

Ensemble methods and random forests



Today's topics

- Ensemble Methods
 - Overview and rationale
 - Boosting
 - Bagging
- Random Forests
 - Overview
 - Some theory and mechanics
 - Practical examples with R



Ensemble Methods

Improve classification accuracy by combining classifiers

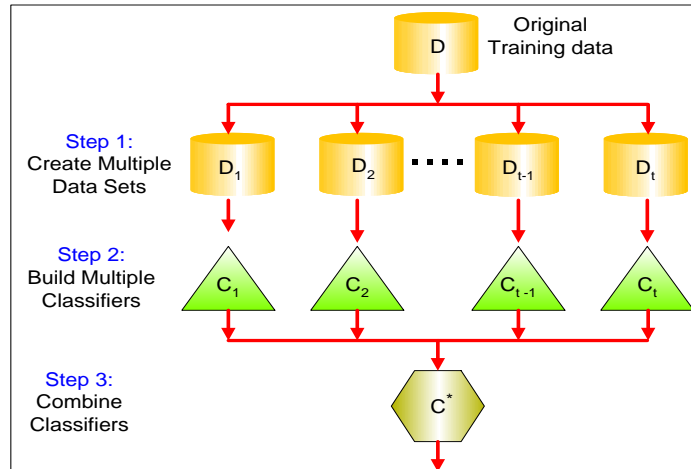


Note before we proceed

Our focus will be on explaining the algorithms intuitively
Mathematical proofs and assumptions are left for all of you
as an exercise

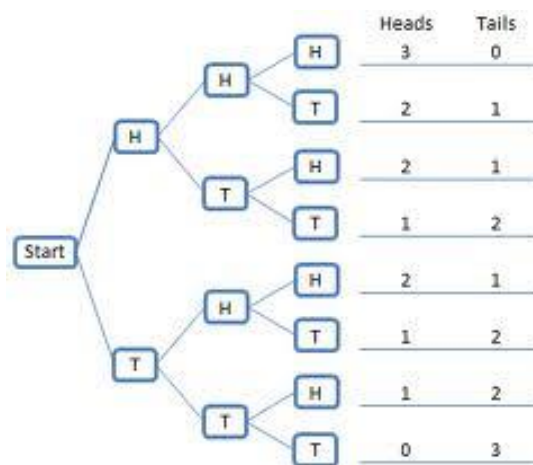


General Idea of ensemble methods



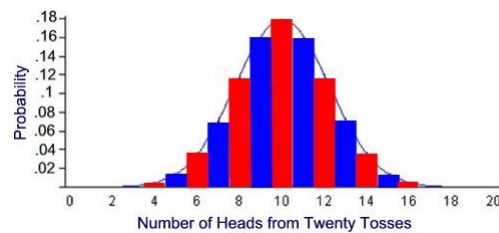
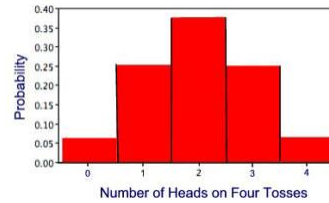
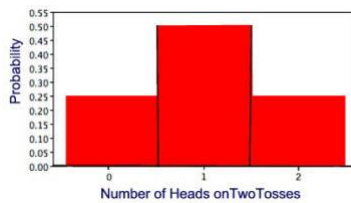
data science dojo
unleash the data scientist in you

Digression – Binomial distribution



data science dojo
unleash the data scientist in you

Number of heads in n trials



datasciencedojo
unleash the data scientist in you

Binomial distribution

$$f(k; n, p) = \Pr(X = k) = \binom{n}{k} p^k (1 - p)^{n-k}$$

datasciencedojo
unleash the data scientist in you

Tossing a biased coin



What is the probability of 0,1,2,3,4,5,6 heads in six tosses of a biased coin in six tosses total?

