

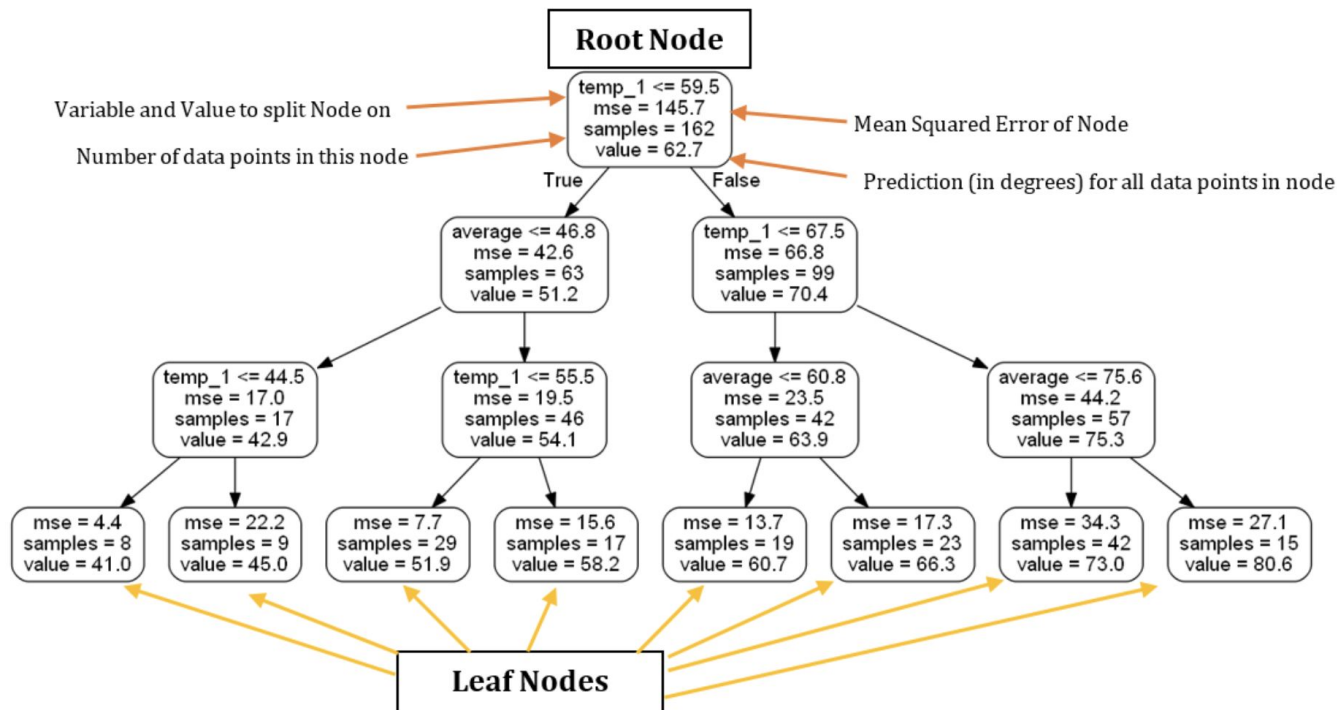
Random Forests

Data Science Immersive

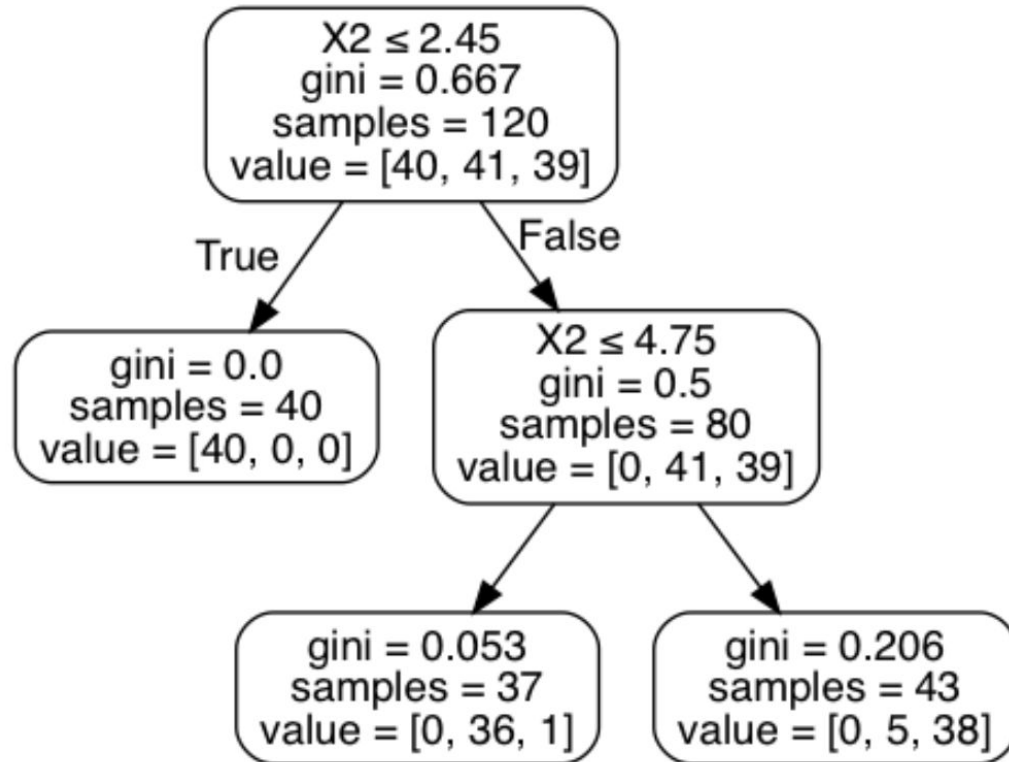
Agenda today

- Decision Trees Review
- Learn Bootstrapping and Aggregating (Bagging) to Decision Trees
- Learn the Random Forests algorithm

