

DATA ANALYTICS AND BUSINESS INTELLIGENCE (8696/8697)

DECISION TREES

Graham.Williams@togaware.com

Data Scientist
Australian Taxation Office

Adjunct Professor, University of Canberra

Graham.Williams@togaware.com
<http://datamining.togaware.com>



OVERVIEW

1 INTRODUCTION

2 DECISION TREES

- Basics
- Example
- Algorithm

3 BUILDING DECISION TREES

- In Rattle
- In R



OVERVIEW

1 INTRODUCTION

2 DECISION TREES

- Basics
- Example
- Algorithm

3 BUILDING DECISION TREES

- In Rattle
- In R

OVERVIEW

1 INTRODUCTION

2 DECISION TREES

- Basics
- Example
- Algorithm

3 BUILDING DECISION TREES

- In Rattle
- In R



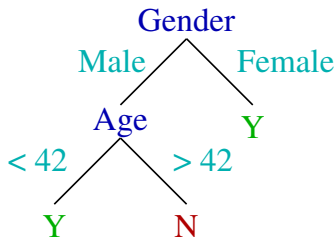
PREDICTIVE MODELLING: CLASSIFICATION

- 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.



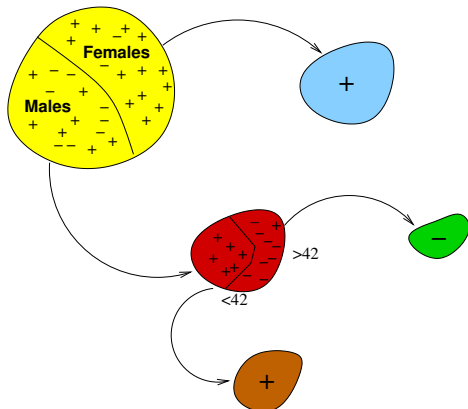
LANGUAGE: DECISION TREES

- 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



TREE CONSTRUCTION: DIVIDE AND CONQUER

- 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?

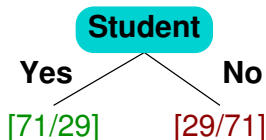
What rule would you “learn” to identify who buys a computer?

Age	Income	Student	Credit	Buys
< 30	High	No	Poor	No
< 30	High	No	Good	Yes
30 – 40	High	No	Poor	Yes
> 40	Medium	No	Poor	Yes
> 40	Low	Yes	Poor	Yes
> 40	Low	Yes	Good	No
30 – 40	Low	Yes	Good	Yes
< 30	Medium	No	Poor	No
< 30	Low	Yes	Poor	No
> 40	Medium	Yes	Poor	Yes
< 30	Medium	Yes	Good	Yes
30 – 40	Medium	No	Good	Yes
30 – 40	High	Yes	Poor	Yes
> 40	Medium	No	Good	No



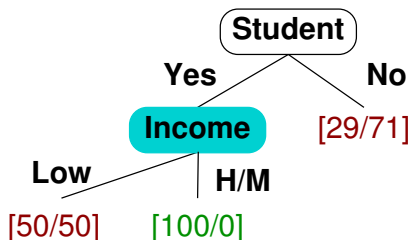
OUTPUT: DECISION TREE FOR BUYS COMPUTER

One possible tree:



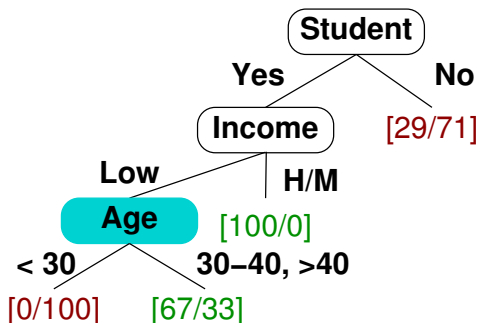
OUTPUT: DECISION TREE FOR BUYS COMPUTER

One possible tree:



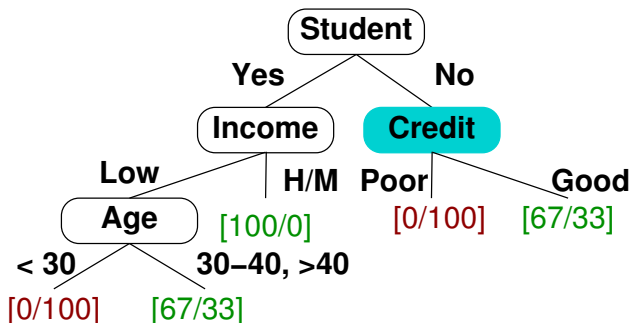
OUTPUT: DECISION TREE FOR BUYS COMPUTER

One possible tree:



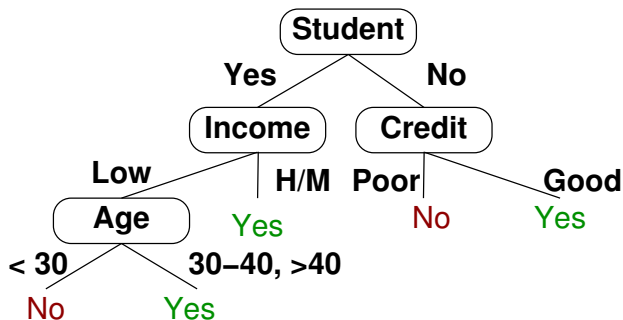
OUTPUT: DECISION TREE FOR BUYS COMPUTER

One possible tree:



OUTPUT: DECISION TREE FOR BUYS COMPUTER

One possible tree:



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!



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!

ENTROPY

We are trying to predict output Y from input X .

X = Course

Y = Purchase Neo1973

Assuming this represents true probabilities:

X	Y
Math	Yes
History	No
CS	Yes
Math	No
Math	No
CS	Yes
History	No
Math	Yes

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

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

$$P(\text{Math} \ \& \ \text{Yes}) = 0.25$$

$$P(\text{History} \ \& \ \text{Yes}) = 0$$

ENTROPY

We are trying to predict output Y from input X .

X = Course

Y = Purchase Neo1973

Assuming this represents true probabilities:

X	Y
Math	Yes
History	No
CS	Yes
Math	No
Math	No
CS	Yes
History	No
Math	Yes

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

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

$$P(\text{Math} \ \& \ \text{Yes}) = 0.25$$

$$P(\text{History} \ \& \ \text{Yes}) = 0$$

ENTROPY

We are trying to predict output Y from input X .

X = Course

Y = Purchase Neo1973

Focus on Y

X	Y
Math	Yes
History	No
CS	Yes
Math	No
Math	No
CS	Yes
History	No
Math	Yes

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

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

Uniform distribution of Y

Entropy of Y is 1

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

$$\log_2(0.5) = -1$$

ENTROPY

We are trying to predict output Y from input X .

X = Course

Focus on Y

Y = Purchase Neo1973

X	Y
Math	Yes
History	No
CS	Yes
Math	No
Math	No
CS	Yes
History	No
Math	Yes

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

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

Uniform distribution of Y

Entropy of Y is 1

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

$$\log_2(0.5) = -1$$

ENTROPY

We are trying to predict output Y from input X .

X = Course

Focus on Y

Y = Purchase Neo1973

X	Y
Math	Yes
History	No
CS	Yes
Math	No
Math	No
CS	Yes
History	No
Math	Yes

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

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

Uniform distribution of Y

Entropy of Y is 1

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

$$\log_2(0.5) = -1$$

ENTROPY

We are trying to predict output Y from input X .

X = Course

Y = Purchase Neo1973

Focus on just students of History

X	Y
Math	Yes
History	No
CS	Yes
Math	No
Math	No
CS	Yes
History	No
Math	Yes

$$P(\text{Yes}) = 0$$

$$P(\text{No}) = 1$$

Skewed distribution of Y

Entropy of Y is 0

$$E(p, n) = -\frac{0}{0+2} \log_2 \frac{0}{0+2} - \frac{2}{0+2} \log_2 \frac{2}{0+2}$$

$$\log_2(0) = -\inf \log_2(1) = 0$$

ENTROPY

We are trying to predict output Y from input X .

X = Course

Y = Purchase Neo1973

Focus on just students of History

X	Y
Math	Yes
History	No
CS	Yes
Math	No
Math	No
CS	Yes
History	No
Math	Yes

$$P(\text{Yes}) = 0$$

$$P(\text{No}) = 1$$

Skewed distribution of Y

Entropy of Y is 0

$$E(p, n) = -\frac{0}{0+2} \log_2 \frac{0}{0+2} - \frac{2}{0+2} \log_2 \frac{2}{0+2}$$

$$\log_2(0) = -\inf \log_2(1) = 0$$

ENTROPY

We are trying to predict output Y from input X .

