Linear Regression Test & Train

02/17/2023

Introduction to linear Regression

Definition:

Linear Regression is a statistical model seeks to find a relationship between a dependent variable and one or more independent variables. It models this relationship in the form y = mx + b.

Linear regression is a predictive model. The accuracy of the model is determined by using a function generated to predict the dependent value based on some input and comparing the output to its actual values

Strengths of Linear Regression:

Linear regressions is simple to understand and easy to implement

Linear regressions is efficient and can handle large data sets easiy

Linear regressions is great and identifying predictors in a data set and quantiying the degree of the relationship between differernt variables

Weaknesses of Linear Regression:

Fails to work with data sets that have non linear relationships

Linear regressions is sensitive to outliers and hte presence of one can greatly affect the accuracy results

Linear regressions cannot handle qualitative variables

Goal:

In this notebook, I intend to demonstrate the life cycle of a machine learning project using linear regression and analyze how good of a model it is.

Machine Learning Life Cycle

Data Set:

For this project, I'm using a data set that measure the Risk Factors for Cardiovascular Heart Disease

```r
heart.data <- read.csv("heart_data.csv")
str(heart.data)
```

```
## 'data.frame':    70000 obs. of  14 variables:
##  $ index      : int  0 1 2 3 4 5 6 7 8 9 ...
##  $ id         : int  0 1 2 3 4 8 9 12 13 14 ...
##  $ age        : int  18393 20228 18857 17623 17474 21914 22113 22584 17668 19834 ...
##  $ gender     : int  2 1 1 2 1 1 1 2 1 1 ...
##  $ height     : int  168 156 165 169 156 151 157 178 158 164 ...
##  $ weight     : num  62 85 64 82 56 67 93 95 71 68 ...
##  $ ap_hi      : int  110 140 130 150 100 120 130 130 110 110 ...
##  $ ap_lo      : int  80 90 70 100 60 80 80 90 70 60 ...
##  $ cholesterol: int  1 3 3 1 1 2 3 3 1 1 ...
##  $ gluc       : int  1 1 1 1 1 2 1 3 1 1 ...
```

```
##  $ smoke      : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ alco       : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ active     : int  1 1 0 1 0 0 1 1 1 0 ...
##  $ cardio     : int  0 1 1 1 0 0 0 1 0 0 ...
```

The data comprises factors such as age, gender, height, smoking habits, etc. and determines how these factors affects one's likelihood of developing heart complications

Step1: Data Cleaning

Before peforming any analysis or computation on data, it must be cleaned. It involves

Properly naming columns and rows In the data set,the age, gender, ap_hi, and gluc attribute are ambigous so it's necessary that we rename them to improve the readability of our data set

```
colnames(heart.data)[1] = "index"
colnames(heart.data)[3] = "age(In days)"
colnames(heart.data)[4] = "gender(1:M, 2:F)"
colnames(heart.data)[5] = "height(cm)"
colnames(heart.data)[6] = "weight(kg)"
str(heart.data)
```

```
## 'data.frame':    70000 obs. of  14 variables:
##  $ index           : int  0 1 2 3 4 5 6 7 8 9 ...
##  $ id              : int  0 1 2 3 4 8 9 12 13 14 ...
##  $ age(In days)    : int  18393 20228 18857 17623 17474 21914 22113 22584 17668 19834 ...
##  $ gender(1:M, 2:F): int  2 1 1 2 1 1 1 2 1 1 ...
##  $ height(cm)      : int  168 156 165 169 156 151 157 178 158 164 ...
##  $ weight(kg)      : num  62 85 64 82 56 67 93 95 71 68 ...
##  $ ap_hi           : int  110 140 130 150 100 120 130 130 110 110 ...
##  $ ap_lo           : int  80 90 70 100 60 80 80 90 70 60 ...
##  $ cholesterol     : int  1 3 3 1 1 2 3 3 1 1 ...
##  $ gluc            : int  1 1 1 1 1 2 1 3 1 1 ...
##  $ smoke           : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ alco            : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ active          : int  1 1 0 1 0 0 1 1 1 0 ...
##  $ cardio          : int  0 1 1 1 0 0 0 1 0 0 ...
```

