Chapter 4 Logistic Regression

Regression is the most commonly used statistical technique. Multiple linear regression model assume that $y = \beta_0 + \beta_1 x_1 + \cdots + \beta_p x_p + \varepsilon = x'\beta + \varepsilon$, where $\varepsilon \sim N(0, \sigma^2)$. The dependent (response) variable y is continuous. The independent (predictor) variables can be continuous, ordinal, nominal or even binary. In any case, the dataset should not contain outliers. In many situations, the dependent variable y is binary or nominal and then multiple linear regression model is no longer suitable. A commonly used statistical method to deal with binary response variable is the logistic regression model. When y is nominal and has more than two classes, the logistic regression model can be extended to multinomial logit. These models will be introduced in this chapter.

4.1 Logistic regression

First we define a parameter

$$\pi_i = \Pr\{Y_i = 1 \mid x_i\} \tag{4.1}$$

This parameter can be interpreted as the probability of $Y_i = 1$ (probability of "success") given x_i The crucial assumption in the logistic regression model is:

$$\ln[\pi_i /(1-\pi_i)] = \beta_0 + \beta_1 x_{i1} + \cdots + \beta_p x_{ip} = x'\beta \tag{4.2}$$

The left hand side of (4.2) is the **log-odd ratio** of probability of success. In the logistic regression we assume that the log-odd ratio of success probability is a linear function of x_i. Re-express (4.2) as

$$\pi_i = \exp(x_i'\beta)/[1+\exp(x_i'\beta)] = 1/[1+\exp(-x_i'\beta)]. \tag{4.3}$$

(4.3) known as the **logit** transformation of x_i Note that π_i is always lies between 0 and 1 and is consistent with the interpretation of success probability. The likelihood function is

$$L(\beta) = \prod_{i=1}^{n} [\pi_i^{y_i} (1-\pi_i)^{(1-y_i)}]. \tag{4.4}$$

or the log-likelihood function

$$\log L(\beta) = \sum_{i=1}^{n} [y_i \log \pi_i + (1-y_i)\log(1-\pi_i)] \tag{4.5}$$

If we have the maximum likelihood estimate of β, then we can compute the predict probability of success given x_i using (4.3). Let us illustrate this by the HMEQ example in Chapter 1 with all the outliers deleted.

```
> d<-read.csv("hmeq1.csv")     # read in dataset
> (n<-nrow(d))                 # get and display sample size
[1] 3067   13
> names(d)                     # display names of d
 [1] "BAD"     "LOAN"    "MORTDUE" "VALUE"   "REASON"  "JOB"     "YOJ"
 [8] "DEROG"   "DELINQ"  "CLAGE"   "NINQ"    "CLNO"    "DEBTINC"
```

First note that variables *REASON* and *JOB* are nominal variables contains characters. We have to exclude them from the logistic regression model.

Now we set the random seed and partition d into training and testing dataset as in Chapter 2.

```
> set.seed(123)
> r<-2/3                                    # set sampling ratio
> id<-sample(1:n,size=round(r*n),replace=F) # generate id
> d1<-d[id,]                                # training dataset
> d2<-d[-id,]                               # testing dataset
```

The logistic regression is implemented in *glm()* (stand for generalized linear model) function.

```
# apply glm, last option binomial means logistic reg.
> summary(glm(BAD~LOAN+MORTDUE+VALUE+YOJ+DEROG+DELINQ+CLAGE+NINQ+CLNO
    +DEBTINC,data=d1,binomial))

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-1.6500  -0.3831  -0.2754  -0.1979   3.2164

Coefficients:
             Estimate Std. Error z value Pr(>|z|)
(Intercept) -4.437e+00  5.599e-01  -7.924 2.30e-15 ***
LOAN        -1.671e-05  1.304e-05  -1.281   0.2002
MORTDUE     -3.054e-06  7.904e-06  -0.386   0.6992
VALUE        1.184e-06  6.694e-06   0.177   0.8597
YOJ         -3.250e-02  1.410e-02  -2.305   0.0212 *
DEROG        1.217e+00  1.966e-01   6.191 5.96e-10 ***
DELINQ       1.021e+00  1.124e-01   9.083  < 2e-16 ***
CLAGE       -6.305e-03  1.489e-03  -4.235 2.28e-05 ***
NINQ         2.551e-01  5.793e-02   4.403 1.07e-05 ***
CLNO         2.041e-03  1.091e-02   0.187   0.8515
DEBTINC      8.030e-02  1.291e-02   6.221 4.93e-10 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

In the *glm()* function, the last option binomial means that we are fitting the data using a logistic regression model. Obviously, not all the input variables are useful. By looking at the last column, *Pr(>|z|)*, we know that *VALUE* has the largest p-value (=0.8597). Therefore we fit the data again without the variable *VALUE*.

