RMSC4002 Tutorial 9

Chapter 5-6

November 21, 2017

1 Dummy Variable in Logistic Regression

```
> g<-(d3$ln_MV>9.4766)+1  # create dummy var g=2 if ln_MV>9.4766; g=1
otherwise
>summary(glm(HSI~EY+CFTP+g+DY+BTME+DTE+EY*g+CFTP*g+DY*g+BTME*g+DTE*g,
data=d3, binomial))

Call:
glm(formula = HSI ~ EY + CFTP + g + DY + BTME + DTE + EY * g +
    CFTP * g + DY * g + BTME * g + DTE * g, family = binomial,
    data = d3)
```

```
Coefficients:
            Estimate Std. Error z value Pr(>|z|)
(Intercept) -21.0469     5.6643  -3.716 0.000203 ***
EY           29.0955    38.2277   0.761 0.446592
CFTP         -0.1916     3.8931  -0.049 0.960745
g            15.2686     5.3888   2.833 0.004606 **
DY            1.0979     0.8182   1.342 0.179637
BTME          1.4157     1.5673   0.903 0.366398
DTE          -0.3659     0.8731  -0.419 0.675166
EY:g        -28.8939    38.2187  -0.756 0.449641
CFTP:g        0.1269     3.6225   0.035 0.972054
g:DY         -0.9840     0.7821  -1.258 0.208316
g:BTME       -1.3220     1.4948  -0.884 0.376499
g:DTE         0.2058     0.5665   0.363 0.716429
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
> summary(glm(HSI~EY+g+DY+BTME+DTE+EY*g+DY*g+BTME*g+DTE*g,data=d3,
binomial))

Call:
glm(formula = HSI ~ EY + g + DY + BTME + DTE + EY * g + DY *
    g + BTME * g + DTE * g, family = binomial, data = d3)

Coefficients:
            Estimate Std. Error z value Pr(>|z|)
(Intercept) -21.0169     5.5409  -3.793 0.000149 ***
EY           28.6141    27.2787   1.049 0.294200
g            15.2425     5.2592   2.898 0.003752 **
DY            1.0968     0.7752   1.415 0.157089
BTME          1.4089     1.4991   0.940 0.347279
DTE          -0.3558     0.8474  -0.420 0.674536
EY:g        -28.4191    27.2658  -1.042 0.297274
g:DY         -0.9858     0.7396  -1.333 0.182540
g:BTME       -1.3146     1.4225  -0.924 0.355404
g:DTE         0.2015     0.5513   0.366 0.714692
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
> summary(glm(HSI~g+DY+DY*g,data=d3,binomial))

Call:
glm(formula = HSI ~ g + DY + DY * g, family = binomial, data = d3)
```

```
Coefficients:
             Estimate Std. Error z value Pr(>|z|)
(Intercept) -18.6304     3.4117  -5.461 4.74e-08 ***
g            12.9236     3.1251   4.135 3.54e-05 ***
DY            1.4036     0.6485   2.165   0.0304 *
g:DY         -1.2860     0.6082  -2.115   0.0345 *
---
Signif. codes:  0 `***' 0.001 `**' 0.01 `*' 0.05 `.' 0.1 ` ' 1
```

```
> lreg1<-glm(HSI~g+DY+DY*g,data=d3,binomial)
> pr<-(lreg1$fit>0.5)
> table(pr,d3$HSI)
  pr      0   1
  FALSE 624   3
  TRUE    2  29
```

The interpretation of this threshold-type logistic model is as follow:

$$\ln[\pi/(1-\pi)] = -18.6304 + 12.9236g + 1.4036DY - 1.286(g*DY)$$

$$\Leftrightarrow \begin{cases} \ln[\pi/(1-\pi)] = (-18.6304+12.936)+(1.4036-1.286)DY, & \text{for } g=1 \\ \qquad = -5.6944+0.1176DY & \\ \ln[\pi/(1-\pi)] = (-18.6304+12.9236\times 2)+(1.4036-1.286\times 2)DY, & \text{for } g=2 \\ \qquad = 7.2416-1.1684DY & \end{cases}$$

where π= probability of HSI=1. Note that for g=2, the intercept is much bigger than that of g=1, hence the probability of HSI=1 is higher.

2 Classification Tree

```
> d<-read.csv("fin-ratio.csv")                       # read in data in csv format
> library(rpart)                                     # load rpart library
> ctree<-rpart(HSI~EY+CFTP+ln_MV+DY+BTME+DTE,data=d,method="class")
> plot(ctree,asp=0.5,main=" Fin-ratio" )     # plot ctree
> text(ctree,use.n=T,cex=0.6)                        # add text
```

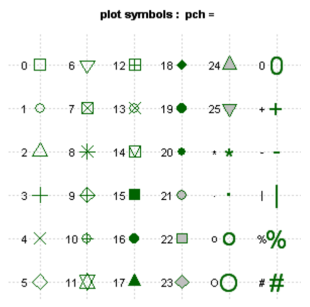
```
> print(ctree)
n= 680
node), split, n, loss, yval, (yprob)
      * denotes terminal node
1) root 680 32 0 (0.952941176 0.047058824)
  2) d$ln_MV< 9.4776 647  3 0 (0.995363215 0.004636785) *
  3) d$ln_MV>=9.4776 33  4 1 (0.121212121 0.878787879) *
```

```
# plot ln_MV versus HSI with color, red=group 0 and blue=group 1
> plot(d$HSI,d$ln_MV,pch=21,bg=c("red","blue")[d$HSI+1])
> abline(h=9.478)       # add a horizontal line at y=9.478
> pr<-predict(ctree)              # pr has 2 columns of prob. in group 0 or 1
> cl<-0*(pr[,1]>0.5)+1*(pr[,2]>0.5)    # assign group label if prob>0.5
> table(cl,d$HSI)                 # cross tabluation

cl   0   1
  0 644   3
  1   4  29
```

