پروژه ی تحلیل رگرسیون

امیرمحمد محمدقلیها

Classification

توضیحات مختصر مربوط به هر ستون داده به صورت زیر است:

GRE Scores (out of 340)

نمره ی آزمون GRE

TOEFL Scores (out of 120)

نمره ی آزمون Tofel

University Rating (out of 5)

رتبه یا rate دانشگاه

Statement of Purpose Strength (SOR) (out of 5)

قوی بودن انگیزه ی فرد

Letter of Recommendation Strength (LOR) (out of 5)

قوی بودن recommendation های فرد

Undergraduate GPA (CGPA) (out of 10)

معدل کارشناسی

Research Experience (either 0 or 1)

تجربه ی تحقیق و research

Chance of Admit (ranging from 0 to 1)

شانس پذیرفته شدن

Admit (either 0 or 1)

پذیرفته شدن یا نشدن (که جلوتر آن را میسازیم).

حدود داده ها به صورت زیر است:

```
summary(admission)
```

```
##  Serial No.      GRE Score      TOEFL Score    University Rating ##
##  Min.   :  1.0   Min.   :290.0   Min.   : 92.0   Min.   :1.000 ##
```

```
      1st Qu.:100.8   1st Qu.:308.0   1st Qu.:103.0   1st Qu.:2.000 ##
                Median :200.5   Median :317.0   Median :107.0   Median :3.000 ##
                  Mean   :200.5   Mean   :316.8   Mean   :107.4   Mean   :3.087 ##
      3rd Qu.:300.2   3rd Qu.:325.0   3rd Qu.:112.0   3rd Qu.:4.000 ##
                  Max.   :400.0   Max.   :340.0   Max.   :120.0   Max.   :5.000 ##
                        SOP             LOR             CGPA           Research      ##
                     Min.   :1.0   Min.   :1.000   Min.   :6.800   Min.   :0.0000 ##
       1st Qu.:2.5   1st Qu.:3.000   1st Qu.:8.170   1st Qu.:0.0000 ##
                Median :3.5   Median :3.500   Median :8.610   Median :1.0000 ##
                  Mean   :3.4   Mean   :3.453   Mean   :8.599   Mean   :0.5475 ##
       3rd Qu.:4.0   3rd Qu.:4.000   3rd Qu.:9.062   3rd Qu.:1.0000 ##
                  Max.   :5.0   Max.   :5.000   Max.   :9.920   Max.   :1.0000 ##
                                                                Chance of Admit ##
                                                                     Min.   :0.3400 ##
                                                                1st Qu.:0.6400 ##
                                                                Median :0.7300 ##
                                                                Mean   :0.7244 ##
                                                                3rd Qu.:0.8300 ##
                                                                Max.   :0.9700 ##
```

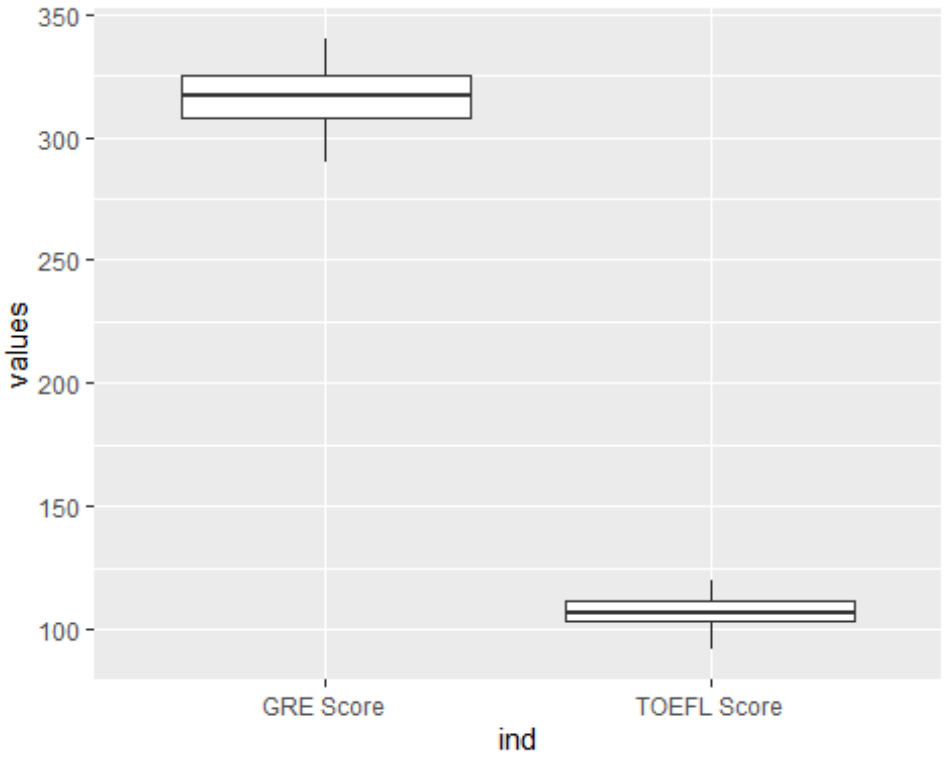
Task

در اینجا قصد داریم پذیرفته شدن یا نشدن فرد را با استفاده از اطلاعات تحصیلی او پیش بینی کنیم. این کار به برنامه ریزی فرد بسیار میتواند کمک کننده باشد تا بداند در چه بخش های سرمایه گذاری بیشتری انجام دهد تا در آزمون پذیرفته شود.

