```
---
title: "Assignment 4"
author: "Tenzin Tashi"
date: "May 07, 2022"
---
############## Exercise 4.1 ##############

### a.Import the survey data survey.csv into R, storing rows 1 through 600 as training data and rows 601 through 750 as testing data.
```{r}
#Import survey.csv
data = read.csv("Dataset/survey.csv")
#Split data
data_train = head(data,600)
data_test = tail(data, 150)
```

b. Build a classification tree from the training data using the “rpart” package, according to the formula “MYDEPV ~ Price + Income + Age”. Use the information gain splitting index. Which features were actually used to construct the tree? (see the “printcp” function) Plot the tree using the “rpart.plot” package.
```{r}
library(rpart)
library(rpart.plot)
decision_tree <- rpart(as.factor(MYDEPV) ~ Price + Income + Age, data = data_train, method = 'class', parms = list(split = 'information'))
printcp(decision_tree)
#Plot the tree
rpart.plot(decision_tree,extra = 106)
```
```

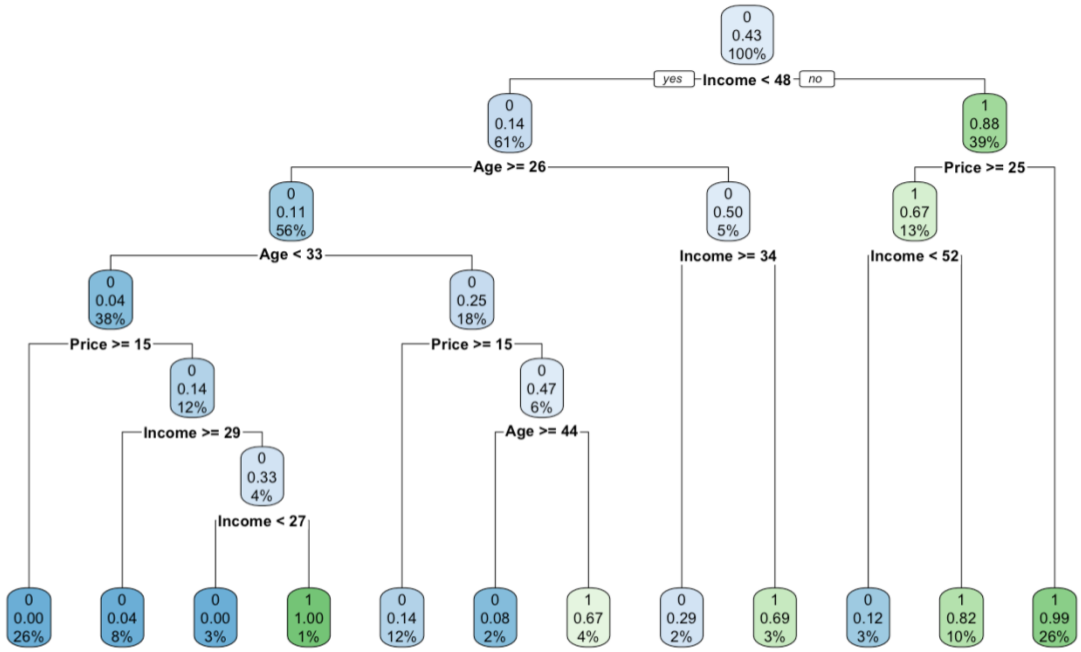
```
Classification tree:
rpart(formula = as.factor(MYDEPV) ~ Price + Income + Age, data = data_train,
    method = "class", parms = list(split = "information"))

Variables actually used in tree construction:
[1] Age    Income Price

Root node error: 260/600 = 0.43333

n= 600

        CP nsplit rel error  xerror     xstd
1 0.692308      0   1.00000 1.00000 0.046685
2 0.025000      1   0.30769 0.31154 0.032194
3 0.011538      3   0.25769 0.28462 0.030978
4 0.010256      5   0.23462 0.28462 0.030978
5 0.010000     11   0.17308 0.27692 0.030615
```




```
#### Features were actually used to construct the tree: Age, Income, Price

#### There are 11 internal nodes in the tree, and the tree high is 6.

### c. Score the model with the training data and create the model's confusion matrix.  Which class of MYDEPV was the model better able to classify?

