Assign2-2_Small

```
library(caret)
library(ggplot2)
library(e1071)
library(tree)
library(rpart)
```

Reading and cleaning the data

```
data <- read.csv("/Users/Nick/Desktop/eBay.csv", stringsAsFactors=FALSE)
Duration <- as.factor(data$Duration)
Competitive <- as.factor(data$Competitive)
data <- cbind(Duration, data[,-4])
data <- cbind(Competitive, data[,-8])
str(data)
```

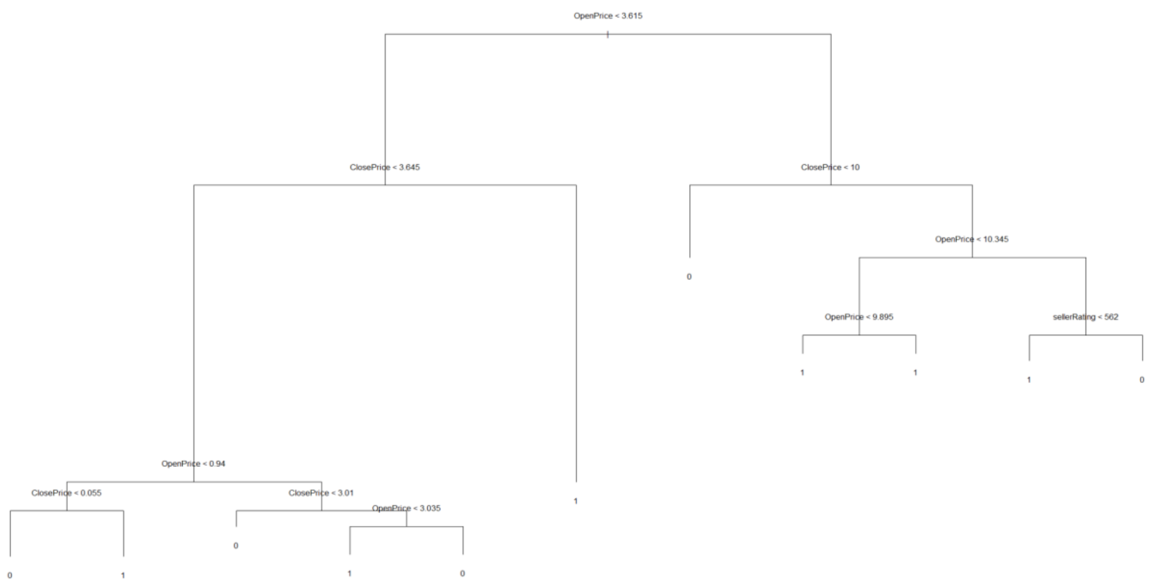
```
## 'data.frame':    1972 obs. of  8 variables:
##  $ Competitive : Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 1 1 1 ...
##  $ Duration    : Factor w/ 5 levels "1","3","5","7",..: 3 3 3 3 3 3 3 3 3
3 ...
##  $ Category    : chr  "Music/Movie/Game" "Music/Movie/Game" "Music/Movie/G
ame" "Music/Movie/Game" ...
##  $ currency    : chr  "US" "US" "US" "US" ...
##  $ sellerRating: int  3249 3249 3249 3249 3249 3249 3249 3249 3249 3249 ..
.
##  $ endDay      : chr  "Mon" "Mon" "Mon" "Mon" ...
##  $ ClosePrice  : num  0.01 0.01 0.01 0.01 0.01 0.01 0.01 0.01 0.01 0.01 ..
.
##  $ OpenPrice   : num  0.01 0.01 0.01 0.01 0.01 0.01 0.01 0.01 0.01 0.01 ..
```

The Dataset is split using a createDataPartition function which randomly samples the specified 70% of the data in the train set and 30% of the data into the test set. We will be determining if the model developed is competitive or not.