Remarks:

- "asp": numeric, giving the aspect ratio y/x.
- "text": labels the current plot of the tree dendrogram with text.
- "use.n": logical, if true, adds to label (#event level 1/#event level 1/etc.)
- "cex": number indicating the amount by which plotting text and symbols should be scaled relative to the default. 1 is default; 1.5 is 50% larger; 0.5 is 50% smaller, etc.
- "pch": specifying symbols to use when plotting points as shown below. For symbols 21 through 25, specify border color (col=) and fill color (bg=).



```
> d<-read.csv("iris.csv")
> library(rpart)
> names(d)
[1] "Sepal_len" "Sepal_wid" "Petal_len" "Petal_wid" "Species"
> ctree<-rpart(Species~Sepal_len+Sepal_wid+Petal_len+Petal_wid,data=d,method="class")
> plot(ctree,asp=2)
> text(ctree,use.n=T,cex=0.6)
```

```
> print(ctree)
n= 150

node), split, n, loss, yval, (yprob)
      * denotes terminal node

1) root 150 100 1 (0.33333333 0.33333333 0.33333333)
  2) Petal_len< 2.45 50   0 1 (1.00000000 0.00000000 0.00000000) *
  3) Petal_len>=2.45 100  50 2 (0.00000000 0.50000000 0.50000000)
    6) Petal_wid< 1.75 54   5 2 (0.00000000 0.90740741 0.09259259) *
    7) Petal_wid>=1.75 46   1 3 (0.00000000 0.02173913 0.97826087) *
```

```
# plot Petal_wid versus Petal_len with different color for each species
> plot(d$Petal_len,d$Petal_wid,pch=21,bg=c("red","blue","green")[d$Species])
> abline(h=1.75)        # add a horizontal line
> abline(v=2.45)        # add a vertical line
```

```
> pr<-predict(ctree)                  # pr has 3 columns of prob.
> cl<-max.col(pr)                 # cl is the col. no. such that pr is the maximum.
> table(cl,d$Species)                 # cross tabulation table

cl  1  2  3
  1 50  0  0
  2  0 49  5
  3  0  1 45
```

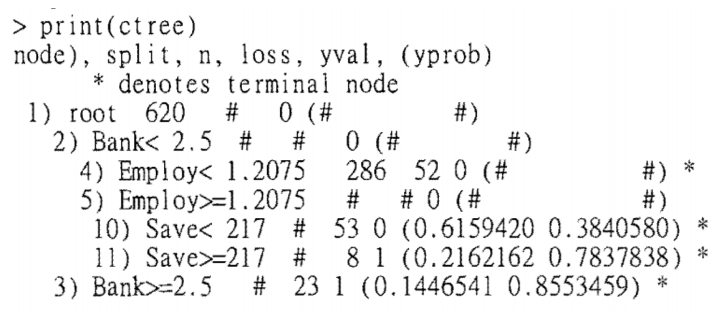
Remarks:

- "max.col": Find the maximum position for each row of a matrix.

3 Tree Quality assessment

- Support = number of cases in that rule / total number of data in dataset
- Confidence = number of correct-classified data in the node / number of data at the node
- Capture = number of correct-classified data in the node / number of data of this class in the whole dataset

Exercise 2014-15 final Q5 Classification tree is built using *rpart*() and the result is



```
> print(ctree)
node), split, n, loss, yval, (yprob)
      * denotes terminal node
 1) root  620   #   0 (#         #)
   2) Bank< 2.5  #   #   0 (#        #)
     4) Employ< 1.2075   286  52 0 (#          #) *
     5) Employ>=1.2075   #   # 0 (#          #)
      10) Save< 217  #  53 0 (0.6159420 0.3840580) *
      11) Save>=217  #   8 1 (0.2162162 0.7837838) *
   3) Bank>=2.5   #  23 1 (0.1446541 0.8553459) *
```

Some numbers are missing (denoted by #), please fill in these numbers.

- Find out all the missing numbers.
- Produce the classification table from this output and compute the error rate.
- Consider the terminal node 4). Compute support, confidence and capture.

4 Building Classification Tree

4.1 Impurity Measure

There are three measures of the impurity of the C-Tree:

- $Entropy(t) = -\sum_{i=0}^{c-1} p(i|t)\log_2 p(i|t)$ with $0\log_2 0 = 0$
- $Gini(t) = 1 - \sum_{i=0}^{c-1} p(i|t)^2$
- $Classificationerror(t) = 1 - max_i[p(i|t)]$

Since the objective of C-tree is to find the rule such that the terminal nodes are as "pure" as possible, so for every splitting, we would like to find a variable v such that the goodness of splitting variable v is the largest.

4.2 Algorithm

The main idea to build a classification tree is to find the attribute that minimize entropy at one step, then after we splitting the dataset into several sub datasets, we find the best attributes for each sub dataset. Then rerun this procedure recursively.

Algorithm 1: Classify

```
Input: Data
Output: Classification tree
Initialize tree
Find the best attribute that minimize the entropy.
foreach Sub dataset do
    subtree=Classify(sub dataset)
    Append subtree on tree.
end
Return tree
```