Decision Trees

Truong Thi An Hai

12/18/2020

```r
#Import survey.csv
data = read.csv("C:/Users/hp/Desktop/survey.csv")
#Split data
data_train = head(data,600)
data_test = tail(data, 150)
```

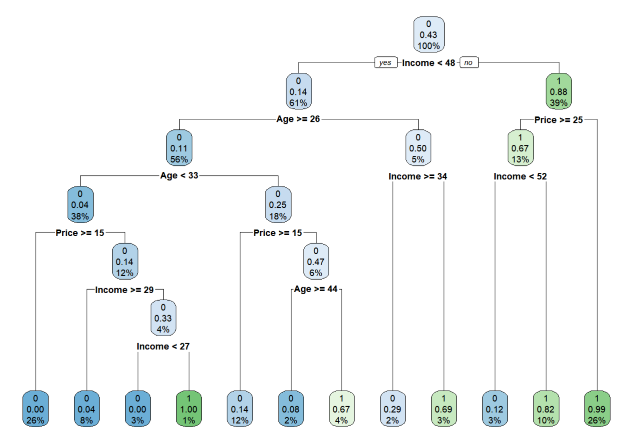
a. Build a classification tree from the training data. Which features were actually used to construct the tree? Plot the tree using the “rpart.plot” package.

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

decision_tree <- rpart(as.factor(MYDEPV) ~ Price + Income + Age, data = data_train, method = 'class', parms = lis
t(split = 'information'))
printcp(decision_tree)
```

```
##
## 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 0.25769 0.029672
## 4 0.010256      5   0.23462 0.25000 0.029281
## 5 0.010000     11   0.17308 0.26154 0.029865
```

```r
#Plot the tree
rpart.plot(decision_tree,extra = 106)
```



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.

b. 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
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) = 0.0764706

The one class missclassification rate: 19/(19+241) = 0.0730769

c. 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.

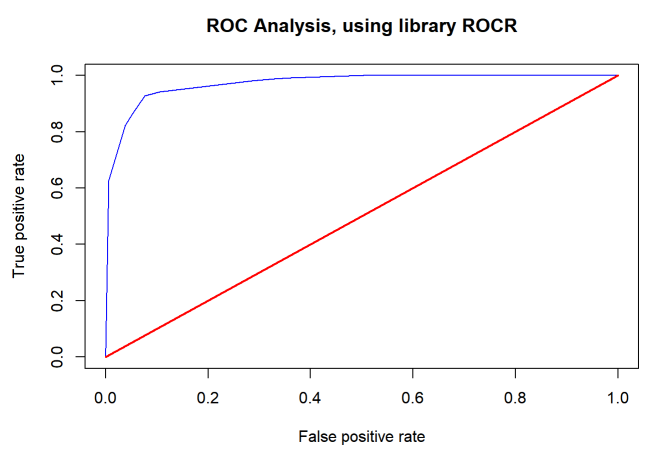
The resubstitution error rate: (19 + 26)/(19 + 26 + 314 +241) = 0.075

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.

d.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)
```




```r
# 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.

e. 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) = 0.1162791

The one class missclassification rate: 6/(6+58) = 0.09375

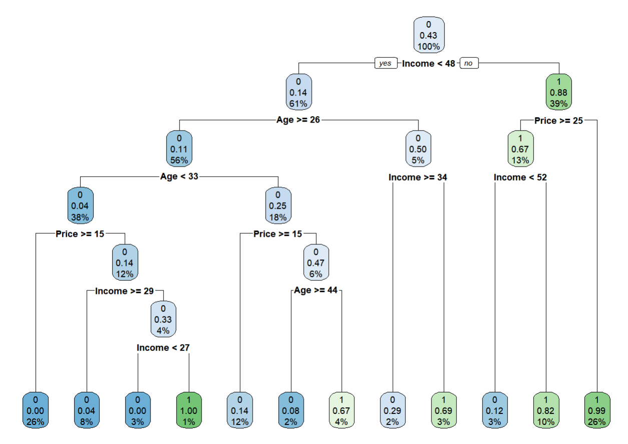
- The model performed well for both classes
- Due to the small amount of testing data, one can conclude that each class was classified almost equally well or bad.

f. 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(sp
lit = 'gini'))
printcp(gini_tree)
```

```
##
## 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 0.25769 0.029672
## 4 0.010256      5   0.23462 0.26154 0.029865
## 5 0.010000     11   0.17308 0.27692 0.030615
```

```r
#Plot the tree
rpart.plot(gini_tree,extra = 106)
```



In this case, the same model is created regardless of our choice of splitting index. That difference/similarity is not generally the case.

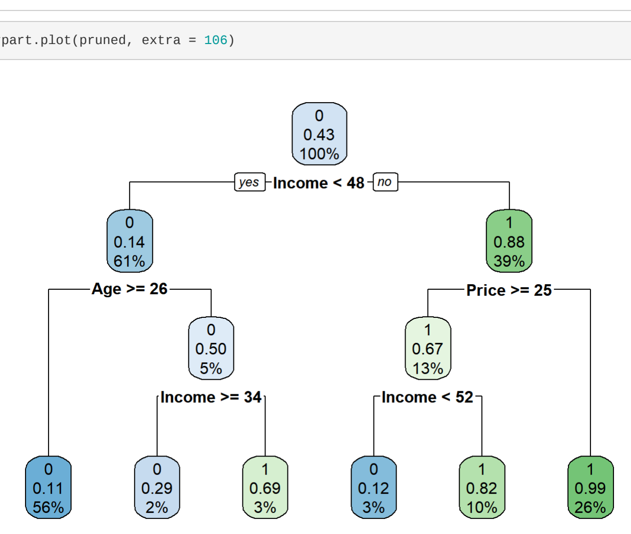
g. Pruning is a technique that reduces the size/depth of a decision tree by removing sections with low classification power, which helps reduce overfitting and simplifies the model, reducing the computational cost. One way to prune a tree is according to the complexity parameter associated with the smallest 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)
```

```
##
## 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 0.25769 0.029672
## 4 0.011538      5   0.23462 0.25000 0.029281
```

```r
rpart.plot(pruned, extra = 106)
```



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.

h. 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) = 0.0529412

The one class missclassification rate: 43/(43+217) = 0.1653846

The overall missclassification rate: (18+43)/600 = 0.1016667

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