Data Science Using Open Source Tools
Decision Trees and Random Forest Using R

Jennifer Evans
Clickfox

jennifer.evans@clickfox.com

January 14, 2014
Text Questions to Twitter Account

JenniferE_CF
Example Code in R

All the R Code is Hosted –includes additional code examples–

www.clickfox.com/ds_rcode
Overview

1. Data Science a Brief Overview
2. Data Science at Clickfox
3. Data Preparation
4. Algorithms
   - Decision Trees
   - Knowing your Algorithm
   - Example Code in R
5. Evaluation
   - Evaluating the Model
   - Evaluating the Business Questions
6. Kaggle and Random Forest
7. Visualization
8. Recommended Reading
Data Science a Brief Overview
What is Data Science?

The meticulous process of iterative testing, proving, revising, retesting, resolving, redoing, programming (because you got smart here and thought automate), debugging, recoding, debugging, tracing, more debugging, documenting (maybe should have started here...) analyzing results, some tweaking, some researching, some hacking, and start over.
Data Science at Clickfox
Software Development

Actively engaged in development of product capabilities in ClickFox Experience Analytics Platform (CEA).

Client Specific Analytics

Engagements in client specific projects.

Force Multipliers

Focus on enabling everyone to be more effective at using data to make decisions.
Will it Rain Tomorrow?
Data Preparation
Receive the Data

Raw Data
Begin Creating Analytic Data Set
Data Munging

Data Munging and Meta Data Creation
Data Preparation

Checking that Data Quality has been Preserved
Bad Data

Types of bad data

- missing, unknown, does not exist
- inaccurate, invalid, inconsistent - false records, or wrong information
- corrupt, wrong character encoding
- poor interpretation, often because lack of context.
- polluted - too much data and overlook what is important
Bad Data

A lot can go wrong in the data collection process, the data storage process, and the data analysis process.

- Nephew and the movie survey
- Protection troops and flooded with information, overlooked that the group gathering nearby was women and children aka. Civilians.
- Manufacturing with acceptable variance, but every so often the measurement machine was bumped, causing miss measurements
- Chemists were meticulous about data collection, but inconsistent with data storage. Used flat files and spreadsheets. They did not have a central data center. The data base grew over time. e.g. Threshold limits listed as zero and less than some threshold number.
Bad Data

Parrot helping you write code...
Not to mention all the things that we can do to really screw things up.
“The combination of some data and an aching desire for an answer does not ensure that a reasonable answer can be extracted from a given body of data”

~John Tukey
Final Analytic Data Set

<table>
<thead>
<tr>
<th>Courtier</th>
<th>Lettee</th>
<th>Cordero</th>
<th>Janua</th>
<th>Lennione</th>
<th>Dinnione</th>
</tr>
</thead>
<tbody>
<tr>
<td>June</td>
<td>12</td>
<td>2</td>
<td>3</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>18</td>
<td>3</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td></td>
<td>28</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td></td>
<td>40</td>
<td>5</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>50</td>
<td>4</td>
<td>5</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>35</td>
<td>9</td>
<td>2</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>40</td>
<td>10</td>
<td>1</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>50</td>
<td>11</td>
<td>2</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>55</td>
<td>11</td>
<td>2</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>60</td>
<td>12</td>
<td>2</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>
Example (Variables)

```r
> names(ds)
[1] "Date"        "Location"     "MinTemp"     "MaxTemp"
[5] "Rainfall"    "Evaporation"  "Sunshine"    "WindGustDir"
[9] "WindGustSpeed" "WindDir9am"  "WindDir3pm"  "WindSpeed9am"
[13] "WindSpeed3pm" "Humidity9am"  "Humidity3pm" "Pressure9am"
[17] "Pressure3pm" "Cloud9am"     "Cloud3pm"    "Temp9am"
[21] "Temp3pm"    "RainToday"     "RISK_MM"     "RainTomorrow"
```
## Example (First Four Rows of Data)

| Date       | Location | MinTemp | MaxTemp | Rainfall | Evaporation | Sunshine | WindGustDir | WindGustSpeed | WindDir9am | WindDir3pm | WindSpeed9am | WindSpeed3pm | Humidity9am | Humidity3pm | Pressure9am | Pressure3pm | Cloud9am | Cloud3pm | Temp9am | Temp3pm | RainToday | RISK_MM | RainTomorrow |
|------------|----------|---------|---------|----------|-------------|----------|-------------|---------------|-------------|------------|--------------|--------------|--------------|-------------|-------------|-------------|------------|----------|----------|--------|--------|----------|--------|--------------|
| 2007-11-01 | Canberra | 8.0     | 24.3    | 0.0      | 3.4         | 6.3      | NW          | 30            | SW          | NW         | 6            | 20           | 68           | 29          | 1019.7      | 1015.0      | 7          | 7        | 14.4     | 23.6    | No        | 3.6     | Yes         |
| 2007-11-02 | Canberra | 14.0    | 26.9    | 3.6      | 4.4         | 9.7      | ENE         | 39            | E           | W          | 4            | 17           | 80           | 36          | 1012.4      | 1008.4      | 5          | 3        | 17.5     | 25.7    | Yes       | 3.6     | Yes         |
| 2007-11-03 | Canberra | 13.7    | 23.4    | 3.6      | 5.8         | 3.3      | NW          | 85            | N           | NNE        | 6            | 6            | 82           | 69          | 1009.5      | 1007.2      | 8          | 7        | 15.4     | 20.2    | Yes       | 39.8    | Yes         |
| 2007-11-04 | Canberra | 13.3    | 15.5    | 39.8     | 7.2         | 9.1      | NW          | 54            | WNW         | W          | 30           | 24           | 62           | 56          | 1005.5      | 1007.0      | 2          | 7        | 13.5     | 14.1    | Yes       | 2.8     | Yes         |
Data Checking

Make sure that the values make sense in the context of the field.

