Decision Tree visualization

Siddharth.S.Chandran (18BCE1003)

26/03/2021

1. Load the dataset boston.csv

```
mydata= read.csv("C:/Users/Siddharth.S.Chandran/Downloads/Boston.csv")
names(mydata)
```

```
##  [1] "X"       "crim"    "zn"      "indus"   "chas"    "nox"     "rm"
##  [8] "age"     "dis"     "rad"     "tax"     "ptratio" "black"   "lstat"
## [15] "medv"
```

```
str(mydata)
```

```
## 'data.frame':    506 obs. of  15 variables:
##  $ X      : int  1 2 3 4 5 6 7 8 9 10 ...
##  $ crim   : num  0.00632 0.02731 0.02729 0.03237 0.06905 ...
##  $ zn     : num  18 0 0 0 0 0 12.5 12.5 12.5 12.5 ...
##  $ indus  : num  2.31 7.07 7.07 2.18 2.18 2.18 7.87 7.87 7.87 7.87 ...
##  $ chas   : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ nox    : num  0.538 0.469 0.469 0.458 0.458 0.458 0.524 0.524 0.524 0.524 ...
##  $ rm     : num  6.58 6.42 7.18 7 7.15 ...
##  $ age    : num  65.2 78.9 61.1 45.8 54.2 58.7 66.6 96.1 100 85.9 ...
##  $ dis    : num  4.09 4.97 4.97 6.06 6.06 ...
##  $ rad    : int  1 2 2 3 3 3 5 5 5 5 ...
##  $ tax    : int  296 242 242 222 222 222 311 311 311 311 ...
##  $ ptratio: num  15.3 17.8 17.8 18.7 18.7 18.7 15.2 15.2 15.2 15.2 ...
##  $ black  : num  397 397 393 395 397 ...
##  $ lstat  : num  4.98 9.14 4.03 2.94 5.33 ...
##  $ medv   : num  24 21.6 34.7 33.4 36.2 28.7 22.9 27.1 16.5 18.9 ...
```

Splitting into training and testing dataset

```
dt = sort(sample(nrow(mydata), nrow(mydata)*.7))
train<-mydata[dt,]
val<-mydata[-dt,]
nrow(train)
```

```
## [1] 354
```

354 records used for training

Loading the libraries for decision tree visualization

```
library(rpart)
library(rpart.plot)
library(RColorBrewer)
library(rattle)
```

```
## Loading required package: tibble
```

```
## Loading required package: bitops
```

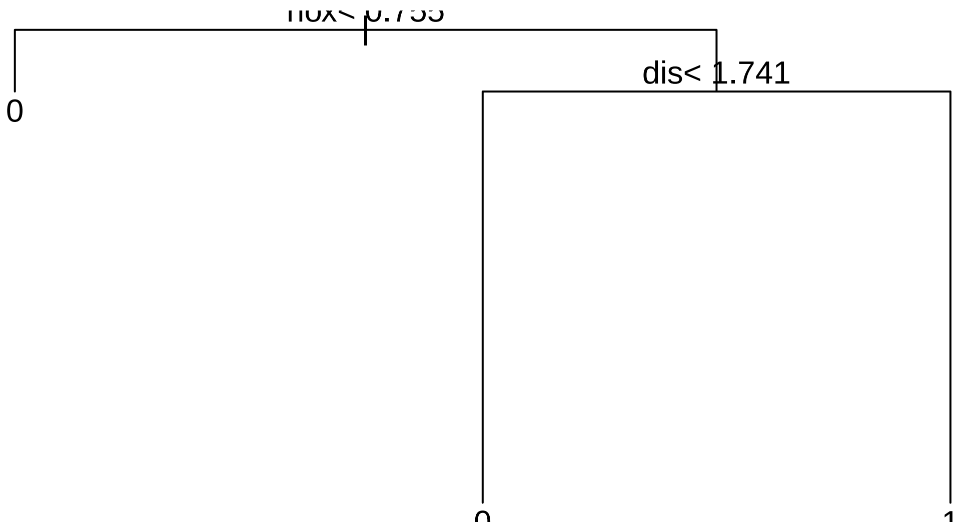
```
## Rattle: A free graphical interface for data science with R.
## Version 5.4.0 Copyright (c) 2006-2020 Togaware Pty Ltd.
## Type 'rattle()' to shake, rattle, and roll your data.
```

```
mytree <- rpart(chas~., data = train, method="class", control = rpart.control(minsplit = 20, minbucket
mytree
```

