

MA554 Multivariate Analysis Project

Erik Erhardt, Gang Shen and Ning Liu

June 10, 2003

Contents

1	Introduction	2
2	Data Description	3
2.1	Story	3
2.2	Variable reduction	3
2.3	Transformations to Normality	4
2.4	Data Cut	4
2.5	Brute-force	4
3	Rules	6
3.1	Definitions and Background	6
3.2	Equal costs, proportionate priors	8
3.3	Weighted costs	9
3.4	Comparison on Test data set	12
4	Appendix	13

1 Introduction

Discrimination and classification are concerned with separating objects from different populations into different groups and with allocating new observations to one of these groups. Thus, the first goal is (1) to describe (graphically or algebraically) the difference between objects from several known populations. We construct discriminants that have numerical values which separate the different collections as much as possible. The second goal is (2) to assign objects into several labeled classes.

A good procedure should result in as few misclassifications as possible, thus minimizing expected cost of misclassification (ECM) over the whole random variable domain. It should take into account the likelihood of objects to belong to each of the classes (prior probability of occurrence). One often also takes into account the costs of misclassification.

We apply these techniques to tree census data in an attempt to classify trees by mortality. That is, we wish to classify a tree into ‘alive’ (mort=0) or ‘dead’ (mort=1) based on the characteristics of the local neighborhood of each tree.

Further, we develop a method to improve classification by combining different rules. Thus we can get rules based on the Linear Discrimination Function (LDF), Quadratic Discrimination Function (QDF) and Nonparametric (NPAR) rules based on different numbers of nearest neighbors for classification. After defining a number of rules, we show how they can work together to get a better classification result.

First we introduce a method based on equal costs. Next we introduce a similar method based on weighted costs. We find that classification can be improved using both methods. However, allocation of new observations from our test data based on rules from our training data is very poor. We attribute this difficulty to the dataset itself.

2 Data Description

2.1 Story

The data for this project comes from original research for a paper¹ by Tim Parshall, PhD., a friend of Erik's, while performing post-PhD. research for Harvard University.

We use a five-year census of mapped plots of trees originally used to test one of the mechanisms that could enhance the forest mosaic at Sylvania.

Spatial patterning of plants is common in most ecosystems and may be the result of a variety of driving factors including disturbances, edaphic characteristics, and plant interactions. The importance of plant interactions in spatial patterning has been discussed for decades but is one of the most difficult to prove. Plants interact primarily within neighborhoods, where the strength and nature of the interaction leads to the type of landscape pattern that develops. Neighborhood interactions with a negative effect on conspecifics will enhance diversity, while neighborhood interactions with positive effect on conspecifics promote self-replacement, resulting in patches dominated by a single or few species. When the interaction between two dominant species are symmetrical, either reciprocal replacement or a vegetation mosaic can occur depending on the nature of the interaction.

2.2 Variable reduction

Our original data set consists of 30 variables, listed in the Appendix. We were able to easily reduce the number of variables to 15 since many of the variables describe the same data in different ways. We excluded the *Total* and *Ratio* variables, and used *crown* instead of *canopy*. Then we coded the variables so they would all be numeric.

Comments on the data Although we have 4655 observations in our training data, only 242 of these are from population 1 ($mort=1$), or almost exactly 5%. Therefore, any rule we define is almost surely to over classify into population 0 ($mort=0$) since the likelihood that an observation is from population 0 is 95%.

¹“Does differential mortality contribute to neighborhood pattern in hemlock-hardwood forest?” by Parshall, Davis, Calcote, Walker, Douglas, not yet published

The distributions of the original data are given by the plot in Figure 1 on page 5.

2.3 Transformations to Normality

We performed Box-Cox transformations on each of the variables. For almost all of the variables, transformation to normality was an impossibility. Therefore, we include results based on the natural data only.

2.4 Data Cut

So that we can perform classification on a data set similar to but separate from the data we train our classification rules on, we cut our data. The entire data set has 9155 observations. In SAS we create a column of random numbers and sort the observations by those random numbers. We keep the first 4655 observations for our training data and designate the remaining 4500 observations for our test data.

2.5 Brute-force

With 15 variables, it is now reasonable to perform Brute-force on all variable combinations for a given classification rule. SAS Proc Discrim was run in many ways (QDF, LDF, NPAR for various k 's) and we selected the "best" few variable combinations based on lowest misclassification error with respect of resubstitution method and/or highest correct classification into population 1.

We found that the recommended k -values for the NPAR method ($n^{2/8} < k < n^{3/8}$) did not work well at all. We tried $k=9$, 15 and 23. We suspect this is due to the population priors being very bias toward population 0 (95% pop 0, 5% pop 1). We eventually used $k=3$ and $k=5$.

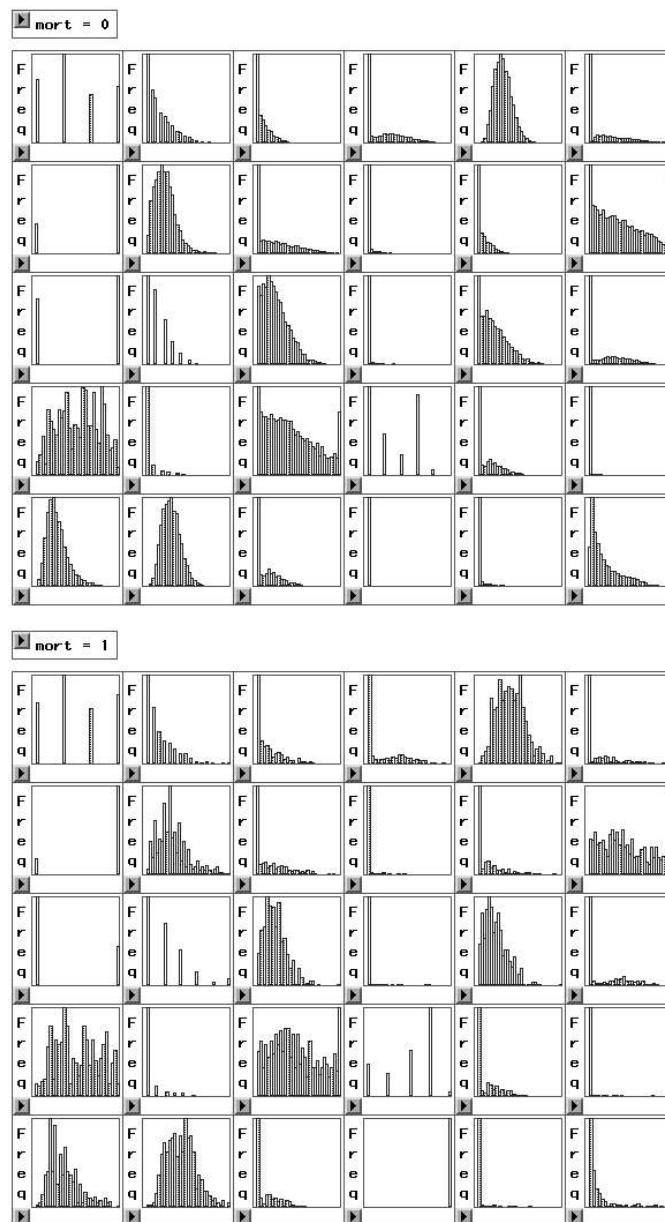


Figure 1: Original data distributions by mortality (population indicator)