- Dates are in the date field.
- A measurement field has numerical values
- Counts of occurrences should be zero or greater.
<table>
<thead>
<tr>
<th>Date</th>
<th>Location</th>
<th>MinTemp</th>
<th>MaxTemp</th>
<th>Rainfall</th>
<th>Evaporation</th>
<th>Sunshine</th>
<th>WindGustDir</th>
<th>WindGustSpeed</th>
<th>WindDir9am</th>
<th>WindDir3pm</th>
<th>WindSpeed9am</th>
<th>WindSpeed3pm</th>
<th>Humidity9am</th>
<th>Humidity3pm</th>
<th>Pressure9am</th>
<th>Pressure3pm</th>
<th>Cloud9am</th>
<th>Cloud3pm</th>
<th>Temp9am</th>
<th>Temp3pm</th>
<th>RainToday</th>
<th>RISK_MM</th>
<th>RainTomorrow</th>
</tr>
</thead>
<tbody>
<tr>
<td>2007-11-01</td>
<td>Canberra</td>
<td>8.0</td>
<td>24.3</td>
<td>0.0</td>
<td>3.4</td>
<td>6.3</td>
<td>NW</td>
<td>30</td>
<td>SW</td>
<td>NW</td>
<td>6</td>
<td>20</td>
<td>68</td>
<td>29</td>
<td>1019.7</td>
<td>1015.0</td>
<td>7</td>
<td>7</td>
<td>14.4</td>
<td>23.6</td>
<td>No</td>
<td>3.6</td>
<td>Yes</td>
</tr>
<tr>
<td>2007-11-02</td>
<td>Canberra</td>
<td>14.0</td>
<td>26.9</td>
<td>3.6</td>
<td>4.4</td>
<td>9.7</td>
<td>ENE</td>
<td>39</td>
<td>E</td>
<td>W</td>
<td>4</td>
<td>17</td>
<td>80</td>
<td>36</td>
<td>1012.4</td>
<td>1008.4</td>
<td>5</td>
<td>3</td>
<td>17.5</td>
<td>25.7</td>
<td>Yes</td>
<td>3.6</td>
<td>Yes</td>
</tr>
<tr>
<td>2007-11-03</td>
<td>Canberra</td>
<td>13.7</td>
<td>23.4</td>
<td>3.6</td>
<td>5.8</td>
<td>3.3</td>
<td>NW</td>
<td>85</td>
<td>N</td>
<td>NNE</td>
<td>6</td>
<td>6</td>
<td>82</td>
<td>69</td>
<td>1009.5</td>
<td>1007.2</td>
<td>8</td>
<td>7</td>
<td>15.4</td>
<td>20.2</td>
<td>Yes</td>
<td>39.8</td>
<td>Yes</td>
</tr>
<tr>
<td>2007-11-04</td>
<td>Canberra</td>
<td>13.3</td>
<td>15.5</td>
<td>39.8</td>
<td>7.2</td>
<td>9.1</td>
<td>NW</td>
<td>54</td>
<td>WNW</td>
<td>W</td>
<td>30</td>
<td>24</td>
<td>62</td>
<td>50</td>
<td>SSE</td>
<td>ESE</td>
<td>20</td>
<td>28</td>
<td>68</td>
<td>68</td>
<td>Yes</td>
<td>2.8</td>
<td>Yes</td>
</tr>
<tr>
<td>2007-11-05</td>
<td>Canberra</td>
<td>7.6</td>
<td>16.1</td>
<td>2.8</td>
<td>5.6</td>
<td>10.6</td>
<td>SSE</td>
<td>44</td>
<td>SE</td>
<td>E</td>
<td>20</td>
<td>24</td>
<td>70</td>
<td>49</td>
<td>1018.3</td>
<td>1018.5</td>
<td>7</td>
<td>7</td>
<td>11.1</td>
<td>15.4</td>
<td>No</td>
<td>0.0</td>
<td>No</td>
</tr>
<tr>
<td>2007-11-06</td>
<td>Canberra</td>
<td>6.2</td>
<td>16.9</td>
<td>0.0</td>
<td>5.8</td>
<td>8.2</td>
<td>SE</td>
<td>57</td>
<td>SE</td>
<td>E</td>
<td>20</td>
<td>24</td>
<td>70</td>
<td>57</td>
<td>1023.8</td>
<td>1021.7</td>
<td>7</td>
<td>5</td>
<td>10.9</td>
<td>14.8</td>
<td>No</td>
<td>0.2</td>
<td>No</td>
</tr>
</tbody>
</table>
There are numeric and categoric variables.
Check the max/min do they make sense? What are the ranges? Do the numerical values need to be normalized?
<table>
<thead>
<tr>
<th>Date</th>
<th>Location</th>
<th>MinTemp</th>
<th>MaxTemp</th>
</tr>
</thead>
<tbody>
<tr>
<td>Min. :2007-11-01</td>
<td>Canberra :366</td>
<td>Min. : -5.300</td>
<td>Min. : 7.60</td>
</tr>
<tr>
<td>1st Qu.:2008-01-31</td>
<td>Adelaide : 0</td>
<td>1st Qu.: 2.300</td>
<td>1st Qu.:15.03</td>
</tr>
<tr>
<td>Median :2008-05-01</td>
<td>Albany : 0</td>
<td>Median : 7.450</td>
<td>Median : 19.65</td>
</tr>
<tr>
<td>Mean :2008-05-01</td>
<td>Albury : 0</td>
<td>Mean : 7.266</td>
<td>Mean : 20.55</td>
</tr>
<tr>
<td>3rd Qu.:2008-07-31</td>
<td>Alice Springs : 0</td>
<td>3rd Qu.:12.500</td>
<td>3rd Qu.:25.50</td>
</tr>
<tr>
<td>Max. :2008-10-31</td>
<td>BadgerysCreek : 0</td>
<td>Max. : 20.900</td>
<td>Max. : 35.80</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Date</th>
<th>Location</th>
<th>Rainfall</th>
<th>Evaporation</th>
<th>Sunshine</th>
<th>WindGustDir</th>
</tr>
</thead>
<tbody>
<tr>
<td>Min. : 0.000</td>
<td>Min. : 0.200</td>
<td>Min. : 0.000</td>
<td>NW : 73</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1st Qu. : 0.000</td>
<td>1st Qu. : 2.200</td>
<td>1st Qu. : 5.950</td>
<td>NNW : 44</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Median : 0.000</td>
<td>Median : 4.200</td>
<td>Median : 8.600</td>
<td>E : 37</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Mean : 1.428</td>
<td>Mean : 4.522</td>
<td>Mean : 7.909</td>
<td>WNW : 35</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3rd Qu. : 0.200</td>
<td>3rd Qu. : 6.400</td>
<td>3rd Qu. : 10.500</td>
<td>ENE : 30</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Max. : 39.800</td>
<td>Max. : 13.800</td>
<td>Max. : 13.600</td>
<td>(Other): 144</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Date</th>
<th>Location</th>
<th>WindGustSpeed</th>
<th>WindDir9am</th>
<th>WindDir3pm</th>
<th>WindSpeed9am</th>
<th>WindSpeed3pm</th>
</tr>
</thead>
<tbody>
<tr>
<td>Min. : 13.00</td>
<td>Min. : 47</td>
<td>WNW : 61</td>
<td>Min. : 0.000</td>
<td>Min. : 0.00</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1st Qu. : 31.00</td>
<td>SSE : 40</td>
<td>NW : 61</td>
<td>1st Qu. : 6.000</td>
<td>1st Qu. : 11.00</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Median : 39.00</td>
<td>NNW : 36</td>
<td>NNW : 47</td>
<td>Median : 7.000</td>
<td>Median : 17.00</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3rd Qu. : 46.00</td>
<td>NW : 30</td>
<td>ESE : 27</td>
<td>3rd Qu. : 13.000</td>
<td>3rd Qu. : 24.00</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Max. : 98.00</td>
<td>(Other): 151</td>
<td>(Other): 139</td>
<td>Max. : 41.000</td>
<td>Max. : 52.00</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Date</th>
<th>Location</th>
<th>Humidity9am</th>
<th>Humidity3pm</th>
<th>Pressure9am</th>
<th>Pressure3pm</th>
</tr>
</thead>
<tbody>
<tr>
<td>Min. : 36.00</td>
<td>Min. : 13.00</td>
<td>Min. : 996.5</td>
<td>Min. : 996.8</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1st Qu. : 64.00</td>
<td>1st Qu. : 32.25</td>
<td>1st Qu. : 1015.4</td>
<td>1st Qu. : 1012.8</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Median : 72.00</td>
<td>Median : 43.00</td>
<td>Median : 1020.1</td>
<td>Median : 1017.4</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Mean : 72.04</td>
<td>Mean : 44.52</td>
<td>Mean : 1019.7</td>
<td>Mean : 1016.8</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3rd Qu. : 81.00</td>
<td>3rd Qu. : 55.00</td>
<td>3rd Qu. : 1024.5</td>
<td>3rd Qu. : 1021.5</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Max. : 99.00</td>
<td>Max. : 96.00</td>
<td>Max. : 1035.7</td>
<td>Max. : 1033.2</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Plot variables against one another.
Example (Scatterplot)

```r
pairs(~MinTemp+MaxTemp+Rainfall+Evaporation, data = ds,
     main="Simple Scatterplot Matrix")
```
Simple Scatterplot Matrix

