ITERA

Modul 9 Praktikum
Statistika Sains Data

Metode Berbasis Pohon

Program Studi Sains Data
Fakultas Sains
Institut Teknologi Sumatera

2024

A. Tujuan Praktikum

1. Mahasiswa dapat menggunakan model decision tree
2. Mahasiswa dapat menerapkan regresi tree untuk melakukan prediksi model

A. Teori Dasar

Decision Tree (pohon keputusan) dapat dipandang sebagai diagram alir (flowchart) untuk memberikan label kelas terhadap suatu pengamatan. Observasi yang ada dalam satu cabang pohon memiliki ciri-ciri yang sama .Hal-hal yang harus diperhatikan dalam membangun pohon keputusan;

1. Semakin banyak peubah yang terlibat, maka semakin banyak kemungkinan pohon keputusan yang bisa dibuat.
2. Algoritma yang paling efisien dalam membangun pohon keputusan yang optimum kemungkinan besar tidak ada.
3. Sehingga banyak alternatif algoritma yang digunakan.

Dalam hal study kasus kali ini, pohon klasifikasi diterapkan pada kumpulan data Carseats setelah mengubah Penjualan (sales) menjadi variabel respons kualitatif. Sekarang kita akan berusaha memprediksi Penjualan menggunakan pohon regresi dan pendekatan terkait, memperlakukan respon sebagai variabel kuantitatif.

Pisahkan kumpulan data menjadi kumpulan pelatihan dan kumpulan pengujian.

Splited data into 60% to training set and 40% to test set.

```
library(ISLR)
library(knitr)
data(Carseats)
str(Carseats)
```

```
## 'data.frame':    400 obs. of  11 variables:
##  $ Sales       : num  9.5 11.22 10.06 7.4 4.15 ...
##  $ CompPrice   : num  138 111 113 117 141 124 115 136 132 132 ...
##  $ Income      : num  73 48 35 100 64 113 105 81 110 113 ...
##  $ Advertising: num  11 16 10 4 3 13 0 15 0 0 ...
##  $ Population : num  276 260 269 466 340 501 45 425 108 131 ...
##  $ Price       : num  120 83 80 97 128 72 108 120 124 124 ...
##  $ ShelveLoc   : Factor w/ 3 levels "Bad","Good","Medium": 1 2 3 3 1 1 3 2 3 3 ...
##  $ Age         : num  42 65 59 55 38 78 71 67 76 76 ...
##  $ Education   : num  17 10 12 14 13 16 15 10 10 17 ...
##  $ Urban       : Factor w/ 2 levels "No","Yes": 2 2 2 2 2 1 2 2 1 1 ...
##  $ US          : Factor w/ 2 levels "No","Yes": 2 2 2 2 1 2 1 2 1 2 ...
```

```
attach(Carseats)
set.seed(702)
train <- sample(1:nrow(Carseats), .6*nrow(Carseats))

Carseats.train = Carseats[train, ]
Carseats.test = Carseats[-train, ]
kable(head(Carseats))
```

Sales	CompPrice	Income	Advertising	Population	Price	ShelveLoc	Age	Education	Urban	US
9.50	138	73	11	276	120	Bad	42	17	Yes	Yes
11.22	111	48	16	260	83	Good	65	10	Yes	Yes
10.06	113	35	10	269	80	Medium	59	12	Yes	Yes
7.40	117	100	4	466	97	Medium	55	14	Yes	Yes

Sales	CompPrice	Income	Advertising	Population	Price	ShelveLoc	Age	Education	Urban	US
4.15	141	64	3	340	128	Bad	38	13	Yes	No
10.81	124	113	13	501	72	Bad	78	16	No	Yes

Gunakan pohon regresi ke set pelatihan. Plot pohonnya, dan interpretasikan hasilnya. Tes MSE apa yang dapatkan.

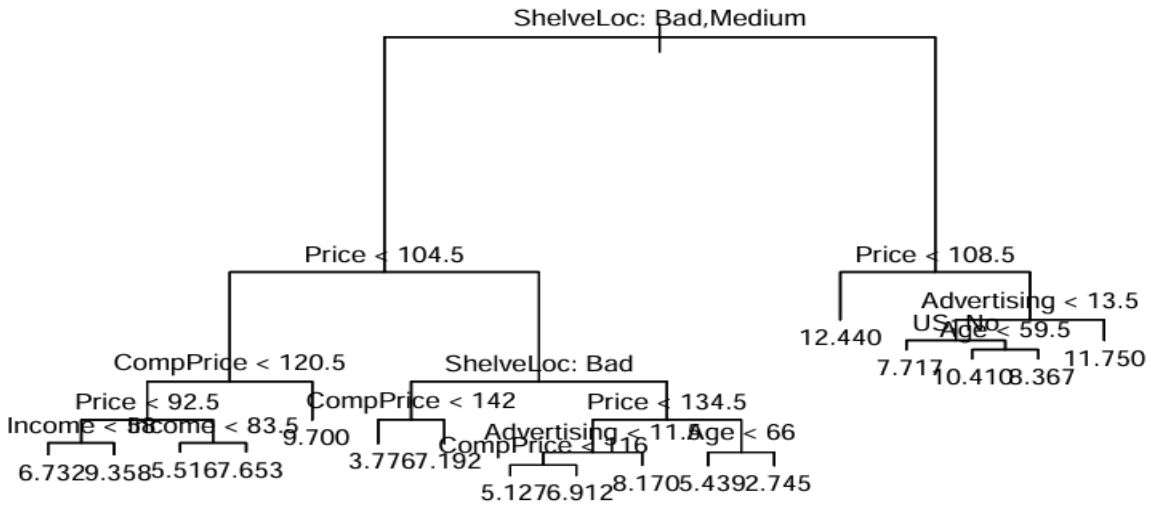
```
library(tree)
```

Warning: package 'tree' was built under R version 4.2.3

```
tree.carseats = tree(Sales ~ ., data = Carseats.train)
summary(tree.carseats)
```

```
##
## Regression tree:
## tree(formula = Sales ~ ., data = Carseats.train)
## Variables actually used in tree construction:
## [1] "ShelveLoc"   "Price"  "CompPrice"          "Income"       "Advertising"
## [6] "Age"             "US"
## Number of terminal nodes:  17
## Residual mean deviance:  2.239 = 499.3 / 223
## Distribution of residuals:
##       Min.   1st Qu.  Median          Mean   3rd Qu.          Max.
## -4.13700 -1.04200  0.08303  0.00000  1.09400  4.54400
```

```
plot(tree.carseats)
text(tree.carseats,pretty = 0, cex=.75)
```