3 Rules

We divide the development and usage of rules into these parts:

1. Definitions and Background.
2. Equal costs and proportionate priors, that is, $c(1|0) = c(0|1)$.
3. Weighted costs, that is $c(1|0)p(0) = c(0|1)p(1)$.
4. Include Weighted cost rule into Equal cost rule for comparison on Test data set.

3.1 Definitions and Background

A general Confusion matrix is given in Table 1 on page 6.

		Into	
	pop	0	1
From	0	A	B
	1	C	D

Table 1: General Confusion Matrix

$$\text{Prior for population "0": } P(0) = \frac{A + B}{A + B + C + D} \quad (1)$$

$$\text{Prior for population "1": } P(1) = \frac{C + D}{A + B + C + D} \quad (2)$$

$$\text{Posterior misclassification from "0" into "1": } P(1|0) = \frac{B}{A + B} \quad (3)$$

$$\text{Posterior misclassification from "1" into "0": } P(0|1) = \frac{C}{C + D} \quad (4)$$

$$\text{Conditional ECM from "0" into "1": } ECM(1|0) = P(0)P(1|0)C(1|0) \quad (5)$$

$$\text{Conditional ECM from "1" into "0": } ECM(0|1) = P(1)P(0|1)C(0|1) \quad (6)$$

Here we show how we improve classification by combining rules and taking the one with the least ECM.

$$ECM_{\Omega} = \sum_{i=1}^n ECM_k$$

So the ECM_{Ω}^k is sum over the observations for the k^{th} rule over Ω , the sample space. The conventional method is to take the minimum of the ECMs for the rule. However, it can be easily shown that one can improve by taking the minimum for each observation. That is,

$$\begin{aligned} ECM(\star) &= \sum_{i=1}^n \min(ECM_{i1}, ECM_{i2}, \dots, ECM_{ik}) \\ &\leq \min\left(\sum_{i=1}^n ECM_{i1}, \sum_{i=1}^n ECM_{i2}, \dots, \sum_{i=1}^n ECM_{ik}\right) \end{aligned}$$

3.2 Equal costs, proportionate priors

In our first part, working on the training data set, and with the six classification rules, we want to derive a new but better classification rule. Below is the procedure that how we got the new rule. (we need to describe this much better, sorry.)

In Table 3 on page 9 are the results by the existing six rules for your reference.

We assume all the conditional costs are equal. Therefore the Apparent Error Rate (APER) may be the appropriate measure of whether our new rule is better than other existing rules. In this case we still need to compare, on each particular observation, the conditional ECM (actually the priors \times posteriors) from all the rules, then select the minimum of them. We will calssify observation into the population instructed by the rule which gives the minimum ECM. Here are the results by our new rule again.

		Into	
		0	1
From	0	4356	57
	1	146	96

Table 2: Equal Costs Matrix

Now we can see the total number of error is 203, better than that of any existing rules:

$$\begin{aligned}
 APER &= \frac{57 + 146}{4356 + 57 + 146 + 96} \\
 &= 4.36\%
 \end{aligned}
 \tag{7}$$

It is possible to invite rules that provide no marginal benefit to the ECM. This phenomenon is analagous to inviting the uncles to the wedding. Therefore, we not only need to discriminate among factors, but also among individual rules.

Rule	From 0 into 1	From 1 into 0	APER	
Quadratics	212	217	9.22%	
NPar k=3	18	189	4.45%	minimum
NPar k=3	22	190	4.51%	
NPar k=3	20	190	4.51%	
NPar k=5	13	217	4.94%	
NPar k=5	9	216	4.83%	

Table 3: Equal Costs Misclassification comparisons

3.3 Weighted costs

Here the idea is similar to equal costs, but here we assume the costs $C(1|0)$ and $C(0|1)$ are purposely selected so that $P(0)C(1|0) = P(1)C(0|1)$. This effectively cancels the prior effect on the classification rule with the conditional costs.

Again, we employ six rules to work in classification. So for any particular observation, we have six judgements (responses), once from each of the six rules. We calculate the conditional ECM for each rule based on the judgement (response) it gives. Then we generate our “new” rule by taking the same judgement as the rule among the six that has the minimal ECM. This is the same as choosing the one which has the minimal posterior.

The results of our new rule, in the form of a Cost-Weighted Confusion matrix is given in Table 4 on page 9.

		Into	
		0	1
From	0	4144	267
	1	132	110

Table 4: Cost-Weighted Confusion Matrix

At first look, it gives even worse APER than the best rule (Non-Parametric, k=3). However, as the conditional costs here are not equal, the apparent error rate

$$APER = \frac{B + C}{A + B + C + D} \quad (8)$$

is not appropriate here.

This is because the APER “automatically” treats the conditional costs $C(1|0)$ and $C(0|1)$ as equal.

$$\begin{aligned} APER &= \frac{A+B}{A+B+C+D} \times \frac{B}{A+B} + \frac{C+D}{A+B+C+D} \times \frac{C}{C+D} \quad (9) \\ &= P(0) \times P(1|0) + P(1) \times P(0|1) \end{aligned}$$

However, our new rule is based on the assumption that they are not equal. This motivates the need to develop a Cost-Weighted Error Rate (CWER) to account for the error rate.

$$CWER \propto ECM(1|0) + ECM(0|1)$$

Here, let us choose

$$\begin{aligned} C(1|0) &= \frac{1}{2} \times P(1) \\ C(0|1) &= \frac{1}{2} \times P(0) \end{aligned} \quad (10)$$

which simplifies CWER as

$$CWER = \frac{1}{2} \times P(1|0) + \frac{1}{2} \times P(0|1). \quad (11)$$

This choice of cost ensure the CWER is between 0 and 1.