MinTemp

MaxTemp

Rainfall

Evaporation
Simple Scatterplot Matrix
Create a histogram of numerical values in a data field, or kernel density estimate.
Example (Histogram)

```r
histogram(ds$MinTemp, breaks=20, col="blue")
```
MinTemp

[Chart showing MinTemp distribution with bins from -5 to 20 on the x-axis and percent of total on the y-axis, with peaks around 0 and 5.]
Example (Kernel Density Plot)

```r
plot(density(ds$MinTemp))
```
density.default(x = ds$MinTemp)

N = 366   Bandwidth = 1.666

Density

Jennifer Evans (Clickfox)
Twitter: JenniferE_CF
January 14, 2014 37 / 164
Kernel Density Plot for all Numerical Variables
density.default(x = ds$MinTemp)

N = 366   Bandwidth = 1.666

Density
density.default(x = ds$MaxTemp)

N = 366   Bandwidth = 1.849
density.default(x = ds$Rainfall)

N = 366   Bandwidth = 0.04125
density.default(x = ds$Evaporation)

N = 366   Bandwidth = 0.7378
density.default(x = ds.complete$Sunshine)
density.default(x = ds.complete$WindSpeed9am)

N = 328   Bandwidth = 1.476
density.default(x = ds$WindSpeed3pm)

N = 366   Bandwidth = 2.448
density.default(x = ds$Humidity9am)

N = 366   Bandwidth = 3.507

Density

Jennifer Evans (Clickfox)
Twitter: JenniferE_CF
January 14, 2014 46 / 164
density.default(x = ds$MinTemp)

Density

N = 366   Bandwidth = 1.666

N = 366   Bandwidth = 1.666
```r
density.default(x = ds$Humidity3pm)
```

N = 366   Bandwidth = 4.658
density.default(x = ds$Pressure9am)

N = 366   Bandwidth = 1.848
```r
density.default(x = ds$Pressure3pm)
N = 366   Bandwidth = 1.788
```
density.default(x = ds$Cloud9am)
density.default(x = ds$Cloud3pm)

N = 366   Bandwidth = 0.737
density.default(x = ds$Temp9am)

N = 366   Bandwidth = 1.556
density.default(x = ds$Temp3pm)
There are missing values in 'Sunshine' and 'Wind-Speed9am'.
Missing and Incomplete

A common pitfall is to assume that you are working with data that is correct and complete. Usually a round of simple checks will reveal any problems; such as counting records, aggregating totals, plotting and comparing to known quantities.
Spillover of time-bound data

Check for duplicates - do not expect that data is perfectly partitioned.
“All models are wrong, some are useful.”

~George Box
Difference between Decision Trees and Random Forest
Willow is a decision tree.
Willow does not generalize well, so you want to ask a few more friends.
Random Friend

Rainbow Dash
Random Friend

Cartman

YOU WILL RESPECT MY AUTHORITY!
Random Friend

Stay Puff Marshmallow
Random Friend

Professor Cat

PROFESSOR CAT
FOUND EQUATION
\[ a+b+c+d+g/q = \text{you give me da cheezburger} \]

JENNIFERCHEEZBURGER.COM ©&®
Your friends are an ensemble of decision trees. But you don't want them all having the same information and giving the same answer.
Good and Bad Predictions

- Willow thinks you like vampire movies more than you do
- Stay Puff thinks you like candy
- Rainbowdash thinks you can fly
- Cartman thinks you just hate everything
- Professor Cat wants a cheeseburger
Thus, your friends now form a bagged (bootstrap aggregated) forest of your movie preferences.
There is still one problem with your data. You don’t want all your friends asking the same questions and basing their decisions on whether a movies is scary or not. So when each friend asks a question, only a random subset of the possible questions is allowed. About the square root of all variables.
Conclusion

Random forest is just an ensemble of decision trees. Really bad, over-fit beasts. A whole lot of trees that really have no idea about what is going on, but we let them vote anyways. Their votes all cancel each other out.
Random Forest Voting

Theorem (Bad Predictors Cancel Out)

\[ \text{Willow} + \text{Cartman} + \text{StayPuff} + \text{ProfCat} + \text{Rainbowdash} = \text{AccuratePrediction} \]
Boosting and Bagging Technique

Bagging decision trees, an early ensemble method, builds multiple decision trees by repeatedly resampling training data with replacement, and voting the trees for a consensus prediction.
Decision Trees
There are a lot of tree algorithm choices in R.
Trees in R

- rpart (CART)
- tree (CART)
- ctree (conditional inference tree)
- CHAID (chi-squared automatic interaction detection)
- evtree (evolutionary algorithm)
- mvpart (multivariate CART)
- knnTree (nearest-neighbor-based trees)
- RWeka (J4.8, M50, LMT)
- LogicReg (Logic Regression)
- BayesTree
- TWIX (with extra splits)
- party (conditional inference trees, model-based trees)
There are a lot of forest algorithm choices in R.
- randomForest (CART-based random forests)
- randomSurvivalForest (for censored responses)
- party (conditional random forests)
- gbm (tree-based gradient boosting)
- mboost (model-based and tree-based gradient boosting)
There are a lot of other ensemble methods and useful packages in R.
Other Useful R Packages

- library(rattle) #Fancy tree plot, nice graphical interface
- library(rpart.plot) #Enhanced tree plots
- library(RColorBrewer) #Color selection for fancy tree plot
- library(party) #Alternative decision tree algorithm
- library(partykit) #Convert rpart object to BinaryTree
- library(doParallel)
- library(caret)
- library(ROCR)
- library(Metrics)
- library(GA) #genetic algorithm, this is the most popular EA
# Example (Useful Commands)

```r
# summary functions
dim(ds)
head(ds)
tail(ds)
summary(ds)
str(ds)

# list functions in package party
ls(package:party)

# save plots as pdf
pdf("plot.pdf")
fancyRpartPlot(model)
dev.off()
```
Knowing your Algorithm
Choose the best split from among the candidate set. Rank order each splitting rule on the basis of some quality-of-split criterion ‘purity’ function. The most frequently used ones are:

- Entropy reduction (nominal / binary targets)
- Gini-index (nominal / binary targets)
- Chi-square tests (nominal / binary targets)
- F-test (interval targets)
- Variance reduction (interval targets)
Locally-Optimal Trees

Commonly use a greedy heuristic, where split rules are selected in a forward stepwise search. The split rule at each internal node is selected to maximize the homogeneity of only its child nodes.
Example Code in R
Example Code in R

Example (R Packages Used for Example Code)