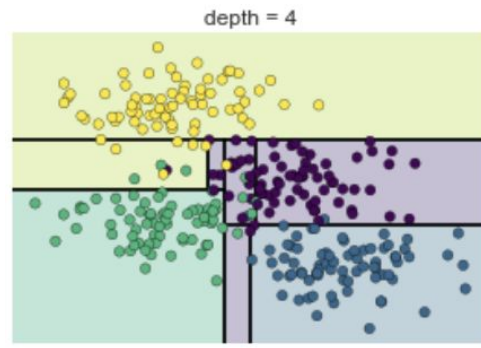
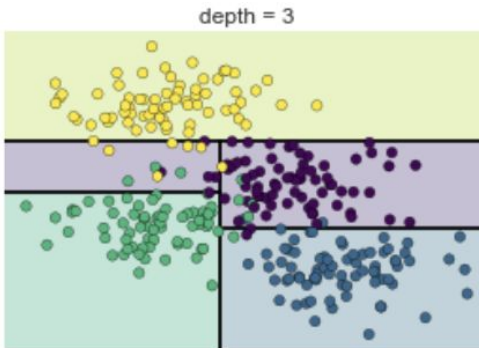
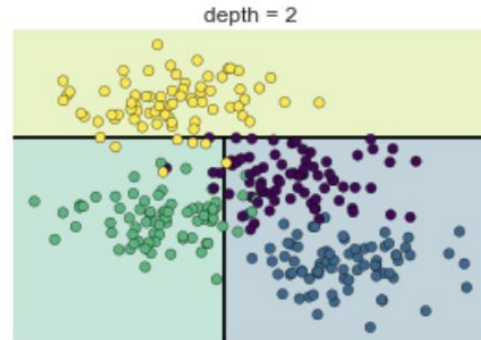
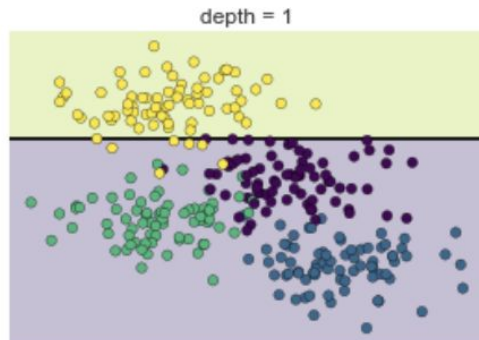
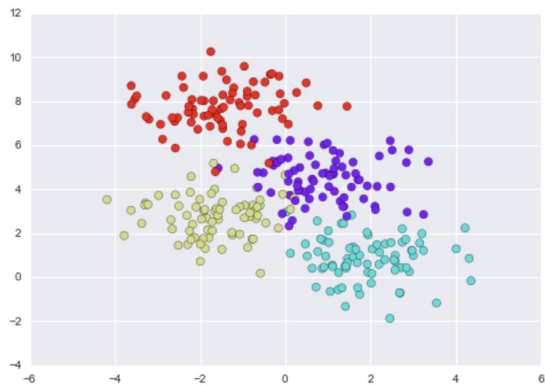
Quick Review of Regression Trees