```
> summary(glm(BAD~LOAN+MORTDUE+YOJ+DEROG+DELINQ+CLAGE+NINQ+CLNO+DEBTINC,
    data=d1,binomial))       # fit the logistic reg. model without VALUE
> summary(lreg)              # display result summary
```

In this step, we found that *CLNO* has largest p-value (=0.8711) and should be excluded. Next, *MORTDUE* is excluded due to large p-value (=0.5687); and so on. We continue this process (known as backward elimination) to eliminate variable with largest p-value one by one, until all the variables has small p-value (say <0.1). Finally, we arrive at the following model:

```
> summary(glm(BAD~YOJ+DEROG+DELINQ+CLAGE+NINQ+DEBTINC,data=d1,binomial))
> lreg<-glm(BAD~YOJ+DEROG+DELINQ+CLAGE+NINQ+DEBTINC,data=d1,binomial) # save lreg

Call:
glm(formula = BAD ~ YOJ + DEROG + DELINQ + CLAGE + NINQ + DEBTINC,
    family = binomial, data = d1)

Coefficients:
             Estimate Std. Error z value Pr(>|z|)
(Intercept) -4.630497   0.535192  -8.652  < 2e-16 ***
YOJ         -0.033931   0.013903  -2.441   0.0147 *
DEROG        1.204119   0.194295   6.197 5.74e-10 ***
DELINQ       1.045063   0.110734   9.438  < 2e-16 ***
CLAGE       -0.006636   0.001409  -4.710 2.48e-06 ***
NINQ         0.244523   0.057199   4.275 1.91e-05 ***
DEBTINC      0.077702   0.012627   6.153 7.58e-10 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Note that all the p-values are less than 0.05. There are many items inside the object *lreg*. Among them, the fitted.values gives the probability of each record belonging to *BAD=1*. Therefore if fitted.values>0.5, we should predict that record to *BAD=1*.

```
> names(lreg)         # display items in lreg
 [1] "coefficients"      "residuals"         "fitted.values"
 [4] "effects"           "R"                 "rank"
 [7] "qr"                "family"            "linear.predictors"
[10] "deviance"          "aic"               "null.deviance"
[13] "iter"              "weights"           "prior.weights"
[16] "df.residual"       "df.null"           "y"
[19] "converged"         "boundary"          "model"
[22] "call"              "formula"           "terms"
[25] "data"              "offset"            "control"
[28] "method"            "contrasts"         "xlevels"

> pr<-lreg$fit>0.5    # pr=T if prob>0.5
> table(pr,d1$BAD)    # classification table

pr         0    1     # training error rate = (12+135)/2045 = 7.2%
  FALSE 1856  135
  TRUE    12   42
```

The training error rate is (12+135)/2045=7.2%. To apply this logistic model to *d2*, we first save the estimated coefficients to b.

```
> (b<-lreg$coef) # save and display the coefficient
 (Intercept)          YOJ        DEROG       DELINQ        CLAGE         NINQ
DEBTINC
-4.63049705  -0.03393086    1.20411916    1.04506255  -0.00663617    0.24452284
0.07770220
> names(d2)       # display names in d2
 [1] "BAD"     "LOAN"    "MORTDUE" "VALUE"   "reason"  "job"     "YOJ"
 [8] "DEROG"   "DELINQ"  "CLAGE"   "NINQ"    "CLNO"    "DEBTINC"
```

