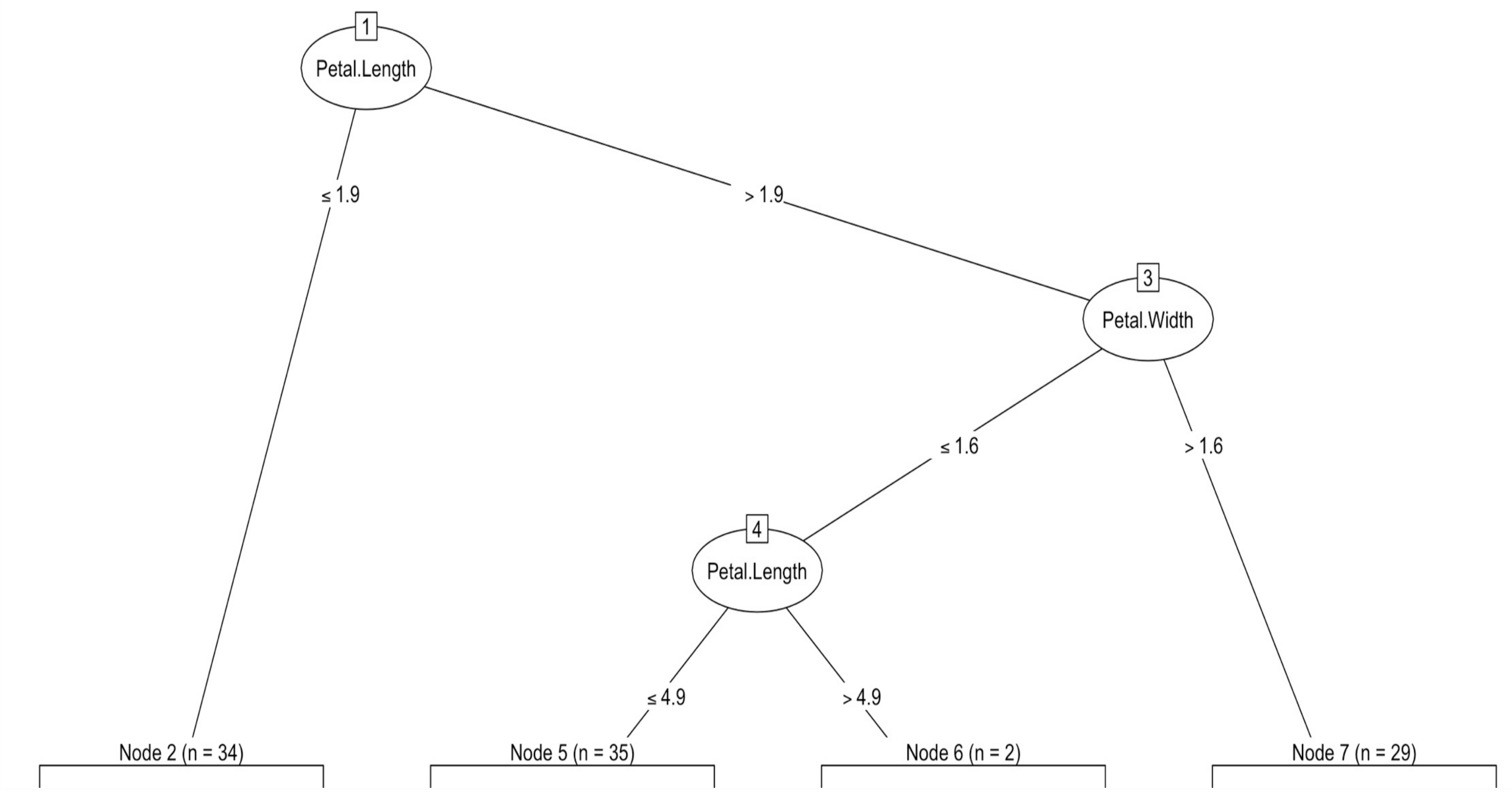
CLASSIFICATION TASKS - Decision Trees

IS 665 Data Mining, Data Warehousing and Visualization

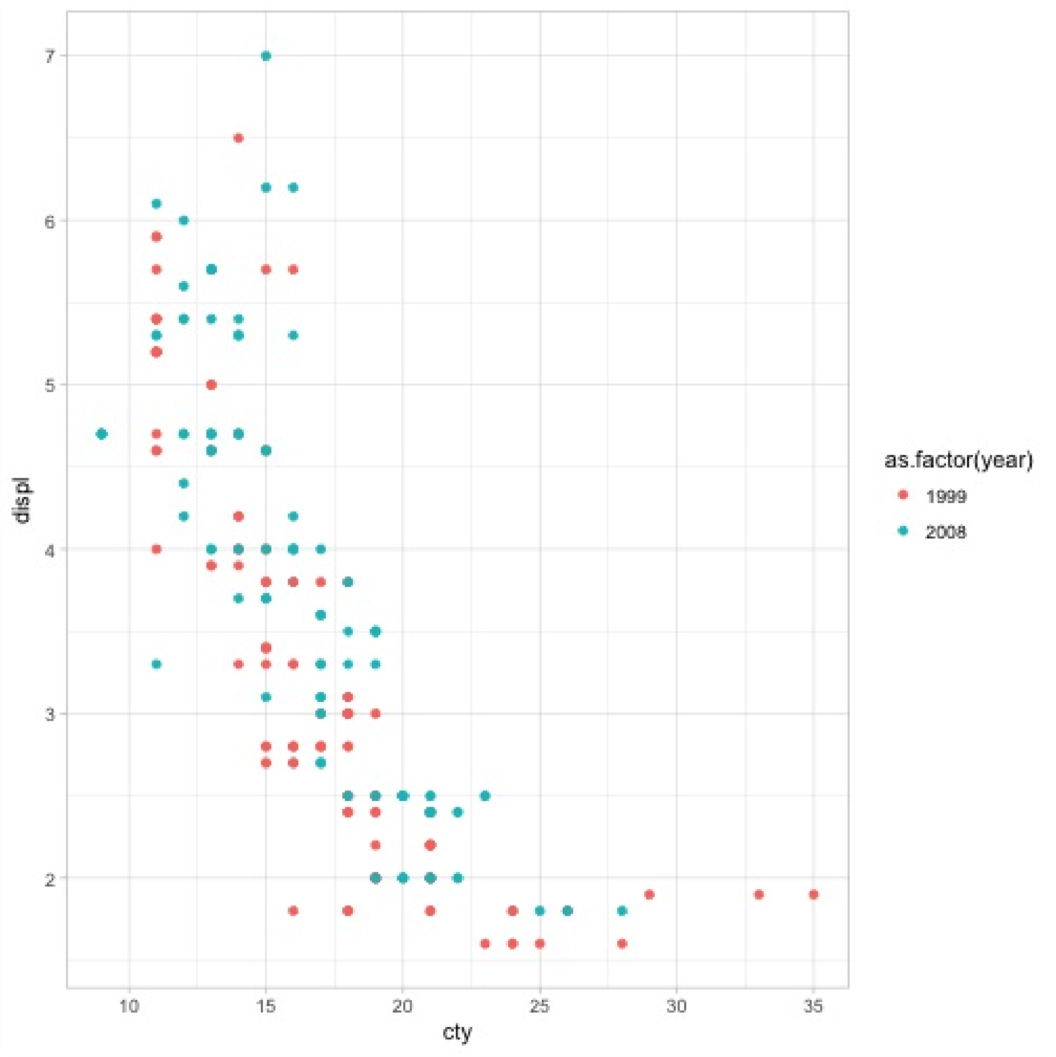
Decision Tree

One attractive classification method involves the construction of a **decision tree**, a collection of *decision nodes*, connected by *branches*, extending downward from the root node until terminating in *leaf nodes*.

