Tree Based Method

Semanti Ghosh

2025-03-15

Tree Based Methods

1.Importing the designated file

```
#setwd("C:\\Study\\Semester_6\\Statistical_Learning_Lab\\assignment_7")

drug <- read.csv("drug200.csv", header=TRUE)
head(drug)
```

```
##   Age Sex     BP Cholesterol Na_to_K  Drug
## 1  23   F   HIGH        HIGH  25.355 drugY
## 2  47   M    LOW        HIGH  13.093 drugC
## 3  47   M    LOW        HIGH  10.114 drugC
## 4  28   F NORMAL        HIGH   7.798 drugX
## 5  61   F    LOW        HIGH  18.043 drugY
## 6  22   F NORMAL        HIGH   8.607 drugX
```

2.Data Cleaning and Preprocessing

First, the structure of the dataset is viewed (mostly to check for the categorical columns). Then, it is checked for missing values and unique values in the categorical columns are viewed. Later on, the categorical values are then converted to factors (so that things can be seen numerically). They are just viewed for now.

```
# Viewing the structure of the dataset
str(drug)
```

```
## 'data.frame':    200 obs. of  6 variables:
##  $ Age        : int  23 47 47 28 61 22 49 41 60 43 ...
##  $ Sex        : chr  "F" "M" "M" "F" ...
##  $ BP         : chr  "HIGH" "LOW" "LOW" "NORMAL" ...
##  $ Cholesterol: chr  "HIGH" "HIGH" "HIGH" "HIGH" ...
##  $ Na_to_K    : num  25.4 13.1 10.1 7.8 18 ...
##  $ Drug       : chr  "drugY" "drugC" "drugC" "drugX" ...
```

```
# Checking for missing values
colSums(is.na(drug))
```

```
##         Age         Sex          BP Cholesterol     Na_to_K        Drug
##           0           0           0           0           0           0
```

```r
# Viewing unique values in categorical columns
unique(drug$Sex)
```

```
## [1] "F" "M"
```

```r
unique(drug$BP)
```

```
## [1] "HIGH"   "LOW"    "NORMAL"
```

```r
unique(drug$Cholesterol)
```

```
## [1] "HIGH"   "NORMAL"
```

```r
unique(drug$Drug)
```

```
## [1] "drugY" "drugC" "drugX" "drugA" "drugB"
```

3.Identifying the Response Variable

The response variable in this case is **Drug**. It is a categorical variable with classes drugA, drugB, drugC, drugX and drugY.

Converting categorical inputs to consider while fitting the data

```r
# Converting categorical variables to factors
drug$Sex <- as.factor(drug$Sex)
drug$BP <- as.factor(drug$BP)
drug$Cholesterol <- as.factor(drug$Cholesterol)
drug$Drug <- as.factor(drug$Drug)
```