X = Course

Y = Purchase Neo1973

Focus on just students of History

X	Y
Math	Yes
History	No
CS	Yes
Math	No
Math	No
CS	Yes
History	No
Math	Yes

$$P(\text{Yes}) = 0$$

$$P(\text{No}) = 1$$

Skewed distribution of Y

Entropy of Y is 0

$$E(p, n) = -\frac{0}{0+2} \log_2 \frac{0}{0+2} - \frac{2}{0+2} \log_2 \frac{2}{0+2}$$

$$\log_2(0) = -\text{Inf} \quad \log_2(1) = 0$$

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: GINI

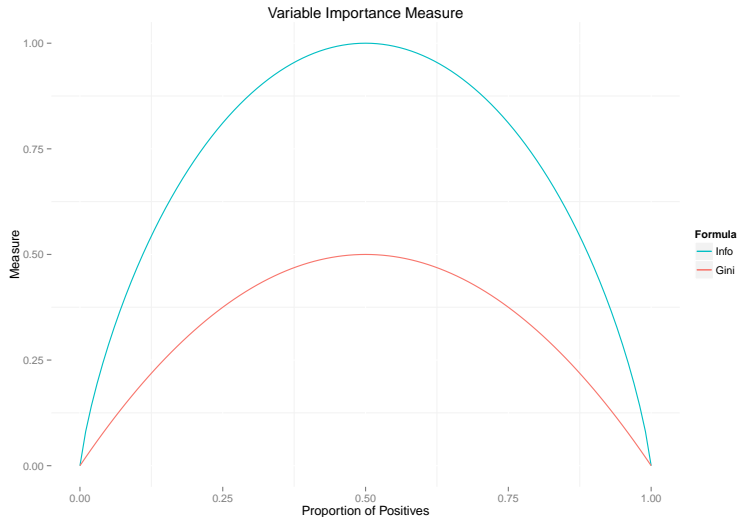
- Gini index of impurity – traditional statistical measure – CART
- Measure how often a randomly chosen observation is incorrectly classified if it were randomly classified in proportion to the actual classes.
- Calculated as the sum of the probability of each observation being chosen times the probability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



INFORMATION GAIN

- Now use variable A to partition S into v cells: $\{S_1, S_2, \dots, S_v\}$
- If S_i contains p_i examples of P and n_i examples of N , the information now needed to classify objects in all subtrees S_i is:

$$E(A) = \sum_{i=1}^v \frac{p_i + n_i}{p + n} I(p_i, n_i)$$

- So, the information gained by branching on A is:

$$\text{Gain}(A) = I(p, n) - E(A)$$

So choose the variable A which results in the greatest gain in information.

OVERVIEW

1 INTRODUCTION

2 DECISION TREES

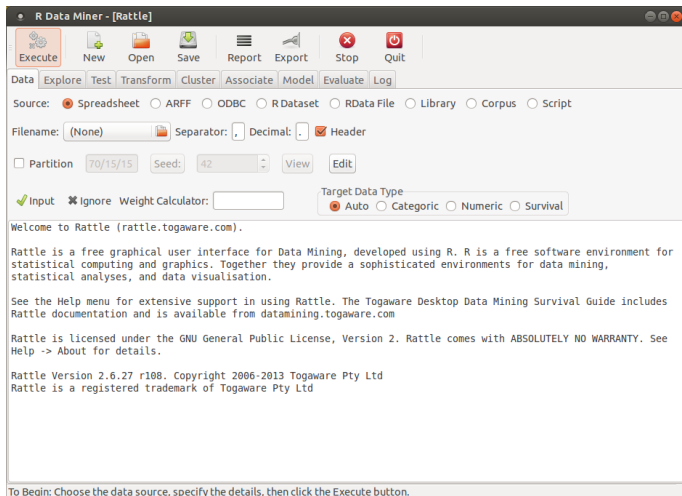
- Basics
- Example
- Algorithm

3 BUILDING DECISION TREES

- In Rattle
- In R

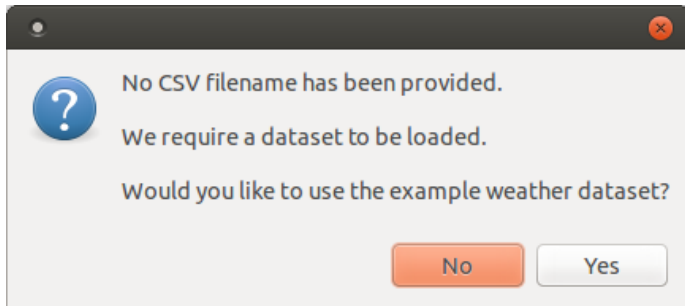
STARTUP RATTLE

```
library(rattle)
rattle()
```



LOAD EXAMPLE WEATHER DATASET

- Click on the Execute button and an example dataset is offered.
- Click on Yes to load the weather dataset.



