Sardar Patel Institute of Technology,Mumbai
Department of Electronics and Telecommunication Engineering
T.E. Sem-V (2018-2019)
ETL54-Statistical Computational Laboratory

Lab-4: Classification Analysis and Modeling

Name:Manish Dsilva **Roll No.15**

Objective: To carry out logistic regression (including multiple regression and multiclass regression) and build a regression model

Outcomes:

1. Part-I: Implement Logistic Regression for University Admission dataset :
   Target variable: Admit or reject
   Predictors or independent variables: GRE, RANK, GPA
2. Part-II: Implement Multinomial Logistics Regression for iris

System Requirements: Ubuntu OS with R and RStudio installed

Logistic Regression

We use the logistic regression equation to predict the probability of a dependent variable taking the dichotomy values 0 or 1. Suppose x1, x2, ..., xp are the independent variables, α and βk (k = 1, 2, ..., p) are the parameters, and E(y) is the expected value of the dependent variable y, then the logistic regression equation is:

$$E(y) = 1/(1 + e^{-(\alpha + \sum_k \beta_k x_k)})$$

For example, in the built-in data set ***mtcars,*** the data column am represents the transmission type of the automobile model (0 = automatic, 1 = manual).
With the logistic regression equation, we can model the probability of a manual transmission in a vehicle based on its engine horsepower and weight data.

$$P(Manual\ Transmission) = 1/(1 + e^{-(\alpha + \beta_1 * Horsepower + \beta_2 * Weight)})$$

Estimated Logistic Regression Equation
Using the generalized linear model, an estimated logistic regression equation can be formulated as below. The coefficients a and bk (k = 1, 2, ..., p) are determined according to a maximum likelihood approach, and it allows us to estimate the probability of the dependent variable y taking on the value 1 for given values of xk (k = 1, 2, ..., p).

$$Estimate\ of\ P(y=1 \mid x_1, \ldots x_p) = 1/(1+e^{-(a+\sum_k b_k x_k)})$$

We apply the function ***glm*** to a formula that describes the transmission type (am) by the horsepower (hp) and weight (wt). This creates a generalized linear model (GLM) in the binomial family.
In R:

```
#Build a model:
am.glm = glm(formula=am ~ hp + wt, data=mtcars, family=binomial)
#Test data
newdata = data.frame(hp=120, wt=2.8)
#Predict
predict(am.glm, newdata, type="response")
```

Let's start with an **example confusion matrix for a binary classifier** (though it can easily be extended to the case of more than two classes):

n=165	**Predicted: NO**	**Predicted: YES**
Actual: NO	50	10
Actual: YES	5	100

What can we learn from this matrix?

- There are two possible predicted classes: "yes" and "no". If we were predicting the presence of a disease, for example, "yes" would mean they have the disease, and "no" would mean they don't have the disease.
- The classifier made a total of 165 predictions (e.g., 165 patients were being tested for the presence of that disease).
- Out of those 165 cases, the classifier predicted "yes" 110 times, and "no" 55 times.
- In reality, 105 patients in the sample have the disease, and 60 patients do not.

Let's now define the most basic terms, which are whole numbers (not rates):

- **true positives (TP):** These are cases in which we predicted yes (they have the disease), and they do have the disease.
- **true negatives (TN):** We predicted no, and they don't have the disease.

- **false positives (FP):** We predicted yes, but they don't actually have the disease. (Also known as a "Type I error.")
- **false negatives (FN):** We predicted no, but they actually do have the disease. (Also known as a "Type II error.")

I've added these terms to the confusion matrix, and also added the row and column totals:

n=165	Predicted: NO	Predicted: YES	
Actual: NO	TN = 50	FP = 10	60
Actual: YES	FN = 5	TP = 100	105
	55	110	

This is a list of rates that are often computed from a confusion matrix for a binary classifier:

- **Accuracy:** Overall, how often is the classifier correct?
    - (TP+TN)/total = (100+50)/165 = 0.91
- **Misclassification Rate:** Overall, how often is it wrong?
    - (FP+FN)/total = (10+5)/165 = 0.09
    - equivalent to 1 minus Accuracy
    - also known as "Error Rate"
- **True Positive Rate:** When it's actually yes, how often does it predict yes?
    - TP/actual yes = 100/105 = 0.95
    - also known as "Sensitivity" or "Recall"
- **False Positive Rate:** When it's actually no, how often does it predict yes?
    - FP/actual no = 10/60 = 0.17
- **True Negative Rate:** When it's actually no, how often does it predict no?
    - TN/actual no = 50/60 = 0.83
    - equivalent to 1 minus False Positive Rate
    - also known as "Specificity"
- **Precision:** When it predicts yes, how often is it correct?
    - TP/predicted yes = 100/110 = 0.91
- **Prevalence:** How often does the yes condition actually occur in our sample?
    - actual yes/total = 105/165 = 0.64

