Tuncten_code

Sara Tuncten

2018-03-27

The Business Problem

You were recently hired as a data scientist by Universal Bank. The bankâs Vice President is interested in building a model to help predict when users might respond to a campaign to take out a personal loan. She supplies you with a dataset containing information of 5,000 customers and attaches a description about the dataset (see page 3 of this assignment).

The VP tells you that another analyst has created a logistic regression model and hands you an R code file (please download the Template_HW_1.Rmd file).

Data file description

The file **UniversalBank.csv** contains data on 5,000 customers. The data include customer demographic information (age, income, etc.), the customerâs relationship with the bank (mortgage, securities accounts, etc.), and the customer response to the last personal loan campaign (Personal.Loan). Among these 5,000 customers, only 480 (9.6%) accepted the personal loan that was offered to them in the earlier campaign.

Here is a description of each variable in the Universal Bank dataset:

ID: Customer ID

Age: Customerâs age in years

Experience: Number of years of professional work experience

Income: Annual income in thousands of dollars ($000)

Zip.Code: Zip code of home address

Family: Customer’s family size

CC Avg: Average spending on credit cards per month in thousands of dollars ($000)

Education: Education level where 1 = Undergraduate; 2 = Graduate; and 3=Advanced/Professional

Mortgage: Value of house mortgage if any; in thousands of dollar ($000)

Personal.Loan: Did the customer accept a personal loan offered in the bankâs last campaign? 1=Yes; 0 = No

Securities.Account: Does the customer have a securities account with the bank? 1 = Yes; 0 = No

CD.Account: Does the customer have a certificate of deposit (CD) account with the bank? 1 = Yes; 0 = No

Online: Does the customer use Internet banking facilities? 1 = Yes; 0 = No

Credit.Card: Does the customer use a credit card issued by Universal Bank? 1 = Yes; 0 = No

Data File Prep

We will drop the ID and Zip.Code columns and also recode Education variable into a factor variable.

```
bank.df <- read.csv("UniversalBank.csv")
bank.df <- bank.df[,-c(1,5)] # drop ID and zip code columns.
# create categorical variable for education
bank.df$Education <- factor(bank.df$Education, levels = c(1,2,3), labels = c("Undergr
ad", "Graduate", "Advanced/Professional"))
```

Logistic Regression Model

```
logit.reg <- glm(bank.df$Personal.Loan ~., data = bank.df, family = "binomial")
options(scipen=999)
summary(logit.reg)
```

```
##
## Call:
## glm(formula = bank.df$Personal.Loan ~ ., family = "binomial",
##     data = bank.df)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -3.1306  -0.1900  -0.0696  -0.0218   4.1835
##
## Coefficients:
##                                   Estimate  Std. Error z value
## (Intercept)                    -12.3108411   1.8175314  -6.773
## Age                             -0.0359088   0.0672659  -0.534
## Experience                       0.0450304   0.0668215   0.674
## Income                           0.0601828   0.0029663  20.289
## Family                           0.6181733   0.0770409   8.024
## CCAvg                            0.1633658   0.0440570   3.708
## EducationGraduate                3.9654074   0.2696021  14.708
## EducationAdvanced/Professional   4.0640722   0.2669264  15.225
## Mortgage                         0.0007105   0.0005940   1.196
## Securities.Account              -0.8701273   0.3006906  -2.894
## CD.Account                       3.8389147   0.3415635  11.239
## Online                          -0.7605257   0.1657436  -4.589
## CreditCard                      -1.0381977   0.2130956  -4.872
##                                            Pr(>|z|)
## (Intercept)                     0.00000000001258031 ***
## Age                                        0.593457
## Experience                                 0.500381
## Income                         < 0.0000000000000002 ***
## Family                          0.00000000000000102 ***
## CCAvg                                      0.000209 ***
## EducationGraduate              < 0.0000000000000002 ***
## EducationAdvanced/Professional < 0.0000000000000002 ***
## Mortgage                                   0.231633
## Securities.Account                         0.003807 **
## CD.Account                     < 0.0000000000000002 ***
## Online                          0.00000446295762500 ***
## CreditCard                      0.00000110484246161 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 3162.0  on 4999  degrees of freedom
## Residual deviance: 1172.3  on 4987  degrees of freedom
## AIC: 1198.3
##
## Number of Fisher Scoring iterations: 8
```

