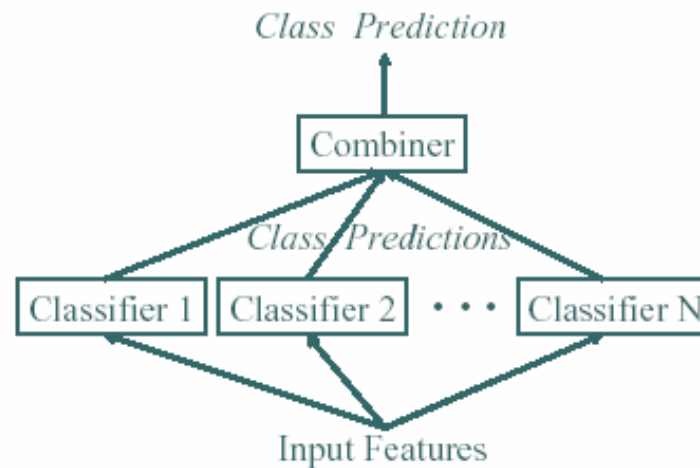


Chapter 12
Bagging and Random Forests

Xiaogang Su
Department of Statistics and Actuarial Science
University of Central Florida

Outline

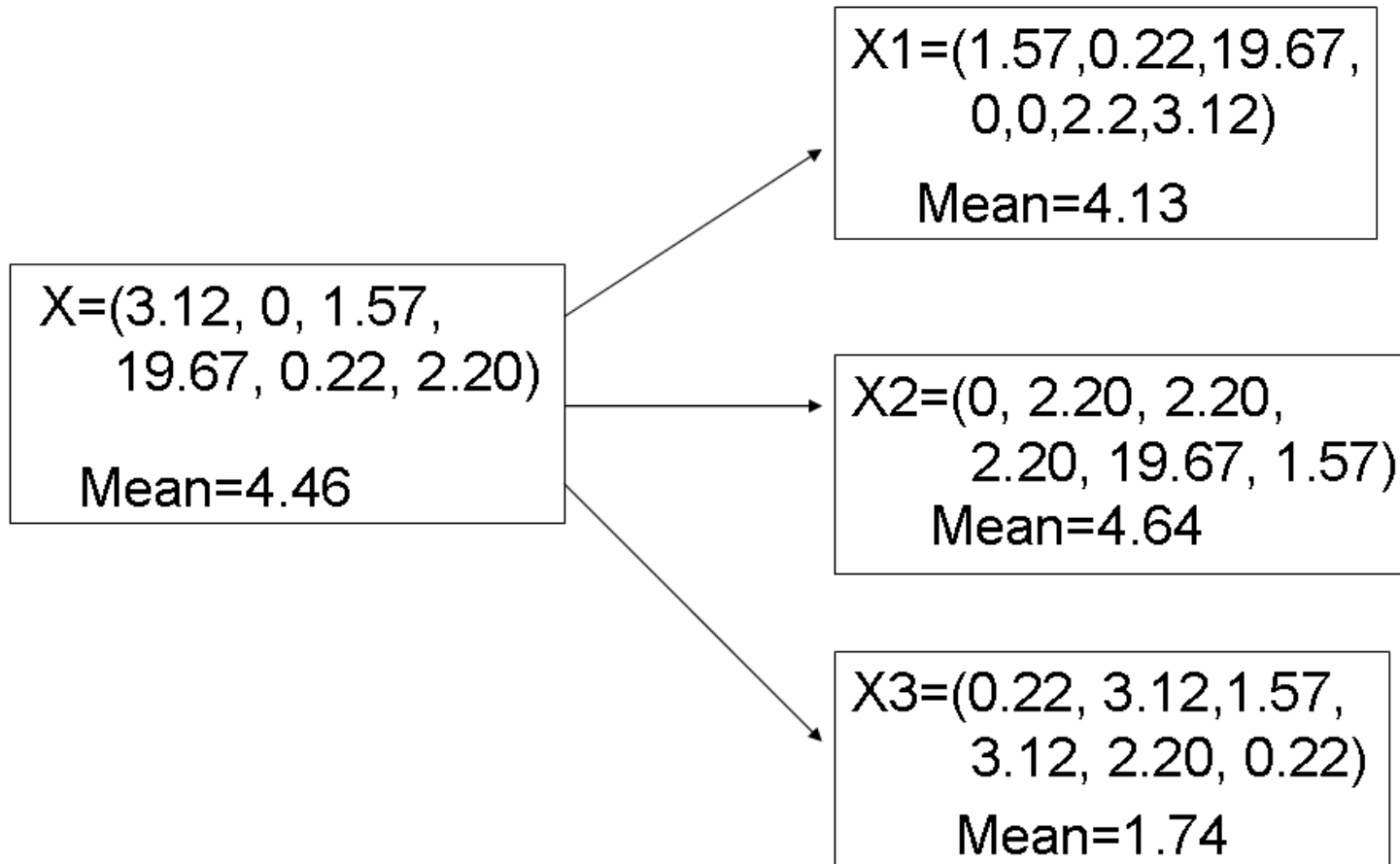
- A brief introduction to the bootstrap
- Bagging: basic concepts (Breiman, 1996)
- Random forest