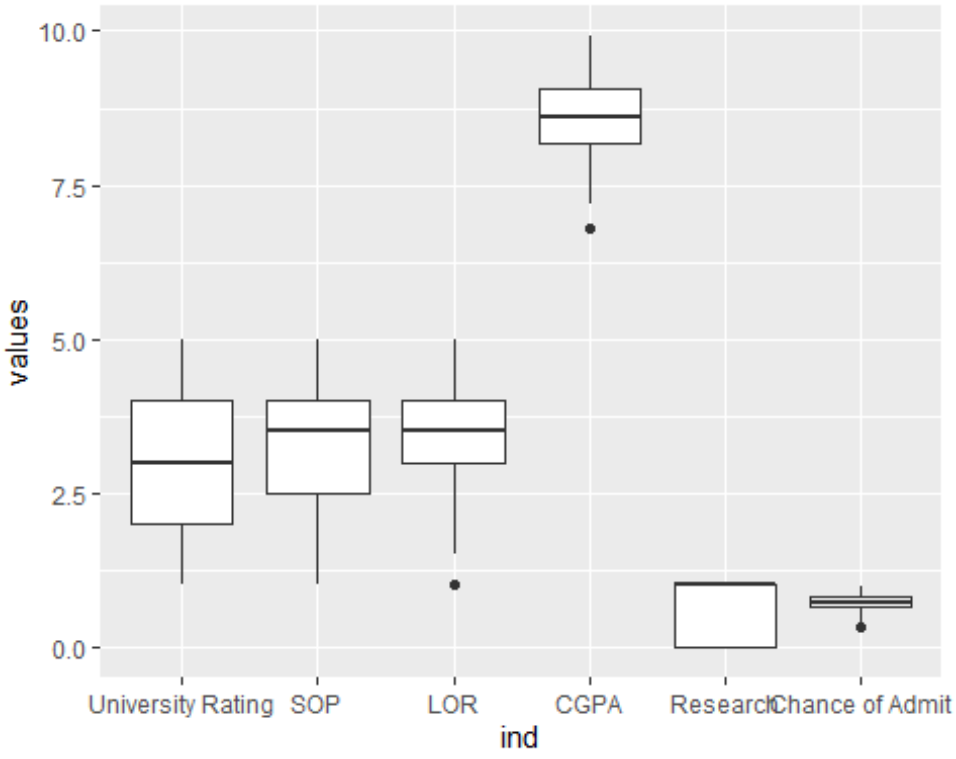
متغیر هدف: Admit

ابتدا نمودار جعبه ای داده ها رسم میکنیم تا پراکندگی داده ها را بررسی کنیم.

```
# before dropping the outliers
ggplot(stack(admission[,2:3]),aes(x=ind,y=values))+geom_boxplot()
```




```
ggplot(stack(admission[,4:9]),aes(x=ind,y=values))+geom_boxplot()
```

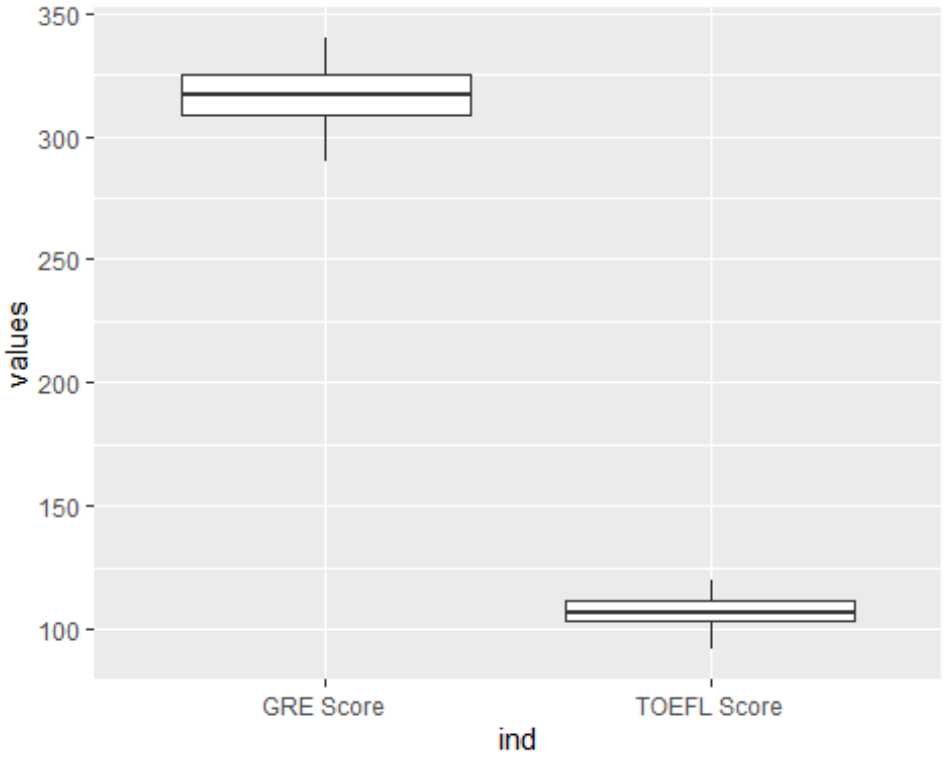


در ابتدا ستون اول که مربوط به شماره ی سریال است را حذف میکنیم زیرا اطلاعاتی از آن نمیتوان بدست آورد. سپس با استفاده از z-score داده های پرت را تبدیل به NA کرده تا هنگام حذف NA های داده، تمام داده های پرت حذف گردند.