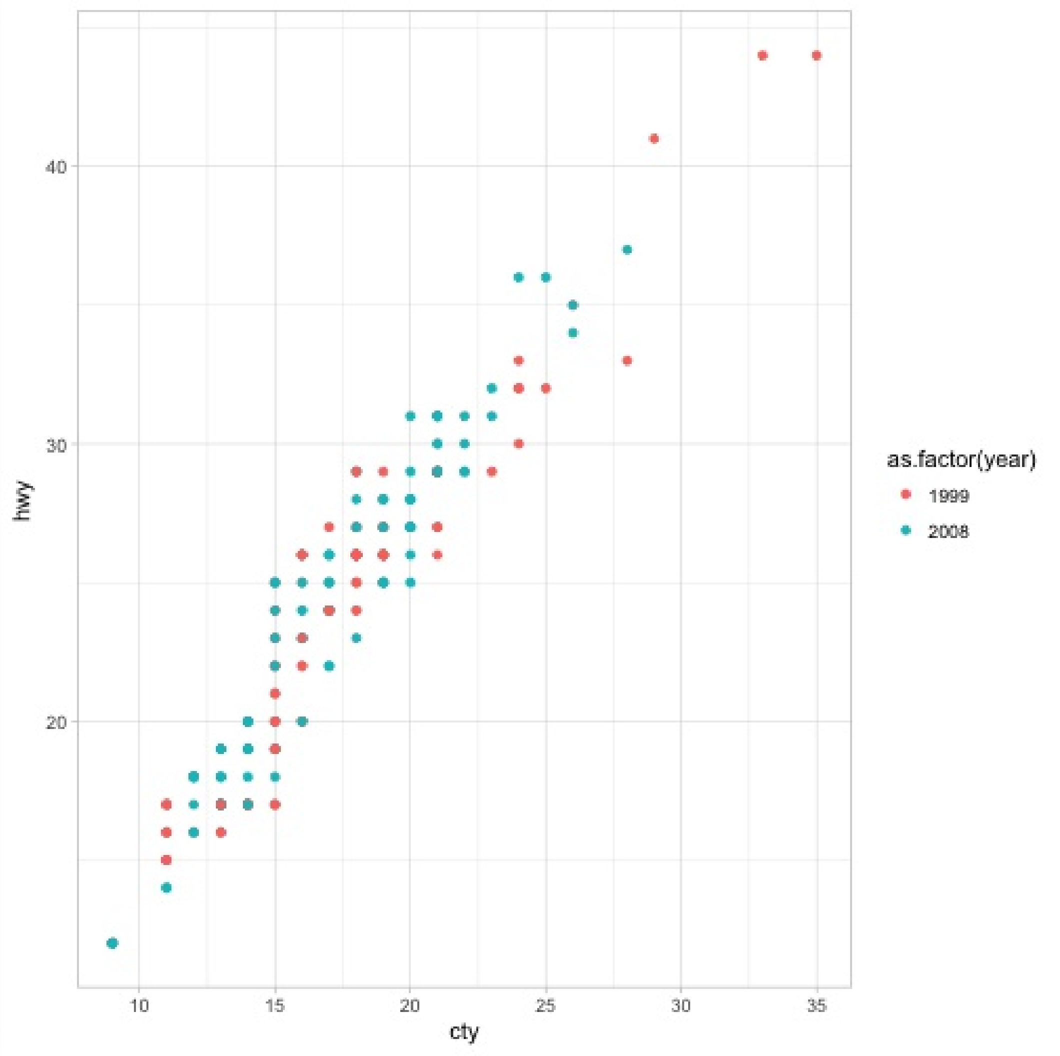
Decision Tree



Method



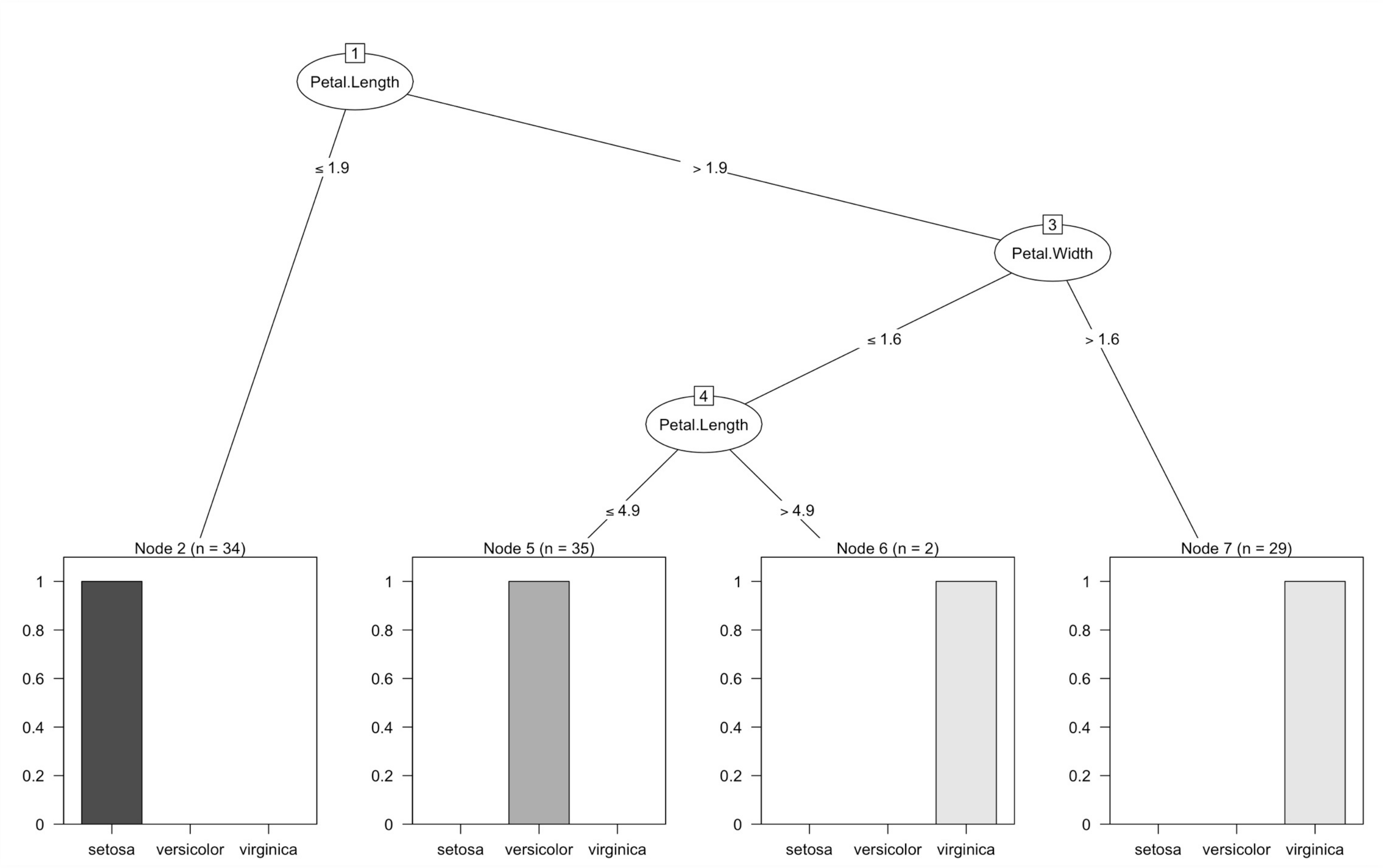
displ
cty
as.factor(year)
1999
2008



hwy
cty
as.factor(year)
1999
2008

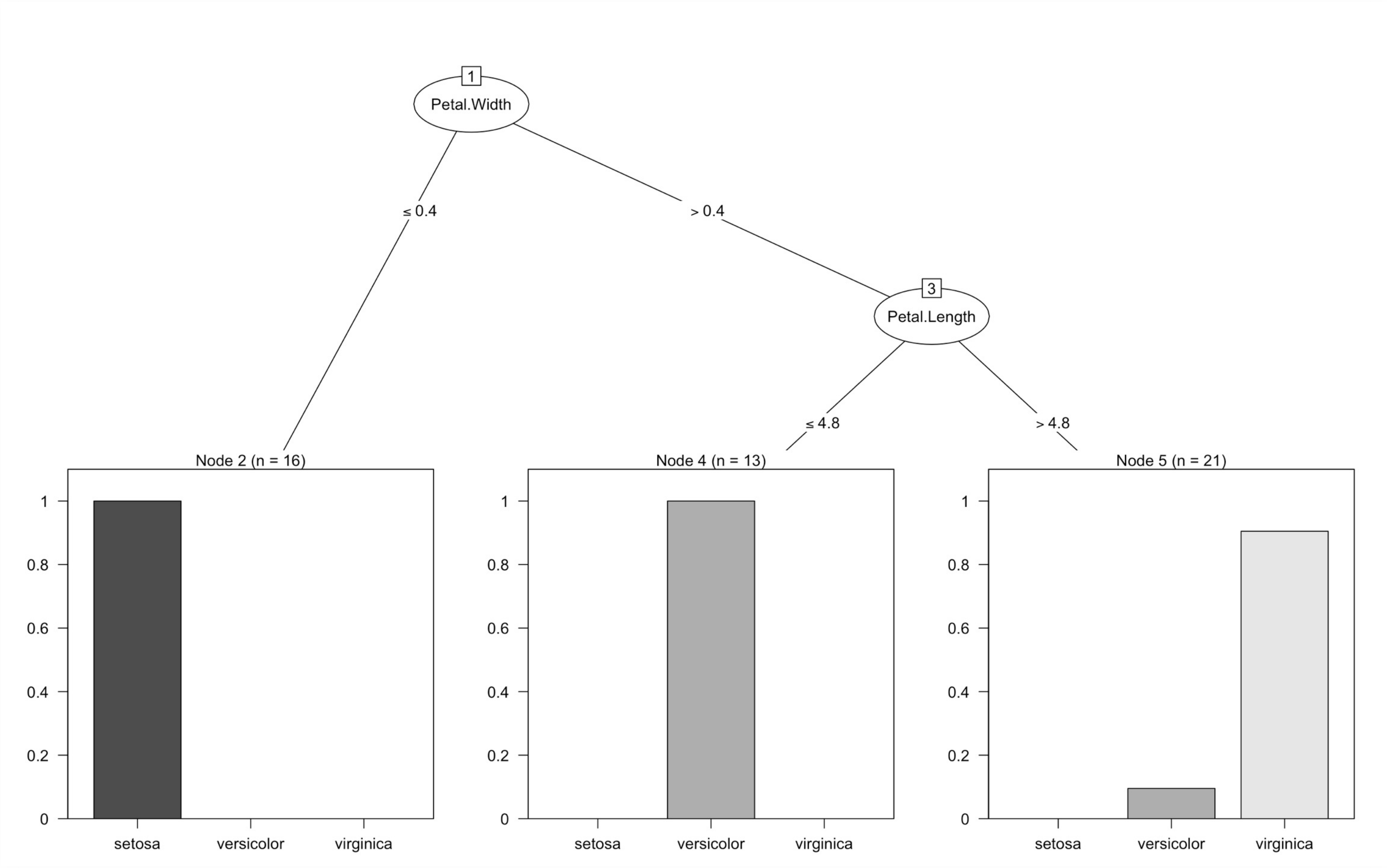
Decision Trees

Decision trees seek to create a set of leaf nodes that are as "pure" as possible



Decision Trees

But usually, there predictions may yield errors



Methodology

- Prepare data (same as knn)
  - normalize non numeric values
  - split training and test data
- Grow the tree (build the model)
- Examine results
- Prune the tree (improve by reducing overfitting errors)

Prepare data

Reminder from last week

```
input_variables <- subset(mpg, select = c(cty, hwy, displ))
n_input_variables = sapply(input_variables, function(x) {
    (x - min(x))/(max(x) - min(x))
})
label = as.factor(mpg$year)
```

Split the data for test

Building a classification model requires a training dataset to train the classification model, and testing data is needed to then validate the prediction performance.

```
# Split
set.seed(1234)
sample_indicies = sample(1:2, size = length(mpg$year), replace = T,
    prob = c(0.8, 0.2))


## Data Split
train_data = n_input_variables[sample_indicies == 1, ]
test_data = n_input_variables[sample_indicies == 2, ]


## Label Split
train_labels = label[sample_indicies == 1]
test_labels = label[sample_indicies == 2]

train_data = data.frame(train_data, train_labels)
test_data = data.frame(test_data, test_labels)
```