Fitting a Classification and Regression Tree Model

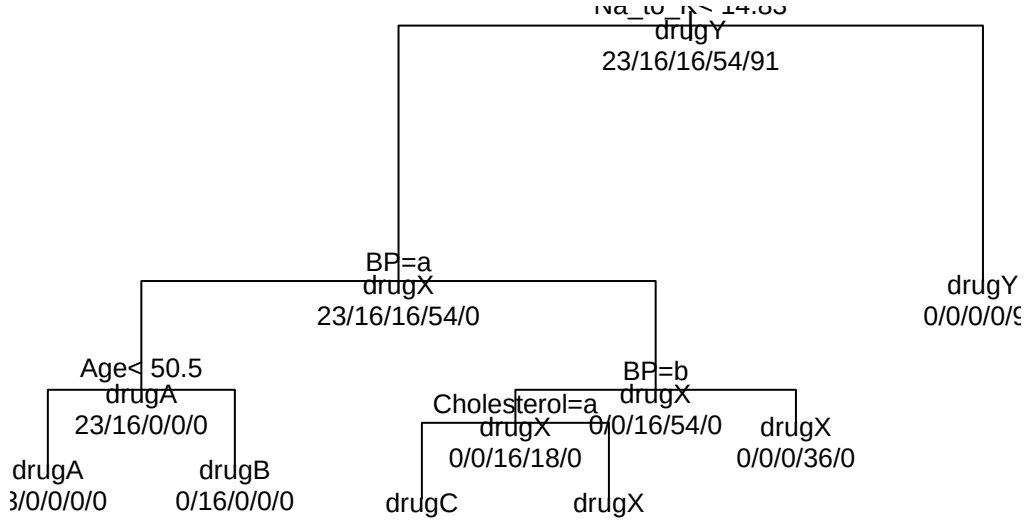
```r
# Loading the necessary library
library(rpart)

# Fitting the decision tree model
tree_model <- rpart(Drug ~ Age + Sex + BP + Cholesterol + Na_to_K, data = drug, method = "class")

# Displaying the model summary
summary(tree_model)
```

```
## Call:
## rpart(formula = Drug ~ Age + Sex + BP + Cholesterol + Na_to_K,
##     data = drug, method = "class")
##   n= 200
##
##          CP nsplit rel error     xerror       xstd
## 1 0.4954128      0 1.0000000 1.00000000 0.06460892
## 2 0.2110092      1 0.5045872 0.51376147 0.05825507
## 3 0.1467890      2 0.2935780 0.30275229 0.04815858
## 4 0.0733945      3 0.1467890 0.17431193 0.03804299
## 5 0.0100000      5 0.0000000 0.02752294 0.01577075
##
## Variable importance
##     Na_to_K          BP         Age Cholesterol
##          48          23          17          11
##
## Node number 1: 200 observations,    complexity param=0.4954128
##   predicted class=drugY  expected loss=0.545  P(node) =1
##     class counts:    23    16    16    54    91
##    probabilities: 0.115 0.080 0.080 0.270 0.455
##   left son=2 (109 obs) right son=3 (91 obs)
##   Primary splits:
##       Na_to_K     < 14.8285 to the left,  improve=66.1127500, (0 missing)
##       BP          splits as  LRR,         improve=16.3840700, (0 missing)
##       Age         < 50.5    to the left,  improve= 3.8444080, (0 missing)
##       Cholesterol splits as  LR,          improve= 2.4284570, (0 missing)
##       Sex         splits as  RL,          improve= 0.3933333, (0 missing)
##   Surrogate splits:
##       Age < 16.5    to the right, agree=0.555, adj=0.022, (0 split)
##
## Node number 2: 109 observations,    complexity param=0.2110092
##   predicted class=drugX  expected loss=0.5045872  P(node) =0.545
##     class counts:    23    16    16    54     0
##    probabilities: 0.211 0.147 0.147 0.495 0.000
##   left son=4 (39 obs) right son=5 (70 obs)
##   Primary splits:
##       BP          splits as  LRR,         improve=29.1397400, (0 missing)
##       Age         < 50.5    to the left,  improve= 7.0423030, (0 missing)
##       Cholesterol splits as  LR,          improve= 4.4277060, (0 missing)
##       Na_to_K     < 9.444   to the right, improve= 2.1769230, (0 missing)
##       Sex         splits as  RL,          improve= 0.3958872, (0 missing)
##   Surrogate splits:
##       Age < 69.5    to the right, agree=0.651, adj=0.026, (0 split)
##
## Node number 3: 91 observations
##   predicted class=drugY  expected loss=0  P(node) =0.455
##     class counts:     0     0     0     0    91
##    probabilities: 0.000 0.000 0.000 0.000 1.000
##
## Node number 4: 39 observations,    complexity param=0.146789
##   predicted class=drugA  expected loss=0.4102564  P(node) =0.195
##     class counts:    23    16     0     0     0
##    probabilities: 0.590 0.410 0.000 0.000 0.000
##   left son=8 (23 obs) right son=9 (16 obs)
```

```