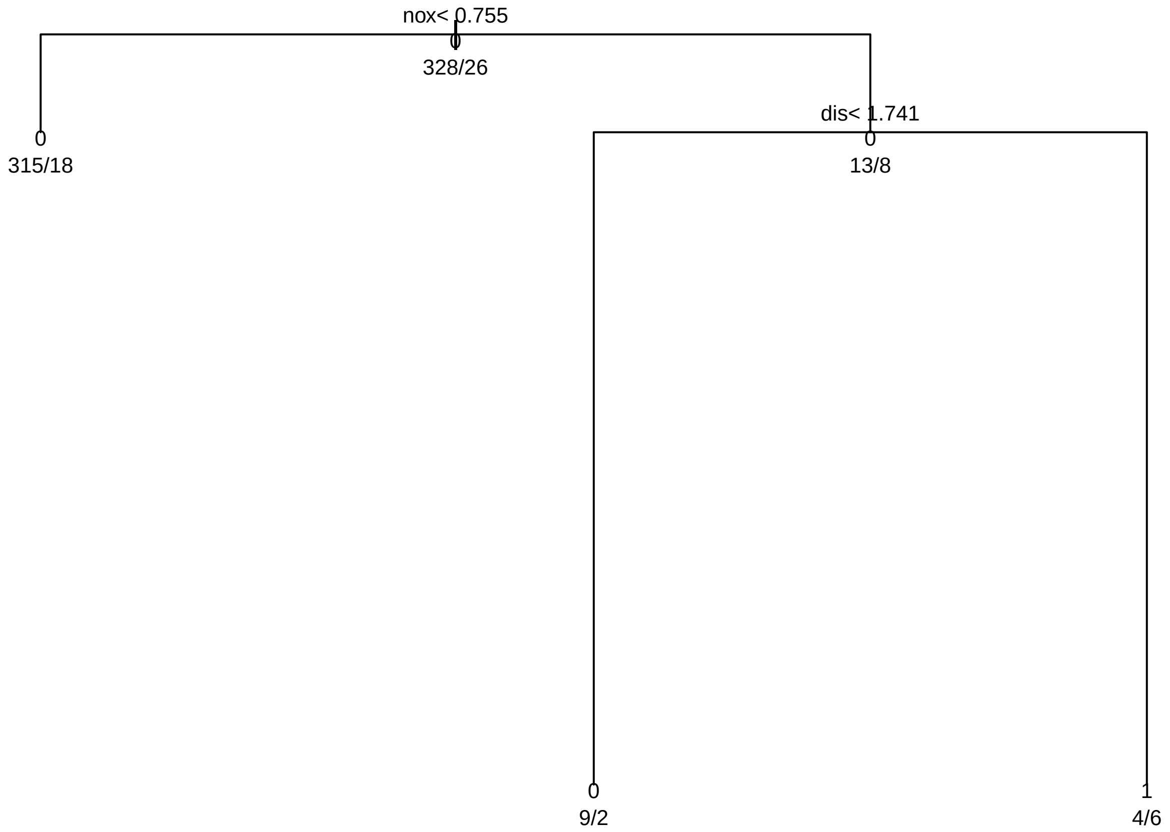
```
## n= 354
##
## node), split, n, loss, yval, (yprob)
##       * denotes terminal node
##
## 1) root 354 26 0 (0.92655367 0.07344633)
##   2) nox< 0.755 333 18 0 (0.94594595 0.05405405) *
##   3) nox>=0.755 21  8 0 (0.61904762 0.38095238)
##     6) dis< 1.74095 11  2 0 (0.81818182 0.18181818) *
##     7) dis>=1.74095 10  4 1 (0.40000000 0.60000000) *
```

