Ensemble Methods

- In statistics and machine learning, ensemble methods use multiple learning algorithms to obtain better predictive performance than could be obtained from any of the constituent learning algorithms.
- Ensemble methods construct a set of classifiers and then classify new data points by taking a weighted *vote* of their predictions.

sources: https://en.wikipedia.org/wiki/Ensemble_learning and *Ensemble Methods in Machine Learning* by Thomas G□ Dietterich

Ensemble Methods

- Supervised learning algorithms are commonly described as performing the task of searching through a hypothesis space to find a suitable hypothesis that will make good predictions with a particular problem.
- Even if the hypothesis space contains hypotheses that are very well-suited for a particular problem, it may be very difficult to find (a good) one,
  - local minima in ANN’s
  - too expensive to fit a decision surface in a highly non-linear situation (SVM) – fitting an SVM with large values of C

Ensemble Methods

- Ensembles combine multiple hypotheses to form a (hopefully) better hypothesis.
- The term *ensemble* is usually reserved for methods that generate multiple hypotheses using the same base learner, *i.e.* decision tree

Note: The broader term of *multiple classifier systems* also covers hybridization of hypotheses that are not induced by the same base learner.

Ensemble Methods

- Evaluating the prediction of an ensemble typically requires more computation than evaluating the prediction of a single model
- Ensembles may be thought of as a way to compensate for poor learning algorithms by performing a lot of extra computation.

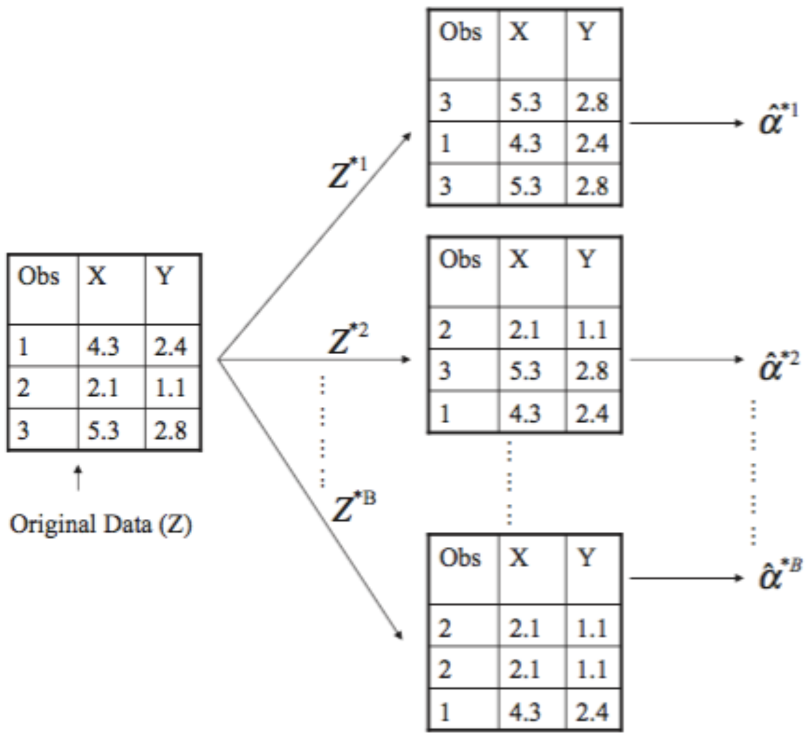
Ensemble Methods

- An ensemble is itself a supervised learning algorithm
  - it is trained on labeled data and then used to make predictions on unseen data points
- A trained ensemble represents a single hypothesis,
  - This hypothesis, however, is not necessarily contained within the hypothesis space of the models from which it is built.
  - Thus, ensembles can be shown to have more flexibility in the functions they can represent – we will see that with Random Forrests