Predict new customer’s personal loan

Create data set for customer

```
#create data set for new customer
new_Age <- (38)
new_Experience <- (17)
new_Income <- (150)
new_Family <- (1)
new_CCAvg <- (0.2)
new_Education <- (2)
new_Mortgage <- (0)
new_Personal.Loan <- (0)
new_Securities.Account <- (0)
new_CD.Account <- (0)
new_Online <- (1)
new_CreditCard <- (1)
#make sure values match original data set
Age <- as.integer(new_Age)
Experience <- as.integer(new_Experience)
Income <- as.integer(new_Income)
Family <- as.integer(new_Family)
CCAvg <- as.numeric(new_CCAvg)
Education <- as.factor(new_Education)
Mortgage <- as.integer(new_Mortgage)
Personal.Loan <- as.integer(new_Personal.Loan)
Securities.Account <- as.integer(new_Securities.Account)
CD.Account <- as.integer(new_CD.Account)
Online <- as.integer(new_Online)
CreditCard <- as.integer(new_CreditCard)

#combine new data frame
new_customer.df <- data.frame(Age,Experience,Income,Family,CCAvg,Education,Mortgage,P
ersonal.Loan,Securities.Account,CD.Account,Online,CreditCard)

#create education factor
new_customer.df$Education <- factor(new_customer.df$Education, levels = c(1,2,3), lab
els = c("Undergrad", "Graduate", "Advanced/Professional"))
```

Put new customer’s data into the glm.

```
glm_prediction <- predict(logit.reg, newdata = new_customer.df, type = "response")
glm_prediction
```

```
##        1
## 0.256372
```

```
#https://stackoverflow.com/questions/47325648/running-into-error-in-evalpredvars-data
-env-object-age-not-found-while
#https://stat.ethz.ch/R-manual/R-devel/library/stats/html/predict.glm.html
```

26% chance the new customer will take out a personal loan.

Naive Bayes Classification

```
library(e1071)
#put target variable first in data set

bank.df_rearrange <- bank.df[c(8,1:7,9:12)]
str(bank.df_rearrange)
```

```
## 'data.frame':    5000 obs. of  12 variables:
##  $ Personal.Loan     : int  0 0 0 0 0 0 0 0 0 1 ...
##  $ Age               : int  25 45 39 35 35 37 53 50 35 34 ...
##  $ Experience        : int  1 19 15 9 8 13 27 24 10 9 ...
##  $ Income            : int  49 34 11 100 45 29 72 22 81 180 ...
##  $ Family            : int  4 3 1 1 4 4 2 1 3 1 ...
##  $ CCAvg             : num  1.6 1.5 1 2.7 1 0.4 1.5 0.3 0.6 8.9 ...
##  $ Education         : Factor w/ 3 levels "Undergrad","Graduate",..: 1 1 1 2 2 2 2
3 2 3 ...
##  $ Mortgage          : int  0 0 0 0 0 155 0 0 104 0 ...
##  $ Securities.Account: int  1 1 0 0 0 0 0 0 0 0 ...
##  $ CD.Account        : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ Online            : int  0 0 0 0 0 1 1 0 1 0 ...
##  $ CreditCard        : int  0 0 0 0 1 0 0 1 0 0 ...
```

Change data to factors

```
Personal.Loan_nb <- as.factor(bank.df_rearrange$Personal.Loan)
Age_nb <- as.factor(bank.df_rearrange$Age)
Experience_nb <- as.factor(bank.df_rearrange$Experience)
Income_nb <- as.factor(bank.df_rearrange$Income)
Family_nb <- as.factor(bank.df_rearrange$Family)
CCAvg_nb <- as.factor(bank.df_rearrange$CCAvg)
Education_nb <- (bank.df_rearrange$Education)
Mortgage_nb <- as.factor(bank.df_rearrange$Mortgage)
Securities.Account_nb <- as.factor(bank.df_rearrange$Securities.Account)
CD.Account_nb <- as.factor(bank.df_rearrange$CD.Account)
Online_nb <- as.factor(bank.df_rearrange$Online)
CreditCard_nb <- as.factor(bank.df_rearrange$CreditCard)

bank.df_nb <- data.frame(Personal.Loan_nb,Age_nb,Experience_nb,Income_nb,Family_nb,CC
Avg_nb,Education_nb,Mortgage_nb,Securities.Account_nb,CD.Account_nb,Online_nb,CreditC
ard_nb)
```