A couple other terms are also worth mentioning:

- **Null Error Rate:** This is how often you would be wrong if you always predicted the majority class. (In our example, the null error rate would be 60/165=0.36 because if you always predicted yes, you would only be wrong for the 60 "no" cases.) This can be a useful baseline metric to compare your classifier against. However, the best classifier for a particular application will sometimes have a higher error rate than the null error rate, as demonstrated by the **Accuracy Paradox**
- **F Score:** This is a weighted average of the true positive rate (recall) and precision.

Output: Binary Logistic Regression

Step 1: Data Load and Pre-Processing

```
> mydata <- read.csv("~/Downloads/binary.csv",header=T)
> str(mydata)
    'data.frame':   400 obs. of  4 variables:
     $ admit: int  0 1 1 1 0 1 1 0 1 0 ...
     $ gre  : int  380 660 800 640 520 760 560 400 540 700 ...
     $ gpa  : num  3.61 3.67 4 3.19 2.93 3 2.98 3.08 3.39 3.92 ...
     $ rank : int  3 3 1 4 4 2 1 2 3 2 ...
> mydata$admit <- as.factor(mydata$admit)
> mydata$rank <- as.factor(mydata$rank)
> str(mydata)
    'data.frame':   400 obs. of  4 variables:
     $ admit: Factor w/ 2 levels "0","1": 1 2 2 2 1 2 2 1 2 1 ...
     $ gre  : int  380 660 800 640 520 760 560 400 540 700 ...
     $ gpa  : num  3.61 3.67 4 3.19 2.93 3 2.98 3.08 3.39 3.92 ...
     $ rank : Factor w/ 4 levels "1","2","3","4": 3 3 1 4 4 2 1 2 3 2 ...
> xtabs(~admit+rank,data=mydata)
         rank
    admit  1  2  3  4
        0 28 97 93 55
        1 33 54 28 12
```

Step 2: Split Train and Test Data

```
> set.seed(1234)
> ind <- sample(2,nrow(mydata),replace = T,prob=c(0.8,0.2))
> trainData <- mydata[ind == 1,]
> testData <- mydata[ind == 2,]
> testData
      admit gre  gpa rank
5       0 520 2.93    4
14      0 700 3.08    2
16      0 480 3.44    3
26      1 800 3.66    1
28      1 520 3.74    4

> str(testData)
'data.frame':   75 obs. of  4 variables:
 $ admit: Factor w/ 2 levels "0","1": 1 1 1 2 2 2 2 2 1 2 ...
 $ gre  : int  520 700 480 800 520 780 500 520 600 620 ...
 $ gpa  : num  2.93 3.08 3.44 3.66 3.74 3.22 3.13 2.68 2.82 3.18 ...
 $ rank : Factor w/ 4 levels "1","2","3","4": 4 2 3 1 4 2 2 3 4 2 ...
> str(trainData)
'data.frame':   325 obs. of  4 variables:
 $ admit: Factor w/ 2 levels "0","1": 1 2 2 2 2 2 1 2 1 1 ...
 $ gre  : int  380 660 800 640 760 560 400 540 700 800 ...
 $ gpa  : num  3.61 3.67 4 3.19 3 2.98 3.08 3.39 3.92 4 ...
 $ rank : Factor w/ 4 levels "1","2","3","4": 3 3 1 4 2 1 2 3 2 4 ...
```

Step 3: Build a Classifying Model

```
> model <- glm(formula = admit ~ gre +gpa + rank,data = trainData,family = 'binomial' )
> model

Call:  glm(formula = admit ~ gre + gpa + rank, family = "binomial",
    data = trainData)

Coefficients:
(Intercept)         gre          gpa        rank2        rank3        rank4
  -5.009514    0.001631     1.166408    -0.570976    -1.125341    -1.532942
```