Removing duplicates from a data set

```
print (sum(duplicated(heart.data)=="TRUE"))
```

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

Addressing NA's (unavailable data) if required. In this case there are none

```
print (sum(is.na(heart.data)))
```

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

Step 2: Creating Linear Models Test Data

A) Creating Test Data
To ensure that the model is trained accurately, it is necessary to randomize the order of entries in the data frame

```
# ensures that the same random numbers are generated every time
set.seed(123)

# Creates a set of random values
random.values <- runif(nrow(heart.data))

# Created new data with reoredered elements

reordered.heart.data <- (heart.data[order(random.values),  ])
head(reordered.heart.data)
```

```
##       index    id age(In days) gender(1:M, 2:F) height(cm) weight(kg) ap_hi
## 19296 19295 27564        22659                2        168         77   160
## 15829 15828 22605        21264                1        175         68   120
## 3934   3933  5560        16826                1        153         73   122
## 43484 43483 62123        19134                2        173         56   100
## 9036   9035 12886        21005                1        170         65   120
## 39632 39631 56628        18266                2        165         60   120
##       ap_lo cholesterol gluc smoke alco active cardio
## 19296  1000           1    1     0    0      1      1
## 15829    80           1    1     0    0      1      1
## 3934     85           2    1     0    0      1      0
## 43484    70           1    1     0    0      0      0
## 9036     80           1    1     0    0      1      0
## 39632    80           1    1     0    0      1      0
```

Next, we assign 80% of the data to train set and the rest to the test set

```
train <- reordered.heart.data[1:56000, ]
test <- reordered.heart.data[56001: 70000, ]
paste("Training set Size: ", nrow(train))
```

```
## [1] "Training set Size:  56000"
```

```
paste("Test set Size:     ", nrow(test))
```

```
## [1] "Test set Size:      14000"
```

B) Exploring the data
The Average age of healthy and sick participants:

```
paste("Average age of healty participants: ", mean(train$age[train$cardio==0])/365)
```

```
## [1] "Average age of healty participants:  51.7232344479725"
```

```
paste("Average age of Sick participants: ", mean(train$age[train$cardio==1])/365)
```

```
## [1] "Average age of Sick participants:  54.9360069400782"
```

The Gender of the participants:

```
paste("Number of Males: ", sum(train$gender==1))
```

```
## [1] "Number of Males:  36324"
```

```
paste("Number of Females: ", sum(train$gender==2))
```

```
## [1] "Number of Females:  19676"
```

The number of people with heart complications in the set

```
paste("Number of people at risk/have heart complications: ",sum(train$cardio==TRUE))
```

```
## [1] "Number of people at risk/have heart complications:  27997"
```

Number of people who smoke/drink/both and have heart complication

Step 3: Models

A) Initial Logistic regression Model (Rate of Activity vs. Chance of Cardiac Issues)

This logistic regression models tracks the liklyhood of inactivity with the chance to get cardiac complications

As seen in the information below, surprisingly, while an increase in activity levels does decrease one's chances of developing heart issues, accoring to the estimate activity levels aren't a primary factor.

