# advanced_trees_sample.R

*nicol*

*2020-05-06*

```
### Advanced Trees ### ------------------------------------------------------------------------

# By: Nicole Davila
# Date: 2020-05-06

### Import required libraries
library(ISLR)
```

```
## Warning: package 'ISLR' was built under R version 3.6.3
```

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

```
## Warning: package 'rpart.plot' was built under R version 3.6.1
```

```
library(ROCR)
```

```
## Warning: package 'ROCR' was built under R version 3.6.3

## Loading required package: gplots

## Warning: package 'gplots' was built under R version 3.6.1

##
## Attaching package: 'gplots'

## The following object is masked from 'package:stats':
##
##     lowess
```

```
library(caret)
```

```
## Warning: package 'caret' was built under R version 3.6.1

## Loading required package: lattice

## Loading required package: ggplot2
```

```
library(randomForest)
```

```
## Warning: package 'randomForest' was built under R version 3.6.1
```

```
## randomForest 4.6-14

## Type rfNews() to see new features/changes/bug fixes.

##
## Attaching package: 'randomForest'

## The following object is masked from 'package:ggplot2':
##
##     margin
```

```
library(gbm)
```

```
## Warning: package 'gbm' was built under R version 3.6.1

## Loaded gbm 2.1.5
```

```
### Import Data
head(OJ)
```

```
##   Purchase WeekofPurchase StoreID PriceCH PriceMM DiscCH DiscMM SpecialCH
## 1       CH            237       1    1.75    1.99   0.00    0.0         0
## 2       CH            239       1    1.75    1.99   0.00    0.3         0
## 3       CH            245       1    1.86    2.09   0.17    0.0         0
## 4       MM            227       1    1.69    1.69   0.00    0.0         0
## 5       CH            228       7    1.69    1.69   0.00    0.0         0
## 6       CH            230       7    1.69    1.99   0.00    0.0         0
##   SpecialMM  LoyalCH SalePriceMM SalePriceCH PriceDiff Store7 PctDiscMM
## 1         0 0.500000        1.99        1.75      0.24     No  0.000000
## 2         1 0.600000        1.69        1.75     -0.06     No  0.150754
## 3         0 0.680000        2.09        1.69      0.40     No  0.000000
## 4         0 0.400000        1.69        1.69      0.00     No  0.000000
## 5         0 0.956535        1.69        1.69      0.00    Yes  0.000000
## 6         1 0.965228        1.99        1.69      0.30    Yes  0.000000
##   PctDiscCH ListPriceDiff STORE
## 1  0.000000          0.24     1
## 2  0.000000          0.24     1
## 3  0.091398          0.23     1
## 4  0.000000          0.00     1
## 5  0.000000          0.00     0
## 6  0.000000          0.30     0
```

```
OJ=OJ

### Split the data into a train and test sample such that 70% of the data is in the train sample
set.seed(1234)
split=sample.split(Y=OJ$Purchase,SplitRatio=0.7)
train=OJ[split,]
test=OJ[!split,]

### Let's take a look at how many observations we have on our train sample
nrow(train)
```

```
## [1] 749
```

```
# 749

### How many Minute Maid purchases were made in the trian sample?
sum(train$Purchase=="MM")
```

```
## [1] 292
```

```
# 292

### Let's take a lok at the average price for Minute Maid in the train sample.
mean(train$PriceMM)
```

```
## [1] 2.087223
```

```
# 2.087223

### How about the average discount?
mean(train$DiscMM)
```

```
## [1] 0.1237116
```

```
# 0.1237116

### How many purchases of Minute Maid were made in Week 275?
nrow(train[which(train$Purchase=="MM" & train$WeekofPurchase==275),])
```

