Classification Tree Rmd

JackMoorer

```
train <- read.csv("/Users/jackmoorer/Stat154/Projects/Project/data/clean_train.csv", header = TRUE)
test <- read.csv("/Users/jackmoorer/Stat154/Projects/Project/data/clean_test.csv", header = TRUE)
```

```
library(ISLR)
library(tree)
library(rpart)
```

```
train_tree <- train[,-ncol(train)]
classification_tree <- tree(Over50k ~., data = train_tree)
summary(classification_tree)
```

```
##
## Classification tree:
## tree(formula = Over50k ~ ., data = train_tree)
## Variables actually used in tree construction:
## [1] "relationship" "capital_gain" "education"    "occupation"
## Number of terminal nodes:  8
## Residual mean deviance:  0.7118 = 21460 / 30150
## Misclassification error rate: 0.1589 = 4794 / 30162
```

Now I will run cross validation on the tree using the function prune.misclass.

```
set.seed(100)
classification_tree_cv <- cv.tree(classification_tree, FUN = prune.misclass)
```

We can compare the size of the tree or the cost complexicity, k, of the tree with dev

```
classification_tree_cv
```

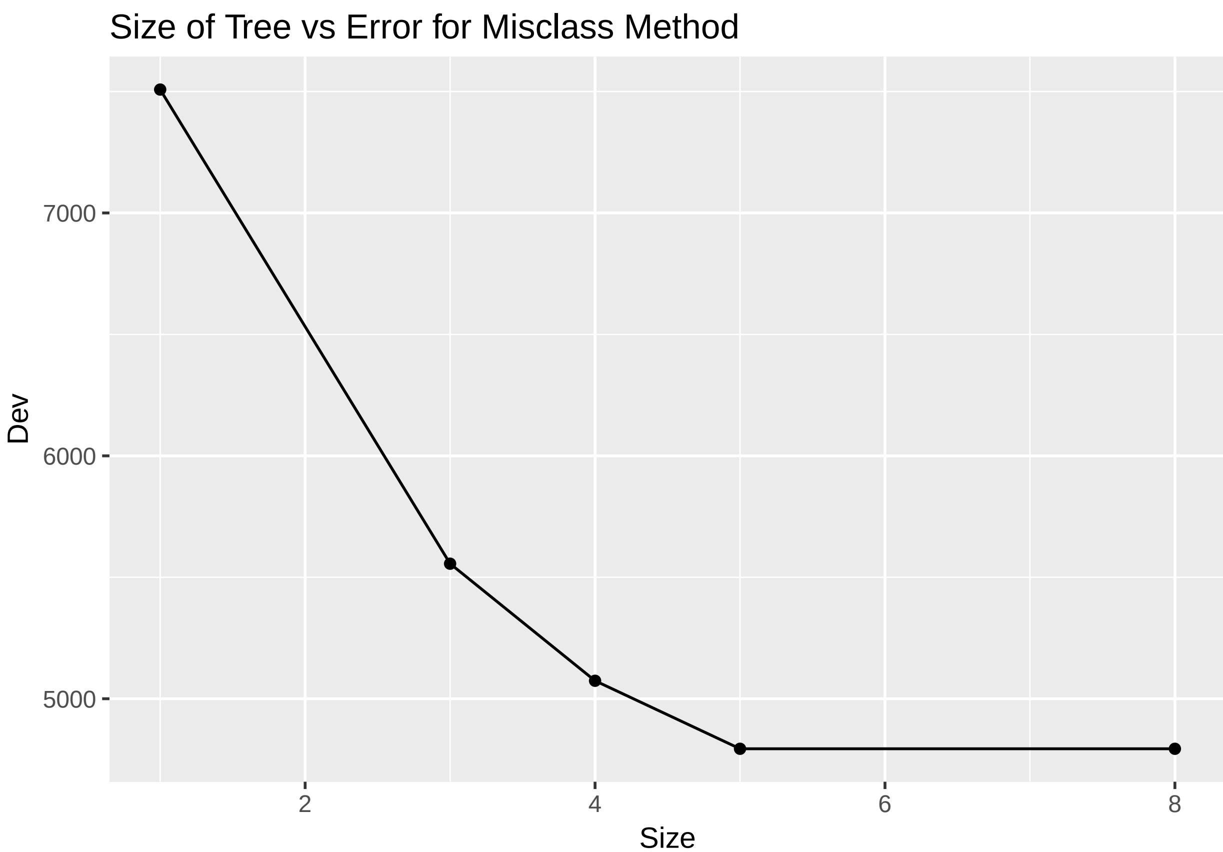
```
## $size
## [1] 8 5 4 3 1
##
## $dev
## [1] 4794 4794 5074 5556 7508
##
## $k
## [1] -Inf    0  280  482  976
##
## $method
## [1] "misclass"
##
## attr(,"class")
## [1] "prune"         "tree.sequence"
```