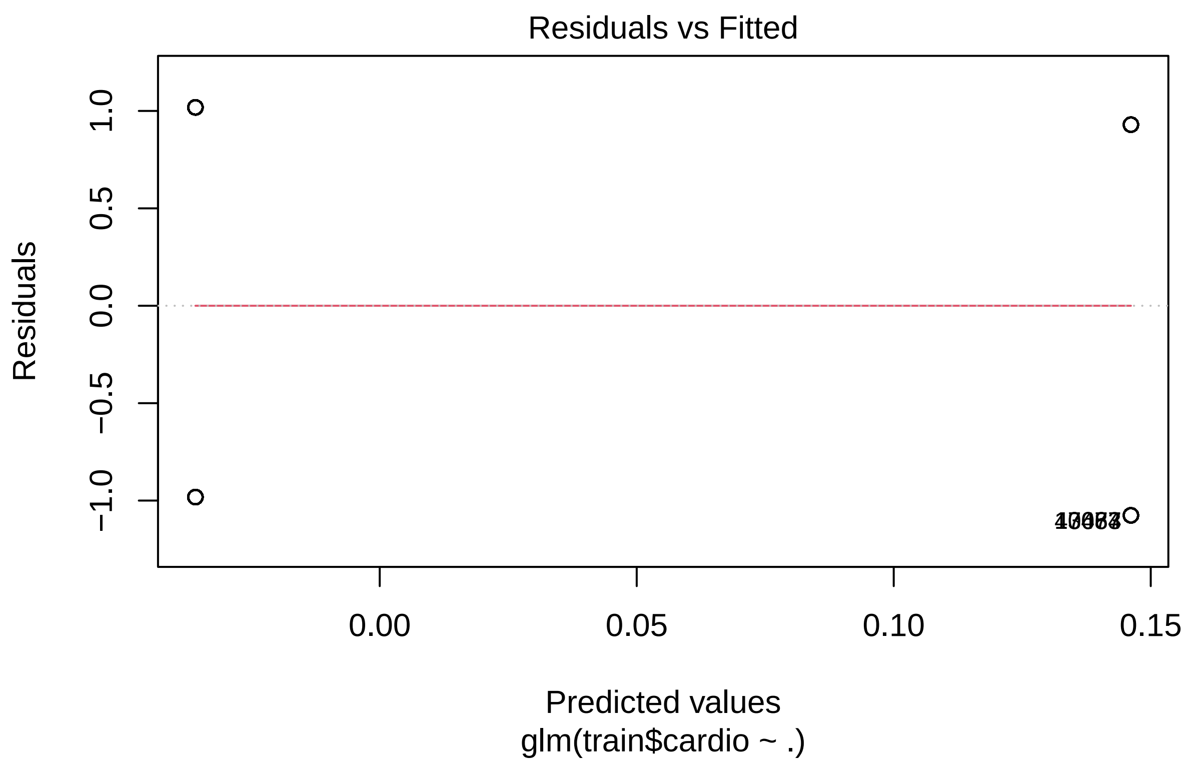
Aside from the low correlation, the standear error indicates that this model is good and varies minutely from actual values.