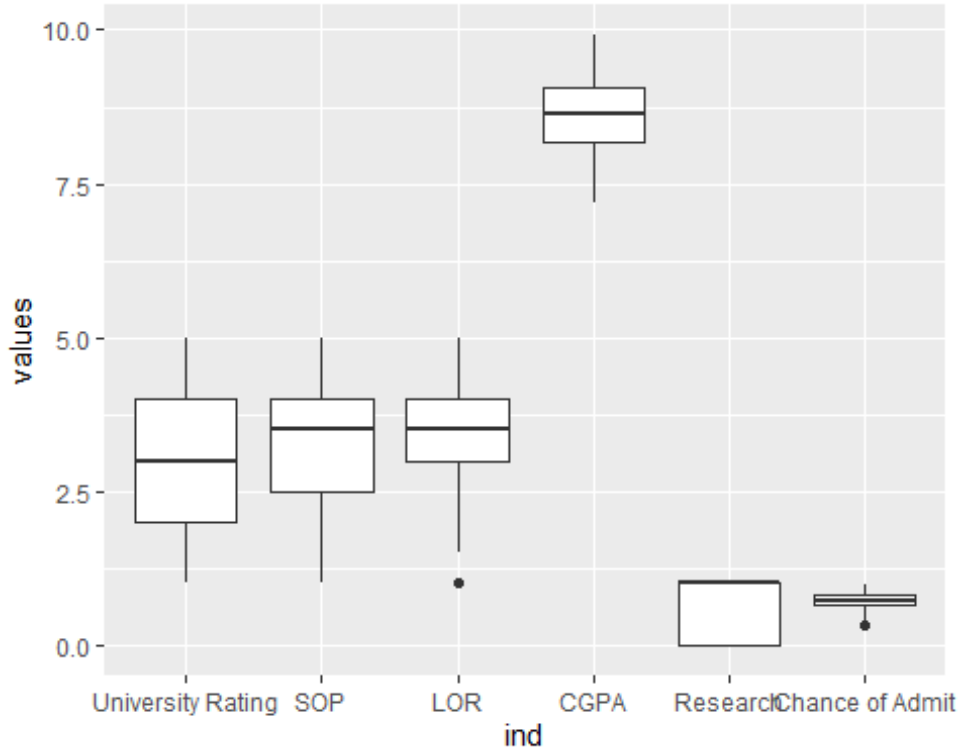
```
# omit the first column (Serial No.) which has no information
admission=admission[-which(colnames(admission)=="Serial No.") ]

# remove outliers using z-score
scaled=scale(admission, center = TRUE, scale = TRUE)
scaled=as.data.frame(scaled)
admission[abs(scaled)>3]=NA

# remove NA from dataset
admission=na.omit(admission)
```

اکنون باری دیگر به نمودار جعبه ای داده ها پس از حذف داده های پرت نگاهی میندازیم.

```
# after dropping the outliers
ggplot(stack(admission[,1:2]),aes(x=ind,y=values))+geom_boxplot()
```




```
ggplot(stack(admission[,3:8]),aes(x=ind,y=values))+geom_boxplot()
```



اکنون برای آنکه متغیر هدف یک کتغیر کیفی باشد، با در نظر گرفتن treshold برابر با 0.75 برای متغیر Chance of Admit یک متغیر dummy به نام Admit تعریف کرده ایم تا احتمال های بیشتر از حد تعیین شده را 1 و احتمال های کمتر از حد تعیین شده را 0 در نظر بگیریم.

```
admission$Research=as.factor(admission$Research)

admission[admission$`Chance of Admit` >=0.75,"Admit"]=1
admission[admission$`Chance of Admit` <0.75,"Admit"]=0
admission$Admit=as.factor(admission$Admit)

#drop the (Chance of Admit) probability column
admission=admission[-which(colnames(admission)=="Chance of Admit") ]

summary(admission)
```

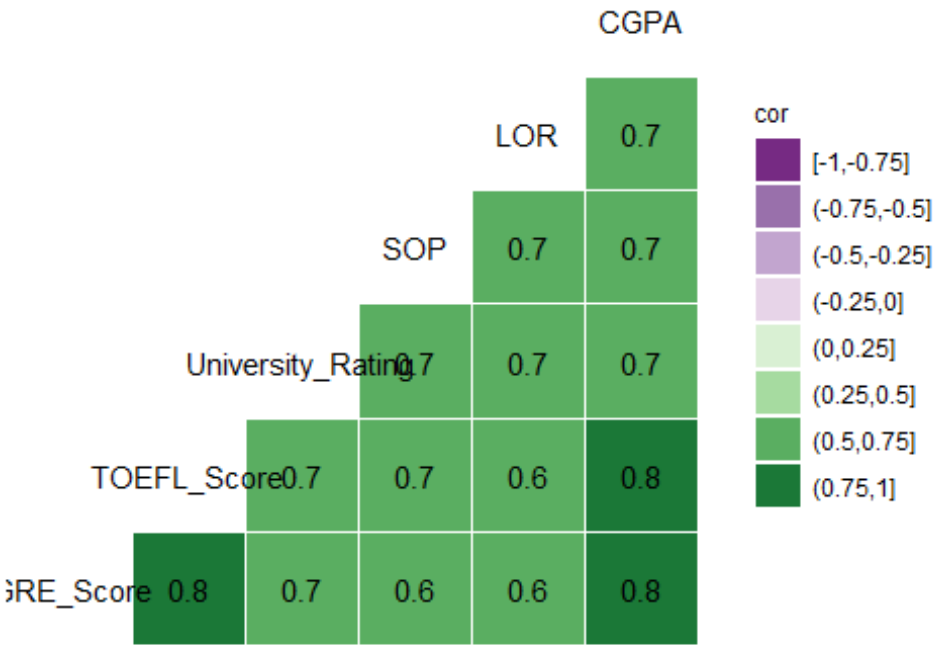
```
##  GRE Score      TOEFL Score     University Rating      SOP        ##
##  Min.   :290.0   Min.   : 92.0   Min.   :1.000   Min.   :1.000   ##
##  1st Qu.:308.5   1st Qu.:103.0   1st Qu.:2.000   1st Qu.:2.500   ##
##  Median :317.0   Median :107.0   Median :3.000   Median :3.500   ##
##  Mean   :316.8   Mean   :107.4   Mean   :3.093   Mean   :3.401   ##
##  3rd Qu.:325.0   3rd Qu.:112.0   3rd Qu.:4.000   3rd Qu.:4.000   ##
##  Max.   :340.0   Max.   :120.0   Max.   :5.000   Max.   :5.000   ##
##       LOR             CGPA           Research   Admit      ##
##  Min.   :1.000   Min.   :7.200   0:181    0:219   ##
##  1st Qu.:3.000   1st Qu.:8.175   1:218    1:180   ##
##  Median :3.500   Median :8.620                    ##
```

```
                                                Mean   :3.456   Mean   :8.603 ##
                                              3rd Qu.:4.000     3rd Qu.:9.065 ##
                                                Max.   :5.000   Max.   :9.920 ##
```

