*i*Journals: International Journal of Software & Hardware Research in Engineering
ISSN-2347-4890
Volume 7 Issue 4 April 2019

Machine Learning: Logistic Regression - a Review

Author: Satyanarayana Medicherla

Consultant, Canon U.S.A. Inc, 1, Canon Park, New York - 11747, email- satyams@hotmail.com

Abstract

Logistic Regression is a technique used to establish the relationship between a binary outcome or dependent variable and a set of independent input variables [4]. Linear Regression is used when the dependent or response variable is continuous. We will use Logistic Regression method when the response variable is binary like 0 or 1. A good example where we can use Logistic Regression is to classify if an email is spam or non-spam. In contrast to the Linear Regression, the training labels are 0/1 or some kind of categorical values like M/F. In this article, we will discuss the need for Logistic Regression Model. We will briefly discuss Maximum Likelihood Estimation and study different types of decision boundaries in the context of Logistic Regression.

Introduction

In Linear Regression, we will choose a line that fits best for a set of training data points upon which we train our Model. We will find the best fitting line by minimizing the residuals, which are differences between the estimated and observed values of the response variable. In case of Logistic Regression, we will find a best fitting line that separates +ve and -ve examples in our data set. Maximum Likelihood Estimation is used to find the line that separates the +ve and -ve examples in the data set. The line that separates the +ve and -ve examples is called the decision boundary.

Let us illustrate the Maximum Likelihood Estimation with a simple example.

Maximum Likelihood Estimation Illustration

Maximum Likelihood Estimation is method used to determine the parameter values of a distribution. We find the parameter values such that they maximize the likelihood as if that distribution produced the data that we observed. These parameters are the mean and standard deviation of the distribution. The following expression gives the Probability Density Function (PDF) for the Normal/Gaussian Distribution [7]

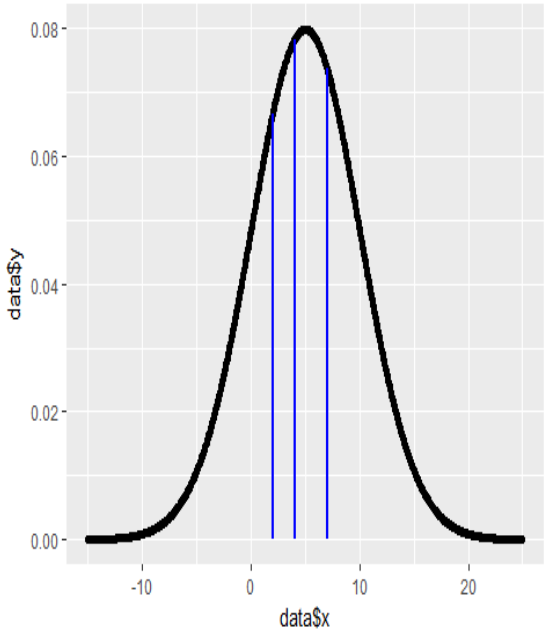
$$P(x) = \frac{1}{\sqrt{(2\pi\sigma^2)}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$

Where μ is the mean of the distribution and σ is the standard deviation of the distribution. We know that when we integrate the PDF from -Inf to +Inf P(x)dx = 1, which is the area under the probability distribution curve, since the total probability is always 1.

We will use a distribution with a mean of 5 and standard deviation of 5 and find the overall likelihood of observing our data points. The following R-code generates a plot where the likelihoods for each of our three data points (x=2, x=4 and x=7) are illustrated graphically.

```
library(ggplot2)
x <-seq(-15, 25, length=1000)
y <-dnorm(x, mean=5, sd=5)

data =data.frame(x,y)

mu =5
sigma =5

x1=2
yend1 =(1.0/sqrt(2*pi*sigma*sigma)) *exp((-(x1 -
mu)^2)/(2*sigma*sigma))

x2=4
yend2 =(1.0/sqrt(2*pi*sigma*sigma)) *exp((-(x2 -
mu)^2)/(2*sigma*sigma))

x3=7
yend3 =(1.0/sqrt(2*pi*sigma*sigma)) *exp((-(x3 -
```

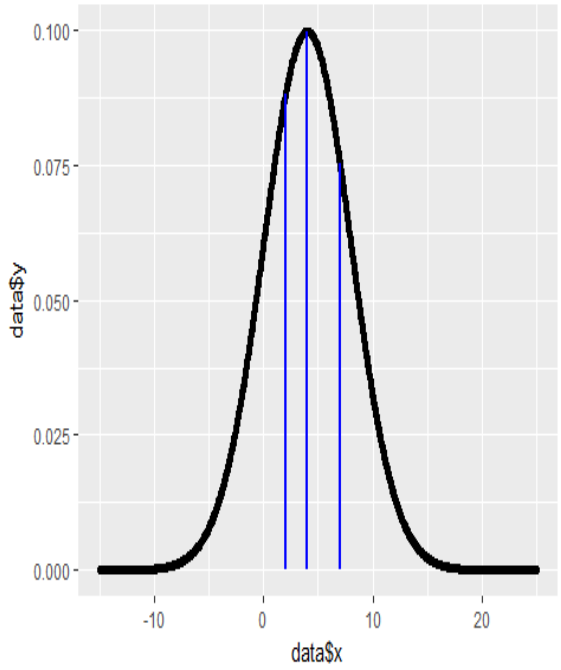
```
mu)^2)/(2*sigma*sigma))

p <-ggplot(data)
+geom_point(aes(x=data$x,y=data$y)) +
geom_segment
(mapping=aes(x=x1,y=0,xend=x1,yend=yend1),
size=1, linetype=1, color="blue")+
geom_segment
(mapping=aes(x=x2,y=0,xend=x2,yend=yend2),
size=1, linetype=1, color="blue")+
geom_segment
(mapping=aes(x=x3,y=0,xend=x3,yend=yend3),
size=1, linetype=1, color="blue")
p
```




```
#Now the overall likelihood
 tl1 =yend1*yend2*yend3
```

