STATISTICS 452/652: Statistical Learning and Prediction

October 19, 2020

Lecture 15: Ensembles2 - Random Forest

(Reading: Section 8.2.2)

1 Goals of lecture

- Bootstrap Aggregation (Bagging) is a remarkable technique that can reduce the variance of a prediction machine
- It turns out that it can be improved by a few small adjustments when $p > 1$
- The result is one of the most popular and universally successful machines in our toolkit
  - And it is relatively automatic!

2 Random Forests

- Resampling data and refitting a regression tree is easy and effective
- However, its effectiveness is limited when $p >$ 1 by the fact that each tree tends to look "mostly the same" with relatively few differences.
  - Major variables are split more near the top
  - Less important variables appear near the bottom or not at all
  - If estimates means for major variables are influenced badly by random errors, ALL trees are impacted
  - Averaging them still reflects this impact
- To get around this, we need a way to "mix the trees up" a little bit, while still keeping them as useful

2.1 The Random Forest tweak

A RANDOM FOREST (RF) is just bagging regression trees with one tiny, little difference:

- In addition to using bootstrap resampling, *also randomly select a subsample of explanatory variables!*
- **For each parent node (i.e. potential split):**
  - Randomly choose $m \leq p$ variables.
    - Common recommendation: $m = \sqrt{p}$ for classification, $= p/3$ for regression
    - m is a tuning parameter; can play with it.
    - Note that $m = p$ uses all of the variables, and so is exactly bagging.
  - Pick best split only from among candidate variables for that parent node.
- Resampled data is chosen for whole tree; subsets of explanatories are chosen for each split.
- With small number of variables available at each split, can't expect each tree to be a great model. But each will address some aspect(s) of Y
  - "Weak learners"
  - On the one hand, an inferior predictor: possibly larger variance of individual predictors
  - On the other hand:
    - Effects of some minor variables may finally come through
    - Might get advantage of combining dominant trend (chosen most of the time) with secondary trends (chosen some of the time)
- **Reduces correlation between trees** because they won't always be based on as many similar splits
- Except for the selection of a subset of explanatory variables, use the same algorithm for tree building, aggregation through averaging, and computing out-of-bag samples as in bagging.
  - Predicted value for any x is the average mean from that x's terminal node in each of the trees in the forest
    - Predicted values for any x in the training sample is based on OOB predictions
- Need more trees because they are weaker learners than in bagging: often as many as 500–1000
  - May depend on p and m: larger p or smaller m need larger B to achieve adequate aggregation (more variety in learners and more potential variables contributing to response)
- Produces a smoother step function than individual trees, but loses tree structure

3 Variable Importance

1. Despite lack of tree structure, can gauge relative importance of variables
2. 2 ways to do this. (ISLR authors prefer one, I prefer the other!)

Variable importance by Mean Decrease in RSS (Preferred by ISLR)

1. In every parent node we have a set of candidate variables for splitting.
2. One gets chosen, and the split reduces the RSS by some amount.
3. Having different candidate subsets means that more variables get used at one time or another
   (a) "Poor" variables' splits will not change RSS much when they are chosen
   (b) "Good" variables will make relatively larger changes when they are chosen.
   (c) "Good" variables will be chosen more frequently, including near the top when they are candidates there
      i. Greater potential for reduction
4. In each tree we can measure how much each variable contributed to that tree's reduction in RSS
   (a) Average this across trees: Mean Decrease in RSS (or Increase in Node Purity)
   (b) Preferred by HTF

Variable importance by Mean Decrease in Accuracy (Preferred by Izenman, Adele Cutler, and me)

1. For each tree, the OOB observations get predicted. Compute measure of "accuracy" ($MSPE_{b,\text{orig}}$, in regression)
2. For the same tree's OOB observations, force one variable, X_j, to be a useless predictor and see how much worse predictions get
   (a) For tree b and all $i \in \text{OOB}_b$, permute the values of X_{ji} and re-attach them to the data
      i. Run these altered data down the tree to get predictions for each $i \in \text{OOB}_b$.
      ii. If there are any splits on X_j, data get sent randomly left or right in the same proportion as before
   (b) Compute measure of accuracy (again, $MSPE_{b,\text{perm}}$) for the adjusted data
   (c) The difference $MSPE_{b,\text{perm}} - MSPE_{b,\text{orig}}$ measures much decrease results from randomizing X_j.