We have to created a suitable matrix x for computing the probability in (4.3). Note that we pick up the columns in *d2* that correspond to the variables in the logistic model. We add a column of ones in the front correspond to the intercept term in the logistic regression model. Now we are ready to compute the "success" probability

```
> x<-cbind(1,d2[,c(7,8,9,10,11,13)])   # select the suitable columns from d2
> b<-as.matrix(b)                      # convert b into matrix
> x<-as.matrix(x)                      # convert x into matrix
> pr<-exp(x%*%b)/(1+exp(x%*%b))        # compute pr using (4.3)
> pr<-pr>0.5                           # create label for prediction
> table(pr,d2$BAD)                     # classification table for d2
pr        0   1
  FALSE 912  68                  # prediction error rate = (9+68)/1022=7.5%
  TRUE    9  33
```

Note that the prediction error rate is (9+68)/1022=7.5%. In fact, we can produce this classification table using the built-in function `predict()` as well.

```
> pr<-predict(lreg,d2)
> prob<-exp(pr)/(1+exp(pr))    # compute pr using (4.3)
> cl<-prob>0.5                 # create label for prediction
> table(cl,d2$BAD)
cl        0   1
  FALSE 912  68
  TRUE    9  33
```

4.2 Dummy variables in logistic regression

Sometimes in logistic regression, as well as in ordinary linear regression model, nominal variables are included. Including these dummy variables in the regression model is valid as long as we know how to interpret it correctly. Let us illustrate it by the above example.

Among all the variables selected in the above logistic regression model, *DEROG* and *DELINQ* are the number of major derogatory report and the number delinquent trade lines. These are integers and of course we can treat them as continuous variables. Note that their coefficients are positive means that larger values will increase the chance of default. To look at the distribution of these two variables:

```
> table(d1$DEROG)      # dist. of DEROG
   0    1    2    3    4
1912  113   12    7    1

> table(d1$DELINQ)     # dist. of DELINQ
   0    1    2    3    4    5    6    7    8
1764  197   60    5    7    3    4    4    1
```

Now we convert them to binary variables by creating two new variables: *g1=0* if *DEROG=0* and *g1=1* otherwise; *h1=0* if *DELINQ=0* and *h1=1* otherwise.

```
> g1<-as.numeric(d1$DEROG>0)       # convert DEROG to a binary var. g1
> h1<-as.numeric(d1$DELINQ>0)      # convert DELINQ to a binary var. h1

> table(g1)                        # dist. of g1
g1
   0    1
1912  133

> table(h1)
h1                                 # dist. of h1
   0    1
1764  281
```

Now we use *g1* and *h1* instead of *DEROG* and *DELINQ* and their product terms with *YOJ, CLAGE, NINQ* and *DEBTINC* (this will be explained later).

```
> lreg<-glm(BAD~YOJ+CLAGE+NINQ+DEBTINC+g1+h1+g1*YOJ+g1*CLAGE+g1*NINQ+g1*DEBTINC
    +h1*YOJ+h1*CLAGE+h1*NINQ+h1*DEBTINC,data=d1,binomial)
> summary(lreg)
Call:
glm(formula = BAD ~ YOJ + CLAGE + NINQ + DEBTINC + g1 + h1 +
    g1 * YOJ + g1 * CLAGE + g1 * NINQ + g1 * DEBTINC + h1 * YOJ +
    h1 * CLAGE + h1 * NINQ + h1 * DEBTINC, family = binomial,
    data = d1)

Coefficients:
             Estimate Std. Error z value Pr(>|z|)
(Intercept) -5.4432393  0.6933686  -7.850 4.15e-15 ***
YOJ         -0.0340995  0.0187342  -1.820  0.06873 .
CLAGE       -0.0110050  0.0018267  -6.024 1.70e-09 ***
NINQ         0.1859500  0.0687522   2.705  0.00684 **
DEBTINC      0.1209502  0.0174646   6.925 4.35e-12 ***
g1           3.3894894  1.5048666   2.252  0.02430 *
h1           3.0001914  1.1529980   2.602  0.00927 **
YOJ:g1      -0.0389201  0.0313107  -1.243  0.21386
CLAGE:g1    -0.0005681  0.0038192  -0.149  0.88176
NINQ:g1     -0.3007556  0.1580892  -1.902  0.05711 .
DEBTINC:g1  -0.0400497  0.0319327  -1.254  0.20977
YOJ:h1       0.0412036  0.0275041   1.498  0.13411
CLAGE:h1     0.0115575  0.0027439   4.212 2.53e-05 ***
NINQ:h1      0.2930188  0.1450976   2.019  0.04344 *
DEBTINC:h1  -0.1098793  0.0276240  -3.978 6.96e-05 ***
```