Ensemble Techniques

- Bagged Trees
    - Bootstrap AGGregatED decision Trees
- Random Forests

Bagged Trees



Obs	X	Y
1	4.3	2.4
2	2.1	1.1
3	5.3	2.8

Obs	X	Y
3	5.3	2.8
1	4.3	2.4
3	5.3	2.8

Obs	X	Y
2	2.1	1.1
3	5.3	2.8
1	4.3	2.4

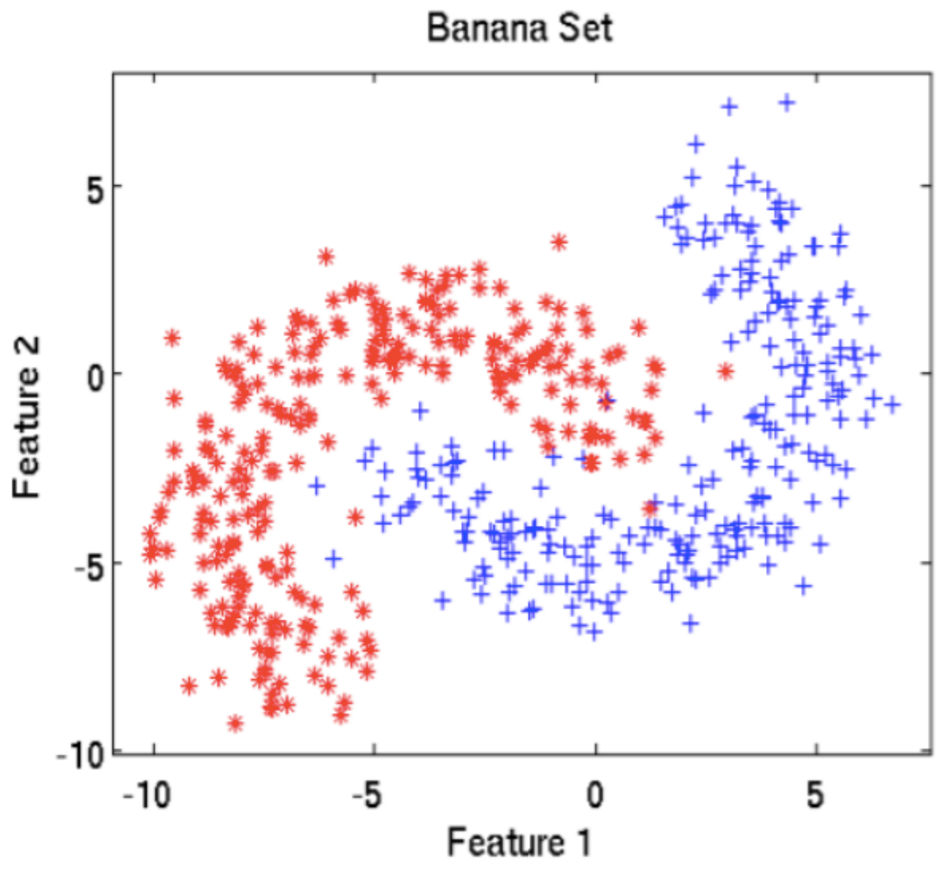
Obs	X	Y
2	2.1	1.1
2	2.1	1.1
1	4.3	2.4

- Recall the bootstrap
  - resample with replacement – *B bootstrap samples* - Z^{*k}
- Build *B* decision trees on the bootstrap samples - $\hat{\alpha}^{*k}$
- Given a point have each tree vote on the prediction,
  - The prediction with the most votes is the prediction of the ensemble.

Bagged Trees

- Notice that due to the resampling with replacement each bootstrap sample represents the input domain slightly differently.
- This means that bootstrapping might eliminate some irregularities from the original data that a single tree might have a difficult time learning/representing.