```{r message=FALSE}
library(caret)
Pred <- predict(decision_tree, data_train, type = 'class')
Matrix <- confusionMatrix(Pred, as.factor(data_train$MYDEPV))
Matrix
```
```

```
Confusion Matrix and Statistics

          Reference
Prediction   0   1
         0 314  19
         1  26 241

               Accuracy : 0.925
                 95% CI : (0.9009, 0.9448)
    No Information Rate : 0.5667
    P-Value [Acc > NIR] : <2e-16

                  Kappa : 0.8478

 Mcnemar's Test P-Value : 0.3711

            Sensitivity : 0.9235
            Specificity : 0.9269
         Pos Pred Value : 0.9429
         Neg Pred Value : 0.9026
             Prevalence : 0.5667
         Detection Rate : 0.5233
   Detection Prevalence : 0.5550
      Balanced Accuracy : 0.9252

       'Positive' Class : 0
```

````
#### As the missclassification rates for both classes are almost equal, one can conclude that each class was classified equally well

#### The zero class missclassification rate: 26/(26+314) = `r 26/(26+314)`

#### The one class missclassification rate: 19/(19+241) = `r 19/(19+241)`


### d. Define the resubstitution error rate, and then calculate it using the confusion matrix from the previous step.  Is it a good indicator of predictive performance?  Why or why not?

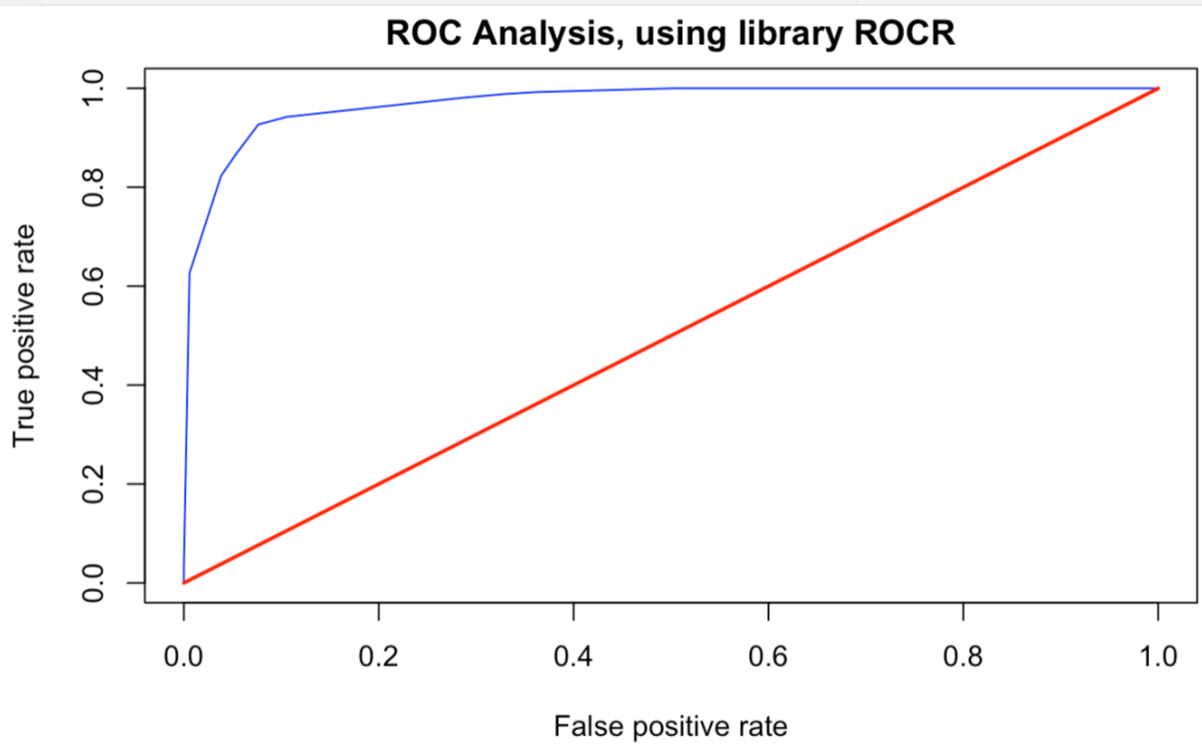

#### The resubstitution error rate is the number of incorrect classifications divided by the total number of classifications.

#### The resubstitution error rate: (19 + 26)/(19 + 26 + 314 +241) = `r (19 + 26)/(19 + 26 + 314 +241)`

#### In that case, it is a good indicator of predictive performance because the training data is used to train the tree and the tree usually doing well on its training data.

### e.Using the “ROCR” package, plot the receiver operating characteristic (ROC) curve. Calculate the area under the ROC curve (AUC). Describe the usefulness of this statistic.