1. `library(rpart)` # Popular decision tree algorithm
2. `library(rattle)` # Fancy tree plot, nice graphical interface
3. `library(rpart.plot)` # Enhanced tree plots
4. `library(RColorBrewer)` # Color selection for fancy tree plot
5. `library(party)` # Alternative decision tree algorithm
6. `library(partykit)` # Convert rpart object to BinaryTree
7. `library(RWeka)` # Weka decision tree J48
8. `library(evtree)` # Evolutionary Algorithm, builds the tree from the bottom up
9. `library(randomForest)`
10. `library(doParallel)`
11. `library(CHAID)` # Chi-squared automatic interaction detection tree
12. `library(tree)`
13. `library(caret)`
```r
data(weather)
dsname <- "weather"
target <- "RainTomorrow"
risk <- "RISK_MMM"
ds <- get(dsname)
vars <- colnames(ds)
(ignore <- vars[c(1, 2, if (exists("risk")) which(risk==vars))])
  names(ds)[1] = 'Date'
  names(ds)[2] = 'Location'
```
Example (Data Prep)

vars <- setdiff(vars, ignore)
(inputs <- setdiff(vars, target))
(nobs <- nrow(ds))
dim(ds[vars])

(form <- formula(paste(target, "~ .")))
set.seed(1426)
length(train <- sample(nobs, 0.7*nobs))
length(test <- setdiff(seq_len(nobs), train))
It is okay to split the data set like this if the outcome of interest is not rare. If the outcome of interest occurs in some small fraction of cases, use a different technique so that 30% or so of cases with the outcome are in the training set.
Example (rpart Tree)

```r
model <- rpart(formula=form, data=ds[train, vars])
```
The default parameter for predict is `na.action = na.pass`. If there are Na’s in the data set, `rpart` will use surrogate splits.
Example (rpart Tree Object)

```r
print(model)
summary(model)
```
print(model)

n= 256

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

1) root 256 38 No (0.85156250 0.14843750)
   2) Humidity3pm< 71 238 25 No (0.89495798 0.10504202)
      4) Pressure3pm >=1010.25 208 13 No (0.93750000 0.06250000) *
      5) Pressure3pm < 1010.25 30 12 No (0.60000000 0.40000000)
         10) Sunshine >=9.95 14 1 No (0.92857143 0.07142857) *
         11) Sunshine < 9.95 16 5 Yes (0.31250000 0.68750000) *
   3) Humidity3pm >=71 18 5 Yes (0.27777778 0.72222222) *
summary(model)

Call:
  rpart(formula = form, data = ds[train, vars])
  n= 256

            CP nsplit rel error xerror xstd
1  0.21052632       0 1.0000000 1.000000 0.1496982
2  0.07894737       1 0.7894737 1.052632 0.1528809
3  0.01000000       3 0.6315789 1.052632 0.1528809

Variable importance
      Humidity3pm Sunshine Pressure3pm Temp9am Pressure9am Temp3pm
        25         17         14          9          8          8
      Cloud3pm MaxTemp MinTemp
        7          6          5

Node number 1: 256 observations, complexity param=0.2105263
  predicted class=No  expected loss=0.1484375  P(node) =1
  class counts:     218      38
  probabilities: 0.852 0.148
  left son=2 (238 obs) right son=3 (18 obs)
  Primary splits:
    Humidity3pm < 71 to the left, improve=12.748630, (0 missing)
    Pressure3pm < 1010.65 to the right, improve=11.244900, (0 missing)
    Cloud3pm < 6.5 to the left, improve=11.006840, (0 missing)
    Sunshine < 6.45 to the right, improve= 9.975051, (2 missing)
    Pressure9am < 1018.45 to the right, improve= 8.380711, (0 missing)
  Surrogate splits:
    Sunshine < 0.75 to the right, agree=0.949, adj=0.278, (0 split)
    Pressure3pm < 1001.55 to the right, agree=0.938, adj=0.111, (0 split)
    Temp3pm < 7.6 to the right, agree=0.938, adj=0.111, (0 split)
    Pressure9am < 1005.3 to the right, agree=0.934, adj=0.056, (0 split)

Node number 2: 238 observations, complexity param=0.07894737
  predicted class=No  expected loss=0.105042 P(node) =0.9296875
  class counts:     213      25
  probabilities: 0.895 0.105
  left son=4 (208 obs) right son=5 (30 obs)
  Primary splits:
    Pressure3pm < 1010.25 to the right, improve=5.972899, (0 missing)
    Cloud3pm < 6.5 to the left, improve=4.475485, (0 missing)
    Pressure9am < 1019.75 to the right, improve=4.279291, (0 missing)
    WindGustSpeed < 64 to the left, improve=3.249967, (1 missing)
    Sunshine < 6.45 to the right, improve=2.650559, (2 missing)
  Surrogate splits:
    Pressure9am < 1012.65 to the right, agree=0.950, adj=0.600, (0 split)
    Temp9am < 22.7 to the left, agree=0.887, adj=0.100, (0 split)
    Humidity3pm < 14.5 to the right, agree=0.882, adj=0.067, (0 split)
    MaxTemp < 33.5 to the left, agree=0.878, adj=0.033, (0 split)
    Rainfall < 16.8 to the left, agree=0.878, adj=0.033, (0 split)

Node number 3: 18 observations
  predicted class=Yes  expected loss=0.2777778 P(node) =0.0703125
  class counts:     5      13
  probabilities: 0.278 0.722

Node number 4: 208 observations
  predicted class=No  expected loss=0.0625 P(node) =0.8125
  class counts:     195      13
  probabilities: 0.938 0.062

Node number 5: 30 observations, complexity param=0.07894737
  predicted class=No  expected loss=0.4 P(node) =0.1171875
  class counts:     18      12
  probabilities: 0.600 0.400
  left son=10 (14 obs) right son=11 (16 obs)
  Primary splits:
    Sunshine < 9.95 to the right, improve=5.667857, (0 missing)
    Temp9am < 17.55 to the right, improve=4.789140, (0 missing)
    Humidity3pm < 35.5 to the left, improve=3.471429, (0 missing)
    MaxTemp < 31.25 to the right, improve=2.921739, (0 missing)
    Temp3pm < 30.25 to the right, improve=2.921739, (0 missing)
  Surrogate splits:
    Temp9am < 17.8 to the right, agree=0.867, adj=0.714, (0 split)
    Cloud3pm < 4.5 to the left, agree=0.833, adj=0.643, (0 split)
    MinTemp < 14.15 to the right, agree=0.767, adj=0.500, (0 split)
    MaxTemp < 29.15 to the right, agree=0.767, adj=0.500, (0 split)
    Temp3pm < 30.25 to the right, agree=0.767, adj=0.500, (0 split)
Example Code in R

**Example (rpart Tree Object)**

```r
printcp(model)  # printcp for rpart objects
plotcp(model)
```
plotcp(model)

size of tree

X-val Relative Error

cp

Jennifer Evans (Clickfox) Twitter: JenniferE_CF

January 14, 2014 96 / 164
Example Code in R

Example (rpart Tree Object)

```r
plot(model)
text(model)
```
Humidity3pm< 71
Pressure3pm>=1010
Sunshine>=9.95
No
No
Yes
Yes

Jennifer Evans (Clickfox)
Example Code in R

Example (rpart Tree Object)
fancyRpartPlot(model)
Example Code in R

Example (rpart Tree Object)

```r
prp(model)
prp(model, type=2, extra=104, nn=TRUE, fallen.leaves=TRUE, faclen=0, varlen=0, shadow.col="grey", branch.lty=3)
```
prp(model, type=2, extra=104, nn=TRUE, fallen.leaves=TRUE, faclen=0, varlen=0, shadow.col="grey", branch.lty=3)
Example Code in R

