Decision Trees (and More)

Tree-based methods for classification and regression.

Sean Kang

Decision Trees and Overfitting

Overfitting and Decision Trees

Running the training data on the decision tree, we can always get 100% accuracy. This should give us concern about possible overfitting.

Decision tree can easily get overfitted.

The decision tree can get 100% accuracy on the training data.

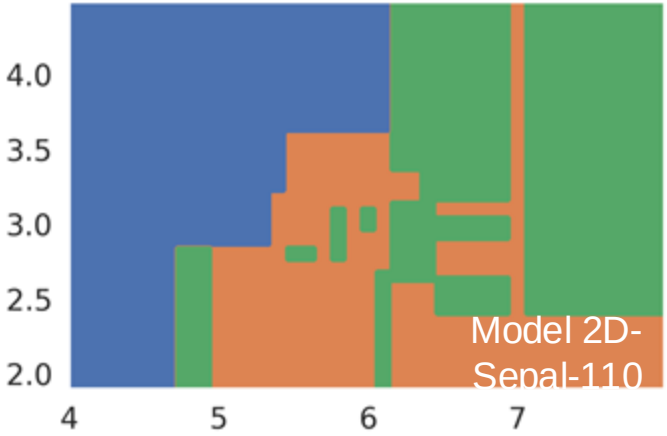
But it can perform poorly on unseen test data.

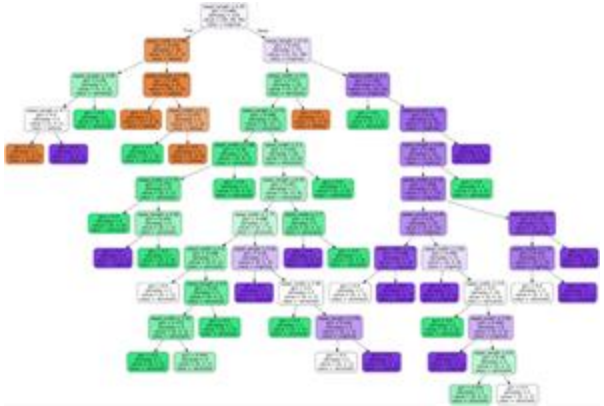
Overfitting Visualized

The decision boundaries for our sepal model were quite complex

- Or drawn out as a tree, we also see a highly complex structure

Next, we'll discuss strategies for preventing overfitting





Technique to avoid overfitting on trees

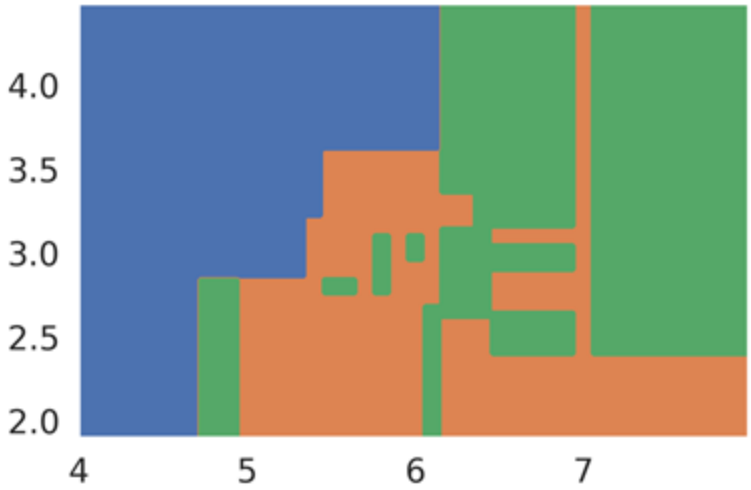
- Pruning
  - Pre-pruning
  - Post-pruning
- Ensemble – a new idea of doing it
  - Random Forest

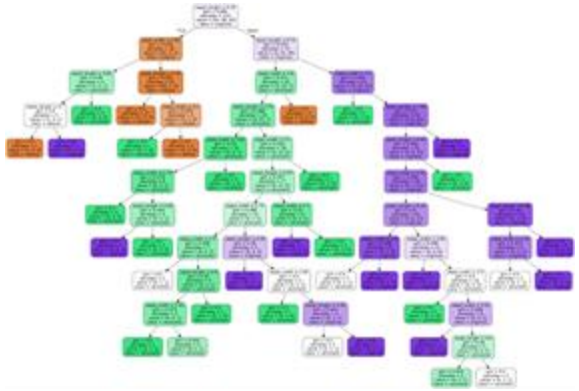
Restricting Decision Tree Complexity or Remove Overfitting

Overfitting and Our Algorithm

A “fully grown” decision tree built with our algorithm runs the risk of overfitting

One idea to avoid overfitting: Don’t allow fully grown trees. By default, a decision tree grows to its full height.