Bootstrap - Motivation

- Problem: What's the average price of house prices?
 - From F , get a sample $L = (x_1, \dots, x_n)$, and calculate the average u .
 - Question: how reliable is u ? What's the standard error of u ? what's the confidence interval?
- Now a harder estimation problem: What is the IQR of the house prices with a confidence interval?
 - Repeat B time:
 - Generate a sample L_b of size n from L by sampling with replacement.
 - Compute IQR_b for L_b .
 - ➔ Now we end up with bootstrap values
 $(IQR_1, \dots, IQR_b, \dots, IQR_B)$
 - Use these values for calculating all the quantities of interest (e.g., standard deviation, confidence intervals)

Bootstrap – An Example on Estimating Mean



Bootstrap – An Overview

- Introduced by Bradley Efron in 1979. Named from the phrase “to pull oneself up by one’s bootstraps”, which is widely believed to come from “the Adventures of *Baron Munchausen’s*”.
- Popularized in 1980s due to the introduction of computers in statistical practice. It has a strong mathematical background. It is well known as a method for estimating standard errors, bias, and constructing confidence intervals for parameters.
- **Bootstrap Distribution:**
 - The bootstrap does not replace or add to the original data.
 - We use bootstrap distribution as a way to estimate the variation in a statistic based on the original data.
 - Sampling distribution vs. bootstrap distribution
 - Multiple samples → sampling distribution
 - Bootstrapping:
 - One original sample → B bootstrap samples
 - B bootstrap samples → bootstrap distribution

Bagging

- Introduced by Breiman (1996).
 - “Bagging” stands for “**bootstrap aggregating**”.
 - An ensemble method: a method of combining multiple predictors.
- **The Main Idea:**

Given a training set $D = \{(y_i, \underline{x}_i) : i = 1, 2, \dots, n\}$, want to make prediction for an observation with \underline{x} .

 - Sample B data sets, each consisting of n observations randomly selected from D with replacement. Then $\{D_1, D_2, \dots, D_B\}$ form B quasi replicated training sets.
 - Train a machine or model on each D_b for $b = 1, \dots, B$ and obtain a sequence of B predictions
 - The final aggregate classifier can be obtained by averaging (regression) or majority voting (classification).

Bagging – the Detailed Algorithm

- Construct B bootstrap replicates of L (e.g., $B = 200$): L_1, \dots, L_B
- Apply learning algorithm to each replicate L_b to obtain hypothesis or prediction h_b
- Let $T_b = L \setminus L_b$ be the data points that do not appear in L_b (out of bag points)
- Compute predicted value $h_b(\underline{x})$ for each \underline{x} in T_b
- For each data point \underline{x} , we will now have the observed corresponding value y and several predictions $\hat{y}_1(\underline{x}), \dots, \hat{y}_K(\underline{x})$. Compute the average prediction $\hat{h}(\underline{x})$.
- Estimate bias and variance
 - Bias $y - \hat{h}(\underline{x})$
 - Variance $\sum_{k=1}^K \left\{ \hat{y}_k(\underline{x}) - \hat{h}(\underline{x}) \right\}^2$

Variants of Re/Sub-Sampling Methods

- “Standard” bagging: each of the T subsamples has size n and created with replacement.
- “Sub-bagging”: create T subsamples of size α only ($\alpha < n$).
- No replacement: same as bagging or sub-bagging, but using sampling without replacement
- Overlap vs. non-overlap: Should the T subsamples overlap? i.e. create T subsamples each with T training data.
- Others Issues:
 - Permutation; Jackknife (Leave-One-Out); V-fold Cross-Validation
 - Out-Of-Bag Estimates of the Prediction/Classification Errors