Example (rpart Tree Predictions)

```r
pred <- predict(model, newdata=ds[test, vars], type="class")
pred.prob <- predict(model, newdata=ds[test, vars], type="prob")
```
Example Code in R

```r
Example (Na values and pruning)
table(is.na(ds))
dc.complete <- ds[complete.cases(ds),]
(nobs <- nrow(dc.complete))
set.seed(1426)
length(train.complete <- sample(nobs, 0.7*nobs))
length(test.complete <- setdiff(seq_len(nobs), train.complete))

#Prune tree
model$ cptable[which.min(model$ cptable[, "xerror"]), "CP"]
model <- rpart(formula=form, data=ds[train.complete, vars], cp=0)
printcp(model)
prune <- prune(model, cp=.01)
printcp(prune)
```
Example (Random Forest)

```r
# Random Forest from library(randomForest)
table(is.na(ds))
table(is.na(ds.complete))

# subset(ds, select=-c(Humidity3pm, Humidity9am, Cloud9am, Cloud3pm))
setnum <- colnames(ds.complete)[16:19]
ds.complete[,setnum] <- lapply(ds.complete[,setnum],
function(x) as.numeric(x))

ds.complete$Humidity3pm <- as.numeric(ds.complete$Humidity3pm)
ds.complete$Humidity9am <- as.numeric(ds.complete$Humidity9am)
```
Variables in the randomForest algorithm must be either factor or numeric, factors can not have more than 32 levels.
Example Code in R

Example (Random Forest)

1. `begTime <- Sys.time()`  
2. `set.seed(1426)`  
3. `model <- randomForest(formula=form, data=ds.complete[train.complete, vars])`  
4. `runTime <- Sys.time()-begTime`  
5. `runTime`  
6. `#Time difference of 0.3833725 secs`
Na values must be imputed, removed or otherwise fixed.
Bagging

Given a standard training set $D$ of size $n$, bagging generates $m$ new training sets $D_i$, each of size $n'$, by sampling from $D$ uniformly and with replacement. By sampling with replacement, some observations may be repeated in each $D_i$. If $n' = n$, then for large $n$ the set $D_i$ is expected to have the fraction $(1 - 1/e) \ (63.2)$ of the unique examples of $D$, the rest being duplicates.
Random Forest

Sampling with replacement (default)

VS

Sampling without replacement (sample size equals $1-1/e \approx .632$)
Example Code in R

Example (Random Forest, sampling without replacement)

```r
begTime <- Sys.time()
set.seed(1426)
model <- randomForest(formula=form, data=ds.complete[train, vars],
                      ntree=500, replace = FALSE, sampsize = .632*.7*nrow(ds),
                      na.action=na.omit)
runTime <- Sys.time()-begTime
runTime
#Time difference of 0.2392061 secs
```
Call:
randomForest(formula = form, data = ds.complete[train, vars],
ntree = 500, replace = FALSE,
sampsize = 0.632 * 0.7 * nrow(ds),
na.action = na.omit)

Type of random forest: classification
Number of trees: 500
No. of variables tried at each split: 4

OOB estimate of error rate: 11.35%

Confusion matrix:
No Yes class.error
No 186 4 0.02105263
Yes 22 17 0.56410256
### summary(model)

<table>
<thead>
<tr>
<th></th>
<th>Length</th>
<th>Class</th>
<th>Mode</th>
</tr>
</thead>
<tbody>
<tr>
<td>call</td>
<td>7</td>
<td>--none--</td>
<td>call</td>
</tr>
<tr>
<td>type</td>
<td>1</td>
<td>--none--</td>
<td>character</td>
</tr>
<tr>
<td>predicted</td>
<td>229</td>
<td>factor</td>
<td>numeric</td>
</tr>
<tr>
<td>err.rate</td>
<td>1500</td>
<td>--none--</td>
<td>numeric</td>
</tr>
<tr>
<td>confusion</td>
<td>6</td>
<td>--none--</td>
<td>numeric</td>
</tr>
<tr>
<td>votes</td>
<td>458</td>
<td>matrix</td>
<td>numeric</td>
</tr>
<tr>
<td>oob.times</td>
<td>229</td>
<td>--none--</td>
<td>numeric</td>
</tr>
<tr>
<td>classes</td>
<td>2</td>
<td>--none--</td>
<td>character</td>
</tr>
<tr>
<td>importance</td>
<td>20</td>
<td>--none--</td>
<td>numeric</td>
</tr>
<tr>
<td>importanceSD</td>
<td>0</td>
<td>--none--</td>
<td>NULL</td>
</tr>
<tr>
<td>localImportance</td>
<td>0</td>
<td>--none--</td>
<td>NULL</td>
</tr>
<tr>
<td>proximity</td>
<td>0</td>
<td>--none--</td>
<td>NULL</td>
</tr>
<tr>
<td>ntree</td>
<td>1</td>
<td>--none--</td>
<td>numeric</td>
</tr>
<tr>
<td>mtry</td>
<td>1</td>
<td>--none--</td>
<td>numeric</td>
</tr>
<tr>
<td>forest</td>
<td>14</td>
<td>--none--</td>
<td>list</td>
</tr>
<tr>
<td>y</td>
<td>229</td>
<td>factor</td>
<td>numeric</td>
</tr>
<tr>
<td>test</td>
<td>0</td>
<td>--none--</td>
<td>NULL</td>
</tr>
<tr>
<td>inbag</td>
<td>0</td>
<td>--none--</td>
<td>NULL</td>
</tr>
<tr>
<td>terms</td>
<td>3</td>
<td>terms</td>
<td>call</td>
</tr>
</tbody>
</table>
str(model)

List of 19
$ call : language randomForest(formula = form, data = ds.complete[train, vars], nrow = nrow(ds), na.action = na.omit)
$ type : chr "classification"
$ predicted : Factor w/ 2 levels "No","Yes": 1 2 1 1 1 1 1 1 2 1 ...
..$ attr(*, "names")= chr [1:229] "1" "305" "299" "161" ...
$ err.rate : num [1:500, 1:3] 0.25 0.197 0.197 0.203 0.193 ...
..$ attr(*, "dimnames")=List of 2
.. ..$ : NULL
.. ..$ : chr [1:3] "OOB" "No" "Yes"
$ confusion : num [1:2, 1:3] 186 22 4 17 0.0211 ...
..$ attr(*, "dimnames")=List of 2
.. ..$ : chr [1:2] "No" "Yes"
.. ..$ : chr [1:3] "No" "Yes" "class.error"
$ votes : matrix [1:229, 1:2] 0.821 0.373 0.993 0.938 0.648 ...
..$ attr(*, "dimnames")=List of 2
.. ..$ : chr [1:229] "1" "305" "299" "161" ...
.. ..$ : chr [1:2] "No" "Yes"
..$ attr(*, "class")= chr [1:2] "matrix" "votes"
$ oob.times : num [1:229] 156 158 153 162 145 163 144 140 162 156 ...
$ classes : chr [1:2] "No" "Yes"
$ importance : num [1:20, 1] 1.942 2.219 0.812 1.66 4.223 ...
..$ attr(*, "dimnames")=List of 2
.. ..$ : chr [1:20] "MinTemp" "MaxTemp" "Rainfall" "Evaporation" ...
.. ..$ : chr "MeanDecreaseGini"
$ importanceSD : NULL
$ localImportance : NULL
$ proximity : NULL
$ ntree : num 500
$ mtry : num 4
$ forest : List of 14
..$ nbigtree : int [1:500] 55 59 47 41 45 45 41 45 45 53 ...
$ nodestatus : int [1:1500] 1 1 1 1 1 1 1 1 1 1 ...
importance(model)