Quick Review of Classification Trees

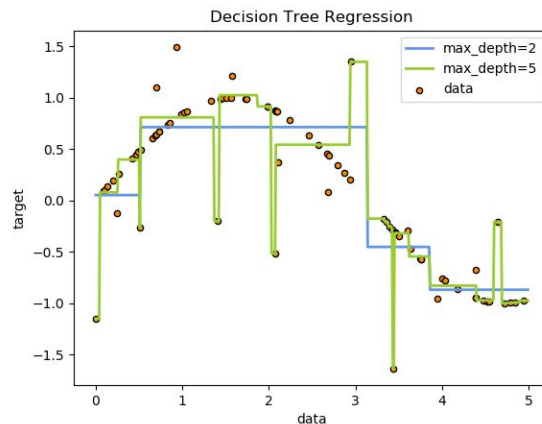
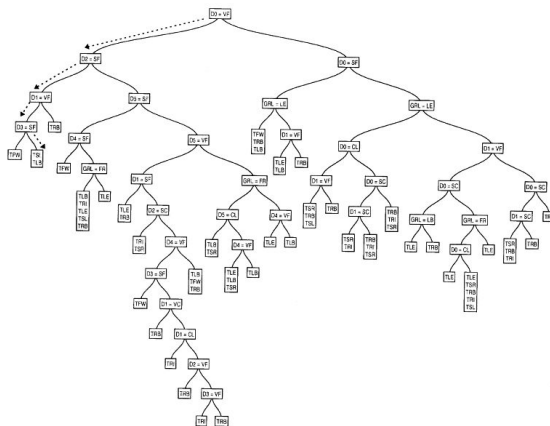
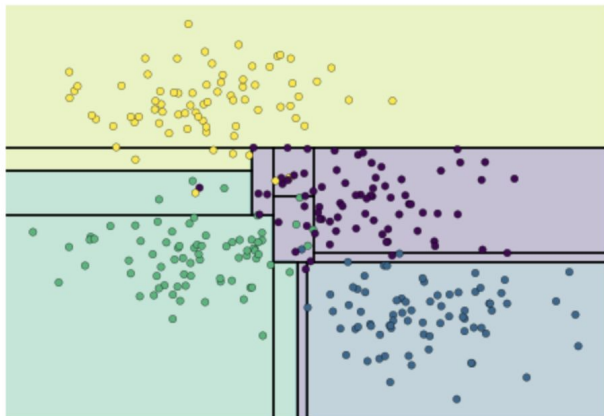


Visualizing Classification Decision Trees



Tree Depth and Overfitting

The deeper our trees go, the higher the chance that we overfit our data



Trees are pretty great :)



Decision Tree Review

Advantages of Decision Trees:

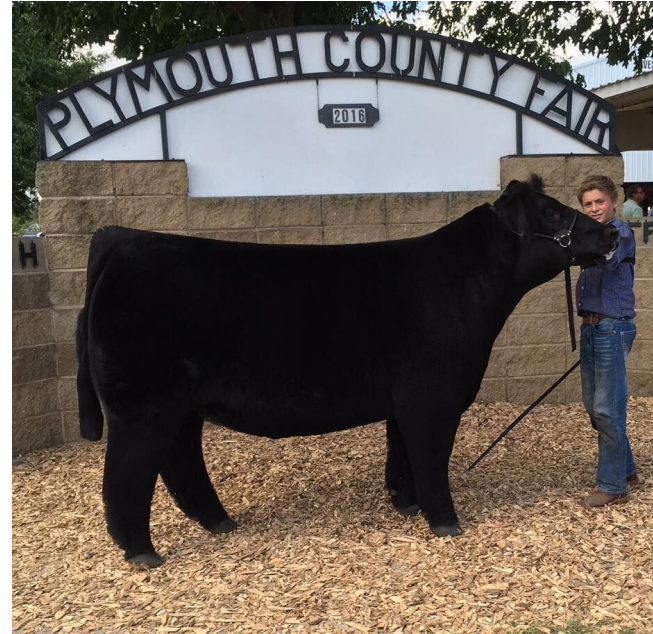
- Work well with non-linear relationships
- Easy to interpret
- Implicit feature selection
- Account for interaction

Disadvantages of Decision Trees:

- Decision trees tend to overfit, especially if they are completely pure.
- Instability: Small changes in the input data can cause large changes to the structure of the tree.
- “Trees have one aspect that prevents them from being the ideal tool for predictive modeling, namely inaccuracy” -- Elements of Statistical Learning

Wisdom of the Crowd - expert or the crowd?

At a 1906 fair in Plymouth, England, statistician Francis Galton noticed how when 800 people guessed how much a “dressed” ox weighed. It turns out that the actual weight had only a 1% error from the median guess

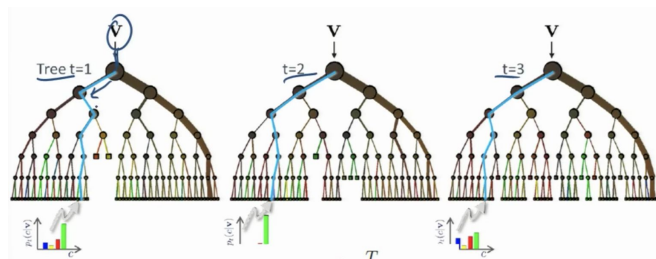


Takeaway

We can learn a lot by combining many different learners. This is called an *ensemble method*.

In essence, we can combine the output of models that are not accurate enough on their own. We can get wisdom from the crowd!

How can we grow a forest of trees?



Bagged Tree: Recipe

To construct a Bagged Tree estimator, what we need is:

- 1) **Bootstrap** the entire dataset
- 2) Build a tree using a **the bootstrapped dataset**
- 3) Repeat step 1 and 2 many, many, many times, and **aggregate** all the trees
- 4) Output prediction through each tree
- 5)
 - For regression, take the average of the prediction
 - For classification, take the majority predicted value

Step 1: Bootstrapping

- Trees are prone to overfitting. (They have a high variance). This is especially true if the trees are built out to full “purity” in each of the leaves.
- To help prevent overfitting we take bootstrap samples **with replacement** from our training data that is the same size as our training data.



original sample

1 2 3 4 5 6 7 8 9 bootstrap sample 1

1 2 3 4 5 6 7 8 9 bootstrap sample 2

1 2 3 4 5 6 7 8 9 bootstrap sample 3

Step 1: Bootstrapping

Original Dataset

	Chest Pain	Good Blood Circ.	Blocked Arteries	Weight	Heart Disease
1	No	No	No	125	No
2	Yes	Yes	Yes	180	Yes
3	Yes	Yes	No	210	No
4	Yes	No	Yes	167	Yes