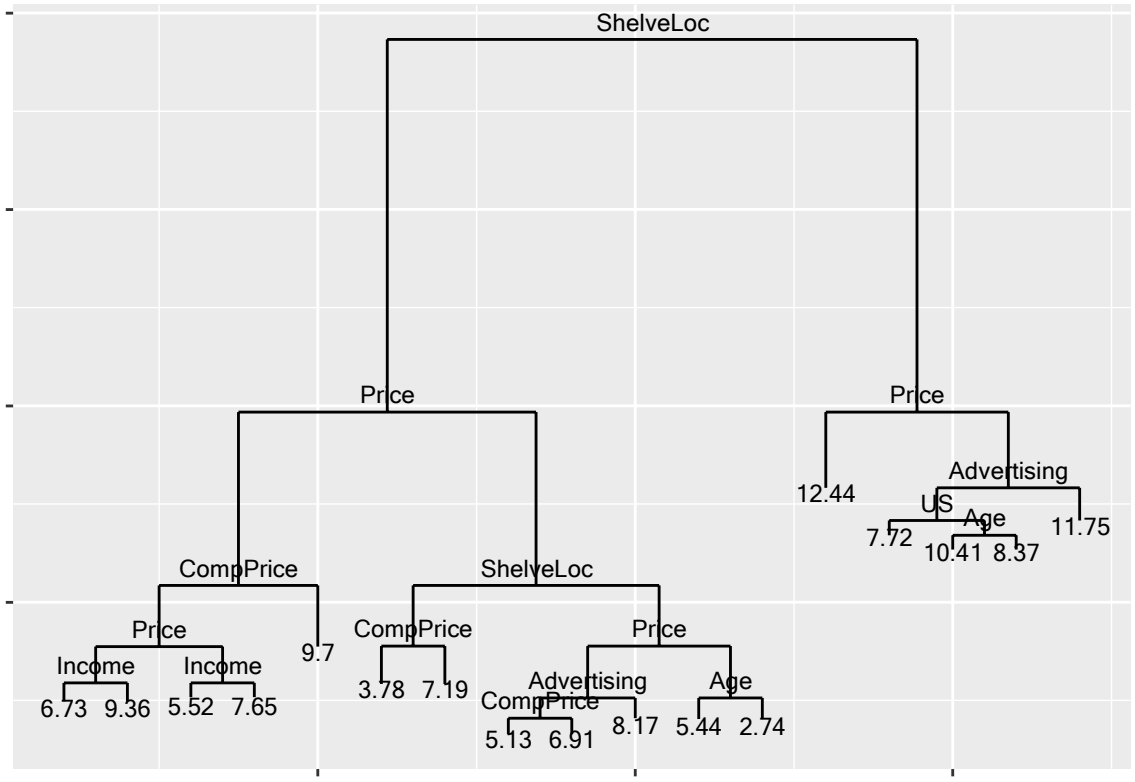


ggplot for carseats tree

```
library(ggplot2)
library(ggdendro)
```

```
## Warning: package 'ggdendro' was built under R version 4.2.3
```

```
tree_data <- dendro_data(tree.carseats)
ggplot() +
  geom_segment(data = tree_data$segments,
               aes(x = x, y = y, xend = xend, yend = yend)) +
  geom_text(data = tree_data$labels,
            aes(x = x, y = y, label = label), size = 3, vjust = -0.5) +
  geom_text(data = tree_data$leaf_labels,
            aes(x = x, y = y, label = label), size = 3, vjust = 0.85)
```




```
#theme_dendro()
```

```
pred.carseats = predict(tree.carseats, Carseats.test)MSE<-
mean((Carseats.test$Sales - pred.carseats)^2)
paste("Test MSE of tree model  = ", MSE)
```

```
## [1] "Test MSE of tree model  =  4.66309786076511"
```