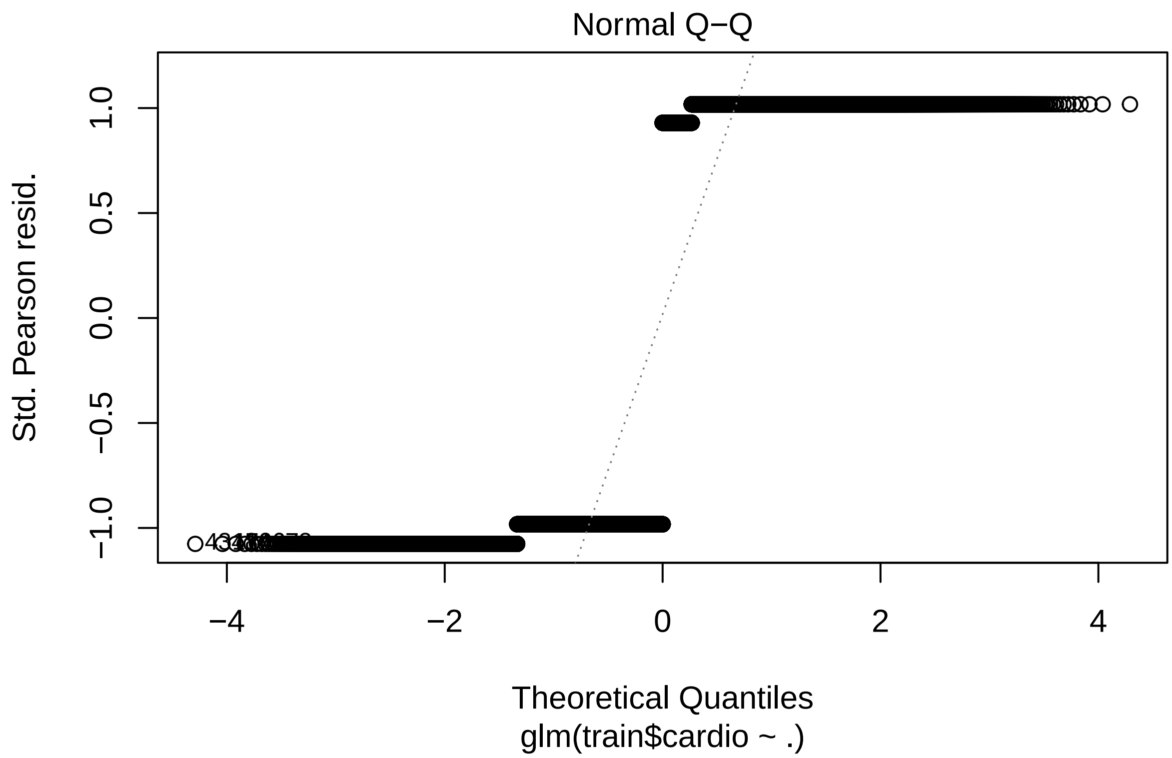
```
logistic.regression.model <- glm(train$cardio ~., data = train["active"], family = "binomial")
summary(logistic.regression.model)
```

```
##
## Call:
## glm(formula = train$cardio ~ ., family = "binomial", data = train["active"])
##
## Deviance Residuals:
##    Min      1Q  Median      3Q     Max
## -1.240  -1.162  -1.162   1.193   1.193
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept)  0.14615    0.01914   7.637 2.22e-14 ***
## active      -0.18200    0.02133  -8.532  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

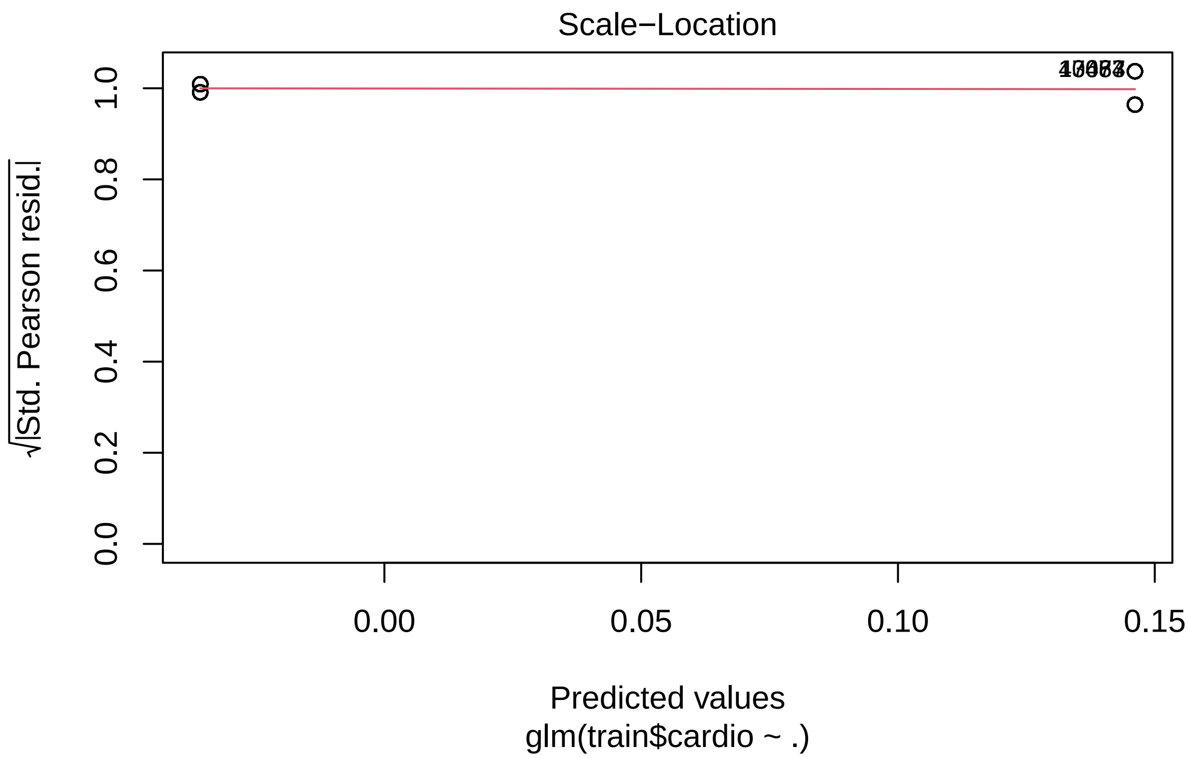
```
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 77632  on 55999  degrees of freedom
## Residual deviance: 77560  on 55998  degrees of freedom
## AIC: 77564
##
## Number of Fisher Scoring iterations: 3
```