Variance Reduction

- Bagging reduces variance (Intuition)
 - If each single classifier is unstable – that is, it has high variance, the aggregated classifier \bar{f} has a smaller variance than a single original classifier.
 - The aggregated classifier \bar{f} can be thought of as an approximation to the true average f obtained by replacing the probability distribution p with the bootstrap approximation to p obtained concentrating mass $1/n$ at each point (x_i, y_i) .
- Bagging works well for “unstable” learning algorithms. Bagging can slightly degrade the performance of “stable” learning algorithms.
 - Unstable learning algorithms: small changes in the training set result in large changes in predictions.
 - Neural network;
 - Decision trees and Regression trees;
 - Subset selection in linear/logistic regression
 - Stable learning algorithms:
 - K-nearest neighbors

Variance Reduction in Regression (Breiman)

- Data $(\underline{x}_i, y_i), i = 1, \dots, n$ are independently drawn from a distribution P .
- Consider the ideal the aggregate estimator $f_{ag}(\underline{x}) = E_P \widehat{f}^*(\underline{x})$. Here \underline{x} is fixed and the bootstrap dataset consists of observations $(\underline{x}_i^*, y_i^*), i = 1, \dots, n$ sampled from P (rather than from the current data). \rightarrow sampling distribution

- Then we have

$$\begin{aligned} E_P \left\{ y - \widehat{f}^*(x) \right\}^2 &= E_P \left\{ y - f_{ag}(x) + f_{ag}(x) - \widehat{f}^*(x) \right\}^2 \\ &= E_P \left\{ y - f_{ag}(x) \right\}^2 + E \left\{ f_{ag}(x) - \widehat{f}^*(x) \right\}^2 \\ &\geq E_P \left\{ y - f_{ag}(x) \right\}^2 \end{aligned}$$

- Therefore, true population aggregation never increases mean squared error for regression.
- The above argument does not hold for classification under 0-1 loss, because of the nonadditivity of bias and variance.

Bagging Can Hurt

- Bagging helps when a learning algorithm is good on average but unstable with respect to the training set. But if we bag a stable learning algorithm, we can actually make it worse.
- Bagging almost always helps with regression, but even with unstable learners it can hurt in classification. If we bag a poor and stable classifier we can make it horrible.

“The vital element is the instability of the prediction method. If perturbing the learning set can cause significant changes in the predictor constructed, then bagging can improve accuracy.” (Breiman, 1996)

- There are theoretical works that quantify the term “stability” precisely.

R Implementation – the `ipred` Package

```
ipredbagg.factor(y, X=NULL, nbagg=25, control=
  rpart.control(minsplit=2, cp=0, xval=0),
  comb=NULL, coob=FALSE, ns=length(y), keepX = TRUE, ...)
ipredbagg.numeric(y, X=NULL, nbagg=25, control=rpart.control(xval=0),
  comb=NULL, coob=FALSE, ns=length(y), keepX = TRUE, ...)
ipredbagg.Surv(y, X=NULL, nbagg=25, control=rpart.control(xval=0),
  comb=NULL, coob=FALSE, ns=dim(y)[1], keepX = TRUE, ...)
## S3 method for class 'data.frame':
bagging(formula, data, subset, na.action=na.rpart, ...)
```

OPTIONS:

`nbagg`: an integer giving the number of bootstrap replications.

