Practical - 7

Shivam Tawari (A-58)

Aim: Write a R program to implement logistic regression.

Theory:

What is Regression analysis:
Regression analysis is predictive technique used to predict a continuous quantity. A continuous variable has infinite possibilities.

Logistic Regression:
It is one of the most widely used algorithm for solving classification problem. It is a method used to predict a dependant variable (y) given independent variable (x) such that dependant variable is categorical.

Types of Logistic Regression:
1. Binary Logistic Regression
2. Multinomical Regression
3. Ordinal Logistic Regression

Code:

```
mydata ← read.csv ("binary.csv")
head (mydata)
xtabs (~admit + rank, data = mydata)
mydata $ rank ← factor (mydata $ rank)
mylogit ← glm (admit ~ gre + gpa + rank,
               data = mydata, family = "binomial")
summary (mylogit)
```

→ In the above output the first thing we see is the call, this is R reminding us what the options we specified.

→ Next we see the deviance residual which are measure of model fit. This part of output, show the distribution of the deviance residual of individual cases used in model.

→ Next part of output shows the coefficients their standard error, the z statistic and associated P values. Both GRE and GPA are important as three terms for rank. The logistic regression coefficients gives the change in the log odds of the outcome for one unit increase in predicted variable.

Example 2:

```
install.packages ('mlbench')
install.packages ('caret')
library (caret)

data (BreastCancer , package = 'mlbench')
bc ← BreastCancer [complete.cases (BreastCancer), ]

print (str(bc))

model ← glm (Class ~ Cell.shape , family = 'binomial',
                                   data = bc)

bc ← bc [, -1]
for (i in 1:9)
{
  bc [, i] ← as.numeric (as.character (bc [, i]))
}

bc $ Class ← ifelse (bc $ Class == "malignant", 1, 0)
bc $ Class ← factor (bc $ Class , levels = c(0, 1))

'%ni%' ← Negate ('%in%')
options (scipen = 999)
set.seed (100)
```

```
trainDataIndex ← createDataPartition (bc$Class,
                    p = 0.7, list = F)
trainData ← bc [trainDataIndex,]
testData ← bc [trainDataIndex,]

set.seed (100)
down_train ← downSample (x = trainData [, colnames
              (trainData) %ni% "Class"],
              y = trainData $Class)
table (down_train $Class)

set.seed (100)
up_train ← upSample (x = trainData [, colnames
            (trainData) %ni% "Class"],
            y = trainData $Class)
table (up_train $Class)

logitmod ← glm (Class ~ Cl.thickness + Cell.size +
            Cell.Shape, family = "binomial",
            data = down_train)
summary (logitmod)

pred ← predict (logitmod, newdata = testData,
                type = "response")
pred
```

```
y_pred_num ← ifelse (pred > 0.5, 1, 0)
y_pred ← factor (y_pred_num, levels = c(0,1))
y_act ← testData $ Class

mean (y_pred == y_act)
```

Conclusion: Hence, we have successfully implemented the program in R for implementing logistic regression.

Code:

```
1  #Example 1
2  #Shivam Tawari A-58
3  mydata <- read.csv("/cloud/project/binary.csv")
4  head(mydata)
5
6  xtabs (~admit + rank,data = mydata)
7
8  mydata$rank <- factor(mydata$rank)
9  mylogit <- glm(admit ~ gre + gpa + rank, data=mydata, family = "binomial")
10 summary(mylogit)
11
```

Output:

```
Console  Terminal  Jobs
/cloud/project/
> #Shivam Tawari A-58
> mydata <- read.csv("/cloud/project/binary.csv")
> head(mydata)
  admit gre  gpa rank
1     0 380 3.61    3
2     1 660 3.67    3
3     1 800 4.00    1
4     1 640 3.19    4
5     0 520 2.93    4
6     1 760 3.00    2
> xtabs (~admit + rank,data = mydata)
     rank
admit  1  2  3  4
    0 28 97 93 55
    1 33 54 28 12
> mydata$rank <- factor(mydata$rank)
> mylogit <- glm(admit ~ gre + gpa + rank, data=mydata, family = "binomial")
> summary(mylogit)

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 ***
```

```
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
```