3. Repeat this for all variables on tree b

4. Repeat this for all trees and take average decrease for each variable across all trees.

   (a) `randomForest()` in `R` converts this to a percentage increase relative to the SD of the differences across trees and calls this "`%IncMSE`"

Notes

- Variable importance measures are somewhat variable, especially the permutation-based measure.
  - It is sometimes recommended to run RF several times and the average variable importance measure if the goal is to measure this
  - Has advantage of providing standard error across multiple runs thanks to bootstrap.
- Depending on the proportion of "important" and "unimportant" variables in a problem, may need to use larger m for measuring variable importance than for prediction
  - If m is too small, unimportant variables get used more, important ones get used less, and importance measures are compressed
  - But if m too large, dominant variables may obscure less important variables, and measures become overly skewed

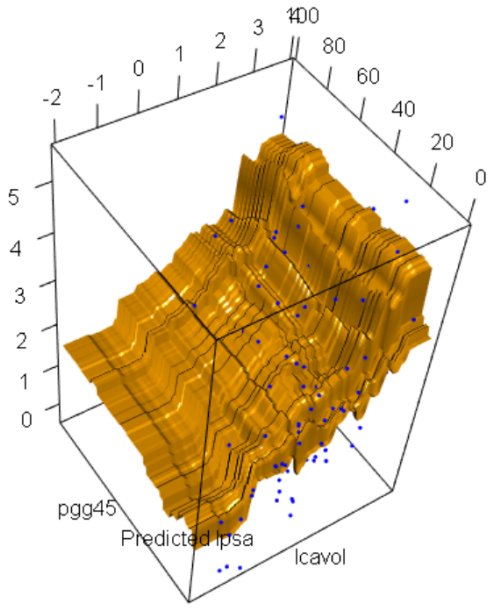
Example: Random Forest on Prostate Data (L15 - Random Forest Prostate.R)

The function `randomForest()` in the package `randomForest` has lots of options for computing, assessing, and presenting RF predictors. There are numerous additional helper functions to summarize and assess the fitted machine. See the package help page for details. I show some of the most useful things in the example.

I use the function on the Prostate data. I first use just two variables, `lcavol` and `pgg45`, to show some of the features and make a plot of the surface. Then I use all 8 explanatory variables to show variable importance. Finally, I show how to simultaneously tune the number of variables at each split (`mtry`) and the minimum size of node that can be split, (`nodesize`). Details are in the program.

First the fit to two variables leaves only two choices for `mtry`: 1 or 2. Just to get started, I chose 1. We revisit this later. The resulting fitted surface is in Figure 1. The structure as an average of many trees is evident in the sort of "smooth jumpiness" of the picture. We again see the dominant trend of increasing `lpsa` with increasing `lcavol`. It seems not quite linear. There is little evidence of an effect from `pgg45`, despite the fact that `mtry=1` forced it to be used in about half of all splits. The result of this is a somewhat lumpier surface than I might like to see. You can see whether setting `mtry=2` improves smoothness.

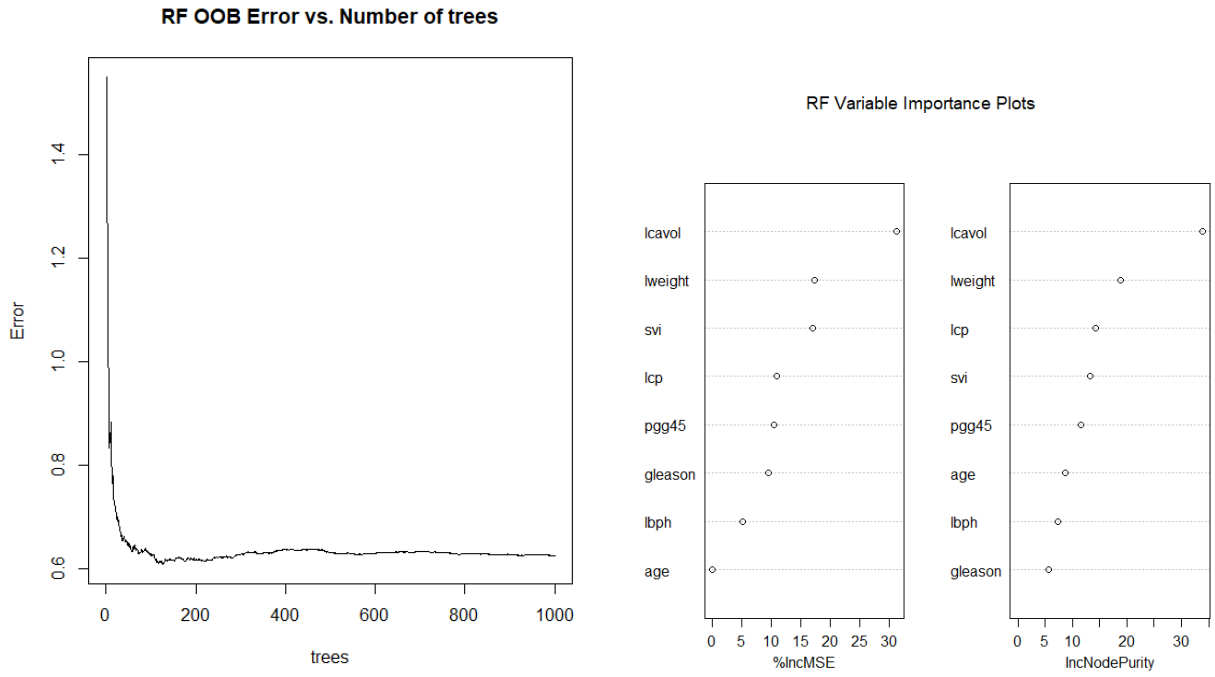
Figure 1: Fitted RF surface for 2 variables, `lcavol` and `pgg45` in Prostate data