```{r}
library(ROCR)
pred <- prediction(predict(decision_tree, type="prob")[,2], data_train$MYDEPV)
#Plot the ROC curve
roc <- performance(pred, "tpr", "fpr")
plot(roc, col='blue', main='ROC Analysis, using library ROCR')
lines(x=c(0, 1), y=c(0, 1), col="red", lwd=2)
# Calculate the area under the ROC curve
auc <- performance(pred, "auc")
auc@y.values
```
````




```
[[1]]
[1] 0.9720645
```

The value of AUC is equal to the probability that a classifier will rank a randomly chosen positive instance higher than a randomly chosen negative one.
ROC analysis for the tree:

- For your tree, the ROC curve is a non-decreasing curve.

- For your tree, the ROC curve is in the form of two connected line segments.

f. Score the model with the testing data. How accurate are the tree’s predictions?

```{r}
Pred <- predict(decision_tree, data_test, type = 'class')
Matrix <- confusionMatrix(Pred, as.factor(data_test$MYDEPV))
Matrix
```

```
Confusion Matrix and Statistics

          Reference
Prediction  0  1
         0 76  6
         1 10 58

               Accuracy : 0.8933
                 95% CI : (0.8326, 0.9378)
    No Information Rate : 0.5733
    P-Value [Acc > NIR] : <2e-16

                  Kappa : 0.7837

 Mcnemar's Test P-Value : 0.4533

            Sensitivity : 0.8837
            Specificity : 0.9062
         Pos Pred Value : 0.9268
         Neg Pred Value : 0.8529
             Prevalence : 0.5733
         Detection Rate : 0.5067
   Detection Prevalence : 0.5467
      Balanced Accuracy : 0.8950

       'Positive' Class : 0
```

````
#### The zero class missclassification rate: 10/(10+76) = `r 10/(10+76)`

#### The one class missclassification rate: 6/(6+58) = `r 6/(6+58)`

- The model performed well for both classes

- Due to the small amount of testing data, one can conclude that each class was classified almost equally well or bad.

### g. Repeat part (a), but set the splitting index to the Gini coefficient splitting index.  How does the new tree compare to the previous one?

```{r}
gini_tree <- rpart(as.factor(MYDEPV) ~ Price + Income + Age, data = data_train, method = 'class', parms = list(split = 'gini'))
printcp(gini_tree)
#Plot the tree
rpart.plot(gini_tree,extra = 106)
```
````

