخاطر انعطاف بیشتر منحنی را تا حدی بر دیتا fit کرده ایم.

################  
#f

حال تین بار می خواهیم به روش logistic regression عمل کنیم و ببینیم چقدر تفاوت ایجاد خواهد شد و آیا از روش های قبلی بهتر خواهد بود.

mod=glm(mpg01~cylinders+displacement+horsepower+weight+year,data = train)  
summary(mod)

##   
## Call:  
## glm(formula = mpg01 ~ cylinders + displacement + horsepower +   
## weight + year, data = train)  
##   
## Deviance Residuals:   
## Min 1Q Median 3Q Max   
## -0.8963 -0.2213 0.1045 0.2156 0.9935   
##   
## Coefficients:  
## Estimate Std. Error t value Pr(>|t|)   
## (Intercept) 6.024e-01 5.177e-01 1.164 0.24546   
## cylinders -9.744e-02 3.405e-02 -2.862 0.00451 \*\*   
## displacement -6.038e-05 7.310e-04 -0.083 0.93423   
## horsepower 2.155e-03 1.065e-03 2.024 0.04381 \*   
## weight -3.082e-04 5.858e-05 -5.262 2.71e-07 \*\*\*  
## year 1.440e-02 6.841e-03 2.105 0.03614 \*   
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## (Dispersion parameter for gaussian family taken to be 0.09939563)  
##   
## Null deviance: 72.169 on 306 degrees of freedom  
## Residual deviance: 29.918 on 301 degrees of freedom  
## AIC: 170.41  
##   
## Number of Fisher Scoring iterations: 2

هر چند این تخمین نشان می­دهد برخی پارامترهای انتخابی ما خیلی هم مناسب نبوده اما با همین پارامترها ادامه خواهیم داد تا بتوان مقایسه ای بین روش های مختلف پیش بینی داشته باشیم.

pred=predict(mod,test,type = "response")  
pred[pred<=0.5]=0  
pred[pred>0.5]=1

با توجه به اینکه عملگر $class بر روی پیش بینی به روش glm تعریف نشده به روش بالا خودمان تقریب زده ایم.

table(pred,mpg01[!ind])

##   
## pred 0 1  
## 0 4 5  
## 1 1 75

mean(pred!=mpg01[!ind])

## [1] 0.07058824

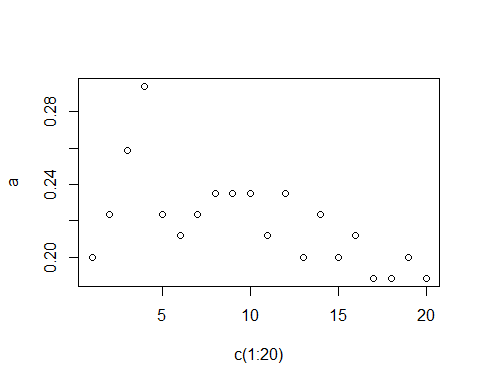
مشاهده می شود که 7 درصد خطا داریم که بیان کننده این است که هرچند از روش QDA بهتر است اما از روش LDA بهتر نمی باشد.

################  
#g

حال نوبت به روش KNN رسیده و می خواهیم ببینیم این روش نسبت به کدام روش ها بر روی این داده ها پاسخ بهتری می دهد.

library(class)  
train.Y=mpg01[ind]  
test.Y=mpg01[!ind]  
set.seed(10)  
a={};K=1;  
pred=knn(train[,-9],test[,-9],mpg01[ind],k=1)  
a[1]=mean(test.Y!=pred)  
for (i in 2:20)  
{pred=knn(train[,-9],test[,-9],mpg01[ind],k=i)  
a[i]=mean(test.Y!=pred)  
K=ifelse(a[i]<min(a[-i]),i,K)  
}  
par(mfrow=c(1,1))

plot(c(1:20),a)



K

## [1] 17

پس از محاسبه بهترین K که در اینجا 17 بدست آمد خطای تخمین را به ازای K=17 محاسبه می کنیم.

pred=knn(train[,-9],test[,-9],mpg01[ind],k=K)

knn\_er=mean(test.Y!=pred)

knn\_er

## [1] 0.1882353

مشاهده می کنیم که 19 درصد خطا داریم که یکی از دلایل اصلی آن زیاد بودن تعداد پیش بینی کننده ها و تاثیر منفی زیاد پارامترهایی است که ارتباط کمتری با mpg01 دارد.