$n = 6$

$k = 0,1,2,3,4,5,6$

$P=0.3$ (prob of head)

$$\Pr(0 \text{ heads}) = f(0) = \Pr(X = 0) = \binom{6}{0} 0.3^0 (1 - 0.3)^{6-0} \approx 0.1176$$

$$\Pr(1 \text{ head}) = f(1) = \Pr(X = 1) = \binom{6}{1} 0.3^1 (1 - 0.3)^{6-1} \approx 0.3025$$

$$\Pr(2 \text{ heads}) = f(2) = \Pr(X = 2) = \binom{6}{2} 0.3^2 (1 - 0.3)^{6-2} \approx 0.3241$$

$$\Pr(3 \text{ heads}) = f(3) = \Pr(X = 3) = \binom{6}{3} 0.3^3 (1 - 0.3)^{6-3} \approx 0.1852$$

$$\Pr(4 \text{ heads}) = f(4) = \Pr(X = 4) = \binom{6}{4} 0.3^4 (1 - 0.3)^{6-4} \approx 0.0595$$

$$\Pr(5 \text{ heads}) = f(5) = \Pr(X = 5) = \binom{6}{5} 0.3^5 (1 - 0.3)^{6-5} \approx 0.0102$$

$$\Pr(6 \text{ heads}) = f(6) = \Pr(X = 6) = \binom{6}{6} 0.3^6 (1 - 0.3)^{6-6} \approx 0.0007$$

datasciencedojo
unleash the data scientist in you

Rolling a die



- When rolling a die 100 times, what is the probability of rolling a "4" exactly 25 times?
- Number of trials $n = 100$
- Number of successes $k = 25$
- Probability of a '4' in a single roll $= \frac{1}{6}$

$$\binom{100}{25} \left(\frac{1}{6}\right)^{25} \left(1 - \left(\frac{1}{6}\right)\right)^{100-25}$$

R functions:

`dbinom(25,size=100,prob=1/6) = 0.009825882`

`Pbinom` is for calculating cdfs

datasciencedojo
unleash the data scientist in you

Multiple-choice Test



- A test consists of 10 multiple choice questions with five choices for each question. As an experiment, you **guess** on each and every answer without even reading the questions.
 - What is the probability of getting **exactly 6** questions correct on this test?
 - What is the probability of getting **at least 6** questions correct on this test?
 - What is the probability of getting **less than 6** questions correct on this test?
 - What is the probability of getting **all 10** questions correct on this test?

datasciencedojo
unleash the data scientist in you

Solve on whiteboard

datasciencedojo
unleash the data scientist in you

applications

- Number of life insurance holders who will claim in a given period
 - Number of loan holders who will default in a certain period
 - Number of false starts of a car in n attempts
 - Number of faulty items in n samples from a production line
-
- **AND Ensemble Methods**



Why does it work?

- Suppose there are 25 base classifiers
 - Each classifier has error rate, $\epsilon = 0.35$
 - Assume classifiers are independent
 - Probability that the ensemble classifier makes a wrong prediction:

$$\sum_{i=13}^{25} \binom{25}{i} \epsilon^i (1 - \epsilon)^{25-i} = 0.06$$



Examples of Ensemble Methods

Bagging

- All classifiers are created equal

Boosting

- **Not** all classifiers are created equal



Bagging

- Sampling with replacement

Original Data	1	2	3	4	5	6	7	8	9	10
Bagging (Round 1)	7	8	10	8	2	5	10	10	5	9
Bagging (Round 2)	1	4	9	1	2	3	2	7	3	2
Bagging (Round 3)	1	8	5	10	5	5	9	6	3	7

- Build classifier on each bootstrap sample
- For a training data of size n , each sample has probability $[1 - (1 - 1/n)^n]$ of being selected
- If n is large, this approximates to $1 - 1/e \sim 0.632$