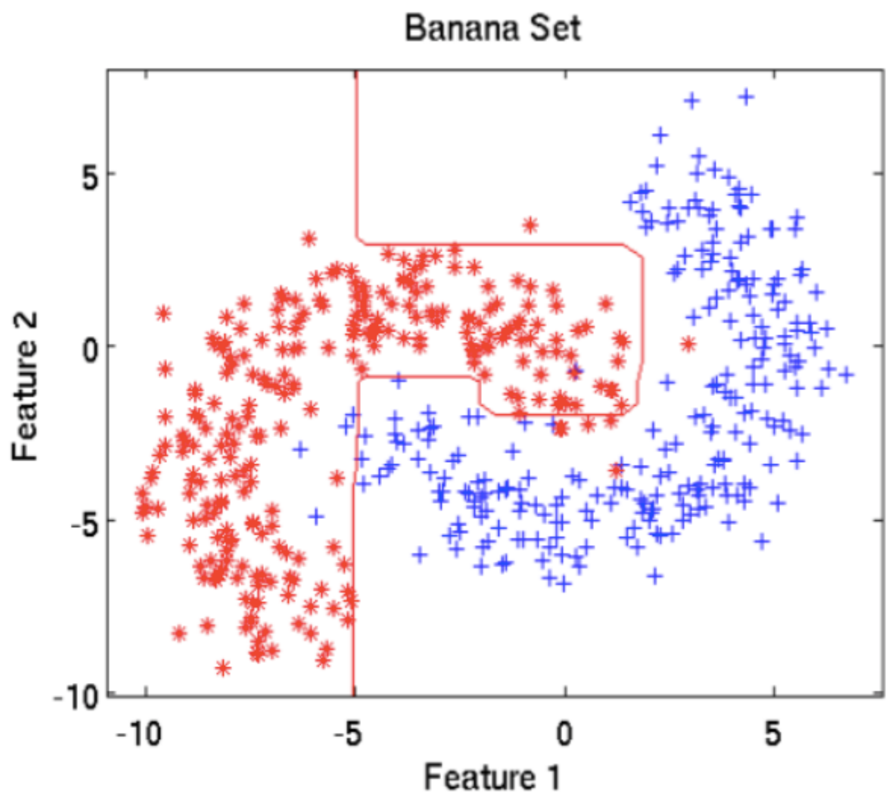
Bagged Trees



Training data

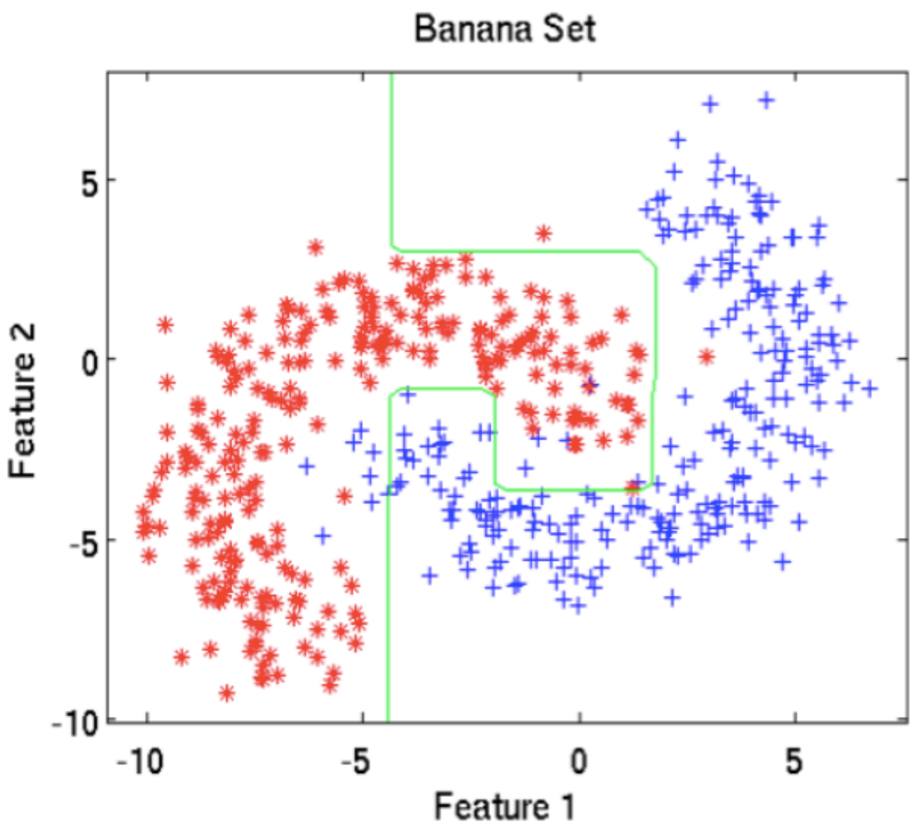
source: https://www.cs.rit.edu/~rlaz/prec20092/slides/Bagging_and_Boosting.pdf

