Random Forests

Erich Wellinger

Galvanize, Inc

Last updated: 20. September 2016

Objectives

Morning Objectives:

- Review
- Explain the construction of a Random Forest (classification or regression) algorithm
- Explain the relationship and difference between bagging and random forests
- Explain why random forests often perform better than a single decision tree

Afternoon Objectives:

- Explain how to get feature importances from a random forest and what those importances mean
- Explain OOB error is calculated and what it is an estimate of

Agenda

Morning Agenda:

- Review decision trees
- Discuss ensemble methods
- Discuss Bagging (bootstrap aggregation)
- Discuss Random Forests

Afternoon Agenda:

- Discuss Feature Importance
- Discuss Out-Of-Bag Error

Classification Trees

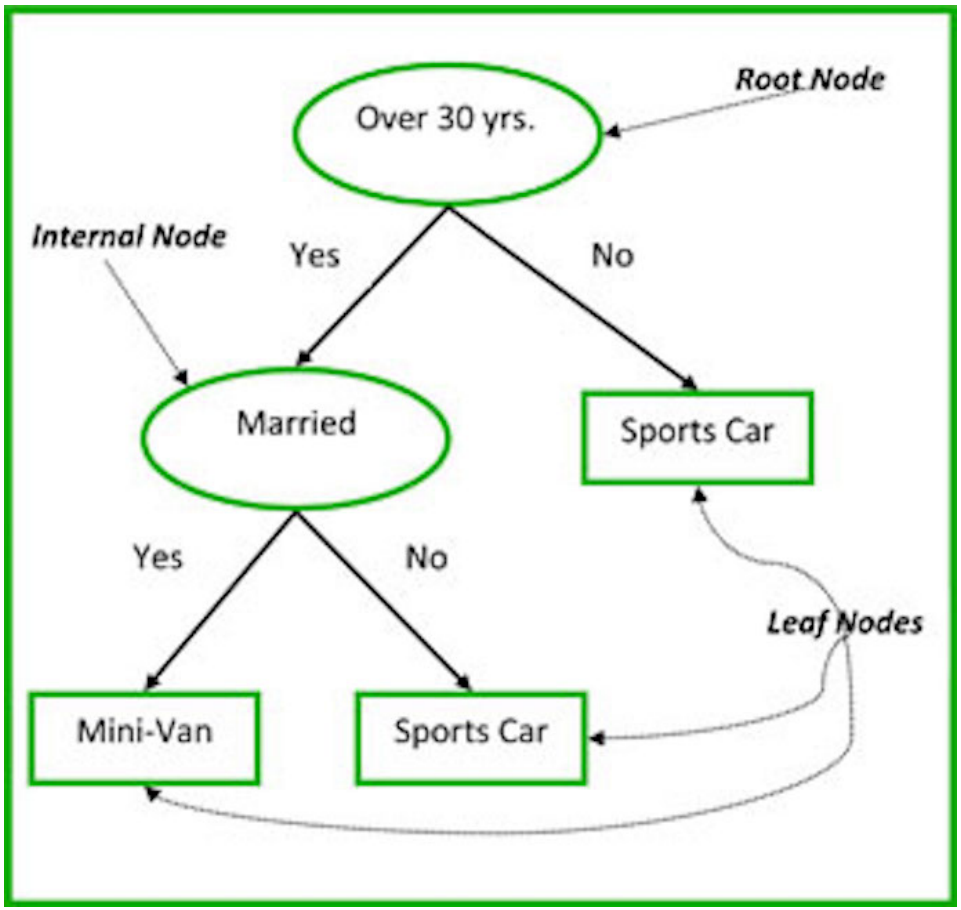
Training:

- Iteratively divide the nodes into subnodes such that (entropy/gini impurity) is minimized
- Various stopping conditions like a depth limit
- Prune trees by merging nodes

Inference:

- Take the most common class in the leaf node

Classification Trees



Regression Trees

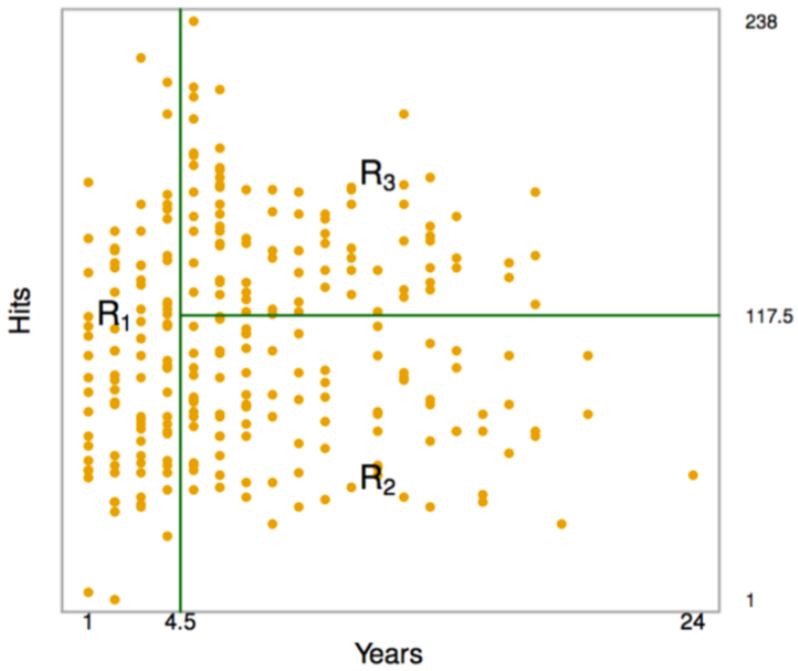
Similar to Classification Trees but...

- Instead of predicting a class label we're trying to predict a number
- We minimize *total squared error* instead of entropy or impurity

$$\sum_{j=1}^{J} \sum_{i \in R_j} (y_i - \hat{y}_{R_j})^2 \tag{1}$$

- Our prediction for any point that ends at that leaf node is simply the mean of the leaf node

Regression Trees



At each split, we aim to minimize:

$$\sum_{j=1}^{J} \sum_{i \in R_j} (y_i - \hat{y}_{R_j})^2$$

Regression Trees: Example

x_1	x_2	y
1	1	1
0	0	2
1	0	3
0	1	4

Prior to the split we guess the mean, 2.5, for everything, giving total squared error:

$$E = (1 - 2.5)^2 + (2 - 2.5)^2 + (3 - 2.5)^2 + (4 - 2.5)^2 = 5$$

After we split on x_1 we guess 2 for rows 1 & 3 and 3 for rows 2 & 4:

$$E = (1 - 2)^2 + (3 - 2)^2 + (2 - 3)^2 + (4 - 3)^2 = 4$$

Decision Tree Summary

Pros

- No feature scaling required
- Model nonlinear relationships
- Highly interpretable
- Can do both classification and regression
- Easily deal with continuous or categorical data*
- Easily handles missing values*

Cons

- Can be expensive to train
- Often poor predictors - high variance

* Not in Python

Agenda

Morning Agenda:

- ✓ Review decision trees
- Discuss ensemble methods
- Discuss Bagging (bootstrap aggregation)
- Discuss Random Forests

Afternoon Agenda:

- Discuss Feature Importance
- Discuss Out-Of-Bag Error

What is an Ensemble Method?

An **ensemble model** combines many weak learners to form a strong learner.

Train multiple different models on the data. To predict:

- For regressor, average the predictions of the models
- For classifier, take the plurality choice or average the percentages of each class

Ensembles: Intuition

Suppose we have 5 *independent* binary classifiers, each 70% accurate.
Overall accuracy:

$$\binom{5}{3}0.7^3 0.3^2 \times \binom{5}{4}0.7^4 0.3 \times \binom{5}{5}0.7^5 \approx 0.83$$

With 101 such classifiers we can achieve 99.9% accuracy.

- Why wouldn't this work?

Ensembles: Independence

Unfortunately, if all the learners are the same, creating an ensemble model won't help.

A solution to this is to train each learner on a different subset of the data. But how do we do this when we only have one set of data to work with?

Bootstrapping to the rescue!