Plot the trees

```
plot(mytree)
text(mytree)
```



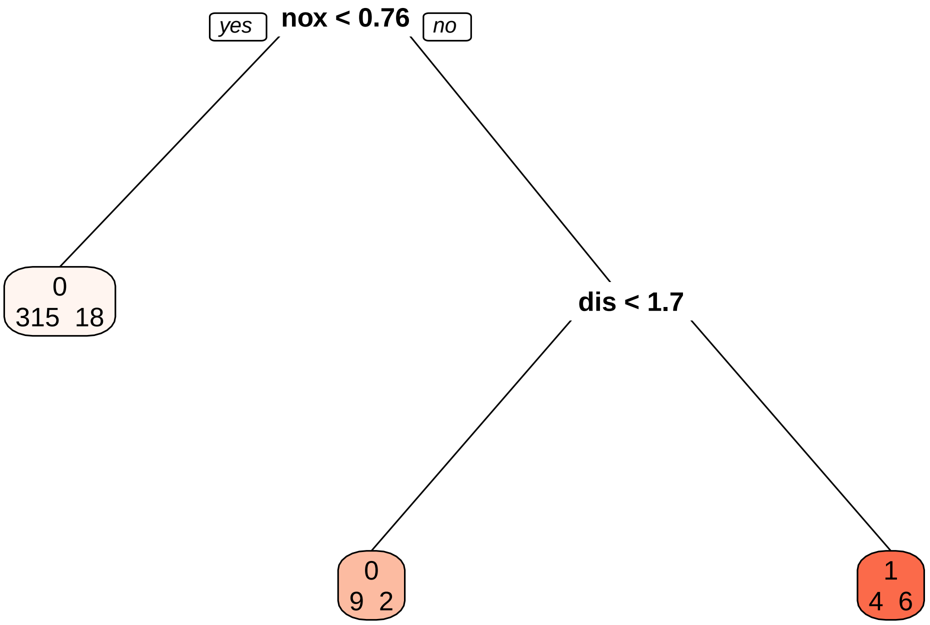
```
par(xpd = NA, mar = rep(0.7, 4))
plot(mytree, compress = TRUE)
text(mytree, cex = 0.7, use.n = TRUE, fancy = FALSE, all = TRUE)
```



Here the decision tree is constructed wrt tax attribute

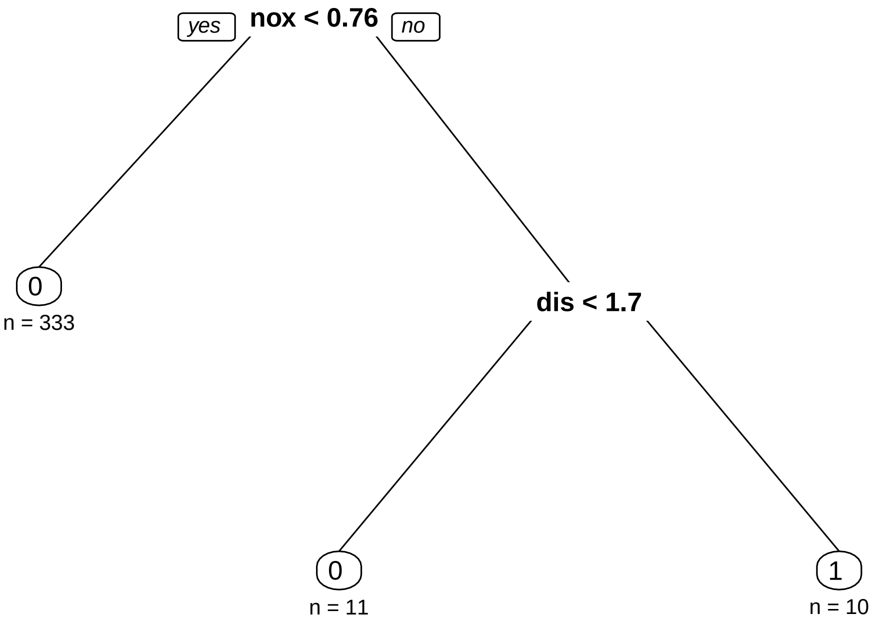
Visualizing the tree using prp function

```
library(rpart.plot)#view1
prp(mytree, faclen = 0,box.palette = "Reds", cex = 0.8, extra = 1)
```



