Google

Reading: More about random forests

Bagging + random feature sampling = random forest

You know that bootstrap aggregating—or bagging—can be an effective way to make predictions by building many base learners that are each trained on bootstrapped data and then combining their results. If you build a bagging ensemble of decision trees but take it one step further by randomizing the features used to train each base learner, the result is called a **random forest**. In this reading, you'll learn how random forests use this additional randomness to make better predictions, making them a powerful tool for the data professional.

Why randomize?

Random forest models leverage randomness to reduce the likelihood that a given base learner will make the same mistakes as other base learners. When mistakes between learners are uncorrelated, it reduces variance. In bagging, this randomization occurs by training each base learner on a sampling of the observations, with replacement.

To illustrate this, consider a dataset with five observations: 1, 2, 3, 4, and 5. If you were to create a new, bootstrapped dataset of five observations from this original data, it might look like 1, 1, 3, 5, 5. It's still five observations long, but some observations are missing and some are counted twice. The result is that the base learners are trained on data that is randomized *by observation*.

Random forest goes further. It randomizes the data *by features too*. This means that if there are five available features: A, B, C, D, and E, you can set the model to only sample from a subset of them. In other words, each base learner will only have a limited number of features available to it, but what those features are will vary between learners.

Here's an example to illustrate how this might work when combining bootstrapping and feature sampling. The sample below contains five observations and four features from a larger dataset related to cars.

Model	Year	Kilometers	Price
Honda Civic	2007	54,000	$2,739
Toyota Corolla	2018	25,000	$22,602
Ford Fiesta	2012	90,165	$6,164

Audi A4	2013	86,000	$21,643
BMW X5	2019	30,000	$67,808

If you were to build a random forest model of 3 base learners, each trained on bootstrapped samples of 3 observations and 2 features, it may result in the following three samples:

1.		**2.**		**3.**	
Kilometers	**Price**	**Year**	**Kilometers**	**Model**	**Price**
54,000	$2,739	2012	90,165	Honda Civic	$2,739
54,000	$2,739	2013	86,000	Ford Fiesta	$6,164
90,165	$6,164	2019	30,000	Ford Fiesta	$6,164

Notice what happened. Each sample contains three observations of just two features, and it's possible that some of the observations may be repeated (because they're sampled with replacement). A unique base learner would then be trained on each sample.

These are just toy datasets. In practice, you'll have much more data, so there will be a lot more available to grow each base learner. But as you can imagine, randomizing the samples of both the observations and the features of a very large dataset allows for a near-infinite number of combinations, thus ensuring that no two training samples are identical.

How does all this sampling affect predictions?

The effect of all this sampling is that the base learners each see only a fraction of the possible data that's available to them. Surely this would result in a model that's not as good as one that was trained on the full dataset, right?

No! In fact, not only is it possible for model scores to improve with sampling, but they also require significantly less time to run, since each tree is built from less data.

Here is a comparison of five different models, each trained and 5-fold cross-validated on the bank churn dataset from earlier in this course. The full training data had 7,500 observations and 10 features. Aside from the bootstrap sample size and number of features sampled, all other hyperparameters remained the same. The accuracy score is from each model's performance on the test data.

	Bootstrap sample size	Features sampled	Accuracy score	Runtime
Bagging:	100%	10	0.8596	15m 49s
Bagging:	30%	10	0.8692	7m 41s
Random forest:	100%	4	0.8704	8m 19s
Random forest:	30%	4	0.8736	4m 53s
Random forest:	5%	4	0.8652	3m 41s

The bagging model with only 30% bootstrapped samples performed better than the one that used 100% samples, and the random forest model that used 30% bootstrapped samples and just 4 features performed better than all the others. Not only that, but runtime was cut by nearly 70% using the random forest model with 30% bootstrap samples.

It may seem counterintuitive, but you can often build a well-performing model with even lower bootstrapping samples. Take for example the above random forest model whose base learners were each built from just 5% samples of the training data. It still was able to achieve a 0.8652 accuracy score—not much worse than the champion model!

Key takeaways

Random forest builds on bagging, taking randomization even further by using only a fraction of the available features to train its base learners. This randomization from sampling often leads to both better performance scores and faster execution times, making random forest a powerful and relatively simple tool in the hands of any data professional.

Resources for more information

More detailed information about random forests can be found here.

- scikit-learn documentation:
  - Random forest classifier: Documentation for model used for classification tasks
  - Random forest regressor: Documentation for model used for regression tasks

---