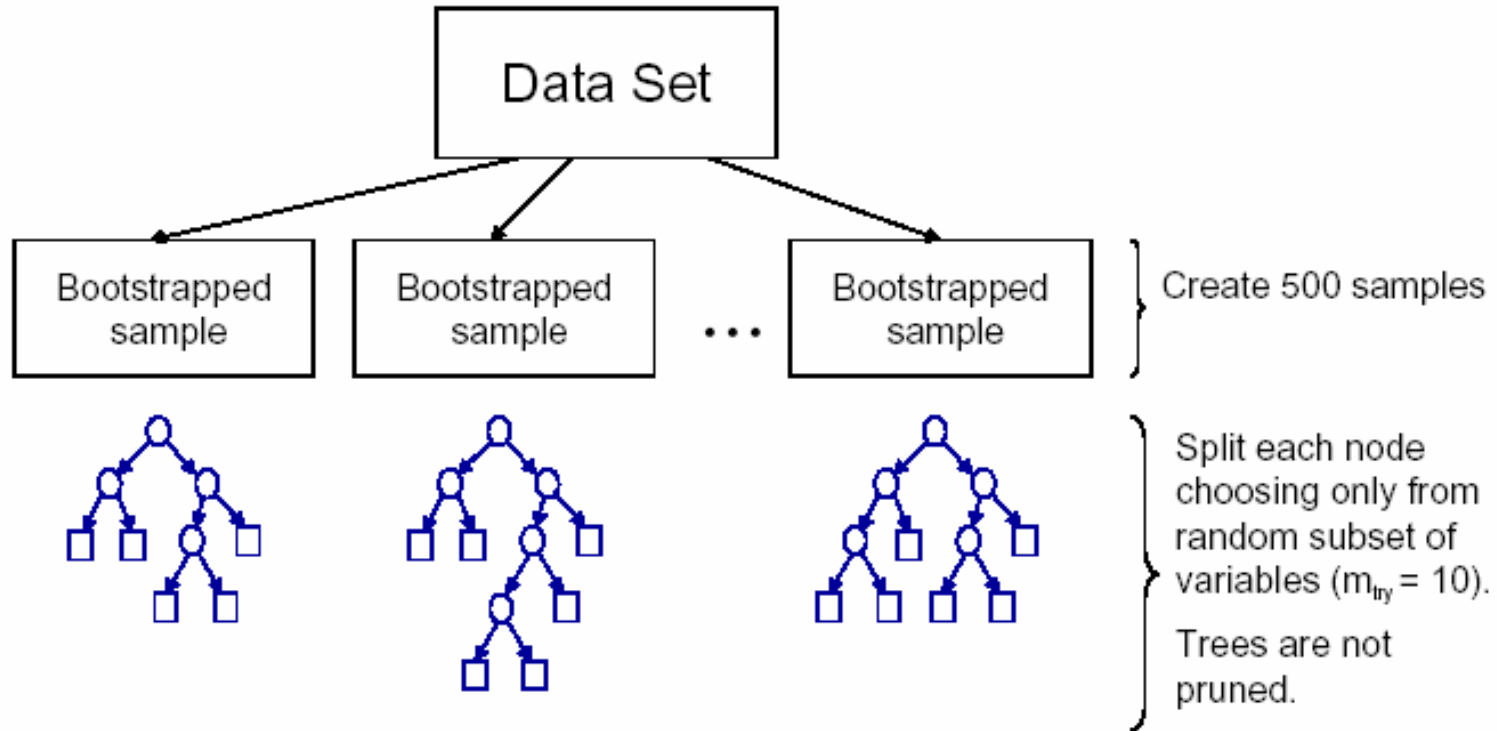
`coob`: a logical indicating whether an out-of-bag estimate of the error rate (misclassification error, root mean squared error or Brier score) should be computed. See `'predict.classbagg'` for details.

`control`: options that control details of the `'rpart'` algorithm, see `'rpart.control'`. It is wise to set `'xval = 0'` in order to save computing time. Note that the default values depend on the class of `'y'`.