Note that *g1*CLAGE* has the largest p-value and should be omitted in next step. We perform the backward elimination as discussed previously and arrive at the final model:

```
> lreg<-glm(BAD~CLAGE+NINQ+DEBTINC+g1+h1+h1*CLAGE+h1*DEBTINC,data=d1,binomial)
> summary(lreg)

Call:
glm(formula = BAD ~ CLAGE + NINQ + DEBTINC + g1 + h1 + h1 * CLAGE +
    h1 * DEBTINC, family = binomial, data = d1)

Coefficients:
             Estimate Std. Error z value Pr(>|z|)
(Intercept) -5.459265   0.657887  -8.298  < 2e-16 ***
CLAGE       -0.011418   0.001760  -6.488 8.68e-11 ***
NINQ         0.210005   0.054938   3.823 0.000132 ***
DEBTINC      0.114983   0.016559   6.944 3.81e-12 ***
g1           1.071226   0.241054   4.444 8.83e-06 ***
h1           3.688465   1.034916   3.564 0.000365 ***
CLAGE:h1     0.011400   0.002637   4.323 1.54e-05 ***
DEBTINC:h1  -0.110752   0.025842  -4.286 1.82e-05 ***


> pr<-lreg$fit>0.5
> table(pr,d1$BAD)

pr         0    1          # error rate = (1+54)/2045 = 7.6%
  FALSE 1867  154
  TRUE     1   23
```

The training error based on this model is (1+154)/2045=7.6%. Although training error is a little higher than the logistic model with the previous model but the interpretation of this model is simpler. Let π = probability of default, the logistic regression model is

$$\ln \pi/(1-\pi) = -5.459 - 0.0114 * CLAGE + 0.21 * NINQ + 0.115 * DEBTINC + 1.071 * g1 + 3.688 * h1 + 0.0114 * h1 * CLAGE - 0.111 * h1 * DEBTINC$$

Since g1 and h1 is binary, the model can be rewritten as:

$$\ln \pi/(1-\pi) = \begin{cases} -5.459 - 0.0114 * CLAGE + 0.21 * NINQ + 0.115 * DEBTINC & \text{if g1} = 0 \text{ and h1} = 0 \\ -4.388 - 0.0114 * CLAGE + 0.21 * NINQ + 0.115 * DEBTINC & \text{if g1} = 1 \text{ and h1} = 0 \\ -1.802 + 0.21 * NINQ + 0.004 * DEBTINC & \text{if g1} = 0 \text{ and h1} = 1 \\ -0.7 + 0.21 NINQ + 0.004 * DEBTINC & \text{if g1} = 1 \text{ and h1} = 1 \end{cases}$$

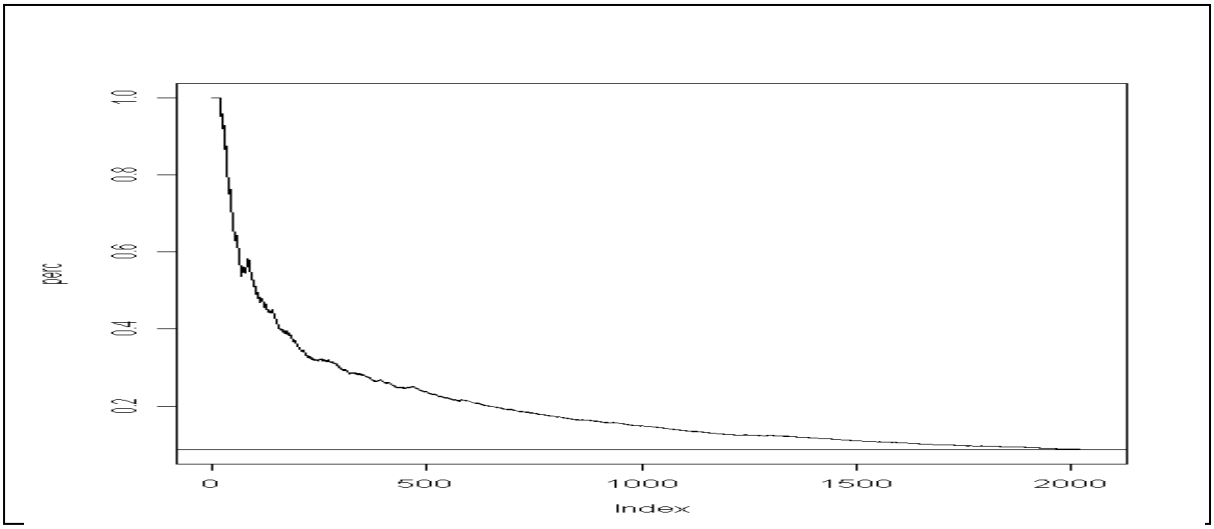
This is equivalent to fit logistic regression to these four groups separately.

