Summer Project 2-2020

Shuting Wang 002251577
Zhaoyi Wang 002253523
Yukuan Hao 002253251
Liang Gu 002253248

Logistic Regression and Applications using R Language

2. Logistic Regression and Classification

(a) Show that:

$$p(y) = S(\beta_0 + \beta_1 x_1 + \ldots + \beta_k x_k)$$

where S(ω) is the logistic sigmoid function given by:

$$S(\omega) = \frac{1}{1+e^{-\omega}} = \frac{e^{\omega}}{1+e^{\omega}}$$

The logistic regression is given by:

$$\log \frac{p(y)}{1-p(y)} = \beta_0 + \beta_1 x_1 + \ldots + \beta_k x_k \qquad (1)$$

From the exponential functions:

$$\log_b^y = x \text{ means } b^x = y$$

We got:

$$\frac{p(y)}{1-p(y)} = e^{\beta_0+\beta_1 x_1+\ldots+\beta_k x_k}$$

Let $\omega = \beta_0 + \beta_1 x_1 + \ldots + \beta_k x_k$

$$\frac{p(y)}{1-p(y)} = e^{\omega}$$

$$p(y) = e^{\omega} - p(y)e^{\omega}$$

$$p(y) + p(y)e^{\omega} = p(y)(1+e^{\omega}) = e^{\omega}$$

$$p(y) = \frac{e^{\omega}}{1+e^{\omega}}$$

As the logistic sigmoid function:

$$S(\omega) = \frac{1}{1+e^{-\omega}} = \frac{e^{\omega}}{1+e^{\omega}}$$

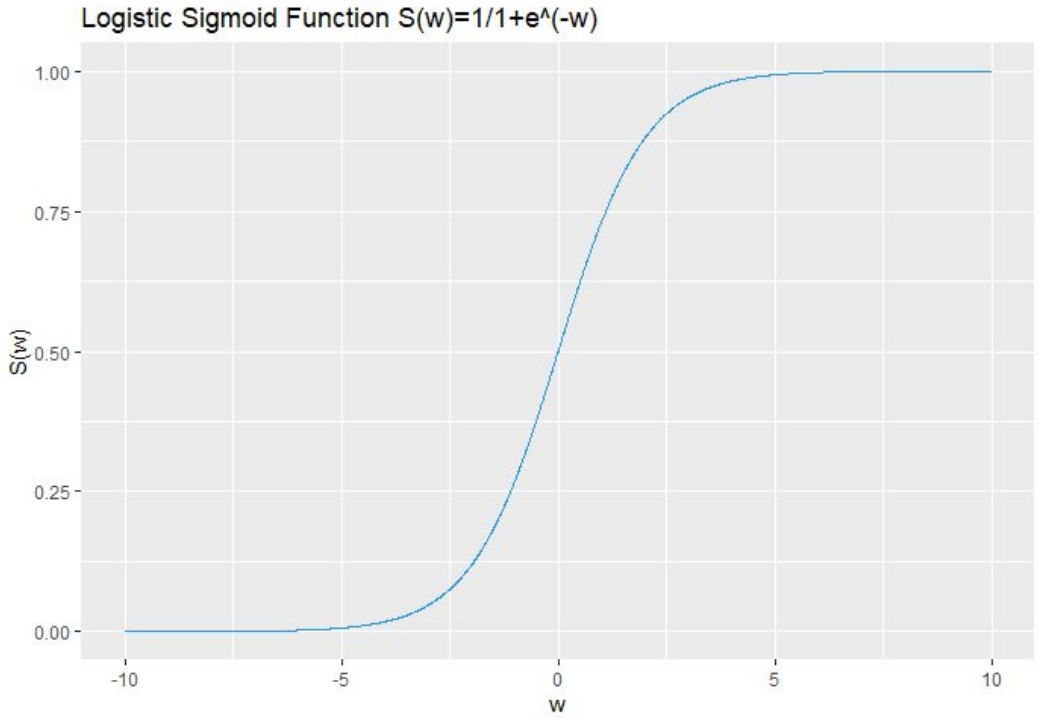
We got:

$$p(y) = S(\beta_0 + \beta_1 x_1 + \ldots + \beta_k x_k)$$

[1]