We can plot this

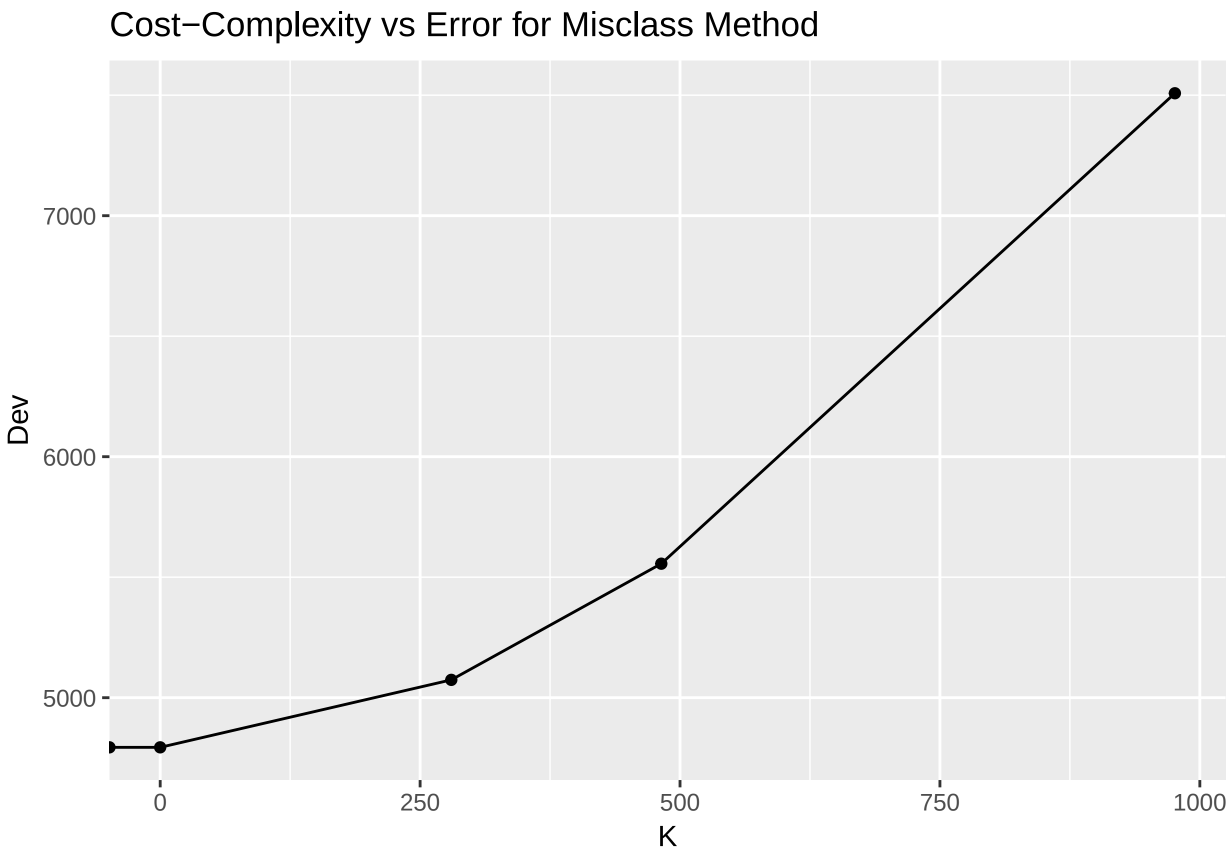
```
library(ggplot2)
Size <- classification_tree_cv$size
K <- classification_tree_cv$k
Dev <- classification_tree_cv$dev
Misclass <- data.frame(Size, K, Dev)
Misclass
```

```
##   Size    K  Dev
## 1    8 -Inf 4794
## 2    5    0 4794
## 3    4  280 5074
## 4    3  482 5556
## 5    1  976 7508
```

```r
ggplot(data = Misclass, aes(x = Size, y = Dev)) + geom_point() + geom_line() + ggtitle("Size of Tree vs
```




```r
ggplot(data = Misclass, aes(x = K, y = Dev)) + geom_point() + geom_line() + ggtitle("Cost-Complexity vs
```