Under the measure of CWER, the result from our new rule is the best:

$$\frac{1}{2} \times \frac{132}{242} + \frac{1}{2} \times \frac{267}{4413} = 30.30\%$$

as compared to the CWER for the other six rules.

$$\begin{aligned}
 \text{Quadratic:} & \quad \frac{1}{2} \times \frac{217}{242} + \frac{1}{2} \times \frac{212}{4413} = 47.24\% \quad (12) \\
 \text{Non-Parametric with k=3:} & \quad \frac{1}{2} \times \frac{189}{242} + \frac{1}{2} \times \frac{18}{4413} = 39.25\% \\
 \text{Non-Parametric with k=3:} & \quad \frac{1}{2} \times \frac{190}{242} + \frac{1}{2} \times \frac{22}{4413} = 39.50\% \\
 \text{Non-Parametric with k=3:} & \quad \frac{1}{2} \times \frac{190}{242} + \frac{1}{2} \times \frac{20}{4413} = 39.50\% \\
 \text{Non-Parametric with k=5:} & \quad \frac{1}{2} \times \frac{217}{242} + \frac{1}{2} \times \frac{13}{4413} = 47.52\% \\
 \text{Non-Parametric with k=5:} & \quad \frac{1}{2} \times \frac{216}{242} + \frac{1}{2} \times \frac{9}{4413} = 44.73\%
 \end{aligned}$$

The error rate defined as CWER appears very large compared to the APER in the previous section. This is because the costs are weighted proportionately to the priors. Since classification into population 1 (mort=1) was very poor and had a small prior of 5%, the error rate is inflated to reflect this.

However, it seems somewhat “unfair” to let the rules to do classification at equal cost first, then to measure their “performance” by the cost-weighted error rate at last. So we re-did the test. This time under premise of $P(0) \times P(1|0) = P(1) \times P(0|1)$, We set equal priors for each rule so that it will be treated with equal cost in SAS. Then we continued our procedure to obtain our new rule. Below is the exciting result:

Please be noted that APER is now applicable again.

Training Data	Counts		Posteriors		APER
	$p(0 1)$	$p(1 0)$	$p(0 1)$	$p(1 0)$	APER
Method	96	1511	0.396694	0.342397	0.345220
QDF	0	242	0.000000	0.054838	0.051987
NPAR k=3	0	441	0.000000	0.099932	0.094737
NPAR k=3	0	394	0.000000	0.089282	0.084640
NPAR k=5	0	763	0.000000	0.172898	0.163910
NPAR k=5	4	972	0.016529	0.220258	0.209667
New Rule	4	73	0.016529	0.016542	0.016541
Total	of 242	of 4413			

Table 5: Training data vs Test Data Posterior

3.4 Comparison on Test data set

Lastly, we apply our new rule obtained in the second part to our test data set. We assume all the posteriors remain unchanged. That is, the posteriors in the test data are the same as those in the training data. However, after a careful check of the posteriors in the test data, we found this assumption does not hold at all. So it is no wonder the new rule does not work well in the test data.

This dilemma of unequal posteriors may be the result of high bias toward population 0 (trees that did not die in 5 years). There is not enough information in the data to clearly separate the populations. It is worth investigating how these rules perform on data where the population proportions are roughly equal.

In Table 6 on page 13 we provide a comparison of classification under different rules for the training data and test data.

We should include our best rule in this table

Training Data	Counts		Posteriors	
	$p(0 1)$	$p(1 0)$	$p(0 1)$	$p(1 0)$
Method				
QDF	217	212	0.896694215	0.048039882
NPAR k=3	189	18	0.780991736	0.004078858
NPAR k=3	190	22	0.785123967	0.004985271
NPAR k=3	190	20	0.785123967	0.004532064
NPAR k=5	217	13	0.896694215	0.002945842
NPAR k=5	216	9	0.892561983	0.002039429
Total	of 242	of 4413		
Test Data	Counts		Posteriors	
	$p(0 1)$	$p(1 0)$	$p(0 1)$	$p(1 0)$
Method				
QDF	207	204	0.958333333	0.047619048
NPAR k=3	210	54	0.972222222	0.012605042
NPAR k=3	214	55	0.990740741	0.012838469
NPAR k=3	211	50	0.976851852	0.011671335
NPAR k=5	214	15	0.990740741	0.003501401
NPAR k=5	214	17	0.990740741	0.003968254
Total	of 216	of 4284		

Table 6: Training data vs Test Data Posterior

4 Appendix

SAS code

```

/*****
Tim Parshall's Mortality data
This file represents the combination of the two datasets omorttab and umorttab.
Variable definitions:
plot $      : one of five plots, A B C D, in which trees are mapped
spp $      : species, HE-hemlock, SM-sugar maple (YB-yellow birch, TI-basswood)
canopy $   : location of crown within the forest, UN-understory OV-overstory
tree_num   : unique for each tree
totneiden  : total neighborhood density (number of trees within 10 meters)
totnei_he_den : number of trees of HE within 10 meters
totnei_sm_den : number of trees of SM within 10 meters
totnei_yb_den : number of trees of YB within 10 meters
totnei_ti_den : number of trees of TI within 10 meters
totnei_ba  : total neighborhood basal area (all basal area within 10 meters)
totnei_he_ba : basal area of HE within 10 meters
per_tot_he_ba : percent basal area attributed to HE
totnei_sm_ba : basal area of SM within 10 meters
per_tot_sm_ba : percent basal area attributed to SM
totnei_yb_ba : basal area of YB within 10 meters
per_tot_yb_ba : percent basal area attributed to YB
totnei_ti_ba : basal area of TI within 10 meters

```

```

per_tot_ti_ba : percent basal area attributed to TI
crown $      : position of crown within canopy (more specific than "canopy")
              CAN INT OT SUB GAP SUPER
mort        : 1=dead at second census; 0=alive at second census
ovneiba     : total basal area of overstory trees within 10 meters
ovheba10   : basal area of overstory HE within 10 meters
ovsmba10   : basal area of overstory SM within 10 meters
ovybba10   : basal area of overstory YB within 10 meters
ovtiba10   : basal area of overstory TI within 10 meters
ovrelheba10 : percent basal area of HE within 10 meters
ovrelsmba10 : percent basal area of SM within 10 meters
ovrelybba10 : percent basal area of YB within 10 meters
ovreltiba10 : percent basal area of TI within 10 meters
dbh        : diameter of tree

*****/
data mortall;
  input plot $ spp $ canopy $ tree_num totneiden totnei_he_den
        totnei_sm_den totnei_yb_den totnei_ti_den totneiba
        totnei_he_ba per_tot_he_ba totnei_sm_ba per_tot_sm_ba
        totnei_yb_ba per_tot_yb_ba totnei_ti_ba per_tot_ti_ba crown $
        mort ovneiba ovheba10 ovsmba10 ovybba10 ovtiba10
        ovrelheba10 ovrelsmba10 ovrelybba10 ovreltiba10 dbh;
cards;
[data here]
;
run;

%adxgen
%adxtrans(mortall,tmortall, ,tree_num totneiden totnei_he_den totnei_sm_den
totnei_yb_den totnei_ti_den totneiba totnei_he_ba per_tot_he_ba
totnei_sm_ba per_tot_sm_ba totnei_yb_ba per_tot_yb_ba totnei_ti_ba
per_tot_ti_ba mort ovneiba ovheba10 ovsmba10 ovybba10 ovtiba10
ovrelheba10 ovrelsmba10 ovrelybba10 ovreltiba10 dbh)

/*****/
* input data;
options nocenter;
data train;
  infile 'C:\USERS\Erik\WPI\mva\mvhw10\train.dat';
  input pop x1-x15;
run;
data test;
  infile 'C:\USERS\Erik\WPI\mva\mvhw10\test.dat';
  input x1-x15;
run;

* quad best;
proc discrim data=train method=normal pool=no crossvalidate;
  priors equal;
  class pop;
  var x1 x4 x6 x7 x10 x11 x12;
run;

* npar best k=10;
proc discrim data=train method=npar k=10;
  priors equal;
  class pop;
  var x1 x7 x12;
run;

* npar best k=5;
proc discrim data=train method=npar k=5;
  priors equal;
  class pop;

