Project

Le Pham Tuyen

November 21, 2015

```
###########
# Read CSV
###########
population = rbind(read.csv("/data/ss13pusa.csv"),read.csv("/data/ss13pusb.csv"))
population = population[,c("SERIALNO","AGEP","CIT","COW","DIS","HICOV","INDP","JWTR","MAR","MIL","MSP",

housing = rbind(read.csv("/data/ss13husa.csv"),read.csv("/data/ss13husb.csv"))
housing = housing[,c("SERIALNO","BLD","FES","FS","HHT","HINCP","MV","NOC","NP","RWAT","TYPE","VEH","WIF

###########
# Merge table
###########
dataset = merge(population,housing,by="SERIALNO")

rm(population)
rm(housing)

###########
# Pre-processing data
###########
dataset$DIS[dataset$DIS==2]=0
dataset$FS[dataset$FS==2]=0
dataset$HICOV[dataset$HICOV==2]=0
dataset$RWAT[dataset$RWAT==2]=0
dataset$SEX[dataset$SEX==2]=0
summary(dataset)
```

```
##     SERIALNO            AGEP            CIT             COW
##  Min.   :      1   Min.   : 0.00   Min.   :1.000   Min.   :1.0
##  1st Qu.: 373240   1st Qu.:20.00   1st Qu.:1.000   1st Qu.:1.0
##  Median : 745893   Median :41.00   Median :1.000   Median :1.0
##  Mean   : 746334   Mean   :40.46   Mean   :1.419   Mean   :2.2
##  3rd Qu.:1119273   3rd Qu.:59.00   3rd Qu.:1.000   3rd Qu.:3.0
##  Max.   :1492843   Max.   :95.00   Max.   :5.000   Max.   :9.0
##                                                    NA's   :1288573
##       DIS             HICOV            INDP             JWTR
##  Min.   :0.0000   Min.   :0.0000   Min.   : 170      Min.   : 1
##  1st Qu.:0.0000   1st Qu.:1.0000   1st Qu.:4970      1st Qu.: 1
##  Median :0.0000   Median :1.0000   Median :7390      Median : 1
##  Mean   :0.1479   Mean   :0.8725   Mean   :6389      Mean   : 2
##  3rd Qu.:0.0000   3rd Qu.:1.0000   3rd Qu.:8270      3rd Qu.: 1
##  Max.   :1.0000   Max.   :1.0000   Max.   :9920      Max.   :12
##                                    NA's   :1288573   NA's   :1750932
##       MAR             MIL              MSP              PAP
##  Min.   :1.000   Min.   :1.0      Min.   :1        Min.   :    0.0
##  1st Qu.:1.000   1st Qu.:4.0      1st Qu.:1        1st Qu.:    0.0
##  Median :3.000   Median :4.0      Median :2        Median :    0.0
```

```
##  Mean   :2.948   Mean   :3.8      Mean   :3        Mean   :   45.7
##  3rd Qu.:5.000   3rd Qu.:4.0      3rd Qu.:6        3rd Qu.:    0.0
##  Max.   :5.000   Max.   :4.0      Max.   :6        Max.   :30000.0
##                  NA's   :632061   NA's   :550753   NA's   :550753
##      RAC1P            SCH             SEMP             SEX
##  Min.   :1.000   Min.   :1.0     Min.   : -9000   Min.   :0.000
##  1st Qu.:1.000   1st Qu.:1.0     1st Qu.:     0   1st Qu.:0.000
##  Median :1.000   Median :1.0     Median :     0   Median :0.000
##  Mean   :1.861   Mean   :1.3     Mean   :  1770   Mean   :0.488
##  3rd Qu.:1.000   3rd Qu.:2.0     3rd Qu.:     0   3rd Qu.:1.000
##  Max.   :9.000   Max.   :3.0     Max.   :525000   Max.   :1.000
##                  NA's   :97281   NA's   :550753
##        ST              WKHP             BLD              FES
##  Min.   : 1.00   Min.   : 1.0      Min.   : 1.00    Min.   :1
##  1st Qu.:12.00   1st Qu.:32.0      1st Qu.: 2.00    1st Qu.:1
##  Median :27.00   Median :40.0      Median : 2.00    Median :2
##  Mean   :27.64   Mean   :37.9      Mean   : 2.72    Mean   :3
##  3rd Qu.:42.00   3rd Qu.:42.0      3rd Qu.: 2.00    3rd Qu.:4
##  Max.   :56.00   Max.   :99.0      Max.   :10.00    Max.   :8
##                  NA's   :1541931   NA's   :148256   NA's   :642989
##        FS              HHT              HINCP              MV
##  Min.   :0.0000   Min.   :1.00     Min.   : -19770   Min.   :1.00
##  1st Qu.:0.0000   1st Qu.:1.00     1st Qu.:  33500   1st Qu.:3.00
##  Median :0.0000   Median :1.00     Median :  63300   Median :4.00
##  Mean   :0.1499   Mean   :2.06     Mean   :  85536   Mean   :4.13
##  3rd Qu.:0.0000   3rd Qu.:3.00     3rd Qu.: 107000   3rd Qu.:5.00
##  Max.   :1.0000   Max.   :7.00     Max.   :2090000   Max.   :7.00
##                   NA's   :148256   NA's   :148256    NA's   :148267
##       NOC               NP              RWAT             TYPE
##  Min.   : 0.00    Min.   : 1.000   Min.   :0        Min.   :1.000
##  1st Qu.: 0.00    1st Qu.: 2.000   1st Qu.:1        1st Qu.:1.000
##  Median : 0.00    Median : 3.000   Median :1        Median :1.000
##  Mean   : 0.88    Mean   : 3.192   Mean   :1        Mean   :1.071
##  3rd Qu.: 2.00    3rd Qu.: 4.000   3rd Qu.:1        3rd Qu.:1.000
##  Max.   :18.00    Max.   :20.000   Max.   :1        Max.   :3.000
##  NA's   :148256                    NA's   :148256
##       VEH              WIF
##  Min.   :0.00     Min.   :0.0
##  1st Qu.:1.00     1st Qu.:1.0
##  Median :2.00     Median :2.0
##  Mean   :2.07     Mean   :1.6
##  3rd Qu.:3.00     3rd Qu.:2.0
##  Max.   :6.00     Max.   :3.0
##  NA's   :148256   NA's   :634560
```

```
#Recast data type to appropriate class
dataset$COW[is.na(dataset$COW)]="NA"
dataset$COW = as.factor(dataset$COW)

dataset$DIS[is.na(dataset$DIS)]="NA"
dataset$DIS = as.factor(dataset$DIS)

dataset$HICOV[is.na(dataset$HICOV)]="NA"
dataset$HICOV = as.factor(dataset$HICOV)
```

```
dataset$INDP[is.na(dataset$INDP)]="NA"
dataset$INDP = as.factor(dataset$INDP)

dataset$JWTR[is.na(dataset$JWTR)]="NA"
dataset$JWTR = as.factor(dataset$JWTR)

dataset$MAR[is.na(dataset$MAR)]="NA"
dataset$MAR = as.factor(dataset$MAR)

dataset$MIL[is.na(dataset$MIL)]="NA"
dataset$MIL = as.factor(dataset$MIL)

dataset$MSP[is.na(dataset$MSP)]="NA"
dataset$MSP = as.factor(dataset$MSP)

dataset$PAP[is.na(dataset$PAP)]="NA"
dataset$PAP = as.factor(dataset$PAP)
dataset$PAP = as.numeric(dataset$PAP)

dataset$RAC1P[is.na(dataset$RAC1P)]="NA"
dataset$RAC1P = as.factor(dataset$RAC1P)

dataset$SCH[is.na(dataset$SCH)]="NA"
dataset$SCH = as.factor(dataset$SCH)

dataset$SEMP[is.na(dataset$SEMP)]="NA"
dataset$SEMP = as.factor(dataset$SEMP)
dataset$SEMP = as.numeric(dataset$SEMP)

dataset$SEX[is.na(dataset$SEX)]="NA"
dataset$SEX = as.factor(dataset$SEX)

dataset$ST[is.na(dataset$ST)]="NA"
dataset$ST = as.factor(dataset$ST)

dataset$WKHP[is.na(dataset$WKHP)]="NA"
dataset$WKHP = as.factor(dataset$WKHP)
dataset$WKHP = as.integer(dataset$WKHP)

dataset$BLD[is.na(dataset$BLD)]="NA"
dataset$BLD = as.factor(dataset$BLD)

dataset$FES[is.na(dataset$FES)]="NA"
dataset$FES = as.factor(dataset$FES)

dataset$FS[is.na(dataset$FS)]="NA"