We could use k = 0 as a cost complexicity hyper-parameter. However, if we look at number of trees using 5 or 8 both give us the same minimum. We can also look at a different type of cross validation approach using the default prune.tree function from cv.tree().

```
set.seed(200)
classification_tree_cv_default <- cv.tree(classification_tree, FUN = prune.tree)
```

```
classification_tree_cv_default
```

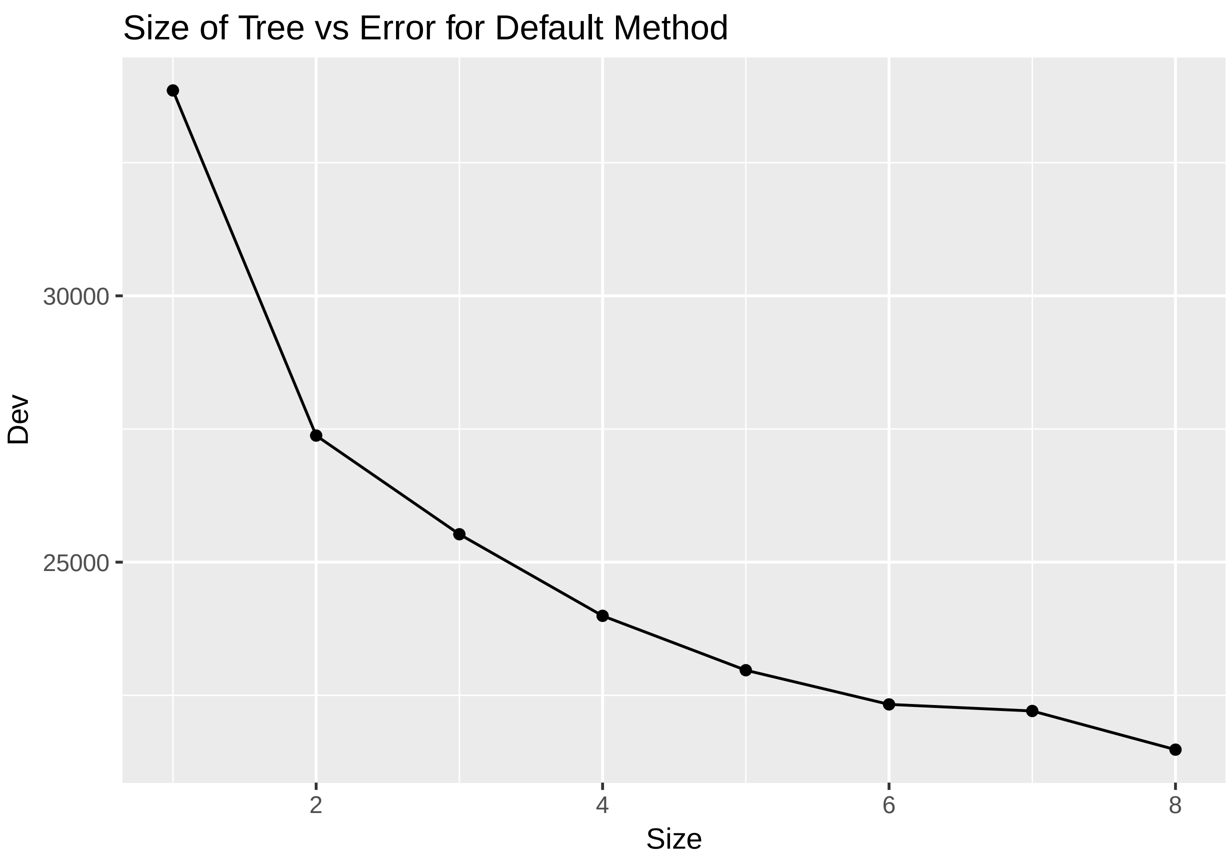
```
## $size
## [1] 8 7 6 5 4 3 2 1
##
## $dev
## [1] 21480.40 22206.32 22330.03 22971.42 23992.21 25524.19 27376.34 33855.37
##
## $k
## [1]      -Inf  408.9358  445.8046  643.1663 1021.8448 1534.7488 1854.1982
## [8] 6478.6784
##
## $method
## [1] "deviance"
##
## attr(,"class")
## [1] "prune"         "tree.sequence"
```

```
Size <- classification_tree_cv_default$size
K <- classification_tree_cv_default$k
Dev <- classification_tree_cv_default$dev
default <- data.frame(Size, K, Dev)
```