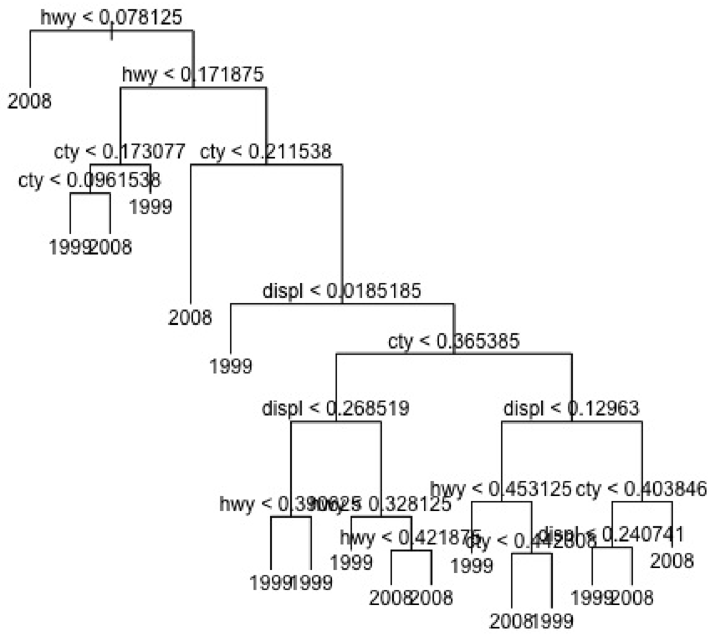
Application of Decision trees

We will use the 'Tree' package

```r
# install.packages('tree')
require("tree")
my.model <- tree(train_labels ~ ., data = train_data)
# or
my.model <- tree(train_labels ~ cty + hwy + displ, data = train_data,
    method = "C50")
```

Examine the model

```r
plot(my.model)
text(my.model, pretty = 0)
```



Examine Model

```
(my.model)
```

```
node), split, n, deviance, yval, (yprob)
      * denotes terminal node

  1) root 190 263.300 2008 ( 0.48947 0.51053 )
    2) hwy < 0.078125 6   0.000 2008 ( 0.00000 1.00000 ) *
    3) hwy > 0.078125 184 255.100 1999 ( 0.50543 0.49457 )
      6) hwy < 0.171875 41  47.690 1999 ( 0.73171 0.26829 )
       12) cty < 0.173077 28  36.500 1999 ( 0.64286 0.35714 )
         24) cty < 0.0961538 14  11.480 1999 ( 0.85714 0.14286 ) *
         25) cty > 0.0961538 14  19.120 2008 ( 0.42857 0.57143 ) *
       13) cty > 0.173077 13   7.051 1999 ( 0.92308 0.07692 ) *
      7) hwy > 0.171875 143 196.200 2008 ( 0.44056 0.55944 )
       14) cty < 0.211538 16   0.000 2008 ( 0.00000 1.00000 ) *
       15) cty > 0.211538 127 176.100 2008 ( 0.49606 0.50394 )
         30) displ < 0.0185185 5   0.000 1999 ( 1.00000 0.00000 ) *
         31) displ > 0.0185185 122 168.800 2008 ( 0.47541 0.52459 )
           62) cty < 0.365385 64  85.640 1999 ( 0.60938 0.39062 )
            124) displ < 0.268519 28  22.970 1999 ( 0.85714 0.14286 )
              248) hwy < 0.390625 12  15.280 1999 ( 0.66667 0.33333 ) *
              249) hwy > 0.390625 16   0.000 1999 ( 1.00000 0.00000 ) *
            125) displ > 0.268519 36  48.900 2008 ( 0.41667 0.58333 )
              250) hwy < 0.328125 14  18.250 1999 ( 0.64286 0.35714 ) *
              251) hwy > 0.328125 22  25.780 2008 ( 0.27273 0.72727 )
                502) hwy < 0.421875 12   6.884 2008 ( 0.08333 0.91667 ) *
                503) hwy > 0.421875 10  13.860 2008 ( 0.50000 0.50000 ) *
           63) cty > 0.365385 58  73.360 2008 ( 0.32759 0.67241 )
            126) displ < 0.12963 31  42.940 1999 ( 0.51613 0.48387 )
              252) hwy < 0.453125 5   0.000 1999 ( 1.00000 0.00000 ) *
```

Prediction

```
my.prediction = predict(my.model, test_data, type = "class")

table(my.prediction, test_data$test_labels)
```

```
my.prediction 1999 2008
         1999   19    7
         2008    5   13
```

```
require(caret)
# install.packages('e1071')
require(e1071)
confusionMatrix(table(my.prediction, test_data$test_labels))
```