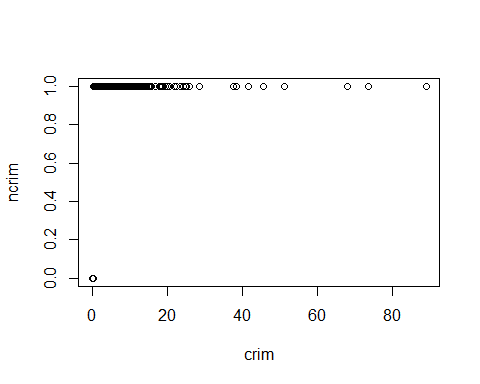
**Chapter4 Question 13:**

دقیقا همان کارهایی که در سوال قبل انجام دادیم را روی این دیتا انجام خواهیم داد. هرچند QDA را نخواسته است اما آن را نیز مقایسه خواهیم کرد.

rm(list=ls())  
library(MASS)  
attach(Boston)  
names(Boston)

## [1] "crim" "zn" "indus" "chas" "nox" "rm" "age"   
## [8] "dis" "rad" "tax" "ptratio" "black" "lstat" "medv"

ncrim=ifelse(crim<median(crim),0,1)  
plot(crim,ncrim)

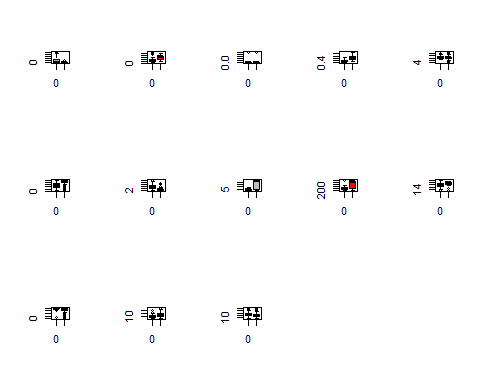


شکل بالا نشان می دهد مقادیر بزرگتر از میانه که عددی کوچک است پراکندگی بالایی دارند بطوری که تا 100 برابر بزرگتر از مقدار میانه نیز نمونه داریم.

nBoston=data.frame(ncrim,Boston[-1])  
par(mfrow=c(3,5))  
cor(ncrim,medv)

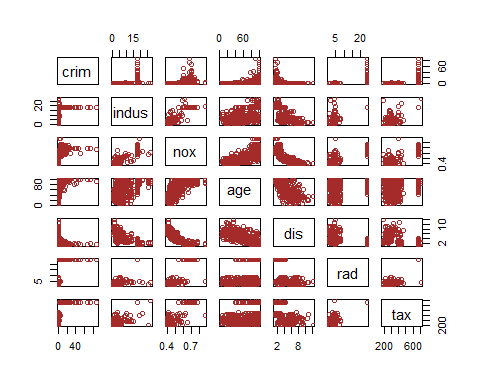
## [1] -0.2630167

boxplot(zn~ncrim, col=8)  
boxplot(indus~ncrim, col=2)  
boxplot(chas~ncrim, col=3)  
boxplot(nox~ncrim, col=4)  
boxplot(rm~ncrim, col=5)  
boxplot(age~ncrim, col=6)  
boxplot(dis~ncrim, col=7)  
boxplot(rad~ncrim, col=8)  
boxplot(tax~ncrim, col=2)  
boxplot(ptratio~ncrim, col=3)  
boxplot(black~ncrim, col=4)  
boxplot(lstat~ncrim, col=5)  
boxplot(medv~ncrim, col=6)  
#indus-nox-age-dis-rad-tax  
pairs(crim~indus-nox-age-dis-rad-tax,col="brown")



با توجه به اینکه از plotbox های بالا چیزی معلوم نیست و امکان زوم در این فایل وجود ندارد؛ فقط نتایج آن را شرح می دهم.

