Data Science Using Open Souce 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

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

www.clickfox.com/ds_rcode

Overview

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

Data Science at Clickfox

Software Development

Activly 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?

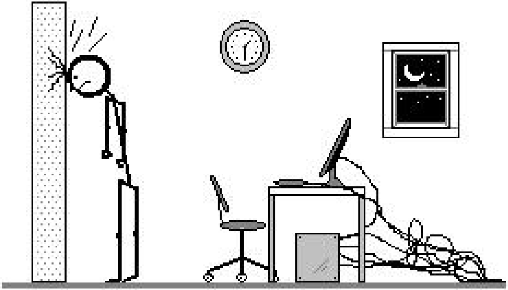
Data Preparation

Receive the Data

Raw Data

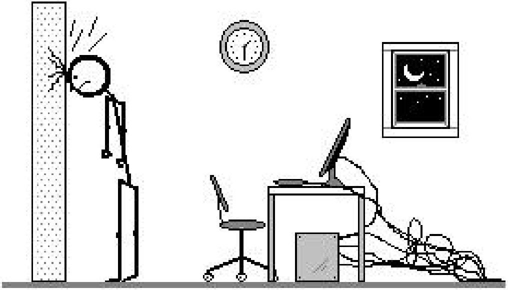
Data Munging

Begin Creating Analytic Data Set



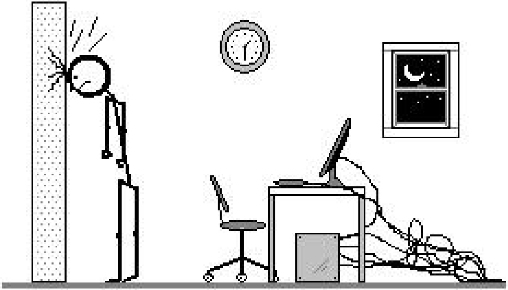
Data Munging

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

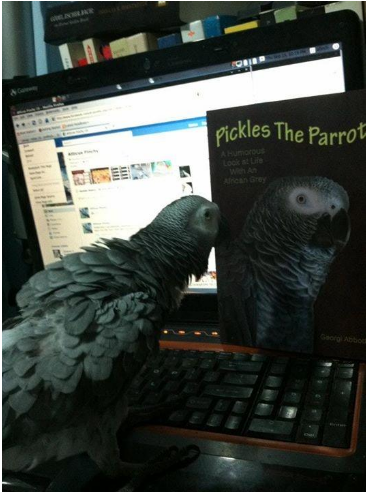
Bad Data

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...

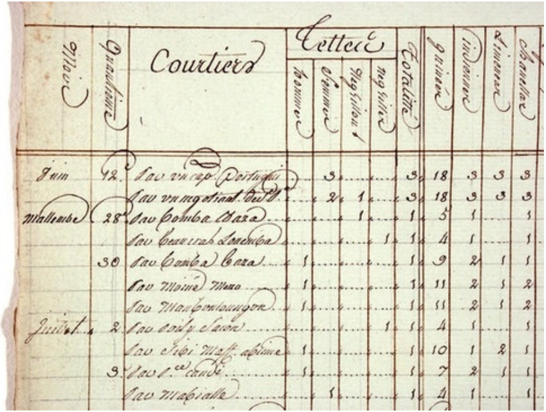


Not to mention all the things that we can do to really screw things up.

"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



Mois	Quantième	Courtiers	Lettres: Hommes	Femmes	Negrillons	Negrittes	Totalité	[illegible]	[illegible]	[illegible]	[illegible]
Juin	12	Sav un cap. Portugais		3			3	18	3	3	3
		Sav un negotiant [illegible]		2	1		3	18	3	3	3
Malembe	28	Sav Bomba Mara			1		1	5	1		1
		Sav [illegible]				1	1	4	1		1
	30	Sav Bomba Cara	1				1	9	2	1	1
		Sav Moine Mano	1				1	11	2	1	2
		Sav Manbouloungou	1				1	11	2	1	2
Juillet	2	Sav [illegible]				1	1	4	1		1
		Sav [illegible]	1				1	10	1	2	1
	3	Sav 1er [illegible]	1				1	7	2	1	1
		Sav Mabialle		1			1	4	1		1

Will it Rain Tomorrow?

Will it Rain Tomorrow?

Example (Variables)

```
> 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"
```

Will it Rain Tomorrow?

Example (First Four Rows of Data)

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

	WindGustSpeed	WindDir9am	WindDir3pm	WindSpeed9am	WindSpeed3pm	Humidity9am
1	30	SW	NW	6	20	68
2	39	E	W	4	17	80
3	85	N	NNE	6	6	82
4	54	WNW	W	30	24	62

	Humidity3pm	Pressure9am	Pressure3pm	Cloud9am	Cloud3pm	Temp9am	Temp3pm
1	29	1019.7	1015.0	7	7	14.4	23.6
2	36	1012.4	1008.4	5	3	17.5	25.7
3	69	1009.5	1007.2	8	7	15.4	20.2
4	56	1005.5	1007.0	2	7	13.5	14.1

	RainToday	RISK_MM	RainTomorrow
1	No	3.6	Yes
2	Yes	3.6	Yes
3	Yes	39.8	Yes
4	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.

head(ds)

```
        Date Location MinTemp MaxTemp Rainfall Evaporation Sunshine WindGustDir
1 2007-11-01 Canberra     8.0    24.3      0.0         3.4      6.3          NW
2 2007-11-02 Canberra    14.0    26.9      3.6         4.4      9.7         ENE
3 2007-11-03 Canberra    13.7    23.4      3.6         5.8      3.3          NW
4 2007-11-04 Canberra    13.3    15.5     39.8         7.2      9.1          NW
5 2007-11-05 Canberra     7.6    16.1      2.8         5.6     10.6         SSE
6 2007-11-06 Canberra     6.2    16.9      0.0         5.8      8.2          SE
  WindGustSpeed WindDir9am WindDir3pm WindSpeed9am WindSpeed3pm Humidity9am
1            30         SW         NW            6           20          68
2            39          E          W            4           17          80
3            85          N        NNE            6            6          82
4            54        WNW          W           30           24          62
5            50        SSE        ESE           20           28          68
6            44         SE          E           20           24          70
  Humidity3pm Pressure9am Pressure3pm Cloud9am Cloud3pm Temp9am Temp3pm
1          29      1019.7      1015.0        7        7    14.4    23.6
2          36      1012.4      1008.4        5        3    17.5    25.7
3          69      1009.5      1007.2        8        7    15.4    20.2
4          56      1005.5      1007.0        2        7    13.5    14.1
5          49      1018.3      1018.5        7        7    11.1    15.4
6          57      1023.8      1021.7        7        5    10.9    14.8
  RainToday RISK_MM RainTomorrow
1        No     3.6          Yes
2       Yes     3.6          Yes
3       Yes    39.8          Yes
4       Yes     2.8          Yes
5       Yes     0.0           No
6        No     0.2           No
```