##   Primary splits:
##       Age         < 50.5    to the left,  improve=18.871790000, (0 missing)
##       Na_to_K     < 13.197  to the left,  improve= 1.577152000, (0 missing)
##       Cholesterol splits as  LR,          improve= 0.008636977, (0 missing)
##       Sex         splits as  LR,          improve= 0.005128205, (0 missing)
##   Surrogate splits:
##       Na_to_K < 13.197  to the left,  agree=0.667, adj=0.187, (0 split)
##
## Node number 5: 70 observations,    complexity param=0.0733945
##   predicted class=drugX  expected loss=0.2285714  P(node) =0.35
##     class counts:     0     0    16    54     0
##    probabilities: 0.000 0.000 0.229 0.771 0.000
##   left son=10 (34 obs) right son=11 (36 obs)
##   Primary splits:
##       BP          splits as  -LR,         improve=7.74453800, (0 missing)
##       Cholesterol splits as  LR,          improve=6.90793700, (0 missing)
##       Na_to_K     < 9.7105  to the right, improve=0.78354040, (0 missing)
##       Age         < 49.5    to the left,  improve=0.36571430, (0 missing)
##       Sex         splits as  RL,          improve=0.06806723, (0 missing)
##   Surrogate splits:
##       Na_to_K     < 10.1085 to the right, agree=0.643, adj=0.265, (0 split)
##       Age         < 49.5    to the left,  agree=0.586, adj=0.147, (0 split)
##       Sex         splits as  RL,          agree=0.543, adj=0.059, (0 split)
##       Cholesterol splits as  RL,          agree=0.543, adj=0.059, (0 split)
##
## Node number 8: 23 observations
##   predicted class=drugA  expected loss=0  P(node) =0.115
##     class counts:    23     0     0     0     0
##    probabilities: 1.000 0.000 0.000 0.000 0.000
##
## Node number 9: 16 observations
##   predicted class=drugB  expected loss=0  P(node) =0.08
##     class counts:     0    16     0     0     0
##    probabilities: 0.000 1.000 0.000 0.000 0.000
##
## Node number 10: 34 observations,    complexity param=0.0733945
##   predicted class=drugX  expected loss=0.4705882  P(node) =0.17
##     class counts:     0     0    16    18     0
##    probabilities: 0.000 0.000 0.471 0.529 0.000
##   left son=20 (16 obs) right son=21 (18 obs)
##   Primary splits:
##       Cholesterol splits as  LR,          improve=1.694118e+01, (0 missing)
##       Age         < 33      to the left,  improve=1.633484e+00, (0 missing)
##       Na_to_K     < 10.6885 to the left,  improve=1.412605e+00, (0 missing)
##       Sex         splits as  RL,          improve=8.255934e-04, (0 missing)
##   Surrogate splits:
##       Age     < 30      to the left,  agree=0.647, adj=0.25, (0 split)
##       Na_to_K < 10.6885 to the left,  agree=0.647, adj=0.25, (0 split)
##
## Node number 11: 36 observations
##   predicted class=drugX  expected loss=0  P(node) =0.18
##     class counts:     0     0     0    36     0
##    probabilities: 0.000 0.000 0.000 1.000 0.000
##
```

```
## Node number 20: 16 observations
##   predicted class=drugC  expected loss=0  P(node) =0.08
##     class counts:     0     0    16     0     0
##    probabilities: 0.000 0.000 1.000 0.000 0.000
##
## Node number 21: 18 observations
##   predicted class=drugX  expected loss=0  P(node) =0.09
##     class counts:     0     0     0    18     0
##    probabilities: 0.000 0.000 0.000 1.000 0.000
```