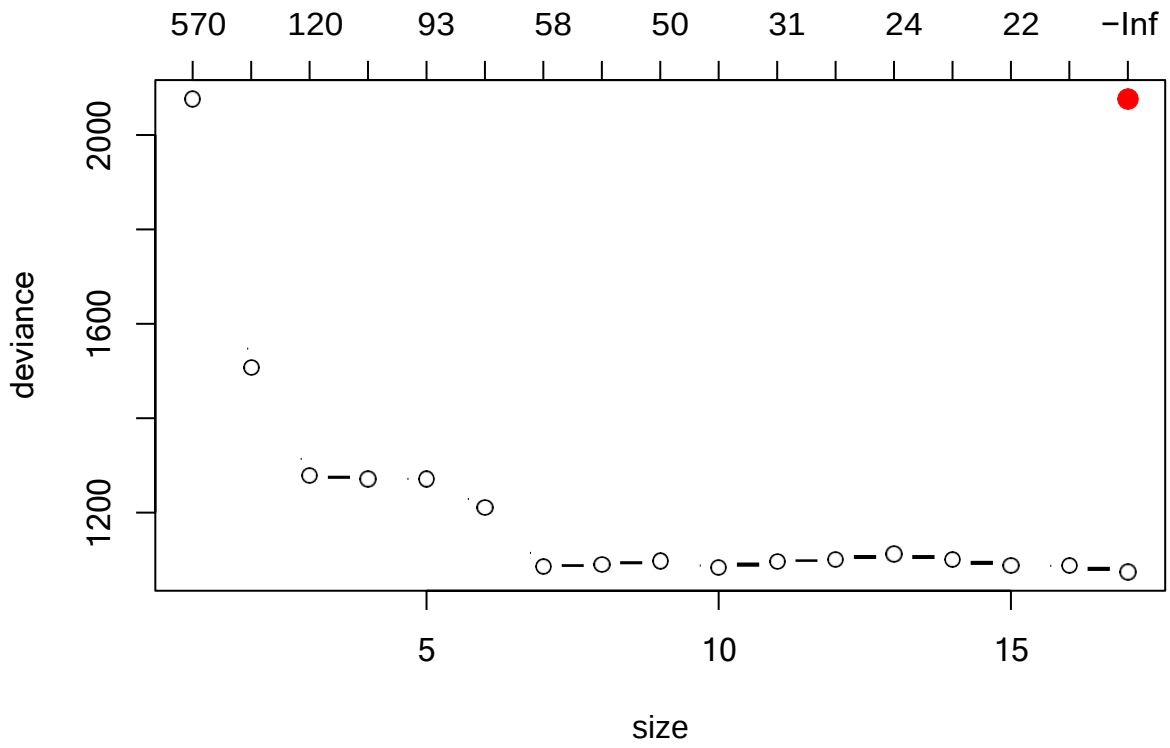
Gunakan validasi silang untuk menentukan tingkat kompleksitas pohon yang optimal. Apakah pemangkasan (pruning) pohon meningkatkan tes MSE.

```
# Perform cross-validation and plot
set.seed(702)
cv.carseats <- cv.tree(tree.carseats, FUN=prune.tree)
plot(cv.carseats, type="b")
```

```
size.min <- cv.carseats$size[which.min(cv.carseats$dev)]
paste("Size with the lowest deviance: ", size.min)
```

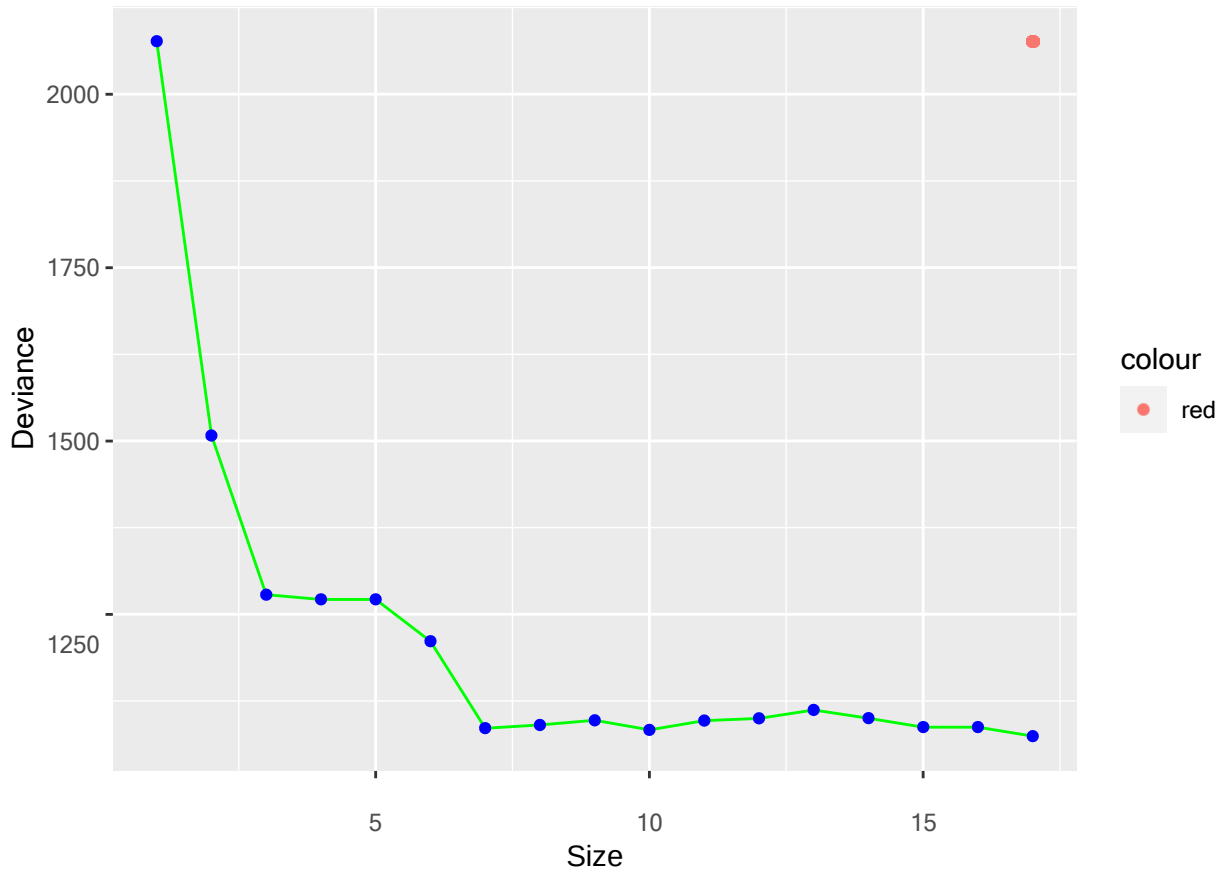
```
## [1] "Size with the lowest deviance:  17"
```

```
points(size.min, cv.carseats$dev[size.min], col = "red", cex = 2, pch = 20)
```