Data Checking

♪ **There are numeric and categoric variables.**

Data Checking

Check the max/min do they make sense? What are the ranges? Do the numerical values need to be normalized?

summary(ds)

```
      Date                     Location      MinTemp          MaxTemp
 Min.   :2007-11-01   Canberra     :366   Min.   :-5.300   Min.   : 7.60
 1st Qu.:2008-01-31   Adelaide     :  0   1st Qu.: 2.300   1st Qu.:15.03
 Median :2008-05-01   Albany       :  0   Median : 7.450   Median :19.65
 Mean   :2008-05-01   Albury       :  0   Mean   : 7.266   Mean   :20.55
 3rd Qu.:2008-07-31   AliceSprings :  0   3rd Qu.:12.500   3rd Qu.:25.50
 Max.   :2008-10-31   BadgerysCreek:  0   Max.   :20.900   Max.   :35.80
                      (Other)      :  0
    Rainfall        Evaporation        Sunshine       WindGustDir
 Min.   : 0.000   Min.   : 0.200   Min.   : 0.000   NW     : 73
 1st Qu.: 0.000   1st Qu.: 2.200   1st Qu.: 5.950   NNW    : 44
 Median : 0.000   Median : 4.200   Median : 8.600   E      : 37
 Mean   : 1.428   Mean   : 4.522   Mean   : 7.909   WNW    : 35
 3rd Qu.: 0.200   3rd Qu.: 6.400   3rd Qu.:10.500   ENE    : 30
 Max.   :39.800   Max.   :13.800   Max.   :13.600   (Other):144
                                   NA's   :3        NA's   :  3
 WindGustSpeed     WindDir9am    WindDir3pm    WindSpeed9am     WindSpeed3pm
 Min.   :13.00   SE     : 47   WNW    : 61   Min.   : 0.000   Min.   : 0.00
 1st Qu.:31.00   SSE    : 40   NW     : 61   1st Qu.: 6.000   1st Qu.:11.00
 Median :39.00   NNW    : 36   NNW    : 47   Median : 7.000   Median :17.00
 Mean   :39.84   N      : 31   N      : 30   Mean   : 9.652   Mean   :17.99
 3rd Qu.:46.00   NW     : 30   ESE    : 27   3rd Qu.:13.000   3rd Qu.:24.00
 Max.   :98.00   (Other):151   (Other):139   Max.   :41.000   Max.   :52.00
 NA's   :2       NA's   : 31   NA's   :  1   NA's   :7
  Humidity9am     Humidity3pm      Pressure9am      Pressure3pm
 Min.   :36.00   Min.   :13.00   Min.   : 996.5   Min.   : 996.8
 1st Qu.:64.00   1st Qu.:32.25   1st Qu.:1015.4   1st Qu.:1012.8
 Median :72.00   Median :43.00   Median :1020.1   Median :1017.4
 Mean   :72.04   Mean   :44.52   Mean   :1019.7   Mean   :1016.8
 3rd Qu.:81.00   3rd Qu.:55.00   3rd Qu.:1024.5   3rd Qu.:1021.5
 Max.   :99.00   Max.   :96.00   Max.   :1035.7   Max.   :1033.2

    Cloud9am        Cloud3pm         Temp9am          Temp3pm        RainToday
```

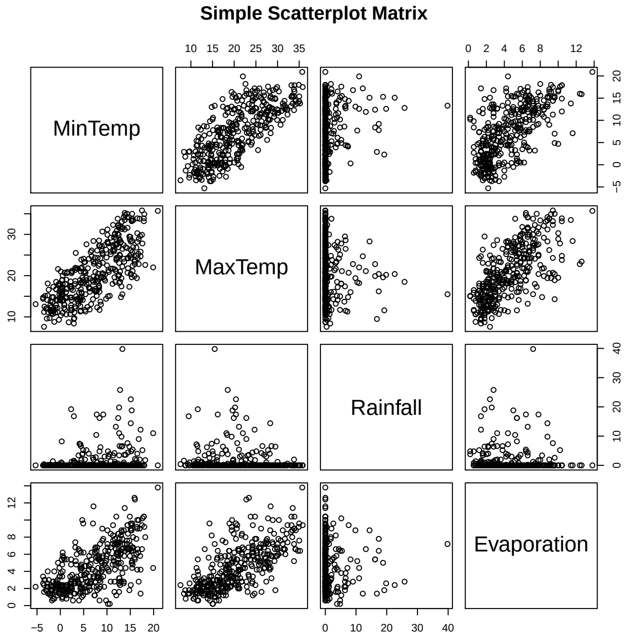
Data Checking

Plot variables against one another.