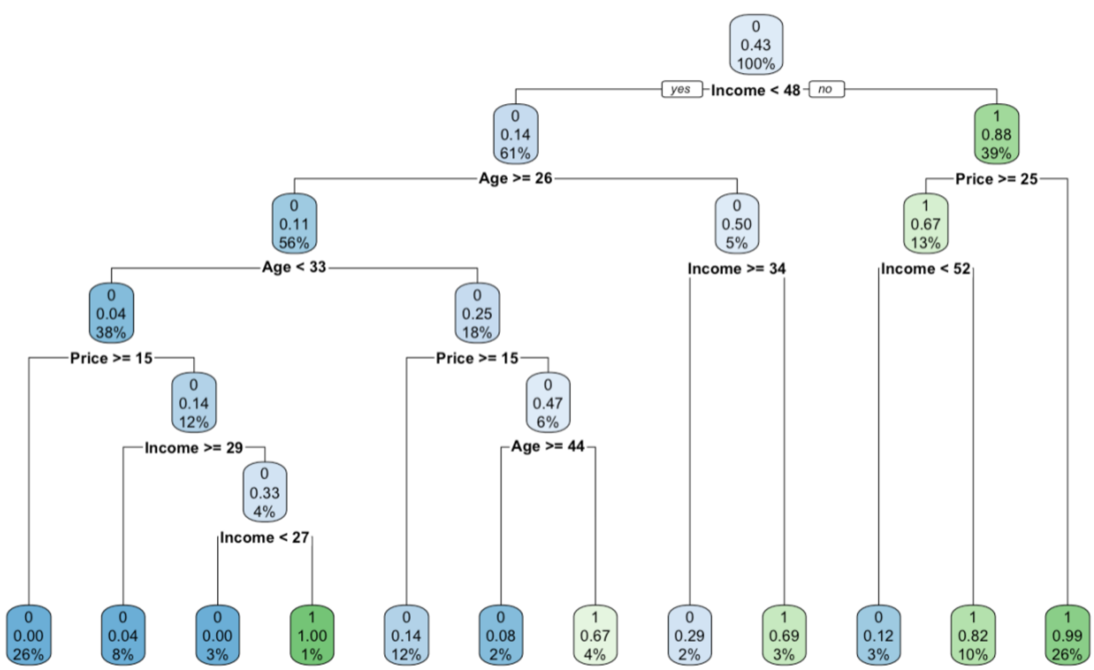

```
Classification tree:
rpart(formula = as.factor(MYDEPV) ~ Price + Income + Age, data = data_train,
    method = "class", parms = list(split = "gini"))

Variables actually used in tree construction:
[1] Age    Income Price

Root node error: 260/600 = 0.43333

n= 600

        CP nsplit rel error  xerror     xstd
1 0.692308      0   1.00000 1.00000 0.046685
2 0.025000      1   0.30769 0.31154 0.032194
3 0.011538      3   0.25769 0.26923 0.030244
4 0.010256      5   0.23462 0.27692 0.030615
5 0.010000     11   0.17308 0.26538 0.030055
```



h. Pruning is a technique that reduces the size/depth of a decision tree by removing sections with low classification power, which helps reduce overfitting and simplifies the model, reducing the computational cost. One way to prune a tree is according to the complexity parameter associated with the smallest cross-validation error. Prune the new tree in this way using the “prune” function. Which features were actually used in the pruned tree? Why were certain variables not used?

Based on the results of step a, cross-validation error min when cp = 0.011538

```{r}
pruned <- prune(decision_tree, cp=0.011538)
printcp(pruned)
rpart.plot(pruned, extra = 106)
```

R Console

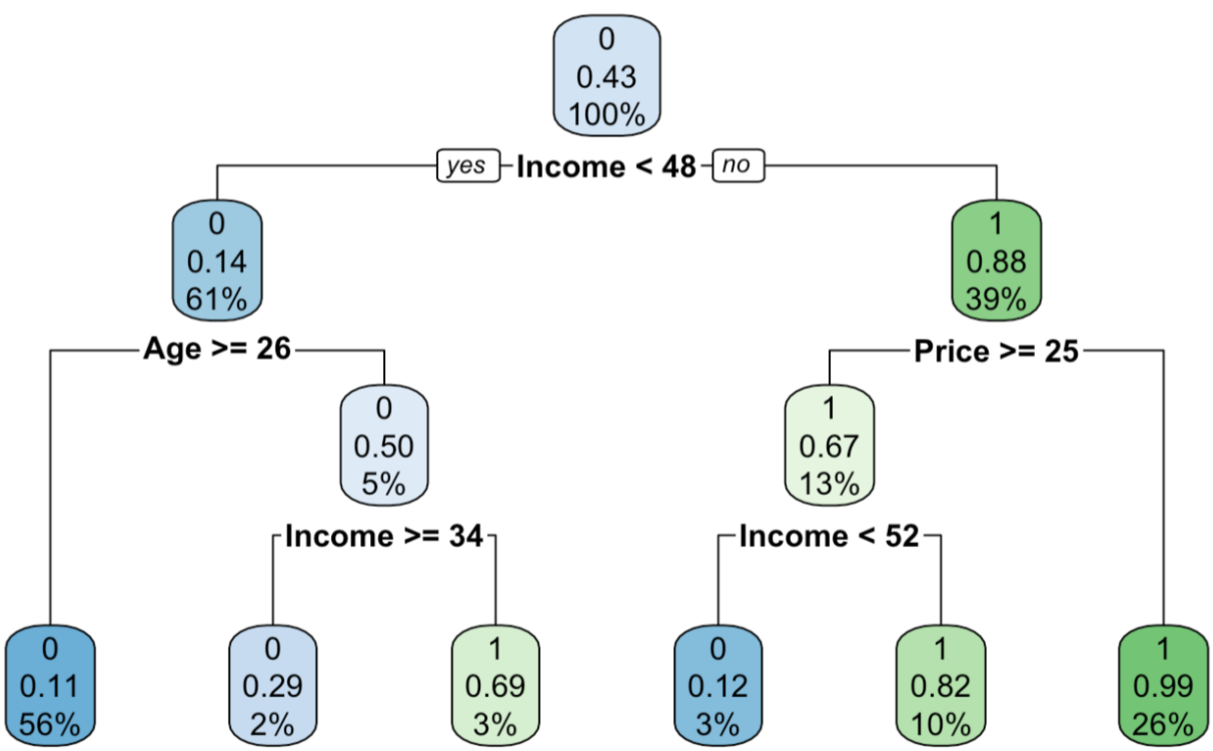
```
Classification tree:
rpart(formula = as.factor(MYDEPV) ~ Price + Income + Age, data = data_train,
    method = "class", parms = list(split = "information"))

Variables actually used in tree construction:
[1] Age    Income Price

Root node error: 260/600 = 0.43333

n= 600

        CP nsplit rel error  xerror     xstd
1 0.692308      0   1.00000 1.00000 0.046685
2 0.025000      1   0.30769 0.31154 0.032194
3 0.011538      3   0.25769 0.28462 0.030978
4 0.011538      5   0.23462 0.28462 0.030978
```



````
#### Features were actually used to construct the tree: Age, Income, Price

#### There are 5 internal nodes in the tree, and the tree high is 3.

### i. Create the confusion matrix for the new model, and compare the performance of the model before and after pruning.