Bootstrapped Dataset

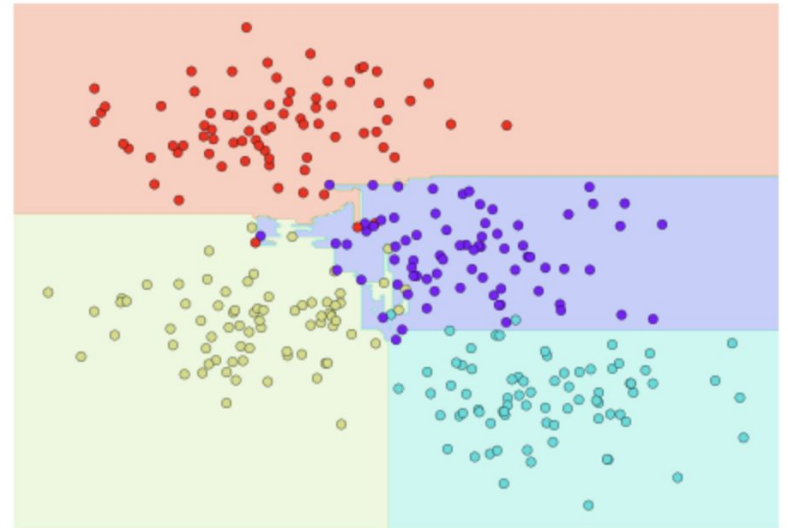
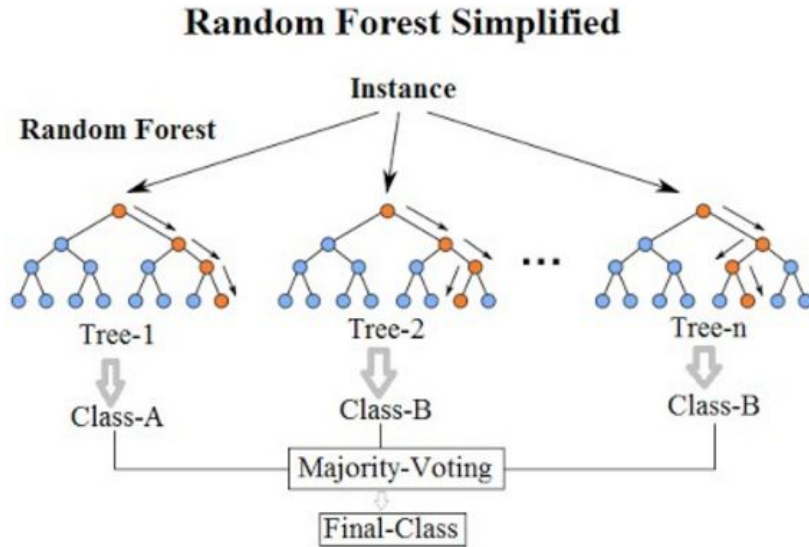
	Chest Pain	Good Blood Circ.	Blocked Arteries	Weight	Heart Disease
2	Yes	Yes	Yes	180	Yes
1	No	No	No	125	No
4	Yes	No	Yes	167	Yes
4	Yes	No	Yes	167	Yes

Step 2: Aggregating

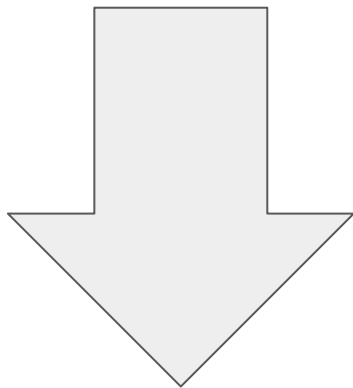
- Once we have taken bootstrapped samples, we fit a decision tree to each sample. This tree will have a **low bias and high variance**
- Repeat this process for however many trees you want in your model
- Now, we can feed data through all of the bootstrapped trees and take
 - Classification: whichever class is predicted most by the bootstrapped decision trees
 - Regression: take an average of the predicted values for each decision tree

Voting Method

Majority Class Labels (Majority/Hard Voting)