از شکل های بدست آمده می توان فهمید که Ncrim به پارامترهای indus و nox و age و dis و rad و tax مرتبط می باشد.



داده های train و test را بر اساس تعداد اتاق هر واحد جداکرده ام به طوری که با توجه به واحدهای با کمتر از 7 اتاق عدد جرایم برای واحد های با تعداد اتاق بیش از 7 را پیش بینی می کنیم

ind=(rm<7);  
train=nBoston[ind,]  
test=nBoston[!ind,]  
##################  
#predicting  
mod=lda(ncrim~indus+nox+age+dis+rad+tax,data = train)  
summary(mod)

## Length Class Mode   
## prior 2 -none- numeric   
## counts 2 -none- numeric   
## means 12 -none- numeric   
## scaling 6 -none- numeric   
## lev 2 -none- character  
## svd 1 -none- numeric   
## N 1 -none- numeric   
## call 3 -none- call   
## terms 3 terms call   
## xlevels 0 -none- list

pred=predict(mod,test,type = "response")  
head(pred$class)

## [1] 0 0 0 0 0 0  
## Levels: 0 1

class\_pred=pred$class  
table(class\_pred,ncrim[!ind])

##   
## class\_pred 0 1  
## 0 35 12  
## 1 0 17

mean(class\_pred!=ncrim[!ind])

## [1] 0.1875

خطای LDA برابر 18.75 % می باشد.

################  
mod=qda(ncrim~indus+nox+age+dis+rad+tax,data = train)  
summary(mod)

## Length Class Mode   
## prior 2 -none- numeric   
## counts 2 -none- numeric   
## means 12 -none- numeric   
## scaling 72 -none- numeric   
## ldet 2 -none- numeric   
## lev 2 -none- character  
## N 1 -none- numeric   
## call 3 -none- call   
## terms 3 terms call   
## xlevels 0 -none- list

pred=predict(mod,test,type = "response")  
head(pred$class)

## [1] 0 0 0 0 0 0  
## Levels: 0 1

class\_pred=pred$class  
table(class\_pred,ncrim[!ind])

##   
## class\_pred 0 1  
## 0 35 6  
## 1 0 23

mean(class\_pred!=ncrim[!ind])

## [1] 0.09375

خطای QDA برابر 9.3 % می باشد.

################  
mod=glm(ncrim~indus+nox+age+dis+rad+tax,data = train)  
summary(mod)

##   
## Call:  
## glm(formula = ncrim ~ indus + nox + age + dis + rad + tax, data = train)  
##   
## Deviance Residuals:   
## Min 1Q Median 3Q Max   
## -0.6155 -0.1806 -0.0509 0.1027 0.8959   
##   
## Coefficients:  
## Estimate Std. Error t value Pr(>|t|)   
## (Intercept) -0.7068608 0.1585024 -4.460 1.05e-05 \*\*\*  
## indus 0.0051373 0.0043130 1.191 0.23426   
## nox 1.6035450 0.2486188 6.450 2.99e-10 \*\*\*  
## age 0.0028177 0.0008612 3.272 0.00115 \*\*   
## dis -0.0063852 0.0130372 -0.490 0.62455   
## rad 0.0215304 0.0042866 5.023 7.45e-07 \*\*\*  
## tax -0.0003269 0.0002565 -1.275 0.20308   
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## (Dispersion parameter for gaussian family taken to be 0.1014715)  
##   
## Null deviance: 110.48 on 441 degrees of freedom  
## Residual deviance: 44.14 on 435 degrees of freedom  
## AIC: 252  
##   
## Number of Fisher Scoring iterations: 2

pred=predict(mod,test,type = "response")  
pred[pred<=0.5]=0  
pred[pred>0.5]=1  
table(class\_pred,ncrim[!ind])