```{r}
Pred <- predict(pruned, data_train, type = 'class')
Matrix <- confusionMatrix(Pred, as.factor(data_train$MYDEPV))
Matrix
```
````

```
Confusion Matrix and Statistics

          Reference
Prediction   0   1
         0 322  43
         1  18 217

               Accuracy : 0.8983
                 95% CI : (0.8713, 0.9213)
    No Information Rate : 0.5667
    P-Value [Acc > NIR] : < 2e-16

                  Kappa : 0.7906

 Mcnemar's Test P-Value : 0.00212

            Sensitivity : 0.9471
            Specificity : 0.8346
         Pos Pred Value : 0.8822
         Neg Pred Value : 0.9234
             Prevalence : 0.5667
         Detection Rate : 0.5367
   Detection Prevalence : 0.6083
      Balanced Accuracy : 0.8908

       'Positive' Class : 0
```

```
#### The zero class missclassification rate: 18/(18+322) = `r 18/(18+322)`

#### The one class missclassification rate: 43/(43+217) = `r 43/(43+217)`

#### The overall missclassification rate: (18+43)/600 = `r (18+43)/600`

#### Overall model performance is slightly deteriorated, but essentially they are same
```

```
############## Exercise 4.2 ##############
## *** Part I ***

###  (Naïve Bayes) In this assignment you will train a Naïve Bayes classifier on categorical data and predict individuals' incomes.

### a. Import the nbtrain.csv file. Use the first 9010 records as training data and the remaining 1000 records as testing data.

```{r}
#Import nbtrain.csv
data = read.csv("Dataset/nbtrain.csv")
#Split training data vs testing data
library(caret)
data_train <- head(data, 9010)
data_test <- tail(data, 1000)
```
b. Construct the Naïve Bayes classifier from the training data, according to the formula "income ~ age + sex + educ". To do this, use the "naiveBayes" function from the "e1071" package. Provide the model's a priori and conditional probabilities.
```{r}
library(e1071)
NBclassfier <- naiveBayes(as.factor(income) ~ age + sex + educ, data=data_train)
NBclassfier
```
```

```

Naive Bayes Classifier for Discrete Predictors

Call:
naiveBayes.default(x = X, y = Y, laplace = laplace)

A-priori probabilities:
Y
    10-50K     50-80K     GT 80K 
0.80266371 0.12563818 0.07169811 

Conditional probabilities:
        age
Y             20-30      31-45      GT 45
  10-50K 0.20796460 0.34457965 0.44745575
  50-80K 0.08303887 0.39752650 0.51943463
  GT 80K 0.06811146 0.34055728 0.59133127

        sex
Y                F         M
  10-50K 0.4798119 0.5201881
  50-80K 0.2871025 0.7128975
  GT 80K 0.2058824 0.7941176

        educ
Y           College     Others   Prof/Phd
  10-50K 0.24585177 0.73976770 0.01438053
  50-80K 0.49558304 0.44257951 0.06183746
  GT 80K 0.53869969 0.29566563 0.16563467
```