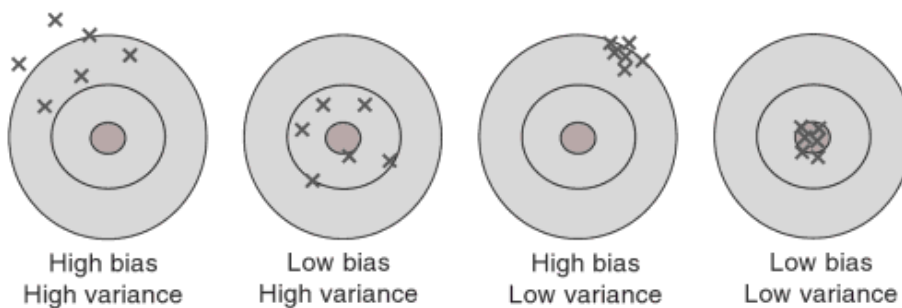


bagging

- Reduces variance in estimate
- Prevents overfitting
- Robust to outliers

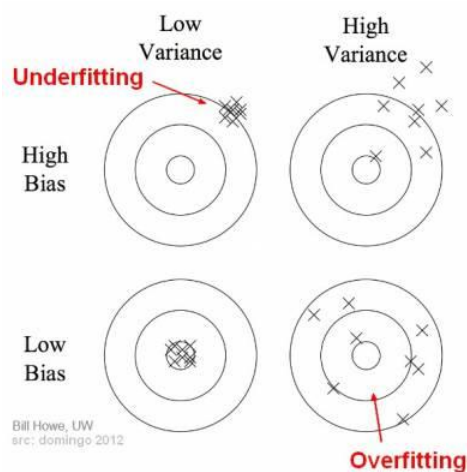
data^{science}dojo
unleash the data scientist in you

UNDERSTANDING BIAS and variance In estimate



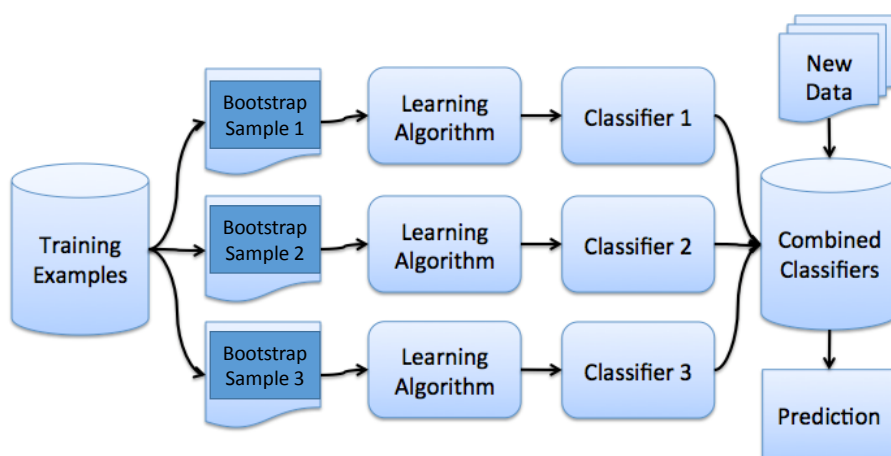
data^{science}dojo
unleash the data scientist in you

UNDERSTANDING BIAS and variance In estimate



data science dojo
unleash the data scientist in you

bagging



data science dojo
unleash the data scientist in you

Boosting

- An iterative procedure to adaptively change distribution of training data by focusing more on previously misclassified records
 - Initially, all N records are assigned equal weights
 - Unlike bagging, weights may change at the end of boosting round

datasciencedojo
unleash the data scientist in you

Boosting

- Records that are wrongly classified will have their weights increased
- Records that are classified correctly will have their weights decreased

Original Data	1	2	3	4	5	6	7	8	9	10
Boosting (Round 1)	7	3	2	8	7	9	4	10	6	3
Boosting (Round 2)	5	4	9	4	2	5	1	7	4	2
Boosting (Round 3)	4	4	8	10	4	5	4	6	3	4

- Example 4 is hard to classify
- Its weight is increased, therefore it is more likely to be chosen again in subsequent rounds

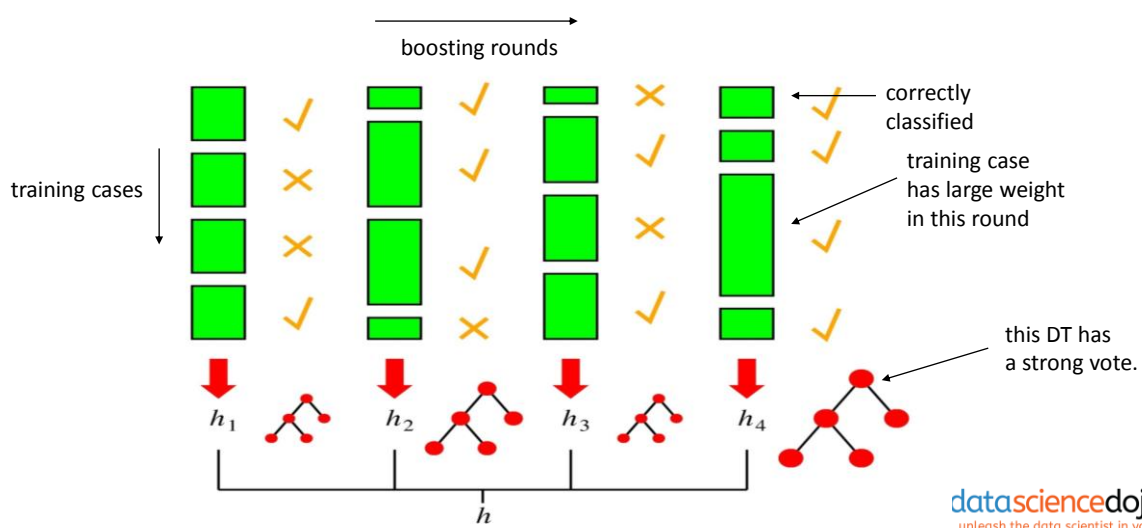
datasciencedojo
unleash the data scientist in you