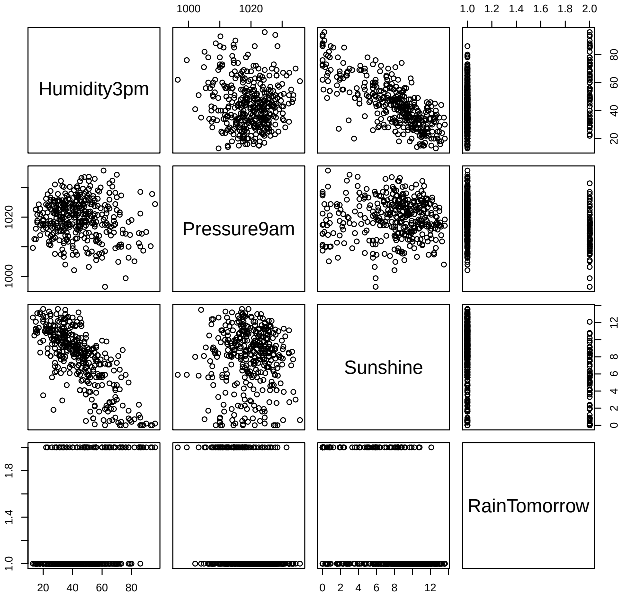
R Code

Example (Scatterplot)

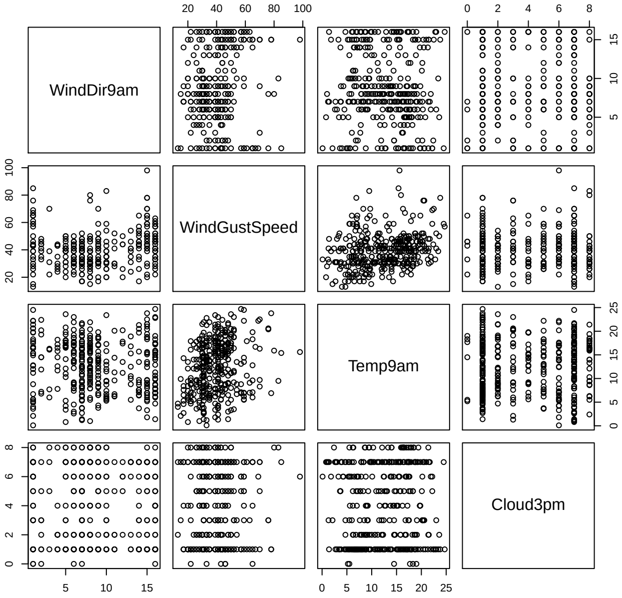
```
pairs(~MinTemp+MaxTemp+Rainfall+Evaporation, data = ds,
  main="Simple Scatterplot Matrix")
```


Simple Scatterplot Matrix
MinTemp
MaxTemp
Rainfall
Evaporation
10 15 20 25 30 35
0 2 4 6 8 12
−5 0 5 10 15 20
10 20 30
0 10 20 30 40
0 2 4 6 8 12
−5 0 5 10 15 20
0 10 20 30 40

Simple Scatterplot Matrix



Simple Scatterplot Matrix



Data Checking

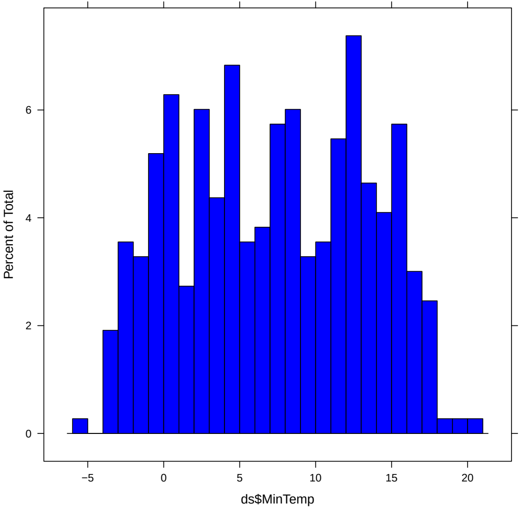
☆ **Create a histogram of numerical values in a data field, or kernel density estimate.**

R Code

Example (Histogram)