Approach 1: Preventing Growth (pre)

Approach 1: Set one or more special rules to prevent growth

- Don’t let a node get split from an outlier or a few data records
- Don’t allow the tree height to get too tall

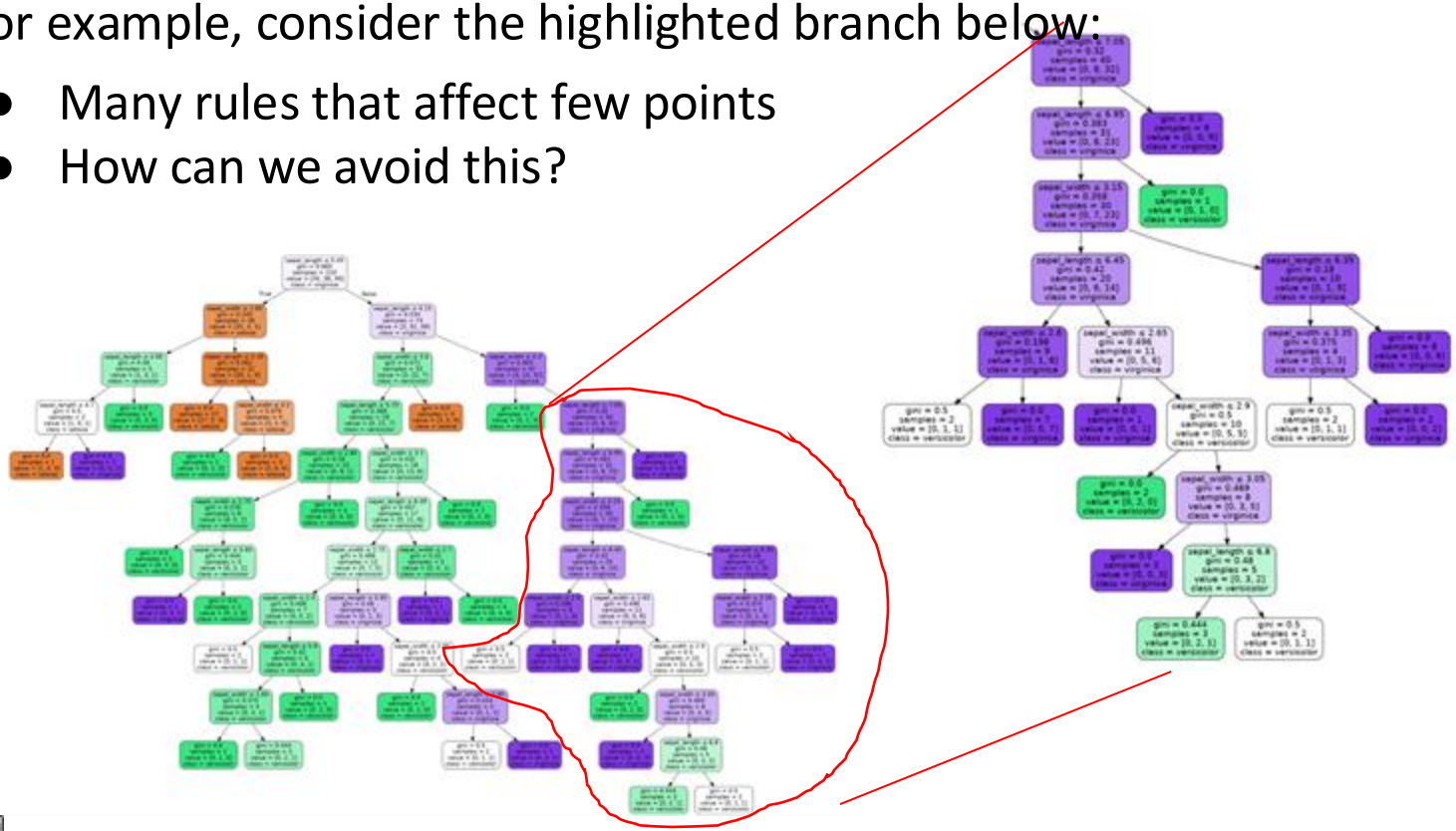
Examples:

- Don’t split nodes with < 1% of the samples
- Don’t allow nodes to be more than 7 levels deep in the tree

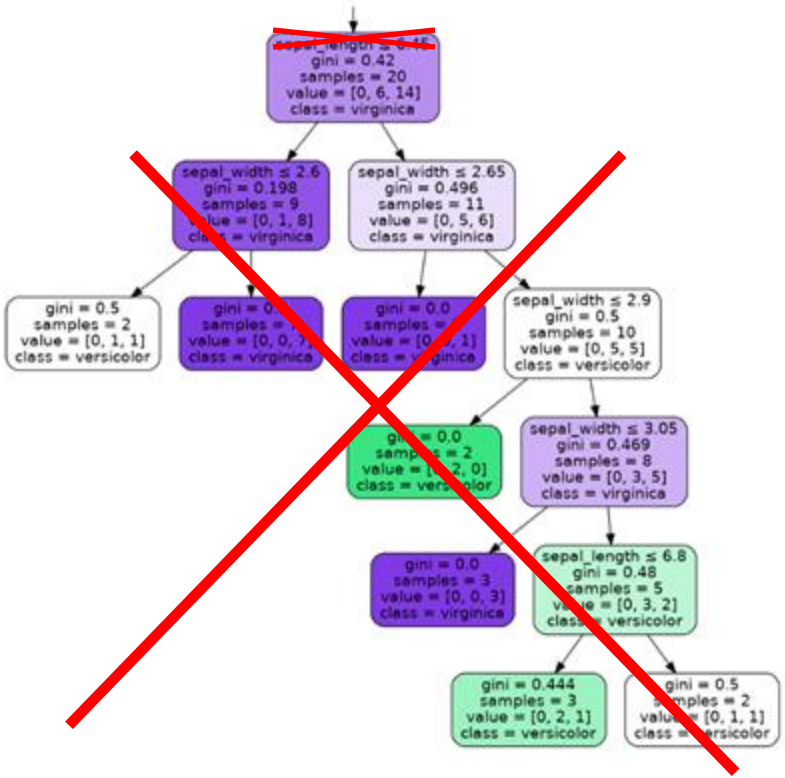
Approach 2: Pruning (post)

Approach 2: Let tree fully grow, then cut off less useful branches of the tree.
For example, consider the highlighted branch below:

- Many rules that affect few points
- How can we avoid this?



Specific Pruning Example



One way to prune:

- Before creating the tree, set aside a validation set
- If replacing a node by its most common prediction has no impact on the validation error, then don't split that node

How to Interpret the Post-Pruning

- The best AUC ROC score should be the highest..
- The corresponding cch-alpha is the optimal value for post-pruning.

Summary of Pruning: Overfitting and Our Algorithm

A “fully grown” decision tree built with our algorithm runs the risk of overfitting

One idea to avoid overfitting: Don’t allow fully grown trees

- Approach 1: Set rules to prevent full growth
- Approach 2: Allow full growth then prune branches afterwards

Won’t discuss these in any great detail

- There’s a completely different idea called a “random forest” that is more popular and more beautiful