```

```

    var x1 x6 x7 x12;
run;

** histogram of quad rates;
*data quadrate;
*   infile 'C:\USERS\Erik\WPI\mva\mvhw10\mvhw10vxz.grp';
*   input a $ b c qrate;
*run;
*
** histogram of npar rates;
*data nparrate;
*   infile 'C:\USERS\Erik\WPI\mva\mvhw10\mvhw10wxz.grp';
*   input a $ b c nrate;
*run;
*
*data diff;
*   merge quadrate nparrate;
*run;

* quad classify test data;
proc discrim data=train method=normal pool=no crossvalidate testdata=test testlist;
  priors equal;
  class pop;
  var x1 x4 x6 x7 x10 x11 x12;
run;

* npar classify test data;
proc discrim data=train method=npar k=5 testdata=test testlist;
  priors equal;
  class pop;
  var x1 x6 x7 x12;
run;

** BEGIN HERE **;
*****;
*****;
*****;
*population column;
data pop;
  set sasuser.train;
  keep pop;
run;
*****;
*individual columns;
proc discrim data=sasuser.train method=normal pool=no out=out1;
  class pop; priors '0'=0.95 '1'=0.05;
  var x2 x3 x4 x5 x6 x7 x8 x9 x10 x11 x12 x13;
run;
data out1;
  set out1;
  into1=_INTO_;
  keep into1;
run;
proc discrim data=sasuser.train method=npar k=3 out=out2;
  class pop; priors '0'=0.95 '1'=0.05;
  var x1 x3 x6 x7 x9 x10 x14 x15;
run;
data out2;
  set out2;
  into2=_INTO_;
  keep into2;
run;
proc discrim data=sasuser.train method=npar k=3 out=out3;
  class pop; priors '0'=0.95 '1'=0.05;
  var x1 x3 x7 x9 x10 x15;
run;
data out3;
  set out3;
  into3=_INTO_;
  keep into3;
run;
proc discrim data=sasuser.train method=npar k=3 out=out4;

```

```

class pop; priors '0'=0.95 '1'=0.05;
var x1 x2 x3 x6 x7 x12 x13;
run;
data out4;
set out4;
into4=_INTO_;
keep into4;
run;
proc discrim data=sasuser.train method=npair k=5 out=out5;
class pop; priors '0'=0.95 '1'=0.05;
var x1 x2 x3 x6 x7 x10 x13 x14;
run;
data out5;
set out5;
into5=_INTO_;
keep into5;
run;
proc discrim data=sasuser.train method=npair k=5 out=out6;
class pop; priors '0'=0.95 '1'=0.05;
var x3 x6 x7 x8 x10 x14;
run;
data out6;
set out6;
into6=_INTO_;
keep into6;
run;
*proc discrim data=sasuser.train method=npair k=5 out=out7;
* class pop; *priors '0'=0.95 '1'=0.05;
* var x2 x3 x6 x7 x8 x10 x14;
*run;
*proc discrim data=sasuser.train method=npair k=3 out=out7;
* class pop; *priors '0'=0.95 '1'=0.05;
* var x3 x6 x7 x10 x13 x14 x15;
*run;
*proc discrim data=sasuser.train method=normal pool=no out=out7;
* class pop; *priors '0'=0.95 '1'=0.05;
* var x2 x3 x4 x5 x6 x7 x9 x11 x12 x13 x14;
*run;
*
*data out7;
* set out7;
* into7=_INTO_;
* keep into7;
*run;
*****
* merge all columns;
data alldata;
* merge pop out1 out2 out3 out4 out5 out6 out7;
merge pop out1 out2 out3 out4 out5 out6;
run;

*****
*****
**[Begin] Multiple Classification under normal measure of error rate;
*****
*****
proc iml;
use alldata;
* read all;
* read all var {pop into1 into2 into3 into4 into5 into6 into7} into intodata;
read all var {pop into1 into2 into3 into4 into5 into6} into intodata;
k=ncol(intodata)-1; *number of outs;
n=nrow(intodata); *number of obs;
print n;
c1={-1 1};
c2={ 1 0};
j=J(n,1,1);
****p10 is for misclassification into population 1|0;
****p01 is for misclassification into population 0|1;
p10=repeat(0,k,1);
p01=repeat(0,k,1);

```

```

*for out[i];
  p10[1]=212/4413;
  p01[1]=217/242;
  p10[2]=18/4413;
  p01[2]=189/242;
  p10[3]=22/4413;
  p01[3]=190/242;
  p10[4]=20/4413;
  p01[4]=190/242;
  p10[5]=13/4413;
  p01[5]=217/242;
  p10[6]=9/4413;
  p01[6]=216/242;
* p10[7]=0.044628099;
* p01[7]=0.001937458;
* p01[7]=(217/242);
* p10[7]=(207/4413);
*** multiply by priors;
  do i=1 to k;
    p01[i]=p01[i]*0.05;
    p10[i]=p10[i]*0.95;
  end;
  print p10 p01;
  print c1 c2;
  y=J(n,k,.);
  do i=1 to k;
    lp10=log(p10[i]);
    lp01=log(p01[i]);
    print lp10 lp01;
    prob=lp01//lp10;
    coli=intodata[,i+1];      *coli is the response from rule i+1;
    p=(coli*c1+j*c2)*prob;   *first column is 1-ri, second column is ri;
    y[:,i]=p;
  end;
  outdata=y;
  create yiout from outdata;
  append from outdata;
run;
quit;

*take the minimum of each row;
* find the observation corresponding to column yi;
data all;
merge yiout alldata;
run;

proc iml;
  use all;
* read all var {col1 col2 col3 col4 col5 col6 col7} into x;
* read all var {into1 into2 into3 into4 into5 into6 into7} into z;
  read all var {col1 col2 col3 col4 col5 col6} into x;
  read all var {into1 into2 into3 into4 into5 into6} into z;
  read all var {pop} into pup;
  k=ncol(x);
  n=nrow(x);
  pmin=J(n,k,.);
  sgn=J(n,k,.);
  do i=1 to n;
*   pmin[i,]=min(x[i,1],x[i,2],x[i,3],x[i,4],x[i,5],x[i,6],x[i,7]);
    pmin[i,]=min(x[i,1],x[i,2],x[i,3],x[i,4],x[i,5],x[i,6]);
  end;
  sgn=sign(pmin-x)+J(n,k,1);
  r=(z#sgn)*J(k,1,1);
  sum=0;
  do i=1 to n;
    score=abs(r[i]-pup[i]);
    sum=score+sum;
  end;