A-priori probabilities:

- Income is in the range 10-50K: 0.803

- Income is in the range 50-80K: 0.126

- Income is in the range GT 80K: 0.072

Conditional probabilities

Age

```{r}
NBclassfier$tables$age
```

```
        age
Y             20-30      31-45      GT 45
  10-50K 0.20796460 0.34457965 0.44745575
  50-80K 0.08303887 0.39752650 0.51943463
  GT 80K 0.06811146 0.34055728 0.59133127
```

Sex

```{r}
NBclassfier$tables$sex
```

```
        sex
Y                F         M
  10-50K 0.4798119 0.5201881
  50-80K 0.2871025 0.7128975
  GT 80K 0.2058824 0.7941176
```

Education

```{r}
NBclassfier$tables$educ
```

```
        educ
Y           College     Others   Prof/Phd
  10-50K 0.24585177 0.73976770 0.01438053
  50-80K 0.49558304 0.44257951 0.06183746
  GT 80K 0.53869969 0.29566563 0.16563467
```

c. Score the model with the testing data and create the model’s confusion matrix. Also, calculate the overall, 10-50K, 50-80K, and GT 80K misclassification rates. Explain the variation in the model’s predictive power across income classes.

```{r}
testPred <- predict(NBclassfier, data_test, type="class")
message("Confusion Matrix for Test Data")
Matrix <- confusionMatrix(testPred, as.factor(data_test$income))
Matrix
```

```
Confusion Matrix for Test Data
Confusion Matrix and Statistics

          Reference
Prediction 10-50K 50-80K GT 80K
    10-50K    787    127     67
    50-80K      0      0      0
    GT 80K      6      5      8

Overall Statistics

               Accuracy : 0.795
                 95% CI : (0.7686, 0.8196)
    No Information Rate : 0.793
    P-Value [Acc > NIR] : 0.4564

                  Kappa : 0.0709

 Mcnemar's Test P-Value : <2e-16

Statistics by Class:

                     Class: 10-50K Class: 50-80K Class: GT 80K
Sensitivity                 0.9924         0.000        0.1067
Specificity                 0.0628         1.000        0.9881
Pos Pred Value              0.8022           NaN        0.4211
Neg Pred Value              0.6842         0.868        0.9317
Prevalence                  0.7930         0.132        0.0750
Detection Rate              0.7870         0.000        0.0080
Detection Prevalence        0.9810         0.000        0.0190
Balanced Accuracy           0.5276         0.500        0.5474
```

The overall misclassification rate: 1 - Accuracy = `r 1 - Matrix$overall[1]`

```{r}
library(shipunov)
Misclass(testPred, as.factor(data_test$income))
```

```
package 'shipunov', version 1.17
Classification table:
        obs
pred     10-50K 50-80K GT 80K
  10-50K    787    127     67
  50-80K      0      0      0
  GT 80K      6      5      8
Misclassification errors (%):
10-50K 50-80K GT 80K
   0.8  100.0   89.3
Mean misclassification error: 63.4%
```

- The 10-50K misclassification rate: 0.8%

- The 50-80K misclassification rate: 100%

- The GT 80K misclassification rate: 89.3%

In this model variation is explaeined mostly by confusion matrix

*** Part II ***

a. Construct the classifier according to the formula “sex ~ age + educ + income”, and calculate the overall, female, and male misclassification rates. Explain the misclassification rates?

```{r}
NBclassfier <- naiveBayes(as.factor(sex) ~ age + income + educ, data=data_train)
NBclassfier
testPred <- predict(NBclassfier, data_test, type="class")
Matrix <- confusionMatrix(testPred, as.factor(data_test$sex))
```

```
Naive Bayes Classifier for Discrete Predictors

Call:
naiveBayes.default(x = X, y = Y, laplace = laplace)

A-priori probabilities:
Y
      F       M
0.43596 0.56404

Conditional probabilities:
   age
Y       20-30     31-45     GT 45
  F 0.1802444 0.3475051 0.4722505
  M 0.1837859 0.3536009 0.4626131

   income
Y       10-50K     50-80K     GT 80K
  F 0.88340122 0.08273931 0.03385947
  M 0.74025974 0.15879575 0.10094451

   educ
Y      College     Others    Prof/Phd
  F 0.32128310 0.65707739 0.02163951
  M 0.28040142 0.68103109 0.03856749
```