Bagged Trees

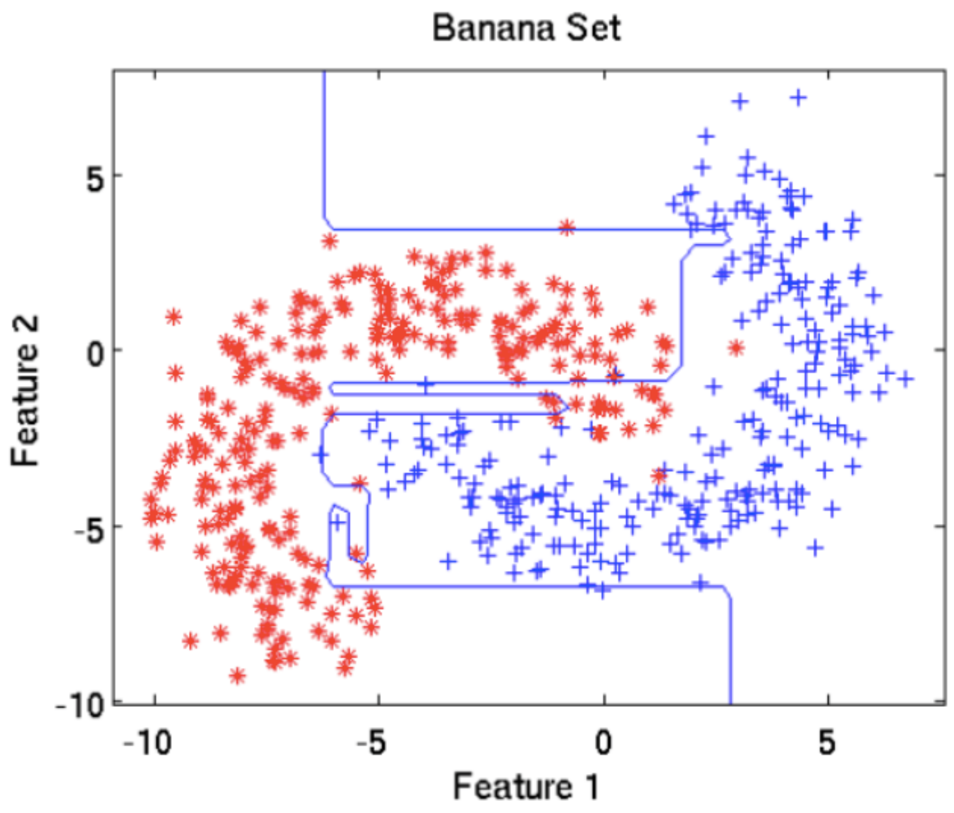


Decision boundary produced by one tree

Bagged Trees

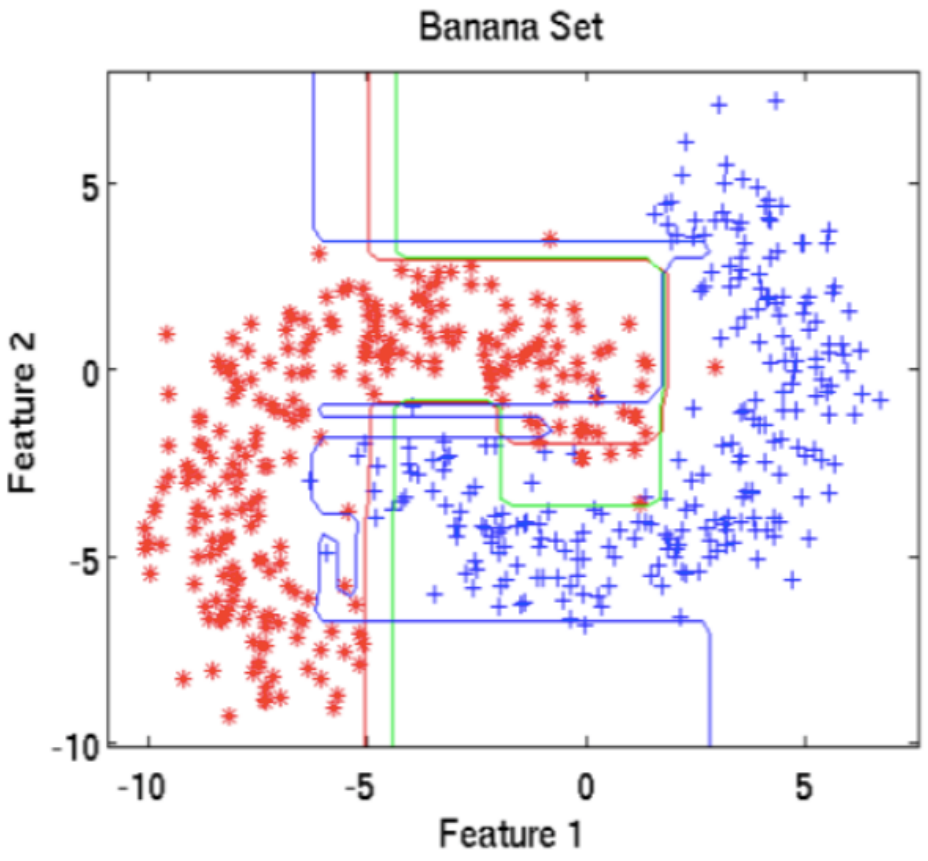


Decision boundary produced by a second tree