```
plot(logistic.regression.model)
```

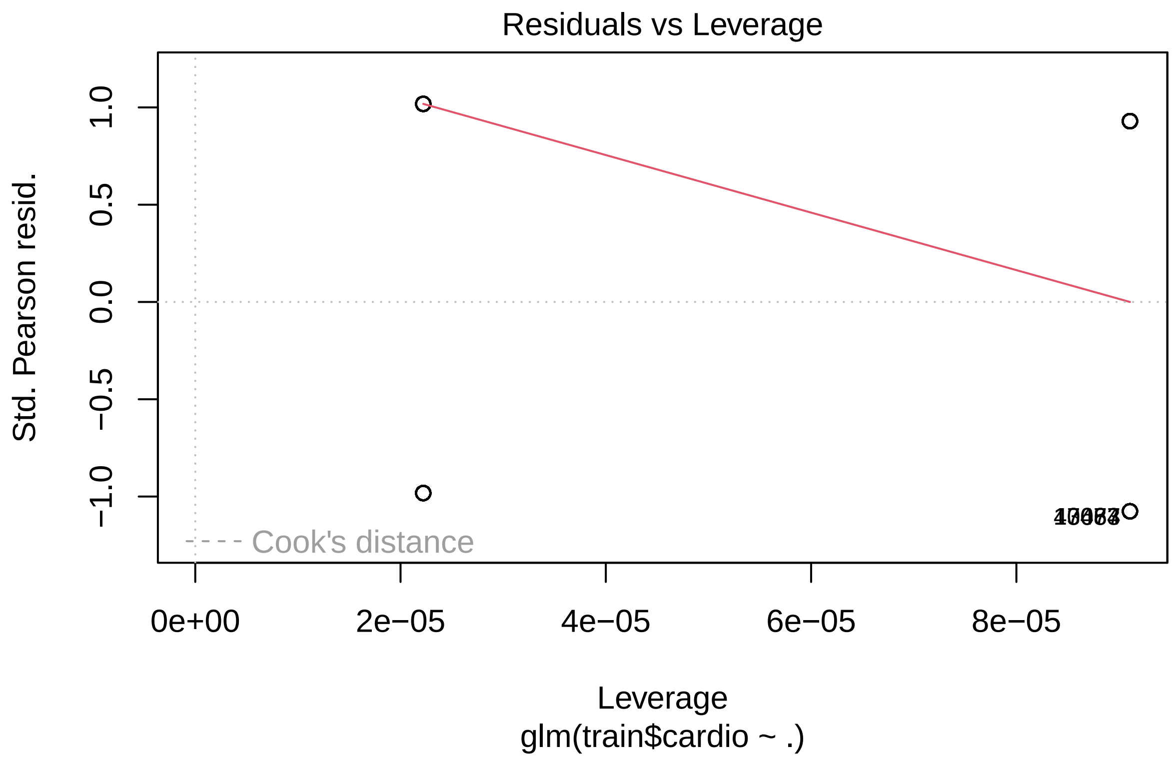




Normal Q−Q
Std. Pearson resid.
1.0
0.5
0.0
−0.5
−1.0
−4
−2
0
2
4
Theoretical Quantiles
glm(train$cardio ~ .)



Scale−Location
1.0
0.8
0.6
0.4
0.2
0.0
√|Std. Pearson resid.|
0.00
0.05
0.10
0.15
Predicted values
glm(train$cardio ~ .)



B) Additional Models

Alcohol and Smoking vs Chance of heart condition