Degrees of Freedom: 324 Total (i.e. Null); 319 Residual

Null Deviance:	 404.4

Residual Deviance: 370 AIC: 382

```
> summary(model)
```

```
 (Intercept) -5.009514   1.316514  -3.805 0.000142 ***
gre          0.001631   0.001217   1.340 0.180180
gpa          1.166408   0.388899   2.999 0.002706 **
rank2       -0.570976   0.358273  -1.594 0.111005
rank3       -1.125341   0.383372  -2.935 0.003331 **
rank4       -1.532942   0.477377  -3.211 0.001322 **
--- Call:
>glm(formula = admit ~ gre + gpa + rank, family = "binomial",
    data = trainData)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-1.5873  -0.8679  -0.6181   1.1301   2.1178

Coefficients:
            Estimate Std. Error z value Pr(>|z|)

Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 404.39  on 324  degrees of freedom
Residual deviance: 369.99  on 319  degrees of freedom
AIC: 381.99
Number of Fisher Scoring iterations: 4
```

Step 4: Rebuilding the model with relevant features

```
> model <- glm(formula = admit ~ gpa + rank,data = trainData,family = 'binomial' )
> summary(model)
Call:
```

```
glm(formula = admit ~ gpa + rank, family = "binomial", data = trainData)

Deviance Residuals:

    Min       1Q   Median       3Q      Max

-1.5156  -0.8880  -0.6318   1.1091   2.1688

Coefficients:

            Estimate Std. Error z value Pr(>|z|)

(Intercept)  -4.7270     1.2918  -3.659 0.000253 ***

gpa           1.3735     0.3590   3.826 0.000130 ***

rank2        -0.5712     0.3564  -1.603 0.108976

rank3        -1.1645     0.3804  -3.061 0.002203 **

rank4        -1.5642     0.4756  -3.289 0.001005 **

---

Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

Null deviance: 404.39  on 324  degrees of freedom

Residual deviance: 371.81  on 320  degrees of freedom

AIC: 381.81

Number of Fisher Scoring iterations: 4
```

Step 5: Predicting the model based on Test Data

```
> pred <- predict(model,testData,type = "response")

> head(pred)

         5         14         16         26         28         29

0.09390783 0.25582628 0.23746963 0.57446309 0.23971008 0.29411490


> pred[pred>0.5]

       26        90        92       140       158       369       396

0.5744631 0.5488115 0.5677344 0.5474076 0.5474076 0.6828897 0.5488115

> p1 <- ifelse(pred>0.5,1,0)


> head(p1)

 5 14 16 26 28 29

 0  0  0  1  0  0


> table(p1)

p1
```

```
 0  1
68  7
> table(testData$admit)
 0  1
50 25
```

Step 6: Confusion Matrix

```
> tab <- table(p1,testData$admit)
> tab
p1   0  1
  0 48 20
  1  2  5
> tab <- table(Predicted =p1,Actual =testData$admit)
> tab
         Actual
Predicted  0  1
        0 48 20
        1  2  5
```

Step 7: Model Performance Parameters

Accuracy	0.71	Prevalance	0.3333
Misclass. Rate	0.29	Precision	0.7143
True Pos Rate	0.2	Null Rate Error	0.3333
False Pos Rate	0.04	F-Score	0.3125

Output: Multiclass Logistic Regression

Step 1: Data Load and Pre-Processing

```
> mydata <- iris
> str(mydata)
'data.frame':   150 obs. of  5 variables:
 $ Sepal.Length: num  5.1 4.9 4.7 4.6 5 5.4 4.6 5 4.4 4.9 ...
 $ Sepal.Width : num  3.5 3 3.2 3.1 3.6 3.9 3.4 3.4 2.9 3.1 ...
 $ Petal.Length: num  1.4 1.4 1.3 1.5 1.4 1.7 1.4 1.5 1.4 1.5 ...
 $ Petal.Width : num  0.2 0.2 0.2 0.2 0.2 0.4 0.3 0.2 0.2 0.1 ...
```

```
 $ Species     : Factor w/ 3 levels "setosa","versicolor",..: 1 1 1 1 1 1 1 1 1 1 ...
> ind <- sample(2,nrow(mydata),replace = T,prob=c(0.8,0.2))
> str(trainData)
'data.frame':   118 obs. of  5 variables:
 $ Sepal.Length: num  5.1 4.9 4.7 4.6 5 5.4 4.6 5 4.4 4.9 ...
 $ Sepal.Width : num  3.5 3 3.2 3.1 3.6 3.9 3.4 3.4 2.9 3.1 ...
 $ Petal.Length: num  1.4 1.4 1.3 1.5 1.4 1.7 1.4 1.5 1.4 1.5 ...
 $ Petal.Width : num  0.2 0.2 0.2 0.2 0.2 0.4 0.3 0.2 0.2 0.1 ...
 $ Species     : Factor w/ 3 levels "setosa","versicolor",..: 1 1 1 1 1 1 1 1 1 1 ...
> str(testData)
'data.frame':   32 obs. of  5 variables:
 $ Sepal.Length: num  4.8 4.8 5.8 5.1 4.8 5.5 4.9 5 4.5 5.1 ...
 $ Sepal.Width : num  3.4 3 4 3.7 3.4 4.2 3.6 3.5 2.3 3.8 ...
 $ Petal.Length: num  1.6 1.4 1.2 1.5 1.9 1.4 1.4 1.3 1.3 1.9 ...
 $ Petal.Width : num  0.2 0.1 0.2 0.4 0.2 0.2 0.1 0.3 0.3 0.4 ...
 $ Species     : Factor w/ 3 levels "setosa","versicolor",..: 1 1 1 1 1 1 1 1 1 1 ...
```