اکنون مشتاق هستیم تا میزان همبستگی بین متغیر ها را با بهره گیری از نمودار و شکل بررسی کنیم.

```
ggcorr(admission, palette = "PRGn", name="cor",label=TRUE,nbreaks = 8)
```

```
: ,Warning in ggcorr(admission, palette = "PRGn", name = "cor", label = TRUE ##
data in column(s) 'Research', 'Admit' are not numeric and were ignored ##
```



میبینیم که همبستگی زیادی بین ستون ها هست و احتمالا افرادی که در تحصیلات خود موفق بوده اند، قوی بودنشان در تمام زمینه های تحصیلی به چشم میخورد.

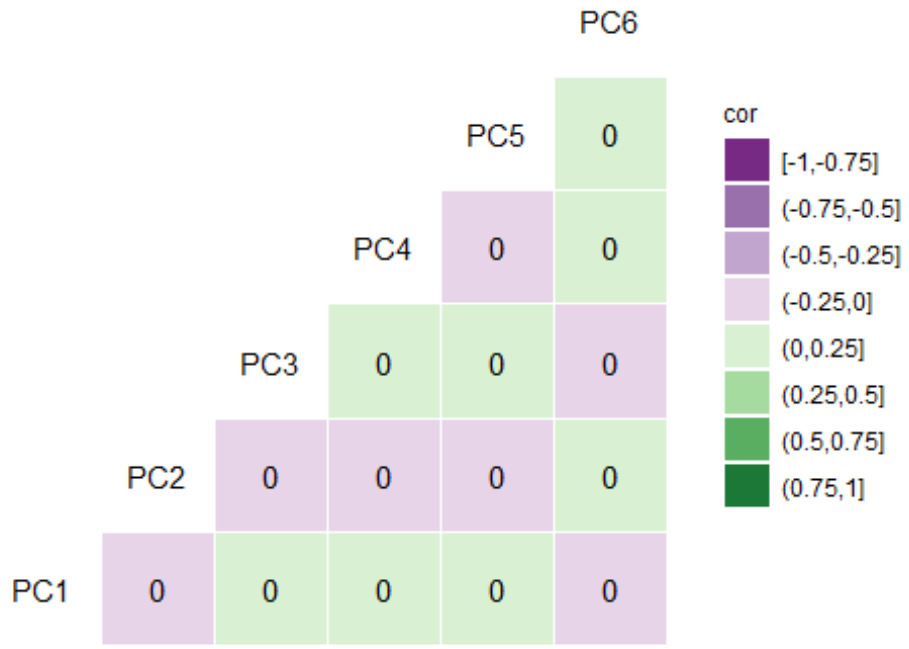
یکی از راه های از بین بردن این همیستگی استفاده از PCA است.

```
pca=prcomp(admission[,1:6],scale = TRUE)
summary(pca)
```

```
:Importance of components ##
                          PC1    PC2     PC3     PC4     PC5     PC6          ##
Standard deviation     2.124 0.7897 0.56007 0.49274 0.40403 0.38063 ##
Proportion of Variance 0.752 0.1040 0.05228 0.04047 0.02721 0.02415 ##
Cumulative Proportion  0.752 0.8559 0.90818 0.94865 0.97585 1.00000 ##
```

میبینیم که واریانس نسبی تجمعی تا متغیر 4 ام حدود 95٪ است پس توقع داریم که بتوانیم مدل ها را با چهار متغیر توضیح دهنده برازش دهیم.

```
loadings=as.data.frame(pca$x)
ggcorr(loadings, palette = "PRGn", name="cor",label=TRUE,nbreaks = 8)
```




```
admission=cbind(loadings,admission[,7:8])
```

حال شش ستون جدید یعنی PC1 تا PC6 و Admit و Research را به یکدیگر میچسبانیم تا dataframe جدید تولید شود و اکنون با این داده ها کار میکنیم.

```
pca=prcomp(admission[,1:6],scale = TRUE)
summary(pca)
```

```
## Importance of components:
##                           PC1    PC2    PC3    PC4    PC5    PC6
## Standard deviation     1.0000 1.0000 1.0000 1.0000 1.0000 1.0000
## Proportion of Variance 0.1667 0.1667 0.1667 0.1667 0.1667 0.1667
## Cumulative Proportion  0.1667 0.3333 0.5000 0.6667 0.8333 1.0000
```