Boosting Intuition

- We adaptively weigh each data case.
- Data cases which are wrongly classified get high weight (the algorithm will focus on them).
- Each boosting round learns a new (simple) classifier on the weighed dataset.
- These classifiers are weighed to combine them into a single powerful classifier.
- Classifiers that obtain low training error rate have high weight.
- We stop by monitoring a hold out set (cross-validation).

datasciencedojo
unleash the data scientist in you

Boosting in a Picture



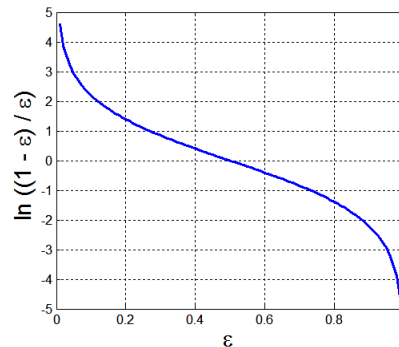
AdaBoost (adaptive Boosting)

- Base classifiers: C_1, C_2, \dots, C_T
- Error rate [Weighted loss function]:

$$\varepsilon_i = \frac{1}{N} \sum_{j=1}^N w_j \delta(C_i(x_j) \neq y_j)$$

- Importance of a classifier:

$$\alpha_i = \frac{1}{2} \ln \left(\frac{1 - \varepsilon_i}{\varepsilon_i} \right)$$



datasciencedojo
unleash the data scientist in you

AdaBoost

- Weight update:

$$w_i^{(j+1)} = \frac{w_i^{(j)}}{Z_j} \begin{cases} \exp^{-\alpha_j} & \text{if } C_j(x_i) = y_i \\ \exp^{\alpha_j} & \text{if } C_j(x_i) \neq y_i \end{cases}$$

where Z_j is the normalization factor

- If any intermediate rounds produce error rate higher than 50%, the weights are reverted back to $1/n$ and the resampling procedure is repeated.

- Classification:
$$C^*(x) = \arg \max_y \sum_{j=1}^T \alpha_j \delta(C_j(x) = y)$$

datasciencedojo
unleash the data scientist in you

Random forests classifier

- *Random forests are a **combination of tree predictors** such that each tree depends on the values of a random vector sampled independently and with same distribution for all trees in the forest.*

• - Leo Breiman



What is a random forest?

- An ensemble classifier using many decision tree models
- Can be used for classification or regression
- Accuracy and variable importance information is built-in



How random forests work?

- A different subset of the training data are selected ($\sim 2/3$), with replacement, to train each tree
- Remaining training data (aka out-of-bag data or simply OOB) is used to estimate error and variable importance
- Class assignment is made by the number of votes from all of the trees, and for regression, the average of the results is used



Which features are used for learning?