Create train and test data sets

```
#using 80/20 rule
train_nb <- bank.df_nb[1:4000,]
test_nb <- bank.df_nb[4001:5000,]
prop.table(table(train_nb$Personal.Loan_nb))
```

```
##
##       0       1
## 0.90075 0.09925
```

```
prop.table(table(test_nb$Personal.Loan_nb))
```

```
##
##     0     1
## 0.917 0.083
```

Let's create the model using our train data set

```
nb_model_bank.df <- naiveBayes(Personal.Loan_nb ~., data = train_nb)
#nb_model_bank.df
```

Test model w/test data set

```
nb_pred_test <- predict(nb_model_bank.df,test_nb)
nb_table <- table(nb_pred_test,test_nb$Personal.Loan_nb)
#nb_table
prop.table(nb_table)
```

```
##
## nb_pred_test     0     1
##            0 0.892 0.032
##            1 0.025 0.051
```

Our NB factor predicts 94.3% of the data accurately Predict new customer's results

```
predict_nb <- predict(nb_model_bank.df,newdata = new_customer.df)
predict_nb
```

```
## [1] 0
## Levels: 0 1
```

Predicts new customer would not accept new loan.

kNN Classification w/Min Max Normalization

What columns need to be normalized?

```
#summary(bank.df_rearrange)
```

Need to normalize Age, Experience, Income, Family, CCAvg, Education, and Mortgage [,c(2,3,4,5,6,7,8)]

```
bank.df_knn <- read.csv("UniversalBank.csv")
bank.df_knn <- bank.df_knn[,-c(1,5)] # drop ID and zip code columns.
bank.df_knn$Education <- as.integer(bank.df_knn$Education)
#summary(bank.df_knn)
bank.df_knn_r <- bank.df_knn[c(8,1:7,9:12)]
#summary(bank.df_knn_r)

#new customer data
new_customer.df_knn <- new_customer.df[,c(8,1:7,9:12)]
new_customer.df_knn$Education <- as.integer(new_customer.df_knn$Education)
comb_new_customer.df_knn <- rbind(bank.df_knn_r,new_customer.df_knn)
tail(comb_new_customer.df_knn)
```

```
##      Personal.Loan Age Experience Income Family CCAvg Education Mortgage
## 4996             0  29          3     40      1   1.9         3        0
## 4997             0  30          4     15      4   0.4         1       85
## 4998             0  63         39     24      2   0.3         3        0
## 4999             0  65         40     49      3   0.5         2        0
## 5000             0  28          4     83      3   0.8         1        0
## 5001             0  38         17    150      1   0.2         2        0
##      Securities.Account CD.Account Online CreditCard
## 4996                  0          0      1          0
## 4997                  0          0      1          0
## 4998                  0          0      0          0
## 4999                  0          0      1          0
## 5000                  0          0      1          1
## 5001                  0          0      1          1
```

```
normalize<- function(x){return((x-min(x))/(max(x)-min(x)))}
library(scales)
##library(scales)
##rescale(bank.df_rearrange$Education)
bank.df_knn_dummy <- as.data.frame(lapply(bank.df_knn_r[,c(2,3,4,5,6,7,8)],normalize)
)
comb_bank.df_knn_dummy <- as.data.frame(lapply(comb_new_customer.df_knn[,c(2,3,4,5,6,
7,8)],normalize))
#summary(bank.df_knn_dummy)
#summary(comb_bank.df_knn_dummy)
```

Data is normalized. We need to combine normalized data with variables in the data set that were already on a 0-1 scale

```
bank.knn.df_knn_mmn <- cbind(bank.df_knn_r[,1],bank.df_knn_dummy[,],bank.df_knn_r[,c(
9:12)])
comb_bank.df_knn_mnn <- cbind(comb_new_customer.df_knn[,1],comb_bank.df_knn_dummy[,],
comb_new_customer.df_knn[,c(9:12)])
#summary(bank.knn.df_knn_mmn)
#summary(comb_bank.df_knn_mnn)
```