Agenda

Morning Agenda:

- ✓ Review decision trees
- ✓ Discuss ensemble methods
  - Discuss Bagging (bootstrap aggregation)
  - Discuss Random Forests

Afternoon Agenda:

- Discuss Feature Importance
- Discuss Out-Of-Bag Error

Review: Bootstrapping

Questions:
What is a bootstrap sample?

What have we learned that bootstrap samples are good for so far?

Review: Bootstrapping

Questions:
What is a bootstrap sample?

- Given n data points we select a sample of n points *with replacement*

What have we learned that bootstrap samples are good for so far?

Review: Bootstrapping

Questions:
What is a bootstrap sample?

- Given n data points we select a sample of n points *with replacement*

What have we learned that bootstrap samples are good for so far?

- We use bootstrap samples to construct confidence intervals around sample statistics
- Ex: If I want a confidence interval around my sample median I could take 1,000 bootstrap samples and take the mean of all 1,000 samples. These means will follow a normal distribution.

Decision Trees & Variance

- Remember that one of the biggest downfalls of using Decision Trees are their inherently large variance, but this is something we can control!
- Recall that given a set of n independent observations $Z_1, ..., Z_n$, each with variance σ^2, the variance of the mean $\bar{Z}$ of the observations is given by σ^2/n
  - In other words, *averaging a set of observations reduces variance*
- If we would like to reduce the variance of our models predictions (and hence increase the prediction accuracy), we can take many training sets from the population (or bootstrap them!), build separate models for each set, and average the predictions from each

$$\hat{f}_{bag}(x) = \frac{1}{B}\sum_{b=1}^{B}\hat{f}^{*b}(x) \tag{2}$$

Bagging = Bootstrap, Fit, Repeat

- Bootstrap Aggregation, or bagging, is a general-purpose procedure for reducing the variance of a statistical learning method
- The resulting trees are grown deep and are not pruned. Hence each individual tree has high variance, but low bias
- Averaging these B trees reduces the variance significantly at the cost of a marginal increase in bias
- The number of trees B is not a critical parameter with bagging; using a very large B won't lead to overfitting
- In practice we want to use a large enough B such that our test error has settled down

Agenda

Morning Agenda:

- ✓ Review decision trees
- ✓ Discuss ensemble methods
- ✓ Discuss Bagging (bootstrap aggregation)
- Discuss Random Forests

Afternoon Agenda:

- Discuss Feature Importance
- Discuss Out-Of-Bag Error

Random Forests

- Random Forests provide an improvement over bagged trees by the way of a small tweak that serves to decorrelate the trees
- At each split in a given tree, a *random sample of m predictors* are chosen as candidates from the full set of p predictors.
  - Often $m = \sqrt{p}$
- This technique, called **subset sampling**, serves to prevent trees from always choosing the same splits
- As with bagging, random forests will not overfit if we increase B, so in practice we use a value of B sufficiently large for the error rate to have settled down

Random Forest Parameters

Random Forest Parameters:

- Total number of trees (`n_estimators`)
- Number of features to use at each split (`max_features`)
- Number of points for each bootstrap sample
- Individual decision tree parameters
  - e.g. tree depth, pruning (`min_samples_split`, `min_samples_leaf`, & c.)

In general, Random Forests are fairly robust to the choice of parameters and overfitting

Pros and Cons of Random Forests

Pros:

- Often give near state-of-the-art performance
- Good out-of-the-box performance
- No feature scaling needed
- Model nonlinear relationships

Cons:

- Can be expensive to train (though can be done in parallel)
- Models can be quite large (the pickled version of a several hundred tree model can easily be several GBs)
- Not interpretable (although techniques such as predicted value plots can help)

Random Forests in Scikit-Learn

The following is an example of how one can implement a Random Forest using scikit-learn:

```
1  from sklearn.ensemble import RandomForestRegressor
2  from sklearn.metrics import mean_squared_error
3  from sklearn.cross_validation import train_test_split
4
5  y = df.pop('target').values
6  X = df.values
7  X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.33, \
8                                                      random_state=42)
9
10 rf = RandomForestRegressor(n_estimators=100)
11 rf.fit(X_train, y_train)
12 y_pred = rf.predict(X_test)
13 print(mean_squared_error(y_test, y_pred))
```