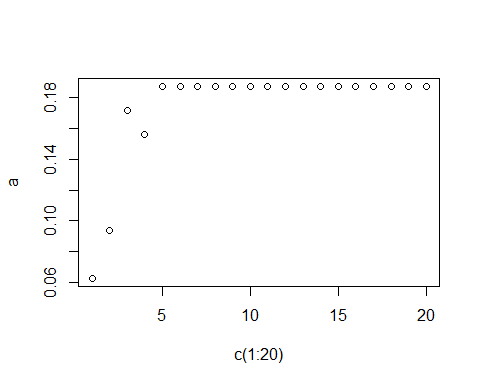
##   
## class\_pred 0 1  
## 0 35 6  
## 1 0 23

mean(class\_pred!=ncrim[!ind])

## [1] 0.09375

خطای logistic regression برابر 9.3 % می باشد.

################  
library(class)  
train.Y=ncrim[ind]  
test.Y=ncrim[!ind]  
set.seed(10)  
a={};K=1;  
pred=knn(train[,c(3,5,7,9,10)],test[,c(3,5,7,9,10)],ncrim[ind],k=1)  
a[1]=mean(test.Y!=pred)  
for (i in 2:20)  
{pred=knn(train[,c(3,5,7,9,10)],test[,c(3,5,7,9,10)],ncrim[ind],k=i)  
a[i]=mean(test.Y!=pred)  
K=ifelse(a[i]<min(a[-i]),i,K)  
}  
par(mfrow=c(1,1))  
plot(c(1:20),a)



K

## [1] 1

pred=knn(train[,c(3,5,7,9,10)],test[,c(3,5,7,9,10)],ncrim[ind],k=i)

knn\_er=mean(test.Y!=pred)

knn\_er

##[1] 0.0625

خطای KNN در بهترین شرایط یعنی برای K=1 برابر 6 % می باشد.

از مشاهدات انجام شده می توان فهمید KNN بهترین روش برای تخمین عدد جرم می باشد که می توان گفت یکی از مهمترین دلایل این برتری بدلیل اهمیت بالای پارامترهای انتخابی بوده است.

**Chapter5 Question5:**

rm(list=ls())  
library(ISLR)

## Warning: package 'ISLR' was built under R version 3.1.3

attach(Default)  
#a  
set.seed(1)  
ind=sample(1:10000,9000)  
train=Default[ind,]  
test=Default[-ind,]  
mod=glm(default~income+balance,data = Default,family=binomial)  
summary(mod)

##   
## Call:  
## glm(formula = default ~ income + balance, family = binomial,   
## data = Default)  
##   
## Deviance Residuals:   
## Min 1Q Median 3Q Max   
## -2.4725 -0.1444 -0.0574 -0.0211 3.7245   
##   
## Coefficients:  
## Estimate Std. Error z value Pr(>|z|)   
## (Intercept) -1.154e+01 4.348e-01 -26.545 < 2e-16 \*\*\*  
## income 2.081e-05 4.985e-06 4.174 2.99e-05 \*\*\*  
## balance 5.647e-03 2.274e-04 24.836 < 2e-16 \*\*\*  
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## (Dispersion parameter for binomial family taken to be 1)  
##   
## Null deviance: 2920.6 on 9999 degrees of freedom  
## Residual deviance: 1579.0 on 9997 degrees of freedom  
## AIC: 1585  
## Number of Fisher Scoring iterations: 8

pred=predict(mod,test,type = "response")  
pred[pred<=0.5]="YES"  
pred[pred>0.5]="No"  
table(pred,default[-ind])

## pred No Yes  
## No 971 29

mean(pred!=default[-ind])

## [1] 0.029

تا اینجا یک تخمین logistic regression ساده روی تمام 10000 داده موجود در دیتاست Default زده ایم و روی 1000 داده تست کرده ایم که میزان خطا 2.9 % بدست آمده است.