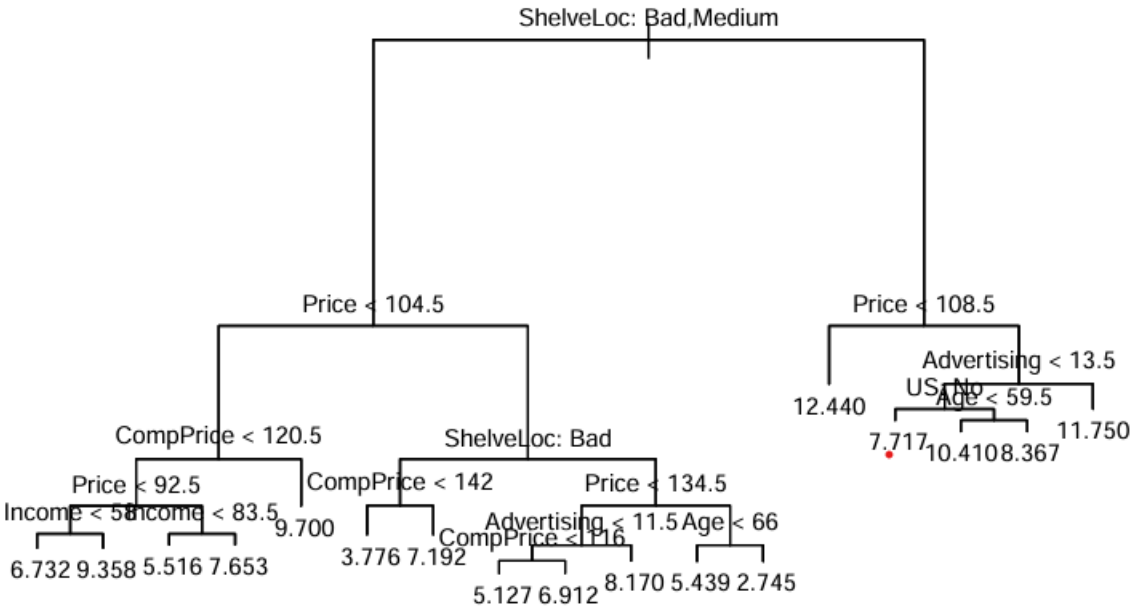
ggplot

```
Size<-c(cv.carseats$size)
Deviance<-c(cv.carseats$dev)
da<-data.frame(Size,Deviance)
ggplot(da, aes(x=Size, y=Deviance))+  geom_line(colour="green")+geom_point(colour="blue") +geom_point(ae
```



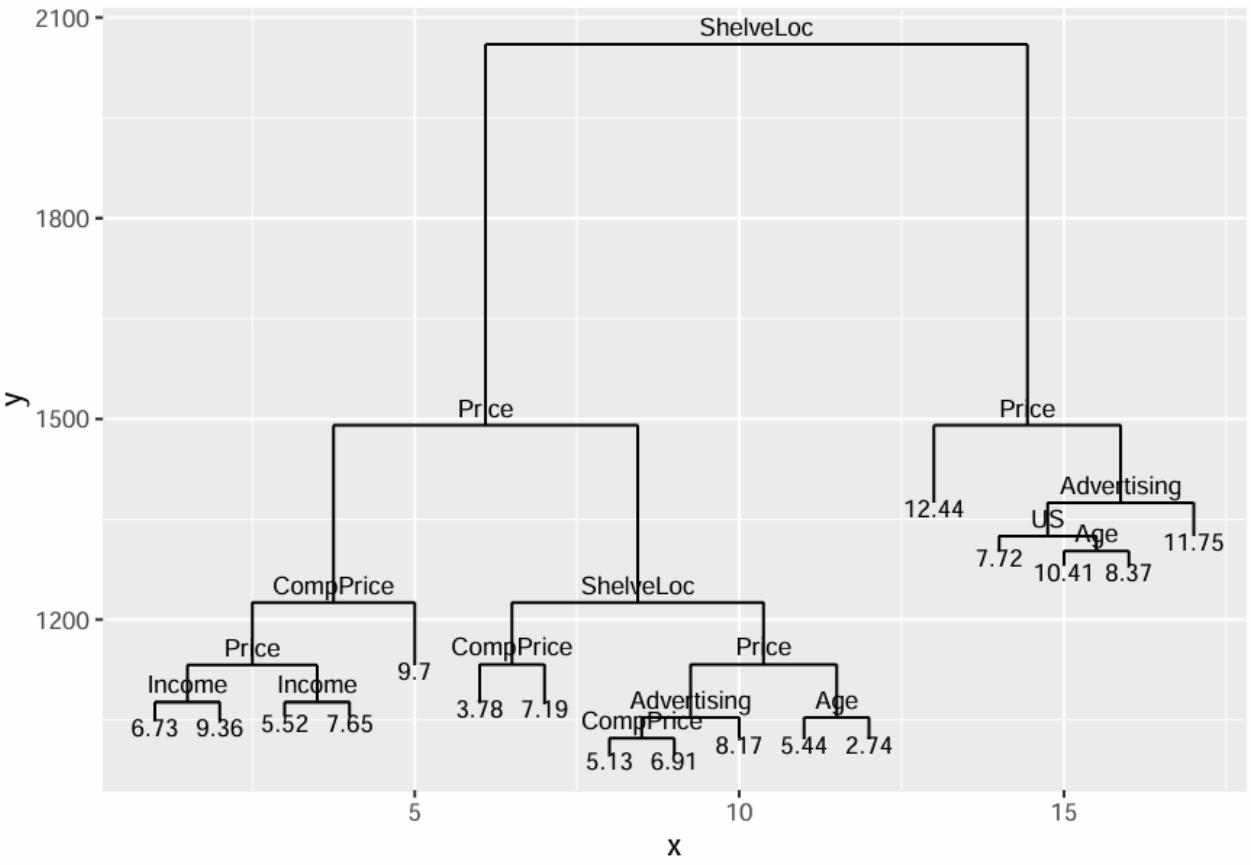
** Prune and create new tree**

```
prune.carseats <- prune.tree(tree.carseats, best=size.min)
plot(prune.carseats)
text(prune.carseats, pretty=0,cex=.65)
```



ggplot purne tree

```
library(ggplot2)
library(ggdendro)
tree_data1 <- dendro_data(prune.carseats)
ggplot() +
  geom_segment(data  =  tree_data1$segments,
                    aes(x = x, y = y, xend = xend, yend = yend)) +
  geom_text(data = tree_data1$labels,
                aes(x = x, y = y, label = label), size = 3, vjust = -0.5) +
  geom_text(data = tree_data1$leaf_labels,
                aes(x = x, y = y, label = label), size = 3, vjust = 0.85)
```




```
#theme_dendro()
```

```
pred.pruned = predict(prune.carseats, Carseats.test)
purn_mse<-mean((Carseats.test$Sales - pred.pruned)^2)
paste("Test MSE of Purned tree model =", purn_mse)
```