Data is normalized. We need to assure there are no missing values.

```
bank.knn.df_mmn_cc <- bank.knn.df_knn_mmn[complete.cases(bank.knn.df_knn_mmn),]
comb_bank.df_knn_mnn_cc <- comb_bank.df_knn_mnn[complete.cases(comb_bank.df_knn_mnn),
]
```

Create test and train data sets

```
#using 80/20 rule
knn_train <- bank.knn.df_mmn_cc[1:4000,2:12]
knn_test <- bank.knn.df_mmn_cc[4001:5000,2:12]
#data with new customer included
knn_train_newcustomer <- comb_bank.df_knn_mnn_cc[1:4000,2:12]
knn_test_newcustomer <- comb_bank.df_knn_mnn_cc[5001,2:12]
```

Create labels

```
bank.knn_trainlabel <- bank.knn.df_mmn_cc[1:4000,1]
bank.knn_testlabel <- bank.knn.df_mmn_cc[4001:5000,1]
#new customer test data set
bank.knn_trainlabel_newcustomer <- comb_bank.df_knn_mnn_cc[1:4000,1]
bank.knn_testlabel_newcustomer <- comb_bank.df_knn_mnn_cc[5001,1]
```

Create kNN Model

```
library(class)
set.seed(123)
bank.df_knn_pred <-knn(train= knn_train,test=knn_test,cl = bank.knn_trainlabel ,k=71)
#bank.df_knn_pred
knn_mm_table <- table(bank.knn_testlabel,bank.df_knn_pred)
knn_mm_table
```

```
##                    bank.df_knn_pred
## bank.knn_testlabel   0   1
##                   0 916   1
##                   1  72  11
```

```
prop.table(knn_mm_table)
```

```
##                    bank.df_knn_pred
## bank.knn_testlabel     0     1
##                  0 0.916 0.001
##                  1 0.072 0.011
```

```
#predict knn for new customer
set.seed(123)
newcustomer_knn_pred <- knn(train = knn_train_newcustomer, test = knn_test_newcustome
r, cl = bank.knn_trainlabel_newcustomer, k = 71)
newcustomer_knn_pred
```

```
## [1] 0
## Levels: 0 1
```

kNN with min max standardization predicts 92.7% of data accurately. Not as accurate as NB model. Predicts customer would not accept personal loan.

Decision Tree

create Train and Test data sets

```
#80/20 rule
train_dc <- bank.df[1:4000,]
test_dc <- bank.df[4001:5000,]
table(train_dc$Personal.Loan)
```

```
##
##    0    1
## 3603  397
```

```
table(test_dc$Personal.Loan)
```

```
##
##   0   1
## 917  83
```

```
prop.table(table(train_dc$Personal.Loan))
```

```
##
##       0       1
## 0.90075 0.09925
```

```
prop.table(table(test_dc$Personal.Loan))
```

```
##
##     0     1
## 0.917 0.083
```

Information Gain

```
library(caret)
```

```
## Warning: package 'caret' was built under R version 3.4.4
```

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

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

```
library(rpart.plot)
```

