DATA SCIENCE WITH R

DECISION TREES

Graham.Williams@togaware.com

Data Scientist
Australian Taxation Office

Adjunct Professor, Australian National University
Adjunct Professor, University of Canberra
Fellow, Institute of Analytics Professionals of Australia

Graham.Williams@togaware.com
http://datamining.togaware.com
Toyota Corolla price homework
Roger Bohn, April 2016
Overview

1 Introduction

2 Decision Trees
   - Basics
   - Example
   - Algorithm

3 Building Decision Trees
   - In Rattle
   - In R
Goal of classification is to build models (sentences) in a knowledge representation (language) from examples of past decisions.

The model is to be used on unseen cases to make decisions.

Often referred to as supervised learning.

Common approaches: decision trees; neural networks; logistic regression; support vector machines. 

algorithms
**Language: Decision Trees**

- Knowledge representation: A flow-chart-like tree structure
- Internal nodes denotes a test on a variable
- Branch represents an outcome of the test
- Leaf nodes represent class labels or class distribution

![Decision Tree Diagram]

- Gender
  - Male
  - Female
- Age
  - < 42
  - > 42
- Y
- N
Decision tree induction is an example of a recursive partitioning algorithm: divide and conquer.

At start, all the training examples are at the root

Partition examples recursively based on selected variables
**Training Dataset: Buys Computer?**

What rule would you “learn” to identify who buys a computer?

<table>
<thead>
<tr>
<th>Age</th>
<th>Income</th>
<th>Student</th>
<th>Credit</th>
<th>Buys</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt; 30</td>
<td>High</td>
<td>No</td>
<td>Poor</td>
<td>No</td>
</tr>
<tr>
<td>&lt; 30</td>
<td>High</td>
<td>No</td>
<td>Good</td>
<td>Yes</td>
</tr>
<tr>
<td>30 – 40</td>
<td>High</td>
<td>No</td>
<td>Poor</td>
<td>Yes</td>
</tr>
<tr>
<td>&gt; 40</td>
<td>Medium</td>
<td>No</td>
<td>Poor</td>
<td>Yes</td>
</tr>
<tr>
<td>&gt; 40</td>
<td>Low</td>
<td>Yes</td>
<td>Poor</td>
<td>Yes</td>
</tr>
<tr>
<td>&gt; 40</td>
<td>Low</td>
<td>Yes</td>
<td>Good</td>
<td>No</td>
</tr>
<tr>
<td>30 – 40</td>
<td>Low</td>
<td>Yes</td>
<td>Good</td>
<td>Yes</td>
</tr>
<tr>
<td>&lt; 30</td>
<td>Medium</td>
<td>No</td>
<td>Poor</td>
<td>No</td>
</tr>
<tr>
<td>&lt; 30</td>
<td>Low</td>
<td>Yes</td>
<td>Poor</td>
<td>No</td>
</tr>
<tr>
<td>&gt; 40</td>
<td>Medium</td>
<td>Yes</td>
<td>Poor</td>
<td>Yes</td>
</tr>
<tr>
<td>&lt; 30</td>
<td>Medium</td>
<td>Yes</td>
<td>Good</td>
<td>Yes</td>
</tr>
<tr>
<td>30 – 40</td>
<td>Medium</td>
<td>No</td>
<td>Good</td>
<td>Yes</td>
</tr>
<tr>
<td>30 – 40</td>
<td>High</td>
<td>Yes</td>
<td>Poor</td>
<td>Yes</td>
</tr>
<tr>
<td>&gt; 40</td>
<td>Medium</td>
<td>No</td>
<td>Good</td>
<td>No</td>
</tr>
</tbody>
</table>
Output: Decision Tree for Buys Computer

One possible tree:

```
          Student
         /     \
        Yes    No
       [71/29] [29/71]
```
**Training Dataset: Buys Computer?**

What rule would you “learn” to identify who buys a computer?