Now let us take a different distribution with a mean of 4 and standard deviation of 4 and find the overall likelihood of observing our three data points.

```
x <-seq(-15, 25, length=1000)
y <-dnorm(x, mean=4, sd=4)

data =data.frame(x,y)

mu =4
sigma =4

x1=2
yend1 =(1.0/sqrt(2*pi*sigma*sigma)) *exp((-(x1 -
mu)^2)/(2*sigma*sigma))

x2=4
yend2 =(1.0/sqrt(2*pi*sigma*sigma)) *exp((-(x2 -
mu)^2)/(2*sigma*sigma))

x3=7
yend3 =(1.0/sqrt(2*pi*sigma*sigma)) *exp((-(x3 -
mu)^2)/(2*sigma*sigma))

p <-ggplot(data)
+geom_point(aes(x=data$x,y=data$y)) +
geom_segment
(mapping=aes(x=x1,y=0,xend=x1,yend=yend1),
size=1, linetype=1, color="blue")+
geom_segment
(mapping=aes(x=x2,y=0,xend=x2,yend=yend2),
size=1, linetype=1, color="blue")+
geom_segment
(mapping=aes(x=x3,y=0,xend=x3,yend=yend3),
size=1, linetype=1, color="blue")
p
```




```
#Now the overall likelihood

tl2 =yend1*yend2*yend3

tl1

## [1] 0.0003838997

tl2

## [1] 0.0006608731
```

We can clearly see that the overall likelihood of

observing all the three data points is higher for the distribution with mean 4 and standard deviation 4 than for the distribution with mean 5 and standard deviation 5. Continuing this process, we will choose the distribution with highest likelihood. Instead of trying all infinite number of different possible combinations, we differentiate the probability density function with respect to the mean and equate it to zero to find the mean of the distribution with highest likelihood. Similarly, by differentiating the PDF with respect to the standard deviation and equating it to zero, we will find the standard deviation of the distribution with highest likelihood [8]. Noting down the fact that any function and its log are monotonically increasing and hence they have the maximum at the same the same point, we will use the log of the likelihoods. The log will convert the product of n terms to sum of n terms, which would highly simplify the differentiation.

Maximum Likelihood Estimation for Logistic Regression

While working with linear regression, we start with some initial theta vector and adjust the theta parameter vector so that the sum of squared residuals are minimized to find the theta parameter vector corresponding to the best fitting line.

The hypothesis is linear in case of linear regression as follows

$Y = \Theta^T X$, where Y is a vector (n - number of training examples) of training labels and X is an m x n feature vector of the training set.

In case of logistic regression, the hypothesis is non-linear as follows [1]

$$Y = \frac{1}{(1 + e^{-\theta^T X})}$$

Since the above hypothesis is non-linear, we will not be able to fit a straight line to this data. Now let us see how we can find an optimal Theta vector for logistic regression.

As we know, the above hypothesis maps the X vector to either 0 or 1, indicating that each training example is binary classified.

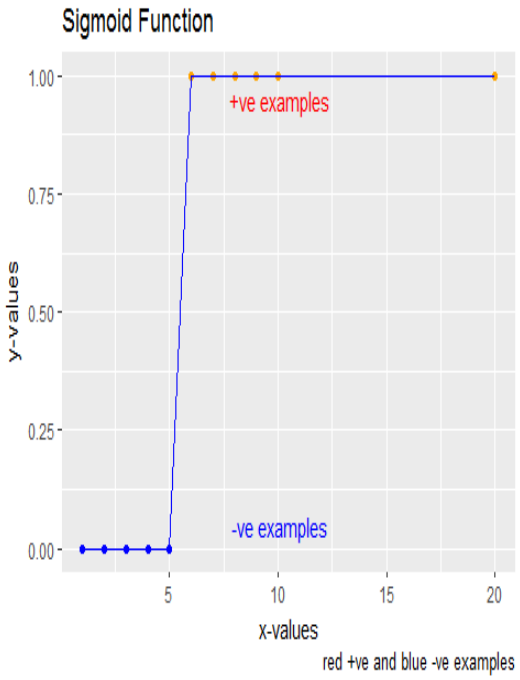
Since it is the probability, we will write it as

$$p = \frac{1}{(1 + e^{-\theta^T X})} - - - -> equation - 1$$

This is also known as the Sigmoid function or S-curve.

It looks as follows

```
library(ggplot2)
x =c(1,2,3,4,5,6,7,8,9,10,20)
y =c(0,0,0,0,0,1,1,1,1,1,1)
data =data.frame(x,y)

p=ggplot(data)
+geom_point(aes(x=x,y=y),col=ifelse(y>0.5,"orange",
"blue")) +
geom_path(aes(x=x,y=y),col=I("blue"))+
annotate("text", x =10, y =0.95, label =" +ve
examples",col=I("red")) +
annotate("text", x =10, y =0.05, label =" -ve
examples",col=I("blue")) +
labs(title ="Sigmoid Function",x="x-values" ,y="y-
values",
caption ="red +ve and blue -ve examples")

p
```



When we predict the probability of a new example, we use some threshold of eg. 0.5. If the predicted probability is anything higher 0.5, is treated as a positive example and otherwise it is treated as a negative example.

Let us see how this function is optimized to fit our training set.