- A randomly selected subset of variables is used to split each node
- The number of variables used is decided by the user (mtry parameter in R)
- A smaller subset produces less correlation (lower error rate) but lower predictive power (high error rate)

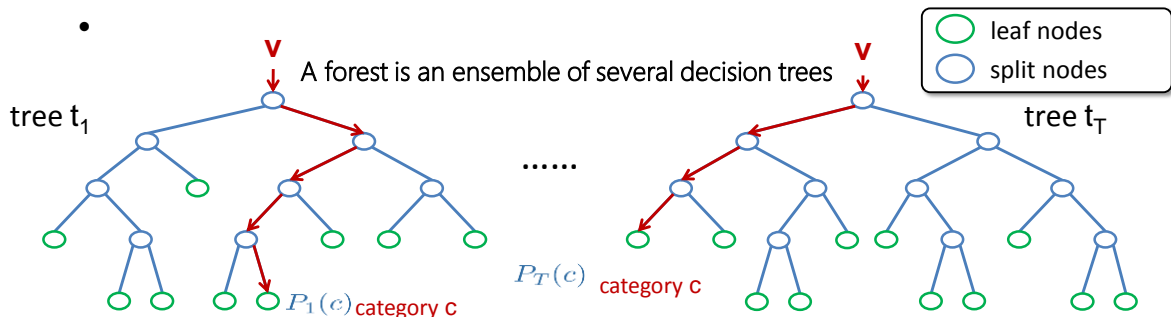


Rules of thumb

- **Given:**
- **N:** Total number of training data points
- **M:** Number of features in training data
- **m:** Number of features randomly selected for training each tree
- Sample the data with replacement N times for building the training data for each tree.
- $m \ll M$
- Classification: $m = \sqrt{M}$
- Regression: $m = M/3$

datasciencedojo
unleash the data scientist in you

A Forest of Trees



Classification is

$$P(c|\mathbf{v}) = \frac{1}{T} \sum_{t=1}^T P_t(c|\mathbf{v})$$

[Amit & Geman 97]
[Breiman 01]
[Lepetit *et al.* 06]

datasciencedojo
unleash the data scientist in you

Learning a Forest

- Dividing training examples into T subsets improves generalization
 - Reduces memory requirements & training time
- Train each decision tree t on subset I_t
 - Same decision tree learning as before
- Multi-core friendly (GPU implementation)



Implementation Details

- How many features and thresholds to try?
 - Just one = "extremely randomized" [Geurts *et al.* 06]
 - Few → fast training, may under-fit, may be too deep
 - Many → slower training, may over-fit
- When to stop growing the tree?
 - Maximum depth
 - Minimum entropy gain
 - Delta class distribution
 - Pruning



Common pitfall

A Random Forest and a Boosted Decision Tree are not the same

WWW.YOTTANEXT.COM

datasciencedojo
unleash the data scientist in you

forest error rate

$$PE^* \leq \rho(1 - s^2) / s^2$$

- The *correlation* between any two trees in the forest. Increasing the correlation increases the forest error rate.
- The *strength* of each individual tree in the forest. A tree with a low error rate is a strong classifier. Increasing the strength of the individual trees decreases the forest error rate.
- Detailed proof: RANDOM FORESTS Leo Breiman

datasciencedojo
unleash the data scientist in you

Random Forest

R package for
Random Forest



Setting up randomForest

Installation

```
> install.packages('randomForest')
```

Loading in R environment

```
> library(randomForest)
```

Documentation:

<http://cran.r-project.org/web/packages/randomForest/randomForest.pdf>



Obtaining the model

```
> iris.rf <- randomForest(
  Species ~ .,
  data=iris,
  importance=TRUE,
  proximity=TRUE
)
```



Printing the model

```
> print(iris.rf)
```

Type of random forest: classification Number of trees: 500

No. of variables tried at each split: 2

OOB estimate of error rate: 4.67%

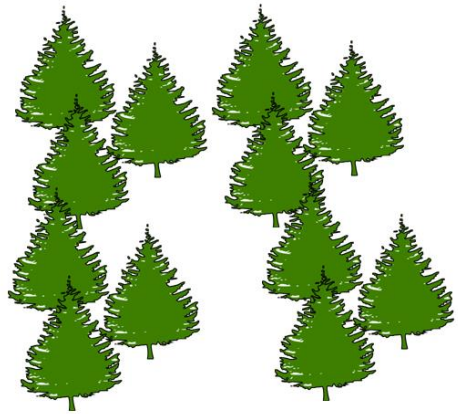
Confusion matrix: setosa versicolor virginica class.error