[1] "Test MSE of Purned tree model = 4.66309786076511"

Metode Cross Validation memilih ukuran 9 dan memperkecil ukuran pohon tetapi tidak memperbaikikesalahan pengujian.

B. Latihan Praktikum

Decision Tree

Tree library digunakan untuk membangun model klasifikasi dan regression trees.

Pada praktikum ini akan menggunakan pohon klasifikasi untuk menganalisis kumpulan data Carseats. Dalam data ini, Penjualan adalah variabel kontinu, jadi kita mulai dengan mengodekannya kembali sebagai variabel biner. Kita menggunakan fungsi ifelse() untuk membuat sebuah variabel, yang disebut ifelse() High, yang mengambil nilai Ya jika variabel Penjualan melebihi 8, dan sebaliknya mengambil nilai No.

```
data <- Carseats %>%
  mutate(high = factor(if_else(Sales > 8, 1, 0)))

names(data) <-str_to_lower(names(data))
```

Kita sekarang menggunakan fungsi tree() agar sesuai dengan pohon klasifikasi untuk memprediksi 'High' tree() menggunakan semua variabel kecuali Sales(Penjualan). Sintaks fungsi tree() sangat mirip dengan fungsi lm().

```
# Set up initial tree
tree <- tree(high ~ . -sales, data)
```

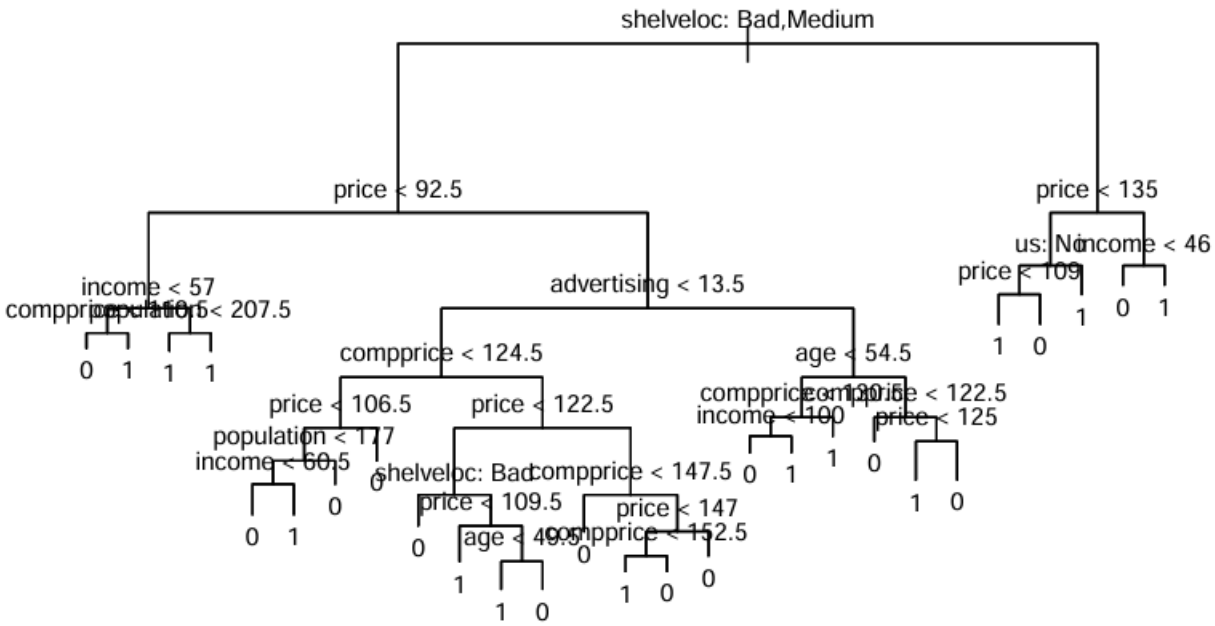
Keluaran dari pohon ini disimpan dalam sebuah list, yang dapat kita rangkum untuk melihat variabel yang digunakan dalam simpul internal, jumlah simpul terminal, dan tingkat kesalahan pelatihan. Anda juga bisa memanggil objek itu sendiri untuk mendapatkan seluruh representasi pohon. Node terminal dilambangkan dengan tanda bintang.

Fungsi summary() mencantumkan variabel yang digunakan sebagai node internal di pohon, jumlah node terminal, dan tingkat kesalahan (pelatihan).

```
summary(tree)
```

```
##
## Classification tree:
## tree(formula = high ~ . - sales, data = data)
## Variables actually used in tree construction:
## [1] "shelveloc"   "price"       "income"      "compprice"   "population"
## [6] "advertising" "age"         "us"
## Number of terminal nodes:  27
## Residual mean deviance:  0.4575 = 170.7 / 373
## Misclassification error rate: 0.09 = 36 / 400
```

Kita tahu bahwa tree dapat dengan mudah diinterpretasikan secara visual, jadi mari kita lihat seperti apa test case kita. Berhati-hatilah saat melakukan plotting karena pohon dapat dengan mudah tumbuh di luar kendali dan membuat grafik tidak dapat dibaca.