SUMMARY OF THE WEATHER DATASET

- A summary of the weather dataset is displayed.

R Data Miner - [Rattle (weather.csv)]

Project Tools Settings Help Rattle Version 3.0.2 togaware.com

Execute New Open Save Report Export Stop Quit

Data Explore Test Transform Cluster Associate Model Evaluate Log

Source: ☒ Spreadsheet ☐ ARFF ☐ ODBC ☐ R Dataset ☐ RData File ☐ Library ☐ Corpus ☐ Script

Filename: Separator: Decimal: ☒ Header

☒ Partition 70/15/15 Seed: View Edit

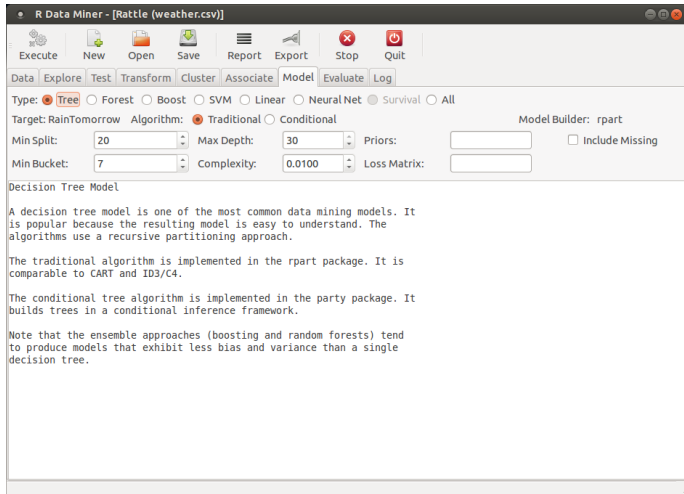
☒ Input ☐ Ignore Weight Calculator: Target Data Type: ☒ Auto ☐ Categorical ☐ Numeric ☐ Survival

No.	Variable	Data Type	Input	Target	Risk	Ident	Ignore	Weight	Comment
16	Pressure9am	Numeric	<input checked="" type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	Unique: 190
17	Pressure3pm	Numeric	<input checked="" type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	Unique: 193
18	Cloud9am	Numeric	<input checked="" type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	Unique: 9
19	Cloud3pm	Numeric	<input checked="" type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	Unique: 9
20	Temp9am	Numeric	<input checked="" type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	Unique: 178
21	Temp3pm	Numeric	<input checked="" type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	Unique: 200
22	RainToday	Categorical	<input checked="" type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	Unique: 2
23	RISK_MM	Numeric	<input type="radio"/>	<input type="radio"/>	<input checked="" type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	Unique: 47
24	RainTomorrow	Categorical	<input type="radio"/>	<input checked="" type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	Unique: 2

Roles noted. 366 observations and 20 input variables. The target is RainTomorrow. Categorical 2. Classification models enabled.

MODEL TAB — DECISION TREE

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



BUILD TREE TO PREDICT RAIN TOMORROW

- Decision Tree is the default model type—simply click Execute.

The screenshot shows the Rattle GUI for building a Decision Tree model. The 'Model' tab is selected, and the 'Tree' model type is chosen. The target variable is 'RainTomorrow'. The algorithm is set to 'Traditional'. The model builder is 'rpart'. The configuration parameters are: Min Split: 20, Max Depth: 30, Priors: (empty), Include Missing: (unchecked), Min Bucket: 7, Complexity: 0.0100, Loss Matrix: (empty), Rules: (button), Draw: (button). The summary of the model is displayed below the configuration fields.