```
logistic.regression.model1 <- glm(train$cardio ~., data = train[11:12], family = "binomial")
summary(logistic.regression.model1)
```

```
##
## Call:
## glm(formula = train$cardio ~ ., family = "binomial", data = train[11:12])
##
## Deviance Residuals:
##    Min      1Q  Median      3Q     Max
## -1.181  -1.181  -1.138   1.174   1.217
##
## Coefficients:
##              Estimate Std. Error z value Pr(>|z|)
## (Intercept)  0.008206   0.008929   0.919  0.35807
## smoke       -0.088387   0.031803  -2.779  0.00545 **
## alco        -0.012515   0.039796  -0.314  0.75316
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
```

```
##     Null deviance: 77632  on 55999  degrees of freedom
## Residual deviance: 77623  on 55997  degrees of freedom
## AIC: 77629
##
## Number of Fisher Scoring iterations: 3
```

Blood Pressure and Age vs Chance of heart condition

```r
logistic.regression.model2 <- glm(train$cardio ~., data = train[c(3,7,8)], family = "binomial")
```

```
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
```

```r
summary(logistic.regression.model2)
```

```
##
## Call:
## glm(formula = train$cardio ~ ., family = "binomial", data = train[c(3,
##     7, 8)])
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -8.4904  -1.0001  -0.0045   1.0239   5.1670
##
## Coefficients:
##                   Estimate Std. Error z value Pr(>|z|)
## (Intercept)     -9.361e+00  1.099e-01 -85.200  < 2e-16 ***
## `age(In days)`  1.590e-04  3.907e-06  40.710  < 2e-16 ***
## ap_hi            4.942e-02  6.818e-04  72.485  < 2e-16 ***
## ap_lo            2.301e-04  7.108e-05   3.237  0.00121 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 77632  on 55999  degrees of freedom
## Residual deviance: 65922  on 55996  degrees of freedom
## AIC: 65930
##
## Number of Fisher Scoring iterations: 7
```

Height and Weight vs Chance of heart condition

```r
logistic.regression.model3 <- glm(train$cardio ~., data = train[c(5,6)], family = "binomial")
summary(logistic.regression.model3)
```

```
##
## Call:
## glm(formula = train$cardio ~ ., family = "binomial", data = train[c(5,
##     6)])
##
## Deviance Residuals:
```

```
##     Min       1Q   Median       3Q      Max
## -2.8383  -1.1156  -0.7639   1.1731   2.1225
##
## Coefficients:
##                Estimate Std. Error z value Pr(>|z|)
## (Intercept)   0.8512853  0.1759501   4.838 1.31e-06 ***
## `height(cm)` -0.0186639  0.0011277 -16.551  < 2e-16 ***
## `weight(kg)`  0.0299398  0.0006759  44.297  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 77632  on 55999  degrees of freedom
## Residual deviance: 75480  on 55997  degrees of freedom
## AIC: 75486
##
## Number of Fisher Scoring iterations: 4
```

Step 4: Summary

In summary, the height and weight regression function was the most accurate predictor of heart conditions. This makes sense because one's BMI is a good general estimate of fitness level. Additionally, a higher BMI typically comes with an increased risk to experience cardiac problems in the future