Untuk mengevaluasi pohon klasifikasi, kita perlu menggunakan set pelatihan dan pengujian. Sekarang mari ulangi apa yang kita lakukan di atas, kali ini termasuk perhitungan tingkat kesalahan pengujian.

```
# Define our training/testing sets
set.seed(2)
train <- sample_n(data, 200)
test <- setdiff(data, train)

# Run the recursive partioning algorithm
ttree <- tree(high ~. -sales, data = train)

# Make predictions and display confusion matrix
test_predictions <- predict(ttree, test, type = "class")
table(test_predictions, test$high)
```

```
##
## test_predictions   0   1
##                0 104  33
##                1  13  50
```

```
(86 + 57) / 200
```

```
## [1] 0.715
```

Kita sekarang dapat menambahkan lapisan kerumitan lain dengan memangkas (pruning) hasilnya. Ingatlah bahwa pohon yang tidak dipruning rentan terhadap overfitting data, jadi metode kita adalah mengamati variasi dalam tingkat kesalahan pengujian saat kita meningkatkan penalti dalam jumlah node terminal. Dapat dilihat pada algoritma berikut:

Algoritma Pruning

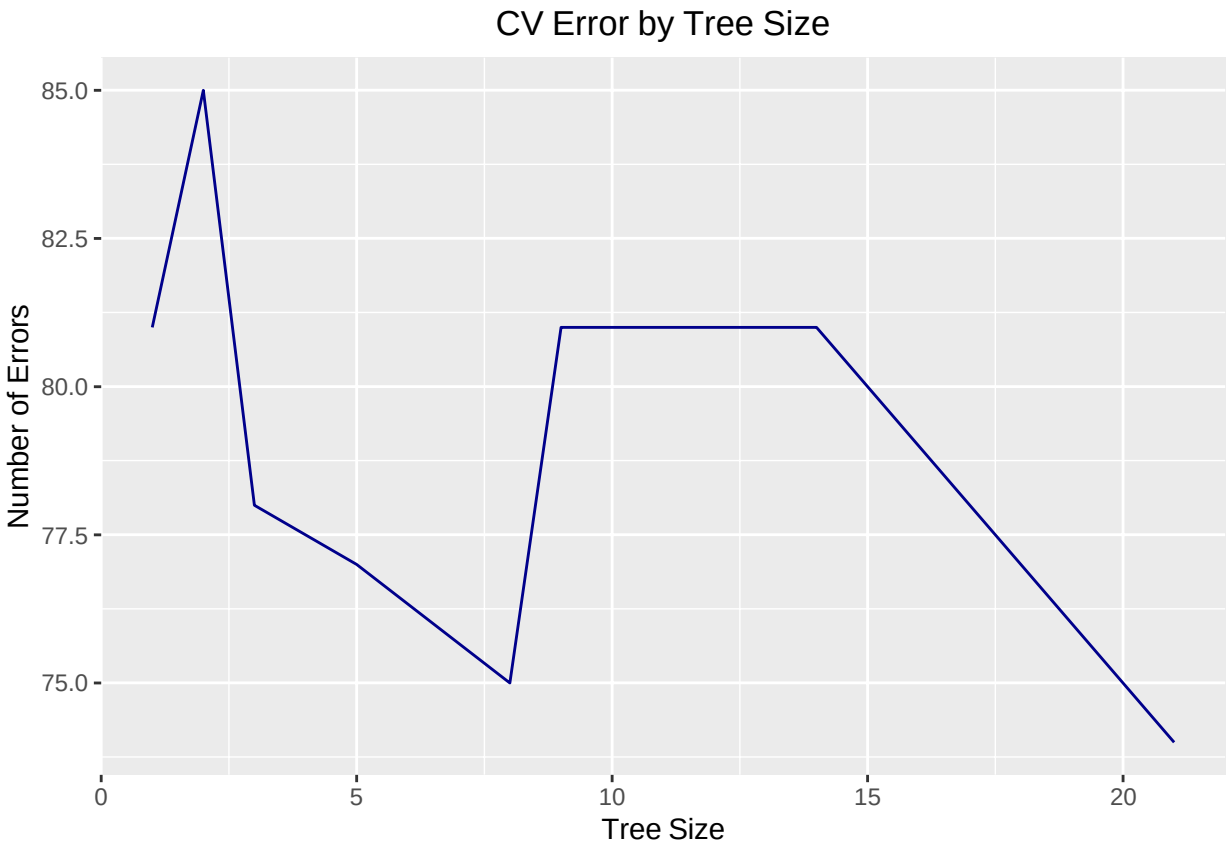
1. Tumbuhkan pohon asli T_0 menggunakan data pelatihan.
2. Sebagai fungsi dari α (parameter pinalti), tentukan urutan yang terbaik subpohon
3. Gunakan validasi silang K-fold untuk menemukan α yang meminimalkan kesalahan prediksi rata-rata kuadrat rata-rata dari kth fold data pelatihan
4. Temukan subpohon terbaik dari Langkah 2 menggunakan α yang ditemukan di langkah sebelumnya.