Total count at each node

```
total_count <- function(x, labs, digits, varlen){paste(labs, "\n\nn =", x$frame$n)}
prp(mytree, faclen = 0, cex = 0.8, node.fun=total_count)
```

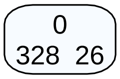


```
printcp(mytree)
```

```
##
## Classification tree:
## rpart(formula = chas ~ ., data = train, method = "class", control = rpart.control(minsplit = 20,
##     minbucket = 10, maxdepth = 10, usesurrogate = 2, xval = 10))
##
## Variables actually used in tree construction:
## [1] dis nox
##
## Root node error: 26/354 = 0.073446
##
## n= 354
##
##         CP nsplit rel error xerror    xstd
## 1 0.038462      0   1.00000 1.0000 0.18878
## 2 0.010000      2   0.92308 1.1538 0.20154
```

Pruning the decision tree and visualizing it

```
best_cp <- mytree$cptable[which.min(mytree$cptable[,"xerror"]),"CP"]
prunedTree <- prune(mytree, cp = best_cp)
prp(prunedTree, box.palette = "Blues",faclen = 0, cex = 0.8, extra = 1)
```



Printing the confusion matrix

```
confusionMatrix <- table(train$chas, predict(prunedTree,type="class"))
rownames(confusionMatrix) <- paste("Actual", rownames(confusionMatrix), sep = ":")
colnames(confusionMatrix) <- paste("Pred", colnames(confusionMatrix), sep = ":")
print(confusionMatrix)
```

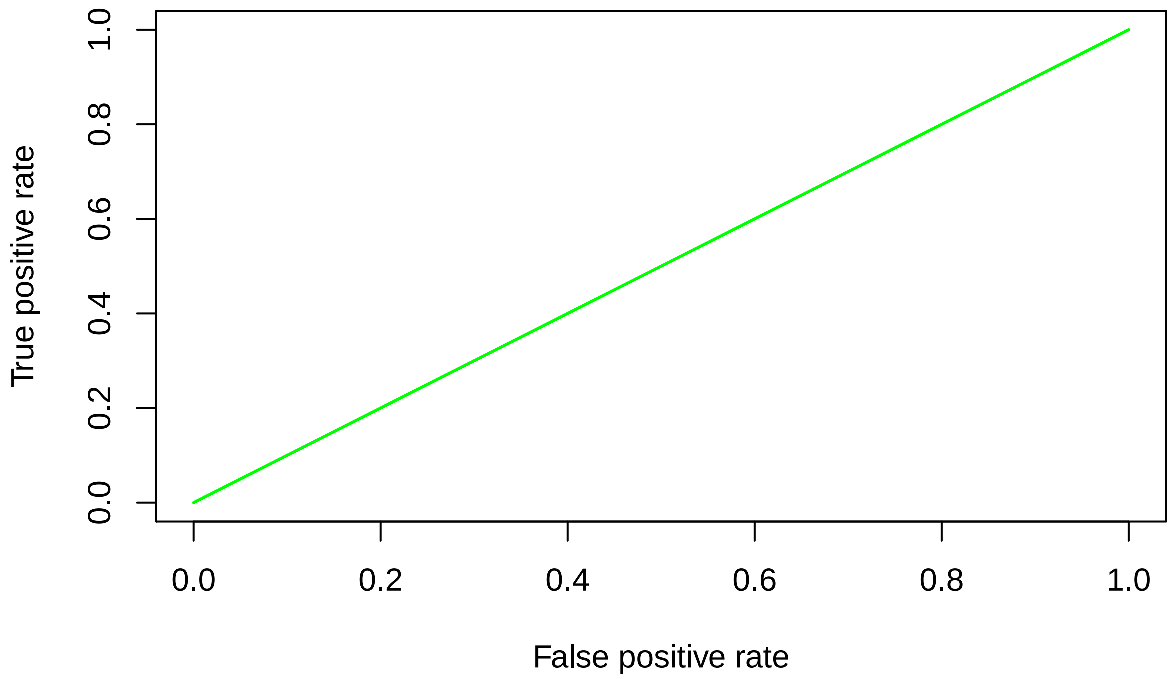
```
##
##           Pred:0 Pred:1
##   Actual:0    328      0
##   Actual:1     26      0
```