Random Forests

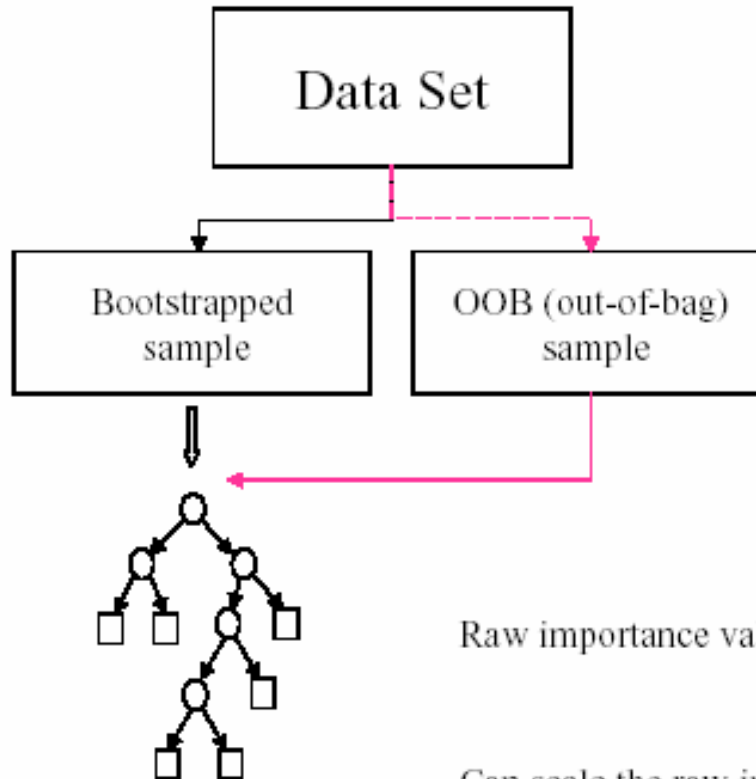
- Refinement of bagged trees; quite popular.
- The term came from **random decision forests** that was first proposed by Tin Kam Ho of Bell Labs in 1995. The method combines Breiman's "bagging" idea and Ho's "random subspace method" to construct a collection of decision trees with controlled variations.
- The Main Idea
 - At each tree split, a random sample of m features is drawn, and only those m features are considered for splitting. Typically $m = \sqrt{p}$ or $\log_2 p$, where p is the number of features
 - For each tree grown on a bootstrap sample, the error rate for observations left out of the bootstrap sample is monitored. This is called the “out-of-bag” error rate.
- Random forests tries to improve on bagging by “de-correlating” the trees. Each tree has the same expectation.

Steps in Random Forests



To classify new observation, use majority vote from the forest.

Out-Of-Bag Estimates and Variable Importance



1. Send observations in OOB data down the tree and count number of correct predictions ($= R_{OOB}$).

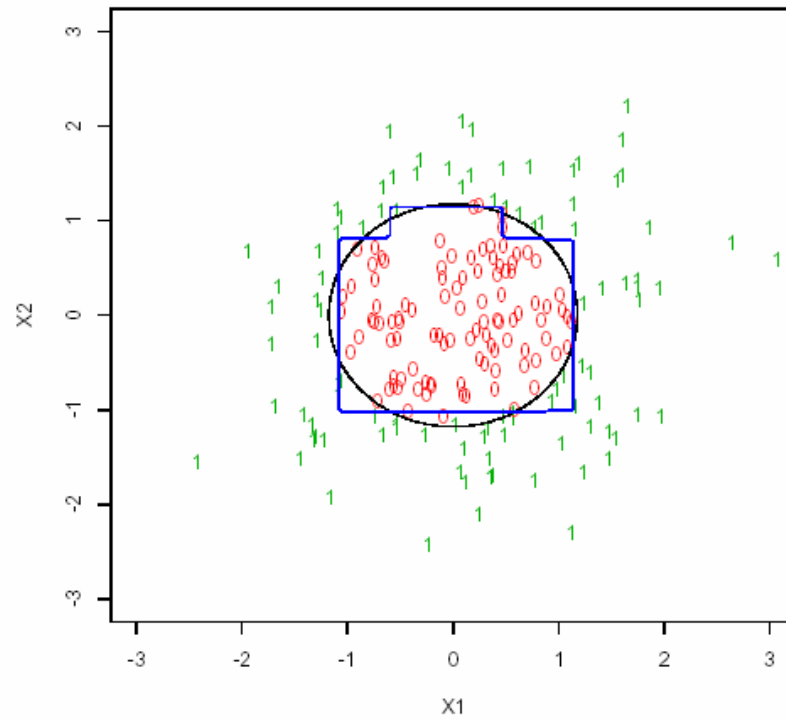
2. For the i -th predictor variable, randomly permute its values in the OOB cases. Send this data down the tree and count the correct predictions ($= R_{perm}$).

$$\text{Raw importance value} = \frac{1}{n_{tree \text{ all_trees}}} \sum (R_{OOB} - R_{perm})$$