مدل GLM

```
glmModel=glm(Admit~.,data=admission,family = binomial)
#summary(model)
glmProb=predict(glmModel,type = "response")
glmPred=c()
Admit=admission$Admit
glmPred[glmProb>=0.75]=1
glmPred[glmProb<0.75]=0
```

```
tableA=table(glmPred,Admit )
tableA
```

```
##        Admit
## glmPred   0   1
##       0 208  45
##       1  11 135
```

```
accuracy=mean(glmPred==Admit)
accuracy
```

```
## [1] 0.8596491
```

```
truePositive=tableA[2,2]
truePositive
```

```
## [1] 135
```

```
falsePositive=tableA[2,1]
falsePositive
```

```
## [1] 11
```

```
trueNegative=tableA[1,1]
trueNegative
```

```
## [1] 208
```

```
falseNegative=tableA[1,2]
falseNegative
```

```
## [1] 45
```

در اینجا هردو خطا اهمیت دارند ولی خطای falseNegative ممکن است زیان بیشتری برساند یعنی به اشتباه پیش بینی کنیم که فرد در آزمون مردود خواهد شد. در این حالت ممکن است فرد نا امید شود و در پذیرش شرکت نکند. که با این استدلال ما در پی آن خواهیم بود که مقدار recall را افزایش دهیم. زیرا recall طبق فرمول زیر falseNegative را به صورت نسبی میسنجد. گرچه ممکن است برای برخی نیز falsePositive زیان آور تر به نظر آید که در این حالت شاخص precision را باید افزایش دهیم.

$$precision = \frac{truePositive}{truepositive + falsepositive}$$

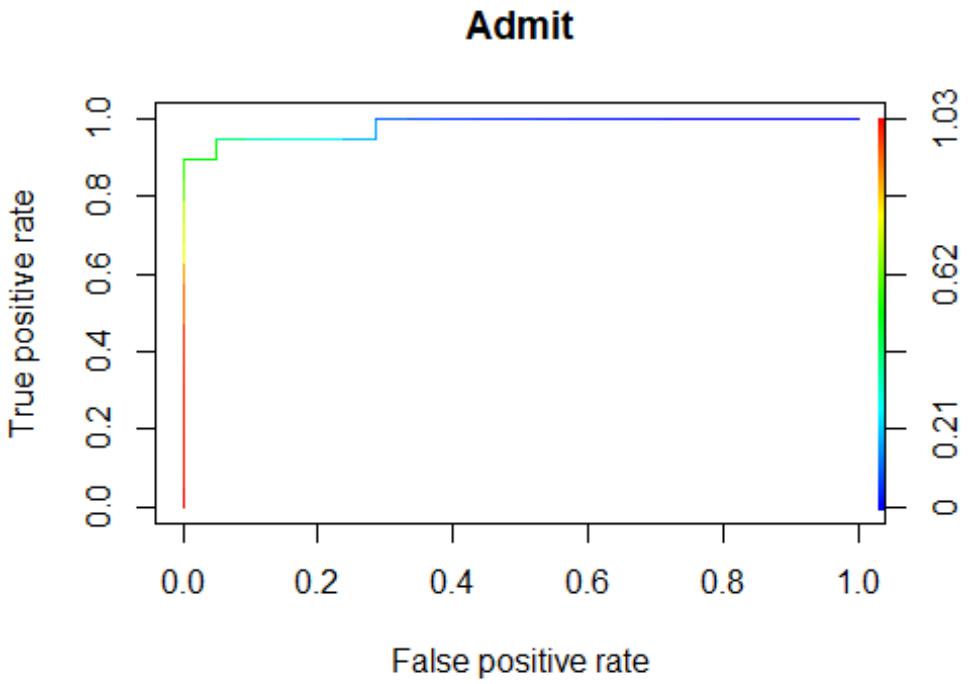
$$recall = \frac{truePositive}{truepositive + falsenegative}$$

```
precision=truePositive/(truePositive+falsePositive)
precision
```

```
## [1] 0.9246575
```

```
recall=truePositive/(truePositive+falseNegative)
recall
```

```
0.75 [1] ##

indexes=sample(nrow(admission) ,replace = FALSE)
foldsIndexes = cut(indexes , breaks=10 , labels=FALSE)
indexesOfTest = which(foldsIndexes==1 , arr.ind=TRUE)
trainData=admission[-indexesOfTest,]
testData=admission[indexesOfTest,]

model=glm(Admit~.,data=trainData,family=binomial)
glmProb=predict(glmModel,testData,type = "response")
pred = prediction(glmProb, testData$Admit)
perf = performance(pred,"tpr","fpr")
plot(perf,colorize=TRUE,main="Admit")
```



مدل KNN

از آنجایی که روش KNN وابسته به فاصله است و تاثیر هر متغیر پیشگو در فاصله وابسته به scale آن متغیر است، روش KNN ممکن است اثر نامطلوبی در این دیتاست داشته باشد. زیرا طبق توضیحات ابتدای این بخش بازه ی GRE تا 340 و بازه ی نمره ی TOFEL تا 120 است و مقیاس بقیه ی متغیر ها نسبت به این دو متغیر کوچکتر است. زمانی که scale یک متغیر بزرگ باشد در واقع انگار اثر بیشتری در مدل داشته است در صورتی که ممکن است تاثیر آن در دنیای واقع به آن اندازه نباشد. همچنین زمانی که scale یک متغیر کوچک باشد در واقع انگار اثر کمتری در مدل داشته است در صورتی که ممکن است تاثیر آن در دنیای واقع به آن اندازه نباشد. بنابراین از آنجا که ما نمیدانیم اثر کدام متغیر های بیشتر و کدام کمتر است scale ها را تغییر نمیدهیم گرچه با آزمون و خطا و کم و زیاد کردن مقیاس متغیر های پیشگو میتوان دقت مدل را بهبود بخشید.

```
set.seed(100)
#we choose 80% of data as train and the rest as test
```

```
nrowTrainData=nrow(admission)*80/100
nrowTestData=nrow(admission)-nrowTrainData

indexOfDataTrain=sample(1:nrow(admission),nrowTrainData)

trainData=admission[indexOfDataTrain,]
testData=admission[-indexOfDataTrain,]

accuracy=c()
for (i in 1:20) {
knnModel=knn(train = trainData,test = testData,trainData$Admit,k=i)
accuracy[i]=(sum(testData$Admit==knnModel)/nrowTestData)
}
#vector of accuracy in term of each k
accuracy

0.9523810 0.9523810 0.9649123 0.9649123 0.9523810 0.9649123 [1] ##
0.9398496
0.9523810 0.9523810 0.9398496 0.9649123 0.9523810 0.9523810 [8] ##
0.9273183
0.9523810 0.9649123 0.9649123 0.9649123 0.9649123 0.9523810 [15] ##

#maximum accuracy
max(accuracy)

0.9649123 [1] ##

#maximum accuracy occures with following k
match(max(accuracy),accuracy)

1 [1] ##
```

