Lab-04.Rmd

Logistic Regression

Part 1:

First we load the data and get the basic info for the dataset. We can use the summary() command

```
mydata <-read.csv("https://stats.idre.ucla.edu/stat/data/binary.csv")

summary(mydata)
```

```
##      admit             gre             gpa             rank
##  Min.   :0.0000   Min.   :220.0   Min.   :2.260   Min.   :1.000
##  1st Qu.:0.0000   1st Qu.:520.0   1st Qu.:3.130   1st Qu.:2.000
##  Median :0.0000   Median :580.0   Median :3.395   Median :2.000
##  Mean   :0.3175   Mean   :587.7   Mean   :3.390   Mean   :2.485
##  3rd Qu.:1.0000   3rd Qu.:660.0   3rd Qu.:3.670   3rd Qu.:3.000
##  Max.   :1.0000   Max.   :800.0   Max.   :4.000   Max.   :4.000
```

Alternatively we can use the view() command

```
#view(mydata)
```

Part 2:

To check the number of observations, we can use the nrow command

```
nrow(mydata)
```

```
## [1] 400
```

Part 3:

We can use sapply to check the standard deviation of the variables in the dataset. We exclude rank as it is a catgorical variable.

```
sapply(mydata [,-4], sd)
```

```
##       admit         gre         gpa
##   0.4660867 115.5165364   0.3805668
```

Likewise we can use sapply for the mean values

```
sapply(mydata [,-4], mean)
```

```
##    admit      gre      gpa
##   0.3175 587.7000   3.3899
```

Part 4:

```r
mydata$rank <- factor(mydata$rank)

mydata$rank
```

```
##   [1] 3 3 1 4 4 2 1 2 3 2 4 1 1 2 1 3 4 3 2 1 3 2 4 4 2 1 1 4 2 1 4 3 3 3 1
##  [36] 2 1 3 2 3 2 2 2 3 2 3 2 4 4 3 3 4 4 2 3 3 3 3 2 4 2 4 3 3 3 2 4 1 1 1
##  [71] 3 4 4 2 4 3 3 3 1 1 4 2 2 4 3 2 2 2 1 2 2 1 2 2 2 2 4 2 2 3 3 3 4 3 2
## [106] 2 1 2 3 2 4 4 3 1 3 3 2 2 1 3 2 2 3 3 3 4 1 4 2 4 2 2 2 3 2 3 4 3 2 1
## [141] 2 4 4 3 4 3 2 3 1 1 1 2 2 3 3 4 2 1 2 3 2 2 2 2 2 1 4 3 3 3 3 3 3 2 4
## [176] 2 2 3 3 3 3 4 2 2 4 2 3 2 2 2 2 3 3 4 2 2 3 4 3 4 3 2 1 4 1 3 1 1 3 2
## [211] 4 2 2 3 2 3 1 1 1 2 3 3 1 3 2 3 2 4 2 2 4 3 2 3 1 2 2 2 4 3 2 1 3 2 1
## [246] 3 2 2 3 3 4 4 2 4 4 3 2 3 2 2 2 2 3 3 3 3 4 3 2 3 2 3 2 1 2 2 3 1 4 2
## [281] 2 3 4 4 2 4 1 4 4 4 2 2 2 1 1 3 1 2 2 3 2 3 2 2 3 4 1 2 2 3 3 2 3 4 4
## [316] 2 2 4 4 1 3 2 4 2 3 1 2 2 2 4 3 3 1 3 3 1 3 4 1 3 4 3 4 2 3 3 2 2 2 2
## [351] 2 3 3 2 2 1 2 1 3 3 1 1 2 2 1 3 3 3 1 2 2 3 1 1 2 4 2 2 3 2 2 2 2 1 2
## [386] 1 2 2 2 2 2 2 3 2 3 2 3 2 2 3
## Levels: 1 2 3 4
```

Part 5:

To fit a logistic model to the data, we can use the glm (generalised linear model) command

```r
glm.admit.fit = glm(admit ~ gre + gpa + rank, data = mydata, family = "binomial")
```

Now we can use the summary() command to get details about the model

```r
summary(glm.admit.fit)
```

```
##
## Call:
## glm(formula = admit ~ gre + gpa + rank, family = "binomial",
##     data = mydata)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.6268  -0.8662  -0.6388   1.1490   2.0790
##
## Coefficients:
##              Estimate Std. Error z value Pr(>|z|)
## (Intercept) -3.989979   1.139951  -3.500 0.000465 ***
## gre          0.002264   0.001094   2.070 0.038465 *
## gpa          0.804038   0.331819   2.423 0.015388 *
## rank2       -0.675443   0.316490  -2.134 0.032829 *
## rank3       -1.340204   0.345306  -3.881 0.000104 ***
## rank4       -1.551464   0.417832  -3.713 0.000205 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 499.98  on 399  degrees of freedom
## Residual deviance: 458.52  on 394  degrees of freedom
## AIC: 470.52
##
```

```
## Number of Fisher Scoring iterations: 4
```

Part 6:

This process is standard One-hot encoding

Part 7:

All the variable z-values are large, and all the variable p values are small (<0.05) hence all the variable are statistically significant

Part 8:

To obtain the raw probability predictions for the training data, we can use the following command

```
admit.prob = predict(glm.admit.fit, type = "response")
```

We can check the values by using the head() command, which will display the first n rows

```
head(admit.prob)
```

```
##         1         2         3         4         5         6
## 0.1726265 0.2921750 0.7384082 0.1783846 0.1183539 0.3699699
```

Part 9:

To convert the probabilities to class, first we create a new vector of lenght 400 (to represent each of the 400 observarions)

```
admit.pred = rep(1,400)
```

We then use the probability data, can use a simple conditional to assign it a class. In this class if a probability if 0.5 or above for a class, it will be assigned to the class.

```
admit.pred[admit.prob<0.5]=0
```

We can again used the head command to show the data

```
head(admit.pred)
```

```
## [1] 0 0 1 0 0 0
```

Part 10:

We can use the table command to create a confusion matrix

```
table(admit.pred, mydata$admit)
```

```
##
## admit.pred   0   1
##          0 254  97
##          1  19  30
```

Part 11:

First we construct a new dataframe with the mean gre and gpa values for each ranks

```
newdata1 <- with(mydata, data.frame(gre = mean(gre), gpa = mean(gpa), rank = factor(1:4)))

newdata1
```

```
##     gre    gpa rank
## 1 587.7 3.3899    1
## 2 587.7 3.3899    2
## 3 587.7 3.3899    3
## 4 587.7 3.3899    4
```

We can then fit the linear model to the mean value to compute raw probabilities

```
newdata1$admit1.prob = predict(glm.admit.fit, newdata = newdata1, type = "response")

newdata1
```

```
##     gre    gpa rank admit1.prob
## 1 587.7 3.3899    1   0.5166016
## 2 587.7 3.3899    2   0.3522846
## 3 587.7 3.3899    3   0.2186120
## 4 587.7 3.3899    4   0.1846684
```

Finally we normalise the probabilities to predict a class, using the same process we used in part 9

```
newdata1$admit1.pred = rep(1,4)

newdata1$admit1.pred[newdata1$admit1.prob<0.5]=0

newdata1
```

```
##     gre    gpa rank admit1.prob admit1.pred
## 1 587.7 3.3899    1   0.5166016           1
## 2 587.7 3.3899    2   0.3522846           0
## 3 587.7 3.3899    3   0.2186120           0
## 4 587.7 3.3899    4   0.1846684           0
```