4.3 Lift chart

The ROC curve described in Chapter 3 can be used to assess the performance of binary logistic regression. Another commonly used method is the lift chart. The fitted value in the logistic regression output (*lreg$fit*) gives the probability of "success", i.e. *Prob(Y=1|x)*. Now

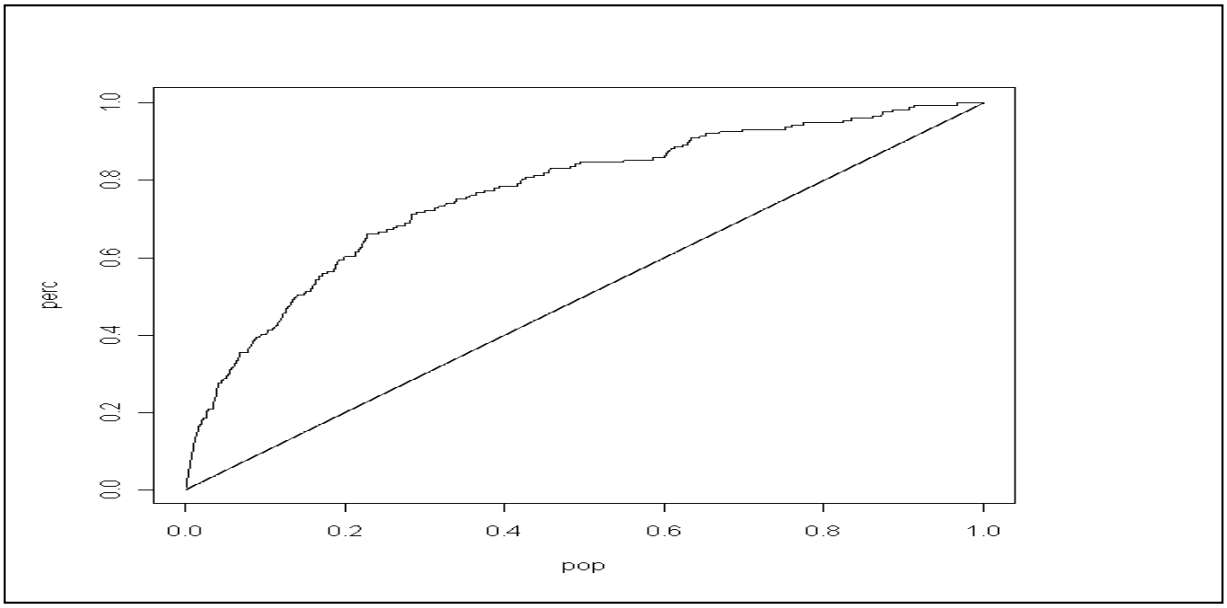
if we sort the variable *Y* (*d1$BAD* in our example) according to the decreasing order of *lreg$fit*, we would expect most of the records with *Y=1* will appear on the top of the list. We can actually compute the cumulative percentage of *Y=1* and plot against the index. This plot is called the lift chart. It measures how well this logistic regression model predicts. The following R commands will create this lift chart.

```
> ysort<-d1$BAD[order(lreg$fit,decreasing=T)]   # sort y according to lreg$fit
> n<-length(ysort)                              # get length of ysort
> perc1<-cumsum(ysort)/(1:n)                    # compute cumulative percentage
> plot(perc1,type="l")                          # plot perc with line type
> abline(h=sum(d1$BAD)/n)                       # add the baseline
```