#b  
mod=glm(default~income+balance,data = train,family=binomial)  
summary(mod)

##   
## Call:  
## glm(formula = default ~ income + balance, family = binomial,   
## data = train)  
##   
## Deviance Residuals:   
## Min 1Q Median 3Q Max   
## -2.4874 -0.1449 -0.0573 -0.0209 3.7210   
##   
## Coefficients:  
## Estimate Std. Error z value Pr(>|z|)   
## (Intercept) -1.153e+01 4.559e-01 -25.281 < 2e-16 \*\*\*  
## income 2.028e-05 5.250e-06 3.862 0.000112 \*\*\*  
## balance 5.668e-03 2.395e-04 23.663 < 2e-16 \*\*\*  
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## (Dispersion parameter for binomial family taken to be 1)  
##   
## Null deviance: 2657.5 on 8999 degrees of freedom  
## Residual deviance: 1433.7 on 8997 degrees of freedom  
## AIC: 1439.7  
##   
## Number of Fisher Scoring iterations: 8

pred=predict(mod,test,type = "response")  
pred[pred<=0.5]="YES"  
pred[pred>0.5]="No"  
table(pred,default[-ind])

##   
## pred No Yes  
## No 971 29

mean(pred!=default[-ind])

## [1] 0.029

در این قسمت دقیقا همان کار را برروی داده ها انجام دادیم با این تفاوت که داده های آموزشی را از کل داده ها به 9000 داده کاهش داده ایم. اما چون تعداد داده های آموزشی همچنان زیاد است تفاوتی ایجاد نشده است. که در ادامه با تعداد داده آموزشی کمتر تفاوت آنها را خواهیم دید.

#c  
a={}  
for (i in 1:3)  
{ind=sample(1:10000,3000\*i)  
train=Default[ind,]  
test=Default[-ind,]  
mod=glm(default~income+balance,data = train,family=binomial)  
summary(mod)  
pred=predict(mod,test,type = "response")  
pred[pred<=0.5]="YES"  
pred[pred>0.5]="No"  
table(pred,default[-ind])  
a[i]=mean(pred!=default[-ind])  
}  
a

## [1] 0.03171429 0.03250000 0.04200000

در این قسمت آزمایش را به ترتیب با 9000 ، 6000 ، 3000 داده آموزشی و بقیه داده تست انجام داده ایم. مشاهده می کنیم که هرچه تعداد داده های آموزشی بیشتر بوده خطای ما نیز کمتر شده است.

#d  
set.seed(1)  
ind=sample(1:10000,9000)  
train=Default[ind,]  
test=Default[-ind,]  
mod=glm(default~income+balance+student,data = Default,family=binomial)  
summary(mod)

##   
## Call:  
## glm(formula = default ~ income + balance + student, family = binomial,   
## data = Default)  
##   
## Deviance Residuals:   
## Min 1Q Median 3Q Max   
## -2.4691 -0.1418 -0.0557 -0.0203 3.7383   
##   
## Coefficients:  
## Estimate Std. Error z value Pr(>|z|)   
## (Intercept) -1.087e+01 4.923e-01 -22.080 < 2e-16 \*\*\*  
## income 3.033e-06 8.203e-06 0.370 0.71152   
## balance 5.737e-03 2.319e-04 24.738 < 2e-16 \*\*\*  
## studentYes -6.468e-01 2.363e-01 -2.738 0.00619 \*\*   
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## (Dispersion parameter for binomial family taken to be 1)  
##   
## Null deviance: 2920.6 on 9999 degrees of freedom  
## Residual deviance: 1571.5 on 9996 degrees of freedom  
## AIC: 1579.5  
##   
## Number of Fisher Scoring iterations: 8

pred=predict(mod,test,type = "response")  
pred[pred<=0.5]="YES"  
pred[pred>0.5]="No"  
table(pred,default[-ind])

##   
## pred No Yes  
## No 971 29

mean(pred!=default[-ind])

## [1] 0.029

در این قسمت یک پارامتر دیگر با وابستگی کمتر که نشان دهنده دانش آموز بودن یا نبودن را اضافه کردیم.