<table>
<thead>
<tr>
<th>Age</th>
<th>Income</th>
<th>Student</th>
<th>Credit</th>
<th>Buys</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt; 30</td>
<td>High</td>
<td>No</td>
<td>Poor</td>
<td>No</td>
</tr>
<tr>
<td>&lt; 30</td>
<td>High</td>
<td>No</td>
<td>Good</td>
<td>Yes</td>
</tr>
<tr>
<td>30 – 40</td>
<td>High</td>
<td>No</td>
<td>Poor</td>
<td>Yes</td>
</tr>
<tr>
<td>&gt; 40</td>
<td>Medium</td>
<td>No</td>
<td>Poor</td>
<td>Yes</td>
</tr>
<tr>
<td>&gt; 40</td>
<td>Low</td>
<td>Yes</td>
<td>Poor</td>
<td>Yes</td>
</tr>
<tr>
<td>&gt; 40</td>
<td>Low</td>
<td>Yes</td>
<td>Good</td>
<td>No</td>
</tr>
<tr>
<td>30 – 40</td>
<td>Low</td>
<td>Yes</td>
<td>Good</td>
<td>Yes</td>
</tr>
<tr>
<td>&lt; 30</td>
<td>Medium</td>
<td>No</td>
<td>Poor</td>
<td>No</td>
</tr>
<tr>
<td>&lt; 30</td>
<td>Low</td>
<td>Yes</td>
<td>Poor</td>
<td>No</td>
</tr>
<tr>
<td>&gt; 40</td>
<td>Medium</td>
<td>Yes</td>
<td>Poor</td>
<td>Yes</td>
</tr>
<tr>
<td>&lt; 30</td>
<td>Medium</td>
<td>Yes</td>
<td>Good</td>
<td>Yes</td>
</tr>
<tr>
<td>30 – 40</td>
<td>Medium</td>
<td>No</td>
<td>Good</td>
<td>Yes</td>
</tr>
<tr>
<td>30 – 40</td>
<td>High</td>
<td>Yes</td>
<td>Poor</td>
<td>Yes</td>
</tr>
<tr>
<td>&gt; 40</td>
<td>Medium</td>
<td>No</td>
<td>Good</td>
<td>No</td>
</tr>
</tbody>
</table>
Output: Decision Tree for Buys Computer

One possible tree:

- **Student**
  - Yes
  - **Income**
    - Low: [50/50]
    - H/M: [100/0]
  - No: [29/71]
Output: Decision Tree for Buys Computer

One possible tree:

```
    Student
     /   \  
    Yes   No
     /\    /\ 
  Income  Income
  /\   /\   /\ 
Low  H/M  Low  H/M
  /\   /\   /\   /\ 
< 30 < 30 < 30 < 30
 [0/100] [100/0] [29/71] [67/33]
  /\   /\   /\   /\ 
30-40, >40 30-40, >40 30-40, >40 30-40, >40
```

http://togaware.com
Copyright © 2014, Graham.Williams@togaware.com
Output: Decision Tree for Buys Computer

One possible tree:

- **Student**
  - Yes
  - **Income**
    - Low
      - Age
        - < 30
          - [0/100]
        - 30–40, >40
          - [100/0]
    - H/M
      - [67/33]
  - No
  - **Credit**
    - Poor
      - [0/100]
    - Good
      - [67/33]
Output: Decision Tree for Buys Computer

One possible tree:

```
Decision Tree:

Student
  | Yes
  | No
  |   Income
  |   Low
  |     Age
  |     < 30
  |       No
  |       Yes
  |     30-40, >40
  |       Yes
  |     H/M
  |     Yes
  |   Credit
  |   Poor
  |     No
  |   Good
  |     Yes
```
**Algorithm for Decision Tree Induction**

- A **greedy algorithm**: takes the best immediate (local) decision while building the overall model

- Tree constructed top-down, recursive, divide-and-conquer

- Begin with all training examples at the root

- Data is partitioned recursively based on selected variables

- Select variables on basis of a **measure**

- **Stop** partitioning when:
  - All samples for a given node belong to the same class
  - There are no remaining variables for further partitioning – majority voting is employed for classifying the leaf
  - There are no samples left
Basic Motivation: Entropy

We are trying to predict output $Y$ (e.g., Yes/No) from input $X$.