همانطور که میبینیم به ازای k=1 مقدار accuracy بیشینه شده است. اما accuracy تنها ملاک نیست و گاهی precision و recall نیز برای بهبود مدل کارآمد تر اند. شاخص accuracy زمانی مناسب است که تقارنی در جدول وجود داشته باشد. یعنی falsePositive و falseNegative تقریبا مقدار مشابهی داشته باشند. ابتدا برای مدل k=1 مقادیر زیر را محاسبه میکنیم:

```
set.seed(100)
bestknn=knn(train = trainData,test = testData,trainData$Admit,k=1)
Admit=testData$Admit
tableA=table(bestknn,Admit)
tableA

Admit     ##
bestknn  0  1 ##
1  48 0    ##
29 2  1    ##

truePositive=sum(testData$Admit==1 & testData$Admit==bestknn)
truePositive
```

```
29 [1] ##
```

```
falsePositive=sum(testData$Admit==0 & testData$Admit!=bestknn)
falsePositive
```

```
2 [1] ##
```

```
trueNegative=sum(testData$Admit==0 & testData$Admit==bestknn)
trueNegative
```

```
48 [1] ##
```

```
falseNegative=sum(testData$Admit==1 & testData$Admit!=bestknn)
falseNegative
```

```
1 [1] ##
```

```
precision=truePositive/(truePositive+falsePositive)
precision
```

```
0.9354839 [1] ##
```

```
recall=truePositive/(truePositive+falseNegative)
recall
```

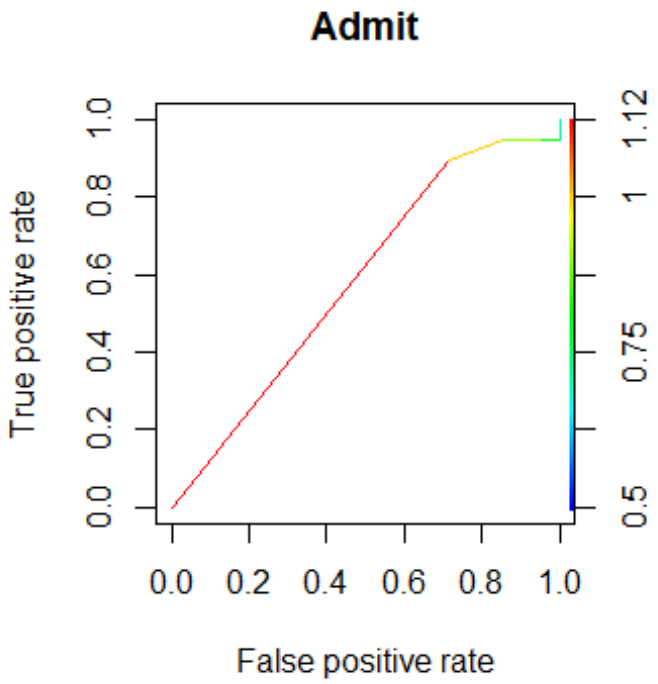
```
0.9666667 [1] ##
```

```
accuracy=sum(testData$Admit==bestknn)/nrowTestData
accuracy
```

```
0.9649123 [1] ##
```

میبینیم که مقدار recall و precision هردو بهبود یافتند. پس مدل KNN مناسب تر از مدل Logistic است.

```
indexes=sample(nrow(admission) ,replace = FALSE)
foldsIndexes = cut(indexes , breaks=10 , labels=FALSE)
indexesOfTest = which(foldsIndexes==1 , arr.ind=TRUE)
trainData=admission[-indexesOfTest,]
testData=admission[indexesOfTest,]

str=paste0("class",i)
trainResponse=admission$Admit[-indexesOfTest]
testResponse=admission$Admit[indexesOfTest]
cl = trainResponse[,drop=TRUE]
model= knn(trainData, testData, cl, k=10, prob=TRUE)
prob = attr(model, "prob")
pred = prediction(prob, testResponse)
perf = performance(pred,"tpr","fpr")
par(pty="s")
plot(perf,colorize=TRUE,main="Admit")
```



مدل LDA

حال یک مدل lda به داده ها برازش میدهیم.

```
ldaModel=lda(Admit~.,data=admission,subset=indexOfDataTrain)
ldaModel

:Call ##
lda(Admit ~ ., data = admission, subset = indexOfDataTrain) ##
##
:Prior probabilities of groups ##
1      0      ##
0.4702194 0.5297806 ##
##
:Group means ##
PC1       PC2       PC3       PC4       PC5       PC6       ##
0.007786459  0.00862079- 0.006063242 0.005238275- 0.11129163- 1.430367  0 ##
0.017065001- 0.02560699  0.005027845 0.018938411- 0.04997214  1.806124- 1 ##
Research1  ##
0.3017751 0 ##
0.8200000 1 ##
##
:Coefficients of linear discriminants ##
LD1            ##
PC1      -0.67515878 ##
PC2       0.20497834 ##
```

```
PC3        0.12227655 ##
PC4       -0.02505001 ##
PC5        0.18252004 ##
PC6       -0.37814140 ##
Research1  0.49703659 ##
```