Plotting ROC curve

```
library(ROCR)
val1 = predict(prunedTree, val, type = "prob")
predictedValue <-prediction(val1[,2],val$chas)
perfVal <- performance(predictedValue,"auc")
perfVal
```

```
## A performance instance
##   'Area under the ROC curve'
```

```
perfVal <- performance(predictedValue, "tpr", "fpr")#Plot the ROC
plot(perfVal, col = "green", lwd = 1.5)
```



Calculating the KS statisitics

```
ks1Tree <- max(attr(perfVal, "y.values")[[1]] - (attr(perfVal, "x.values")[[1]]))
ks1Tree
```

```
## [1] 0
```

Visualizing the tree using random forest algorithm

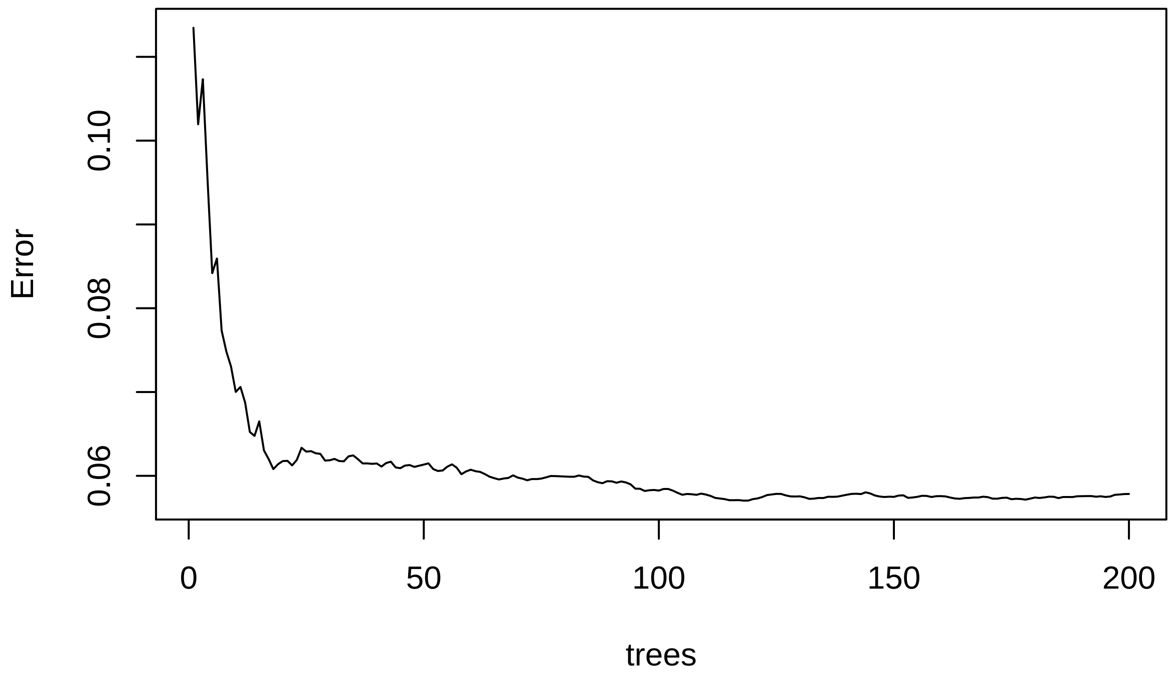
```
library(randomForest)
```

```
## randomForest 4.6-14

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

##
## Attaching package: 'randomForest'

## The following object is masked from 'package:rattle':
##
##     importance
```

```
rf50 <- randomForest(chas ~., data = train, ntree=200, importance=T, proximity=T)
```

```
## Warning in randomForest.default(m, y, ...): The response has five or fewer
## unique values. Are you sure you want to do regression?
```

```
plot(rf50, main="")
```



```
rf50
```