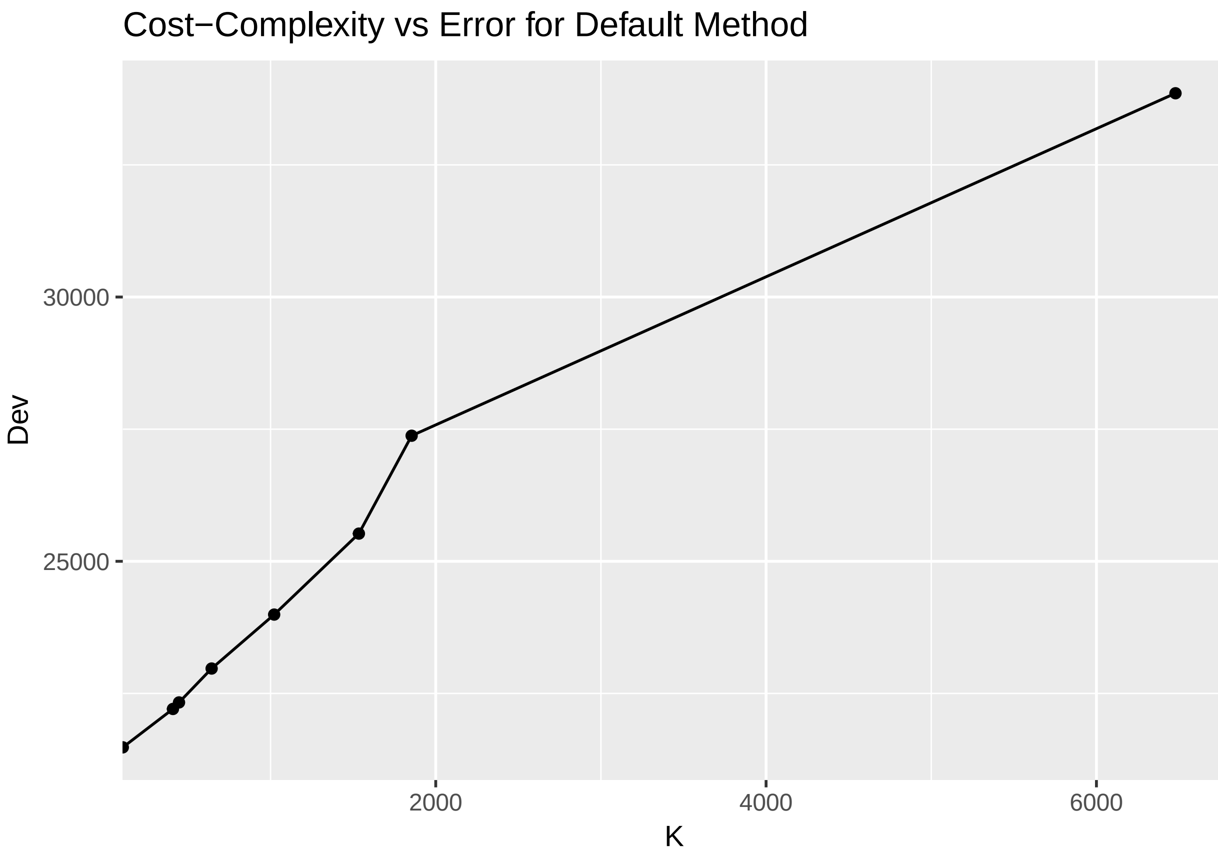
```r
default
```

```
##   Size         K      Dev
## 1    8      -Inf 21480.40
## 2    7  408.9358 22206.32
## 3    6  445.8046 22330.03
## 4    5  643.1663 22971.42
## 5    4 1021.8448 23992.21
## 6    3 1534.7488 25524.19
## 7    2 1854.1982 27376.34
## 8    1 6478.6784 33855.37
```

```r
ggplot(data = default, aes(x = Size, y = Dev)) + geom_point() + geom_line() + ggtitle("Size of Tree vs E
```




```r
ggplot(data = default, aes(x = K, y = Dev)) + geom_point() + geom_line() + ggtitle("Cost-Complexity vs E
```