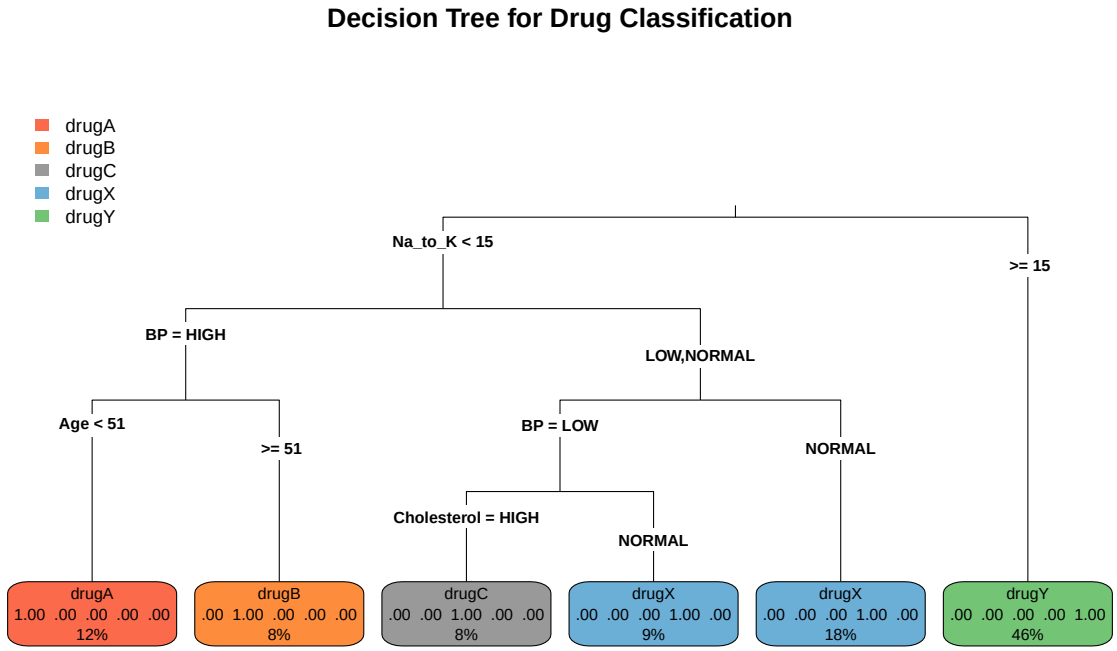
```
# Plotting the tree
plot(tree_model)
text(tree_model, use.n = TRUE, all = TRUE, cex = 0.8)
```