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

MinTemp

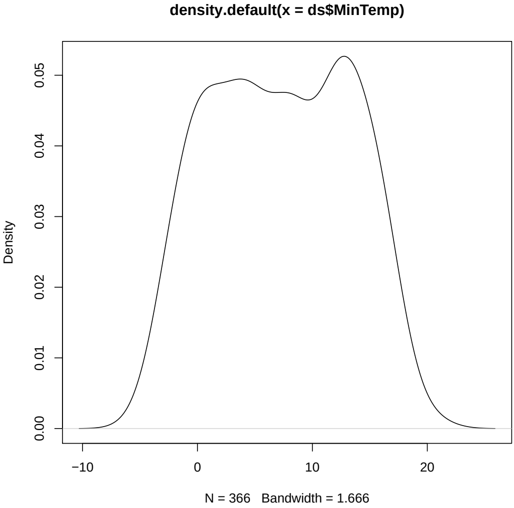


R Code

Example (Kernel Density Plot)

```
plot(density(ds$MinTemp))
```

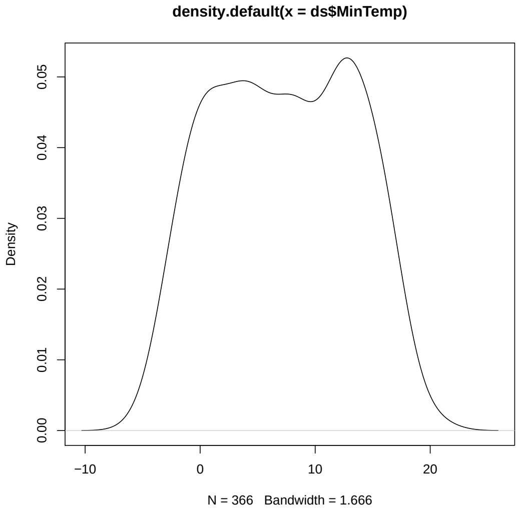
MinTemp

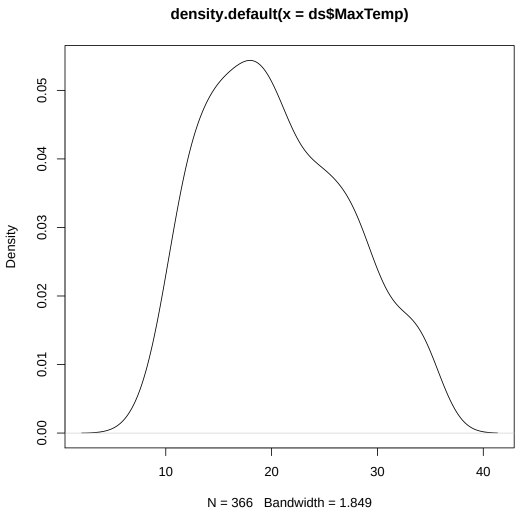


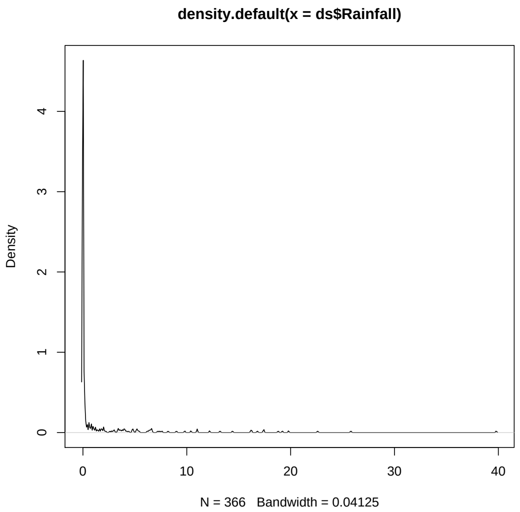
density.default(x = ds$MinTemp)
Density
0.00
0.01
0.02
0.03
0.04
0.05
−10
0
10
20
N = 366 Bandwidth = 1.666

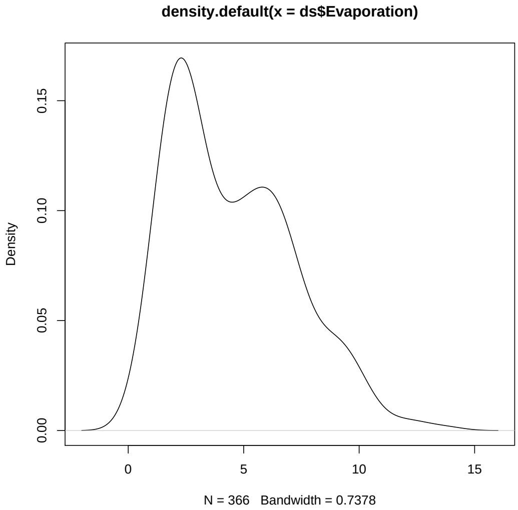
Data Checking

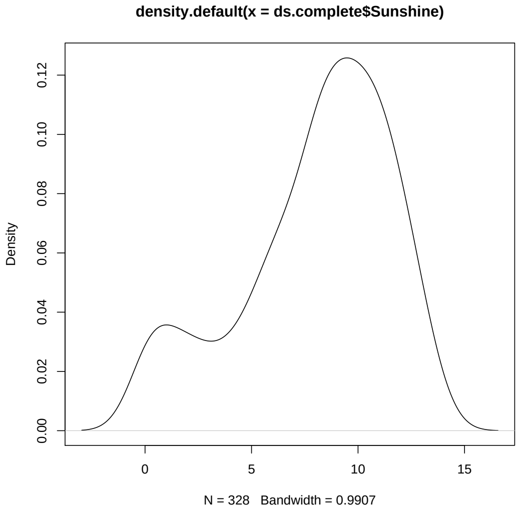
Kernel Density Plot for all Numerical Variables

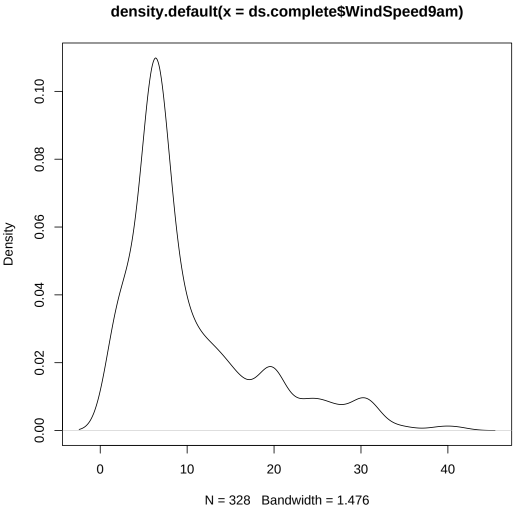

density.default(x = ds$MinTemp)
Density
0.00
0.01
0.02
0.03
0.04
0.05
−10
0
10
20
N = 366   Bandwidth = 1.666

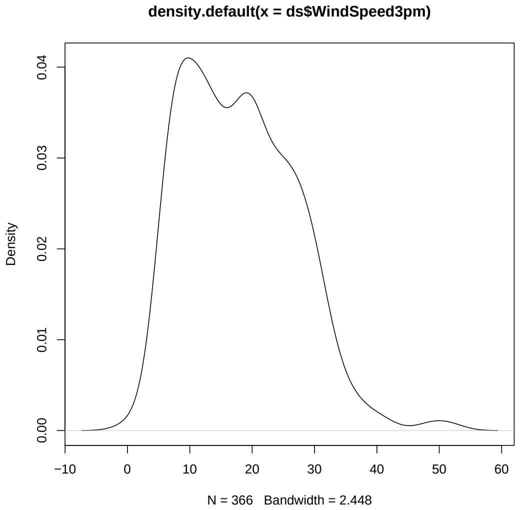

density.default(x = ds$MaxTemp)
Density
0.00
0.01
0.02
0.03
0.04
0.05
10
20
30
40
N = 366   Bandwidth = 1.849

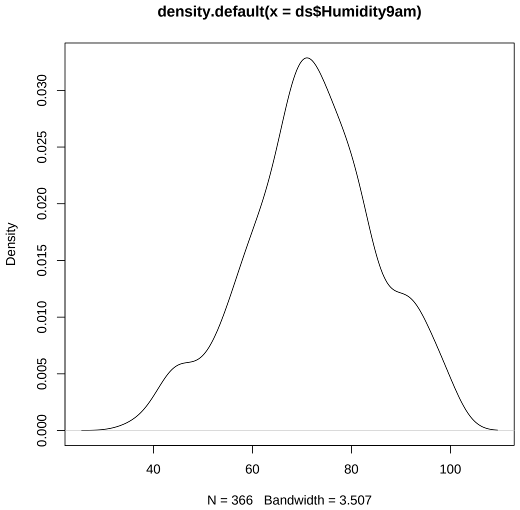

density.default(x = ds$Rainfall)
Density
0
1
2
3
4
0
10
20
30
40
N = 366   Bandwidth = 0.04125