```
## [1] 17
```

```
# 17

### Let's construct a classification tree to predict "Purchase"
tree1=rpart(Purchase~PriceCH+PriceMM+DiscCH+DiscMM+SpecialCH+SpecialMM+LoyalCH+PriceDiff+PctDiscMM+PctDi
# Do predictions
pred1=predict(tree1,newdata=test)
ROCRpred1=prediction(pred1[,2],test$Purchase)
# Calculate AUC
as.numeric(performance(ROCRpred1,"auc")@y.values)
```

```
## [1] 0.8628776
```

```
# 0.8628776

### Let's tune the model using a 10-fold cross-validation and test cp values ranging from 0 to 0.1 in s
trControl=trainControl(method="cv", number=10)
tuneGrid = expand.grid(.cp = seq(0,0.1,0.001))
set.seed(100)
cvModel=train(Purchase~PriceCH+PriceMM+DiscCH+DiscMM+SpecialCH+SpecialMM+LoyalCH+PriceDiff+PctDiscMM+Pct
              method="rpart",trControl=trControl,tuneGrid=tuneGrid)
cvModel$bestTune
```

```
##      cp
## 6 0.005
```

```
# We see that the optimal cp is of 0.004

### Rerun the tree model but using the optimal cp value. What is the auc for this model on the test sam
treeCV=rpart(Purchase~PriceCH+PriceMM+DiscCH+DiscMM+SpecialCH+SpecialMM+LoyalCH+PriceDiff+PctDiscMM+Pct
             control=rpart.control(cp=cvModel$bestTune))
predTreeCV=predict(treeCV,newdata=test)
ROCRpredTreeCV=prediction(predTreeCV[,2],test$Purchase)
as.numeric(performance(ROCRpredTreeCV,"auc")@y.values)
```

```
## [1] 0.8628776
```

```
# 0.8628776, same as above

### Let's construct a bag model, using 1000 trees.
set.seed(100)
bag=randomForest(Purchase~PriceCH+PriceMM+DiscCH+DiscMM+SpecialCH+SpecialMM+LoyalCH+PriceDiff+PctDiscMM
                 mtry=10,ntree=1000)
# In order to get the AUC, we'll need to get the prediction probability for each prediction. To achieve
# and the second column of the output in the predict function.
predBag=predict(bag,newdata=test,type="prob")
ROCRpredBag=prediction(predBag[,2],test$Purchase)
as.numeric(performance(ROCRpredBag,"auc")@y.values)
```

```
## [1] 0.867102
```

```
# We get an AUC of 0.867102 on the test sample

### Now, let's construct a random forest model, using 1000 trees, but using the default instead of esta
#   for the bag model, use argument type = "prob" for the predict function and use the second column of
set.seed(100)
forest=randomForest(Purchase~PriceCH+PriceMM+DiscCH+DiscMM+SpecialCH+SpecialMM+LoyalCH+PriceDiff+PctDis
# Determine the AUC for the test sample
predForest=predict(forest,newdata=test,type="prob")
ROCRpredForest=prediction(predForest[,2],test$Purchase)
as.numeric(performance(ROCRpredForest,"auc")@y.values)
```

```
## [1] 0.8812449
```

```
# 0.8812449

### In this dataset, the levels of variable Purchase are 1 and 2. However, in order to run a boosting m
#   model, the dependent variable will need to have values of 0 and 1. Let's modify the variable and cr
train$Purchase2 = as.numeric(train$Purchase)-1
test$Purchase2 = as.numeric(test$Purchase)-1

### Let's run a gradient boosting model (gbm) with 1000 trees using Purchase2 instead of Purchase and u
#   used in above models. Let's set distribution to "bernoulli", interaction depth to 1, and shrinkage
#   argument type = "response" the predict function and n.trees = 100.
set.seed(100)
```

```
boost=gbm(Purchase2~PriceCH+PriceMM+DiscCH+DiscMM+SpecialCH+SpecialMM+LoyalCH+PriceDiff+PctDiscMM+PctDis
predBoost=predict(boost,newdata=test,n.trees=100,type="response")
ROCRpredBoost=prediction(predBoost,test$Purchase2)
# Determine the AUC for the test sample
as.numeric(performance(ROCRpredBoost,"auc")@y.values)
```

```
## [1] 0.879551
```

```
# 0.879551
```