Plotting a Decision Tree for the fitted model

The model fitted has already been already been plotted once immediately after fitting. It'll be plotted again, this time with some refinements.

```
# loading the required library
library(rpart.plot)

# Plotting using rpart.plot
rpart.plot(tree_model, type = 3, extra = 104, fallen.leaves = TRUE,
           main = "Decision Tree for Drug Classification")
```

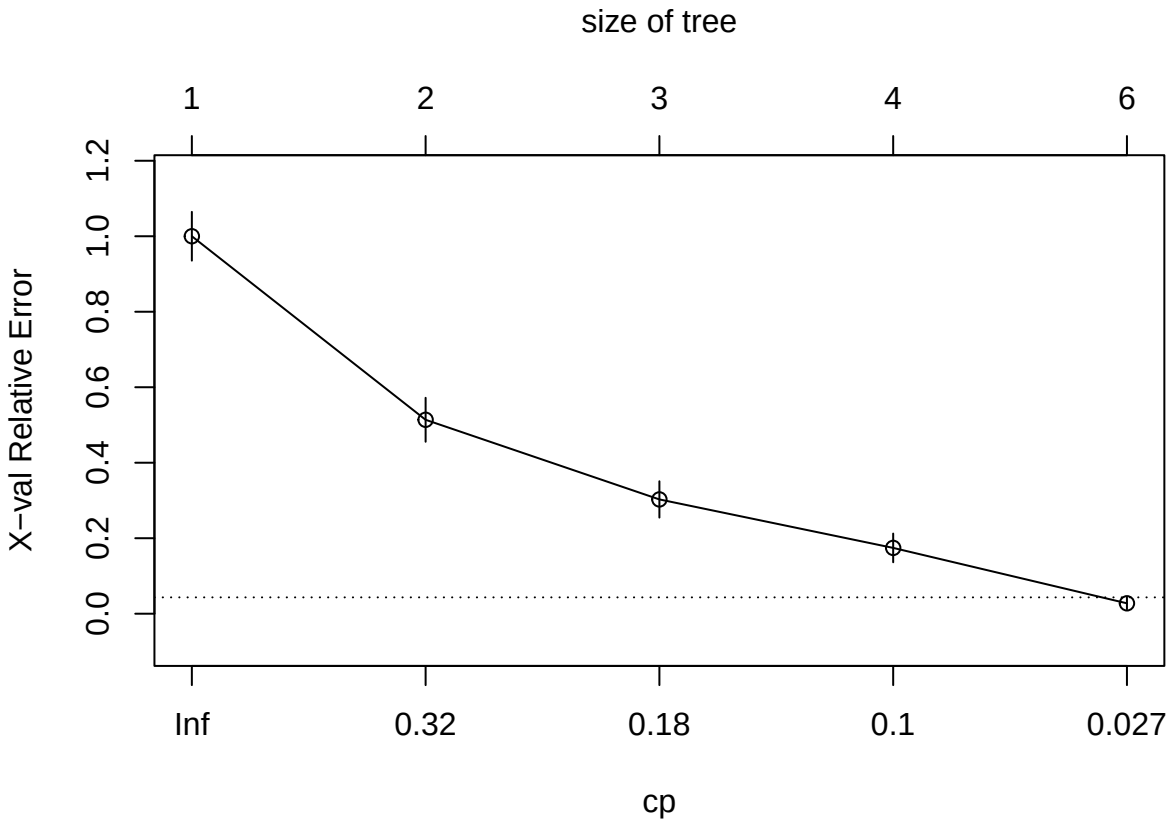


7. Pruning the Tree by changing the Cp value

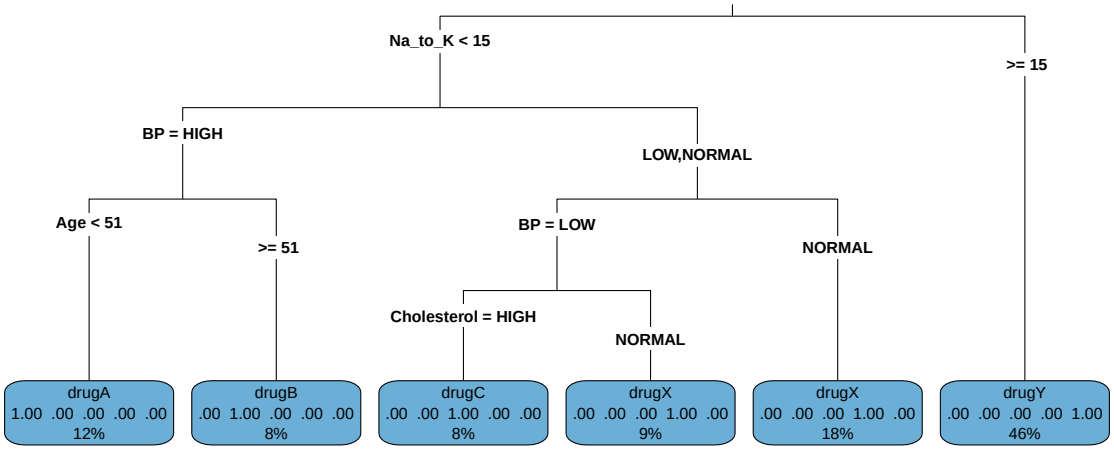
```
printcp(tree_model) # checking the Cp table
```

```
##
## Classification tree:
## rpart(formula = Drug ~ Age + Sex + BP + Cholesterol + Na_to_K,
##     data = drug, method = "class")
##
## Variables actually used in tree construction:
## [1] Age         BP          Cholesterol Na_to_K
##
## Root node error: 109/200 = 0.545
##
## n= 200
##
##         CP nsplit rel error   xerror     xstd
## 1 0.495413      0   1.00000 1.000000 0.064609
## 2 0.211009      1   0.50459 0.513761 0.058255
## 3 0.146789      2   0.29358 0.302752 0.048159
## 4 0.073394      3   0.14679 0.174312 0.038043
## 5 0.010000      5   0.00000 0.027523 0.015771
```

```r
plotcp(tree_model) # plotting the Cp vs error graph
```




```r
# Pruning the tree
pruned_tree <- prune(tree_model, cp = 0.012)

# Plotting the pruned tree
rpart.plot(pruned_tree, type = 3, extra = 104, fallen.leaves = TRUE, box.palette = "Blues", main = "Pru
```