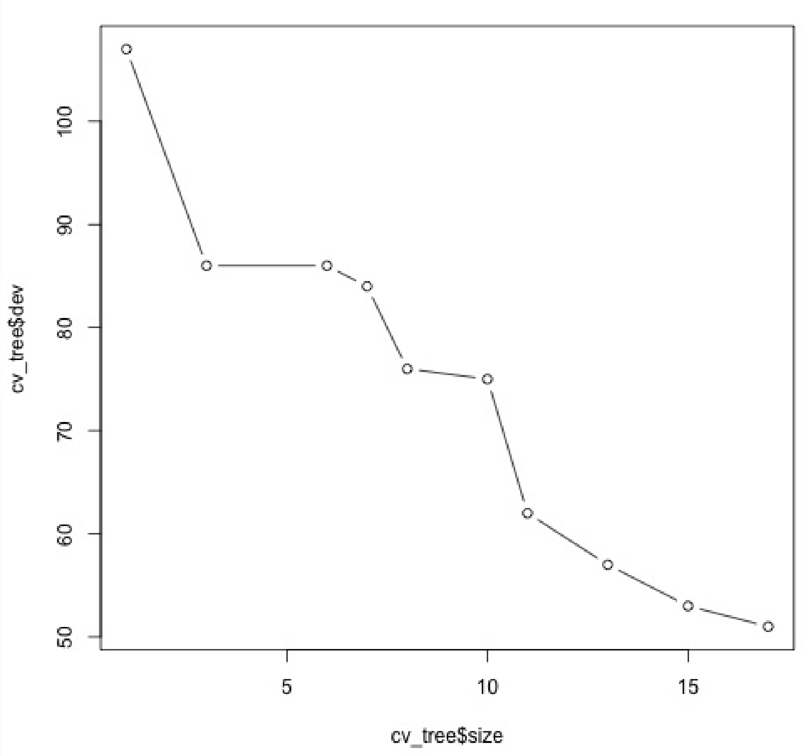
```
Confusion Matrix and Statistics


my.prediction 1999 2008
         1999   19    7
         2008    5   13

               Accuracy : 0.7273
                 95% CI : (0.5721, 0.8504)
    No Information Rate : 0.5455
    P-Value [Acc > NIR] : 0.01043

                  Kappa : 0.4454
 Mcnemar's Test P-Value : 0.77283
```

Pruning

```
cv_tree = cv.tree(my.model, FUN = prune.misclass)
names(cv_tree)
```

```
[1] "size"   "dev"    "k"      "method"
```

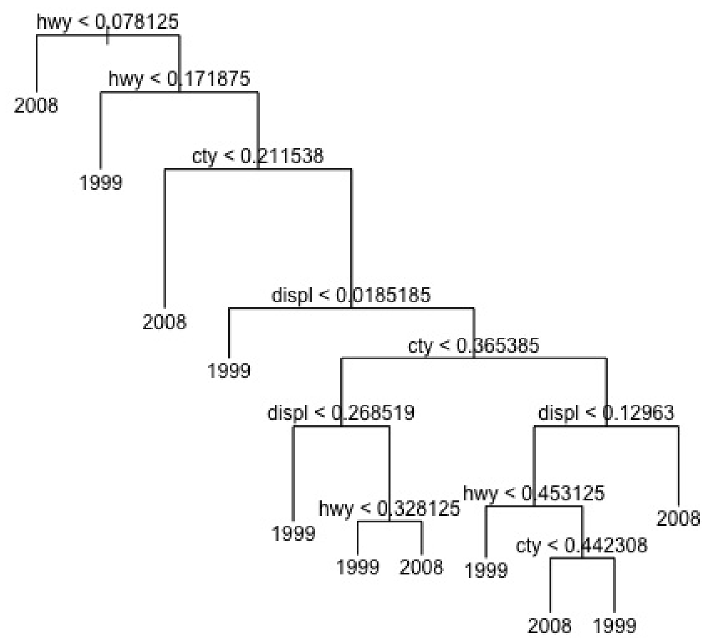
```
plot(cv_tree$size, cv_tree$dev, type = "b")
```



Error is minimized at level 11.

Pruning

```
pruned.model = prune.misclass(my.model, best = 11)
plot(pruned.model)
text(pruned.model, pretty = 0)
```



Updated Predictions

```r
pruned.prediction = predict(pruned.model, test_data, type = "class")

confusionMatrix(table(pruned.prediction, test_data$test_labels))
```

```
Confusion Matrix and Statistics


pruned.prediction 1999 2008
             1999   18    8
             2008    6   12

               Accuracy : 0.6818
                 95% CI : (0.5242, 0.8139)
    No Information Rate : 0.5455
    P-Value [Acc > NIR] : 0.04653

                  Kappa : 0.3529
 Mcnemar's Test P-Value : 0.78927

            Sensitivity : 0.7500
            Specificity : 0.6000
         Pos Pred Value : 0.6923
         Neg Pred Value : 0.6667
             Prevalence : 0.5455
         Detection Rate : 0.4091
   Detection Prevalence : 0.5909
      Balanced Accuracy : 0.6750

       'Positive' Class : 1999
```