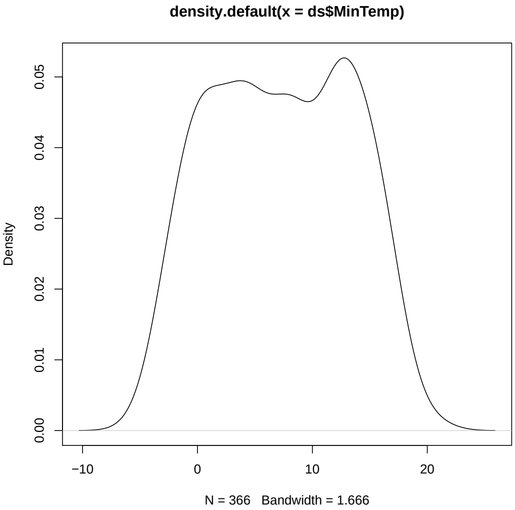

density.default(x = ds$Evaporation)
Density
0.00
0.05
0.10
0.15
0
5
10
15
N = 366 Bandwidth = 0.7378

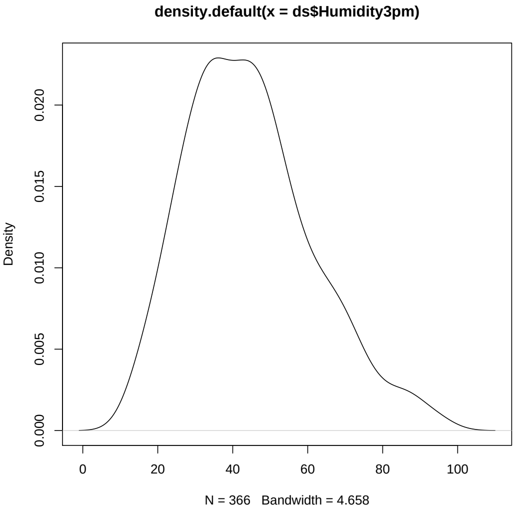

density.default(x = ds.complete$Sunshine)
Density
0.00
0.02
0.04
0.06
0.08
0.10
0.12
0
5
10
15
N = 328   Bandwidth = 0.9907

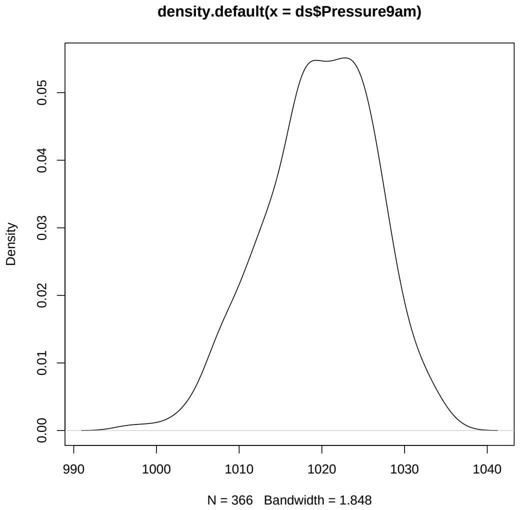

density.default(x = ds.complete$WindSpeed9am)
Density
0.00
0.02
0.04
0.06
0.08
0.10
0
10
20
30
40
N = 328 Bandwidth = 1.476

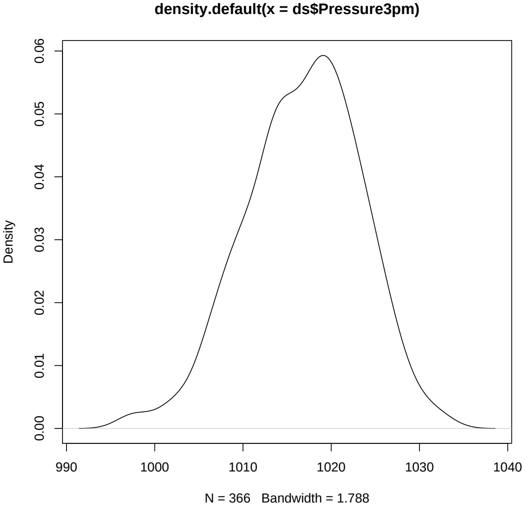

density.default(x = ds$WindSpeed3pm)
Density
0.00
0.01
0.02
0.03
0.04
−10
0
10
20
30
40
50
60
N = 366   Bandwidth = 2.448

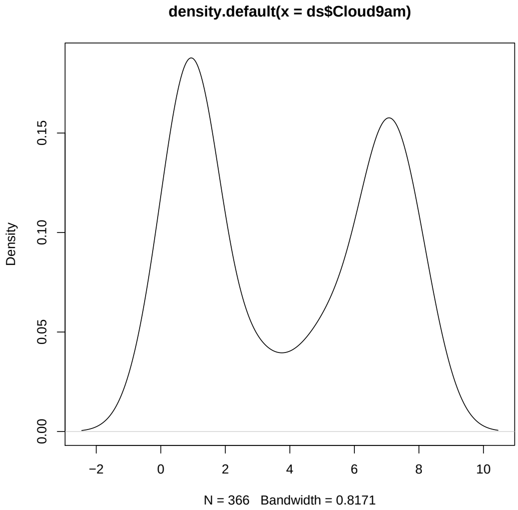

density.default(x = ds$Humidity9am)
Density
0.000
0.005
0.010
0.015
0.020
0.025
0.030
40
60
80
100
N = 366 Bandwidth = 3.507