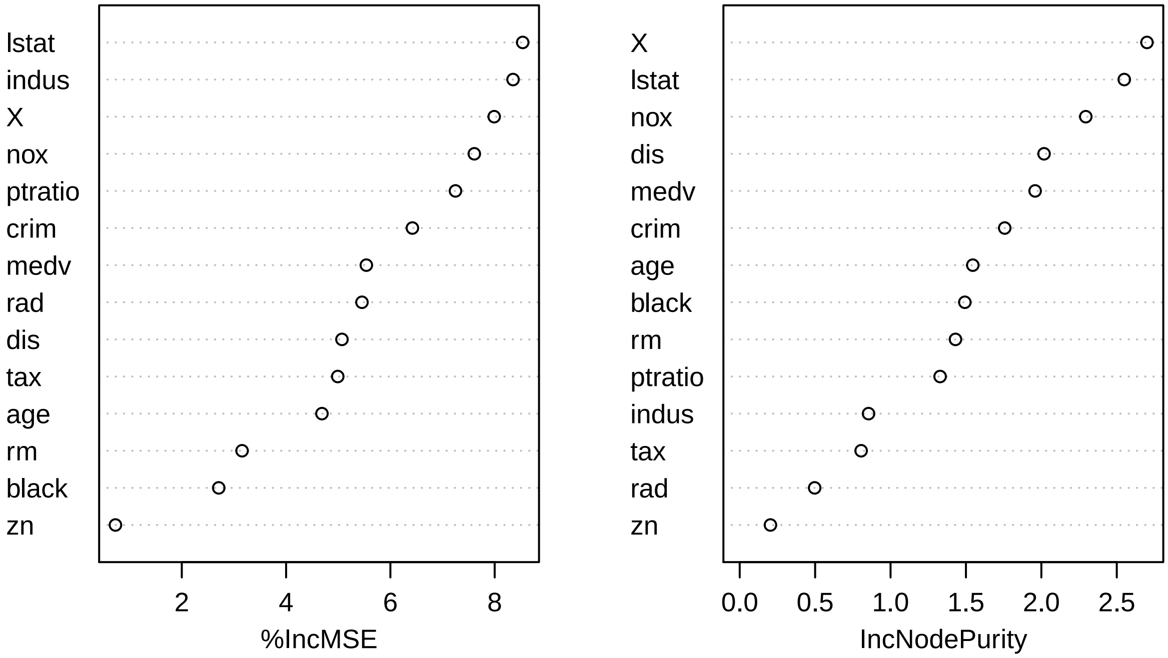
```
##
## Call:
##  randomForest(formula = chas ~ ., data = train, ntree = 200, importance = T,      proximity = T)
##                Type of random forest: regression
##                      Number of trees: 200
## No. of variables tried at each split: 4
##
##           Mean of squared residuals: 0.05783555
##                     % Var explained: 15.01
```

```
Test50_rf_pred <- predict(rf50, val, type="class")
table(Test50_rf_pred, val$chas)
```

```
##
```

