Logistic Regression implementation in R

In this lecture

- Case study
  - Problem statement
- Solve the case study using R
  - Read the data from a ".csv" file
  - Understand the data
  - glm() function
  - Interpret the results

Key points from previous lecture

- Logistic Regression is primarily used as a classification algorithm
- It is supervised learning algorithm
  - Data is labelled
- Parametric approach
- Decision boundary derived based on probability interpretation
- Decision boundary can be linear/ non-linear
- Probabilities are modelled as sigmoidal function

Automotive Crash Testing

Automotive Crash Testing- Problem Statement

- A crash test is a form of destructive testing that is performed in order to ensure high safety standards for various cars

Data Science for Engineers

Automotive Crash Testing

Hatchback

SUV

Logistic Regression in R

Automotive Crash Testing- Problem Statement

- Several cars have rolled into an independent audit unit for crash test
- They are being evaluated on a defined scale {poor (-10) to excellent(10)} on:

1) Manikin head impact
2) Manikin body impact
3) Interior impact
4) HVAC impact
5) Safety alarm system

Data Science for Engineers

Automotive Crash Testing

- Each crash test is very expensive
- The crash test was performed for only 100 cars
- Type of car- Hatchback/SUV, was noted
- However with this data in future they should be able to predict the type of the car
- Part of data reserved for building a model and remaining kept for analysis

Logistic Regression in R

Automotive Crash Testing

- Data for 80 cars is given in crashTest_1.csv
- Data for remaining 20 cars is given in crashTest_1_TEST.csv
- Use logistic regression classification technique to classify the car types as Hatchback/SUV

Data Science for Engineers

Solution to case study using R

Logistic Regression in R

Getting things ready

- Setting working directory, clearing variables in the workspace
- Installing or loading required packages

```
# Set the working directory as the directory which
#contains the data files
# setwd("Path of the directory with data files")
rm(list=ls()) # to clear the environment
# install.packages("caret",dependencies = TRUE)

library(caret) # for confusionMatrix
```

Reading the data

- Data for this case study is provided to you in files with names
  - crashTest_1- training data
  - crashTest_1_TEST- testing data

- To read the data from a ".csv" file we use `read.csv()` function

read.csv()

Reads a file in table format and creates a data frame from it

SYNTAX

```
read.csv(file,row.names=1)
```

file	the name of the file which the data are to be read from. Each row of the table appears as one line of the file.
row.names	a vector of row names. This can be a vector giving the actual row names, or a single number giving the column of the table which contains the row names, or character string giving the name of the table column containing the row names.

Reading the data

```
#Reading the data
crashTest_1<-read.csv("crashTest_1.csv",row.names=1)
crashTest_1_TEST<-read.csv("crashTest_1_TEST.csv",row.names=1)
```

Environment History Connections

Import Dataset List

Global Environment

Data

crashTes…	80 obs. of 6 variables
crashTes…	20 obs. of 6 variables

Viewing the data

```
View(crashTest_1)
```

knn_implementation.R | Classification+Clustering R session cod... | crashTest_1 | crashTest_1_TEST

Filter

	ManHI	ManBI	IntI	HVACi	Safety	CarType
1	-5.27	-1.30	2.86	-4.85	4.04	SUV
2	-4.82	-5.38	9.72	-0.97	-4.57	Hatchback

Understanding the data

- crashTest_1 contains 80 observations of 6 variables
- crashTest_1_TEST contains 20 observations of 6 variables
- The variables are: Manikin head impact, Manikin body impact, Interior impact, HVAC impact, Safety alarm system
  - First five columns are the details about the car and last column is the label which says whether the cartype Hatchback/ SUV

Structure of the data

- Structure of data
  - Variables and their data types
- str()

SYNTAX

```
str(object)
```

object	any R object about which you want to have some information.