Summary of the Decision Tree model for Classification (built using 'rpart'):

```
n= 256
node), split, n, loss, yval, (yprob)
* denotes terminal node

1) root 256 41 No (0.83984375 0.16015625)
2) Pressure3pm>=1011.9 204 16 No (0.92156863 0.07843137)
4) Cloud3pm< 7.5 195 10 No (0.94871795 0.05128205) *
5) Cloud3pm>=7.5 9 3 Yes (0.33333333 0.66666667) *
3) Pressure3pm< 1011.9 52 25 No (0.51923077 0.48076923)
6) Sunshine>=8.85 25 5 No (0.80000000 0.20000000) *
7) Sunshine< 8.85 27 7 Yes (0.25925926 0.74074074) *
```

Classification tree:

```
rpart(formula = RainTomorrow ~ ., data = crs$dataset[crs$train,
c(crs$input, crs$target)], method = "class", parms = list(split = "information"),
control = rpart.control(usesurrogate = 0, maxsurrogate = 0))
```

Variables actually used in tree construction:

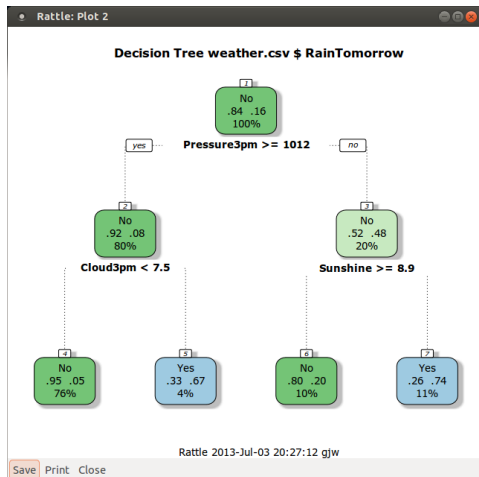
```
[1] Cloud3pm Pressure3pm Sunshine
```

Root node error: 41/256 = 0.16016

The Decision Tree model has been built. Time taken: 0.09 secs

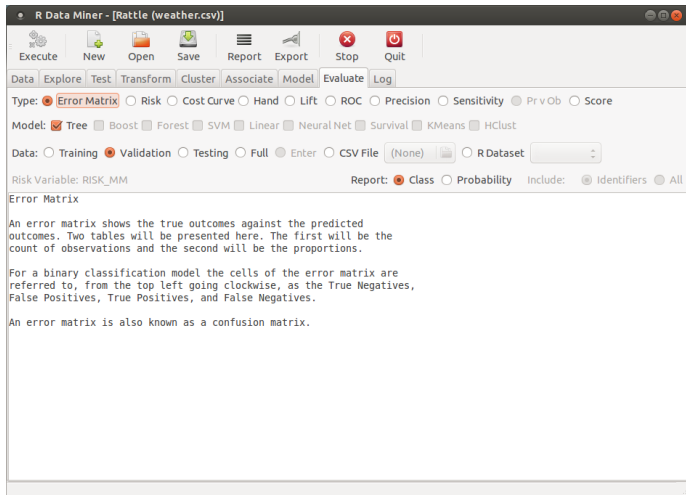
DECISION TREE PREDICTING RAIN TOMORROW

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



EVALUATE DECISION TREE

- Click Evaluate tab—options to evaluate model performance.



EVALUATE DECISION TREE—ERROR MATRIX

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

R Data Miner - [Rattle (weather.csv)]

Execute New Open Save Report Export Stop Quit

Data Explore Test Transform Cluster Associate Model Evaluate Log

Type: ☒ Error Matrix ☐ Risk ☐ Cost Curve ☐ Hand ☐ Lift ☐ ROC ☐ Precision ☐ Sensitivity ☐ Pr v Ob ☐ Score

Model: ☒ Tree ☐ Boost ☐ Forest ☐ SVM ☐ Linear ☐ Neural Net ☐ Survival ☐ KMeans ☐ HClust

Data: ☐ Training ☒ Validation ☐ Testing ☐ Full ☐ Enter ☐ CSV File (None) ☐ R Dataset

Risk Variable: RISK_MM Report: ☒ Class ☐ Probability Include: ☐ Identifiers ☐ All

Error matrix for the Decision Tree model on weather.csv [validate] (counts):

		Predicted	
		No	Yes
Actual	No	39	5
	Yes	5	5

Error matrix for the Decision Tree model on weather.csv [validate] (%):