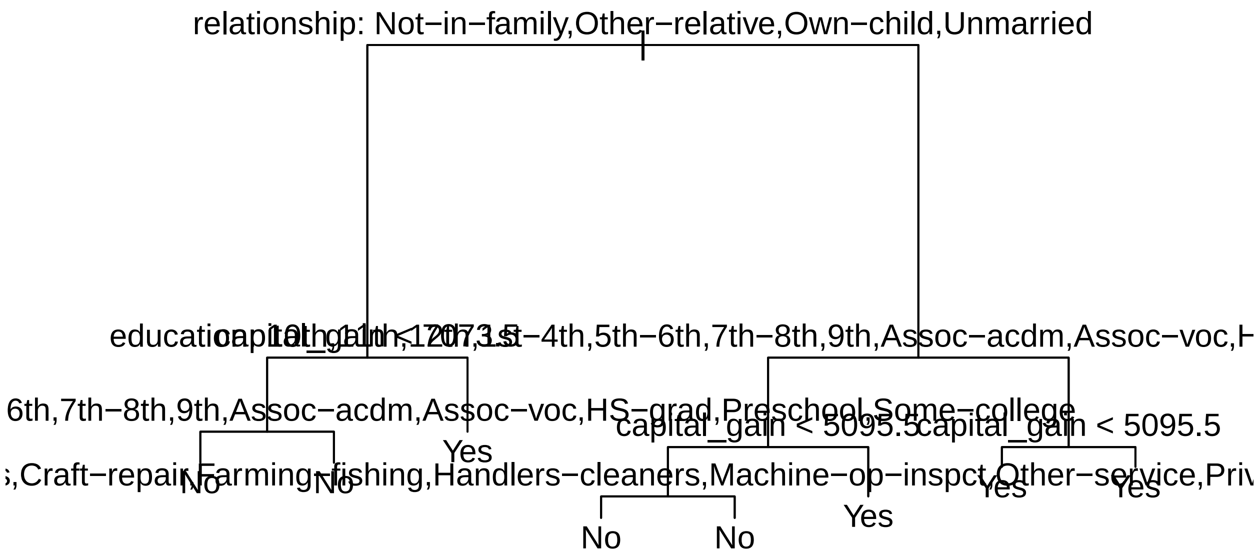


From this cross validation we see similar results in the cost complexity, but it seems using a size of 8 is ideal.

```
names <- classification_tree_cv_default$size
values <- classification_tree_cv_default$dev
names(values) <- names
size <- as.numeric(names(which.min(values)))
```

```
set.seed(4)
prune_classification_tree <- prune.misclass(classification_tree, best = size)
```