Next I fit a default random forest to the data using all 8 explanatory variables. I first check whether we have enough trees in the ensemble by examining whether the OOB error appears to converge to a value within that number of trees. The left panel of Figure 2 shows how the OOB error settles down rather quickly with relatively few trees. After about 200 trees, there is very little change in the error, and certainly by about 500 there is hardly any movement. Note that we are *not* trying to minimize OOB error in this plot.

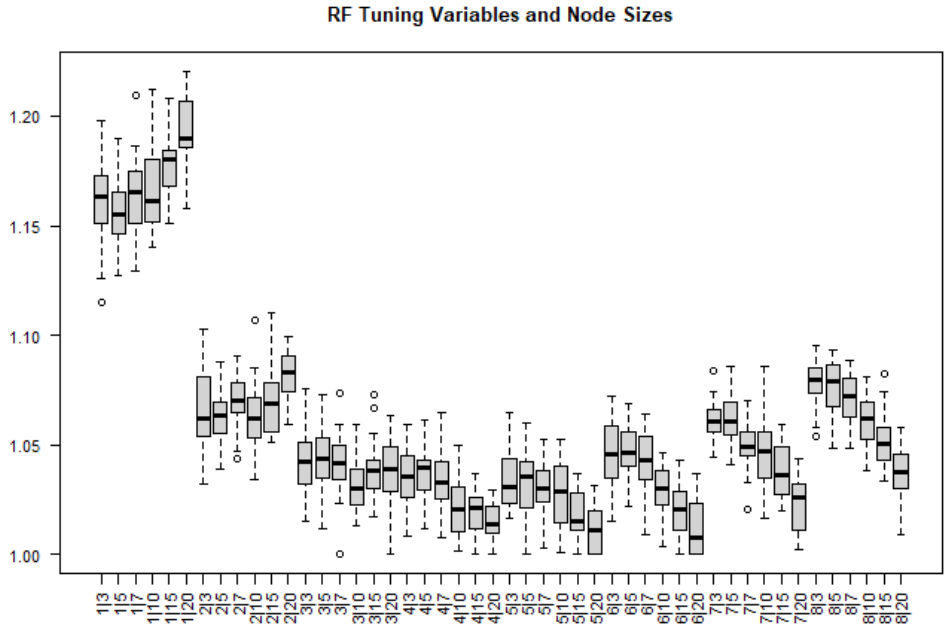
Using the default `mtry=round(p/3)` means using round(8/3)=3 variables for each split in this example. With this setting, we get the two measures of variable importance shown in the right panel of Figure 2. On the left it the permutation-based measure, "`%IncMSE`" that I prefer, and on the right is the one based on training error as presented in the book. Neither of these has a fixed threshold for "important" vs. "unimportant" variables. Instead, it is a matter of relative importance, "more" and "less". Both show that `lcavol` is dominant as the most important variable. The permutation measure then shows what opther analyses have shown: `lweight` and `svi` seem to be useful but to a lesser degree. Other analyses have put very little importance on other variables, and we see the same basic picture here.