```

```

print sum;
  outdata=r;
  create rec from outdata; *rec=rate for equal cost;
  append from outdata;
run;
quit;
*****;
*****;
**[End] Multiple Classification under normal measure of error rate;
*****;
*****;

*****;
*****Cost-Weighted*****;
**[Begin] Multiple Classification under measure of cost-weighted error rate;
*****;
*****;
proc iml;
  use alldata;
  * read all;
  * read all var {pop into1 into2 into3 into4 into5 into6 into7} into intodata;
  read all var {pop into1 into2 into3 into4 into5 into6} into intodata;

  k=ncol(intodata)-1; *number of outs;
  n=nrow(intodata); *number of obs;
  print n;

  c1={-1 1};
  c2={ 1 0};
  j=J(n,1,1);

  *****p10 is for misclassification into population 1|0;
  *****p01 is for misclassification into population 0|1;
  p10=repeat(0,k,1);
  p01=repeat(0,k,1);
  *for out [i];
  p10[1]=212/4413;
  p01[1]=217/242;
  p10[2]=18/4413;
  p01[2]=189/242;
  p10[3]=22/4413;
  p01[3]=190/242;
  p10[4]=20/4413;
  p01[4]=190/242;
  p10[5]=13/4413;
  p01[5]=217/242;
  p10[6]=9/4413;
  p01[6]=216/242;
  * p10[7]=0.044628099;
  * p01[7]=0.001937458;
  * p01[7]=(217/242);
  * p10[7]=(207/4413);
  *** multiply by priors;
  * do i=1 to k;
  *   p01[i]=p01[i]*0.05;
  *   p10[i]=p10[i]*0.95;
  * end;
  print p10 p01;
  print c1 c2;
  y=J(n,k,.);
  do i=1 to k;
    lp10=log(p10[i]);
    lp01=log(p01[i]);
    print lp10 lp01;
    prob=lp01/lp10;
    coli=intodata[,i+1]; *coli+1 is the response from rule i;
    p=(coli*c1+j*c2)*prob; *first column is 1-ri, second column is ri;
    y[,i]=p;
  end;

```

```

    outdata=y;
    create yiout from outdata;
    append from outdata;
run;
quit;

*take the minimum of each row;
* find the observation corresponding to column yi;
data all;
merge yiout alldata;
run;

proc iml;
use all;
* read all var {col1 col2 col3 col4 col5 col6 col7} into x;
* read all var {into1 into2 into3 into4 into5 into6 into7} into z;
read all var {col1 col2 col3 col4 col5 col6} into x;
read all var {into1 into2 into3 into4 into5 into6} into z;
read all var {pop} into pup;
k=ncol(x);
n=nrow(x);
pmin=J(n,k,.);
sgn=J(n,k,.);
do i=1 to n;
* pmin[i,]=min(x[i,1],x[i,2],x[i,3],x[i,4],x[i,5],x[i,6],x[i,7]);
  pmin[i,]=min(x[i,1],x[i,2],x[i,3],x[i,4],x[i,5],x[i,6]);
end;
sgn=sign(pmin-x)+J(n,k,1);
r=(z#sgn)*J(k,1,1);

sum=0;
sum10=0; *classify into 1 given 0;
sum01=0;
do i=1 to n;
  score=abs(r[i]-pup[i]);
  score1=(r[i]-pup[i]);
  if score1 > 0 then sum10=sum10+1;
  if score1 < 0 then sum01=sum01+1;
  sum=score+sum;
end;
print sum sum10 sum01;

rate01=sum01/4413;
rate10=sum10/242;

print rate01 rate10;

outdata=r;
create rcw from outdata; *rcw=rate for cost weighted;
append from outdata;
run;
quit;

data rcw;
set rcw;
rcw=col1;
drop col1;
run;
*****;
*****Cost-Weighted*****;
**[End] Multiple Classification under measure of cost-weighted error rate;
*****;
*****;

*****;
*****;
**[Begin] Comparison - put cost-weighted as one rule in equal cost;
*****;
*****;
*****;
* merge all columns;
data alldatac; *all data with rcw for comparison in equal cost;
* merge pop out1 out2 out3 out4 out5 out6 out7;

```

```

merge pop out1 out2 out3 out4 out5 out6 rcw;
run;
proc iml;
  use alldatac;
  * read all;
  * read all var {pop into1 into2 into3 into4 into5 into6 into7} into intodata;
  read all var {pop into1 into2 into3 into4 into5 into6 rcw} into intodata;

  k=ncol(intodata)-1; *number of outs;
  n=nrow(intodata); *number of obs;
  c1={-1 1};
  c2={ 1 0};
  j=J(n,1,1);

  *****p10 is for misclassification into population 1|0;
  *****p01 is for misclassification into population 0|1;
  p10=repeat(0,k,1);
  p01=repeat(0,k,1);
  *for out[i];
  p10[1]=212/4413;
  p01[1]=217/242;
  p10[2]=18/4413;
  p01[2]=189/242;
  p10[3]=22/4413;
  p01[3]=190/242;
  p10[4]=20/4413;
  p01[4]=190/242;
  p10[5]=13/4413;
  p01[5]=217/242;
  p10[6]=9/4413;
  p01[6]=216/242;
  p01[7]=(132/242); * from rcw;
  p10[7]=(267/4413);

  *** multiply by priors;
  do i=1 to k;
    p01[i]=p01[i]*0.05;
    p10[i]=p10[i]*0.95;
  end;
  print p10 p01;
  print c1 c2;
  y=J(n,k,.);
  do i=1 to k;
    lp10=log(p10[i]);
    lp01=log(p01[i]);
    print lp10 lp01;
    prob=lp01/lp10;
    coli=intodata[,i+1]; *coli is the response from rule i+1;
    p=(coli*c1+j*c2)*prob; *first column is 1-ri, second column is ri;
    y[,i]=p;
  end;
  outdata=y;
  create yioutwc from outdata;
  append from outdata;
run;
quit;

*take the minimum of each row;
* find the observation corresponding to column yi;
data allc;
merge yioutwc alldatac;
run;

proc iml;
use allc;
* read all var {col1 col2 col3 col4 col5 col6 col7} into x;
* read all var {into1 into2 into3 into4 into5 into6 into7} into z;
read all var {col1 col2 col3 col4 col5 col6 col7} into x;
read all var {into1 into2 into3 into4 into5 into6 rcw} into z;
read all var {pop} into pup;
k=ncol(x);