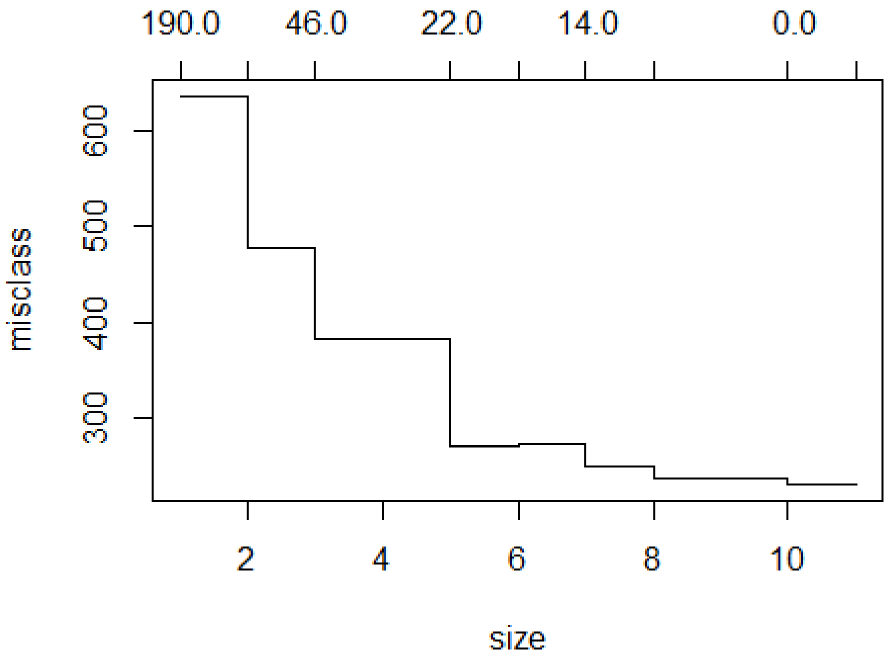
```
set.seed(1234)
inTrain <- createDataPartition(y=data$Competitive, p=0.7, list=FALSE)
trainSet <- data[inTrain,]
testSet <- data[-inTrain,]
```

```
treemod <- tree(Competitive~., data=trainSet)
```

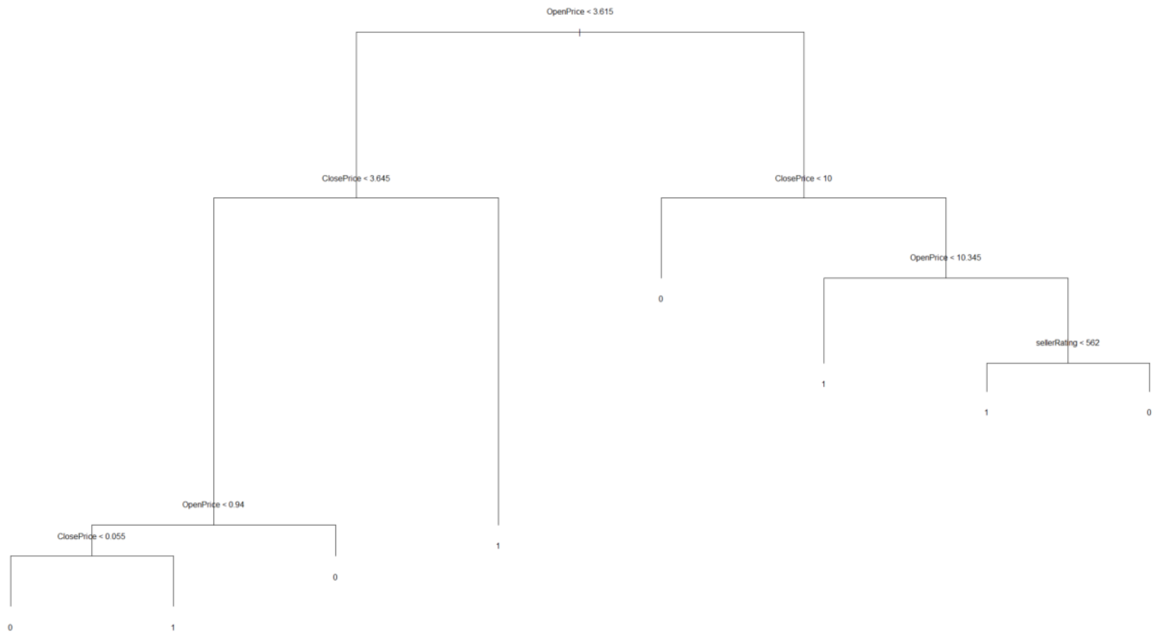
```
plot(treemod)
text(treemod)
```



```
cv.trees <- cv.tree(treemod, FUN=prune.misclass)
plot(cv.trees)
```