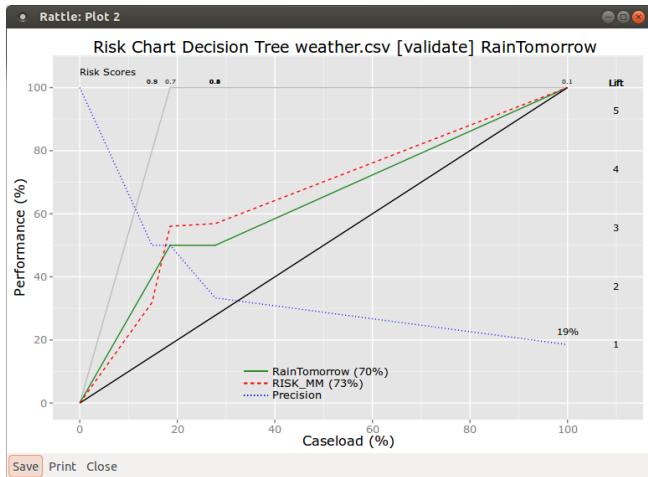
		Predicted	
		No	Yes
Actual	No	72	9
	Yes	9	9

Overall error: 0.1851852

Rattle timestamp: 2013-07-03 20:37:37 gjw

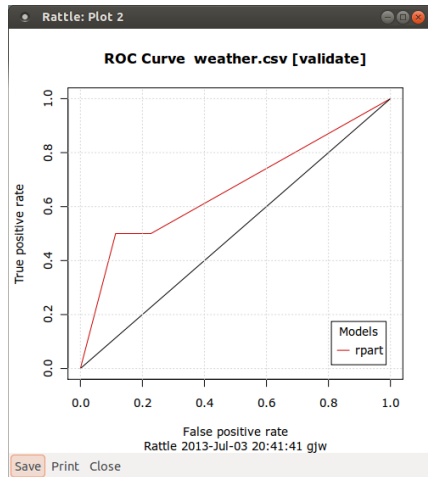
DECISION TREE RISK CHART

- Click the Risk type and then Execute.



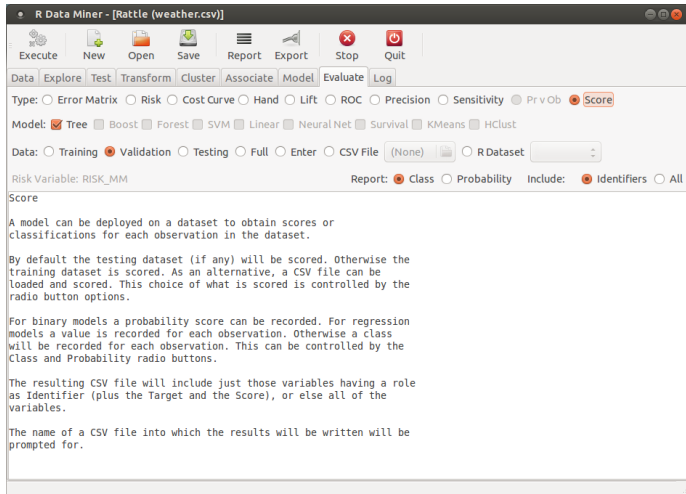
DECISION TREE ROC CURVE

- Click the ROC type and then Execute.



SCORE A DATASET

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



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.