If the probability is p, then the odds are going to be

$$\frac{p}{1-p}$$

If we substitute the value of p from equation-1 and simplify the resulting expression, we get

$$\frac{p}{1-p} = \frac{\frac{1}{(1+e^{-\theta^T X})}}{1-\frac{1}{(1+e^{-\theta^T X})}}$$

$$\frac{p}{1-p} = \frac{\frac{1}{(1+e^{-\theta^T X})}}{\frac{e^{-\theta^T X}}{(1+e^{-\theta^T X})}}$$

$$\frac{p}{1-p} = e^{\theta^T X}$$

Let us take Natural log on both sides

$$log(\frac{p}{1-p}) = \theta^T X$$

The log of odds is a linear function of X. The left hand side of the equation is called the logit.

We can now use the Maximum Likelihood Estimation to find out the theta vector that maximizes the likelihood.

So we will map each point on the sigmoid to log odds [5]

- When p = 0.5, the log odds is 0
- when p = 1, the log odds is +Inf
- when p = 0, the log odds is -Inf

When we plot the x and log odds, the above transformation would push all the labels from 0 and 1 to -Inf and +Inf respectively. If we fit some line to this data, the residuals from these new labels are going to be Inf. Then we will project these labels to the line by drawing a vertical line to the candidate line that we fit. The length of the segments between x-axis to our line gives us the log odds value for each of our training labels. We calculate their corresponding probabilities using the sigmoid function and represent these on the sigmoid or S-curve. Then we calculate the likelihood values for each of these. The likelihood for +ve examples is the corresponding sigmoid value, which is the probability p. For -ve examples, it is 1 - p.

We will adjust the theta vector of our line and calculate the product of all the likelihoods. We repeat this process until we find a line that produces the maximum product of the likelihoods. The algorithm knows which way the theta vector has to be adjusted in order to achieve the convergence. The decision made is based on the derivative of the total log likelihood with respect to any theta at any certain point. It is common practice to use the product of log of the likelihoods instead of product of the likelihoods, because any function and its log are monotonic and log will simplify the product to a sum.

The way we interpret or use this theta vector is slightly different from that of the Linear Regression. We will treat this theta vector as the set of thetas that define best fitting line for the training set in linear regression. In case of Logistic Regression, this theta vector would define the line that separates our +ve and -ve examples in our training set. This is also called the decision boundary that separates the +ve and -ve training examples.

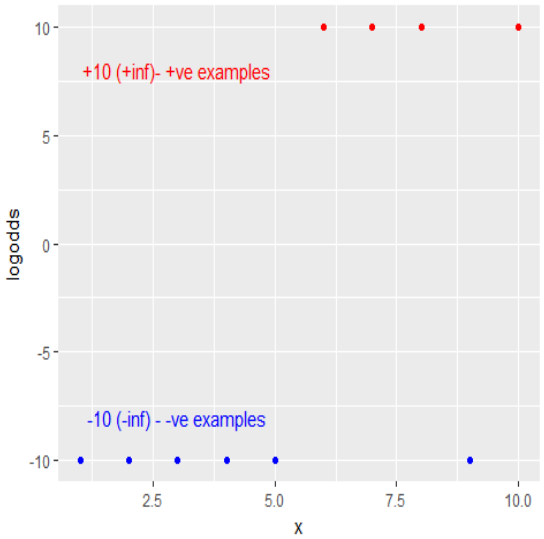
The following sections would illustrate how MLE works for Logistic Regression. We fit three different lines and study the likelihoods. Please note that the logodds would be +Inf when p = 1 and -inf when p = 0. We would take +10 and -10 to represent +Inf and -Inf for graphical convenience purposes.

Illustration of Maximum Likelihood Estimation for Logistic Regression

The following multi-plot function is borrowed [6] as it is from website http://www.cookbook-r.com/Graphs/Multiple_graphs_on_one_page_(ggplot2)/

```
##Ref-multi-plot : http://www.cookbook-r.com/Graphs/Multiple_graphs_on_one_page_(ggplot2)/
# Multiple plot function
#
# ggplot objects can be passed in ..., or to plotlist (as a list of ggplot objects)
# - cols:   Number of columns in layout
# - layout: A matrix specifying the layout. If present, 'cols' is ignored.
#
# If the layout is something like matrix(c(1,2,3,3), nrow=2, byrow=TRUE),
# then plot 1 will go in the upper left, 2 will go in the upper right, and
# 3 will go all the way across the bottom.
#
multiplot <-function(..., plotlist=NULL, file, cols=1, layout=NULL) {
library(grid)
```

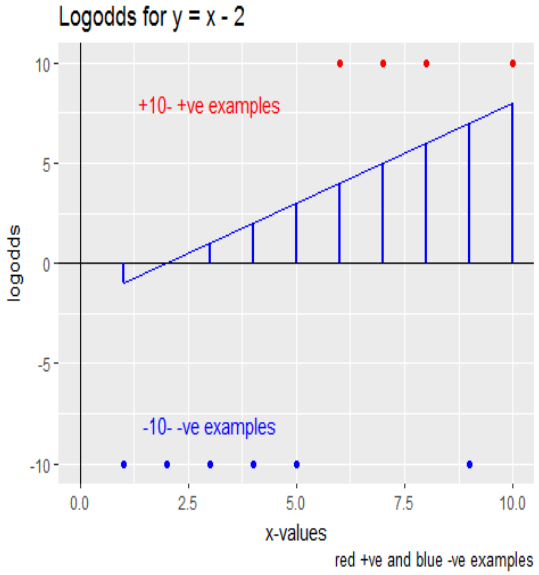
```
# Make a list from the ... arguments and plotlist
  plots <-c(list(...), plotlist)

  numPlots =length(plots)

# If layout is NULL, then use 'cols' to determine layout
  if (is.null(layout)) {
# Make the panel
# ncol: Number of columns of plots
# nrow: Number of rows needed, calculated from # of
cols
    layout <-matrix(seq(1, cols
*ceiling(numPlots/cols)),
ncol = cols, nrow =ceiling(numPlots/cols))
  }

 if (numPlots==1) {
print(plots[[1]])

  } else {
# Set up the page
grid.newpage()
pushViewport(viewport(layout
=grid.layout(nrow(layout), ncol(layout))))

# Make each plot, in the correct location
    for (i in 1:numPlots) {
# Get the i,j matrix positions of the regions that
contain this subplot
      matchidx <-as.data.frame(which(layout ==i,
arr.ind =TRUE))

print(plots[[i]], vp =viewport(layout.pos.row =
matchidx$row,
layout.pos.col = matchidx$col))
    }
  }
}

library(ggplot2)
x =c(1,2,3,4,5,6,7,8,9,10)
y =c(0,0,0,0,0,1,1,1,0,1)

data =data.frame(x,y)

data$logodds =ifelse(y ==0, -10,10)

p=ggplot(data)
+geom_point(aes(x=x,y=logodds),col=ifelse(data$log
odds >0,"red","blue")) +
annotate("text", x =3, y =8, label ="+10 (+inf)- +ve
examples",col=I("red")) +
annotate("text", x =3, y = -8, label ="-10 (-inf) - -ve
examples",col=I("blue"))
p
```