Over here, we tried to use the lowest cp to prune the tree. However, the tree wasn't pruned, in spite of the threshold being greater than the minimum threshold in the table (obtained above). This probably happened because the concerned node had children (split nodes) with cp values greater than that node.

8. Calculating the Misclassification Rate or Accuracy

First, the misclassification rate and accuracy are calculated for the original tree (the one without pruning)

```
# Predicting using original tree
pred <- predict(tree_model, type = "class")

# Printing the confusion matrix
conf_mat <- table(Predicted = pred, Actual = drug$Drug)
conf_mat
```

```
##          Actual
## Predicted drugA drugB drugC drugX drugY
##     drugA    23     0     0     0     0
##     drugB     0    16     0     0     0
##     drugC     0     0    16     0     0
##     drugX     0     0     0    54     0
##     drugY     0     0     0     0    91
```

```r
# Accuracy and misclassification rate
accuracy <- sum(diag(conf_mat)) / sum(conf_mat)
misclass <- 1 - accuracy

cat("Original Tree - Accuracy:", accuracy, "\n")
```

```
## Original Tree - Accuracy: 1
```

```r
cat("Original Tree - Misclassification Rate:", misclass, "\n")
```

```
## Original Tree - Misclassification Rate: 0
```

Then, the misclassification rate and accuracy are calculated for the pruned tree

```r
# Predicting using pruned tree
prune_pred <- predict(pruned_tree, type = "class")

# Getting Confusion matrix
prune_mat <- table(Predicted = prune_pred, Actual = drug$Drug)
prune_mat
```

```
##          Actual
## Predicted drugA drugB drugC drugX drugY
##     drugA    23     0     0     0     0
##     drugB     0    16     0     0     0
##     drugC     0     0    16     0     0
##     drugX     0     0     0    54     0
##     drugY     0     0     0     0    91
```

```r
# Accuracy and misclassification rate
prune_acc <- sum(diag(prune_mat)) / sum(prune_mat)
prune_misclass <- 1 - prune_acc

cat("Pruned Tree - Accuracy:", prune_acc, "\n")
```

```
## Pruned Tree - Accuracy: 1
```

```r
cat("Pruned Tree - Misclassification Rate:", prune_misclass, "\n")
```

```
## Pruned Tree - Misclassification Rate: 0
```

In both cases, we observe a 100% accuracy. This is usually an indicator of overfitting, but because the dataset is small and well-separated, we observe no misclassifications. Also, the accuracy and misclassifications of the two datasets are identical because pruning couldn't take place based on the threshold values assumed.