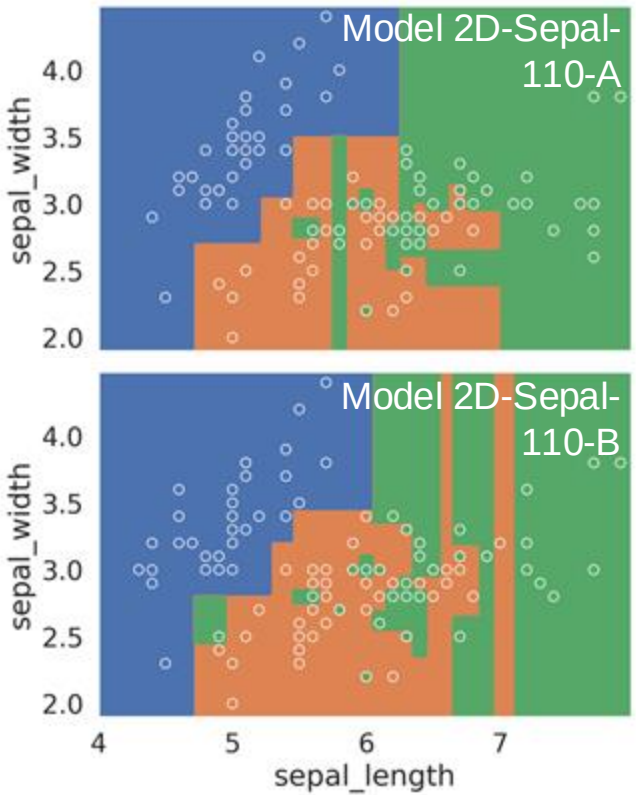
Another Approach: Random Forests

Random Forests: Harnessing Variance

As we've seen, fully-grown decision trees will almost always overfit data

- Low model bias, high model variance
- In other words, small changes in dataset will result in very different decision tree
- Example: Two models on the right trained on different subsets of the same data

Random Forest Idea: Build many decision trees and have them vote



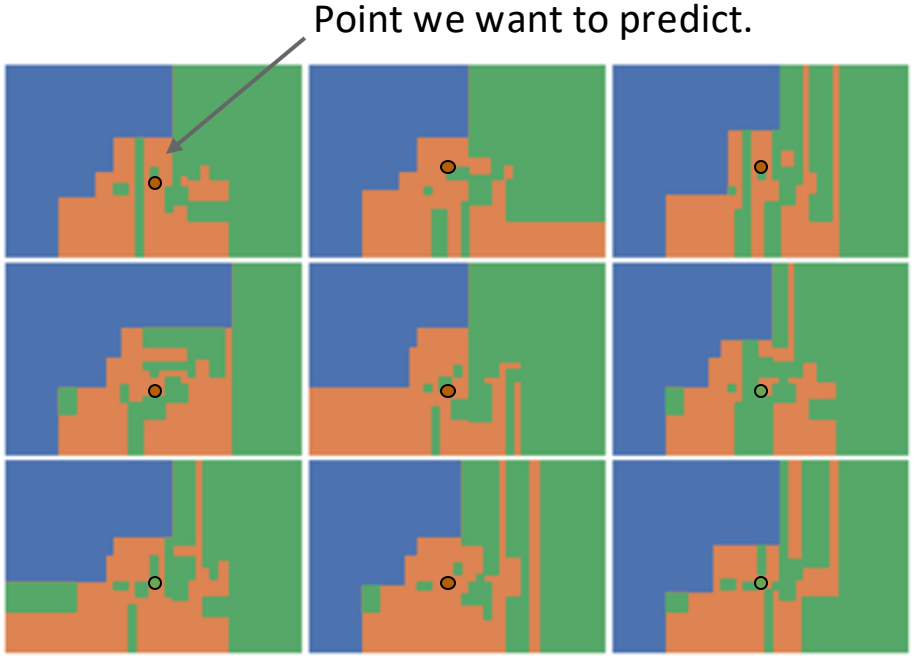
Random Forests: Harnessing Variance

What do we mean by vote?

- For a given x/y, use whichever prediction is most popular

Consider example at right with 9 models

- 6 votes orange, 3 votes green
- Random forest prediction is orange



Building Many Trees

Big fundamental problem: We only have one training set.

How can we build many trees using one training set?