We treated +10 as +ve Inf and -10 as -ve Inf in the above figure. Now let us try to fit different straight lines and compute the overall likelihood for above 10 training examples. We are going to fit a candidate best fitting line y = x - 2 and determine the likelihood for the 10 training examples.

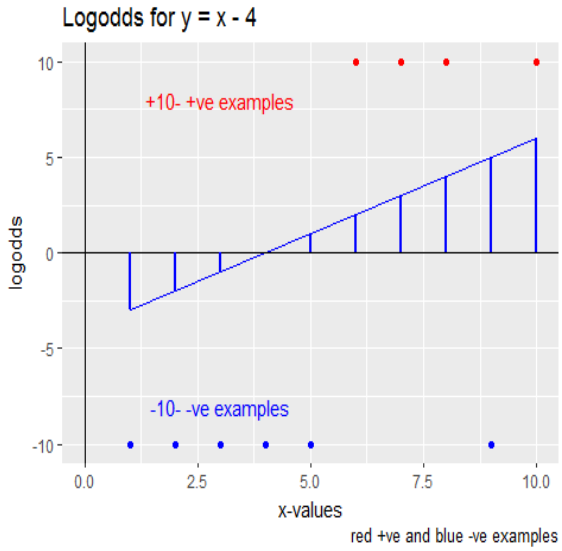
```
#Let us take one candidate best fitting line

data$y1 =x -2

p1 =1/(1+exp(-data$y1))

# Now calculate the likelihoods using the +ve and -ve
examples
# Considering the fact the probability of being 1 is 1 -
the probability of being 0

likelihoods1 =ifelse(y ==0,(1 -p1),p1)

# The total lokelihoods for best candidate best fitting
line 1

p=ggplot(data)
+geom_point(aes(x=x,y=logodds),col=ifelse(data$log
odds >0,"red","blue")) +
geom_path(aes(x=x,y=y1),col=I("blue")) +
geom_vline(xintercept =c(0),col=I("black"))+
geom_hline(yintercept =c(0),col=I("black"))+
geom_segment
(mapping=aes(x=x,y=0,xend=x,yend=y1), size=1,
linetype=1, color="blue")+
annotate("text", x =3, y =8, label ="+10- +ve
```

```
examples",col=I("red")) +
annotate("text", x =3, y = -8, label ="-10- -ve
examples",col=I("blue")) +
labs(title ="Logodds for y = x - 2",x="x-values"
,y="logodds",
caption ="red +ve and blue -ve examples")
p
```



Let us try to fit another candidate best fitting line y = x - 4 and calculate the likelihood.

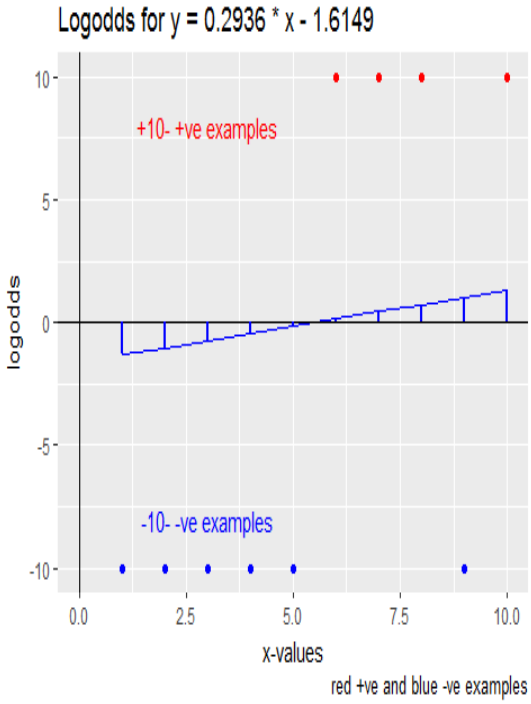
```
#Candidate Best fit line2
library(ggplot2)
x =c(1,2,3,4,5,6,7,8,9,10)
y =c(0,0,0,0,0,1,1,1,0,1)

data =data.frame(x,y)

data$logodds =ifelse(y ==0, -10,10)

data$y2 =x -4

p2 =1/(1+exp(-data$y2))

# Now calculate the likelihoods using the +ve and -ve
examples

likelihoods2 =ifelse(y ==0,(1 -p2),p2)
# The total lokelihoods

p=ggplot(data)
+geom_point(aes(x=x,y=logodds),col=ifelse(data$log
odds >0,"red","blue")) +
geom_path(aes(x=x,y=y2),col=I("blue")) +
geom_vline(xintercept =c(0),col=I("black"))+
geom_hline(yintercept =c(0),col=I("black"))+
geom_segment
(mapping=aes(x=x,y=0,xend=x,yend=y2), size=1,
linetype=1, color="blue")+
annotate("text", x =3, y =8, label ="+10- +ve
examples",col=I("red")) +
annotate("text", x =3, y = -8, label ="-10- -ve
examples",col=I("blue")) +
labs(title ="Logodds for y = x - 4",x="x-values"
,y="logodds",
caption ="red +ve and blue -ve examples")
p
```