- A random data set may have high entropy:
  - $Y$ is from a uniform distribution
  - a frequency distribution would be flat!
  - a sample will include uniformly random values of $Y$

- A data set with low entropy:
  - $Y$’s distribution will be very skewed
  - a frequency distribution will have a single peak
  - a sample will predominately contain just Yes or just No

- Work towards reducing the amount of entropy in the data!
Basic Motivation: Entropy

We are trying to predict output $Y$ (e.g., Yes/No) from input $X$.

- A random data set may have high entropy:
  - $Y$ is from a uniform distribution
  - a frequency distribution would be flat!
  - a sample will include uniformly random values of $Y$

- A data set with low entropy:
  - $Y$’s distribution will be very skewed
  - a frequency distribution will have a single peak
  - a sample will predominately contain just Yes or just No

- Work towards reducing the amount of entropy in the data!
We are trying to predict output $Y$ from input $X$.

$X = \text{Course}$  
$Y = \text{Purchase Neo1973}$

<table>
<thead>
<tr>
<th>$X$</th>
<th>$Y$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Math</td>
<td>Yes</td>
</tr>
<tr>
<td>History</td>
<td>No</td>
</tr>
<tr>
<td>CS</td>
<td>Yes</td>
</tr>
<tr>
<td>Math</td>
<td>No</td>
</tr>
<tr>
<td>Math</td>
<td>No</td>
</tr>
<tr>
<td>CS</td>
<td>Yes</td>
</tr>
<tr>
<td>History</td>
<td>No</td>
</tr>
<tr>
<td>Math</td>
<td>Yes</td>
</tr>
</tbody>
</table>

Focus on $Y$

$P(\text{Yes}) = 0.5$

$P(\text{No}) = 0.5$

Uniform distribution of $Y$

Entropy of $Y$ is 1

$$E(p, n) = -\frac{p}{p+n} \log_2 \frac{p}{p+n} - \frac{n}{p+n} \log_2 \frac{n}{p+n}$$

$log_2(0.5) = -1$
Variable Selection Measure: **Gini**

- Gini index of impurity – traditional statistical measure – CART
- Measure how often a randomly chosen observation is incorrectly classified if it were randomly classified in proportion to the actual classes.
- Calculated as the sum of the probability of each observation being chosen times the probability of incorrect classification, equivalently:

\[
I_G(p, n) = 1 - (p^2 + (1 - p)^2)
\]

- As with Entropy, the Gini measure is maximal when the classes are equally distributed and minimal when all observations are in one class or the other.
**Variable Selection Measure: Entropy**

- Information gain (ID3/C4.5)
- Select the variable with the highest information gain
- Assume there are two classes: $P$ and $N$
- Let the data $S$ contain $p$ elements of class $P$ and $n$ elements of class $N$
- The amount of information, needed to decide if an arbitrary example in $S$ belongs to $P$ or $N$ is defined as

$$I_E(p, n) = - \frac{p}{p + n} \log_2 \frac{p}{p + n} - \frac{n}{p + n} \log_2 \frac{n}{p + n}$$
Variable Selection Measure