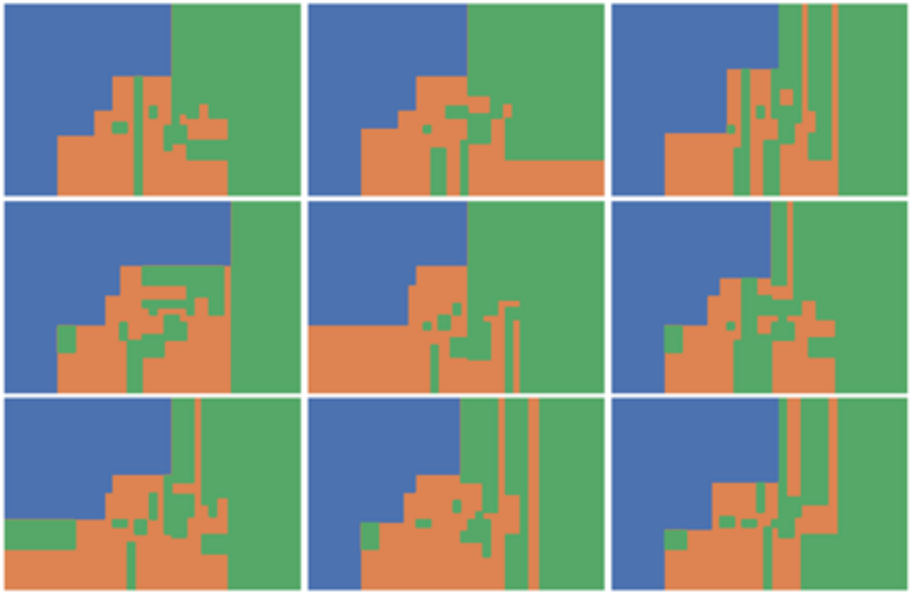
Bagging

Bagging: Short for Bootstrap AGGregatING

- Generate bootstrap resamples of training data (draw samples with replacements)
- Fit one model for each resample
- Final model = average predictions of each small model

Random Forests

- Bagging often isn't enough to reduce model variance!
  - Decision trees often look very similar to each other
  - E.g., one strong feature always used for first split



Random Forests

- Bagging often isn't enough to reduce model variance!
  - Decision trees often look very similar to each other and thus make similar predictions
  - Ensemble will still have low bias and high model variance
  - E.g. one strong feature always used for first split
- Idea: Only use a sample of m features at each split
- Algorithm creates individual trees, each overfit in a different way
  - The hope is that the the overall forest has low variance

Random Forests Algorithm

- *Bootstrap training* data **T** times. For each resample, fit a decision tree
- To predict, ask the T decision trees for their predictions and take majority vote

This approach has two hyperparameters *T* and *m*, where *m* is the number of features selected randomly for each split

Avoiding Overfitting with Heuristics

We've seen many approaches to avoid overfitting decision trees

- Preventing growth
- Pruning
- Random forests

These ideas are generally "heuristic"

- Not probably best or mathematically optimal
- Instead, they are just ideas that somebody thought sounded good, implemented, then found to work in practice acceptably well. LOL

Why Random Forests?

- Versatile: does both regression and classification
- Nonlinear decision boundaries without complicated feature engineering
- Doesn’t overfit as often as other nonlinear models (e.g. polynomial features)

Beyond Decision Trees for Classification

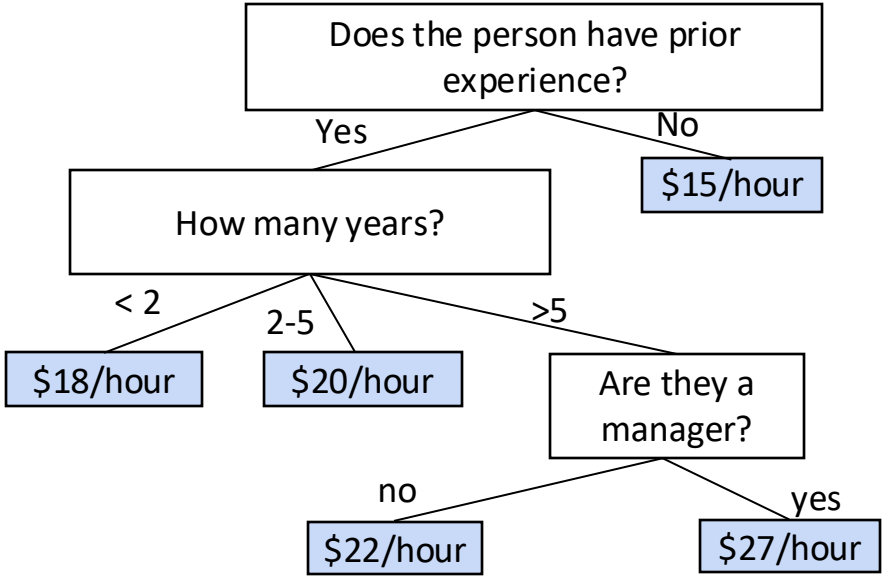
Trees for Regression

In earlier lectures we saw how we could use a logistic regression model for classification

- This lecture: We saw decision trees as an alternative technique for classification

We could do the same exercise for regression

- Rather than using a linear model, we could build a regression tree



Summary

Decision trees provide an alternate non-linear framework for classification and regression

- The underlying principle is fundamentally different
- Decision boundaries can be more complex
- Danger of overfitting is high

Keeping complexity under control is not nearly as mathematically elegant and relies on heuristic rules

- Hard constraints
- Pruning rules
- Random forests
  - Very interesting application of bootstrapping