Step 2: Load Library and Create Model

```
> library(nnet)
>model<multinom(formula=Species~Sepal.Length+Sepal.Width+Petal.Length+Petal.Width,data=trainData)
# weights:  18 (10 variable)
initial  value 129.636250
iter  10 value 10.683012
iter  20 value 5.933903
iter  30 value 5.873500
iter  40 value 5.866866
iter  50 value 5.861992
iter  60 value 5.860395
iter  70 value 5.859634
iter  80 value 5.859340
iter  90 value 5.859208
iter 100 value 5.859118
final  value 5.859118
stopped after 100 iterations
> model
Call:
```

```
multinom(formula = Species ~ Sepal.Length + Sepal.Width + Petal.Length +
    Petal.Width, data = trainData)

Coefficients:
           (Intercept) Sepal.Length Sepal.Width Petal.Length Petal.Width
versicolor    20.95306    -2.019734   -12.17769     10.47244   -2.626553
virginica    -18.78599    -4.541125   -18.31369     19.53561   14.264642

Residual Deviance: 11.71824
AIC: 31.71824
```

```
> summary(model)
Call:
multinom(formula = Species ~ Sepal.Length + Sepal.Width + Petal.Length +
    Petal.Width, data = trainData)

Coefficients:
           (Intercept) Sepal.Length Sepal.Width Petal.Length Petal.Width
versicolor    20.95306    -2.019734   -12.17769     10.47244   -2.626553
virginica    -18.78599    -4.541125   -18.31369     19.53561   14.264642

Std. Errors:
           (Intercept) Sepal.Length Sepal.Width Petal.Length Petal.Width
versicolor    41.61007     134.8689    191.1050     76.27252    17.95107
virginica     42.22799     134.8825    191.1707     76.48059    19.08432

Residual Deviance: 11.71824
AIC: 31.71824
```

Step 3: Predict Based on Model

```
> pred <- predict(model,testData,type = "class")
> pred
 [1] setosa     setosa     setosa     setosa     setosa     setosa     setosa     setosa
 [9] setosa     setosa     setosa     versicolor versicolor versicolor versicolor versicolor
[17] versicolor virginica  virginica  virginica  virginica  virginica  virginica  virginica
[25] virginica  virginica  virginica  virginica  virginica  virginica  virginica  virginica
Levels: setosa versicolor virginica
```

Step 4: Confusion Matrix

> tab <- table(pred,testData$Species)

> tab

pred	setosa	versicolor	virginica
setosa	11	0	0
versicolor	0	6	0
virginica	0	0	15

Step 5: Model Test Parameters

Accuracy	1	Prevalance	1
Misclass. Rate	0	Precision	1
True Pos Rate	1	Null Rate Error	0.53
False Pos Rate	0	F-Score	1

References:

https://www.youtube.com/watch?v=yIYKR4sgzI8

https://www.analyticsvidhya.com/blog/2015/11/beginners-guide-on-logistic-regression-in-r/

https://www.dataschool.io/simple-guide-to-confusion-matrix-terminology/

Conclusion:

- Logistic Regression is a classification model where we predict based on independent variables whether the classification is ‘0’ or ‘1’(in binary)
- We have learnt about multi-variate(multiple independent variables) and single-variable regression and also multi-class logistic regression(multiple dependent classes)
- We have learnt about sigmoid function and realized how it helps for the hypothesis for classification of dependent variable
- We have learnt about various parameters which tell us about the accuracy of our predicted model using confusion matrix
- We have learnt how to build a model for our data,train it and then use it for prediction purposes.