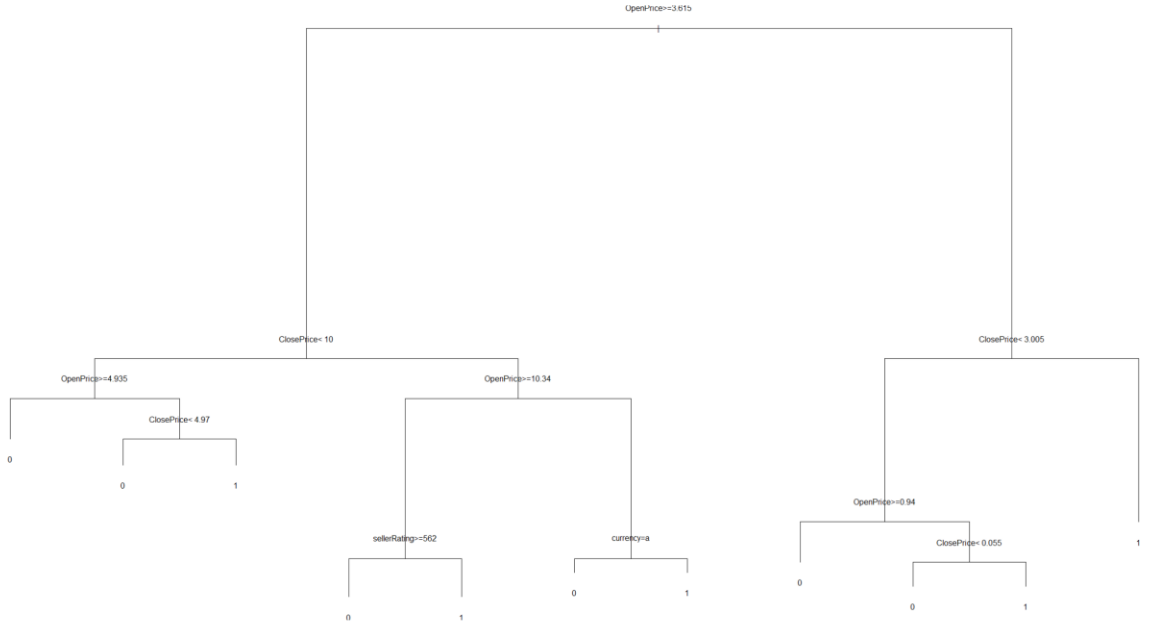
```
prune.trees <- prune.misclass(treemod, best=8)
plot(prune.trees)
text(prune.trees, pretty=0)
```



```
treepred <- predict(prune.trees, testSet, type='class')
confusionMatrix(treepred, testSet$Competitive)
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 249  77
##          1  22 242
##
##                Accuracy : 0.8322
##                  95% CI : (0.7996, 0.8615)
##     No Information Rate : 0.5407
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.6673
##
##  Mcnemar's Test P-Value : 5.724e-08
##
##             Sensitivity : 0.9188
##             Specificity : 0.7586
##          Pos Pred Value : 0.7638
##          Neg Pred Value : 0.9167
##              Prevalence : 0.4593
##          Detection Rate : 0.4220
##    Detection Prevalence : 0.5525
##       Balanced Accuracy : 0.8387
##
##        'Positive' Class : 0
```

The Tree confusion matrix shows the decision tree model is 83% accurate. There are 249 True positives, 242 True Negatives, 77 False Negatives, and 22 False Positives. This is a very good model which shows it is competitive