dataset$FS = as.factor(dataset$FS)

dataset$HHT[is.na(dataset$HHT)]="NA"
dataset$HHT = as.factor(dataset$HHT)

dataset$HINCP[is.na(dataset$HINCP)]="NA"
dataset$HINCP = as.factor(dataset$HINCP)
```

```r
dataset$HINCP = as.numeric(dataset$HINCP)

dataset$MV[is.na(dataset$MV)]="NA"
dataset$MV = as.factor(dataset$MV)

dataset$NOC[is.na(dataset$NOC)]="NA"
dataset$NOC = as.factor(dataset$NOC)
dataset$NOC = as.integer(dataset$NOC)

dataset$RWAT[is.na(dataset$RWAT)]="NA"
dataset$RWAT = as.factor(dataset$RWAT)

dataset$TYPE[is.na(dataset$TYPE)]="NA"
dataset$TYPE = as.factor(dataset$TYPE)

dataset$VEH[is.na(dataset$VEH)]="NA"
dataset$VEH = as.factor(dataset$VEH)
dataset$VEH = as.integer(dataset$VEH)

dataset$CIT[is.na(dataset$CIT)]="NA"
dataset$CIT = as.factor(dataset$CIT)

# Standardize the continous variables to center the data and to have common scale
dataset$AGEP = scale(dataset$AGEP, center = TRUE, scale = TRUE)
dataset$HINCP = scale(dataset$HINCP, center = TRUE, scale = TRUE)
dataset$NOC = scale(dataset$NOC, center = TRUE, scale = TRUE)
dataset$NP = scale(dataset$NP, center = TRUE, scale = TRUE)
dataset$PAP = scale(dataset$PAP, center = TRUE, scale = TRUE)
dataset$SEMP = scale(dataset$SEMP, center = TRUE, scale = TRUE)
dataset$VEH = scale(dataset$VEH, center = TRUE, scale = TRUE)

# IDENTIFYING UNTIDY, REDUNDANT, AND COLLINEAR VARIABLES
# Make a correlation Matrix to identify which pairs have high correlation greater than .80
corel = abs(cor(dataset[c("AGEP", "HINCP", "NOC", "NP", "PAP", "SEMP", "VEH", "WKHP")]))
corel
```

```
##              AGEP       HINCP       NOC         NP        PAP       SEMP
## AGEP  1.000000000 0.004749503 0.3960031 0.46433170 0.64624042 0.61755690
## HINCP 0.004749503 1.000000000 0.1825103 0.09203402 0.03758095 0.03661963
## NOC   0.396003068 0.182510303 1.0000000 0.29208195 0.32278326 0.30431892
## NP    0.464331700 0.092034017 0.2920820 1.00000000 0.37118579 0.35364862
## PAP   0.646240423 0.037580949 0.3227833 0.37118579 1.00000000 0.93823237
## SEMP  0.617556899 0.036619627 0.3043189 0.35364862 0.93823237 1.00000000
## VEH   0.055758495 0.223326663 0.4301180 0.03260487 0.08347769 0.07480337
## WKHP  0.092173235 0.007048512 0.1716680 0.09512984 0.44759470 0.38689380
##              VEH        WKHP
## AGEP  0.05575850 0.092173235
## HINCP 0.22332666 0.007048512
## NOC   0.43011804 0.171668006
## NP    0.03260487 0.095129839
## PAP   0.08347769 0.447594704
## SEMP  0.07480337 0.386893802
## VEH   1.00000000 0.061164794
```

```
## WKHP  0.06116479 1.000000000
```

```
#####################
# ASSOCIATION RULES
#####################
# Drop numeric column
dataset_rules = dataset[-c(1,2,12,18,23,25,     7, 11, 15, 17, 20, 22, 26, 28, 29, 30)]

# Using apriori algorithm with confidence 0.5
#install.packages("arules")
#install.packages("arulesViz")
library(arules)
```

```
## Loading required package: Matrix
##
## Attaching package: 'arules'
##
## The following objects are masked from 'package:base':
##
##     %in%, abbreviate, write
```

```
library(arulesViz)
```

```
## Loading required package: grid
```

```
rules = apriori(dataset_rules, parameter = list(support=0.001,confidence=0.5))
```