Variable Importance Measure

Proportion of Positives

Measure

Formula
- Info
- Gini

http://togaware.com

Copyright © 2014, Graham.Williams@togaware.com
OVERVIEW

1 Introduction

2 Decision Trees
   - Basics
   - Example
   - Algorithm

3 Building Decision Trees
   - In Rattle
   - In R
**Summary of the Weather Dataset**

- A summary of the weather dataset is displayed.

```
No. Variable       Data Type  Input Target Risk Ident Ignore Weight Comment
16 Pressure9am    Numeric     ∅   ∅   ∅   ∅   ∅   ∅      Unique: 190
17 Pressure3pm    Numeric     ∅   ∅   ∅   ∅   ∅   ∅      Unique: 193
18 Cloud9am       Numeric     ∅   ∅   ∅   ∅   ∅   ∅      Unique: 9
19 Cloud3pm       Numeric     ∅   ∅   ∅   ∅   ∅   ∅      Unique: 9
20 Temp9am        Numeric     ∅   ∅   ∅   ∅   ∅   ∅      Unique: 178
21 Temp3pm        Numeric     ∅   ∅   ∅   ∅   ∅   ∅      Unique: 200
22 RainToday      Categoric   ∅   ∅   ∅   ∅   ∅   ∅      Unique: 2
23 RISK_MM        Numeric     ∅   ∅   ∅   ∅   ∅   ∅      Unique: 47
24 RainTomorrow   Categoric   ∅   ∅   ∅   ∅   ∅   ∅      Unique: 2
```

Roles noted. 366 observations and 20 input variables. The target is RainTomorrow. Categoric 2. Classification models enabled.
Model Tab — Decision Tree

- Click on the Model tab to display the modelling options.

![Decision Tree Model](image)

A decision tree model is one of the most common data mining models. It is popular because the resulting model is easy to understand. The algorithms use a recursive partitioning approach.

The traditional algorithm is implemented in the rpart package. It is comparable to CART and ID3/C4.

The conditional tree algorithm is implemented in the party package. It builds trees in a conditional inference framework.

Note that the ensemble approaches (boosting and random forests) tend to produce models that exhibit less bias and variance than a single decision tree.
Corolla tree: age, km,

Decision Tree toyotacorolla_data_binary2_1_4.csv $ Price.median

1. Age_08_04 >= 58
   - yes
   - Age_08_04 >= 68
     - yes
     - Weight < 1018
       - yes
       - KM >= 124e+3
         - yes
         - SAND
           .64 .36
           24%
         - no
         - SAND
           .60 .40
           21%
       - no
       - FALS
         .98 .02
         4%
     - no
     - FALS
       .97 .03
       27%
   - no
   - SAND
     .51 .49
     100%

- no
  - SAND
    .60 .40
    21%
Figure 2: Decision rules for the best model

Rattle timestamp: 2016-04-03 19:43:18 Siff

Tree as rules:

Rule number: 3 [Price.median=SAND cover=421 (42%) prob=0.00]
    Mfg_Year>=2000

Rule number: 23 [Price.median=SAND cover=96 (10%) prob=0.00]
    Mfg_Year< 2000
    Mfg_Year>=1998
    Airco>=0.5
    KM< 8.586e+04

Rule number: 22 [Price.median=FALSK cover=49 (5%) prob=0.00]
    Mfg_Year< 2000
    Mfg_Year>=1998
    Airco>=0.5
    KM>=8.586e+04

Rule number: 10 [Price.median=FALSK cover=161 (16%) prob=0.00]
    Mfg_Year< 2000
    Mfg_Year>=1998
    Airco< 0.5

Rule number: 4 [Price.median=FALSK cover=278 (28%) prob=0.00]
    Mfg_Year< 2000
    Mfg_Year< 1998

[1] 9 8 7 6 5 4 3 2 1

Rattle timestamp: 2016-04-03 19:43:20 Siff

==================================================================
How to build in practice

- Start with (almost) every variable
- Let the algorithm decide what belongs in final model!!!
  - Do NOT pre-screen based on theory
  - Do Throw out obvious junk
Evaluating result: Confusion matrix

- Calculate model using only training data
- Evaluate model using only testing (or validation) data.