(b) Display a graph of S(w) for $-\infty < \omega < +\infty$ and show that no matter what values $\beta_0, \beta_1 \ldots, \beta_k$ or $x_1, \ldots, x_k$ take, p(y) will always have values between 0 and 1.

```
x<-seq(-10,10,0.01)
y=1/(1+exp(-x))
ggplot(data=NULL,aes(x=x,y=y))+
  xlab('w')+ylab('S(w)')+
  ggtitle('Logistic Sigmoid Function S(w)=1/1+e^(-w)')+
  geom_line(col=420)
```



From the plot, it can be found during the growth of w, the value of p(y) approaches to 1. After p(y) arrives 1, it will no longer increase. Similarly, when w reduced, the value of p(y) gradually decreases to 0 and will maintain this value no longer decrease.

2.1 Maximum Likelihood Estimation (MLE) of the Model

(1) Assuming you are given a dataset with n training examples and k features, write down a formula for the conditional likelihood $L(\beta)$ and the log likelihood $l(\beta) = logL(\beta)$ of the training data in terms of the class labels y_i, the features $x_1^{(i)}, \ldots, x_k^{(i)}$, and the parameters $\beta_0, \beta_1, \ldots, \beta_k$, where the superscript (i) denotes the sample index. This will be your objective function (or cost function).

The probability of the class is either p_i, if $y_i = 1$,or $1 - p_i$, if $y_i = 0$, the likelihood is:

$$L(\beta) = \prod_{i=1}^{k} p(x_i)^{y_i}[1 - p(x_i)]^{1-y_i}$$

$$p_i = \frac{e^{x_i^T \beta}}{1+e^{x_i^T \beta}}$$

By using logistic function (1) the $l(\beta)$ can be calculated:

$$l(\beta) = \sum_{i=1}^{k} y_i log\, p(x_i) + (1 - y_i) log[\, 1 - p(x_i)]$$
$$= \sum_{i=1}^{k} log[1 - p(x_i)] + \sum_{i=1}^{k} y_i log\, \frac{p(x_i)}{1-p(x_i)}$$
$$= \sum_{i=1}^{k} log[1 - p(x_i)] + \sum_{i=1}^{k} y_i(\beta_0 + x_i\beta_k)$$
$$= \sum_{i=1}^{k} - log(1 + e^{\beta_0 + x_i\beta_k}) + \sum_{i=1}^{k} y_i(\beta_0 + x_i\beta_k)$$

(2) Compute the partial derivative of the objective function with respect to β_j, i.e. derive $\partial l/\partial\beta_j$, for $0 \le j \le k$ where l is the objective function that you provided above. Show particularly that

$$\frac{\partial l}{\partial \beta_j} = \sum_{i=1}^{n} (y_i - p_i)\, x_j^{(i)},\ 0 \le j \le k \qquad (2)$$

where

$$p_i = p(y_i = 1) = \frac{e^{\beta_0 + \beta_1 x_1^{(i)} + \ldots + \beta_k x_k^{(i)}}}{1 + e^{\beta_0 + \beta_1 x_1^{(i)} + \ldots + \beta_k x_k^{(i)}}}$$

We recall that for j = 0, we set $x_1^{(i)}$ =1 for every i.

$$\frac{\partial l}{\partial \beta_j} = \frac{\partial L(L(\beta))}{\partial \beta_j}$$
$$= \frac{\partial}{\partial \beta_j} \sum_{i=1}^{n} y_i log p_i + \frac{\partial}{\partial \beta_j} \sum_{i=1}^{n} (1 - y_i) log(1 - p_i)$$
$$= \sum_{i=1}^{n} (\frac{y_i}{p_i} - \frac{1-y_i}{1-p_i}) \frac{\partial}{\partial \beta_j} \sum_{i=1}^{n} p_i \qquad derivative\ of\ log\, f(x)$$
$$= \sum_{i=1}^{n} (\frac{y_i}{p_i} - \frac{1-y_i}{1-p_i})\, p_i(1 - p_i)\, x_j^{(i)} \qquad chain\ rule$$
$$= \sum_{i=1}^{n} [\frac{y_i - p_i}{p_i(1-p_i)}]\, p_i(1 - p_i)\, x_j^{(i)} \qquad algebraic\ manipulation$$
$$= \sum_{i=1}^{n} (y_i - p_i)\, x_j^{(i)}$$