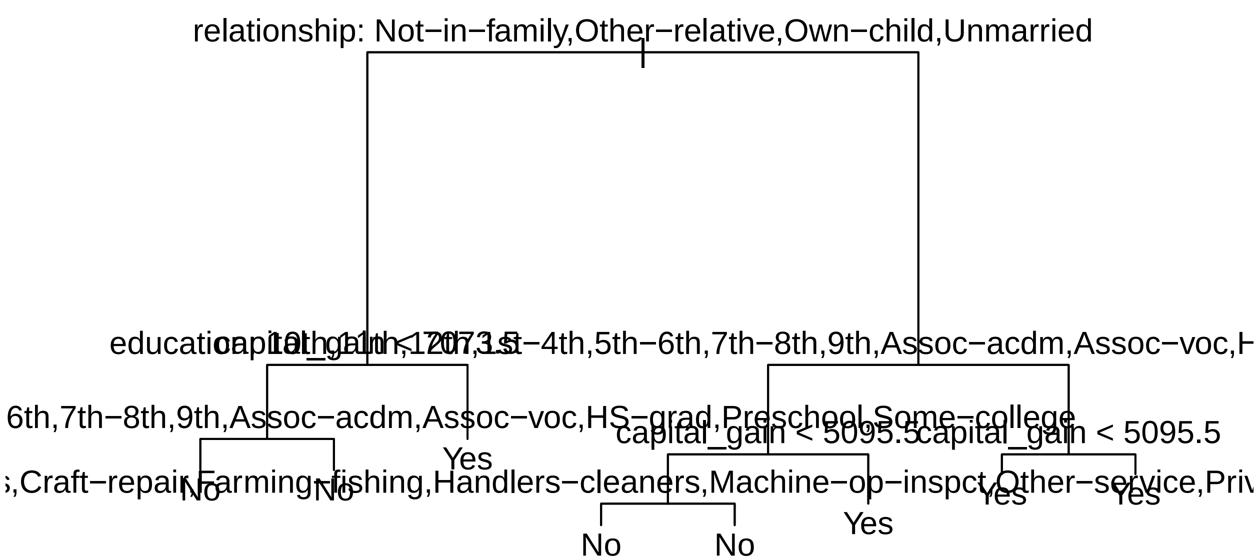
```
plot(prune_classification_tree)
text(prune_classification_tree, pretty = 0)
```




```r
summary(prune_classification_tree)
```

```
##
## Classification tree:
## tree(formula = Over50k ~ ., data = train_tree)
## Variables actually used in tree construction:
## [1] "relationship" "capital_gain" "education"    "occupation"
## Number of terminal nodes:  8
## Residual mean deviance:  0.7118 = 21460 / 30150
## Misclassification error rate: 0.1589 = 4794 / 30162
```

```r
plot(prune_classification_tree)
text(prune_classification_tree, pretty = 0)
```




```r
real_train <- train_tree$Over50k
train_preds <- predict(prune_classification_tree, train_tree, type = "class")
table(train_preds, real_train)
```

```
##            real_train
## train_preds    No   Yes
```

```
##          No  21536  3676
##          Yes  1118  3832
```

```r
err_rate <- mean(train_preds != real_train)
err_rate
```

```
## [1] 0.1589417
```

```r
library(ROCR)
```