Figure 3: Confusion matrix evaluation for the Decision Tree

<table>
<thead>
<tr>
<th>Actual</th>
<th>9900</th>
<th>FALSK</th>
<th>SAND</th>
</tr>
</thead>
<tbody>
<tr>
<td>9900</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>FALSK</td>
<td>0</td>
<td>98</td>
<td>24</td>
</tr>
<tr>
<td>SAND</td>
<td>0</td>
<td>8</td>
<td>85</td>
</tr>
</tbody>
</table>

Error matrix for the Decision Tree

<table>
<thead>
<tr>
<th>Actual</th>
<th>9900</th>
<th>FALSK</th>
<th>SAND</th>
<th>Error</th>
</tr>
</thead>
<tbody>
<tr>
<td>9900</td>
<td>0</td>
<td>0.00</td>
<td>0.00</td>
<td>NaN</td>
</tr>
<tr>
<td>FALSK</td>
<td>0</td>
<td>0.46</td>
<td>0.11</td>
<td>0.20</td>
</tr>
<tr>
<td>SAND</td>
<td>0</td>
<td>0.04</td>
<td>0.40</td>
<td>0.09</td>
</tr>
</tbody>
</table>

Overall error: 15%, Averaged class

Rattle timestamp: 2016-04-03 19:27:
Report what you DID

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Age_08_04</td>
<td>Age in months as in August 2004</td>
</tr>
<tr>
<td>Mfg_Year</td>
<td>Manufacturing Year</td>
</tr>
<tr>
<td>KM</td>
<td>Accumulated Kilometers on odometer</td>
</tr>
<tr>
<td>Fuel_Type</td>
<td>Fuel Type (Petrol, Diesel, CNG)</td>
</tr>
<tr>
<td>HP</td>
<td>Horse Power</td>
</tr>
<tr>
<td>Automatic</td>
<td>Automatic (Yes=1, No=0)</td>
</tr>
<tr>
<td>Doors</td>
<td>Number of doors</td>
</tr>
<tr>
<td>Guarantee_Period</td>
<td>Guarantee period in months</td>
</tr>
<tr>
<td>Airbag_1</td>
<td>Driver_Airbag (Yes=1, No=0)</td>
</tr>
<tr>
<td>Airbag_2</td>
<td>Passenger Airbag (Yes=1, No=0)</td>
</tr>
</tbody>
</table>

**Partition:** 70/30/0

**Seed:** 12345

- Don’t omit so many variables unless sure they don’t matter.
Typical results

- Age (in months)
- Km traveled
- Air conditioning
- Weight
Air conditioning

- AC Yes/No
- Automatic AC Yes/no
- Model thinks this is 2 independent variables
- Use outside knowledge:
  - Convert this to 1 variable with 3 levels
  - vels
Decision Tree is the default model type—simply click Execute.
Decision Tree Predicting RainTomorrow

- Click the Draw button to display a tree (Settings → Advanced Graphics).
Evaluate Decision Tree

- Click Evaluate tab—options to evaluate model performance.

Error Matrix

An error matrix shows the true outcomes against the predicted outcomes. Two tables will be presented here. The first will be the count of observations and the second will be the proportions.

For a binary classification model the cells of the error matrix are referred to, from the top left going clockwise, as the True Negatives, False Positives, True Positives, and False Negatives.

An error matrix is also known as a confusion matrix.
Evaluate Decision Tree—Error Matrix

- Click Execute to display simple error matrix.
- Identify the True/False Positives/Negatives.

Confusion matrix
**Decision Tree Risk Chart**

- Click the Risk type and then Execute.
**Decision Tree ROC Curve**

- Click the ROC type and then Execute.
Score a Dataset

- Click the Score type to score a new dataset using model.

A model can be deployed on a dataset to obtain scores or classifications for each observation in the dataset.

By default the testing dataset (if any) will be scored. Otherwise the training dataset is scored. As an alternative, a CSV file can be loaded and scored. This choice of what is scored is controlled by the radio button options.

For binary models a probability score can be recorded. For regression models a value is recorded for each observation. Otherwise a class will be recorded for each observation. This can be controlled by the Class and Probability radio buttons.