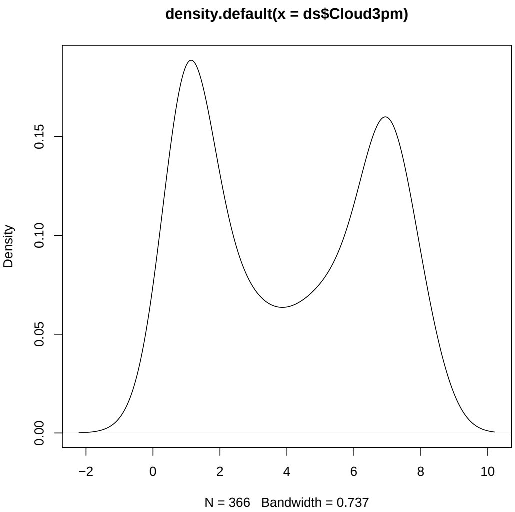

density.default(x = ds$MinTemp)
Density
0.00
0.01
0.02
0.03
0.04
0.05
−10
0
10
20
N = 366   Bandwidth = 1.666

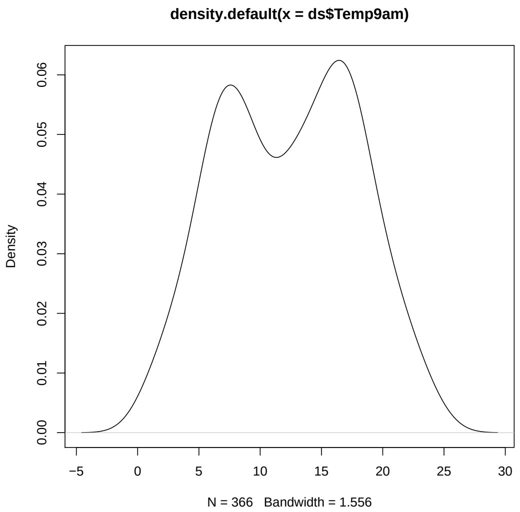

density.default(x = ds$Humidity3pm)
Density
0.000
0.005
0.010
0.015
0.020
0
20
40
60
80
100
N = 366   Bandwidth = 4.658

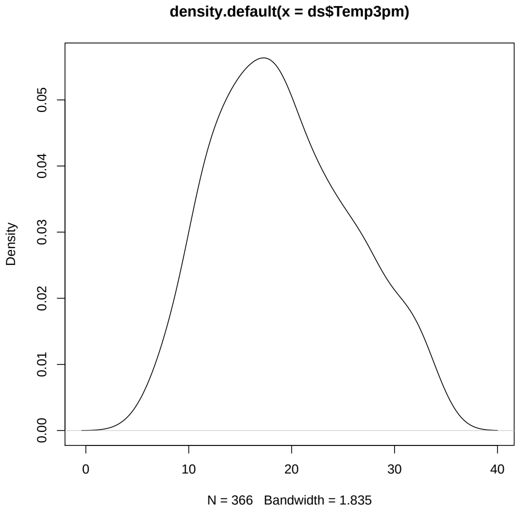

density.default(x = ds$Pressure9am)
Density
0.00
0.01
0.02
0.03
0.04
0.05
990
1000
1010
1020
1030
1040
N = 366   Bandwidth = 1.848


density.default(x = ds$Pressure3pm)
Density
0.00
0.01
0.02
0.03
0.04
0.05
0.06
990
1000
1010
1020
1030
1040
N = 366   Bandwidth = 1.788


density.default(x = ds$Cloud9am)
Density
0.00
0.05
0.10
0.15
−2
0
2
4
6
8
10
N = 366 Bandwidth = 0.8171


density.default(x = ds$Cloud3pm)
Density
0.00
0.05
0.10
0.15
−2
0
2
4
6
8
10
N = 366   Bandwidth = 0.737


density.default(x = ds$Temp9am)
Density
0.00
0.01
0.02
0.03
0.04
0.05
0.06
−5
0
5
10
15
20
25
30
N = 366 Bandwidth = 1.556


density.default(x = ds$Temp3pm)
Density
0.00
0.01
0.02
0.03
0.04
0.05
0
10
20
30
40
N = 366   Bandwidth = 1.835

Data Checking

♪ **There are missing values in 'Sunshine' and 'WindSpeed9am'.**

Data Checking

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.

Data Checking

Spillover of time-bound data

Check for duplicates - do not expect that data is perfectly partitioned.

Algorithms

"All models are wrong, some are useful."

~George Box

Movie Selection Explanation

Difference between Decision Trees and Random Forest

Movie Selection Explanation

Willow is a decision tree.

Movie Selection Explanation

Willow does not generalize well, so you want to ask a few more friends.

Random Friend

Rainbow Dash

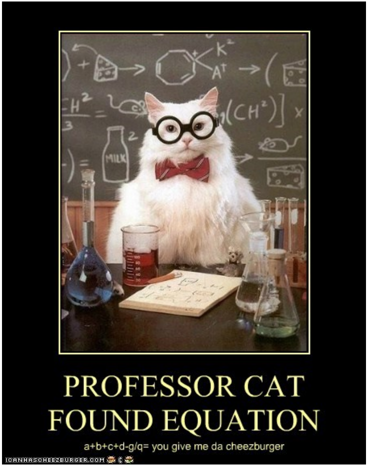
Random Friend

Cartman

Random Friend

Stay Puff Marshmallow

Professor Cat



Movie Selection Explanation

Your friends are an ensebmble of decision trees. But you dont want them all having the same information and giving the same answer.

Good and Bad Predictiors

- 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

Movie Selection Explanation

Thus, your friends now form a bagged (bootstrap aggregated) forest of your movie preferences.

Movie Selection Explanation

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)

$Willow + Cartman + StayPuff + ProfCat + Rainbowdash = AccutatePrediction$

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.