<table>
<thead>
<tr>
<th>Feature</th>
<th>MeanDecreaseGini</th>
</tr>
</thead>
<tbody>
<tr>
<td>MinTemp</td>
<td>1.94218091</td>
</tr>
<tr>
<td>MaxTemp</td>
<td>2.21923946</td>
</tr>
<tr>
<td>Rainfall</td>
<td>0.81216780</td>
</tr>
<tr>
<td>Evaporation</td>
<td>1.65985367</td>
</tr>
<tr>
<td>Sunshine</td>
<td>4.22307365</td>
</tr>
<tr>
<td>WindGustDir</td>
<td>1.28737544</td>
</tr>
<tr>
<td>WindGustSpeed</td>
<td>2.86639513</td>
</tr>
<tr>
<td>WindDir9am</td>
<td>1.32291299</td>
</tr>
<tr>
<td>WindDir3pm</td>
<td>0.98640540</td>
</tr>
<tr>
<td>WindSpeed9am</td>
<td>1.45308318</td>
</tr>
<tr>
<td>WindSpeed3pm</td>
<td>2.03903384</td>
</tr>
<tr>
<td>Humidity9am</td>
<td>2.57789758</td>
</tr>
<tr>
<td>Humidity3pm</td>
<td>4.01479068</td>
</tr>
<tr>
<td>Pressure9am</td>
<td>3.39200505</td>
</tr>
<tr>
<td>Pressure3pm</td>
<td>5.47003943</td>
</tr>
<tr>
<td>Cloud9am</td>
<td>1.19459943</td>
</tr>
<tr>
<td>Cloud3pm</td>
<td>3.52867349</td>
</tr>
<tr>
<td>Temp9am</td>
<td>1.87205125</td>
</tr>
<tr>
<td>Temp3pm</td>
<td>2.43780114</td>
</tr>
<tr>
<td>RainToday</td>
<td>0.09530246</td>
</tr>
</tbody>
</table>
Example Code in R

Example (Random Forest, predictions)

```r
pred <- predict(model, newdata=ds.complete[test.complete, vars])
```
Random Forest in parallel.
# Random Forest in parallel

```r
library(doParallel)

ntree = 500; numCore = 4
rep <- 125 # tree / numCore
registerDoParallel(cores=numCore)

begTime <- Sys.time()

set.seed(1426)

rf <- foreach(ntree=rep(rep, numCore), .combine=combine,
              .packages='randomForest') %dopar%
  randomForest(formula=form, data=ds.complete[train.complete, vars],
               ntree=ntree,
               mtry=6,
               importance=TRUE,
               na.action=na.roughfix, #can also use na.action = na.omit
               replace=FALSE)

runTime <- Sys.time()-begTime
runTime

# Time difference of 0.1990662 secs
```
mtry in model is 4, mtry in rf is 6, length(vars) is 24
### importance(model)

<table>
<thead>
<tr>
<th>Feature</th>
<th>Importance</th>
</tr>
</thead>
<tbody>
<tr>
<td>MinTemp</td>
<td>1.94218091</td>
</tr>
<tr>
<td>MaxTemp</td>
<td>2.21923946</td>
</tr>
<tr>
<td>Rainfall</td>
<td>0.81216780</td>
</tr>
<tr>
<td>Evaporation</td>
<td>1.65985367</td>
</tr>
<tr>
<td>Sunshine</td>
<td>4.22307365</td>
</tr>
<tr>
<td>WindGustDir</td>
<td>1.28737544</td>
</tr>
<tr>
<td>WindGustSpeed</td>
<td>2.86639513</td>
</tr>
<tr>
<td>WindDir9am</td>
<td>1.32291299</td>
</tr>
<tr>
<td>WindDir3pm</td>
<td>0.98640540</td>
</tr>
<tr>
<td>WindSpeed9am</td>
<td>1.45308318</td>
</tr>
<tr>
<td>WindSpeed3pm</td>
<td>2.03903384</td>
</tr>
<tr>
<td>Humidity9am</td>
<td>2.57789758</td>
</tr>
<tr>
<td>Humidity3pm</td>
<td>4.01479068</td>
</tr>
<tr>
<td>Pressure9am</td>
<td>3.39200505</td>
</tr>
<tr>
<td>Pressure3pm</td>
<td>5.47003943</td>
</tr>
<tr>
<td>Cloud9am</td>
<td>1.19459943</td>
</tr>
<tr>
<td>Cloud3pm</td>
<td>3.52867349</td>
</tr>
<tr>
<td>Temp9am</td>
<td>1.87205125</td>
</tr>
<tr>
<td>Temp3pm</td>
<td>2.43780114</td>
</tr>
<tr>
<td>RainToday</td>
<td>0.09530246</td>
</tr>
</tbody>
</table>

Jennifer Evans (Clickfox)
<table>
<thead>
<tr>
<th>Feature</th>
<th>No</th>
<th>Yes</th>
<th>MeanDecreaseAccuracy</th>
<th>MeanDecreaseGini</th>
</tr>
</thead>
<tbody>
<tr>
<td>MinTemp</td>
<td>4.3267184</td>
<td>1.95155029</td>
<td>4.99442421</td>
<td>2.86155742</td>
</tr>
<tr>
<td>MaxTemp</td>
<td>3.9312878</td>
<td>−0.09780772</td>
<td>3.90547258</td>
<td>1.48849836</td>
</tr>
<tr>
<td>Rainfall</td>
<td>2.2855083</td>
<td>−2.20735885</td>
<td>0.98774887</td>
<td>0.90515978</td>
</tr>
<tr>
<td>Evaporation</td>
<td>1.2689707</td>
<td>0.10371215</td>
<td>1.15792468</td>
<td>1.35614483</td>
</tr>
<tr>
<td>Sunshine</td>
<td>6.8039998</td>
<td>5.93794031</td>
<td>8.24985824</td>
<td>4.45780922</td>
</tr>
<tr>
<td>WindGustDir</td>
<td>1.5872508</td>
<td>1.27680275</td>
<td>1.89144917</td>
<td>1.54086784</td>
</tr>
<tr>
<td>WindGustSpeed</td>
<td>3.0957164</td>
<td>0.70399353</td>
<td>3.06926945</td>
<td>1.97903808</td>
</tr>
<tr>
<td>WindDir9am</td>
<td>0.5213394</td>
<td>−0.57654051</td>
<td>0.02179805</td>
<td>0.88987541</td>
</tr>
<tr>
<td>WindDir3pm</td>
<td>0.1040497</td>
<td>−1.44770324</td>
<td>−0.54034743</td>
<td>0.89222294</td>
</tr>
<tr>
<td>WindSpeed9am</td>
<td>−0.1505080</td>
<td>0.02852706</td>
<td>−0.13462800</td>
<td>1.04935574</td>
</tr>
<tr>
<td>WindSpeed3pm</td>
<td>0.1366695</td>
<td>−0.31714524</td>
<td>−0.09851747</td>
<td>1.41884397</td>
</tr>
<tr>
<td>Humidity9am</td>
<td>1.5489961</td>
<td>1.33257660</td>
<td>2.02454227</td>
<td>2.08965160</td>
</tr>
<tr>
<td>Humidity3pm</td>
<td>4.4863077</td>
<td>1.80261751</td>
<td>4.87818606</td>
<td>3.16858964</td>
</tr>
<tr>
<td>Pressure9am</td>
<td>4.2958737</td>
<td>−0.24148691</td>
<td>3.86763218</td>
<td>3.11008464</td>
</tr>
<tr>
<td>Pressure3pm</td>
<td>5.4833604</td>
<td>3.71822295</td>
<td>6.42073201</td>
<td>4.27664751</td>
</tr>
<tr>
<td>Cloud9am</td>
<td>1.0693219</td>
<td>1.13917891</td>
<td>1.48230288</td>
<td>0.80992904</td>
</tr>
<tr>
<td>Cloud3pm</td>
<td>4.9937359</td>
<td>4.99596404</td>
<td>6.86041634</td>
<td>4.23660266</td>
</tr>
<tr>
<td>Temp9am</td>
<td>3.1110895</td>
<td>0.65377234</td>
<td>3.15007711</td>
<td>1.77972882</td>
</tr>
<tr>
<td>Temp3pm</td>
<td>4.6953725</td>
<td>−0.93099648</td>
<td>4.11704265</td>
<td>1.54411562</td>
</tr>
<tr>
<td>RainToday</td>
<td>1.2889082</td>
<td>−0.69026060</td>
<td>0.95731681</td>
<td>0.07791137</td>
</tr>
</tbody>
</table>
Example Code in R