Selanjutnya, kita mempertimbangkan apakah pemangkasan pohon dapat memberikan hasil yang lebih baik. Fungsi cv.tree() melakukan validasi silang untuk cv.tree() menentukan tingkat kompleksitas pohon yang optimal; Pemangkasan kompleksitas yang digunakan untuk memilih urutan pohon untuk dipertimbangkan. Kita menggunakan argumen FUN=prune.misclass untuk menunjukkan bahwa kami ingin tingkat kesalahan klasifikasi memandu proses validasi silang dan pemangkasan, daripada default untuk fungsi cv.tree(), yang merupakan penyimpangan. Fungsi cv.tree() melaporkan jumlah node terminal dari setiap pohon yang dianggap (ukuran) serta tingkat kesalahan yang sesuai dan nilai dari parameter kompleksitas yang digunakan.

```
set.seed(3)
cv_tree <- cv.tree(ttree, FUN = prune.misclass)
cv_tree
```

```
## $size
## [1] 21 19 14  9  8  5  3  2  1
##
## $dev
## [1] 74 76 81 81 75 77 78 85 81
##
## $k
## [1] -Inf  0.0  1.0  1.4  2.0  3.0  4.0  9.0 18.0
##
## $method
## [1] "misclass"
##
## attr(,"class")
## [1] "prune"         "tree.sequence"
```

Yang paling penting dalam hasil ini adalah `$dev`, yang sesuai dengan kesalahan validasi silang di setiap instance. Kita dapat melihat bahwa nilai terkecil terjadi ketika ada 21 node terminal. Mari kita lihat sekilas bagaimana kesalahan bervariasi dalam jumlah node terminal kita:



Sekarang setelah kita tahu berapa tepatnya node terminal yang kita inginkan, kita pangkas pohon kita dengan `prune.misclass()` untuk mendapatkan pohon yang optimal. Kemudian, periksa untuk melihat apakah pohon ini berperforma lebih baik pada set pengujian daripada pohon dasar T_0.

```
pruned <- prune.misclass(ttree, best = 21)

test_predictions <- predict(pruned, data = test, type = "class")
table(test_predictions, test$high)
```

```
##
## test_predictions  0  1
##                0 79 49
##                1 38 34
```

```
(94 + 60) / 200
```

```
## [1] 0.77
```

Fitting Regression Tree

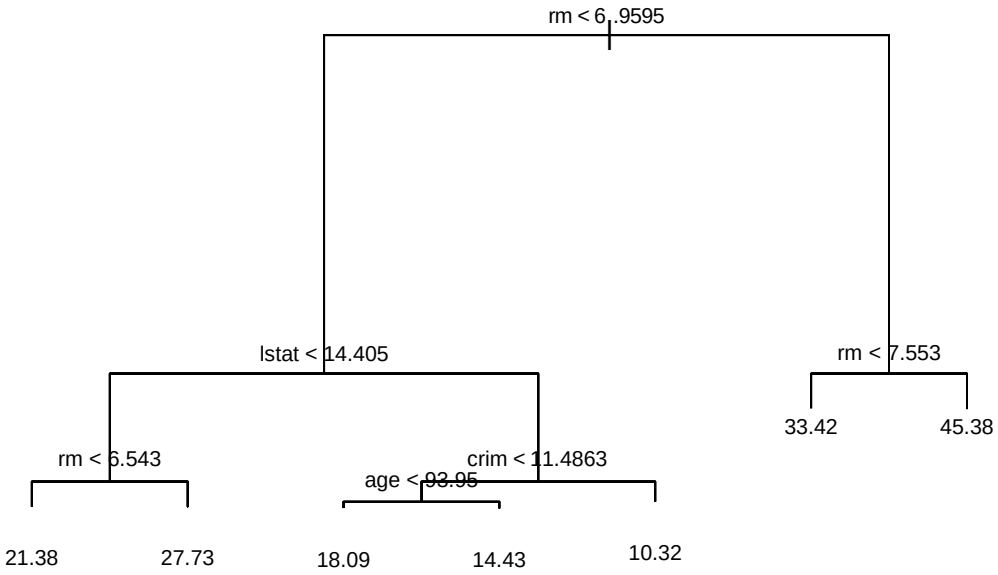
Tidak banyak perubahan dalam hal kode ketika kita beralih ke pohon regresi, jadi bagian ini akan menjadi rekap dari yang sebelumnya, hanya menggunakan data yang berbeda. Kita menggunakan data `Boston` dari library `MASS` untuk latihan ini.

```
Boston <- MASS::Boston
set.seed(1)
train <- sample_frac(Boston, .5)
test <- setdiff(Boston, train)
```

```
tree_train <- tree(medv ~., data = train)
summary(tree_train)
```

```
##
## Regression tree:
## tree(formula = medv ~ ., data = train)
## Variables actually used in tree construction:
## [1] "rm"    "lstat" "crim"  "age"
## Number of terminal nodes:  7
## Residual mean deviance:  10.38 = 2555 / 246
## Distribution of residuals:
##     Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
## -10.1800  -1.7770  -0.1775   0.0000   1.9230  16.5800
```

```
plot(tree_train)
text(tree_train, pretty = 0, cex = .65)
```