The overall misclassification rate: 1 - Accuracy = `r 1 - Matrix$overall[1]`

```{r}
Misclass(testPred, as.factor(data_test$sex))
```

```
Classification table:
    obs
pred   F   M
   F 106  97
   M 321 476
Misclassification errors (%):
   F    M
75.2 16.9
Mean misclassification error: 46.1%
```

- The female misclassification rate: 75.2%

- The male misclassification rate: 16.9%

b. Divide the training data into two partitions, according to sex, and randomly select 3500 records from each partition. Reconstruct the model from part (a) from these 7000 records. Provide the model’s a priori and conditional probabilities.

```{r message= FALSE}
library(dplyr)
#Divide the training data into two partitions
data_female = subset(data_train, data_train$sex == 'F')
data_male = subset(data_train, data_train$sex=='M')
#Randomly select 3500 records from each partition
data_female = sample_n(data_female, 3500)
data_male = sample_n(data_male, 3500)
new_data = rbind(data_male,data_female)
model <- naiveBayes(as.factor(sex) ~ age + income + educ, data=new_data)
message("Model Navie Bayes Classifier")
model
```

```
Naive Bayes Classifier for Discrete Predictors

Call:
naiveBayes.default(x = X, y = Y, laplace = laplace)

A-priori probabilities:
Y
  F   M
0.5 0.5

Conditional probabilities:
   age
Y       20-30     31-45     GT 45
  F 0.1800000 0.3468571 0.4731429
  M 0.1842857 0.3534286 0.4622857

   income
Y       10-50K     50-80K      GT 80K
  F 0.88285714 0.08114286 0.03600000
  M 0.73742857 0.16285714 0.09971429

   educ
Y      College     Others   Prof/Phd
  F 0.32114286 0.65742857 0.02142857
  M 0.27628571 0.68457143 0.03914286
```

The a priori probabilities are equal and the conditional probabilities are very similar.

c. How well does the model classify the testing data?

```{r}
testPred <- predict(model, data_test, type="class")
Matrix <- confusionMatrix(testPred, as.factor(data_test$sex))
Matrix$table
message("Accuracy")
Matrix$overall[1]
```

```
          Reference
Prediction   F   M
         F 369 412
         M  58 161
Accuracy
Accuracy
    0.53
```

d. Repeat step (b) 4 several times. What effect does the random selection of records have on the model's performance?

1.

```{r}
#Divide the training data into two partitions
data_female = subset(data_train, data_train$sex == 'F')
data_male = subset(data_train, data_train$sex=='M')
#Randomly select 3500 records from each partition
data_female = sample_n(data_female, 3500)
data_male = sample_n(data_male, 3500)
new_data = rbind(data_male,data_female)
model <- naiveBayes(as.factor(sex) ~ age + income + educ, data=new_data)
model
testPred <- predict(model, data_test, type="class")
Matrix <- confusionMatrix(testPred, as.factor(data_test$sex))
message("Accuracy")
Matrix$overall[1]
```

```
Naive Bayes Classifier for Discrete Predictors

Call:
naiveBayes.default(x = X, y = Y, laplace = laplace)

A-priori probabilities:
Y
  F   M
0.5 0.5

Conditional probabilities:
   age
Y       20-30     31-45     GT 45
  F 0.1805714 0.3454286 0.4740000
  M 0.1891429 0.3502857 0.4605714

   income
Y       10-50K      50-80K      GT 80K
  F 0.88657143 0.08171429 0.03171429
  M 0.73457143 0.16600000 0.09942857

   educ
Y      College      Others    Prof/Phd
  F 0.31885714 0.65885714 0.02228571
  M 0.27714286 0.68257143 0.04028571

Accuracy
Accuracy
    0.53
```