```
## Apriori
##
## Parameter specification:
##  confidence minval smax arem  aval originalSupport support minlen maxlen
##         0.5    0.1    1 none FALSE            TRUE   0.001      1     10
##  target   ext
##   rules FALSE
##
## Algorithmic control:
##  filter tree heap memopt load sort verbose
##     0.1 TRUE TRUE  FALSE TRUE    2    TRUE
##
## Absolute minimum support count: 3132
##
## set item appearances ...[0 item(s)] done [0.00s].
## set transactions ...[81 item(s), 3132795 transaction(s)] done [2.34s].
## sorting and recoding items ... [75 item(s)] done [0.54s].
## creating transaction tree ... done [5.98s].
## checking subsets of size 1 2 3 4 5 6 7 8 9 10 done [57.82s].
## writing ... [9317657 rule(s)] done [1.76s].
## creating S4 object  ... done [5.66s].
```

```
# Only get rules in which HICOV is 1
subrules1 = subset(rules, (rhs %in% c("HICOV=1")))
```

```r
# Only get rules in which confidence is greater than 0.99
subsubrules1 = subrules1[quality(subrules1)$confidence > 0.99]
# Sort rules by lift
rules_high_lift1=head(sort(subsubrules1,by="lift"),10)
# Show the result
inspect(rules_high_lift1)
```

```
##      lhs                rhs       support     confidence lift
## 76   {MIL=1}         => {HICOV=1} 0.003303759 1          1.146132
## 1325 {COW=5,MIL=1}   => {HICOV=1} 0.003303759 1          1.146132
## 1341 {MIL=1,RWAT=NA} => {HICOV=1} 0.001201802 1          1.146132
## 1351 {MIL=1,BLD=NA}  => {HICOV=1} 0.001201802 1          1.146132
## 1359 {MIL=1,MV=NA}   => {HICOV=1} 0.001201802 1          1.146132
## 1374 {JWTR=1,MIL=1}  => {HICOV=1} 0.002380941 1          1.146132
## 1385 {MAR=5,MIL=1}   => {HICOV=1} 0.001352147 1          1.146132
## 1397 {MAR=1,MIL=1}   => {HICOV=1} 0.001742533 1          1.146132
## 1412 {MIL=1,SEX=1}   => {HICOV=1} 0.002863258 1          1.146132
## 1420 {MIL=1,BLD=2}   => {HICOV=1} 0.001151687 1          1.146132
```

```r
# Only get rules in which HICOV is 0
subrules2 = subset(rules, (rhs %in% c("HICOV=0")))
# Only get rules in which confidence is greater than 0.75
subsubrules2 = subrules2[quality(subrules2)$confidence > 0.75]