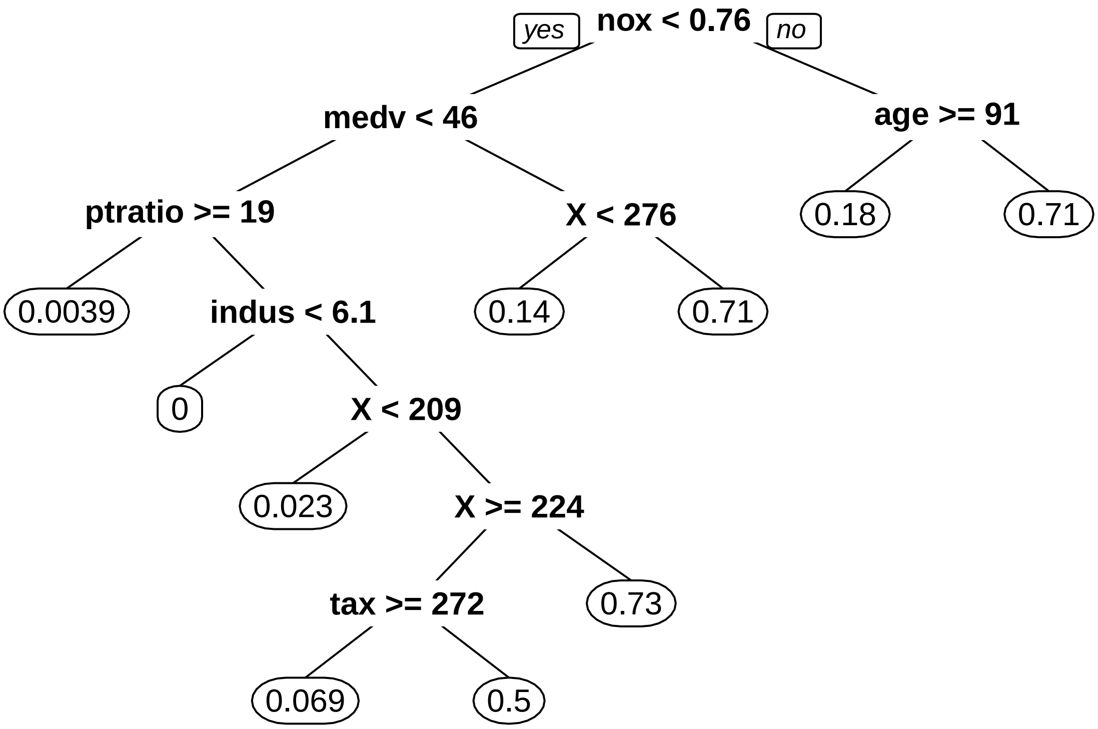
```
## Test50_rf_pred          0 1
##   -5.29437604868122e-17 2 0
##   -5.25968157916168e-17 1 0
##   -5.20417042793042e-17 1 0
##   -5.19029264012261e-17 1 0
##   -5.05151476204446e-17 1 0
##   -4.91967577787022e-17 2 0
##   -4.91273688396632e-17 1 0
##   -4.8988590961585e-17  1 0
##   -4.85028683883115e-17 1 0
##   -4.78089789979208e-17 1 0
##   -4.76702011198427e-17 1 0
##   -4.71844785465692e-17 1 0
##   -4.59354776438659e-17 1 0
##   -4.58660887048268e-17 1 0
##   -4.46170878021235e-17 1 0
##   -4.44089209850063e-17 1 0
##   -4.39925873507718e-17 1 0
##   -4.33680868994202e-17 1 0
##   -4.30905311432639e-17 1 0
##   -4.21190859967169e-17 1 0
##   -4.17721413015215e-17 1 0
##   -4.14251966063262e-17 1 0
##   -3.76088049591772e-17 1 0
##   0.00199999999999995   2 0
##   0.00299999999999996   1 0
##   0.0033333333333333    1 0
##   0.00349999999999996   1 0
##   0.00374999999999996   1 0
##   0.00499999999999997   3 0
##   0.00499999999999998   1 0
##   0.00499999999999999   5 0
##   0.005                 2 0
##   0.00533333333333329   1 0
##   0.00541666666666664   1 0
##   0.00566666666666663   1 0
##   0.00599999999999996   1 0
##   0.00599999999999997   1 0
##   0.006                 1 0
##   0.00624999999999996   1 0
##   0.00624999999999998   1 0
##   0.00699999999999997   1 0
##   0.00725               1 0
##   0.00841666666666663   1 0
##   0.00916666666666666   1 0
##   0.00999999999999998   1 0
##   0.00999999999999999   1 0
##   0.01                  1 0
##   0.01125               3 0
##   0.01175               1 0
##   0.0125                1 0
##   0.01275               1 0
##   0.0133333333333333    1 0
##   0.0138333333333333    1 0
##   0.014                 2 0
##   0.0141666666666667    1 0
##   0.015                 2 0
##   0.0154166666666667    1 0
##   0.016                 1 0
##   0.0166666666666666    1 0
##   0.0185                1 0
##   0.01875               1 0
##   0.0193333333333333    1 0
##   0.02                  1 0
##   0.0206666666666667    1 0
##   0.021                 1 0
##   0.0215                1 0
##   0.0225                1 0
##   0.0243333333333333    1 0
##   0.02625               1 0
##   0.027                 1 0
##   0.0284166666666667    1 0
##   0.03                  1 0
##   0.03025               1 0
##   0.0306785714285714    1 0
##   0.03275               1 0
##   0.0331666666666667    1 0
##   0.0333333333333333    1 0
##   0.034                 1 0
##   0.0364166666666667    1 0
##   0.0381666666666667    1 0
##   0.0383333333333333    1 0
##   0.0390833333333333    1 0
##   0.042                 2 0
##   0.0435                1 0
##   0.04425               1 0
##   0.0491666666666667    1 0
##   0.0505                1 0
##   0.051                 1 0
##   0.0510833333333333    1 0
##   0.0516666666666667    2 0
##   0.0519166666666667    1 0
##   0.0596666666666667    1 0
##   0.0601666666666667    1 0
##   0.064                 1 0
##   0.0646666666666667    1 0
##   0.0668333333333333    1 0
##   0.0699166666666667    1 0
##   0.07225               1 0
##   0.0729166666666667    1 0
##   0.0731666666666667    1 0
##   0.0773333333333333    1 0
##   0.0795                1 0
##   0.0810833333333333    1 0
##   0.0844166666666667    1 0
##   0.0891666666666667    1 0
##   0.0925833333333333    1 0
##   0.09425               1 0
##   0.102833333333333     1 0
##   0.105166666666667     1 0
##   0.109666666666667     1 0
##   0.112833333333333     0 1
##   0.115333333333333     1 0
##   0.13375               1 0
##   0.14075               1 0
##   0.144416666666667     0 1
##   0.165095238095238     0 1
##   0.167083333333333     0 1
##   0.174666666666667     1 0
##   0.195345238095238     0 1
##   0.202666666666667     0 1
##   0.208583333333333     1 0
##   0.21675               0 1
##   0.22525               1 0
##   0.2295                1 0
##   0.240833333333333     1 0
##   0.251416666666667     1 0
##   0.3025                1 0
##   0.33125               1 0
##   0.343166666666667     1 0
##   0.364166666666667     1 0
##   0.3995                1 0
##   0.401583333333333     1 0
##   0.515416666666667     0 1
##   0.557166666666667     1 0
##   0.559333333333333     1 0
##   0.58025               0 1
```

```r
varImpPlot(rf50,  main="", cex=0.8)
```