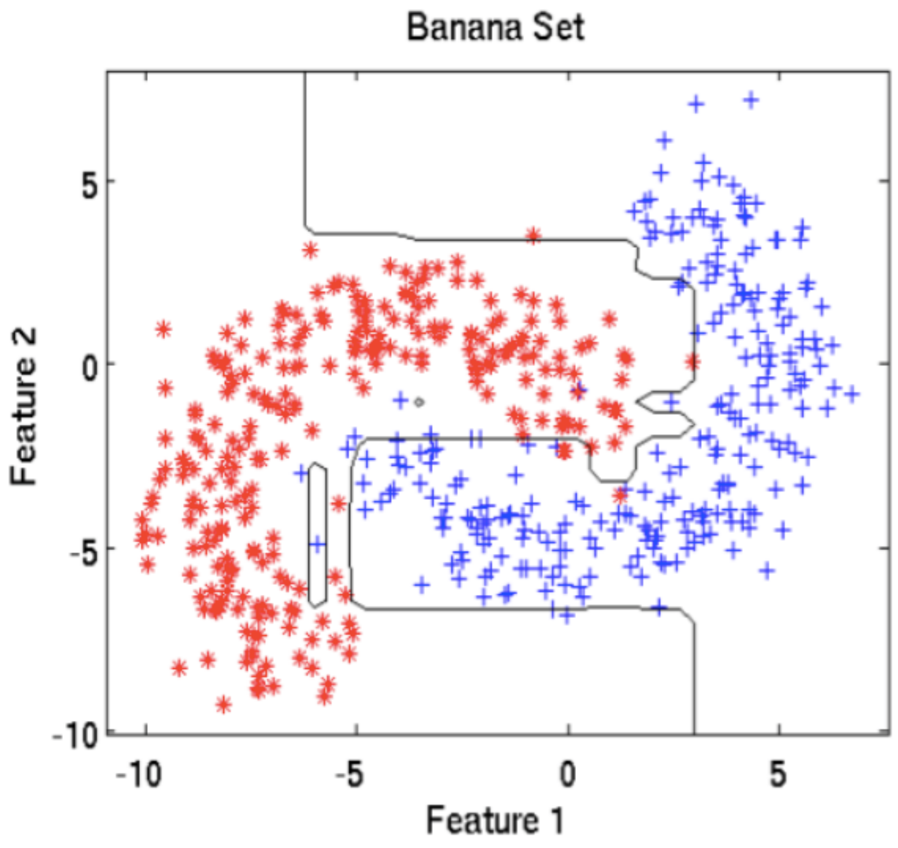


Decision boundary produced by a third tree

Bagged Trees



Three trees and final boundary overlaid



Final result from bagging all trees.