$$\frac{\partial l}{\partial \beta_j} = - \sum_{i=1}^{n} \frac{e^{\beta_0 + \beta_1 x_1^{(i)} + \ldots + \beta_k x_k^{(i)}}}{1 + e^{\beta_0 + \beta_1 x_1^{(i)} + \ldots + \beta_k x_k^{(i)}}} + \sum_{i=1}^{n} y_i x_j^{(i)}$$

$$= \sum_{i=1}^{n} (y_i - p_i)\, x_j^{(i)}$$

(3) To maximize the log-likelihood, we set the partial derivatives to zero. Show that when j = 0, the score equation is equivalent to

$$\sum_{i=1}^{n} y_i = \sum_{i=1}^{n} p_i$$

Does this equation have any statistical interpretation?

As we have the partial derivative of the objective function with respect to β_j function (2), when j=0, $x_j^{(i)}$ =1. To get the maximize log-likelihood, let function (2)=0:

$$\frac{\partial l}{\partial \beta_j} = \sum_{i=1}^{n}(y_i - p_i) = 0$$

$$\sum_{i=1}^{n} y_i = \sum_{i=1}^{n} p_i$$

Which means the final target matches the prediction.

(4) The obtained equations are known to be transcendental and not to have closed form solutions. Explain this statement by studying the equations and by researching relevant literature on this subject.

Consider the equation:

$$l(\beta) = \sum_{i=1}^{k} - log(1 + e^{\beta_0 + x_i\beta_k}) + \sum_{i=1}^{k} y_i(\beta_0 + x_i\beta_k) \tag{3}$$

consists of non algebraic terms like logarithms and exponents is transcendental no single solution, to approximate the solution, we need to use some approach, like Newton-Raphson[2] to get $\nabla_\beta l(\beta^*)$ first.

$$\nabla_\beta l(\beta^*) = (\beta^* - \beta)\nabla_{\beta\beta} l(\beta) + \nabla_\beta l(\beta) = 0$$

$$\beta^* = \beta - \frac{\nabla_\beta l(\beta)}{\nabla_{\beta\beta} l(\beta)} \tag{4}$$

As a derivative of sum is equal to the sum of derivatives, we bring the gradient symbol into the summation.

$$\nabla_\beta l = \nabla_\beta \sum_{i=1}^{k} - log(1 + e^{\beta_0 + x_i\beta_k}) + y_i(\beta_0 + x_i\beta_k)$$

$$= \sum_{i=1}^{k} \nabla_\beta [- log(1 + e^{\beta_0 + x_i\beta_k}) + y_i(\beta_0 + x_i\beta_k)]$$

$$= \sum_{i=1}^{k} - \frac{e^{\beta_0 + x_i\beta_k}}{1 + e^{\beta_0 + x_i\beta_k}} x_i + y_i x_i$$

$$= \sum_{i=1}^{k} y_i x_i - \frac{x_i}{1 + e^{-(\beta_0 + x_i\beta_k)}}$$

$$= \sum_{i=1}^{k} [y_i - p(x_i)]x_i \tag{5}$$

Hessian matrix:

$$\nabla_{\beta\beta} l(\beta) = \nabla_\beta \sum_{i=1}^{k} [y_i - p(x_i)]x_i$$

$$= \sum_{i=1}^{k} \nabla_\beta [y_i - p(x_i)]x_i$$

$$= \sum_{i=1}^{k} \nabla_\beta - p(x_i)x_i$$

$$= \sum_{i=1}^{k} [\frac{1}{1 + e^{-(\beta_0 + x_i\beta_k)}}]^2 e^{-(\beta_0 + x_i\beta_k)}(- x_i)x_i$$