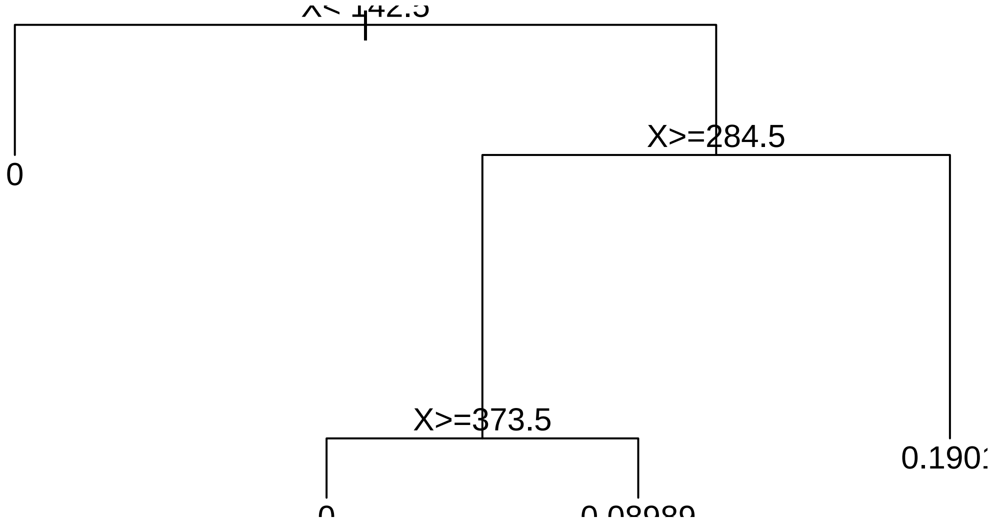


Visualizing using CART model

```
latlontree = rpart(mydata$chas~., data= mydata)# Plot the tree using prp command defined in rpart.plot
prp(latlontree)
```



```
latlontree = rpart(mydata$chas~., data= mydata,minbucket=50)
plot(latlontree)
text(latlontree)
```


X>=142.5
0
X>=284.5
X>=373.5
0.1901
0
0.08989