Let us try to fit another candidate best fitting line y = 0.2936 * x - 1.6149, this is our best fit line as discussed in the following sections.

```
#Candidate Best fit line3

library(ggplot2)
x =c(1,2,3,4,5,6,7,8,9,10)
y =c(0,0,0,0,0,1,1,1,0,1)

data =data.frame(x,y)

data$logodds =ifelse(y ==0, -10,10)

data$y3 =-1.6149 +0.2936 *x

p3 =1/(1+exp(-data$y3))

# Now calculate the likelihoods using the +ve and -ve
examples

likelihoods3 =ifelse(y ==0,(1 -p3),p3)

#Now project the those training examples to our guess
fit line.
```

```
p=ggplot(data)
+geom_point(aes(x=x,y=logodds),col=ifelse(data$log
odds >0,"red","blue")) +
geom_path(aes(x=x,y=y3),col=I("blue")) +
geom_vline(xintercept =c(0),col=I("black"))+
geom_hline(yintercept =c(0),col=I("black"))+
geom_segment
(mapping=aes(x=x,y=0,xend=x,yend=y3), size=1,
linetype=1, color="blue")+
annotate("text", x =3, y =8, label ="+10- +ve
examples",col=I("red")) +
annotate("text", x =3, y = -8, label ="-10- -ve
examples",col=I("blue")) +
labs(title ="Logodds for y = 0.2936 * x - 1.6149
",x="x-values" ,y="logodds",
caption ="red +ve and blue -ve examples")
p
```



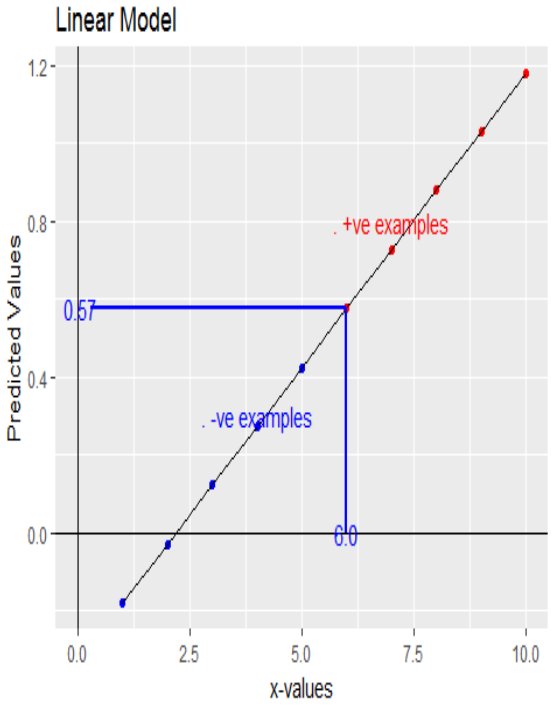

```
prod(likelihoods1)

## [1] 4.924988e-07

prod(likelihoods2)

## [1] 0.0004537146

prod(likelihoods3)

## [1] 0.00588543
```

Out of the above three candidate lines, y = 0.2936 * x - 1.6149 is the one with maximum likelihood of 0.00588543.

Applying Linear Model for categorical data

We will use the Linear Model when we have the outcome variable is continuous. Let us study what happens if we train our model using Linear Regression when our outcome variable is categorical.

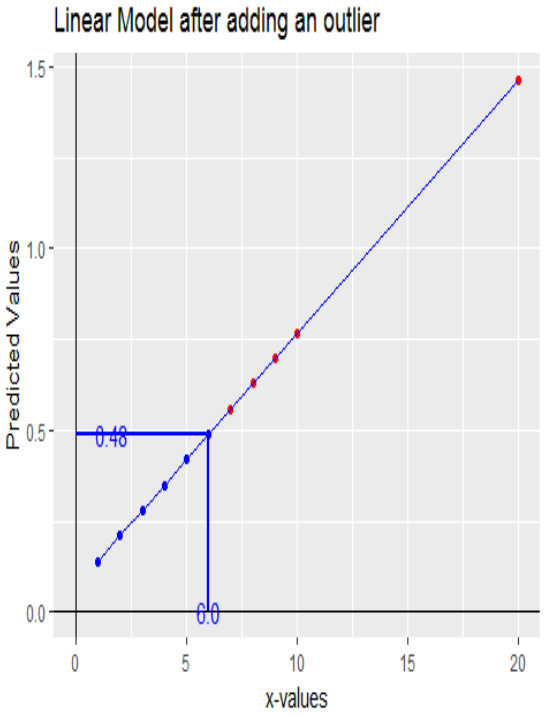
```
x =c(1,2,3,4,5,6,7,8,9,10)
y =c(0,0,0,0,0,1,1,1,1,1)

data =data.frame(x,y)

m =lm(y~x,data=data)

#Now let us see what is the predicted value for 6

new_data =data.frame(x=6)
predict(m,new_data)

##         1
## 0.5757576

y1 <-coef(m)[1] +coef(m)[2] *6

data$pr=predict(m,data)

p=ggplot(data)
+geom_point(aes(x=x,y=pr),col=ifelse(data$pr>0.5,"r
ed","blue")) +
geom_path(aes(x=x,y=pr),col=I("black")) +
geom_vline(xintercept =c(0),col=I("black"))+
geom_hline(yintercept =c(0),col=I("black"))+
annotate("text", x =7, y =0.8, label =". +ve
examples",col=I("red")) +
annotate("text", x =4, y =0.3, label =". -ve
examples",col=I("blue")) +
annotate("text", x =6, y =0.0, label
="6.0",col=I("blue")) +
annotate("text", x =0, y = y1, label ="
0.57",col=I("blue")) +
geom_segment
(mapping=aes(x=6,y=0,xend=6,yend=y1), size=1,
linetype=1, color="blue")+
geom_segment
(mapping=aes(x=0.3,y=y1,xend=6,yend=y1), size=1,
linetype=1, color="blue")+
labs(title ="Linear Model",x="x-values" ,y="Predicted
Values")
p
```



The most important issue that we observe in this is that the predicted values would go beyond 1 and go below 0 which are not relevant for Logistic Regression where the values should be between 0 and 1.