2.

```{r}
#Divide the training data into two partitions
data_female = subset(data_train, data_train$sex == 'F')
data_male = subset(data_train, data_train$sex=='M')
#Randomly select 3500 records from each partition
data_female = sample_n(data_female, 3500)
data_male = sample_n(data_male, 3500)
new_data = rbind(data_male,data_female)
model <- naiveBayes(as.factor(sex) ~ age + income + educ, data=new_data)
model
testPred <- predict(model, data_test, type="class")
Matrix <- confusionMatrix(testPred, as.factor(data_test$sex))
message("Accuracy")
Matrix$overall[1]
```

```
Naive Bayes Classifier for Discrete Predictors

Call:
naiveBayes.default(x = X, y = Y, laplace = laplace)

A-priori probabilities:
Y
  F   M
0.5 0.5

Conditional probabilities:
   age
Y       20-30     31-45     GT 45
  F 0.1817143 0.3477143 0.4705714
  M 0.1860000 0.3477143 0.4662857

   income
Y       10-50K      50-80K      GT 80K
  F 0.88400000 0.08057143 0.03542857
  M 0.73942857 0.16200000 0.09857143

   educ
Y      College      Others    Prof/Phd
  F 0.32085714 0.65800000 0.02114286
  M 0.28000000 0.68200000 0.03800000

Accuracy
Accuracy
    0.53
```

3.

```
```{r}
#Divide the training data into two partitions
data_female = subset(data_train, data_train$sex == 'F')
data_male = subset(data_train, data_train$sex=='M')
#Randomly select 3500 records from each partition
data_female = sample_n(data_female, 3500)
data_male = sample_n(data_male, 3500)
new_data = rbind(data_male,data_female)
model <- naiveBayes(as.factor(sex) ~ age + income + educ, data=new_data)
model
testPred <- predict(model, data_test, type="class")
Matrix <- confusionMatrix(testPred, as.factor(data_test$sex))
message("Accuracy")
Matrix$overall[1]
```
```

```
Naive Bayes Classifier for Discrete Predictors

Call:
naiveBayes.default(x = X, y = Y, laplace = laplace)

A-priori probabilities:
Y
  F   M
0.5 0.5

Conditional probabilities:
   age
Y       20-30     31-45     GT 45
  F 0.1811429 0.3514286 0.4674286
  M 0.1854286 0.3577143 0.4568571

   income
Y       10-50K      50-80K      GT 80K
  F 0.88314286 0.08142857 0.03542857
  M 0.74371429 0.15600000 0.10028571

   educ
Y     College    Others  Prof/Phd
  F 0.3194286 0.6585714 0.0220000
  M 0.2837143 0.6782857 0.0380000

Accuracy
Accuracy
    0.53
```

4.

```
```{r}
#Divide the training data into two partitions
data_female = subset(data_train, data_train$sex == 'F')
data_male = subset(data_train, data_train$sex=='M')
#Randomly select 3500 records from each partition
data_female = sample_n(data_female, 3500)
data_male = sample_n(data_male, 3500)
new_data = rbind(data_male,data_female)
model <- naiveBayes(as.factor(sex) ~ age + income + educ, data=new_data)
model
testPred <- predict(model, data_test, type="class")
Matrix <- confusionMatrix(testPred, as.factor(data_test$sex))
message("Accuracy")
Matrix$overall[1]
```
```

```
Naive Bayes Classifier for Discrete Predictors

Call:
naiveBayes.default(x = X, y = Y, laplace = laplace)

A-priori probabilities:
Y
  F   M
0.5 0.5

Conditional probabilities:
   age
Y       20-30     31-45     GT 45
  F 0.1837143 0.3465714 0.4697143
  M 0.1888571 0.3502857 0.4608571

   income
Y       10-50K      50-80K      GT 80K
  F 0.88571429 0.08142857 0.03285714
  M 0.75028571 0.15800000 0.09171429

   educ
Y      College      Others    Prof/Phd
  F 0.32228571 0.65742857 0.02028571
  M 0.28257143 0.68200000 0.03542857

Accuracy
Accuracy
    0.53
```

e. What conclusions can one draw from this exercise?

Conditional probabilities are very close over the entire sample.