Objectives

Morning Objectives:

- ✓ Review of Decision Trees

Objectives

Morning Objectives:

- ✓ Review of Decision Trees
- ✓ Explain the relationship and difference between bagging and random forests

Objectives

Morning Objectives:

- ✓ Review of Decision Trees
- ✓ Explain the relationship and difference between bagging and random forests
- ✓ Explain the construction of a Random Forest (classification or regression) algorithm

Objectives

Morning Objectives:

- ✓ Review of Decision Trees
- ✓ Explain the relationship and difference between bagging and random forests
- ✓ Explain the construction of a Random Forest (classification or regression) algorithm
- ✓ Explain why random forests often perform better than a single decision tree

Agenda

Morning Agenda:

- ✓ Review decision trees
- ✓ Discuss ensemble methods
- ✓ Discuss Bagging (bootstrap aggregation)
- ✓ Discuss Random Forests

Afternoon Agenda:

- Discuss Feature Importance
- Discuss Out-Of-Bag Error

Measuring Feature Importance

- When we bag a large number of trees, it is no longer possible to represent the resulting statistical learning procedure using a single tree and it is difficult to ascertain which features are most important
- There are several ways we can go about attempting to identify the most important features including:
  - Measuring the total amount that the RSS is decreased due to splits over a given predictor (regression trees).
  - Record what proportion of points pass through a split; the higher in the tree a feature is, the more important. Average this proportion across all trees.
  - When the B_i tree is grown, pass OOB through it. Permute or remove feature X_j and measure the change in accuracy (or other metric) before and after permuting/removing

Feature Importance in Scikit-Learn

When training a random forest model in Scikit-Learn there is an attribute called `model.feature_importances_` which stores normalized importances for each of our models features

- ̈It is sometimes called ́gini importance' or ́mean decrease impurity' and is defined as the total decrease in node impurity (weighted by the probability of reaching that node) averaged over all trees of the ensemble."

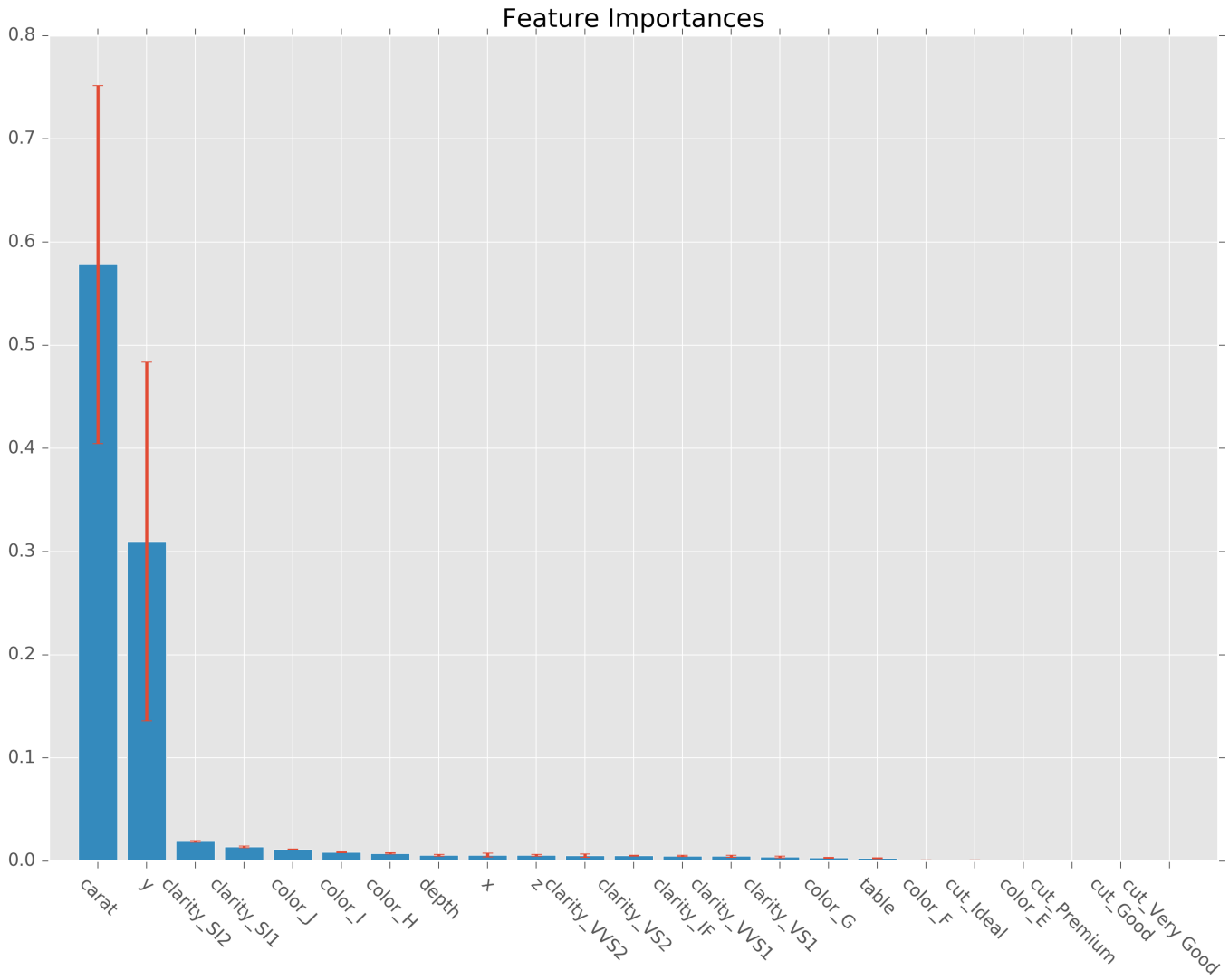
Computing Feature Importance On Single Tree