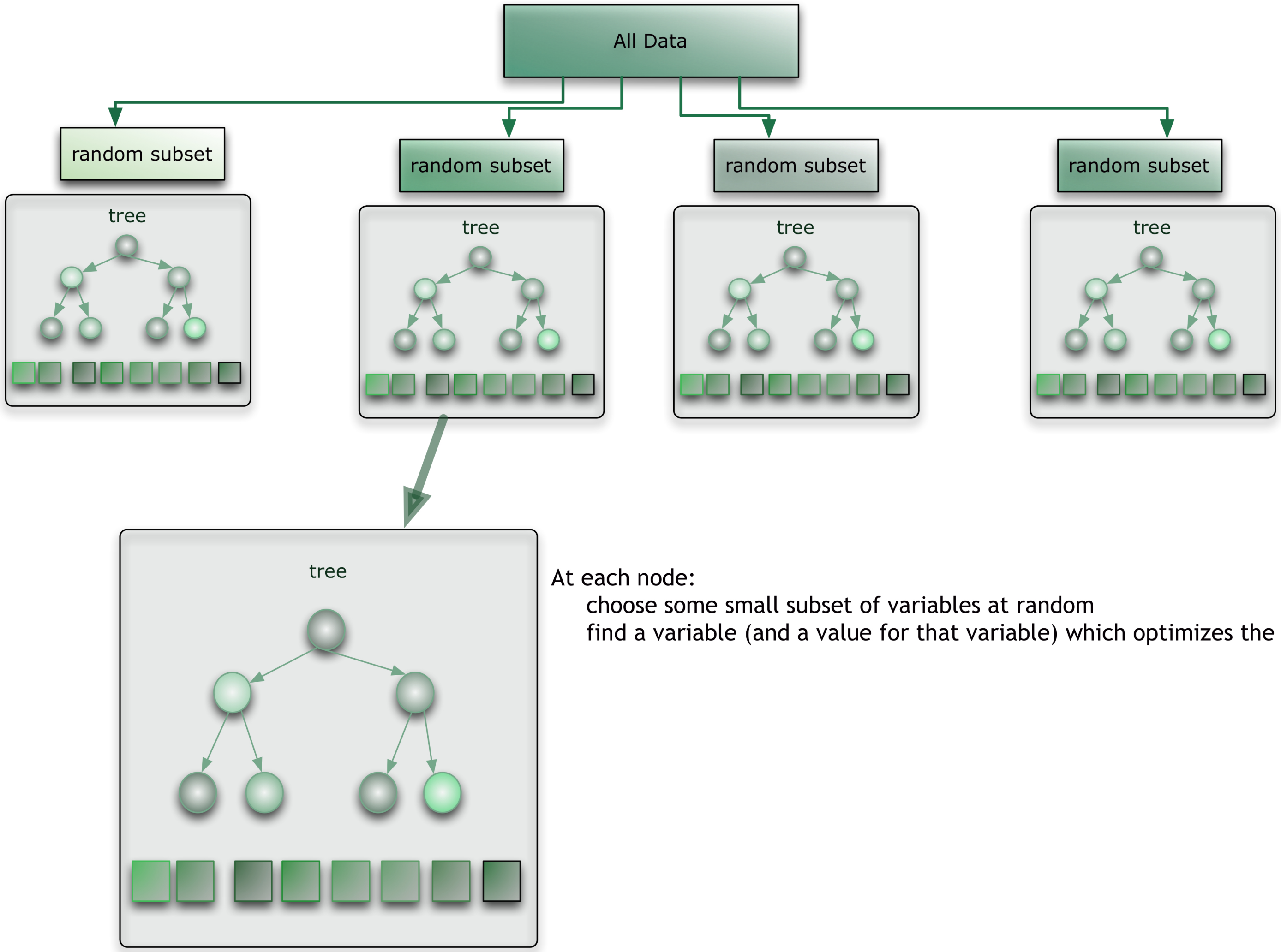
Random Forests

- Very similar to Bagged Trees
- One big difference: the attribute evaluated at a split is drawn from a *random* subset of all possible attributes

source: https://citizennet.com/blog/2012/11/10/random-forests-ensembles-and-performance-metrics/