# setosa	50	0	0	0.00
# versicolor	0	47	3	0.06
# virginica	0	4	46	0.08



Restricting Tree Size

```
# A forest of small trees.
# Specify that you want 30 trees with
# a depth of 4 nodes max each.
```



```
(treesize(randomForest(Species ~ ., data=iris,maxnodes=4, ntree=30)))
```

datasciencedojo
unleash the data scientist in you

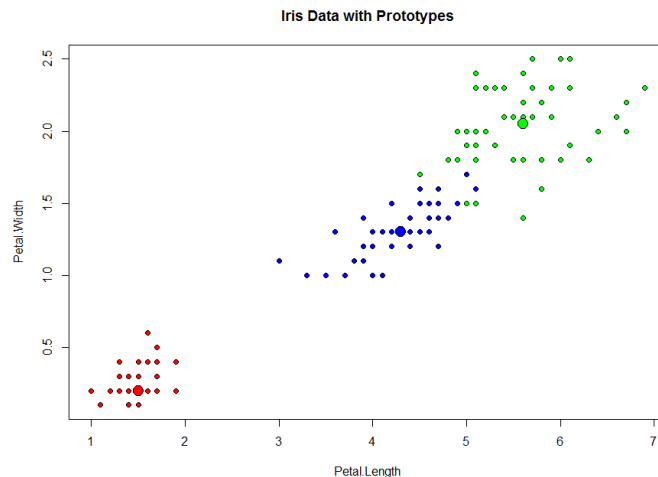
Finding Class Centers

classCenter function

```
> data(iris)
> iris.rf <- randomForest(iris[,-5], iris[,5], prox=TRUE)
> iris.p <- classCenter(iris[,-5], iris[,5], iris.rf$prox)
> plot(iris[,3], iris[,4],
      pch=21, xlab=names(iris)[3], ylab=names(iris)[4],
      bg=c("red", "blue", "green")[as.numeric(factor(iris$Species))],
      main="Iris Data with Prototypes")
> points(iris.p[,3], iris.p[,4], pch=21, cex=2,
        bg=c("red", "blue", "green"))
```

datasciencedojo
unleash the data scientist in you

Finding Class Centers



datasciencedojo
unleash the data scientist in you

Combining Trees

You can combine two or more random forests into one.
The confusion, err.rate, mse, and rsq components will be NULL

```
> data(iris)
> rf1 <- randomForest(Species ~ ., iris, ntree=50,
  norm.votes=FALSE)
> rf2 <- randomForest(Species ~ ., iris, ntree=100,
  norm.votes=FALSE)
> rf3 <- randomForest(Species ~ ., iris, ntree=150, mtry=3,
  norm.votes=FALSE)
> rf.all <- combine(rf1, rf2, rf3)
> print(rf.all)
```

datasciencedojo
unleash the data scientist in you

Combining Trees

```
> print(rf.all)
```

Call:

```
randomForest(formula = Species ~ ., data = iris, ntree = 50,
norm.votes = FALSE)
```

Type of random forest:

classification Number of trees: 150

No. of variables tried at each split: 2



Variable Importance and Gini Importance

```
> data(iris)
```

```
> iris.rf <- randomForest(Species ~., data=iris, importance=TRUE,
proximity=TRUE)
```

```
> round(importance(iris.rf), 2)
```

	setosa	versicolor	virginica	MeanDecreaseAccuracy	MeanDecreaseGini
Sepal.Length	6.28	8.88	7.11	10.48	9.26
Sepal.Width	4.87	0.77	4.85	5.17	2.33
Petal.Length	21.48	33.88	28.44	33.95	42.97
Petal.Width	22.96	32.47	32.10	34.60	44.65



Important factors in determining car mileage

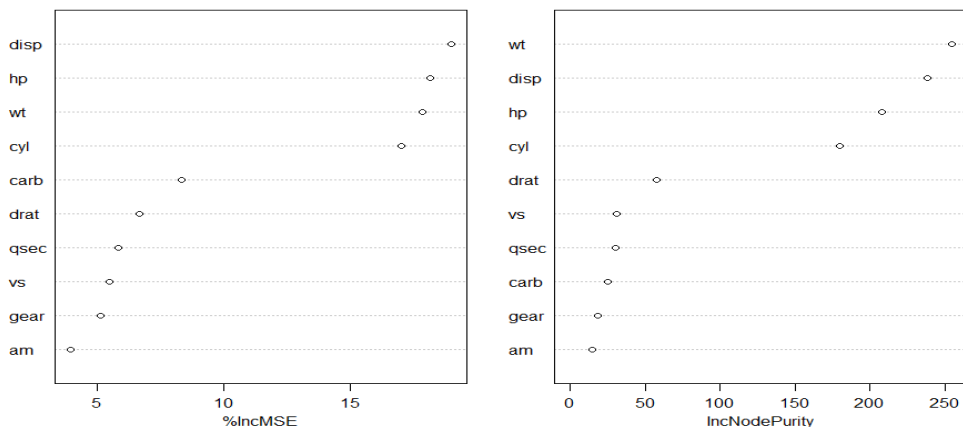
```
> set.seed(4543)
> data(mtcars)
> mtcars.rf <- randomForest(
  mpg ~ .,
  data=mtcars,
  ntree=1000,
  keep.forest=FALSE,
  importance=TRUE
)
> varImpPlot(mtcars.rf)
```