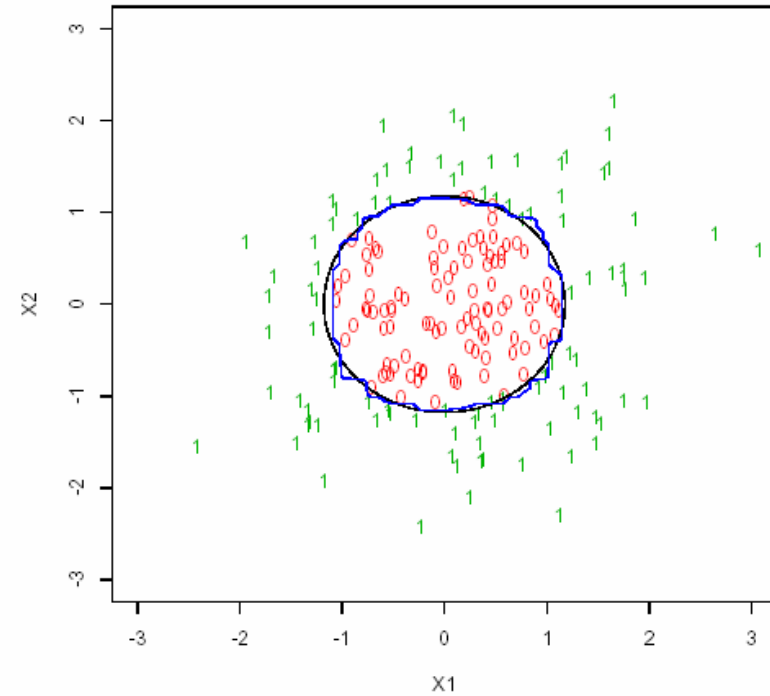
Can scale the raw imp value by the std.err. of $(R_{OOB} - R_{perm})$.
Scaled values are $\sim N(0,1)$ if trees are independent (?).

A Graph Illustrating the Decision Boundary

One Single Tree



Bagging/Random Forests



R Implementation: randomForest

<code>combine</code>	Combine Ensembles of Trees
<code>getTree</code>	Extract a single tree from a forest.
<code>grow</code>	Add trees to an ensemble
<code>importance</code>	Extract variable importance measure
<code>outlier</code>	Compute outlying measures
<code>partialPlot</code>	Partial dependence plot
<code>plot.randomForest</code>	Plot method for randomForest objects
<code>predict.randomForest</code>	predict method for random forest objects
<code>randomForest</code>	Classification and Regression with Random Forest
<code>rfImpute</code>	Missing Value Imputations by randomForest
<code>treesize</code>	Size of trees in an ensemble
<code>tuneRF</code>	Tune randomForest for the optimal mtry parameter
<code>varImpPlot</code>	Variable Importance Plot
<code>varUsed</code>	Variables used in a random forest

R Function randomForest

```
## S3 method for class 'formula':
randomForest(formula, data=NULL, ..., subset, na.action=na.fail)
## Default S3 method:
randomForest(x, y=NULL, xtest=NULL, ytest=NULL, ntree=500,
             mtry=if (!is.null(y) && !is.factor(y))
                 max(floor(ncol(x)/3), 1) else floor(sqrt(ncol(x))),
             replace=TRUE, classwt=NULL, cutoff, strata,
             sampsize = if (replace) nrow(x) else ceiling(.632*nrow(x)),
             nodesize = if (!is.null(y) && !is.factor(y)) 5 else 1,
             importance=FALSE, localImp=FALSE, nPerm=1,
             proximity=FALSE, oob.prox=proximity,
             norm.votes=TRUE, do.trace=FALSE,
             keep.forest=!is.null(y) && is.null(xtest), corr.bias=FALSE,
             keep.inbag=FALSE, ...)
```

For Unsupervised Learning with Random Forest:

<http://www.genetics.ucla.edu/labs/horvath/RFclustering/RFclustering.htm>

Options in Function `randomForest`

`ntree`: Number of trees to grow. This should not be set to too small a number, to ensure that every input row gets predicted at least a few times.

`mtry`: Number of variables randomly sampled as candidates at each split. Note that the default values are different for classification (\sqrt{p} where p is number of variables in 'x') and regression ($p/3$)

`replace`: Should sampling of cases be done with or without replacement?

`importance`: Should importance of predictors be assessed?

`localImp`: Should casewise importance measure be computed? (Setting this to 'TRUE' will override 'importance'.)

`nPerm`: Number of times the OOB data are permuted per tree for assessing variable importance. Number larger than 1 gives slightly more stable estimate, but not very effective. Currently only implemented for regression.

For Detailed Explanation and Examples:

http://oz.berkeley.edu/users/breiman/Using_random_forests_V3.1.pdf