$$= - \sum_{i=1}^{k} [\frac{e^{-(\beta_0 + x_i\beta_k)}}{1 + e^{-(\beta_0 + x_i\beta_k)}}][\frac{1}{1 + e^{-(\beta_0 + x_i\beta_k)}}] x_i^T x_i$$

$$= - \sum_{i=1}^{k} p(x_i)[1 - p(x_i)] x_i^T x_i \tag{6}$$

Convert (5) and (6) into matrix representation:

$$\nabla_{\beta} l = X^T (Y - \hat{Y}) \tag{7}$$

$$\nabla_{\beta\beta} l(\beta) = -X^T P(1-P)X = -X^T W X \tag{8}$$

$$W = P(1-P)$$

Plugging these two terms(7, 8) into Newton-Raphson equation:

$$\beta^{t+1} = \beta^t - \frac{\nabla_{\beta} l(\beta^t)}{\nabla_{\beta\beta} l(\beta^t)} = \beta^t X^T (Y - \hat{Y}^t)[X^T W^t X]^{-1} \tag{9}$$

After that, we need to execute iterative operations on T until the β value converges. Once the coefficient is estimated, we can insert the value of some feature vector X and $l(\beta)$ can be solved.

(5) Typically, numerical approximations and optimization procedures are used to nd the (best) vector satisfying

$$\beta^* = argmax(l(\beta))$$

Algorithm 1 [Algorithm for finding β^*]

Set $\eta \in [0, 1]$ (learning coefficient)
Set $\epsilon > 0$ (tolerance term)
$\beta^{(0)} \leftarrow initial\_value$
for t = 0, 1, ... **do**
  Compute the gradient: $g_t = \nabla \ell(\beta^{(t)})$
  Update the coefficients: $\beta^{(t+1)} \leftarrow \beta^{(t)} + \eta g_t$
  Iterate until: $||\beta^{(t+1)} - \beta^{(t)}|| < \epsilon$.
end for
Return the final coefficients: $\beta^{(t^{\text{final}})}$

Using the first 100 rows of 'ldl' for training and the values in the remaining rows for testing, 'chd' as label.

- **Normalize the feature variable x_1 .**

Create a function named 'f_nor', set 'mu' as μ and 'sigma' as σ,
$x = \frac{x-\mu}{\sigma}$ code set as below:

```
f_nor=function(data){
  mu=mean(data)
  sigma=sd(data)
  res=as.matrix((data-mu)/sigma)
  return(res)
}
```

Data after normalized x:

	sbp	tobacco	ldl	adiposity	typea	obesity	alcohol	age	data$chd
1	1.05741729	1.821098803	0.47789413	-0.295183212	-0.41601699	-0.176594453	3.27418871	0.62865426	1
2	0.27678925	-0.789381743	-0.15950708	0.411694189	0.19313443	0.670645918	-0.61208112	1.38161701	1
3	-0.99173133	-0.774141239	-0.60858520	0.883374200	-0.11244128	0.734722921	-0.54059729	0.21794730	0
4	1.54530982	0.841352143	0.80625232	1.622382392	-0.21429985	1.411091285	0.29474240	1.03936121	1
5	-0.21110328	2.169453171	-0.59892761	0.305019963	0.70242729	-0.012842112	1.64599115	0.42330078	1

• **Calculate the value of the objective function $l(\beta_0, \beta_1)$.**

Initialize β =0 and w=x* β , put into equation:

$$l(\beta) = \sum_{i=1}^{k} y_i(\beta_0 + x_i\beta_k) - log(1 + e^{\beta_0 + x_i\beta_k})$$

```
f_w=function(X,Beta){return(X%*%Beta)}
f_log=function(W){
    res=sum(Y*W-log(1+exp(W)))
  return(res)
}
```