Example (Random Forest)

```r
pred <- predict(rf, newdata=ds.complete[test.complete, vars])
confusionMatrix(pred, ds.complete[test.complete, target])
```
Confusion Matrix and Statistics

<table>
<thead>
<tr>
<th>Reference</th>
<th>Prediction</th>
<th>No</th>
<th>Yes</th>
</tr>
</thead>
<tbody>
<tr>
<td>No</td>
<td>73</td>
<td>11</td>
<td></td>
</tr>
<tr>
<td>Yes</td>
<td>4</td>
<td>11</td>
<td></td>
</tr>
</tbody>
</table>

Accuracy : 0.8485  
95% CI : (0.7624, 0.9126)  
No Information Rate : 0.7778  
P-Value [Acc > NIR] : 0.05355

Kappa : 0.5055  
Mcnemar’s Test P-Value : 0.12134

Sensitivity : 0.9481  
Specificity : 0.5000  
Pos Pred Value : 0.8690  
Neg Pred Value : 0.7333  
Prevalence : 0.7778  
Detection Rate : 0.7374  
Detection Prevalence : 0.8485

'Positive' Class : No
Example Code in R

Example (Random Forest)

```
#Factor Levels
id <- which(!(ds$var.name %in% levels(ds$var.name)))
ds$var.name[id] <- NA
```
How to draw a Random Forest?
Random Forest Visualization
Evaluating the Model
Evaluating the Model

Methods and Metrics to Evaluate Model Performance

1. Resubstitution Estimate (internal estimate, biased)
2. Confusion matrix
3. ROC
4. Test Sample Estimation (independent estimate)
5. V-fold and N-fold Cross-Validation (resampling techniques)
6. RMSLE library(Metrics)
7. lift
Example Code in R

Example (ctree in package party)

```r
#Conditional Inference Tree
model <- ctree(formula=form, data=ds[train, vars])
```
ctree: plot(model)
Model formula:
RainTomorrow ~ MinTemp + MaxTemp + Rainfall + Evaporation + Sunshine +
    WindGustDir + WindGustSpeed + WindDir9am + WindDir3pm + WindSpeed9am +
    WindSpeed3pm + Humidity9am + Humidity3pm + Pressure9am +
    Pressure3pm + Cloud9am + Cloud3pm + Temp9am + Temp3pm + RainToday

Fitted party:
[1] root
  | [2] Sunshine <= 6.4
  |   | [3] Pressure3pm <= 1015.9: Yes (n = 29, err = 24.1%)
  |   | [4] Pressure3pm > 1015.9: No (n = 36, err = 8.3%)
  | [5] Sunshine > 6.4
  |   | [6] Cloud3pm <= 6
  |   |   | [7] Pressure3pm <= 1009.8: No (n = 18, err = 22.2%)
  |   |   | [8] Pressure3pm > 1009.8: No (n = 147, err = 1.4%)
  |   | [9] Cloud3pm > 6: No (n = 26, err = 26.9%)

Number of inner nodes: 4
Number of terminal nodes: 5
Both rpart and ctree recursively perform univariate splits of the dependent variable based on values on a set of covariates.

rpart employs information measures (such as the Gini coefficient) for selecting the current covariate.

ctree uses a significance test procedure in order to select variables instead of selecting the variable that maximizes an information measure. This may avoid some selection bias.
Example Code in R

Example (ctree in package party)

```r
# For class predictions:
library(caret)
pred <- predict(model, newdata=ds[test, vars])
confusionMatrix(pred, ds[test, target])
mc <- table(pred, ds[test, target])
err <- 1.0 - (mc[1,1] + mc[2,2]) / sum(mc) # resubstitution error rate
```
Confusion Matrix and Statistics

<table>
<thead>
<tr>
<th>Prediction</th>
<th>Reference</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>No</td>
<td>74</td>
<td>16</td>
</tr>
<tr>
<td>Yes</td>
<td>8</td>
<td>12</td>
</tr>
</tbody>
</table>

Accuracy : 0.7818
95% CI : (0.693, 0.8549)
No Information Rate : 0.7455
P–Value [Acc > NIR] : 0.2241

Kappa : 0.3654
Mcnemar’s Test P–Value : 0.1530

Sensitivity : 0.9024
Specificity : 0.4286
Pos Pred Value : 0.8222
Neg Pred Value : 0.6000
Prevalence : 0.7455
Detection Rate : 0.6727
Detection Prevalence : 0.8182

'Positive' Class : No
Example Code in R

Example (ctree in package party)

#For class probabilities:
pred.prob <- predict(model, newdata=ds[test, vars], type="prob")
ctree

summary(pred)

<table>
<thead>
<tr>
<th></th>
<th>No</th>
<th>Yes</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>90</td>
<td>20</td>
</tr>
</tbody>
</table>

summary(pred.prob)

<table>
<thead>
<tr>
<th></th>
<th>No</th>
<th>Yes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Min.</td>
<td>0.2414</td>
<td>0.01361</td>
</tr>
<tr>
<td>1st Qu.</td>
<td>0.7308</td>
<td>0.01361</td>
</tr>
<tr>
<td>Median</td>
<td>0.9167</td>
<td>0.08333</td>
</tr>
<tr>
<td>Mean</td>
<td>0.7965</td>
<td>0.20353</td>
</tr>
<tr>
<td>3rd Qu.</td>
<td>0.9864</td>
<td>0.26923</td>
</tr>
<tr>
<td>Max.</td>
<td>0.9864</td>
<td>0.75862</td>
</tr>
</tbody>
</table>

err

[1] 0.2
#For a roc curve:
library(ROCR)
pred <- do.call(rbind, as.list(pred))
summary(pred)
roc <- prediction(pred[,1], ds[<test, target])
plot(performance(roc, measure="tpr", x.measure="fpr"), colorize=TRUE)

#For a lift curve:
plot(performance(roc, measure="lift", x.measure="rpp"), colorize=TRUE)

#Sensitivity/Specificity Curve and Precision/Recall Curve:
#Sensitivity(i.e True Positives/Actual Positives)
#Specificity(i.e True Negatives/Actual Negatives)
plot(performance(roc, measure="sens", x.measure="spec"), colorize=TRUE)
plot(performance(roc, measure="prec", x.measure="rec"), colorize=TRUE)
roc