Forests 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

R Code

Example (Useful Commands)

```
#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

Classification and Regression Tree

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)

```
library(rpart) #Popular decision tree algorithm
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(RWeka) #Weka decision tree J48
library(evtree) #Evolutionary Algorithm, builds the tree from the bottom up
library(randomForest)
library(doParallel)
library(CHAID) #Chi-squared automatic interaction detection tree
library(tree)
library(caret)
```

R Code

Example (Data Prep)

```
data(weather)
dsname <- "weather"
target <- "RainTomorrow"
risk <- "RISK_MM"
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’’
```

R Code

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))
```

Note

♪ **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 Code in R

Example (rpart Tree)

```
model <- rpart(formula=form, data=ds[train, vars])
```

Note

♪ **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 Code in R

Example (rpart Tree Object)

```
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)

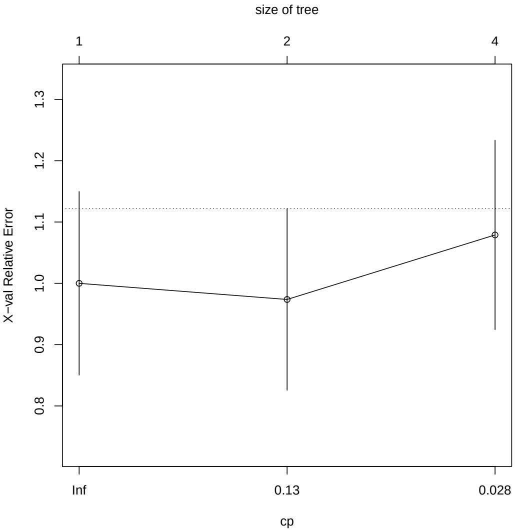
```
Call:
rpart(formula = form, data = ds[train, vars])
  n= 256

          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
```

Example Code in R

Example (rpart Tree Object)

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

plotcp(model)

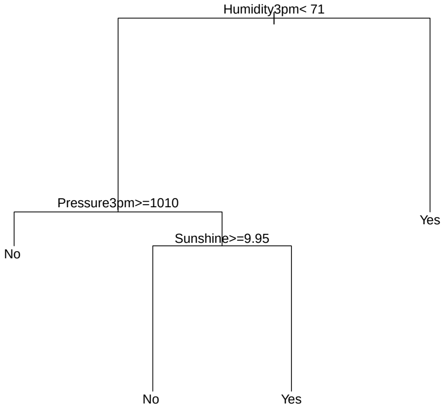


Example Code in R

Example (rpart Tree Object)

```
plot(model)
text(model)
```

plot(model)
text(model)

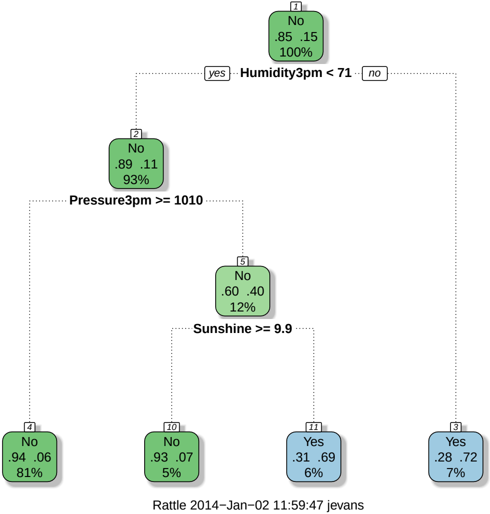


Example Code in R

Example (rpart Tree Object)

```
fancyRpartPlot(model)
```

fancyRpartPlot(model)

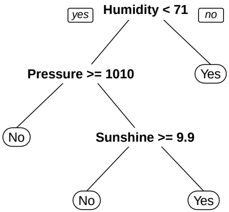


Example Code in R

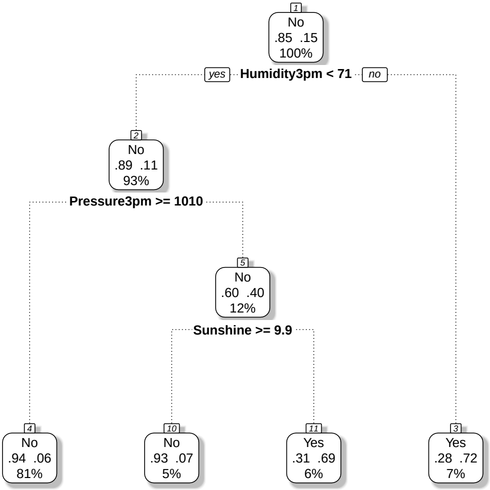
Example (rpart Tree Object)

```
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)



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)

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

Example Code in R

Example (Na values and pruning)

```
table(is.na(ds))
ds.complete <- ds[complete.cases(ds),]
(nobs <- nrow(ds.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 Code in R

Example (Random Forest)

```
#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)
```

Note

♪ **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)

```
begTime <- Sys.time()
set.seed(1426)
model <- randomForest(formula=form,data=ds.complete[train.complete,vars])
runTime <- Sys.time()-begTime
runTime
#Time difference of 0.3833725 secs
```

Note

♪ **Na values must be imputed, removed or otherwise fixed.**

Random Forest

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 = .632)

Example Code in R

Example (Random Forest, sampling without replacement)

```
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
```

print(model)

```
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)

```
                Length Class  Mode
call               7   -none- call
type               1   -none- character
predicted        229   factor numeric
err.rate        1500   -none- numeric
confusion          6   -none- numeric
votes            458   matrix numeric
oob.times        229   -none- numeric
classes            2   -none- character
importance        20   -none- numeric
importanceSD       0   -none- NULL
localImportance    0   -none- NULL
proximity          0   -none- NULL
ntree              1   -none- numeric
mtry               1   -none- numeric
forest            14   -none- list
y                229   factor numeric
test               0   -none- NULL
inbag              0   -none- NULL
terms              3   terms  call
```

str(model)

```
List of 19
 $ call           : language randomForest(formula = form, data = ds.complete[train, vars], n
replace = FALSE, sampsize = 0.632 * 0.7 * 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
  ..$ ndbigtree : int [1:500] 55 59 47 41 45 45 41 45 45 53 ...
  ..$ nodestatus: int [1:67, 1:500] 1 1 1 1 1 1 1 1 1 -1
```

importance(model)

```
              MeanDecreaseGini
MinTemp             1.94218091
MaxTemp             2.21923946
Rainfall            0.81216780
Evaporation         1.65985367
Sunshine            4.22307365
WindGustDir         1.28737544
WindGustSpeed       2.86639513
WindDir9am          1.32291299
WindDir3pm          0.98640540
WindSpeed9am        1.45308318
WindSpeed3pm        2.03903384
Humidity9am         2.57789758
Humidity3pm         4.01479068
Pressure9am         3.39200505
Pressure3pm         5.47003943
Cloud9am            1.19459943
Cloud3pm            3.52867349
Temp9am             1.87205125
Temp3pm             2.43780114
RainToday           0.09530246
```

Example Code in R

Example (Random Forest, predictions)

```r
pred <- predict(model, newdata=ds.complete[test.complete, vars])
```

Note

Random Forest in parallel.