Now lust us study the impact on the Linear Model of adding an outlier to this data.

```
x =c(1,2,3,4,5,6,7,8,9,10,20)
y =c(0,0,0,0,0,1,1,1,1,1,1)
dfx =data.frame(x)
data =data.frame(x,y)

m =lm(y~x,data=data)

new_data =data.frame(x=6)
predict(m,new_data)

##         1
## 0.4883721

y1 =coef(m)[1] +coef(m)[2] *6

data$pr=predict(m,data)

p=ggplot(data)
+geom_point(aes(x=x,y=pr),col=ifelse(data$pr>0.5,"r
ed","blue")) +
geom_path(aes(x=x,y=pr),col=I("blue")) +
geom_vline(xintercept =c(0),col=I("black"))+
geom_hline(yintercept =c(0),col=I("black"))+
annotate("text", x =6, y =0.0, label
="6.0",col=I("blue")) +
annotate("text", x =1.5, y = y1, label ="
0.48",col=I("blue")) +
geom_segment
(mapping=aes(x=6,y=0,xend=6,yend=y1), size=1,
linetype=1, color="blue")+
geom_segment
(mapping=aes(x=0,y=y1,xend=6,yend=y1), size=1,
linetype=1, color="blue")+
labs(title ="Linear Model after adding an
outlier",x="x-values" ,y="Predicted Values")
p
```



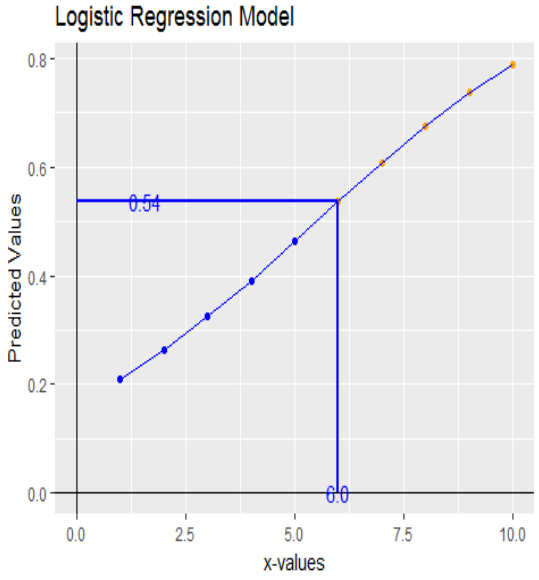
When we add an outlier, the linear regression performs badly and 6 is now treated as a -ve example though it is not.

Logistic Regression Model

Logistic Regression provides a solution to these issues that we observed in the above examples. Let us apply the logistic regression model for the above example and see how it performs in case of outliers.

```
x =c(1,2,3,4,5,6,7,8,9,10)
y =c(0,1,0,0,0,1,1,1,0,1)

data =data.frame(x,y)

m =glm(data=data,y~x,binomial(link ="logit"),maxit
=100)
summary(m)

##
```

```
## Call:
## glm(formula = y ~ x, family = binomial(link =
"logit"), data = data,
##     maxit = 100)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.6331  -0.9691   0.0000   0.9691   1.6331
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept)  -1.6149     1.5589  -1.036    0.300
## x             0.2936     0.2546   1.153    0.249
##
## (Dispersion parameter for binomial family taken to
be 1)
##
##     Null deviance: 13.863  on 9  degrees of freedom
## Residual deviance: 12.326  on 8  degrees of
freedom
## AIC: 16.326
##
## Number of Fisher Scoring iterations: 4

dec_boundary =coef(m)[1] +coef(m)[2] *x

y_pred <-predict(m,data,type="response")

y1 =1/(1+exp(-coef(m)[1] -coef(m)[2] *6))

plot_data <-data.frame(x=x,y=dec_boundary,y_pred)

p=ggplot(plot_data)
+geom_point(aes(x=x,y=y_pred),col=ifelse(dec_boun
dary>0,"orange","blue")) +
geom_path(aes(x=x,y=y_pred),col=I("blue")) +
geom_vline(xintercept =c(0),col=I("black"))+
geom_hline(yintercept =c(0),col=I("black"))+
geom_segment
(mapping=aes(x=6,y=0,xend=6,yend=y1), size=1,
linetype=1, color="blue")+
geom_segment
(mapping=aes(x=0,y=y1,xend=6,yend=y1), size=1,
linetype=1, color="blue")+
annotate("text", x =6, y =0.0, label
="6.0",col=I("blue")) +
annotate("text", x =1.5, y = y1, label ="
0.54",col=I("blue")) +
labs(title ="Logistic Regression Model",x="x-values"
,y="Predicted Values")
p
```