```

```

n=nrow(x);
print n k;
pmin=J(n,k,.);
sig=J(n,k,.);
do i=1 to n;
* pmin[i,]=min(x[i,1],x[i,2],x[i,3],x[i,4],x[i,5],x[i,6],x[i,7]);
  pmin[i,]=min(x[i,1],x[i,2],x[i,3],x[i,4],x[i,5],x[i,6],x[i,7]);
end;
sig=sign(pmin-x)+J(n,k,1);
rc=(z#sig)*J(k,1,1);

sum=0;
sum10=0; *classify into 1 given 0;
sum01=0;
do i=1 to n;
  score=abs(rc[i]-pup[i]);
  score1=(rc[i]-pup[i]);
  if score1 > 0 then sum10=sum10+1;
  if score1 < 0 then sum01=sum01+1;
  sum=score+sum;
end;
print sum sum10 sum01;

  outdata=rc;
  create rcom from outdata; *rcom=rate for comparison;
  append from outdata;

run;
quit;
*****;
*****;
**[End] Multiple Classification under normal measure of error rate;
*****;
*****;

*-----;
*-----;
* TEST DATA TEST DATA;
*-----;
*-----;

*****;
*population column;
data tpop;
  set sasuser.test;
  keep tpop;
run;

*****;
*individual columns;
proc discrim data=sasuser.train method=normal pool=no out=out1 testdata=sasuser.test testout=tout1;
  class pop; priors '0'=0.95 '1'=0.05;
  var x2 x3 x4 x5 x6 x7 x8 x9 x10 x11 x12 x13;
run;
data out1;
  set out1;
  into1=_INTO_;
  keep into1;
run;
data tout1;
  set tout1;
  tinto1=_INTO_;
  keep tinto1;
run;

proc discrim data=sasuser.train method=npair k=3 out=out2 testdata=sasuser.test testout=tout2;
  class pop; priors '0'=0.95 '1'=0.05;
  var x1 x3 x6 x7 x9 x10 x14 x15;
run;
data out2;

```

```

    set out2;
    into2=_INTO_;
    keep into2;
run;
data tout2;
  set tout2;
  tinto2=_INTO_;
  keep tinto2;
run;
proc discrim data=sasuser.train method=npair k=3 out=out3 testdata=sasuser.test testout=tout3;
  class pop; priors '0'=0.95 '1'=0.05;
  var x1 x3 x7 x9 x10 x15;
run;
data out3;
  set out3;
  into3=_INTO_;
  keep into3;
run;
data tout3;
  set tout3;
  tinto3=_INTO_;
  keep tinto3;
run;
proc discrim data=sasuser.train method=npair k=3 out=out4 testdata=sasuser.test testout=tout4;
  class pop; priors '0'=0.95 '1'=0.05;
  var x1 x2 x3 x6 x7 x12 x13;
run;
data out4;
  set out4;
  into4=_INTO_;
  keep into4;
run;
data tout4;
  set tout4;
  tinto4=_INTO_;
  keep tinto4;
run;
proc discrim data=sasuser.train method=npair k=5 out=out5 testdata=sasuser.test testout=tout5;
  class pop; priors '0'=0.95 '1'=0.05;
  var x1 x2 x3 x6 x7 x10 x13 x14;
run;
data out5;
  set out5;
  into5=_INTO_;
  keep into5;
run;
data tout5;
  set tout5;
  tinto5=_INTO_;
  keep tinto5;
run;
proc discrim data=sasuser.train method=npair k=5 out=out6 testdata=sasuser.test testout=tout6;
  class pop; priors '0'=0.95 '1'=0.05;
  var x3 x6 x7 x8 x10 x14;
run;
data out6;
  set out6;
  into6=_INTO_;
  keep into6;
run;
data tout6;
  set tout6;
  tinto6=_INTO_;
  keep tinto6;
run;

*****;
* merge all columns;
data talldata;
  merge tpop tout1 tout2 tout3 tout4 tout5 tout6;
run;

*****;
*****;
**[Begin] Multiple Classification under normal measure of error rate;
*****;

```

```

*****;
proc iml;
  use talldata;
  read all var {tpop tinto1 tinto2 tinto3 tinto4 tinto5 tinto6} into tintodata;
  k=ncol(tintodata)-1; *number of outs;
  n=nrow(tintodata); *number of obs;
  print n;

  c1={-1 1};
  c2={ 1 0};
  j=J(n,1,1);

  *****p10 is for misclassification into population 1|0;
  *****p01 is for misclassification into population 0|1;
  p10=repeat(0,k,1);
  p01=repeat(0,k,1);
  *for out[i];
  p10[1]=212/4413;
  p01[1]=217/242;
  p10[2]=18/4413;
  p01[2]=189/242;
  p10[3]=22/4413;
  p01[3]=190/242;
  p10[4]=20/4413;
  p01[4]=190/242;
  p10[5]=13/4413;
  p01[5]=217/242;
  p10[6]=9/4413;
  p01[6]=216/242;
  **[Begin] Multiple Classification under cost-weighted measure MODIFIED;
  *****;
  *****;
  *population column;
  data pop;
    set sasuser.train;
    keep pop;
  run;
  *****;
  *individual columns;
  proc discrim data=sasuser.train method=normal pool=no out=out1;
    class pop; priors '0'=0.5 '1'=0.5;
    var x2 x3 x4 x5 x6 x7 x8 x9 x10 x11 x12 x13;
  run;
  data out1;
    set out1;
    into1=_INTO_;
    keep into1;
  run;
  proc discrim data=sasuser.train method=npair k=3 out=out2;
    class pop; priors '0'=0.5 '1'=0.5;
    var x1 x3 x6 x7 x9 x10 x14 x15;
  run;
  data out2;
    set out2;
    into2=_INTO_;
    keep into2;
  run;
  proc discrim data=sasuser.train method=npair k=3 out=out3;
    class pop; priors '0'=0.5 '1'=0.5;
    var x1 x3 x7 x9 x10 x15;
  run;
  data out3;
    set out3;
    into3=_INTO_;
    keep into3;
  run;
  proc discrim data=sasuser.train method=npair k=3 out=out4;
    class pop; priors '0'=0.5 '1'=0.5;
    var x1 x2 x3 x6 x7 x12 x13;
  run;
  data out4;