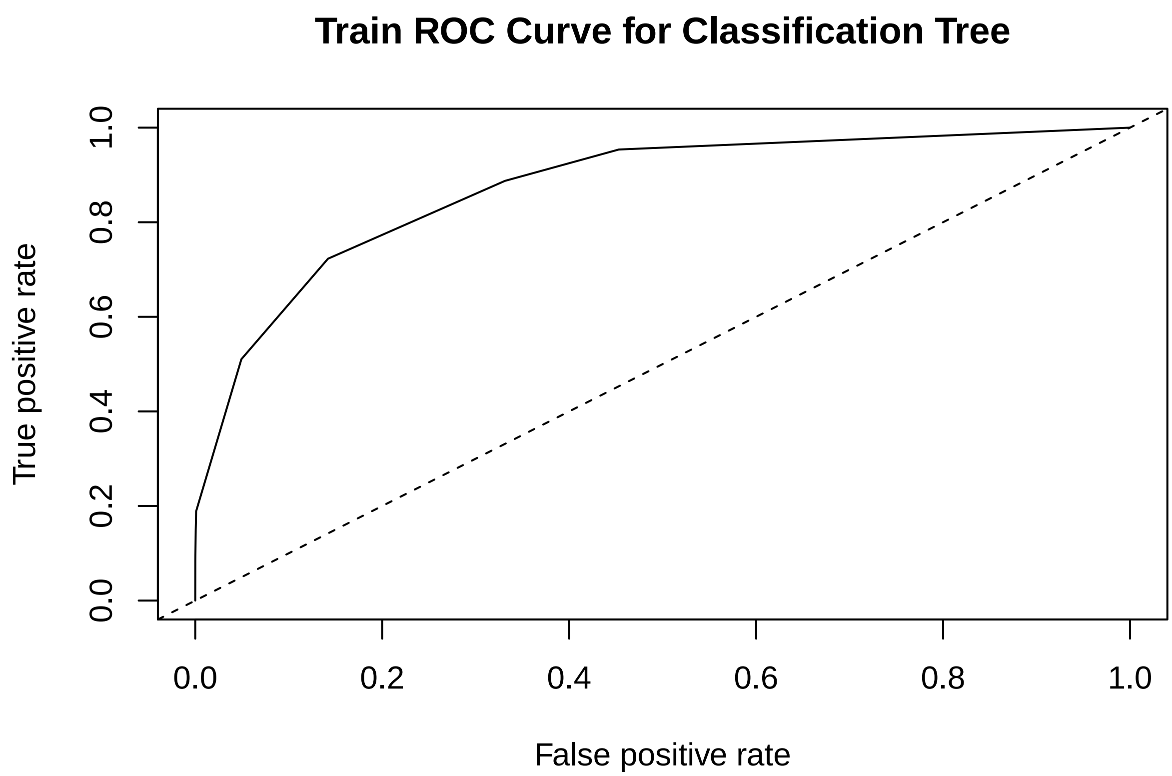
```
## Loading required package: gplots

##
## Attaching package: 'gplots'

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

```r
train_probs <- predict(prune_classification_tree, train_tree)
train_prediction <- prediction(train_probs[,2], real_train)
train_performance <- performance(train_prediction, measure = "tpr", x.measure = "fpr")
```

```r
plot(train_performance, main = "Train ROC Curve for Classification Tree")
abline(a=0, b=1, lty=2)
```




```r
performance(train_prediction, measure="auc")@y.values[[1]]
```

```
## [1] 0.8729808
```

```r
test_tree <- test[, -ncol(test)]
test_tree_preds <- test_tree[, -ncol(test_tree)]
real_test <- test_tree$Over50k
```

```r
ct_test_preds <- predict(prune_classification_tree, test_tree_preds, type = "class")
test_err_rate <- mean(ct_test_preds != real_test)
test_err_rate
```

```
## [1] 0.1610226
```

```r
confusionMatrix <- table(ct_test_preds, real_test)
confusionMatrix
```

```
##              real_test
## ct_test_preds    No   Yes
##           No  10772  1837
##           Yes   588  1863
```