9. Fitting a Bagging Model

```r
library(randomForest)
```

```
## randomForest 4.7-1.2
```

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

```r
# Bagging: mtry is set to the total number of predictors
bagging_model <- randomForest(Drug ~ ., data = drug, mtry = 5, importance = TRUE)

# Printing model summary
print(bagging_model)
```

```
##
## Call:
##  randomForest(formula = Drug ~ ., data = drug, mtry = 5, importance = TRUE)
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 5
##
##         OOB estimate of  error rate: 1%
## Confusion matrix:
##       drugA drugB drugC drugX drugY class.error
## drugA    23     0     0     0     0  0.00000000
## drugB     1    15     0     0     0  0.06250000
## drugC     0     0    16     0     0  0.00000000
## drugX     0     0     0    53     1  0.01851852
## drugY     0     0     0     0    91  0.00000000
```

Fitting a Random Forest Model

```r
# Random Forest: assuming mtry = 2
rf_model <- randomForest(Drug ~ ., data = drug, mtry = 2, importance = TRUE)

# Printing model summary
print(rf_model)
```

```
##
## Call:
##  randomForest(formula = Drug ~ ., data = drug, mtry = 2, importance = TRUE)
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 2
##
##         OOB estimate of  error rate: 1%
## Confusion matrix:
##       drugA drugB drugC drugX drugY class.error
## drugA    23     0     0     0     0  0.00000000
## drugB     1    15     0     0     0  0.06250000
## drugC     0     0    16     0     0  0.00000000
## drugX     0     0     0    53     1  0.01851852
## drugY     0     0     0     0    91  0.00000000
```

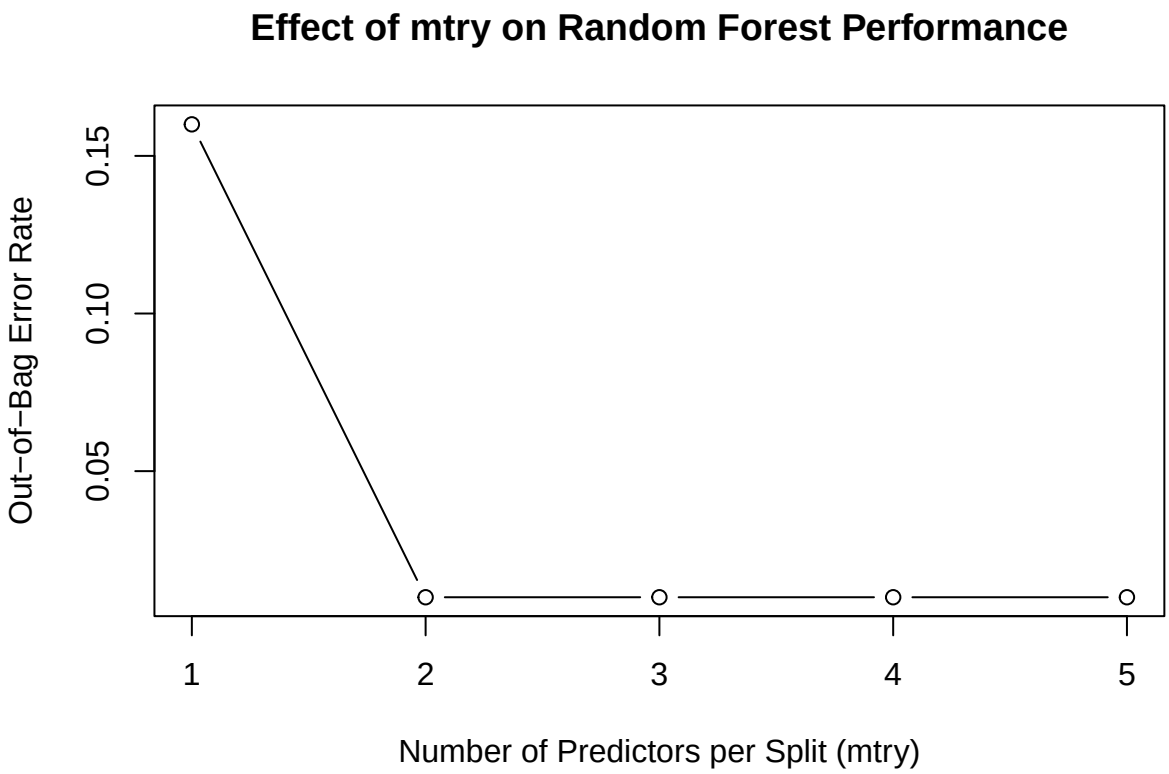
Changing the value of the number of parameters and then observing the results

```
# Initialising vector to store results
mtry_vals <- 1:5  # 5 predictors in the dataset
oob_error <- numeric(length(mtry_vals))

# Looping over different mtry values
for (i in seq_along(mtry_vals)) {
  rf_temp <- randomForest(Drug ~ ., data = drug, mtry = mtry_vals[i])
  oob_error[i] <- rf_temp$err.rate[nrow(rf_temp$err.rate), "OOB"]
}

# Creating a data frame to display results
mtry_results <- data.frame(mtry = mtry_vals, OOB_Error = oob_error)
print(mtry_results)
```

```
##   mtry OOB_Error
## 1    1      0.16
## 2    2      0.01
## 3    3      0.01
## 4    4      0.01
## 5    5      0.01
```

