More Random Forests

Cary Goltermann (mostly stealing from Darren Reger)

Objectives

- Feature Importances
- Tuning Random Forests
- Best Practices for Random Forests
- Handling Categorical Features
- Handling Missing Values

How many trees?

- Variance decreases with more trees
    - With diminishing returns
- Run time scales linearly with more trees
- n_estimators in sklearn
- More is still better, but wait until the end to run 50,000 trees

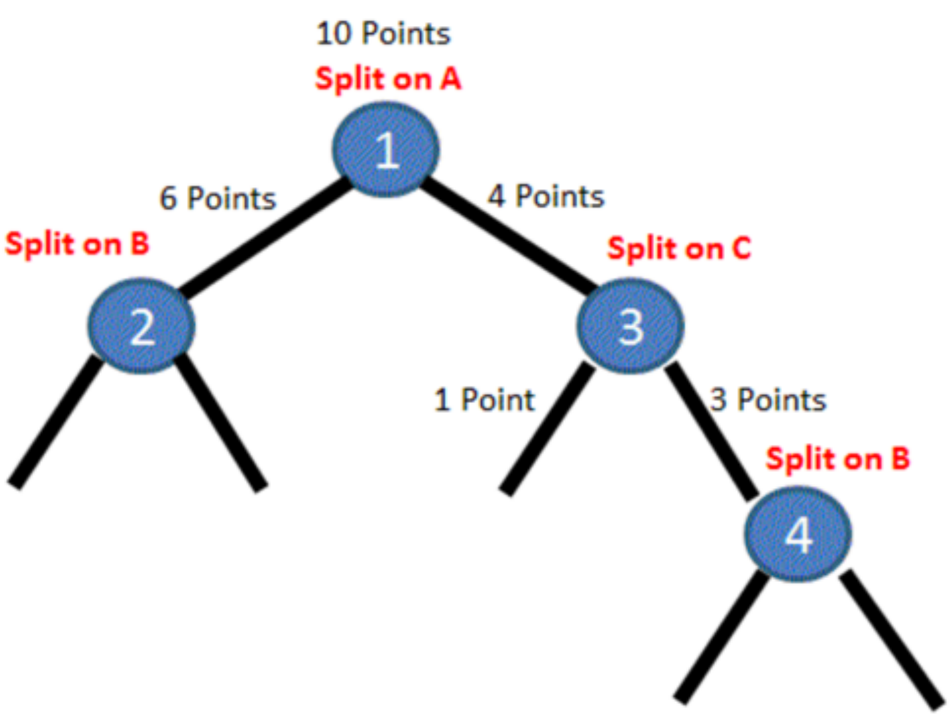
Feature Importances

- Relative importance of the variables in your data
- When you have many features you want to know which are the most important
    - For recommendations
    - For model building

1st Feature Importance

1. Start with an array that is the same size as the number of features in your model, initialize it to 0
2. Begin to traverse the tree, with the data that was used to build the tree
3. Whenever you reach a branch in the tree, determine what feature the branch operated on (every branch splits on only one feature). Additionally, count how many data points reached this branch, as opposed to going down a different path on the tree
4. Calculate the information gain after branching as opposed to before this branch. (Criterion doesn’t matter, can be MSE, Entropy, Gini, etc…)
5. Multiply the information gain by the number of data points that reached this branch and add that product to the array at whatever feature is being split
6. Once all the information gain for all the branches has been summed, normalize it.
7. Repeat for all the trees in the forest and average the values. (best to do this at the time of tree construction)

Feature Importance Example1



Feature Importance				
Split #	# of Data Points	Split on Which Feature	Information Gain	# of Nodes * Information Gain
1	10	A	0.26	2.6
2	6	B	0.4	2.4
3	4	C	0.3	1.2
4	3	B	0.1	0.3

Importance		
Feature	Importance	Normalized
A	2.60	0.40
B	2.70	0.42
C	1.20	0.18

2nd Feature Importance

1. Generate the OOB error for the base tree, without making any changes. This is the baseline OOB error
2. For every feature in the data used to create the tree, determine the range of values that those features can take. i.e. find the min and the max
3. For each feature, one at a time, randomly permute the values of that feature in all the data points between the max and min possible for that feature.
4. Then, after only permuting one feature, find the OOB error
5. Calculate the difference between the new OOB error and the baseline OOB error. The new OOB error will almost certainly be worse.
6. Reset the permuted feature values to their original values and repeat the process with a different feature.
7. The features which have the greatest increase in OOB error when they are permuted are the most important. If you can set a value to basically anything and not change the OOB error, it’s not important.

