Random Forests / Decision Forests

Duke Course Notes

Cynthia Rudin

The idea of averaging the votes of lots of diverse decision trees is very powerful. The first work I know of on this is by Tin Kam Ho (1995), though Leo Breiman (2001)'s work is often cited. It is a simple and elegant idea: if you average many different yet accurate (perhaps even overfitted) models, it reduces the variance in predictions.

Random forests are considered black box models because we usually average together a lot of trees, though one could argue that if you average only a few shallow trees, this could be interpretable to humans.

An analogy I find helpful is when you might need surgery, you might ask for a second opinion. If the second doctor had the same training as the first doctor and has seen exactly the same set of patients, you might be concerned that the second opinion is not giving you any more information than the first opinion. But if the second doctor is actually different, you might trust the combination of them more.

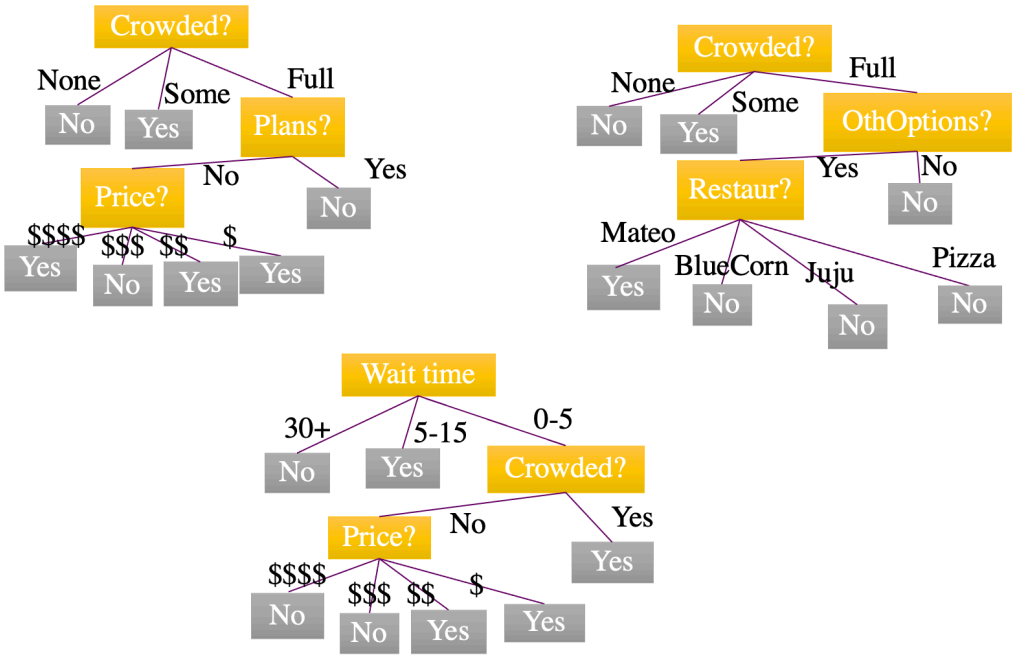
Let us describe the simpler idea of *bagging*. Random forests forces more diversity among trees than bagging.

Bagging and boosting are *ensemble* techniques, where an ensemble of models votes on the prediction.