For some intuition, this is how this is calculated on a tree by tree basis and then averaged across all the trees in our model

1. Initialize array `feature_importances` of all zeros with size `n_features`
2. Traverse the tree
   - For each internal node that splits on feature `i`, compute the error reduction of that node multiplied by the number of samples that were routed to the node
   - Add this quantity to `feature_importances[i]`

The error reduction depends on the impurity criterion that you use (e.g. Gini, Entroy, RSS). The result is, by default, normalized in sklearn.

Example Feature Importance Plot

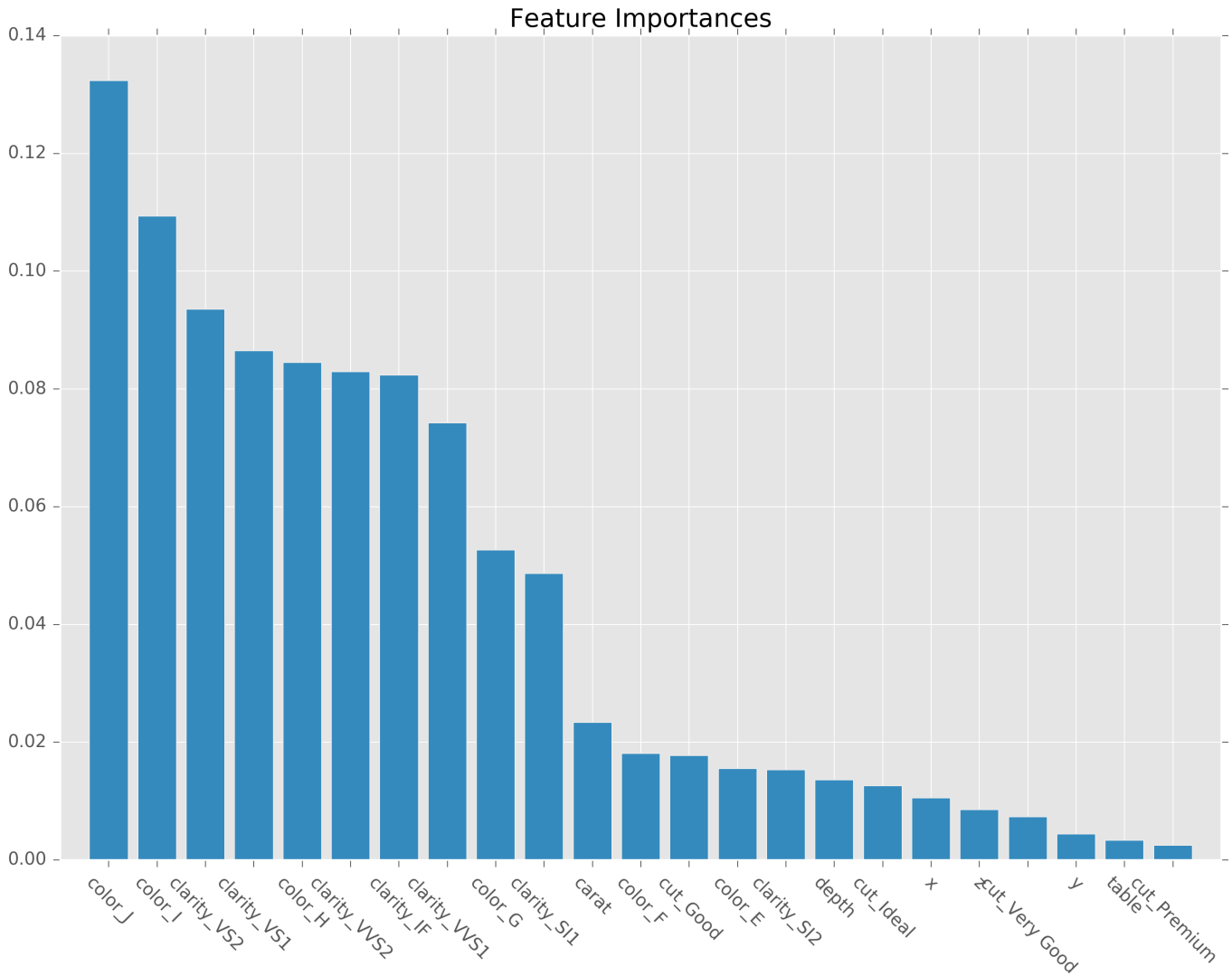


Leave One Out Feature Importance

There are, of course, other ways we might choose to measure how important a feature is.

- One way is to iteratively drop each feature, train a new model, and compare the MSE before and after (in the regression case)
- A feature which wasn't important will have a low effect on the model before and after removal

Leave One Out Feature Importance



Stop and Think

Why do you think these measures of importance were so different?

Stop and Think

Why do you think these measures of importance were so different?

- One possible explaination is that features that were deemed to be important in our initial feature importance plot can be represented by other features in our dataset. For example, `carat` (a measure of weight) can likely be accurately represented by `x`, `y`, `z` (length, width, and depth respectively)

Stop and Think

Why do you think these measures of importance were so different?