Or we can plot this cumulative percentage of success with the reference line:

```
> perc2<-cumsum(ysort)/sum(ysort)       # cum perc of success
> pop<-(1:n)/n                          # x-coordinate
> plot(pop,perc2,type="l")              # plot
> lines(pop,pop)                        # add the reference line
```



4.4 Model Selection

In ordinary linear regression and binary logistic regression, selecting useful and important variables included in the model is an important issue. As in section 4.1, start with all the variables included in the model and eliminate variable with largest p-value one-by-one is called the **backward elimination**. There are other alternative for model selection. One commonly used method is the **stepwise** regression procedure. In each step, variable with largest p-value will be removed from the model and variable with smallest p-value will be entered in the model. An **Akaike Information Criterion (*AIC*)** is also computed in each step to help us to choose the suitable model, where $AIC = -2\log L + k * edf$. *Log L* is the log-likelihood function value; *edf* is the effective degrees of freedom (i.e., the number of free parameters in the model); *k=2* for *AIC* and *k=log(n)* for **Bayesian Information Criterion (*BIC*)**. The *step()* function implements the stepwise regression. Let use the "*hmeq1.csv*" to illustrate this *step()* function.

```
>lreg<-glm(BAD~LOAN+MORTDUE+VALUE+YOJ+DEROG+DELINQ+CLAGE+NINQ+CLNO+DEBTINC,
data=d1,binomial)                                  # logistic regression
> step(lreg)                                       # perform stepwise selection
Start:  AIC=937.03
BAD ~ LOAN + MORTDUE + VALUE + YOJ + DEROG + DELINQ + CLAGE +
    NINQ + CLNO + DEBTINC
          Df Deviance     AIC
- VALUE    1   915.06  935.06
- CLNO     1   915.06  935.06
- MORTDUE  1   915.18  935.18
- LOAN     1   916.72  936.72
<none>         915.03  937.03
- YOJ      1   920.56  940.56
- NINQ     1   933.29  953.29
- CLAGE    1   934.64  954.64
- DEROG    1   954.56  974.56
- DEBTINC  1   954.62  974.62
- DELINQ   1  1020.59 1040.59

Step:  AIC=931.41
BAD ~ LOAN + YOJ + DEROG + DELINQ + CLAGE + NINQ + DEBTINC
          Df Deviance     AIC
<none>         915.41  931.41
- LOAN     1   917.94  931.94
- YOJ      1   920.67  934.67
- NINQ     1   933.73  947.73
- CLAGE    1   937.77  951.77
- DEROG    1   955.52  969.52
- DEBTINC  1   955.58  969.58
- DELINQ   1  1025.45 1039.45

Call:  glm(formula = BAD ~ LOAN + YOJ + DEROG + DELINQ + CLAGE + NINQ +
    DEBTINC, family = binomial, data = d1)
```

Note that the final model (with smallest AIC) is different from the model in section 4.1 selected by the backward elimination.

4.5 Multinomial logit

The binary logistic regression model can be extended to the case where the response *Y* is a categorical variable with more than 2 groups. Suppose *Y* has k categories and we modeled the probabilities as follows:

$\Pr(Y_i = 1 \mid x_i\} = \exp(x_i'\beta_1)/\xi \quad ,\ldots, \quad \Pr(Y_i = k \mid x_i\} = \exp(x_i'\beta_k)/\xi$

where $\xi = \exp(x_i'\beta_1) + \ldots + \exp(x_i'\beta_k)$.

However, the $\beta_1,...,\beta_k$ in the above model cannot be uniquely estimated unless one of these betas are set to zero. It doesn't matter which one is set to zero. Suppose we set β_1 to zero, then the model becomes:

$\Pr(Y_i = 1 \mid x_i\} = 1/\xi, \quad \Pr(Y_i = 2 \mid x_i\} = \exp(x_i'\beta_2)/\xi \ ,\ldots, \ \Pr(Y_i = k \mid x_i\} = \exp(x_i'\beta_k)/\xi$

where $\xi = 1 + \exp(x_i'\beta_2) + \ldots + \exp(x_i'\beta_k)$. (4.4)