- False positive rate
- True positive rate
- Rate of positive predictions
- Lift value
- Sensitivity
- Specificity
- Precision
- Recall

Jennifer Evans (Clickfox)
Twitter: JenniferE_CF
January 14, 2014
# Example of using 10-fold cross-validation to evaluate your model

```r
model <- train(ds[, vars], ds[, target], method='rpart', tuneLength=10)
```

# cross validation

```r
# example
n <- nrow(ds)  # nobs
K <- 10        # for 10 validation cross sections
taille <- n/\text{\%}K
set.seed(5)
alea <- runif(n)
rang <- rank(alea)
bloc <- (rang-1)/\text{\%}taille +1
bloc <- as.factor(bloc)
print(summary(bloc))
```
Example Code in R

```r
all.err <- numeric(0)
  for(k in 1:K){
    model <- rpart(formula=form, data = ds[train,vars], method="class")
    pred <- predict(model, newdata=ds[test,vars], type="class")
    mc <- table(ds[test,target],pred)
    err <- 1.0 - (mc[1,1] +mc[2,2]) / sum(mc)
    all.err <- rbind(all.err,err)
  }
print(all.err)
(err.cv <- mean(all.err))
```
print(all.err)

[,1]
err 0.2
er 0.2
er 0.2
er 0.2
er 0.2
er 0.2
er 0.2
er 0.2
er 0.2
er 0.2
er 0.2
er 0.2
er 0.2
er 0.2

(err.cv <- mean(all.err))

[1] 0.2
Check out the **caret** package if you’re building predictive models in R. It implements a number of out-of-sample evaluation schemes, including bootstrap sampling, cross-validation, and multiple train/test splits.

caret is really nice because it provides a unified interface to all the models, so you don’t have to remember, e.g., that `treeresponse` is the function to get class probabilities from a `ctree` model.
Example Code in R

Example (Random Forest - cforest)

#Random Forest from library(party)
model <- cforest(formula=form, data=ds.complete[train.complete, vars])
Confusion Matrix and Statistics

<table>
<thead>
<tr>
<th>Prediction</th>
<th>No</th>
<th>Yes</th>
</tr>
</thead>
<tbody>
<tr>
<td>No</td>
<td>74</td>
<td>16</td>
</tr>
<tr>
<td>Yes</td>
<td>3</td>
<td>6</td>
</tr>
</tbody>
</table>

Accuracy : 0.8081
95% CI : (0.7166, 0.8803)

No Information Rate : 0.7778
P-Value [Acc > NIR] : 0.277720

Kappa : 0.2963
Mcnemar's Test P-Value : 0.005905

Sensitivity : 0.9610
Specificity : 0.2727
Pos Pred Value : 0.8222
Neg Pred Value : 0.6667
Prevalence : 0.7778
Detection Rate : 0.7475
Detection Prevalence : 0.9091

'Positive' Class : No
Best Model: randomForest with mty=4

Confusion Matrix and Statistics

<table>
<thead>
<tr>
<th>Reference</th>
<th>Prediction</th>
<th>No</th>
<th>Yes</th>
</tr>
</thead>
<tbody>
<tr>
<td>No</td>
<td>75</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>Yes</td>
<td>2</td>
<td>21</td>
<td></td>
</tr>
</tbody>
</table>

Accuracy : 0.9697
95% CI : (0.914, 0.9937)
No Information Rate : 0.7778
P-Value [Acc > NIR] : 6.393e-08

Kappa : 0.9137
Mcnemar’s Test P-Value : 1

Sensitivity : 0.9740
Specificity : 0.9545
Pos Pred Value : 0.9868
Neg Pred Value : 0.9130
Prevalence : 0.7778
Detection Rate : 0.7576
Detection Prevalence : 0.7677

'Positive' Class : No
### Example (Data for Today)

```r
> Today
MinTemp  MaxTemp  Rainfall  Evaporation  Sunshine  WindGustDir  WindGustSpeed
   12.4      24.4      3.4       1.6       2.3       NNW        30
WindDir9am  WindDir3pm  WindSpeed9am  WindSpeed3pm  Humidity9am  Humidity3pm
       N        NW        4         13         97         74
Pressure9am  Pressure3pm  Cloud9am  Cloud3pm  Temp9am  Temp3pm  RainToday
  1015.8    1014.1         8          7       15.3        20.4       Yes
```
Example Code in R

Example (Random Forest - cforest)

```r
> (predict(model, newdata=Today))
[1] Yes
Levels: No Yes

> (predict(model, newdata=Today, type="prob"))
$'50'
  RainTomorrow.No RainTomorrow.Yes
[1,] 0.3942876 0.6057124
```
Example Code in R

Example (Random Forest - randomForest)

> predict(model, newdata=Today)
50
Yes
Levels: No Yes

> predict(model, newdata=Today, type="prob")
  No   Yes
50 0.096 0.904
attr(,"class")
[1] "matrix" "votes"
Yes, it will rain tomorrow. There is a ninety percent chance of rain, and we are ninety-five percent confident that we have a five percent chance of being wrong.
Evaluating the Business Questions
Evaluating the Business Questions

- Is this of value?
- Is it understandable?
- How to communicate this to the business?
- Are you answering the question asked...?
“An approximate answer to the right problem is worth a good deal more than an exact answer to an approximate problem.”

~John Tukey
Tip

Get the advantage with creativity, understanding the data, data munging and meta data creation.
“The best way to have a good idea is to have a lot of ideas.”

~Linus Pauling
A lot of the data munging is done for you, you are given a nice flat file to work with. Knowing and understanding this process will enable you to find data leaks and holes in the data set. What did their data scientists miss?
Tip

Use some type of version control, write notes to yourself, read the forum comments.
Visualization
Pie Chart

Visualization (Sometimes you really just need a Pie Chart)
Recommended Reading

Christopher M. Bishop (2006)
Pattern Recognition and Machine Learning, *Information Science and Statistics*

Leo Breiman (1999)
Random Forest, [http://www.stat.berkeley.edu/~breiman/random-forests.pdf](http://www.stat.berkeley.edu/~breiman/random-forests.pdf)

George Casella and Roger L. Berger
Statistical Inference

Rachel Schutt and Cathy O'Neil (2013)
Doing Data Science, *Straight Talk from the Frontline*

Q. Ethan McCallum (2013)
Bad Data Handbook, *Mapping the World of Data Problems*

Graham Williams (2013)
References

Hothorn, Hornik, and Zeileis (2006)
party: A Laboratory for Recursive Partytioning,
http://cran.r-project.org/web/packages/party/vignettes/party.pdf

Torsten Hothorn and Achim Zeileis (2009)
A Toolbox for Recursive Partytioning,
http://www.r-project.org/conferences/useR-2009/slides/Hothorn+Zeileis.pdf

Torsten Hothorn (2013)
Machine Learning and Statistical Learning
http://cran.r-project.org/web/views/MachineLearning.html

Other Sources
StackExchange http://stackexchange.com
StackOverFlow http://stackoverflow.com
PackageDocumentation http://cran.r-project.org
Acknowledgment

Ken McGuire

Robert Bagley
Questions

Twitter Account: Jen@JenniferE_CF
Website for R Code: www.clickfox.com/ds_rcode
Email: jennifer.evans@clickfox.com