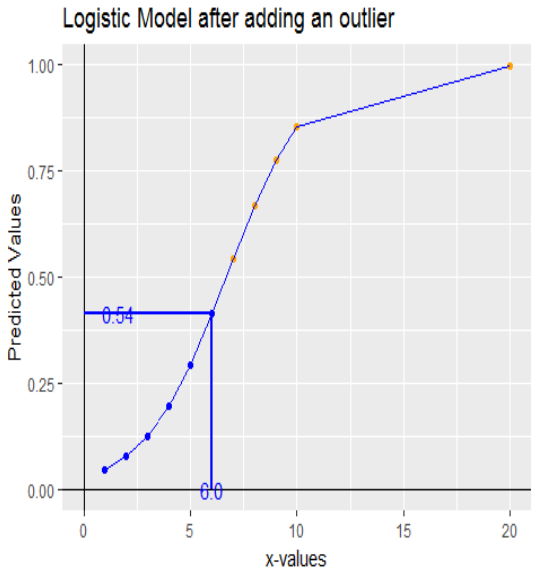


When we apply logistic regression, the predicted values are limited between 0 and 1, which is solving the first issue that we faced when we apply linear regression.

Let us add an outlier and check the how logistic regression performs on this data

```
x =c(1,2,3,4,5,6,7,8,9,10,20)
y =c(0,0,0,0,0,1,1,1,1,0,1)
data =data.frame(x,y)

m =glm(data=data,y~x,binomial(link ="logit"),maxit
=100)

y_pred =round(1/(1+exp(-coef(m)[1] -coef(m)[2]
*x)))
y_pred =1/(1+exp(-coef(m)[1] -coef(m)[2] *x))

dec_boundary =coef(m)[1] +coef(m)[2] *x

y1 =1/(1+exp(-coef(m)[1] -coef(m)[2] *6))

plot_data <-data.frame(x=x,y=y_pred)

p=ggplot(plot_data)
+geom_point(aes(x=x,y=y_pred),col=ifelse(y_pred>0.
5,"orange","blue")) +
geom_path(aes(x=x,y=y_pred),col=I("blue")) +
geom_vline(xintercept =c(0),col=I("black"))+
geom_hline(yintercept =c(0),col=I("black"))+
geom_segment
(mapping=aes(x=6,y=0,xend=6,yend=y1), size=1,
linetype=1, color="blue")+
geom_segment
(mapping=aes(x=0,y=y1,xend=6,yend=y1), size=1,
```

```
linetype=1, color="blue")+
annotate("text", x =6, y =0.0, label
="6.0",col=I("blue")) +
annotate("text", x =1.5, y = y1, label ="
0.54",col=I("blue")) +
labs(title ="Logistic Model after adding an
outlier",x="x-values",y="Predicted Values")
p
```



As expected the outlier did not influence the prediction in logistic model unlike it did in linear model.

Decision Boundary [1]

Decision Boundary separates +ve and -ve examples in Logistic Regression.

The following is the hypothesis function for logistic regression [3]

$$Y = \frac{1}{(1 + e^{-\theta^T X})}$$

where

$$\theta^T X = 0$$

isthe decision boundary. The decision boundary could be linear in x or any higher order polynomial.

Whenever

$$\theta^T X >= 0$$

, then Y would be greater than 0.5 and treated as 1 and in this case Y would identify the +ve cases. Whenever

$$\theta^T X < 0$$

, then Y would be less than 0.5 and treated as 0 and in this case Y would identify the +ve cases.

$$\theta^T X = 0$$

is called the decision boundary that separates +ve and negative examples.

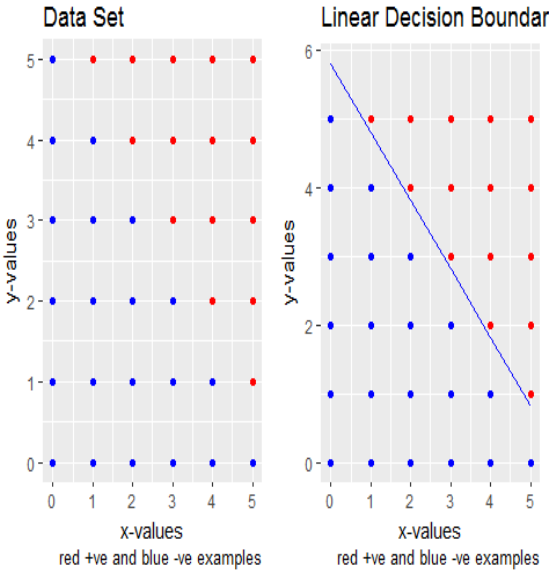
Let us now study different decision boundary scenarios

Linear Decision Boundary

Linear Decision Boundary is a line that separates positive and negative examples. We are using the following training data set

```
library(ggplot2)

x
=c(0,0,0,0,0,0,1,1,1,1,1,1,2,2,2,2,2,2,3,3,3,3,3,3,4,4,4,4
,4,4,5,5,5,5,5,5)
y
=c(0,1,2,3,4,5,0,1,2,3,4,5,0,1,2,3,4,5,0,1,2,3,4,5,0,1,2,3
,4,5,0,1,2,3,4,5)
z
=c(0,0,0,0,1,1,0,0,0,1,1,1,0,0,1,1,1,1,0,1,1,1,1,1,1,1,1,1
,1,1,1,1,1,1,1,0)
y1=c(5,5,5,5,5,5,4,4,4,4,4,4,3,3,3,3,3,3,2,2,2,2,2,2,1,1,
1,1,1,1,0,0,0,0,0,0)

z  =ifelse(x+y >5,1,0)
z[36] <-0
data =data.frame(x,y,z,y1)

m =glm(data=data,z~x+y,binomial(link
="logit"),maxit=100)

# The decision boundary
y1  =(-coef(m)[1] -coef(m)[2] *x)/coef(m)[3]

data1 =data.frame(x,y,y1)

plot1=ggplot(data1)
+geom_point(aes(x=x,y=y),col=ifelse(x+y>-
coef(m)[1]/coef(m)[3],"red","blue")) +
labs(title ="Data Set",x="x-values" ,y="y-values",
caption ="red +ve and blue -ve examples")

plot2=ggplot(data1)
+geom_point(aes(x=x,y=y),col=ifelse(x+y>-
coef(m)[1]/coef(m)[3],"red","blue")) +
geom_path(aes(x=x,y=y1),col=I("blue")) +
labs(title ="Linear Decision Boundary",x="x-values"
,y="y-values",
```

```
caption ="red +ve and blue -ve examples")

multiplot(plot1, plot2, cols=2)
```