Figure 2: Plots from RF on Prostate data with all 8 variables using default settings. Left: OOB error as number of trees increases. Right: Variable importance measures.



Finally, I show how to tune the RF using the built-in OOB error. This is essentially like using bootstrap to compute MSPE. I tune on all values of $m = 1, \ldots, 8$, and on node sizes for splitting in (3, 5, 7, 10, 15, 20). Normally, I might choose, say, 3–4 values of m, like $p/6, p/3, p/2, 2p/3$, since the extreme values are rarely the winners. But with such a small $p = 8$ here, I can show the entire sequence. Similarly, I might normally tune on a smaller set of node sizes, maybe 3, 5, 10, and possibly one larger. Finally, I have replicated the measurements 20 times, which is a lot, but *again* I have a relatively small data set. The whole process took about a minute or two to run—that's nothing.

Figure 3: Relative RMSPEs for tuning `mtry` and `nodesize` in the RF on the Prostate data. Labels are "`mtry | nodesize`"



The results are in Figure 3. Across numbers of variables, we see the usual pattern of a fast descent from a biased machine (`mtry`=1–2), a relative flat spot (`mtry`=3–6), and the slow rise as variance increases (`mtry`=7–8). Within each `mtry` value, we see different impacts of the node sizes. Focusing on the `mtry`=3–6 range, it seems that larger node sizes for splitting are preferred. I might be tempted to try a few larger values than 20 with, say, 4–6 variables. Note, however, that while we can improve on the default values of 3|5, they are not really that bad.

---

4 Notes

1. Bagging is just a special case of RF, where $m = p$.
2. As with Bagging, there is not a general advantage to pruning trees.
   (a) Generally grow trees out to full size, subject only to node-size restrictions.
   (b) It is again possible that this is too bumpy. Can restrict tree size if needed.
   (c) Default in `randomforest()` is `nodesize=5` for regression.
      i. However, what they say and what they do are not the same. They say that `nodesize` is the "Minimum size of terminal nodes"
      ii. It is really the minimum size of nodes that can be split further; i.e., like `min.split` in `rpart()`. Terminal nodes as small as 1 are often produced!

iii. This may be too small in some applications. Tune it.

3. There are many other versions of RFs

   (a) "Extra (**Ext**remely **Ra**ndomized) Trees" (Guerts et al. 2006) are another somewhat popular type of RF with two changes:

      i. For each selected variable, the split point is chosen at random, rather than in the usual optimal way

      ii. No resampling is done—the original sample is used intact at the start of every new tree.

   I don't know how well they perform compared to random forests in general

4. Prediction intervals now available with RFs!

5 What to take away from this

1. Random forests are a premiere prediction tool for regression

2. The default version is usually a reasonably competitive machine, and is relatively fast

   (a) Definitely worth considering if computation time is limited

3. Tuning can help, but usually only a little bit

6 Exercises

Application

Refer to the Air Quality data described previously, and the analyses we have done with `Ozone` as the response variable, and the five explanatory variables (including the two engineered features).

Use Random Forest (RF) to model the relationship between `Ozone` and all five explanatories as specified below.:

1. Fit a default RF to *only the three main variables in the data*—`Temp`, `Wind`, and `Solar.R`—and not the two extra ones that we engineered. A RF should be able to detect interactions automatically if needed.
   (a) **Report the OOB error.**
   (b) **Produce variable importance measures and comment on the relative importance of the variables. How do they compare to what we have seen in earlier analyses of these data?**
2. Repeat the exercise above, adding the two engineered features into the data.
   (a) **Report the OOB error. Does it improve much compared to the previous RF analysis without the variables?**
   (b) **Produce variable importance measures. Are the two engineered features particularly important?**
3. Add both default (no tuning) and tuned versions of the RF into the CV analysis comparing previous methods. Use a grid of `mtry`=2,3,4 and `nodesize`=3,5,10.
   (a) **For the each of the 10 folds, report the chosen best values of the RF tuning parameters.** (I expect that they will vary from one fold to the next).
      i. **Do you think that it might help to try tuning on an extended grid? Explain.**
   (b) **Compare the mean MSPE from the default and best-tuned RF to the other models tried so far.**
   (c) **ADD the default and tuned RF to the relative MSPE boxplots made previously.**
      i. **Comment on how well they performs compared to other methods. Does tuning seem to help?**