The resulting CSV file will include just those variables having a role as Identifier (plus the Target and the Score), or else all of the variables.

The name of a CSV file into which the results will be written will be prompted for.
Log of R Commands

- Click the Log tab for a history of all your interactions.
- Save the log contents as a script to repeat what we did.

```r
# Rattle is Copyright (c) 2006-2013 Togaware Pty Ltd.
#=========================================================
# Rattle timestamp: 2013-07-03 20:09:30 x86_64-pc-linux-gnu
# Rattle version 2.6.27 user 'gjw'
# Export this log textview to a file using the Export button or the Tools
# menu to save a log of all activity. This facilitates repeatability. Exporting
# to file 'myrf01.R', for example, allows us to the type in the R Console
# the command source('myrf01.R') to repeat the process automatically.
# Generally, we may want to edit the file to suit our needs. We can also directly
# edit this current log textview to record additional information before exporting.
# Saving and loading projects also retains this log.
library(rattle)

# This log generally records the process of building a model. However, with very
# little effort the log can be used to score a new dataset. The logical variable
# 'building' is used to toggle between generating transformations, as when building
# a model, and simply using the transformations, as when scoring a dataset.
building <- TRUE
scoring <- ! building

# The colourspace package is used to generate the colours used in plots, if available.
library(colorspace)
```
Log of R Commands—rpart()

- Here we see the call to rpart() to build the model.
- Click on the Export button to save the script to file.

```r
# Rattle timestamp: 2013-07-03 20:25:41 x86_64-pc-linux-gnu
# Decision Tree
# The 'rpart' package provides the 'rpart' function.
require(rpart, quietly=TRUE)
# Reset the random number seed to obtain the same results each time.
set.seed(crs$seed)
# Build the Decision Tree model.
crs$rpart <- rpart(RainTomorrow ~ .,
data=crs$dataset[crs$train, crs$input, crs$target],
method="class",
parms=list(split="information"),
control=rpart.control(use surrogate=0,
max surrogate=0))
# Generate a textual view of the Decision Tree model.
print(crs$rpart)
printcp(crs$rpart)
cat("\n")
# Time taken: 0.09 secs
```
Rattle provides some basic help—click Yes for R help.
Building Decision Trees

In R

Overview

1. Introduction

2. Decision Trees

   - Basics
   - Example
   - Algorithm

3. Building Decision Trees

   - In Rattle
   - In R

   I have left the R material in here, for future reference.
   In week 2 of the course, we need only Rattle. RB
## Weather Dataset - Inputs

```r
ds <- weather
head(ds, 4)

## Date Location MinTemp MaxTemp Rainfall Evaporation Sunshine  
## 1 2007-11-01 Canberra 8.0 24.3 0.0 3.4 6.3  
## 2 2007-11-02 Canberra 14.0 26.9 3.6 4.4 9.7  
## 3 2007-11-03 Canberra 13.7 23.4 3.6 5.8 3.3  
## 4 2007-11-04 Canberra 13.3 15.5 39.8 7.2 9.1  
.
```

```r
summary(ds[\text{c}(3:5, 23)])

## MinTemp MaxTemp Rainfall RISK_MM  
## Min. : -5.30 Min. : 7.6 Min. : 0.00 Min. : 0.00  
## 1st Qu. : 2.30 1st Qu. :15.0 1st Qu. : 0.00 1st Qu. : 0.00  
## Median : 7.45 Median :19.6 Median : 0.00 Median : 0.00  
## Mean : 7.27 Mean :20.6 Mean : 1.43 Mean : 1.43  
.
```
Building Decision Trees

In R

Weather Dataset - Target

target <- "RainTomorrow"
summary(ds[target])

## RainTomorrow
## No :300
## Yes: 66

(form <- formula(paste(target, " ~ ")))

## RainTomorrow ~ 

(vars <- names(ds)[-c(1, 2, 23)])