Structure of train data

```
> str(crashTest_1)
'data.frame':	80 obs. of  6 variables:
 $ ManHI  : num  -5.27 -4.82 9.57 2.84 0 0.4 5.94 5.78 0.86 7.36 ...
 $ ManBI  : num  -1.3 -5.38 -7.5 -2.85 2.68 6.34 3.14 -1.75 -4.32 7.42 ...
 $ IntI   : num  2.86 9.72 -7.61 0.92 -4.15 0.83 -6.65 -6.85 8.1 0.27 ...
 $ HVACi  : num  -4.85 -0.97 1.33 5.51 0.85 5.03 6.62 0.73 -8.96 -8.62 ...
 $ Safety : num  4.04 -4.57 -5.1 -6.64 5.58 -8.1 -1.32 5.5 3.1 3.08 ...
 $ CarType: Factor w/ 2 levels "Hatchback","SUV": 2 1 1 1 2 2 1 1 1 2 ...
```

Summary of the data

- Summary of data
  - The function invokes particular methods which depend on the class of the first argument.
- `summary()`

Summary gives a 5 point summary for numeric attributes in the data

SYNTAX

```
summary(object)
```

Summary of crashTest_1

```
> summary(crashTest_1)
     ManHI             ManBI        
 Min.   :-9.9300   Min.   :-9.9400  
 1st Qu.:-5.1950   1st Qu.:-5.7050  
 Median : 0.6350   Median :-1.8150  
 Mean   :-0.0935   Mean   :-0.9277  
 3rd Qu.: 5.0500   3rd Qu.: 3.4175  
 Max.   : 9.5700   Max.   : 9.6100  
      IntI             HVACi        
 Min.   :-9.9900   Min.   :-9.8200  
 1st Qu.:-5.5725   1st Qu.:-5.6750  
 Median :-0.4150   Median : 0.8700  
 Mean   :-0.1349   Mean   : 0.1197  
 3rd Qu.: 4.9775   3rd Qu.: 5.0625  
 Max.   : 9.7200   Max.   : 9.8900  
     Safety             CarType  
 Min.   :-9.8000   Hatchback:50  
 1st Qu.:-4.6775   SUV      :30  
 Median : 0.8300                 
 Mean   : 0.5437                 
 3rd Qu.: 4.6225                 
 Max.   : 9.9900                 
```

glm()

glm(formula, data, family)

Arguments

formula	object of class "formula" (or one that can be coerced to that class): a symbolic description of the model to be fitted
data	dataframe containing variables
family	a description of the error distribution and link function to be used in the model. For glm this can be a character string naming a family function, a family function or the result of a call to a family function In specific, family='binomial' corresponds to logistic regression

Data Science for Engineers

Building a logistic regression model

```
# Model
logisfit<-glm(formula = crashTest_1$CarType~., family = 'binomial',
              data = crashTest_1)
```

$$p(X) = \frac{e^{(\beta_0+\beta_1 X)}}{1 + e^{(\beta_0+\beta_1 X)}}$$

$$\log\left(\frac{p(X)}{1-p(X)}\right) = \beta_0 + \beta_1 X$$

```
> logisfit

Call:  glm(formula = crashTest_1$CarType ~ ., family = "binomial", data = crashTest_1)

Coefficients:
(Intercept)        ManHI        ManBI         IntI        HVACi       Safety
     -22.76       -13.48        36.02       -44.90       -58.50       -27.36

Degrees of Freedom: 79 Total (i.e. Null);  74 Residual
Null Deviance:      105.9
Residual Deviance: 5.359e-08    AIC: 12
```

Summary of model

```
> summary(logisfit)

Call:
glm(formula = crashTest_1$CarType ~ ., family = "binomial", data = crashTest_1)

Deviance Residuals:
       Min          1Q      Median          3Q         Max
-1.316e-04  -2.100e-08  -2.100e-08   2.100e-08   1.266e-04

Coefficients:
            Estimate Std. Error z value Pr(>|z|)
(Intercept)   -22.76   12007.54  -0.002    0.998
ManHI         -13.48    3077.29  -0.004    0.997
ManBI          36.02    7221.18   0.005    0.996
IntI          -44.90    8853.08  -0.005    0.996
HVACi         -58.50   11461.92  -0.005    0.996
Safety        -27.36    5396.42  -0.005    0.996

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 1.0585e+02  on 79  degrees of freedom
Residual deviance: 5.3590e-08  on 74  degrees of freedom
AIC: 12

Number of Fisher Scoring iterations: 25
```