Random Forests

-



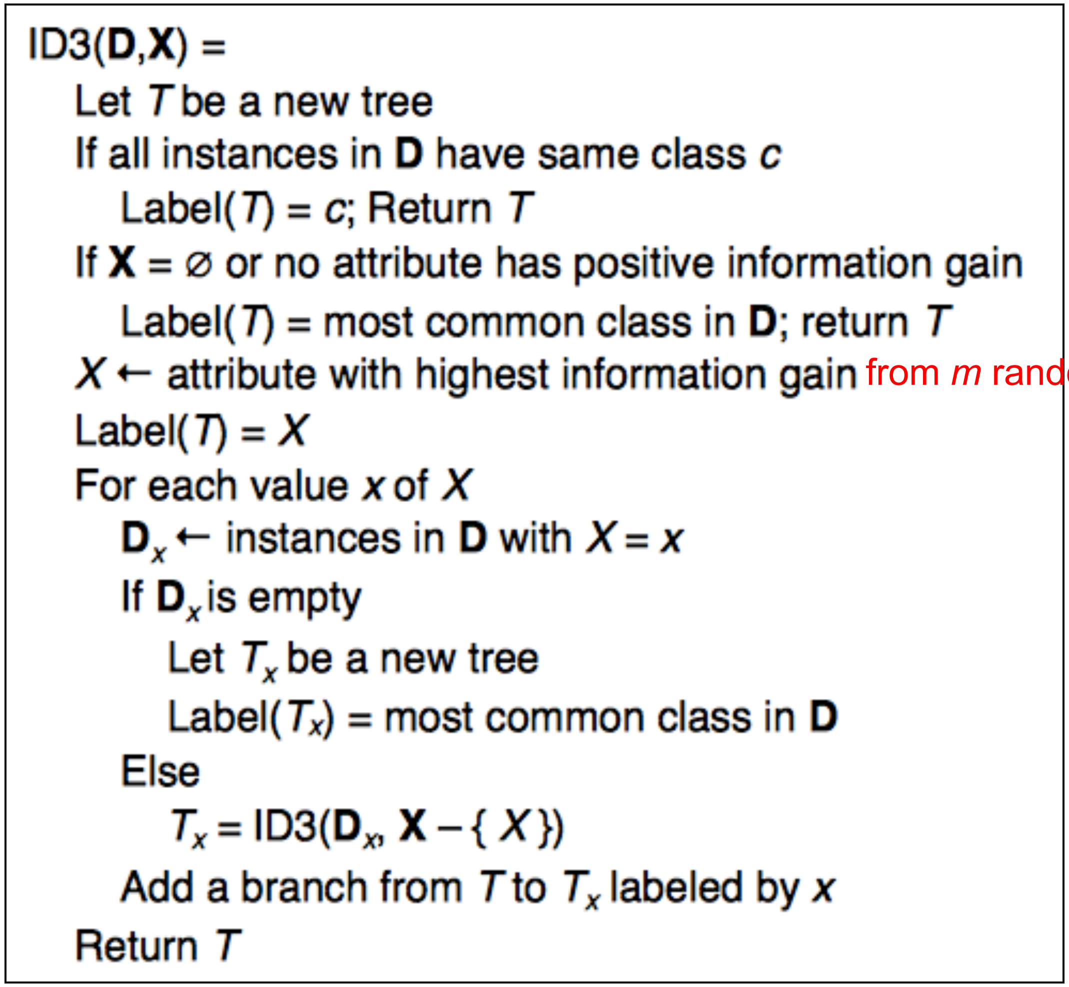
At each node:
choose some small subset of variables at random
find a variable (and a value for that variable) which optimizes the split

Random Forests

Each tree is grown as follows:

- Create a bootstrap sample from the original data. This sample will be the training set for growing the tree.
- For M input variables/attribute, choose a number m<<M,
  - *at each node*, *m variables are selected at random out of the M and the best split on these m is used to split the node*. The value of m is held constant during the forest growing.
- Each tree is grown to the largest extent possible. There is no pruning.

Random Forests



```
ID3($\mathbf{D}$,$\mathbf{X}$) =
    Let $T$ be a new tree
    If all instances in $\mathbf{D}$ have same class $c$
        Label($T$) = $c$; Return $T$
    If $\mathbf{X} = \varnothing$ or no attribute has positive information gain
        Label($T$) = most common class in $\mathbf{D}$; return $T$
    $X \leftarrow$ attribute with highest information gain from $m$ randomly selected attributes from $\mathbf{X}$
    Label($T$) = $X$
    For each value $x$ of $X$
        $\mathbf{D}_x \leftarrow$ instances in $\mathbf{D}$ with $X = x$
        If $\mathbf{D}_x$ is empty
            Let $T_x$ be a new tree
            Label($T_x$) = most common class in $\mathbf{D}$
        Else
            $T_x$ = ID3($\mathbf{D}_x$, $\mathbf{X} - \{ X \}$)
        Add a branch from $T$ to $T_x$ labeled by $x$
    Return $T$
```

- At top level generate bootstrap samples D
- After trees have been grown combine in a majority voting scheme