همانطور که مشاهده می شود برای داده های آموزشی بالا اثر منفی این عامل خنثی می شود ولی در حالت با داده آموزشی کم تاثیر منفی می گذارد.

**Chapter5 Question9:**

rm(list=ls())  
st\_er=function(data,index)  
{  
 len=length(index)  
 miu=mean(data[index])  
 a=mean((data[index]-miu)^2)  
 se=(a/len)^0.5  
}  
library(MASS)  
library(boot)  
attach(Boston)  
#a  
miu=mean(medv)  
miu

## [1] 22.53281

همانطور که خواسته شده بود میانگین مقدار پول با واحد 1000 دلار که مالک دریافت می کند را بدست آوردیم.

#b  
se=st\_er(medv,1:length(medv))  
se

## [1] 0.4084569

خطای استاندارد به ازای تخمین میانگین medv به جای مقادیر آن برابر 40 درصد می شود.

Rss=sum((medv-miu)^2)  
#c

حال می خواهیم خطای استاندارد را و تخمین بازای میانگین را در حالتی انجام دهیم که میانگین به روش bootstrap بدست آورده باشیم.

alpha.fn=function(data,index){  
 X=data$medv[index]  
 }  
alpha.fn(Boston,length(medv))  
set.seed(1)  
alpha.fn(Boston,sample(length(medv),length(medv),replace=T))  
boot(medv,st\_er,R=1000)

##   
## ORDINARY NONPARAMETRIC BOOTSTRAP  
##   
##   
## Call:  
## boot(data = medv, statistic = st\_er, R = 1000)  
##   
##   
## Bootstrap Statistics :  
## original bias std. error  
## t1\* 0.4084569 -6.27049e-05 0.01654656

خطای استاندارد به ازای تخمین میانگین medv به روش bootstrap برابر 1.6 درصد می شود.

حال می توان اختلاف زیاد این روش با روش عادی را مشاهده کرد.

#d  
se\_medv=st\_er(medv,1:length(medv))  
miu=mean(medv)  
con\_int\_medv=c(miu-2\*se\_medv,miu+2\*se\_medv)  
s1=(min(con\_int\_medv)<medv)&(max(con\_int\_medv)>medv)  
nData=Boston[s1,]

در این قسمت فقط داده هایی را جدا کردیم که در بازه 95 % فاصله اطمینان را شامل باشد و نام آن را nData گذاشتیم.

#e  
mu=median(medv)

حال همان روند کاری قبلی را نسبت به میانه انجام می دهیم.

#f  
len=length(medv)  
aprim={}  
set.seed(1)  
for (i in 1:100)  
{  
ind=sample(len,len,replace=T)  
train=Boston[ind,]  
nmedv=train$medv  
miu=median(nmedv)  
a=mean((nmedv-miu)^2)  
se=(a/len)^0.5  
aprim[i]=se  
}  
ahat=mean(se)  
set=sqrt(sum((aprim-ahat)^2)/(length(aprim)-1))

set

## [1] 0.01755158

می بینیم که خطای استاندارد بدست آمده برابر 1.7 % یعنی اندکی بیشتر از حالت قبل خواهد بود.

در این سری از سوالات سعی شد تنوعی از مدل های برنامه نویسی برای رسیدن به هدفمان در برنامه R ارائه شود؛ نظیر تابع نویسی یا بدون تابع که نظیر آن در سوال آخر دیده می شود که هم تابع bootstrap بوسیله کتابخانه ها و هم بصورت پایه ای نوشته شده است. تا هم بر ریاضیات کار مسلط شوم و هم با نحوه به کارگیری این زبان برای کاربرد مورد نظر.

**ممنون از توجهتان**