Logistic regression in medical-school admissions

Gustavo Cepparo and Milica Cudina

The data set we are going to use exists in the `Stat2Data` package. You only need to install it once. Here are the command lines for reference.

```
#install.packages("Stat2Data")
library(Stat2Data)
data(MedGPA)
```

Next, we look at the included data set:

```
data<-MedGPA
data$Acceptance <- as.factor(data$Acceptance)
attach(data)
dim(data)
```

```
## [1] 55 11
```

Now, let's see how good the `GPA` alone is at predicting `Acceptance`.

```
glm.fits=glm(Acceptance~GPA,data=data,family=binomial)
summary(glm.fits)
```

```
##
## Call:
## glm(formula = Acceptance ~ GPA, family = binomial, data = data)
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept)  -19.207      5.629  -3.412 0.000644 ***
## GPA            5.454      1.579   3.454 0.000553 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 75.791  on 54  degrees of freedom
## Residual deviance: 56.839  on 53  degrees of freedom
## AIC: 60.839
##
## Number of Fisher Scoring iterations: 4
```

How do we predict with this model?

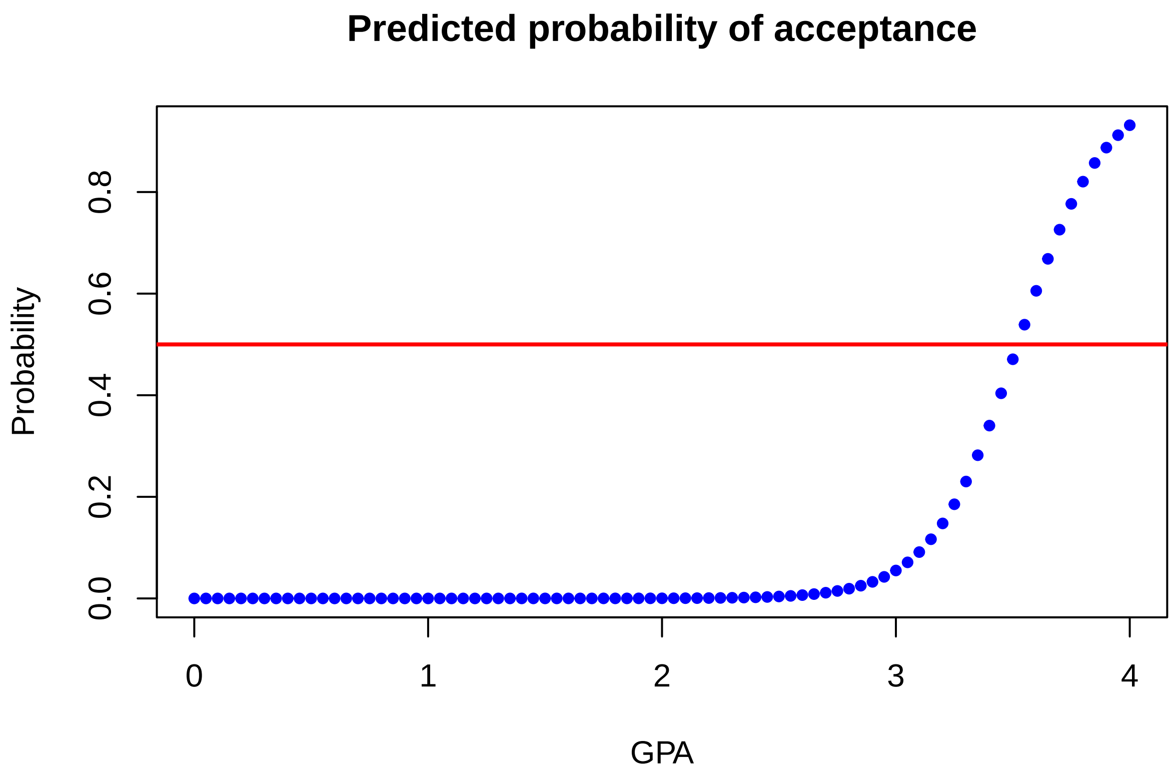
```
#we'll look at an imaginary student with GPA equal to 3.86
violet=data.frame(GPA=3.86)
violet.prob=predict(glm.fits,violet,type="response")
violet.prob
```

```
##         1
```

```
## 0.8637248
```