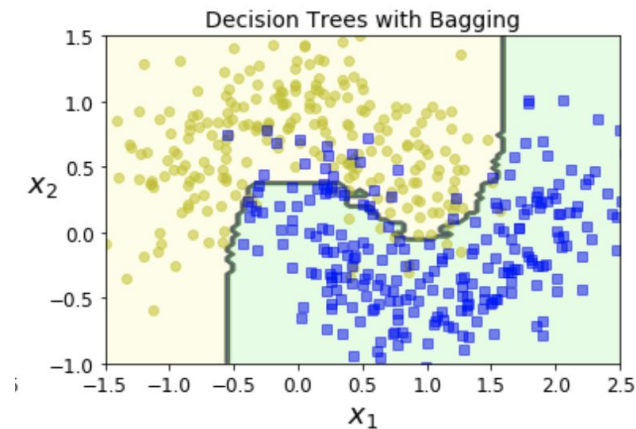
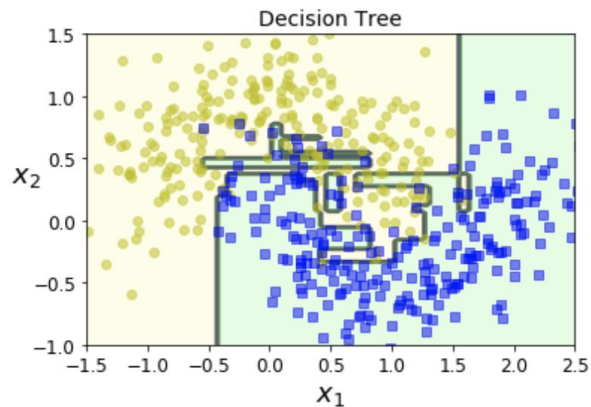


Bootstrapping + Aggregating



Bagging

Decision Tree v. Decision Trees with Bagging

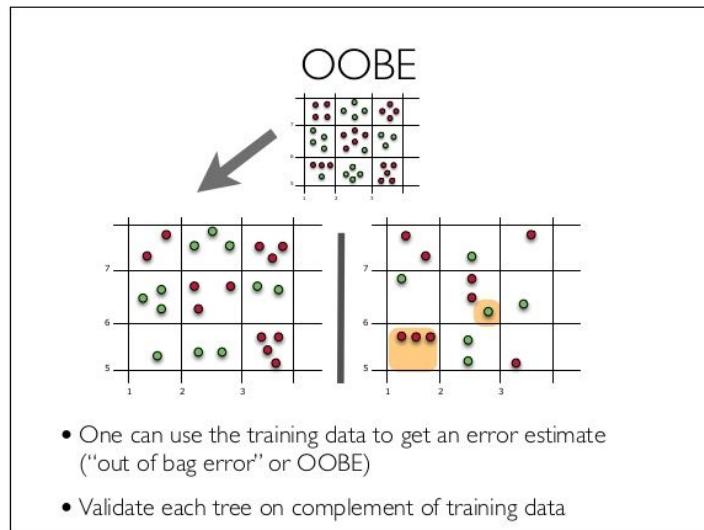


Out-of-Bag Error

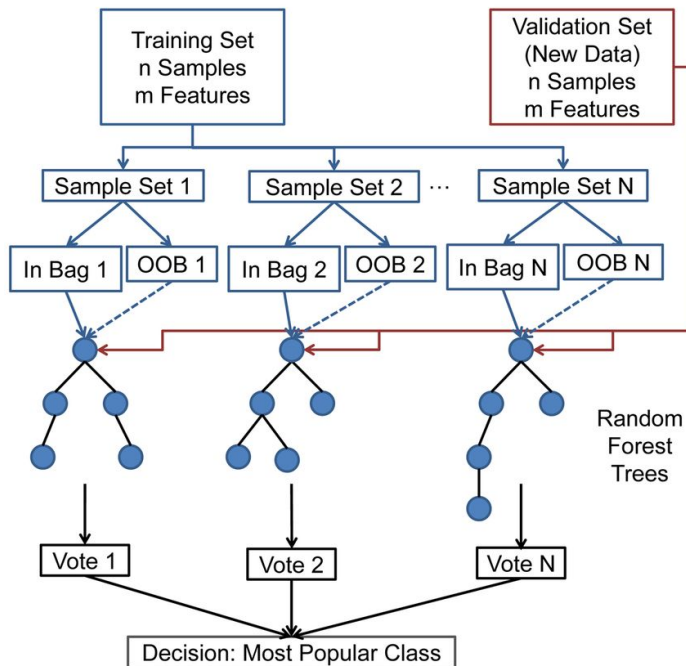
In general only $\frac{2}{3}$ of the observations will be trained on each model. ([Why 2/3?](#))

Performing cross validation on bagged classifiers/regressors is challenging because it can be computationally expensive.