Code:

```r
#Example 2
#Shivam Tawari A-58

install.packages('mlbench')
install.packages('caret')
library(caret)

data(BreastCancer, package='mlbench')
bc <- BreastCancer[complete.cases(BreastCancer),]

print(str(bc))

model <- glm(Class ~ Cell.shape, family = 'binomial', data = bc)

bc <- bc[,-1]
for (i in 1:9)
{
  bc [,i] <- as.numeric(as.character(bc[,i]))
}

bc$Class <- ifelse(bc$Class == "malignant", 1, 0)
bc$Class <- factor(bc$Class, levels = c(0, 1))

'%ni%' <- Negate('%in%')
options(scipen=999)
set.seed(100)
trainDataIndex <- createDataPartition(bc$Class, p=0.7,list=F)
trainData <- bc[trainDataIndex,]
```

```r
testData <- bc[trainDataIndex,]

set.seed(100)
down_train <- downSample(x = trainData[, colnames(trainData) %ni% "Class"],
                         y = trainData$Class)
table(down_train$Class)

set.seed(100)
up_train <- upSample(x = trainData[, colnames(trainData) %ni% "Class"],
                     y = trainData$Class)
table(up_train$Class)

logitmod <- glm(Class ~ Cl.thickness + Cell.size + Cell.shape, family = "binomia
summary(logitmod)

pred <- predict(logitmod, newdata = testData, type = "response")
pred

y_pred_num <- ifelse(pred > 0.5, 1, 0)
y_pred <- factor(y_pred_num, levels=c(0, 1))
y_act <- testData$Class

mean(y_pred == y_act)
```

Output:

```
> library(caret)
> data(BreastCancer, package='mlbench')
> bc <- BreastCancer[complete.cases(BreastCancer),]
> print(str(bc))
'data.frame':   683 obs. of  11 variables:
 $ Id             : chr  "1000025" "1002945" "1015425" "1016277" ...
 $ Cl.thickness   : Ord.factor w/ 10 levels "1"<"2"<"3"<"4"<..: 5 5 3 6 4 8 1 2 2 4
...
 $ Cell.size      : Ord.factor w/ 10 levels "1"<"2"<"3"<"4"<..: 1 4 1 8 1 10 1 1 1 2
...
 $ Cell.shape     : Ord.factor w/ 10 levels "1"<"2"<"3"<"4"<..: 1 4 1 8 1 10 1 2 1 1
...
 $ Marg.adhesion  : Ord.factor w/ 10 levels "1"<"2"<"3"<"4"<..: 1 5 1 1 3 8 1 1 1 1
...
 $ Epith.c.size   : Ord.factor w/ 10 levels "1"<"2"<"3"<"4"<..: 2 7 2 3 2 7 2 2 2 2
...
 $ Bare.nuclei    : Factor w/ 10 levels "1","2","3","4",..: 1 10 2 4 1 10 10 1 1 1 ...
 $ Bl.cromatin    : Factor w/ 10 levels "1","2","3","4",..: 3 3 3 3 3 9 3 3 1 2 ...
 $ Normal.nucleoli: Factor w/ 10 levels "1","2","3","4",..: 1 2 1 7 1 7 1 1 1 1 ...
 $ Mitoses        : Factor w/ 9 levels "1","2","3","4",..: 1 1 1 1 1 1 1 1 5 1 ...
 $ Class          : Factor w/ 2 levels "benign","malignant": 1 1 1 1 1 2 1 1 1 1 ...
NULL
> model <- glm(Class ~ Cell.shape, family = 'binomial', data = bc)
> bc <- bc[,-1]
> for (i in 1:9)
+ {
+   bc [,i] <- as.numeric(as.character(bc[,i]))
+ }
```