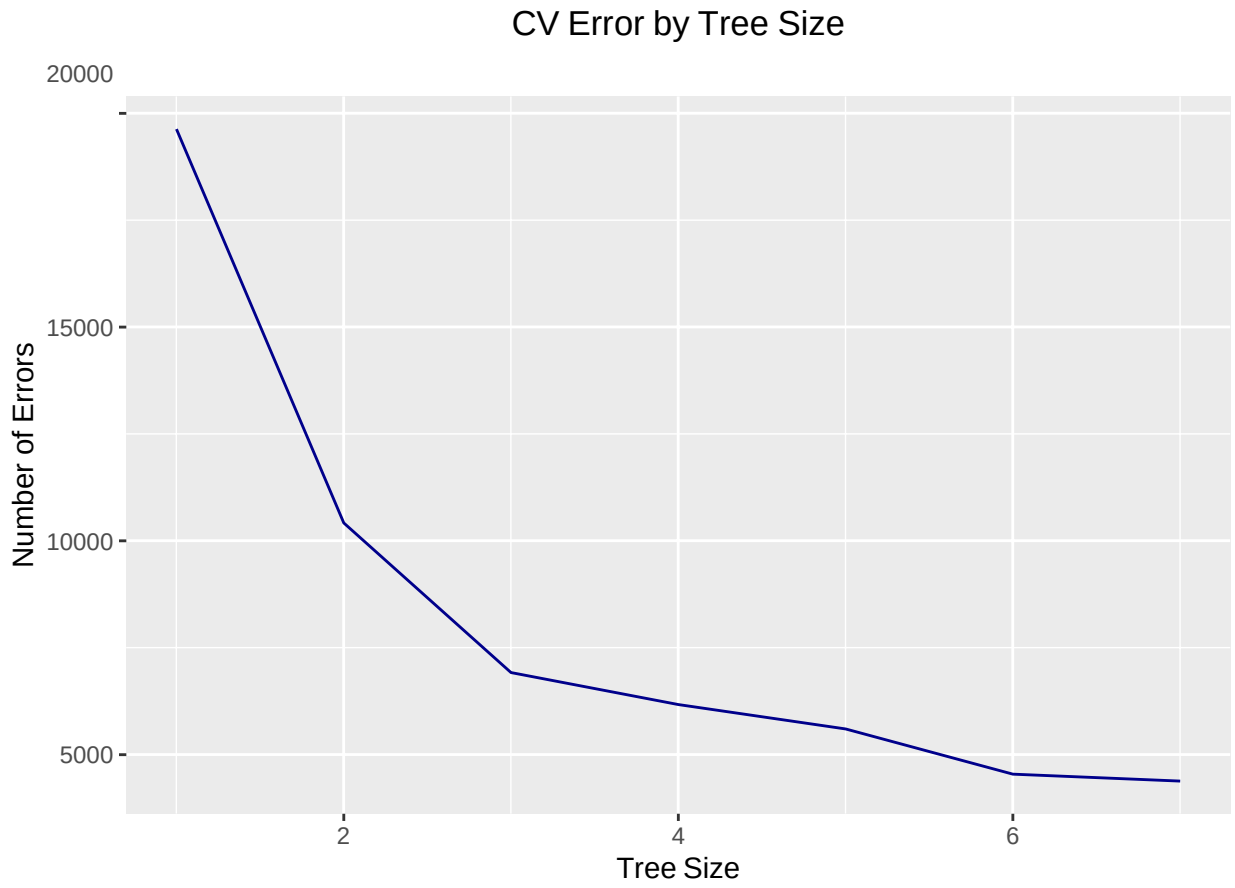


Seperti yang Anda lihat `lstat < 14.405` adalah partisi pertama di pohon ini. Variabel mengukur persentase individu dengan status sosial ekonomi yang lebih rendah di daerah terdekat. Berdasarkan dari node terminal yang berasal dari sisi kiri pohon, ini menunjukkan bahwa wilayah geografis sosial ekonomi yang lebih tinggi berakhir dengan harga rumah rata-rata yang jauh lebih besar.

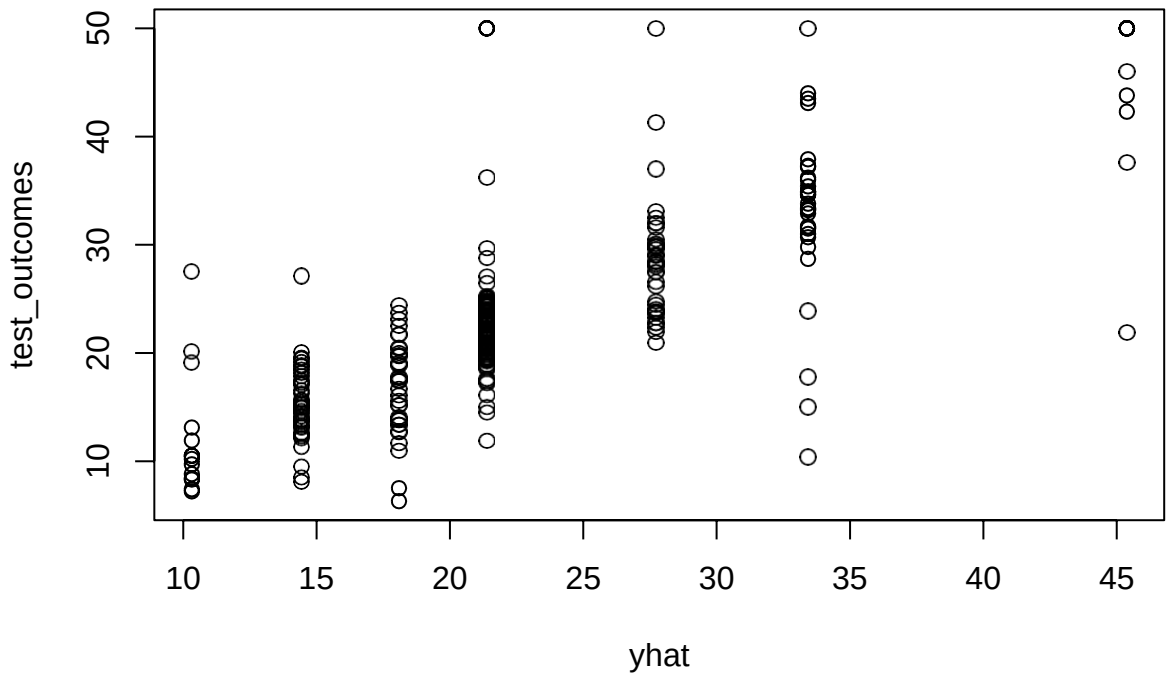
Kita sekarang dapat melanjutkan untuk melihat apakah pemangkasan akan meningkatkan kinerja pohon ini

```
cv_tree <- cv.tree(tree_train)

# Get an idea of change in error by changing tree size
```

```r
ggplot(data = data.frame(cv_tree$size, cv_tree$dev),
  aes(x = cv_tree$size, y = cv_tree$dev)) +
  geom_line(color = "darkblue") +
  labs(x = "Tree Size", y = "Number of Errors", title = "CV Error by Tree Size") +
  theme(plot.title = element_text(hjust = .5))
```




```r
# Predict, plot, and calculate MSE
yhat <- predict(tree_train, newdata = test)
test_outcomes <- test$medv

plot(yhat, test_outcomes)
```




```
mean((yhat - test_outcomes)^2)
```

```
## [1] 35.28688
```