Index	Attribute Name	Description
[, 1]	mpg	Miles/(US) gallon
[, 2]	cyl	Number of cylinders
[, 3]	disp	Displacement (cu.in.)
[, 4]	hp	Gross horsepower
[, 5]	drat	Rear axle ratio
[, 6]	wt	Weight (lb/1000)
[, 7]	qsec	1/4 mile time
[, 8]	vs	V/S
[, 9]	am	Transmission (0 = automatic, 1 = manual)
[,10]	gear	Number of forward gears
[,11]	carb	Number of carburetors

data science dojo
unleash the data scientist in you

Important factors in determining car mileage

mtcars.rf



data science dojo
unleash the data scientist in you

Regression with random forests

```
> set.seed(131)
> ozone.rf <-
  randomForest(Ozone ~
    .,
    data=airquality,
    mtry=3,
    importance=TRUE,
    na.action=na.omit)
> print(ozone.rf)
#Impute Missing Values by median/mode.
> ozone.rf <-
  na.roughfix(ozone.rf)
```

	Ozone	Solar.R	Wind	Temp	Month	Day
1	41	190	7.4	67	5	1
2	36	118	8	72	5	2
3	12	149	12.6	74	5	3
4	18	313	11.5	62	5	4
5	NA	NA	14.3	56	5	5
6	28	NA	14.9	66	5	6
7	23	299	8.6	65	5	7
8	19	99	13.8	59	5	8
9	8	19	20.1	61	5	9
10	NA	194	8.6	69	5	10
11	7	NA	6.9	74	5	11
12	16	256	9.7	69	5	12
13	11	290	9.2	66	5	13
14	14	274	10.9	68	5	14
15	18	65	13.2	58	5	15

datasciencedojo
unleash the data scientist in you

Prediction

```
> predict(ozone.rf, data=airquality)
```

datasciencedojo
unleash the data scientist in you

RPART – Kyphosis Data

The kyphosis data frame has 81 rows and 4 columns representing data on children who have had corrective spinal surgery.

Kyphosis, a factor with levels absent and present indicating if a kyphosis (a type of deformation) was present after the operation.

Age in months.

Number of vertebrae involved.

Start the number of the first (topmost) vertebra operated on.

Kyphosis	Age	Number	Start
absent	71	3	5
absent	158	3	14
present	128	4	5
absent	2	5	1
absent	1	4	15
absent	1	2	16
absent	61	2	17
absent	37	3	16

datasciencedojo
unleash the data scientist in you

RPART

```
> install.packages('rpart')
```

```
> library(rpart)
```

datasciencedojo
unleash the data scientist in you

RPART

```
> fit <- rpart(Kyphosis ~ Age + Number + Start, data = kyphosis)
> fit2 <- rpart(Kyphosis ~ Age + Number + Start, data = kyphosis,
+   parms = list(prior = c(0.65, 0.35), split = "information"))
> fit3 <- rpart(Kyphosis ~ Age + Number + Start, data=kyphosis,
+   control = rpart.control(cp = 0.05))
> par(mfrow = c(1,2), xpd = TRUE)
> plot(fit)
> text(fit, use.n = TRUE)
> plot(fit2)
> text(fit2, use.n = TRUE)
```

data science dojo
unleash the data scientist in you

Resources

Random Forests Homepage

http://www.stat.berkeley.edu/~breiman/RandomForests/cc_home.htm

IRIS Data

IRIS data ships with R. You can learn more about IRIS data here:

<http://archive.ics.uci.edu/ml/datasets/Iris>

data science dojo
unleash the data scientist in you