• **Choose a value of the learning rate η**

learning coefficient η : *eta* = 0.00001

tolerance term ε : *epsilon* = 0.0000001

• **Initialize the parameter value and calculate the gradient $\nabla l(\beta_0, \beta_1)$**

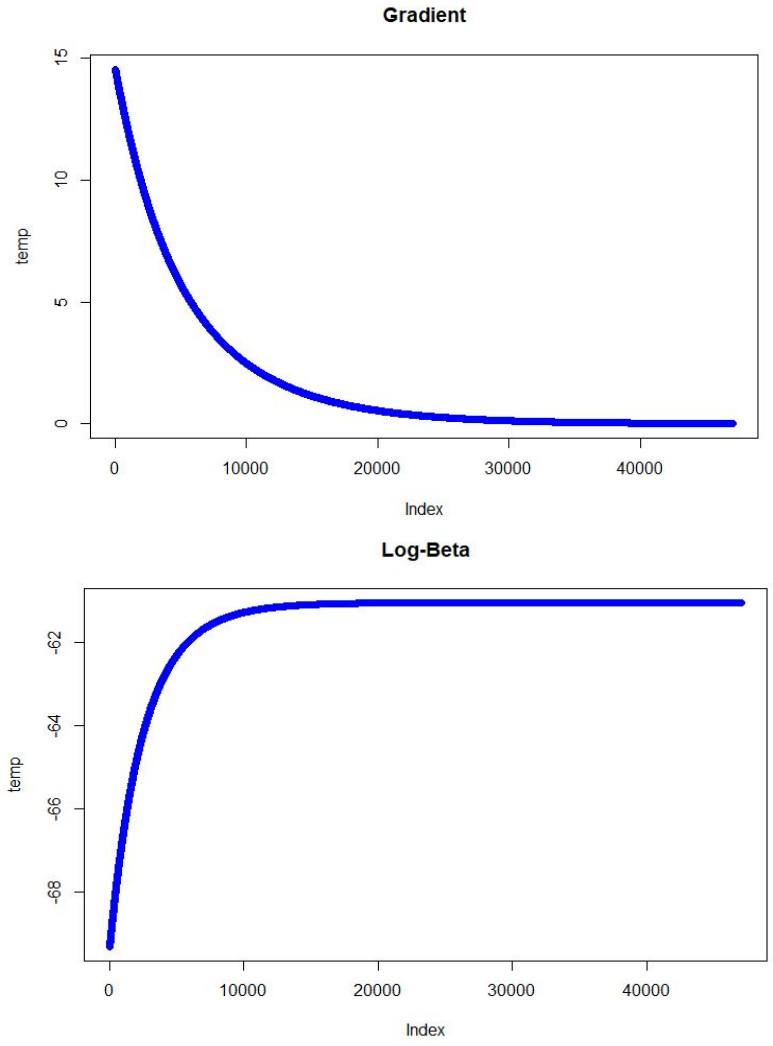
```
f_sigmoid=function(w){
  return(1/(1+exp(-w)))
}
f_gradient=function(X,Y,Beta){
  gra=t(X)%*%(Y-f_sigmoid(f_w(X,Beta)))
  return(gra)
}
```

• **Update the parameter value.**

```
if(all(abs(minus)<epsilon)){
  cat('loop time: ',i)
  break
}
```

• **Check whether gradient ascent has converged.**

```
plot(temp,col='blue',main = 'log-beta')
```



As can be seen from the above figures, the gradient has converged, and the slope of the log-beta map is approaching horizontal.

- **Complete the implementation of gradient ascent.**

```
for(i in 1:50000){
  w=X%*%Beta
  temp[i]=f_log(w)
  gra=f_gradient(X,Y,Beta)
  # temp[i]=gra[2]
  minus=eta*gra
  Beta=Beta+minus
# 5.Update gradient value
  if(all(abs(minus)<epsilon)){
    cat('loop time: ',i)
    break
  }
}
```