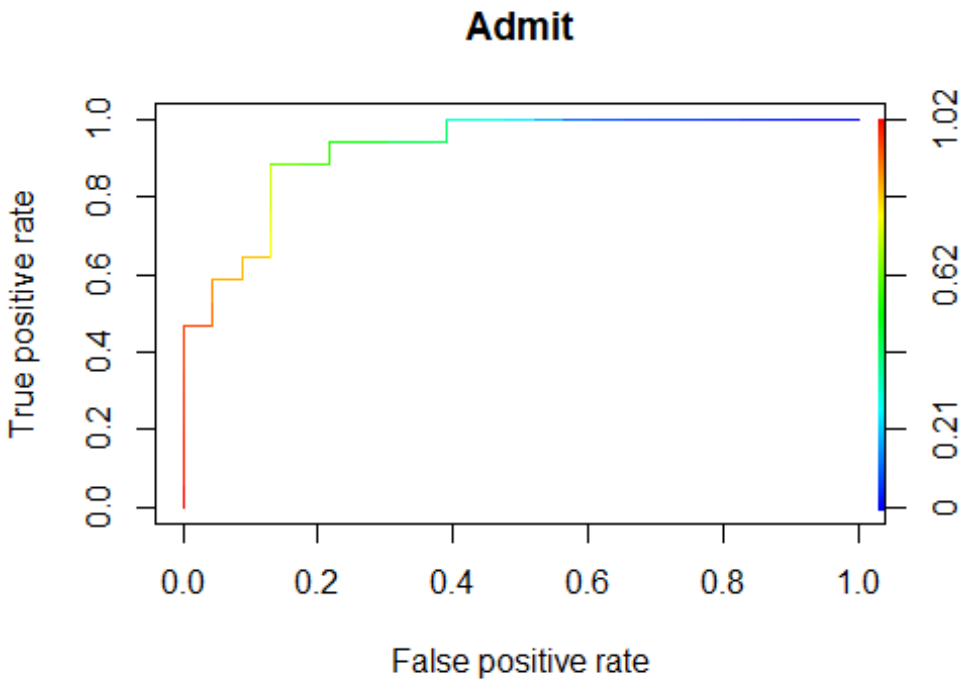
برای آزمودن این مدل از Cross Valodation k-fold استفاده میکنیم که تابع آن به صورت زیر تعریف میشود.

```
ldaCrossValidation=function(df,ldaModel,k){
                                    set.seed(100)
                  indexes=sample(nrow(df) ,replace = FALSE)
        foldsIndexes = cut(indexes , breaks=k , labels=FALSE)
                                     accuracy=c()
myFormula=formula(paste(format(terms(ldaModel)),collapse = ""))

 myResponse=(as.character(attr(terms(myFormula),"variables"))[-
                1])[attr(terms(myFormula),"response")]
       myResponse=str_replace_all(myResponse ,"`","")
                                  for (i in 1:k) {
    indexesOfTest = which(foldsIndexes==i , arr.ind=TRUE)
                          trainData=df[-indexesOfTest,]
                            testData=df[indexesOfTest,]
      myModel=lda(formula = myFormula ,data = trainData )
                        myPred=predict(ldaModel,testData)
 accuracy[i]=sum(myPred$class==testData$Admit)/dim(testData)[1]
                                                  }
                                 output=mean(accuracy)
                                       return(output)
                                                  }

                        ldaCrossValidation(admission,ldaModel,5)

                                    0.8922785 [1] ##

                     ldaPred=predict(ldaModel,admission)
                                Admit=admission$Admit
                              ldaPreds=ldaPred$class
                           tableA=table(ldaPreds,Admit)
                                              tableA

                                      Admit      ##
                                ldaPreds  0   1 ##
                                  26  202 0     ##
                                  154 17  1     ##

truePositive=sum(admission$Admit==1 & admission$Admit==ldaPred$class)
                                         truePositive

                                    154 [1] ##
```

```
falsePositive=sum(admission$Admit==0 & admission$Admit!=ldaPred$class)
falsePositive

## [1] 17

trueNegative=sum(admission$Admit==0 & admission$Admit==ldaPred$class)
trueNegative

## [1] 202

falseNegative=sum(admission$Admit==1 & admission$Admit!=ldaPred$class)
falseNegative

## [1] 26

precision=truePositive/(truePositive+falsePositive)
precision

## [1] 0.9005848

recall=truePositive/(truePositive+falseNegative)
recall

## [1] 0.8555556
```

اکنون ROC را برای lda رسم میکنیم.

```
indexes=sample(nrow(admission) ,replace = FALSE)
foldsIndexes = cut(indexes , breaks=10 , labels=FALSE)
indexesOfTest = which(foldsIndexes==1 , arr.ind=TRUE)
trainData=admission[-indexesOfTest,]
testData=admission[indexesOfTest,]

model=lda(Admit~.,data=trainData)
qdaPred=predict(model,testData,type = "response")
pred = prediction(qdaPred$posterior[,2], testData$Admit)
perf = performance(pred,"tpr","fpr")
plot(perf,colorize=TRUE,main="Admit")
```