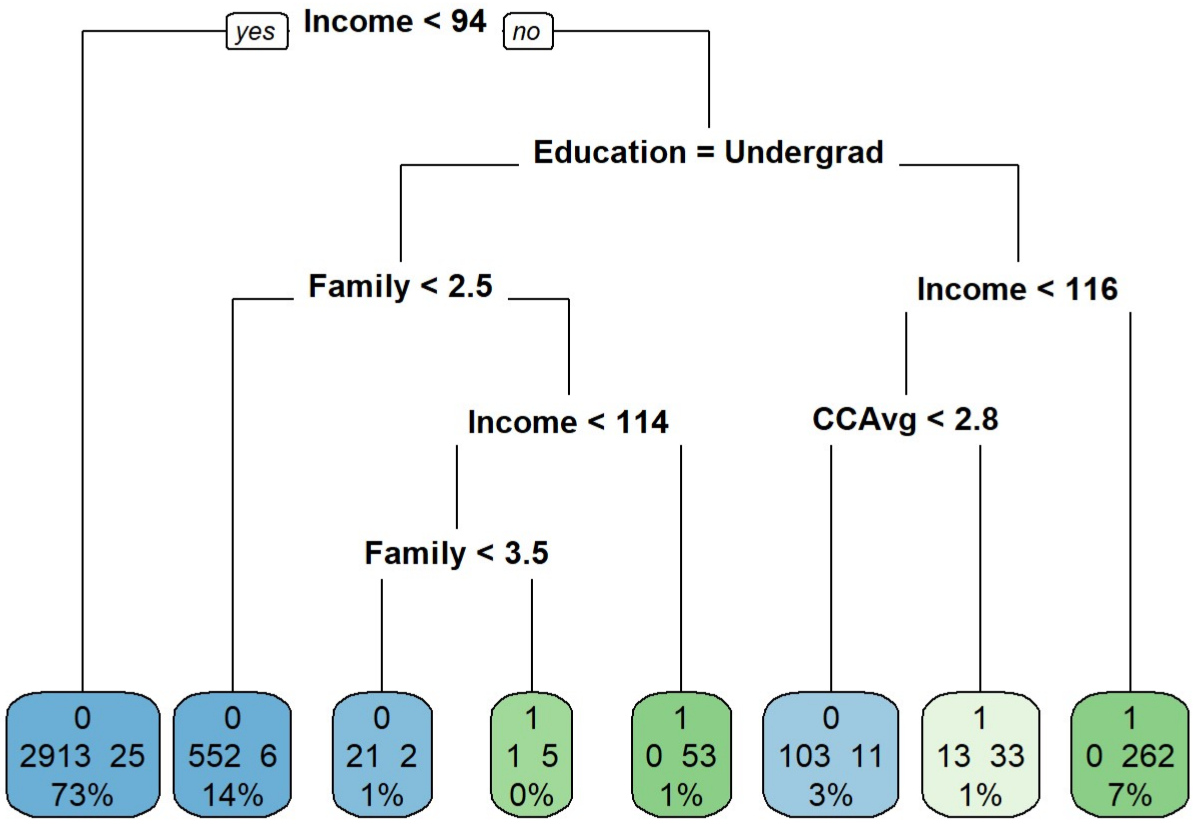
```
## Warning: package 'rpart.plot' was built under R version 3.4.4
```

```
## Loading required package: rpart
```

```
## Warning: package 'rpart' was built under R version 3.4.4
```

```
library(rpart)

bank.df_info <- rpart(train_dc$Personal.Loan ~ ., data=train_dc, method="class",
                    parms=list(split="information"),
                    control=rpart.control(minsplit = 1))
rpart.plot(bank.df_info, type=0, extra=101)
```