$\widehat{\beta_i}$

$i = 0, 1, \ldots, 5$

Finding the odds

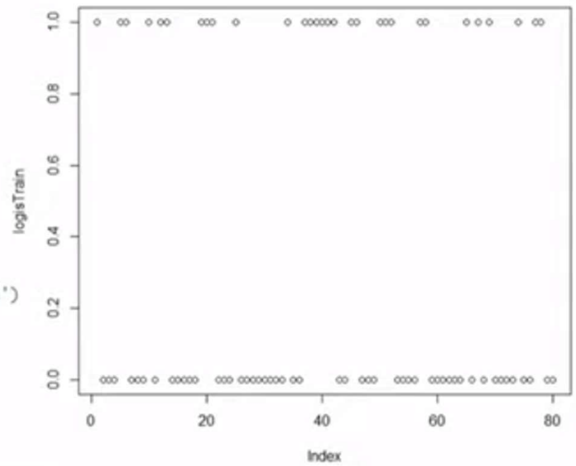
- predict()
- Synxtax: predict(object)

```
# Finding the odds
logisTrain<-predict(logisfit, type = 'response')
```

- predict() with type='response' gives probabilities
- By default otherwise it returns log(odds)

Plotting the probabilities

```
plot(logisTrain)
```




```
# Finding the odds
logisTrain<-predict(logisfit, type = 'response')
```


Identifying probabilities associated with the CarType

- Mean of probabilities
- This helps us identify the probabilities associated with the two classes

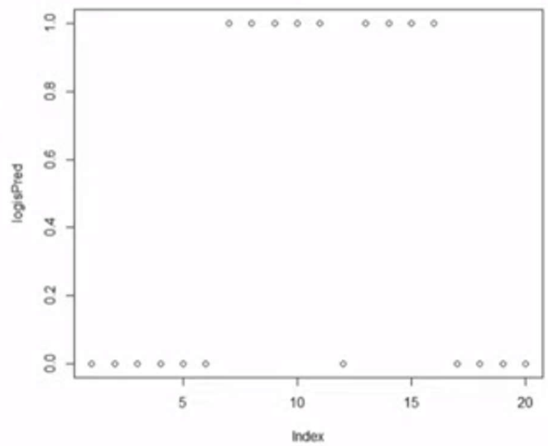
```
> tapply(logisTrain,crashTest_1$CarType,mean)
   Hatchback          SUV
2.851316e-10 1.000000e+00
```

Predicting on test data

```
# Predicting on test set
logisPred<-predict(logisfit,
                   newdata = crashTest_1_TEST,
                   type='response')
plot(logisPred)
```

- "logisPred" is the output which has the probability values

```
> logisPred
          81           82           83           84
4.269083e-13 1.740691e-04 2.220446e-16 2.220446e-16
```



Results

- We classify whether the test point is Hatchback/ SUV by setting a threshold

```
crashTest_1_TEST[logisPred<=0.5,"LogisPred"]<-"Hatchback"
crashTest_1_TEST[logisPred>0.5,"LogisPred"]<-"SUV"
```

crashTest_1_TEST × | Classification+Clustering R session cod... * × | lda_logistic_final.R ×

Filter

	ManHI	ManBI	IntI	HVACi	Safety	CarType	LogisPred
81	1.94	2.21	3.38	1.78	-7.19	Hatchback	Hatchback
82	-0.02	-3.33	0.79	-6.63	7.99	SUV	Hatchback

Confusion matrix

```
confusionMatrix(table(crashTest_1_TEST[,7],
                     crashTest_1_TEST[,6]),positive = 'Hatchback')
```

```
Confusion Matrix and Statistics

              Reference
Prediction  Hatchback SUV
  Hatchback        10   1
  SUV               0   9

               Accuracy : 0.95
                 95% CI : (0.7513, 0.9987)
    No Information Rate : 0.5
    P-Value [Acc > NIR] : 2.003e-05

                  Kappa : 0.9
 Mcnemar's Test P-Value : 1
```


Confusion matrix

```
confusionMatrix(table(crashTest_1_TEST[,7],
                     crashTest_1_TEST[,6]),positive = 'Hatchback')
```

```
            Sensitivity : 1.0000
            Specificity : 0.9000
         Pos Pred Value : 0.9091
         Neg Pred Value : 1.0000
             Prevalence : 0.5000
         Detection Rate : 0.5000
   Detection Prevalence : 0.5500
      Balanced Accuracy : 0.9500

       'Positive' Class : Hatchback
```