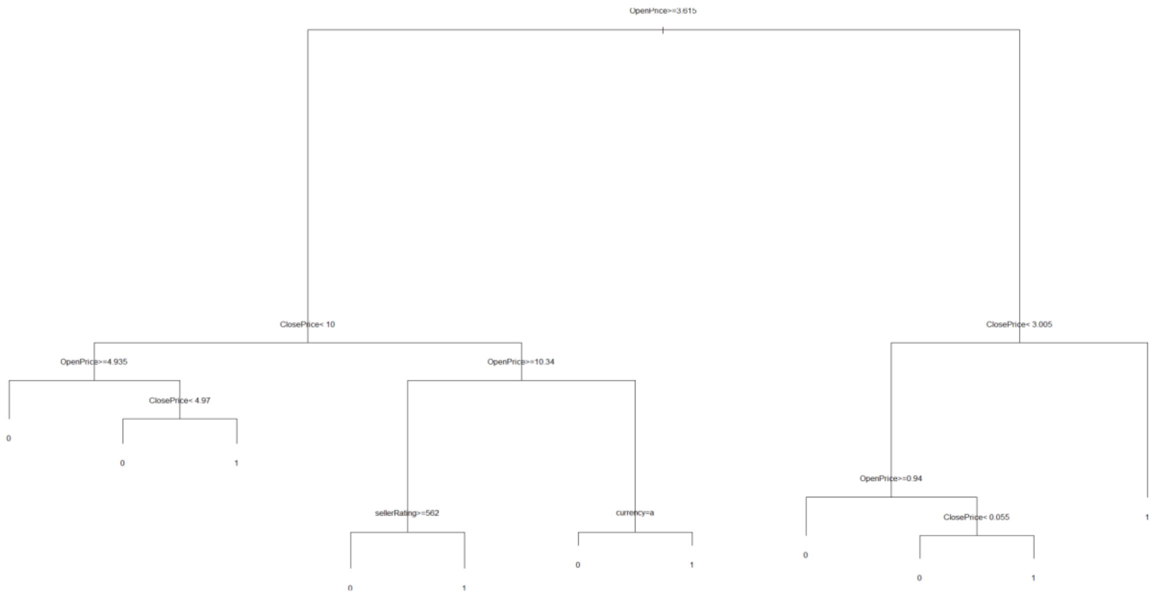
```
rpartmod <- rpart(Competitive~., data=trainSet, method ="class")
plot(rpartmod)
text(rpartmod)
```



```
printcp(rpartmod)

##
## Classification tree:
## rpart(formula = Competitive ~ ., data = trainSet, method = "class")
##
## Variables actually used in tree construction:
## [1] ClosePrice   currency     OpenPrice    sellerRating
##
## Root node error: 635/1382 = 0.45948
##
## n= 1382
##
##          CP nsplit rel error  xerror     xstd
## 1 0.299213      0   1.00000 1.00000 0.029176
## 2 0.148031      1   0.70079 0.70236 0.027370
## 3 0.072441      2   0.55276 0.55433 0.025507
## 4 0.034646      4   0.40787 0.41102 0.022914
## 5 0.018898      5   0.37323 0.38110 0.022250
## 6 0.012598      7   0.33543 0.34803 0.021458
## 7 0.011811      8   0.32283 0.33386 0.021098
## 8 0.010000     10   0.29921 0.33228 0.021057
```

```r
ptree <- prune(rpartmod, cp=rpartmod$cptable[which.min(rpartmod$cptable[,"xer
ror"]),"CP"])
plot(ptree)
text(ptree)
```



```r
rpartpred <- predict(ptree, testSet, type='class')
confusionMatrix(rpartpred, testSet$Competitive)
```

```
## Confusion Matrix and Statistics
##
##             Reference
## Prediction   0   1
##          0 251  69
##          1  20 250
##
##                Accuracy : 0.8492
##                  95% CI : (0.8177, 0.8771)
##     No Information Rate : 0.5407
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.7004
##
##  Mcnemar's Test P-Value : 3.619e-07
##
##             Sensitivity : 0.9262
##             Specificity : 0.7837
##          Pos Pred Value : 0.7844
##          Neg Pred Value : 0.9259
##              Prevalence : 0.4593
##          Detection Rate : 0.4254
##    Detection Prevalence : 0.5424
##       Balanced Accuracy : 0.8549
##
##        'Positive' Class : 0
```

Using the Rpart Prediction and confusion Matrix we can observe that this model is 84% accurate. There are 251 True Positives, 250 True Negatives, 69 False Negatives, and 20 False Positives. This model is the best showing an increase in 1% in accuracy. This shows the model is very competitive.