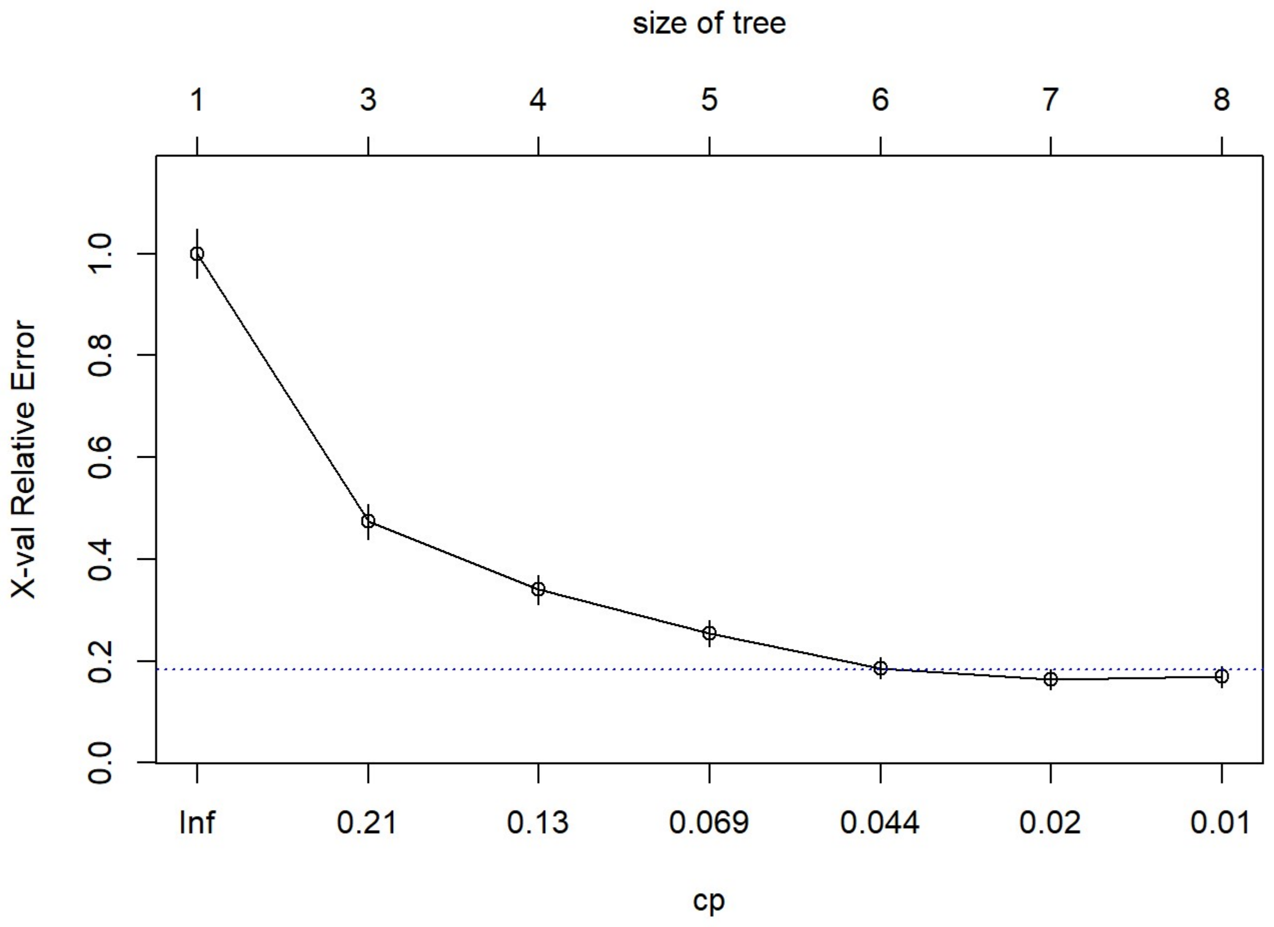


CP Table for Train data

```
#CP is 0.01 at 7 splits
cptable<-printcp(bank.df_info)
```

```
##
## Classification tree:
## rpart(formula = train_dc$Personal.Loan ~ ., data = train_dc,
##     method = "class", parms = list(split = "information"), control = rpart.control
(minsplit = 1))
##
## Variables actually used in tree construction:
## [1] CCAvg     Education Family    Income
##
## Root node error: 397/4000 = 0.09925
##
## n= 4000
##
##         CP nsplit rel error  xerror     xstd
## 1 0.239295      0   1.00000 1.00000 0.047633
## 2 0.181360      2   0.52141 0.47355 0.033716
## 3 0.095718      3   0.34005 0.34005 0.028769
## 4 0.050378      4   0.24433 0.25441 0.024993
## 5 0.037783      5   0.19395 0.18640 0.021467
## 6 0.010076      6   0.15617 0.16373 0.020142
## 7 0.010000      7   0.14610 0.16877 0.020445
```

```
plotcp(bank.df_info, minline=TRUE, col="blue")
```



```
```

How did we do?

```
pred_dc <- predict(bank.df_info, test_dc, type = "class")

library(gmodels)
```

```
## Warning: package 'gmodels' was built under R version 3.4.4
```

```
CrossTable(test_dc$Personal.Loan, pred_dc,
           prop.chisq = FALSE,prop.c = FALSE, prop.r = FALSE, dnn = c("actual","predi
cted"))
```

```
##
##
##    Cell Contents
## |-------------------------|
## |                       N |
## |         N / Table Total |
## |-------------------------|
##
##
## Total Observations in Table:  1000
##
##
##              | predicted
##       actual |         0 |         1 | Row Total |
## -------------|-----------|-----------|-----------|
##            0 |       910 |         7 |       917 |
##              |     0.910 |     0.007 |           |
## -------------|-----------|-----------|-----------|
##            1 |        10 |        73 |        83 |
##              |     0.010 |     0.073 |           |
## -------------|-----------|-----------|-----------|
## Column Total |       920 |        80 |      1000 |
## -------------|-----------|-----------|-----------|
##
##
```

Predicted 98.3% of data accurately.