- One possible explaination is that features that were deemed to be important in our initial feature importance plot can be represented by other features in our dataset. For example, `carat` (a measure of weight) can likely be accurately represented by `x`, `y`, `z` (length, width, and depth respectively)
- This also points out a crucial nuance to Scikit-Learn's measure of feature importance: continuous features and categorical variables with many levels are often deemed more important than categorical variables with fewer levels

Agenda

Morning Agenda:

- ✓ Review decision trees
- ✓ Discuss ensemble methods
- ✓ Discuss Bagging (bootstrap aggregation)
- ✓ Discuss Random Forests

Afternoon Agenda:

- ✓ Discuss Feature Importance
- Discuss Out-Of-Bag Error

Out-Of-Bag Error Estimation

- One nice side effect from using bootstrapped models is we can easily estimate the test error of a bagged model without the need to perform cross-validation!
- Each tree in our forest has only seen 66% of our training data
- We can thus measure the test error of a particular tree by running the remaining *out-of-bag* (OOB)observations through the tree

```
1  rf = RandomForestRegressor(n_estimators=20, oob_score=True)
2  rf.fit(X, y)
3
4  print rf.oob_score_
```

Objectives

Afternoon Objectives:

- ✓ Explain how to get feature importances from a random forest and what those importances mean

Objectives

Afternoon Objectives:

- ✓ Explain how to get feature importances from a random forest and what those importances mean
- ✓ Explain OOB error is calculated and what it is an estimate of