Quadratic Decision Boundary

The Decision Boundary is a quadratic curve that separates positive and negative examples. We are using the following training set to train our model

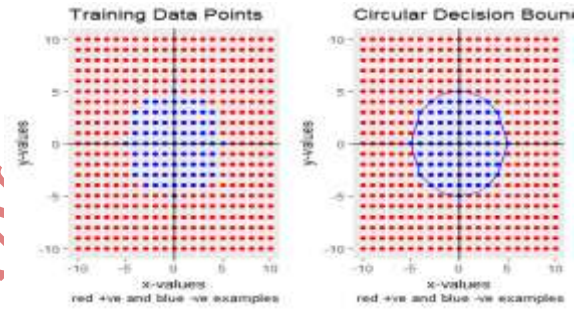
```
x =seq(0,4)
y =seq(0,15)

grid =expand.grid(x = x, y = y)

grid$z =ifelse(grid$y >grid$x*grid$x,1,0)

data =data.frame(x=grid$x,y=grid$y,z=grid$z)

data[52,]$z =0
data[80,]$z =1

data <-data[order(data$x, data$y),]

m =glm(data=data,z~I(x^2)+y,binomial(link
="logit"),maxit=100)

y1 <-(-coef(m)[1] -coef(m)[2] *data$x
*data$x)/coef(m)[3]

data1 <-data.frame(x=data$x,y=data$y,y1)

plot1=ggplot(data1)
+geom_point(aes(x=x,y=y),col=ifelse(coef(m)[3]*dat
a1$y>-coef(m)[2]*data1$x*data1$x -
coef(m)[1],"red","blue")) +
labs(title ="Training Data Points",x="x-values" ,y="y-
values",
caption ="red +ve and blue -ve examples")

plot2=ggplot(data1)
+geom_point(aes(x=x,y=y),col=ifelse(coef(m)[3]*dat
a1$y>-coef(m)[2]*data1$x*data1$x -
coef(m)[1],"red","blue")) +
geom_path(aes(x=data1$x,y=data1$y1),col=I("blue")
) +
labs(title ="Quadratic Decision Boundary",x="x-
values" ,y="y-values",
caption ="red +ve and blue -ve examples")

multiplot(plot1,plot2,cols=2)
```

Circular Decision Boundary

We will create a training data set where the +ve and -ve examples are separated by a circular decision boundary. We will determine the decision boundary using our regression model.

```
library(ggplot2)

seq =seq(-10,10)

x =seq
y =seq

data =expand.grid(x = x, y = y)


data$z =ifelse((data$x^2+data$y^2) >25, 1, 0 )
data[1,]$z =0
data[441,]$z =0
data[221,]$z =1
```

```
data[158,]$z =1

m
=glm(data=data,z~I(data$x^2)+I(data$y^2),binomial(
link ="logit"),maxit =100)

tx =-(0.10390 *x*x +0.10521*y*y -2.62588)

#This is the equation of a circle of radius
2.62588/0.10390, which is sqrt(25.2) but let us use 5 as
the radius for drawing
library(ggplot2)

data$y1 =ifelse(abs(data$x) >5,0,sqrt(abs(25 -
data$x^2)))

data$y2 =ifelse(abs(data$x) >5,0,-sqrt(abs(25 -
data$x^2)))

data <-data[order(data$x),]

plot1 <-ggplot(data)
+geom_point(aes(x=data$x,y=data$y),col=ifelse(data
$x^2+data$y^2>25,"red","blue")) +
geom_vline(xintercept =c(0),col=I("black"))+
geom_hline(yintercept =c(0),col=I("black"))+
labs(title ="Training Data Points",x="x-values" ,y="y-
values",
caption ="red +ve and blue -ve examples")

plot2 <-ggplot(data)
+geom_point(aes(x=data$x,y=data$y),col=ifelse(data
$x^2+data$y^2>25,"red","blue")) +
geom_path(aes(x=data$x,y=data$y1),col=I("blue")) +
geom_path(aes(x=data$x,y=data$y2),col=I("blue")) +
geom_vline(xintercept =c(0),col=I("black"))+
geom_hline(yintercept =c(0),col=I("black"))+
labs(title ="Circular Decision Boundary",x="x-values"
,y="y-values",
caption ="red +ve and blue -ve examples")

multiplot(plot1,plot2,cols=2)
```



References

1. https://www.coursera.org/learn/machine-learning/lecture/WuL1H/decision-boundary, Andrew Ng, Coursera, Stanford University
2. https://www.coursera.org/learn/machine-learning/lecture/RJXfB/hypothesis-representation, Andrew Ng, Coursera, Stanford University
3. https://www.coursera.org/learn/machine-learning/lecture/RJXfB/hypothesis-representation, Andrew Ng, Coursera, Stanford University
4. https://en.wikipedia.org/wiki/Logistic_regression
5. https://www.youtube.com/watch?v=BfKanl1aSG0, StatQuest with Josh Starmer
6. http://www.cookbook-r.com/Graphs/Multiple_graphs_on_one_page_(ggplot2)/, by Winston Chang
7. https://en.wikipedia.org/wiki/Normal_distribution
8. https://www.youtube.com/watch?v=Dn6b9fCIUpM, StatQuest with Josh Starmer