## [1] "MinTemp"  "MaxTemp"  "Rainfall"  "Evaporation"
## [5] "Sunshine" "WindGustDir" "WindGustSpeed" "WindDir9am"
## [9] "WindDir3pm" "WindSpeed9am" "WindSpeed3pm" "Humidity9am"
## [13] "Humidity3pm" "Pressure9am" "Pressure3pm" "Cloud9am"
## [17] "Cloud3pm"  "Temp9am"  "Temp3pm"  "RainToday"
**Simple Train/Test Paradigm**

```r
set.seed(1421)
train <- c(sample(1:nrow(ds), 0.70*nrow(ds)))  # Training dataset
head(train)

## [1] 288 298 363 107  70 232

length(train)

## [1] 256

test <- setdiff(1:nrow(ds), train)  # Testing dataset
length(test)

## [1] 110
```
Display the Model

model <- rpart(form, ds[train, vars])
model

## n= 256
##
## node), split, n, loss, yval, (yprob)
## * denotes terminal node
##
## 1) root 256 44 No (0.82812 0.17188)
## 2) Humidity3pm< 59.5 214 21 No (0.90187 0.09813)
## 4) WindGustSpeed< 64 204 14 No (0.93137 0.06863)
## 8) Cloud3pm< 6.5 163 5 No (0.96933 0.03067) *
## 9) Cloud3pm>=6.5 41 9 No (0.78049 0.21951)
## 18) Temp3pm< 26.1 34 4 No (0.88235 0.11765) *
## 19) Temp3pm>=26.1 7 2 Yes (0.28571 0.71429) *

- Notice the legend to help interpret the tree.
**Performance on Test Dataset**

- The `predict()` function is used to score new data.

```r
head(predict(model, ds[test,], type="class"))
```

```
## 2 4 6 8 11 12
## No No No No No No
## Levels: No Yes
```

```r
table(predict(model, ds[test,], type="class"), ds[test, target])
```

```
## No Yes
## No 77 14
## Yes 11 8
```
Example DTree Plot using Rattle
An R Scripting Hint

- Notice the use of variables ds, target, vars.
- Change these variables, and the remaining script is unchanged.
- Simplifies script writing and reuse of scripts.

```r
ds <- iris
target <- "Species"
vars <- names(ds)
```

- Then repeat the rest of the script, without change.
**An R Scripting Hint — Unchanged Code**

- This code remains the same to build the decision tree.

```r
form <- formula(paste(target, "~ "))
train <- c(sample(1:nrow(ds), 0.70*nrow(ds)))
test <- setdiff(1:nrow(ds), train)
model <- rpart(form, ds[train, vars])
model

## n= 105
##
## node), split, n, loss, yval, (yprob)
##   * denotes terminal node
##
## 1) root 105 69 setosa (0.34286 0.32381 0.33333)
## 2) Petal.Length< 2.6 36 0 setosa (1.00000 0.00000 0.00000) *
## 3) Petal.Length>=2.6 69 34 virginica (0.00000 0.49275 0.50725)
## 6) Petal.Length< 4.95 35 2 versicolor (0.00000 0.94286 0.05714) *
## 7) Petal.Length>=4.95 34 1 virginica (0.00000 0.02941 0.97059)
```
An R Scripting Hint — Unchanged Code

- Similarly for the predictions.

```r
head(predict(model, ds[test,], type="class"))
```

```
## 3 8 9 10 11 12
## setosa setosa setosa setosa setosa setosa setosa
## Levels: setosa versicolor virginica
```

```r
table(predict(model, ds[test,], type="class"), ds[test, target])
```

```
## setosa versicolor virginica
## setosa 14 0 0
## versicolor 0 15 4
## virginica 0 1 11
```
Modelling Framework

Language: Tree with single variable tests

Measure: Entropy, Gini, . . .

Search: Recursive partitioning
Decision Tree Induction.

Most widely deployed machine learning algorithm.

Simple idea, powerful learner.

Available in R through the rpart package.

Related packages include party, Cubist, C50, RWeka (J48).