Example Code in R

Example (Random Forest in parallel)

```r
#Random Forest in parallel
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
```

Note

♪ **mtry in model is 4, mtry in rf is 6, length(vars) is 24**

importance(model)

```
              MeanDecreaseGini
MinTemp             1.94218091
MaxTemp             2.21923946
Rainfall            0.81216780
Evaporation         1.65985367
Sunshine            4.22307365
WindGustDir         1.28737544
WindGustSpeed       2.86639513
WindDir9am          1.32291299
WindDir3pm          0.98640540
WindSpeed9am        1.45308318
WindSpeed3pm        2.03903384
Humidity9am         2.57789758
Humidity3pm         4.01479068
Pressure9am         3.39200505
Pressure3pm         5.47003943
Cloud9am            1.19459943
Cloud3pm            3.52867349
Temp9am             1.87205125
Temp3pm             2.43780114
RainToday           0.09530246
```

importance(rf)

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

Example Code in R

Example (Random Forest)

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

confusionMatrix(pred, ds.complete[test.complete, target])

```
Confusion Matrix and Statistics

          Reference
Prediction No Yes
       No  73  11
       Yes  4  11

               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

Methods and Metrics to Evaluate Model Performance

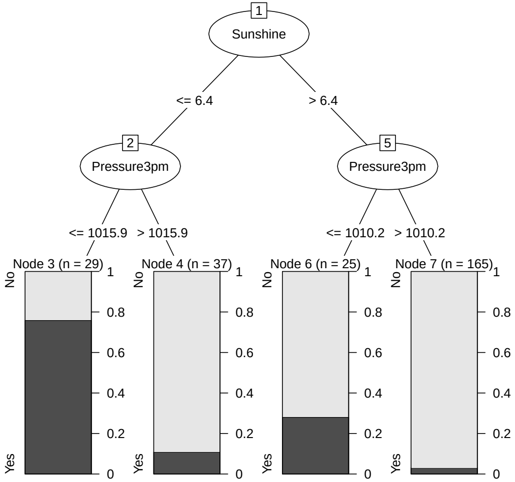
1. Resubstitution Estimate (internal estimate, biased)
2. Confusion matrix
3. ROC
4. Test Sample Estimation (independent estimate)
5. V-fold and N-fold Cross-Validation (resampling techniques)
6. RMSLE library(Metrics)
7. lift

Example Code in R

Example (ctree in package party)

```
#Conditional Inference Tree
model <- ctree(formula=form, data=ds[train, vars])
```

ctree: plot(model)



print(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
```

Difference between ctree and rpart

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)

```
#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
```

ctree

```
Confusion Matrix and Statistics

          Reference
Prediction No Yes
       No  74  16
       Yes  8  12

               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)
 No  Yes
 90   20
```

```
summary(pred.prob)
       No                    Yes
 Min.    :0.2414    Min.    :0.01361
 1st Qu.:0.7308    1st Qu.:0.01361
 Median  :0.9167    Median  :0.08333
 Mean    :0.7965    Mean    :0.20353
 3rd Qu.:0.9864    3rd Qu.:0.26923
 Max.    :0.9864    Max.    :0.75862
```

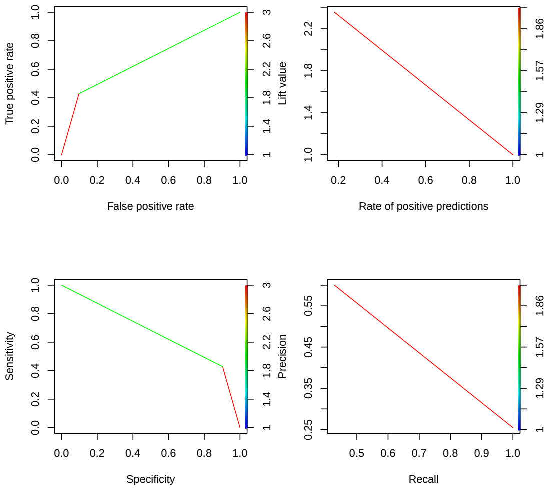
```
err
[1] 0.2
```

Example Code in R

Example (ctree in package party)

```
#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)
#Specifcity(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



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

Example Code in R

Example (crossvalidation)

```
#Example of using 10-fold cross-validation to evaluation your model

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

#cross validation
   #example
   n <- nrow(ds)  #nobs
   K <- 10              #for 10 validation cross sections
   taille <- n%/%K
   set.seed(5)
   alea <- runif(n)
   rang <- rank(alea)
   bloc <- (rang-1)%/%taille +1
   bloc <- as.factor(bloc)
   print(summary(bloc))
```

Example Code in R

Example (cross validation continued)

```
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
err  0.2
err  0.2
err  0.2
err  0.2
err  0.2
err  0.2
err  0.2
err  0.2
err  0.2
```

```
(err.cv <- mean(all.err))
[1] 0.2
```

caret Package

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])
```

cforest

```
Confusion Matrix and Statistics

          Reference
Prediction No Yes
       No  74  16
       Yes  3   6

               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

          Reference
Prediction No Yes
       No  75   1
       Yes  2  21

               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 Code in R

Example (Data for Today)

```
> 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 (Random Forest - cforest)

```
> (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"
```

Will it Rain Tomorrow?

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

Kaggle and Random Forest

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

Tip

A lot of the data munging is done for you, you are given a nice flat file to work with. Knowing and uderstanding 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*

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)
Decision Trees in R, *http://onepager.togaware.com/DTreesR.pdf*

References

Hothorn, Hornik, and Zeileis (2006)
party: : A Laboratory for Recursive Partytioning,
*h*ttp://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*