Rather, we can look at every observation and make a prediction for each of the data points for which that data points was not used to create the tree.



Out-of-Bag Error



It's essentially
just a form of
cross-validation!

The Issue with Bagging

The issue with bagging is that each one of the trees might be correlated to each other. There might be a feature that is powerful in generating a separation between different categories, which results in trees that are correlated to one another despite being from bootstrapped samples.

We need to do something to ensure that the bootstrapped samples are not correlated with one another.....



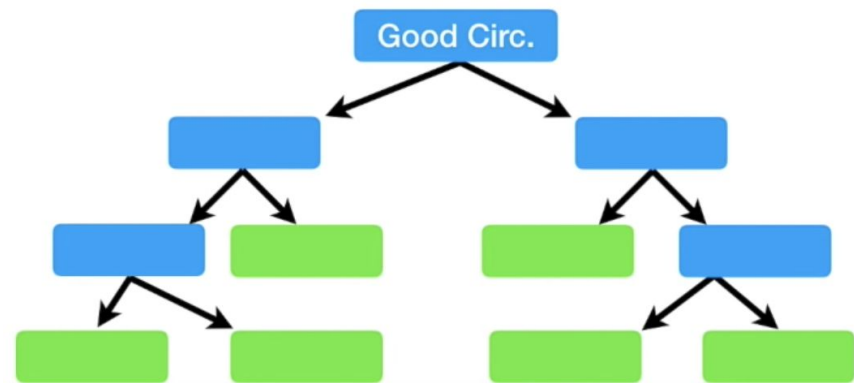
Random Forest: Recipe

To construct a Random Forest estimator, what we need is:

- 1) **Bootstrap** the entire dataset
- 2) Build a tree using only a **random subset of the features at each node** from bootstrapped dataset
- 3) Repeat step 1 and 2 many, many, many times, and **aggregate** all the trees
- 4) Output prediction through each tree
- 5) - For regression, take the average of the prediction
- For classification, take the majority predicted value

Random Forest: Recipe

Step 2--selecting a subset of the feature at each node



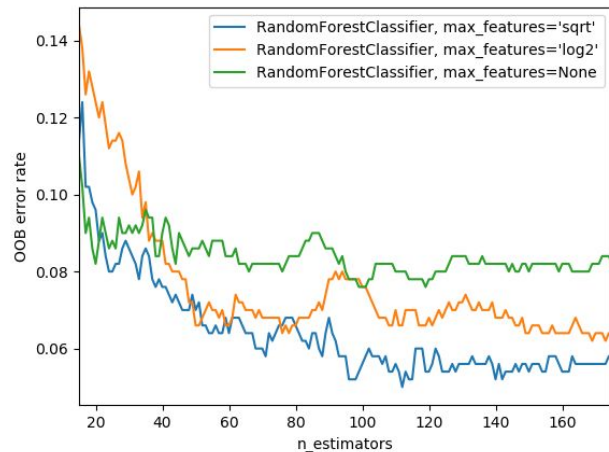
Bootstrapped Dataset

Chest Pain	Good Blood Circ.	Blocked Arteries	Weight	Heart Disease
Yes	Yes	Yes	180	Yes
No	No	No	125	No
Yes	No	Yes	167	Yes
Yes	No	Yes	167	Yes

Random Forests

Random Forests de-correlate each of the decision trees created in bagging by ensuring that at each split, only m features are considered for a given split. (typically $m = \sqrt{p}$)

This means that on average $(p-m)/p$ splits will not even consider a given strong predictor. As we increase the number of trees, this will not lead to overfitting, meaning that we should make as many trees as possible until we have achieved an acceptable error rate.



Random Subspace Sampling Method

Important note: The m features are randomly chosen at each **node** not for the entire decision tree.