```

```

    set out4;
    into4=_INT0_;
    keep into4;
run;
proc discrim data=sasuser.train method=npair k=5 out=out5;
  class pop; priors '0'=0.5 '1'=0.5;
  var x1 x2 x3 x6 x7 x10 x13 x14;
run;
data out5;
  set out5;
  into5=_INT0_;
  keep into5;
run;
proc discrim data=sasuser.train method=npair k=5 out=out6;
  class pop; priors '0'=0.5 '1'=0.5;
  var x3 x6 x7 x8 x10 x14;
run;
data out6;
  set out6;
  into6=_INT0_;
  keep into6;
run;
*proc discrim data=sasuser.train method=npair k=5 out=out7;
* class pop; *priors '0'=0.5 '1'=0.5;
* var x2 x3 x6 x7 x8 x10 x14;
*run;
*proc discrim data=sasuser.train method=npair k=3 out=out7;
* class pop; *priors '0'=0.5 '1'=0.5;
* var x3 x6 x7 x10 x13 x14 x15;
*run;
*proc discrim data=sasuser.train method=normal pool=no out=out7;
* class pop; *priors '0'=0.5 '1'=0.5;
* var x2 x3 x4 x5 x6 x7 x9 x11 x12 x13 x14;
*run;
*
*data out7;
* set out7;
* into7=_INT0_;
* keep into7;
*run;
*****
* merge all columns;
data alldata;
* merge pop out1 out2 out3 out4 out5 out6 out7;
  merge pop out1 out2 out3 out4 out5 out6;
run;
proc iml;
  use alldata;
  read all var {pop into1 into2 into3 into4 into5 into6} into intodata;
  k=ncol(intodata)-1; *number of outs;
  n=nrow(intodata); *number of obs;
  print n;
  c1={-1 1};
  c2={ 1 0};
  j=J(n,1,1);
*****p10 is for misclassification into population 1|0;
*****p01 is for misclassification into population 0|1;
  p10=repeat(0,k,1);
  p01=repeat(0,k,1);
*for out[i];
  p10[1]=1511/4413;
  p01[1]=96/242;
  p10[2]=422/4413;
  p01[2]=0/242;
  p10[3]=441/4413;
  p01[3]=0/242;
  p10[4]=394/4413;
  p01[4]=0/242;
  p10[5]=763/4413;

```

```

p01[5]=0/242;
p10[6]=972/4413;
p01[6]=4/242;
print c1 c2;
y=J(n,k,.);
do i=1 to k;
  lp10=p10[i];
  lp01=p01[i];
  print lp10 lp01;
  prob=lp01//lp10;
  coli=intodata[,i+1];      *coli is the response from rule i+1;
  p=(coli*c1+j*c2)*prob;    *first column is 1-ri, second column is ri;
  y[,i]=p;
end;
outdata=y;
create yiout from outdata;
append from outdata;
run;
quit;

*take the minimum of each row;
* find the observation corresponding to column yi;
data all;
  merge yiout alldata;
run;

proc iml;
  use all;
  * read all var {col1 col2 col3 col4 col5 col6 col7} into x;
  * read all var {into1 into2 into3 into4 into5 into6 into7} into z;
  read all var {col1 col2 col3 col4 col5 col6} into x;
  read all var {into1 into2 into3 into4 into5 into6} into z;
  read all var {pop} into pup;
  k=ncol(x);
  n=nrow(x);
  pmin=J(n,k,.);
  sgn=J(n,k,.);
  do i=1 to n;
  * pmin[i,]=min(x[i,1],x[i,2],x[i,3],x[i,4],x[i,5],x[i,6],x[i,7]);
    pmin[i,]=min(x[i,1],x[i,2],x[i,3],x[i,4],x[i,5],x[i,6]);
  end;
  sgn=sign(pmin-x)+J(n,k,1);
  r=(z#sgn)*J(k,1,1);

  sum=0;
  sum10=0; *classify into 1 given 0;
  sum01=0;
  do i=1 to n;
    score=abs(r[i]-pup[i]);
    score1=(r[i]-pup[i]);
    if score1 > 0 then sum10=sum10+1;
    if score1 < 0 then sum01=sum01+1;
    sum=score+sum;
  end;
  print sum sum10 sum01;

  rate01=sum01/4413;
  rate10=sum10/242;

  print rate01 rate10;

  outdata=r;
  create rcw from outdata; *rcw=rate for cost weighted;
  append from outdata;
run;
quit;

data rcw;
  set rcw;
  rcw=col1;
  drop col1;
run;* p10[7]=0.044628099;
* p01[7]=0.001937458;

```

```

* p01[7]=(217/242);
* p10[7]=(207/4413);
*** multiply by priors;
do i=1 to k;
  p01[i]=p01[i]*0.05;
  p10[i]=p10[i]*0.95;
end;
print p10 p01;
print c1 c2;
y=J(n,k,.);
do i=1 to k;
  lp10=log(p10[i]);
  lp01=log(p01[i]);
  print lp10 lp01;
  prob=lp01//lp10;
  coli=tintodata[,i+1];      *coli is the response from rule i+1;
  p=(coli*c1+j*c2)*prob;    *first column is 1-ri, second column is ri;
  y[,i]=p;
end;
outdata=y;
create tyiout from outdata;
append from outdata;
run;
quit;

*take the minimum of each row;
* find the observation corresponding to column yi;
data tall;
merge tyiout talldata;
run;

proc iml;
use tall;
read all var {col1 col2 col3 col4 col5 col6} into x;
read all var {tinto1 tinto2 tinto3 tinto4 tinto5 tinto6} into z;
read all var {tpop} into tpup;
k=ncol(x);
n=nrow(x);
pmin=J(n,k,.);
sig=J(n,k,.);
do i=1 to n;
* pmin[i,]=min(x[i,1],x[i,2],x[i,3],x[i,4],x[i,5],x[i,6],x[i,7]);
pmin[i,]=min(x[i,1],x[i,2],x[i,3],x[i,4],x[i,5],x[i,6]);
end;
sig=sign(pmin-x)+J(n,k,1);
r=(z#sig)*J(k,1,1);
sum=0;
do i=1 to n;
  score=abs(r[i]-tpup[i]);
  sum=score+sum;
end;
print sum;
  outdata=r;
  create trec from outdata; *trec=rate for equal cost;
  append from outdata;
run;
quit;

*Compare tpop with trec;
data compare;
merge tpop trec;
trec=col1;
drop col1;
run;

proc iml;
use compare;
read all var {trec} into trec;
read all var {tpop} into tpop;