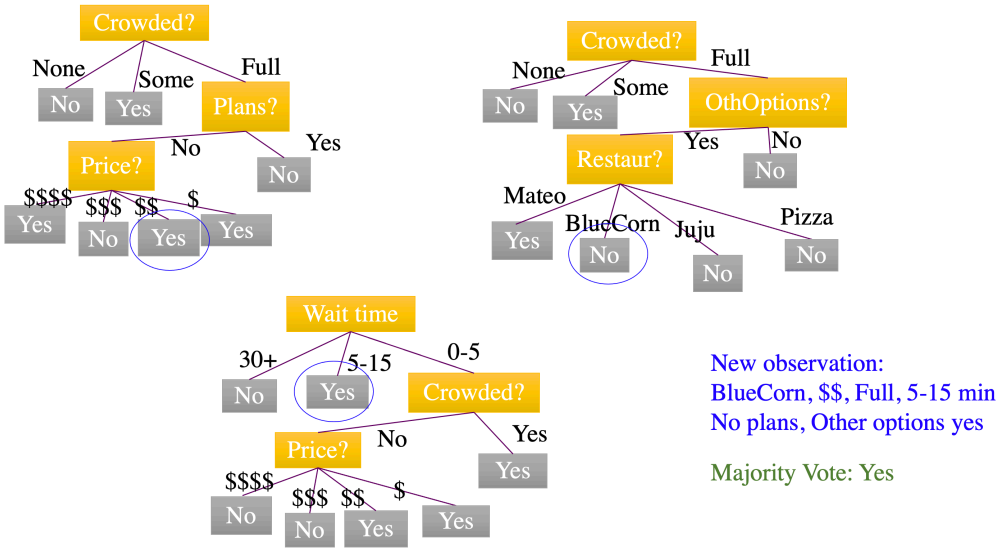
Bagging (Boostrap Aggregating)

Draw n points from the training set *with replacement*, and grow a tree from them. Repeat several times. If doing regression, average the trees together to get the final prediction. If doing classification, take the majority vote among the trees to get the prediction.

Let's say we created 3 trees with our 3 bootstrap samples using the restaurant data from the decision tree lecture.



Now we have a new observation, where the customer is at the Blue Corn Cafe, which is \$\$, the restaurant is full, they were given a 5-15 minute wait time, they have no plans, and there are other options nearby. Following the three separate trees gives two "yes" predictions and one "no" prediction. The ensemble predicts the majority vote, which is "yes" the customer will wait for a table.



Random forests is similar to bagging, except that it encourages more diverse trees by only allowing each split to use a set of randomly-chosen features.

Random Forests (also called Decision Forests)

The number of trees to use in the forest T (which you would choose to be an odd number so you can calculate a majority), and the parameters m and $n_{\min}$, are user-chosen parameters. You might also put a limit on the depth of the trees if desired. The main difference from bagging is that we are only allowed to split on m randomly-chosen features for each split.

- For t=1 to T:
  - Draw a bootstrap sample of size n from the training data.
  - Grow a tree (tree t) using this splitting and stopping procedure:
    - For this split, choose m features at random (out of p total)
    - Evaluate the splitting criteria on all of them, and split on the best feature
    - If any node we create has less than $n_{\min}$ points, then do not allow more splitting on it.
- Output all the trees.

To predict on a new observation x, use the majority vote (or average) of the trees on x.

This procedure forces diversity among the trees, since we are unlikely to be able to repeatedly construct the same tree. This diversity helps to reduce the variance in predictions and helps prevent overfitting. For instance, let's say you created 100 overfitted trees on the same dataset using this procedure and take the majority vote to make a prediction. You could imagine that if you took 200 trees instead, the prediction would be about the same as if you took 100 trees. But if you had only taken 1 overfitted tree, it might not produce the same result as the ensemble. In that sense, the predictions from the ensemble have lower variance than for a single tree.

Since the decision trees are deep, we are using a complex class of models with low bias; deep trees can fit essentially any data set. Thus, the ensemble has both low bias and low variance. We could contrast that with individual sparse trees,

which have high bias and low variance.

What's nice about random forests is that even if each of the trees overfits the data, the combination of them is unlikely to. It's almost as if we're averaging out the quirks that each split of the dataset possesses. It's really a wonderful idea to combine a lot of overfitted models to create something that is powerful and doesn't overfit!

There are some disadvantages of random forests though. 1) They are generally complicated (black box). 2) Producing them is slow. At each node you need to evaluate m different possible splits, and you need to repeat the whole process for T trees. At least you can parallelize it and have multiple processors creating tree.