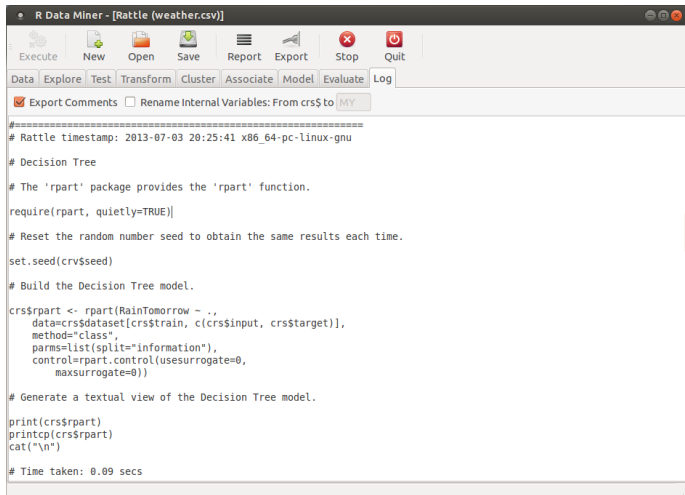
```

R Data Miner - [Rattle (weather.csv)]
Execute New Open Save Report Export Stop Quit
Data Explore Test Transform Cluster Associate Model Evaluate Log
☒ Export Comments ☐ Rename Internal Variables: From crs$ to MY
# 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 'gjlw'
#
# 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 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 colorspace 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 Data Miner - [Rattle (weather.csv)]
Execute New Open Save Report Export Stop Quit
Data Explore Test Transform Cluster Associate Model Evaluate Log
☒ Export Comments ☐ Rename Internal Variables: From crs$t to MY

=====
# 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(crv$seed)

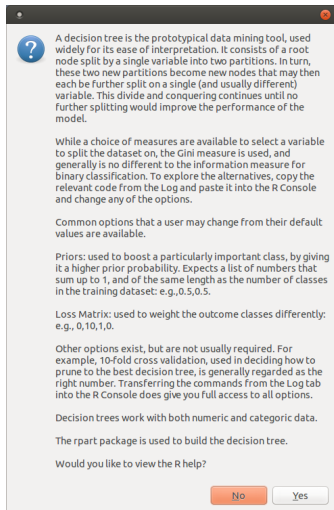
# Build the Decision Tree model.
crs$rpart <- rpart(RainTomorrow ~ .,
  data=crs$dataset[crs$train, c(crs$input, crs$target)],
  method="class",
  parms=list(split="information"),
  control=rpart.control(usesurrogate=0,
    maxsurrogate=0))

# Generate a textual view of the Decision Tree model.
print(crs$rpart)
printcp(crs$rpart)
cat("\n")

# Time taken: 0.09 secs
  
```

HELP → MODEL → TREE

- Rattle provides some basic help—click Yes for R help.



OVERVIEW

1 INTRODUCTION

2 DECISION TREES

- Basics
- Example
- Algorithm

3 BUILDING DECISION TREES

- In Rattle
- In R

WEATHER DATASET - INPUTS

```
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
....
```

```
summary(ds[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
....
```

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"
## [21] "RainTomorrow"
```



SIMPLE TRAIN/TEST PARADIGM

```
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.

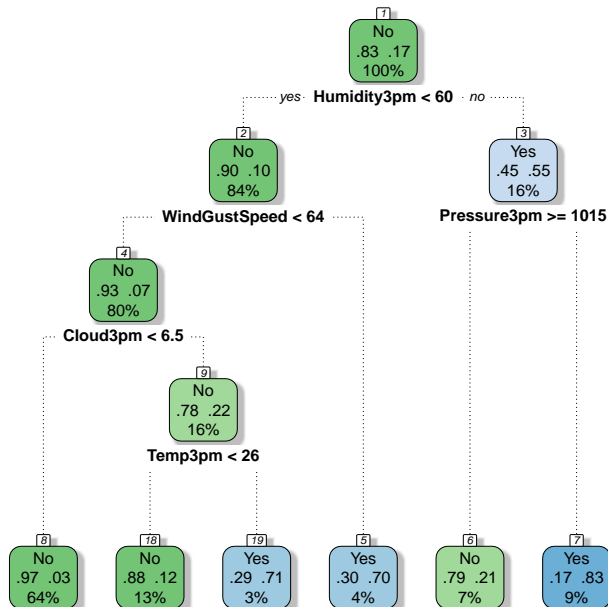
```
head(predict(model, ds[test,], type="class"))
```

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

```
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.

```
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.

```

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.

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

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

```
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

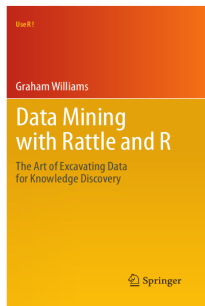
Measure Entropy, Gini, . . .

Search Recursive partitioning

SUMMARY

- 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).

REFERENCE BOOK



Data Mining with Rattle and R

Graham Williams

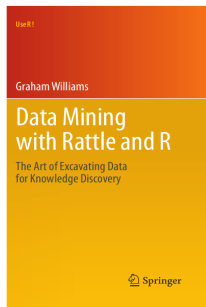
2011, Springer, Use R!

ISBN: 978-1-4419-9889-7.

Chapter 11.



REFERENCE BOOK



Data Mining with Rattle and R

Graham Williams

2011, Springer, Use R!

ISBN: 978-1-4419-9889-7.

Chapter 11.