Console | Terminal | Jobs

/cloud/project/

```
> bc$Class <- ifelse(bc$Class == "malignant", 1, 0)
> bc$Class <- factor(bc$Class, levels = c(0, 1))
> '%ni%' <- Negate('%in%')
> options(scipen=999)
> set.seed(100)
> trainDataIndex <- createDataPartition(bc$Class, p=0.7,list=F)
> trainData <- bc[trainDataIndex,]
> testData <- bc[trainDataIndex,]
> set.seed(100)
> down_train <- downSample(x = trainData[, colnames(trainData) %ni% "Class"],
+                          y = trainData$Class)
> table(down_train$Class)

  0   1
168 168
> set.seed(100)
> up_train <- upSample(x = trainData[, colnames(trainData) %ni% "Class"],
+                      y = trainData$Class)
> table(up_train$Class)

  0   1
311 311
> logitmod <- glm(Class ~ Cl.thickness + Cell.size + Cell.shape, family = "binomial",
data=down_train)
> summary(logitmod)

Call:
glm(formula = Class ~ Cl.thickness + Cell.size + Cell.shape,
    family = "binomial", data = down_train)
```

```
Deviance Residuals:
    Min       1Q   Median       3Q      Max
-3.8967  -0.2158  -0.0370   0.0631   2.7428

Coefficients:
             Estimate Std. Error z value          Pr(>|z|)
(Intercept)    -7.9695     1.0724  -7.431 0.000000000000108 ***
Cl.thickness    0.7035     0.1488   4.728 0.000002264855263 ***
Cell.size       0.7201     0.2543   2.832           0.00462 **
Cell.shape      0.6974     0.2330   2.993           0.00277 **
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 465.795  on 335  degrees of freedom
Residual deviance:  88.447  on 332  degrees of freedom
AIC: 96.447

Number of Fisher Scoring iterations: 8

> pred <- predict(logitmod, newdata = testData, type = "response")
> pred
          1           2           3           4           6           8          12
0.045892971 0.771711712 0.011641255 0.999495516 0.999992743 0.011570800 0.005794679
         14          16          18          19          20          21          22
0.002875879 0.982359695 0.023248829 0.999875120 0.088591878 0.624845947 0.997877544
         23          26          28          29          30          32          33
0.011641255 0.285038606 0.045892971 0.005794679 0.011500766 0.005794679 0.999875120
         35          36          39          40          42          43          44
```

Console | Terminal | Jobs

/cloud/project/

```
0.088097214 0.991753616 0.045892971 0.045624787 0.162563021 0.011641255 0.011641255
        602         603         604         606         607         609         610
0.002875879 0.023248829 0.931677311 0.999934926 0.023248829 0.999940110 0.045892971
        611         613         614         617         619         621         622
0.982676173 0.999992743 0.024014980 0.011641255 0.023248829 0.011641255 0.623405110
        624         626         627         628         629         630         631
0.002875879 0.045358096 0.966087333 0.005794679 0.005794679 0.023248829 0.446186581
        632         633         634         635         636         637         638
0.045892971 0.002875879 0.995884591 0.011641255 0.087115367 0.999985602 0.283788599
        640         641         642         643         644         645         648
0.045892971 0.023248829 0.011641255 0.011641255 0.002875879 0.005794679 0.002875879
        649         650         652         653         654         655         656
0.999940110 0.011641255 0.005891124 0.045892971 0.023248829 0.011641255 0.011641255
        658         659         660         661         662         663         664
0.871621597 0.999750296 0.002875879 0.002875879 0.023248829 0.011500766 0.011500766
        666         667         668         669         670         672         673
0.002875879 0.165622432 0.011641255 0.967021128 0.999940110 0.023628875 0.005794679
        674         677         678         679         680         681         682
0.289696570 0.005759402 0.045892971 0.002875879 0.005794679 0.999998223 0.999940110
        684         685         686         687         688         689         691
0.002875879 0.002875879 0.002875879 0.002875879 0.011641255 0.023248829 0.002875879
        692         695         696
0.999940110 0.011641255 0.005794679
> y_pred_num <- ifelse(pred > 0.5, 1, 0)
> y_pred <- factor(y_pred_num, levels=c(0, 1))
> y_act <- testData$Class
> mean(y_pred == y_act)
[1] 0.9498956
```