The group *Y=1* serve as the baseline in this MNL model. R has a built-in function *multinom()* in the *nnet* (stand for neural network) library to estimate the parameters in this MNL model. Let us illustrate this by the Iris flower data again. The first four columns are the measurement of Sepal length, Sepal width, Petal length and Petal width respectively. The last column indicates three different species of the Iris flower (1= *setosa, 2=versicolor* 3= *virginica*). There are 150 observations and 50 observations from each species.

```
> d<-read.csv("iris.csv")      # read in data
> names(d)                     # display names
[1] "Sepal_len" "Sepal_wid" "Petal_len" "Petal_wid" "Species"
> library(nnet)                # load nnet library
> mnl<-multinom(Species~Sepal_len+Sepal_wid+Petal_len+Petal_wid,data=d)   # MNL
> summary(mnl)                 # display summary of MNL
Call:
multinom(formula = Species ~ Sepal_len + Sepal_wid + Petal_len +
    Petal_wid, data = d)
Coefficients:
  (Intercept) Sepal_len Sepal_wid Petal_len Petal_wid
2    18.69037 -5.458424  -8.70740  14.24477 -3.097684
3   -23.83628 -7.923634 -15.37077  23.65978 15.135301
Std. Errors:
  (Intercept) Sepal_len Sepal_wid Petal_len Petal_wid
2    34.97116  89.89215  157.0415  60.19170  45.48852
3    35.76649  89.91153  157.1196  60.46753  45.93406
```

We can estimate $\Pr\{Y_i = k \mid x_i\}$ according to (4.4) and predict Y belonging to group j if $\Pr\{Y_i = j \mid x_i\}$ is maximum. R has a built-in function *predict()* to obtain these predictions.

```
> pred<-predict(mnl)       # prediction
table(pred,d$Species)      # tabulate pred vs true species
 pred 1  2  3
   1 50  0  0
   2  0 49  1
   3  0  1 49              # error rate = (1+1)/150 = 1.33%
```

From the above output, there are only two misclassification cases and the misclassification rate is 2/150=1.33%.

4.6 Logistic regression using EXCEL

In statistical package, the MLE of logistic regression is obtained by maximizing the log-likelihood function (4.5) by numerical methods. However, it is possible to obtain MLE by the *solver* function in EXCEL.

1. Select the variables: BAD, YOJ, DEROG, DELINQ, CLAGE, NINQ, DEBTINC and save them in columns B to H respectively.
2. We can use EXCEL's built-in uniform random number generator to partition the dataset into training and testing dataset. Use the built-in random number generator to generate 3067 Uniform(0,1) random numbers and save them in column I. Sort the whole dataset in ascending order according to column I. The first 2045 records are considered as training data while the rest is testing data.
3. Set the beta in O2:O5 to zero as initial values.
4. Input the formula `=$O$2+MMULT(C2:H2,$O$3:$O$8)` in K2 and copy it down to K3068.
5. Input the formula `=EXP(K2)/(1+EXP(K2))` in L2 and copy it down to L3068.
6. Input the formula `=B2*LN(L2)+(1-B2)*LN(1-L2)` in M2 and copy it down to M3068.
7. Enter `=SUM(M2:M2046)` in O11. Using solver to maximize this cell with O2:O8 as variable cells. Note that we use the training data (M2 to M2046) for estimating beta.
8. Enter `=0+(L2>0.5)` in A2 and copy it down to A3068. This is the prediction based on the logistic regression.
9. Finally we can produce the classification table for training data by entering

```
P15: =COUNTIFS(A2:A2046,"=0",B2:B2046,"=0")
P16: =COUNTIFS(A2:A2046,"=1",B2:B2046,"=0")
Q15: =COUNTIFS(A2:A2046,"=0",B2:B2046,"=1")
Q16: =COUNTIFS(A2:A2046,"=1",B2:B2046,"=1")
```

Similarly, we can produce the classification table for testing data by entering

```
P20: =COUNTIFS(A2047:A3068,"=0",B2047:B3068,"=0")
P21: =COUNTIFS(A2047:A3068,"=1",B2047:B3068,"=0")
Q20: =COUNTIFS(A2047:A3068,"=0",B2047:B3068,"=1")
Q21: =COUNTIFS(A2047:A3068,"=1",B2047:B3068,"=1")
```