Printing the results of the above experiment

```
plot(mtry_results$mtry, mtry_results$OOB_Error, type = "b",
     xlab = "Number of Predictors per Split (mtry)",
     ylab = "Out-of-Bag Error Rate",
     main = "Effect of mtry on Random Forest Performance")
```



From the results and the plot, we see that there is no significant decrease in error on increasing error beyond 2. So, mtry = 2 would be an ideal value.

11. Obtaining the Best Model using Parameter Tuning

```
# Trying mtry values from 1 to 5
mtry_vals <- 1:5
oob_error <- numeric(length(mtry_vals))
rf_models <- list()

# Looping over mtry values
for (i in seq_along(mtry_vals)) {
  rf_models[[i]] <- randomForest(Drug ~ ., data = drug, mtry = mtry_vals[i])
  oob_error[i] <- rf_models[[i]]$err.rate[nrow(rf_models[[i]]$err.rate), "OOB"]
}

# Getting best mtry (minimum OOB error)
best_index <- which.min(oob_error)
best_mtry <- mtry_vals[best_index]
best_model <- rf_models[[best_index]]

cat("Best mtry value:", best_mtry, "\n")
```

```
## Best mtry value: 2
```

```r
cat("OOB error for best model:", oob_error[best_index], "\n")
```

```
## OOB error for best model: 0.01
```

Calculating and Printing the Accuracy and Misclassification Rate of the Model with Best mtry Value

```r
# Predicting on training data using the best model
best_pred <- predict(best_model, type = "class")

# Confusion matrix
conf_mat <- table(Predicted = best_pred, Actual = drug$Drug)
print(conf_mat)
```

```
##          Actual
## Predicted drugA drugB drugC drugX drugY
##     drugA    23     1     0     0     0
##     drugB     0    15     0     0     0
##     drugC     0     0    16     0     0
##     drugX     0     0     0    53     0
##     drugY     0     0     0     1    91
```

```r
# Accuracy and misclassification
accuracy <- sum(diag(conf_mat)) / sum(conf_mat)
misclass <- 1 - accuracy

cat("Accuracy of Best Model:", accuracy, "\n")
```

```
## Accuracy of Best Model: 0.99
```

```r
cat("Misclassification Rate:", misclass, "\n")
```

```
## Misclassification Rate: 0.01
```

So, we get the best mtry value as 2 with an accuracy of **99%** and and a misclassification rate of **1%**.

Conclusion

- A classification tree was fitted using the `rpart` package.
- The tree was pruned using the lowest`cp` value, although it did not get pruned because the value was not sufficient.
- Accuracy and misclassification rate were calculated for both original and pruned trees.
- Bagging and Random Forest models were implemented using the `randomForest` package.
- The number of predictors per split (`mtry`) was varied to observe its effect on performance.
- OOB error was used as the evaluation metric for model comparison.
- The best model was selected based on lowest OOB error.
- Final accuracy and misclassification rate were reported for the selected model.