```

```

n=nrow(tpop);
sum=0;
sum10=0; *classify into 1 given 0;
sum01=0;
do i=1 to n;
  score=abs(trec[i]-tpop[i]);
  score1=(trec[i]-tpop[i]);
  if score1 > 0 then sum10=sum10+1;
  if score1 < 0 then sum01=sum01+1;
  sum=score+sum;
end;
rate10=sum10/4284;
rate01=sum01/216;

print sum sum10 sum01;
print rate10 rate01;

run;
quit;

data rule2;
merge tpop tout2;
run;

*****;
*****;
**[End] Multiple Classification under normal measure of error rate;
*****;
*****;

**[Begin] Multiple Classification under cost-weighted measure MODIFIED;
*****;
*****;

*****;
*population column;
data pop;
  set sasuser.train;
  keep pop;
run;

*****;
*individual columns;
proc discrim data=sasuser.train method=normal pool=no out=out1;
  class pop; priors '0'=0.5 '1'=0.5;
  var x2 x3 x4 x5 x6 x7 x8 x9 x10 x11 x12 x13;
run;
data out1;
  set out1;
  into1=_INTO_;
  keep into1;
run;

proc discrim data=sasuser.train method=npar k=3 out=out2;
  class pop; priors '0'=0.5 '1'=0.5;
  var x1 x3 x6 x7 x9 x10 x14 x15;
run;
data out2;
  set out2;
  into2=_INTO_;
  keep into2;
run;

proc discrim data=sasuser.train method=npar k=3 out=out3;
  class pop; priors '0'=0.5 '1'=0.5;
  var x1 x3 x7 x9 x10 x15;
run;
data out3;
  set out3;
  into3=_INTO_;
  keep into3;
run;

proc discrim data=sasuser.train method=npar k=3 out=out4;
  class pop; priors '0'=0.5 '1'=0.5;
  var x1 x2 x3 x6 x7 x12 x13;
run;
data out4;
  set out4;
  into4=_INTO_;
  keep into4;
run;

```

```

proc discrim data=sasuser.train method=npar k=5 out=out5;
  class pop; priors '0'=0.5 '1'=0.5;
  var x1 x2 x3 x6 x7 x10 x13 x14;
run;
data out5;
  set out5;
  into5=_INTO_;
  keep into5;
run;

proc discrim data=sasuser.train method=npar k=5 out=out6;
  class pop; priors '0'=0.5 '1'=0.5;
  var x3 x6 x7 x8 x10 x14;
run;
data out6;
  set out6;
  into6=_INTO_;
  keep into6;
run;

*proc discrim data=sasuser.train method=npar k=5 out=out7;
* class pop; *priors '0'=0.5 '1'=0.5;
* var x2 x3 x6 x7 x8 x10 x14;
*run;

*proc discrim data=sasuser.train method=npar k=3 out=out7;
* class pop; *priors '0'=0.5 '1'=0.5;
* var x3 x6 x7 x10 x13 x14 x15;
*run;

*proc discrim data=sasuser.train method=normal pool=no out=out7;
* class pop; *priors '0'=0.5 '1'=0.5;
* var x2 x3 x4 x5 x6 x7 x9 x11 x12 x13 x14;
*run;
*
*data out7;
* set out7;
* into7=_INTO_;
* keep into7;
*run;
*****
* merge all columns;
data alldata;
* merge pop out1 out2 out3 out4 out5 out6 out7;
  merge pop out1 out2 out3 out4 out5 out6;
run;

proc iml;
  use alldata;
  read all var {pop into1 into2 into3 into4 into5 into6} into intodata;

  k=ncol(intodata)-1; *number of outs;
  n=nrow(intodata); *number of obs;
  print n;

  c1={-1 1};
  c2={ 1 0};
  j=J(n,1,1);

*****p10 is for misclassification into population 1|0;
*****p01 is for misclassification into population 0|1;
  p10=repeat(0,k,1);
  p01=repeat(0,k,1);
*for out[i];
  p10[1]=1511/4413;
  p01[1]=96/242;
  p10[2]=422/4413;
  p01[2]=0/242;
  p10[3]=441/4413;
  p01[3]=0/242;
  p10[4]=394/4413;
  p01[4]=0/242;
  p10[5]=763/4413;
  p01[5]=0/242;
  p10[6]=972/4413;
  p01[6]=4/242;
  print c1 c2;

```

```

y=J(n,k,.);
do i=1 to k;
  lp10=p10[i];
  lp01=p01[i];
  print lp10 lp01;
  prob=lp01//lp10;
  coli=intodata[,i+1];      *coli is the response from rule i+1;
  p=(coli*c1+j*c2)*prob;    *first column is 1-ri, second column is ri;
  y[,i]=p;
end;
outdata=y;
create yiout from outdata;
append from outdata;
run;
quit;

*take the minimum of each row;
* find the observation corresponding to column yi;
data all;
merge yiout alldata;
run;

proc iml;
use all;
* read all var {col1 col2 col3 col4 col5 col6 col7} into x;
* read all var {into1 into2 into3 into4 into5 into6 into7} into z;
read all var {col1 col2 col3 col4 col5 col6} into x;
read all var {into1 into2 into3 into4 into5 into6} into z;
read all var {pop} into pup;
k=ncol(x);
n=nrow(x);
pmin=J(n,k,.);
sgn=J(n,k,.);
do i=1 to n;
* pmin[i,]=min(x[i,1],x[i,2],x[i,3],x[i,4],x[i,5],x[i,6],x[i,7]);
  pmin[i,]=min(x[i,1],x[i,2],x[i,3],x[i,4],x[i,5],x[i,6]);
end;
sgn=sign(pmin-x)+J(n,k,1);
r=(z#sgn)*J(k,1,1);

sum=0;
sum10=0; *classify into 1 given 0;
sum01=0;
do i=1 to n;
  score=abs(r[i]-pup[i]);
  score1=(r[i]-pup[i]);
  if score1 > 0 then sum10=sum10+1;
  if score1 < 0 then sum01=sum01+1;
  sum=score+sum;
end;
print sum sum10 sum01;

rate01=sum01/4413;
rate10=sum10/242;

print rate01 rate10;

outdata=r;
create rcw from outdata; *rcw=rate for cost weighted;
append from outdata;
run;
quit;

data rcw;
set rcw;
rcw=coll;
drop coll;
run;
quit;

```