Random forest

Sunday, September 16, 2018 9:44 PM

https://towardsdatascience.com/hyperparameter-tuning-the-random-forest-in-python-using-scikit-learn-28d2aa77dd74

https://machinelearningmastery.com/bagging-and-random-forest-ensemble-algorithms-for-machine-learning/

https://www.analyticsvidhya.com/blog/2016/02/complete-guide-parameter-tuning-gradient-boosting-gbm-python/

Here’s why:

- **Random Forests require almost no input preparation.** They can handle binary features, categorical features, numerical features without any need for scaling.

- **Random Forests perform implicit feature selection** and provide a pretty good indicator of feature importance.

- **Random Forests are very quick to train.** It’s a stroke of brilliance when a performance optimization happens to enhance model precision, or vice versa. The random feature sub-setting that aims at diversifying individual trees, is at the same time a great performance optimization! Tuning down the fraction of features that is considered at any given node can let you easily work on datasets with thousands of features. (The same is applicable for row sampling if your dataset has lots of rows)

- **Random Forests are pretty tough to beat.** Although you can typically find a model that beats RFs for any given dataset (typically a neural net or some boosting algorithm), it’s never by much, and it usually takes much longer to build and tune said model than it took to build the Random Forest. This is why they make for excellent benchmark models.

- **It's really hard to build a bad Random Forest!** Since random forests are not very sensitive to the specific hyper-parameters used, they don't require a lot of tweaking and fiddling to get a decent model, just use a large number of trees and things won't go terribly awry. Most Random Forest implementations have sensible defaults for the rest of the parameters.

- **Versatility.** Random Forest are applicable to a wide variety of modeling tasks, they work well for regression tasks, work very well for classification taks(and even produce decently calibrated probability scores), and even though I've never tried it myself, they can be used for cluster analysis.

- **Simplicity.** If not of the resulting model, then of the learning algorithm itself. The basic RF learning algorithm can be written in a few lines of code. There's a certain irony about that. But a sense of elegance as well.

- **Lots of excellent, free, and open-source implementations.** You can find a good implementation in almost all major ML libraries and toolkits. R, scikit-learn and Weka jump to mind for having exceptionally good implementations.

- As if all of that is not enough, **Random Forests can be easily grown in parallel.** The same cannot be said about boosted models or large neural networks.

From <https://medium.com/rants-on-machine-learning/the-unreasonable-effectiveness-of-random-forests-f33c3ce28883>