# Sort rules by lift
rules_high_lift2=head(sort(subsubrules2,by="lift"),10)
# Show the result
inspect(rules_high_lift2)
```

```
##   lhs          rhs           support confidence     lift
## 1 {COW=1,
##    DIS=0,
##    JWTR=NA,
##    MIL=4,
##    RAC1P=1,
##    SCH=1,
##    SEX=1,
##    FS=0,
##    RWAT=NA} => {HICOV=0} 0.001105083  0.7516283 5.895129
## 2 {COW=1,
##    DIS=0,
##    JWTR=NA,
##    MIL=4,
##    RAC1P=1,
##    SCH=1,
##    SEX=1,
##    BLD=NA,
##    FS=0}    => {HICOV=0} 0.001105083  0.7516283 5.895129
## 3 {COW=1,
##    DIS=0,
##    JWTR=NA,
##    MIL=4,
##    RAC1P=1,
##    SCH=1,
```

```
##     SEX=1,
##     FS=0,
##     MV=NA}   => {HICOV=0} 0.001105083  0.7516283 5.895129
```

```
#################
# SPLIT DATA
#################
# SELECTION FOR 19 VARIABLES USING TRAIN DATA SET
# Leave out the 10 variables that wer found to be untidy, redundant, or collinear
#training_step <- training_noMissing[-c("INDP", "MSP", "SEMP", "ST", "FES", "HHT", "NP", "VEH", "WIF")]
# Test data set
dataset_logistics <- dataset[-c(1, 7, 11, 15, 17, 20, 22, 26, 28, 29, 30)]

#SPLITTING THE KNOWN DATA SET INTO 2 PARTS
#install.packages("caret")
#install.packages("kernlab")

library(caret)
```

```
## Loading required package: lattice
## Loading required package: ggplot2
```

```
# Divide the training data set 50-50 First half is called training data set
# and missing values are removed
intrain = createDataPartition(y = dataset_logistics$HICOV, p = 0.75, list = F)
training = dataset_logistics[intrain, ]
testing = dataset_logistics[-intrain, ]

# Omit NA value
training = na.omit(training)
testing = na.omit(testing)

######################
# Logistics
######################
fit <- glm(HICOV ~ ., data = training, family = "binomial")
prediction<-predict(fit, type="response",newdata=testing)
```

```
## Warning in predict.lm(object, newdata, se.fit, scale = 1, type =
## ifelse(type == : prediction from a rank-deficient fit may be misleading
```

```
table<-table(prediction>0.5,testing$HICOV)
table
```

```