What about the probability of being accepted as a function of the GPA?

```
mesh=data.frame(GPA=seq(0,4,0.05))
acc.prob=predict(glm.fits,mesh,type="response")
plot(seq(0,4,0.05), acc.prob,
     col="blue", pch=20,
     main="Predicted probability of acceptance",
     xlab="GPA", ylab="Probability")
abline(h=0.5, col="red", lwd=2)
```



Now, what about the confusion matrix?

```
glm.probs=predict(glm.fits,type="response")
glm.pred=rep("yes",55)
glm.pred[glm.probs<.5]="no"
#Confusion Matrix Below
tab=table(glm.pred,Acceptance)
tab
```

```
##         Acceptance
## glm.pred  0  1
##      no  16  6
##      yes  9 24
```

```
good=(tab[1,1]+tab[2,2])/sum(tab)
good
```

```
## [1] 0.7272727
```

Let's "improve" our model and method and do a 2−fold cross-validation.

The "improvement" in the model is to also include `MCAT` and `Sex` as explanatory variables.

```r
glm.fits.m=glm(Acceptance~MCAT+GPA+Sex,data=data,
               family=binomial)
summary(glm.fits.m)
```

```
##
## Call:
## glm(formula = Acceptance ~ MCAT + GPA + Sex, family = binomial,
##     data = data)
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept) -23.9851     6.9685  -3.442 0.000578 ***
## MCAT          0.1809     0.1080   1.675 0.093946 .
## GPA           5.1392     1.8508   2.777 0.005491 **
## SexM         -1.2580     0.7303  -1.723 0.084965 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 75.791  on 54  degrees of freedom
## Residual deviance: 50.786  on 51  degrees of freedom
## AIC: 58.786
##
## Number of Fisher Scoring iterations: 5
```

```r
cor(MCAT, GPA)
```

```
## [1] 0.5414202
```

The `GPA` is still the most significant. Let's create the training subset and fit the model on it.

```r
n=length(GPA)
set.seed(1)
train=sample(n,floor(n/2))
glm.fits.tr=glm(Acceptance~MCAT+GPA+Sex,data=data,
                family=binomial, subset=train)
summary(glm.fits.tr)
```

```
##
## Call:
## glm(formula = Acceptance ~ MCAT + GPA + Sex, family = binomial,
##     data = data, subset = train)
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept) -22.0670    10.1235  -2.180   0.0293 *
## MCAT          0.2841     0.1563   1.818   0.0691 .
## GPA           3.3615     2.3373   1.438   0.1504
## SexM         -0.3355     0.9636  -0.348   0.7277
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 37.096  on 26  degrees of freedom
```

```
## Residual deviance: 26.760  on 23  degrees of freedom
## AIC: 34.76
##
## Number of Fisher Scoring iterations: 5
```

How did we do on the training set?

```r
glm.probs.tr=predict(glm.fits.tr, type="response")
glm.pred.tr=rep("yes",floor(n/2))
glm.pred.tr[glm.probs.tr<.5]="no"
#Confusion Matrix Below
tab.tr=table(glm.pred.tr,Acceptance[train])
tab.tr
```

```
##
## glm.pred.tr  0  1
##          no   9  2
##          yes  3 13
```

```r
good.tr=(tab.tr[1,1]+tab.tr[2,2])/sum(tab.tr)
good.tr
```

```
## [1] 0.8148148
```

Next, we predict using the above model on the validation set, i.e., the complement of `train`.

```r
val=data[-train,]
glm.probs.v=predict(glm.fits.tr, data=val ,type="response")
glm.probs.v
```

```
##          4          39           1          34          23          43          14
## 0.84563130 0.64594069 0.71038327 0.42415987 0.91597345 0.81236250 0.23569238
##         18          33          21          46          42          10           7
## 0.64035661 0.72163122 0.88615598 0.64369677 0.26001509 0.66116120 0.88468156
##          9          15          45          37          25          52          38
## 0.58601074 0.09810083 0.25359969 0.43158911 0.22701696 0.31511157 0.04892427
##         40          47          20           3           6          50
## 0.25888331 0.99271563 0.98674824 0.13773764 0.58168610 0.79403402
```

How did we do?

```r
glm.pred.v=rep("yes", n-floor(n/2))
glm.pred.v[glm.probs.v<.5]="no"
#Confusion Matrix Below
tab.v=table(glm.pred.v,val$Acceptance)
tab.v
```

```
##
## glm.pred.v 0 1
##        no   4 7
##        yes 9 8
```

```r
good.pred=(tab.v[1,1]+tab.v[2,2])/sum(tab.v)
good.pred
```

```
## [1] 0.4285714
```