```r
sensitivity <- confusionMatrix[2, 2]/(confusionMatrix[2, 2] + confusionMatrix[1, 2])
sensitivity
```

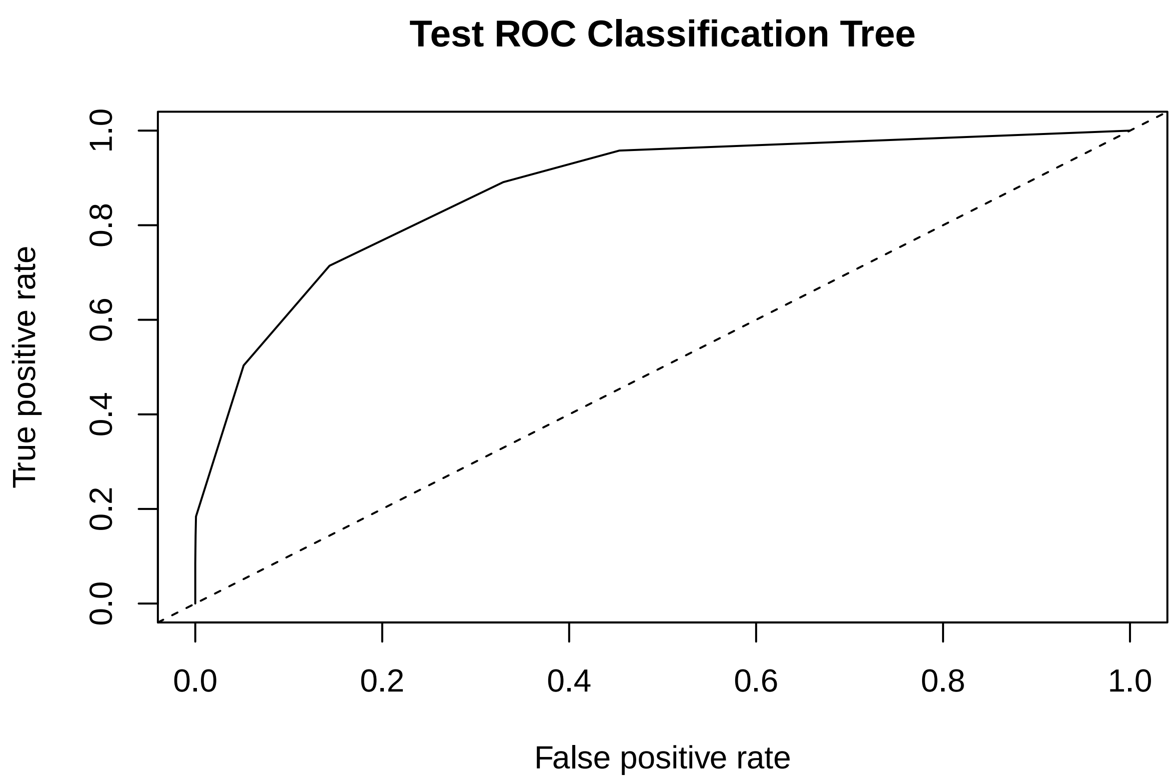
```
## [1] 0.5035135
```

```r
specificity <- confusionMatrix[1, 1]/(confusionMatrix[1, 1] + confusionMatrix[2, 1])
specificity
```

```
## [1] 0.9482394
```

```r
ct_test_probs <- predict(prune_classification_tree, test_tree_preds)
ct_test_prediction <- prediction(ct_test_probs[,2], real_test)
ct_test_performance <- performance(ct_test_prediction,  measure = "tpr", x.measure = "fpr")
```

```r
plot(ct_test_performance, main="Test ROC Classification Tree")
abline(a=0, b=1, lty=2)
```




```r
auc <- performance(ct_test_prediction, measure="auc")@y.values[[1]]
auc
```

```
## [1] 0.8723657
```