Feature Importance Overall

- Feature importances are almost always put forth as normalized values. What is important is that we can compare features to other features.
- The original authors of RF state that you should be interested in rank only and not magnitude
- Typically, the more features you have in a random forest, the less important any individual feature will be.
- Highly correlated features tend to split importance
- Some highly correlated but not super important features will look important

Things We Can Tune

- max_features
    - classification: start with square root of p
    - regression: start with p/3 or full features
- min_sample_leaf
    - start with None and try others
- n_jobs
    - -1 will make it run on maximum # of processors

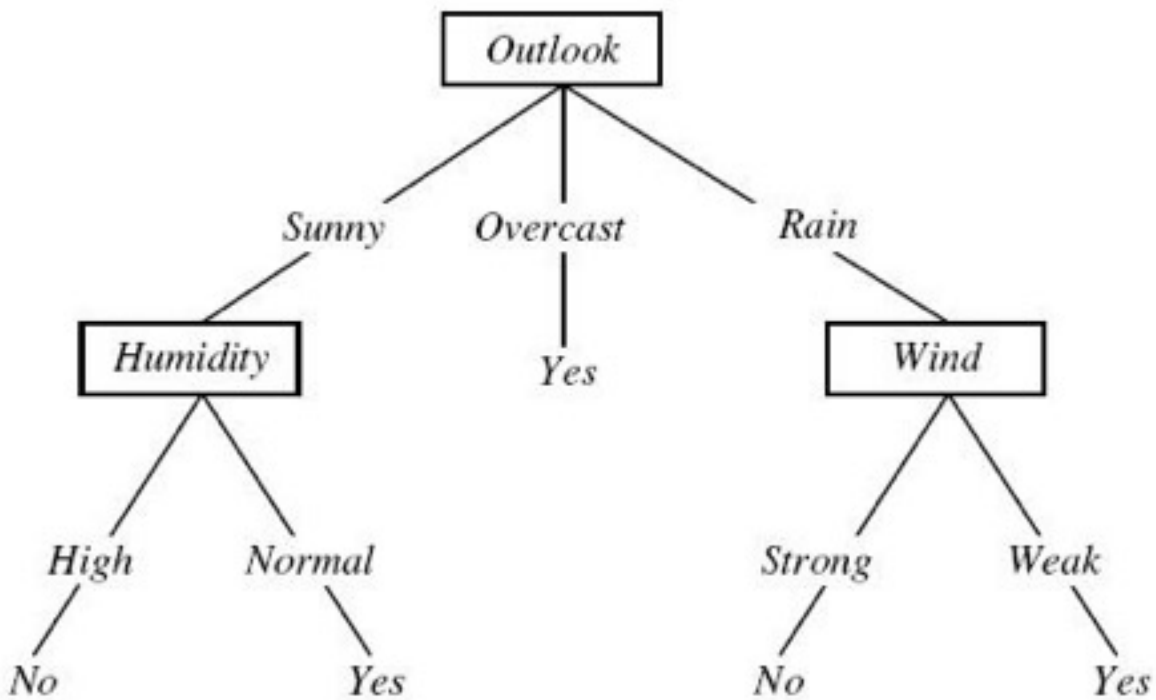
How to Handle Categorical Data

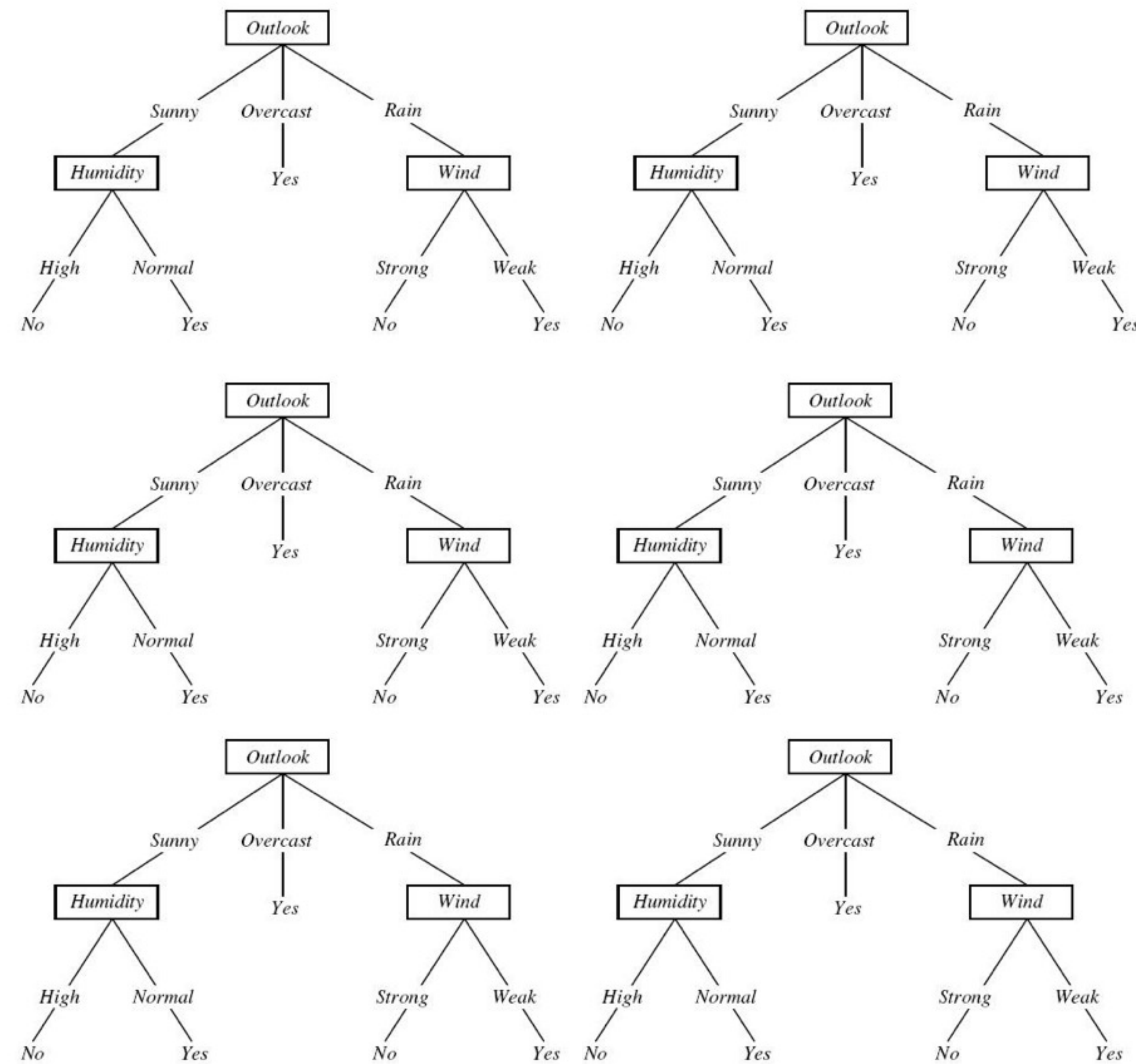
- String values need to be converted into numeric
- If possible, convert to a continuous variable
  - e.g. S, M, L to size in actual weight or height
- sklearn doesn't support splitting on multiple features (uses the <= for all variables)
- DO NOT DROP ONE OF THE CATEGORIES!!!

Random Forest Pros

- For an out of the box model, it has very good accuracy
- Trees can be trained in parallel to make computations faster
- OOB estimates allow for an estimate of generalization error without needing CV
- Can handle thousands of features and be used for feature reduction

When are DT's Better?





Reminders

- Cannot extrapolate well for regression trees
- Just because we have interactions, doesn't mean you'll never want interaction variables
- Just because we have OOB error, doesn't mean you'll **never** use CV
- Start with a small number of trees at first, then increase
- Pickling makes a giant file (~GB)

Missing Values

- Typical implementation will use the median value
- Can first use the median values and then use proximities to calculate a more accurate missing value later

Proximities

- Can be used to see how similar 2 data points are
- After fitting the random forest, for each tree in the random forest:
  - Count the number of times that 2 data points are in the same leaf node
- Normalize at the end