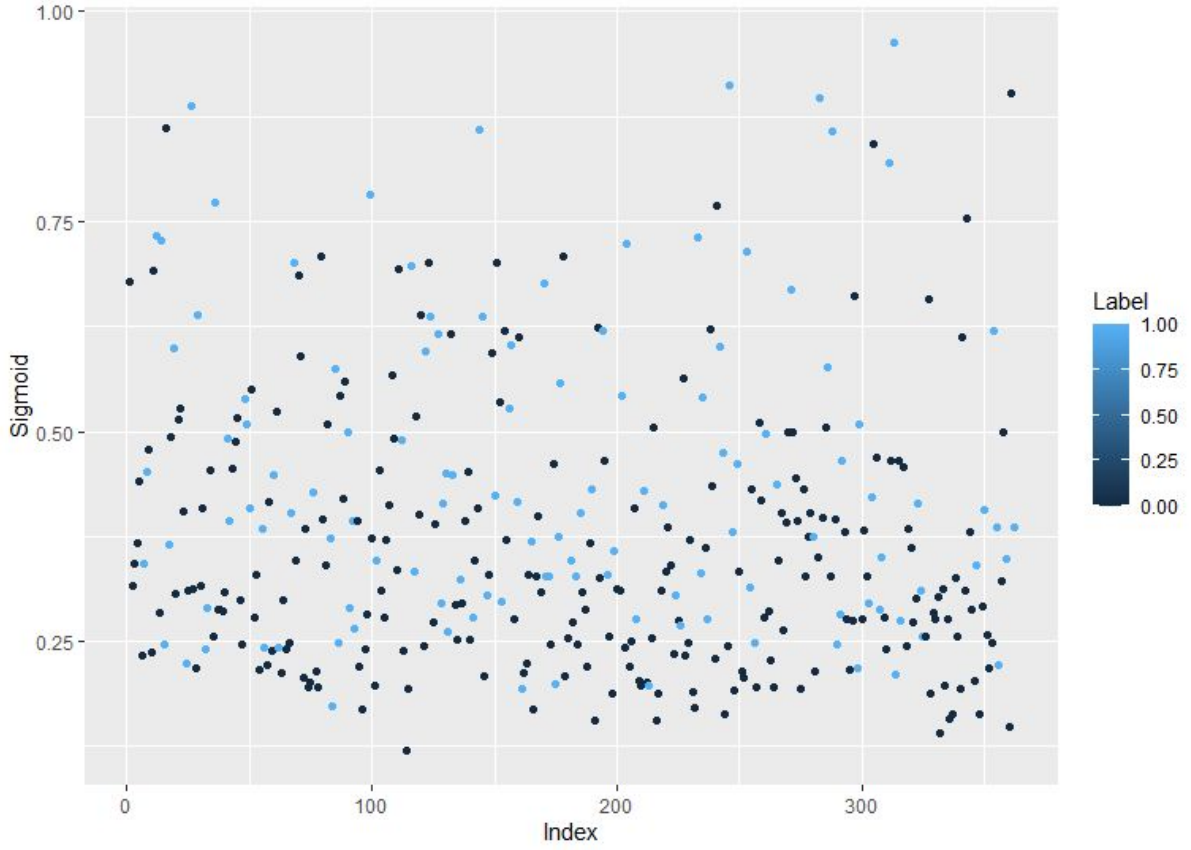
$\beta_0 = -0.492$ $\beta_1 = 0.822$

$\nabla l(\beta_0) = -0.0003524922$ $\nabla l(\beta_1) = 0.001998913$

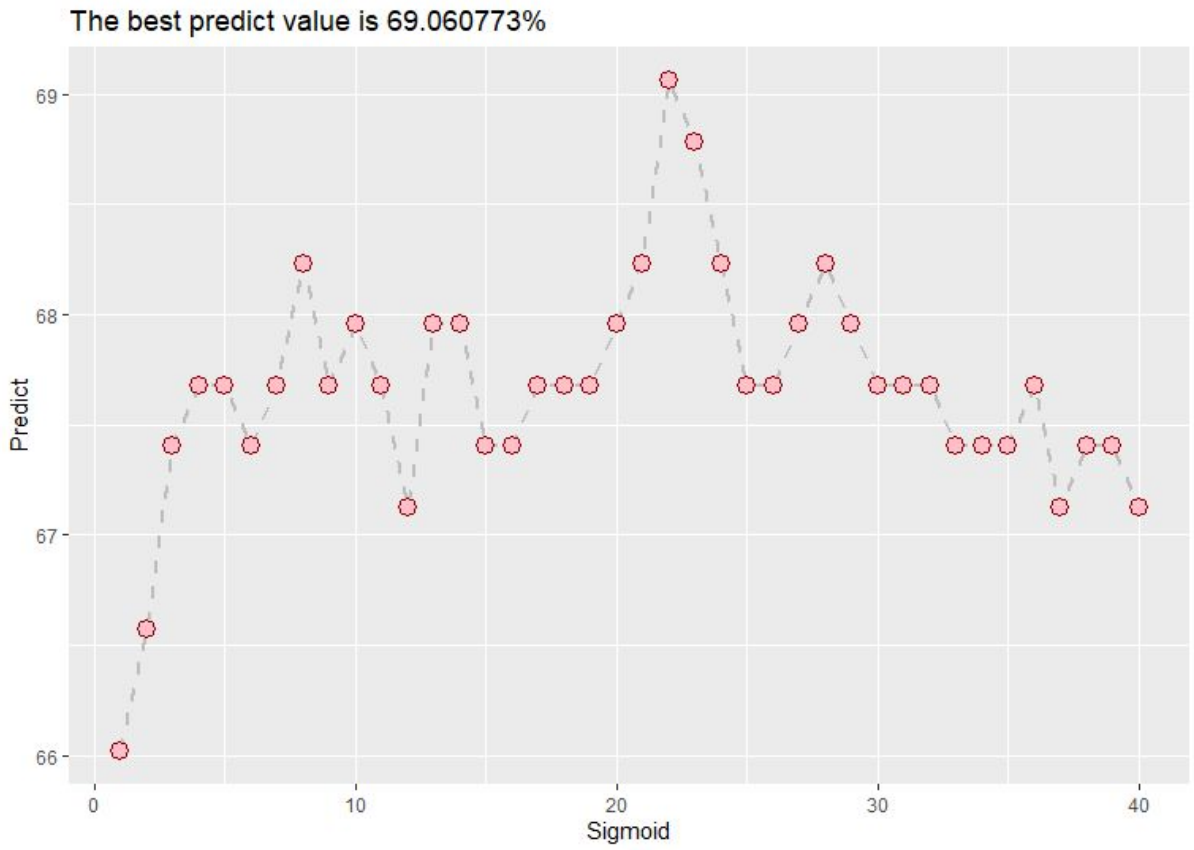
- **Predict the labels for a set of test examples.**

```
f_vector=function(w){
  if(f_sigmoid(w)>0.5){
    return(1.0) }
  else{ return(0) }
}
```

The predict value of ldl is 66.0221%.
Calculate the sigmoid value of rest 362 test examples by using β then plot the scatter distribution of sigmoid combine with test label:



In this plot, black indicates that sigmoid value corresponds to the label 0, and blue is the label 1. We found that most sigmoid values are clustered above 0.5. Then, to find a suitable value to define whether sigmoid is 1 or 0, we set a loop to get the best predicted value.



The best prediction value should be 69.06077 %.

Reference

[1] James, G., Witten, D., Hastie, T., & Tibshirani, R. (2013). *An introduction to statistical learning* (pp.131-133). New York: springer.
[2] Minka, T. P. (2003). A comparison of numerical optimizers for logistic regression. Unpublished draft, (pp..2).
[3] Baxter, J., Weaver, L., & Bartlett, P. L. (1999). Direct gradient-based reinforcement learning: II. Gradient ascent algorithms and experiments. preparation, July.
[4] Lander, J. P. (2014). R for everyone: Advanced analytics and graphics. Pearson Education.
[5] Kim, B. S., Park, S. G., You, Y. K., & Jung, S. I. (2011). Probability & Statistics for engineers & Scientists.