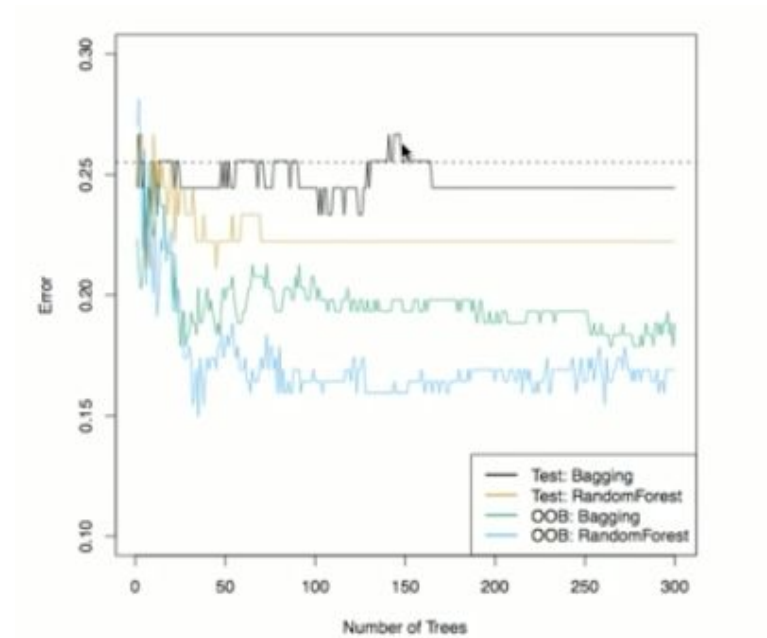
This is to ensure that our trees need to be decorrelated with one another

Our trees should have **diverse** opinions

Comparing Performances

In general:

Random Forest > Bagging > Decision Trees



Random Forest Advantages/Disadvantages

Advantages

- A very powerful model. Will nearly always outperform decision trees
- Able to detect non-linear relationships well
- Harder than other models to overfit

Disadvantages

- Not as interpretable as decision trees
- Many hyperparameters to tune (GridSearch is your friend!)

Random Forest Hyperparameters

`n_estimators` : the number of trees in the forest

`criterion`: “gini”, “entropy”

`max_features`: the number of random features to be considered when looking for the best split

`max_depth`: the maximum number of levels of a tree

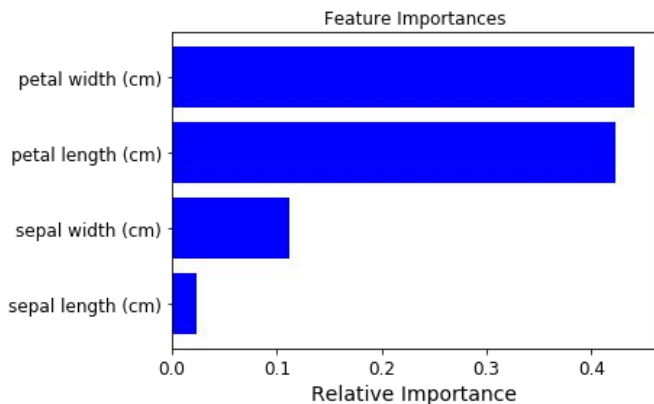
`bootstrap`: whether or not bootstrap samples are used to build trees

`oob_score`: whether or not to use out-of-bag samples to estimate the generalization accuracy

`n_jobs`: how many cores you want to use when training your trees

Feature Importances

The more the accuracy of the random forest decreases due to the exclusion (or permutation) of a single variable, the more important that variable is, and therefore variables with a large mean decrease in accuracy are more important for classification of the data.



There are many other ways to determine the feature importances of Random Forest. Check them out here <https://papers.nips.cc/paper/4928-understanding-variable-importances-in-forests-of-randomized-trees.pdf>

On another note

In sklearn, you can create custom ensemble models by making use of the VotingClassifier/VotingRegressor. This is intended to be used with conceptually different models.

- Within the VotingClassifier, you can specify “Majority Class” vote or “Soft” vote.
 - Majority Class: Select the class that is predicted most
 - Soft: Take an average of the probabilities for each class, make decision based off of that

Question to ponder.....

How do Random Forests handle the bias-variance tradeoff? What would be another way of using ensembling methods to tackle the bias-variance tradeoff?

Additional Resources

<https://www.stat.berkeley.edu/~breiman/randomforest2001.pdf>

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