##
##              0      1
##   FALSE  13595   9693
##   TRUE   86262 673648
```

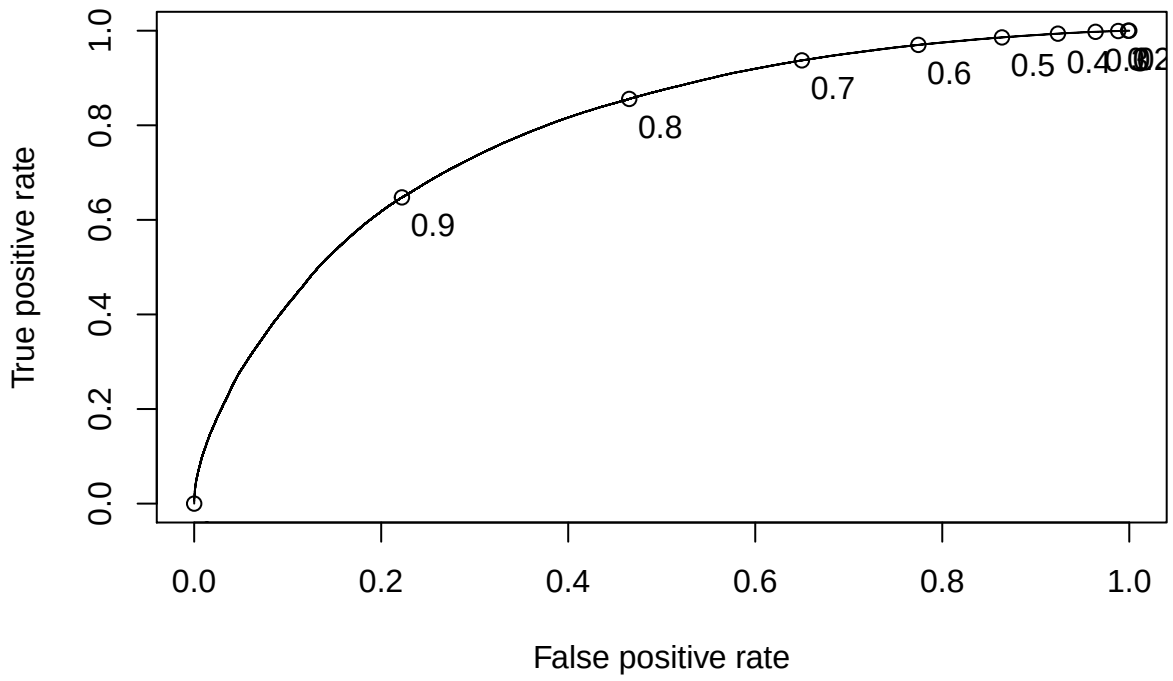
```
accuracy<-sum(diag(table))/(sum(table))
accuracy
```

```
## [1] 0.8774831
```

```
#install.packages('ROCR')
library("ROCR")
```

```
## Loading required package: gplots
##
## Attaching package: 'gplots'
##
## The following object is masked from 'package:stats':
##
##     lowess
```

```
# Show ROC/AUC
ROCRpred = prediction(prediction, testing$HICOV)
# Performance function
ROCRperf = performance(ROCRpred, "tpr", "fpr")
# Plot ROC curve with colors and threshold labels
plot(ROCRperf, print.cutoffs.at=seq(0,1,by=0.1), text.adj=c(-0.2,1.7))
```




```
as.numeric(performance(ROCRpred, "auc")@y.values)
```

```
## [1] 0.7875552
```

```
######################
# Decision tree
######################
#install.packages("rpart.plot")
library("rpart")
library("rpart.plot")
```

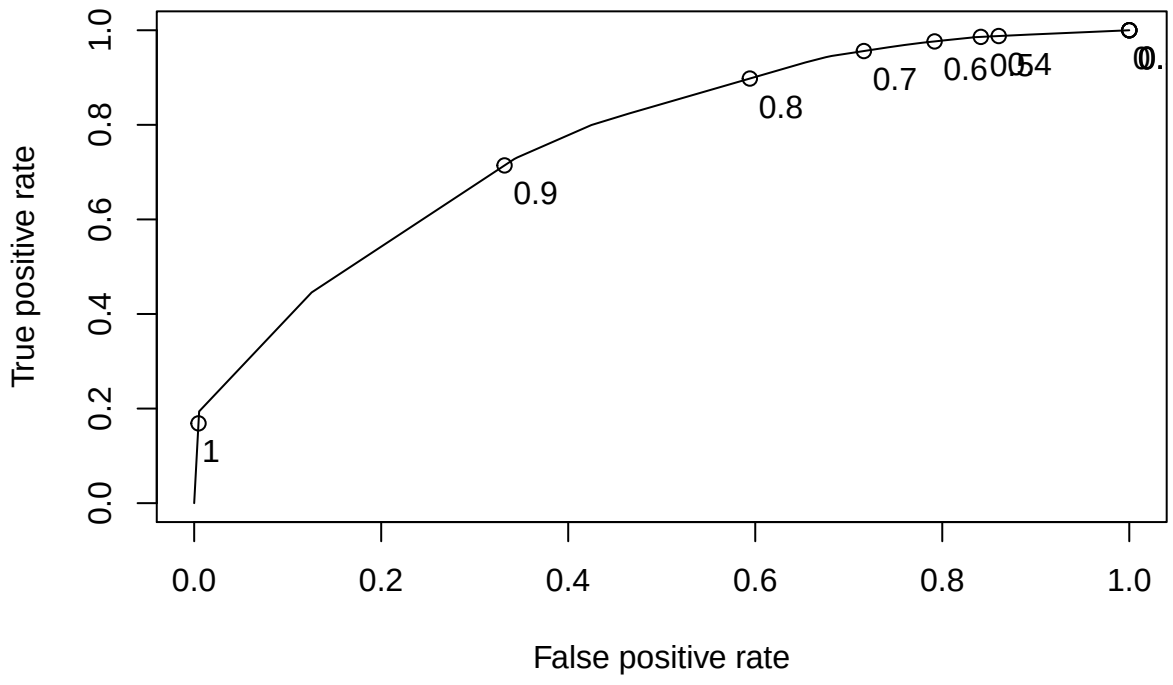
```
fit = rpart(HICOV ~ ., method="class", data=training, cp=0.002)
prediction<-predict(fit, type="class",newdata=testing)
table<-table(prediction,testing$HICOV)
table
```

```
##
## prediction      0      1
##          0  16419   9951
##          1  83438 673390
```

```
accuracy<-sum(diag(table))/(sum(table))
accuracy
```

```
## [1] 0.8807594
```

```
# Show ROC/AUC
prediction<-predict(fit,newdata=testing)[ , 2] #only decision tree
ROCRpred = prediction(prediction, testing$HICOV)
# Performance function
ROCRperf = performance(ROCRpred, "tpr", "fpr")
# Plot ROC curve with colors and threshold labels
plot(ROCRperf, print.cutoffs.at=seq(0,1,by=0.1), text.adj=c(-0.2,1.7))
```




```
as.numeric(performance(ROCRpred, "auc")@y.values)
```

```
## [1] 0.7658136
```

```
##################
# Naive Bayesian
##################
library("e1071")

model <- naiveBayes(HICOV ~ .,data=training, laplace=.01)
prediction <- predict (model,testing)
table<-table(prediction,testing$HICOV)
table
```

```
##
## prediction      0      1
##          0  36353  64166
##          1  63504 619175
```

```
accuracy<-sum(diag(table))/(sum(table))
accuracy
```

```
## [1] 0.8369889
```