مدل QDA

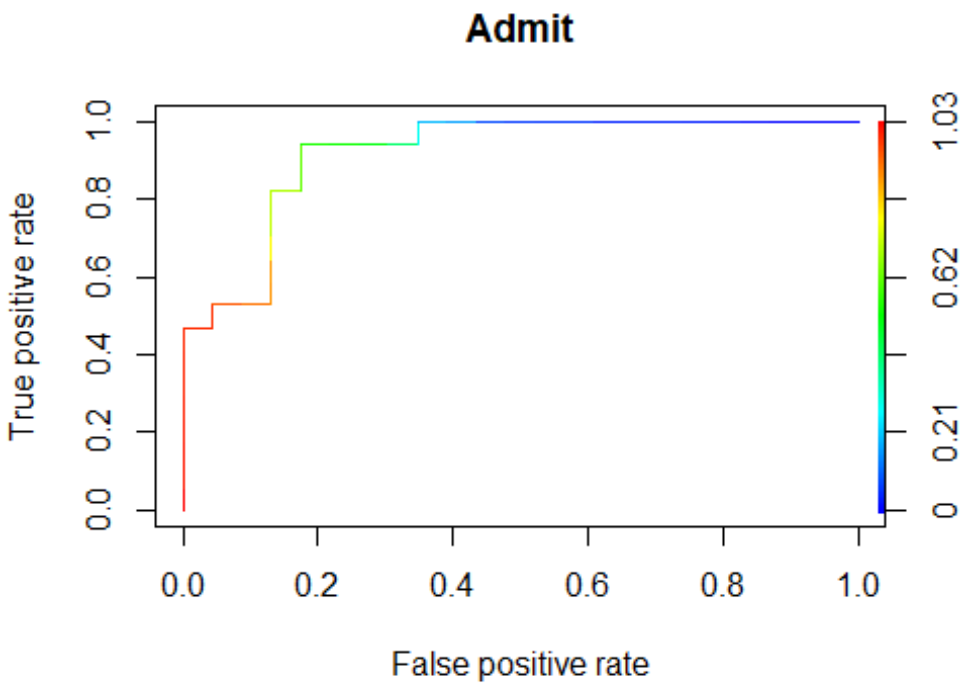
```
qdaModel=qda(Admit~.,data=admission,subset=indexOfDataTrain)
qdaModel

:Call ##
qda(Admit ~ ., data = admission, subset = indexOfDataTrain) ##
##
:Prior probabilities of groups ##
        1         0      ##
0.4702194 0.5297806 ##
##
:Group means ##
                  PC1         PC2         PC3         PC4         PC5         PC6      ##
0.007786459 0.00862079- 0.006063242 0.005238275- 0.11129163- 1.430367  0 ##
0.017065001- 0.02560699 0.005027845 0.018938411- 0.04997214  1.806124- 1 ##
                                                                Research1  ##
                                                                0.3017751 0 ##
                                                                0.8200000 1 ##

qdaCrossValidation=function(df,qdaModel,k){
                                                    set.seed(100)
                                  indexes=sample(nrow(df) ,replace = FALSE)
                        foldsIndexes = cut(indexes , breaks=k , labels=FALSE)
                                                                  v=c()
                  myFormula=formula(paste(format(terms(qdaModel)),collapse = ""))
```

```
myResponse=(as.character(attr(terms(myFormula),"variables"))[-
1])[attr(terms(myFormula),"response")]
myResponse=str_replace_all(myResponse ,"`","")
for (i in 1:k) {
indexesOfTest = which(foldsIndexes==i , arr.ind=TRUE)
trainData=df[-indexesOfTest,]
testData=df[indexesOfTest,]
myModel=qda(formula = myFormula ,data = trainData)
myPred=predict(ldaModel,testData)

v[i]=sum(myPred$class==testData$Admit)/dim(testData)[1]
}
output=mean(v)
return(output)
}

qdaCrossValidation(admission,qdaModel,5)

0.8922785 [1] ##

qdaPred=predict(qdaModel,admission)
Admit=admission$Admit
qdaPreds=qdaPred$class
tableA=table(qdaPreds,Admit)
tableA

Admit      ##
qdaPreds  0   1 ##
23  193 0     ##
157 26  1     ##

truePositive=sum(admission$Admit==1 & admission$Admit==qdaPred$class)
truePositive

157 [1] ##

falsePositive=sum(admission$Admit==0 & admission$Admit!=qdaPred$class)
falsePositive

26 [1] ##

trueNegative=sum(admission$Admit==0 & admission$Admit==qdaPred$class)
trueNegative

193 [1] ##

falseNegative=sum(admission$Admit==1 & admission$Admit!=qdaPred$class)
falseNegative

23 [1] ##
```

```
precision=truePositive/(truePositive+falsePositive)
precision

0.8579235 [1] ##

recall=truePositive/(truePositive+falseNegative)
recall

0.8722222 [1] ##
```

اکنون ROC را برای qda رسم میکنیم.

```
indexes=sample(nrow(admission) ,replace = FALSE)
foldsIndexes = cut(indexes , breaks=10 , labels=FALSE)
indexesOfTest = which(foldsIndexes==1 , arr.ind=TRUE)
trainData=admission[-indexesOfTest,]
testData=admission[indexesOfTest,]

model=qda(Admit~.,data=trainData)
qdaPred=predict(model,testData,type = "response")
pred = prediction(qdaPred$posterior[,2], testData$Admit)
perf = performance(pred,"tpr","fpr")
plot(perf,colorize=TRUE,main="Admit")
```



اکنون با دیدن نمودار های ROC و مقایسه ی precision، recall و accuracy مدل های مختلف میتوانیم بگوییم که مدل KNN با k=1 بهترین عملکرد را داشته.