Bagging, Random Forests, Extra Trees

Decision Trees

Decision trees have a number of **downsides**:

- Overfit easily
- Trouble dealing with correlated predictors

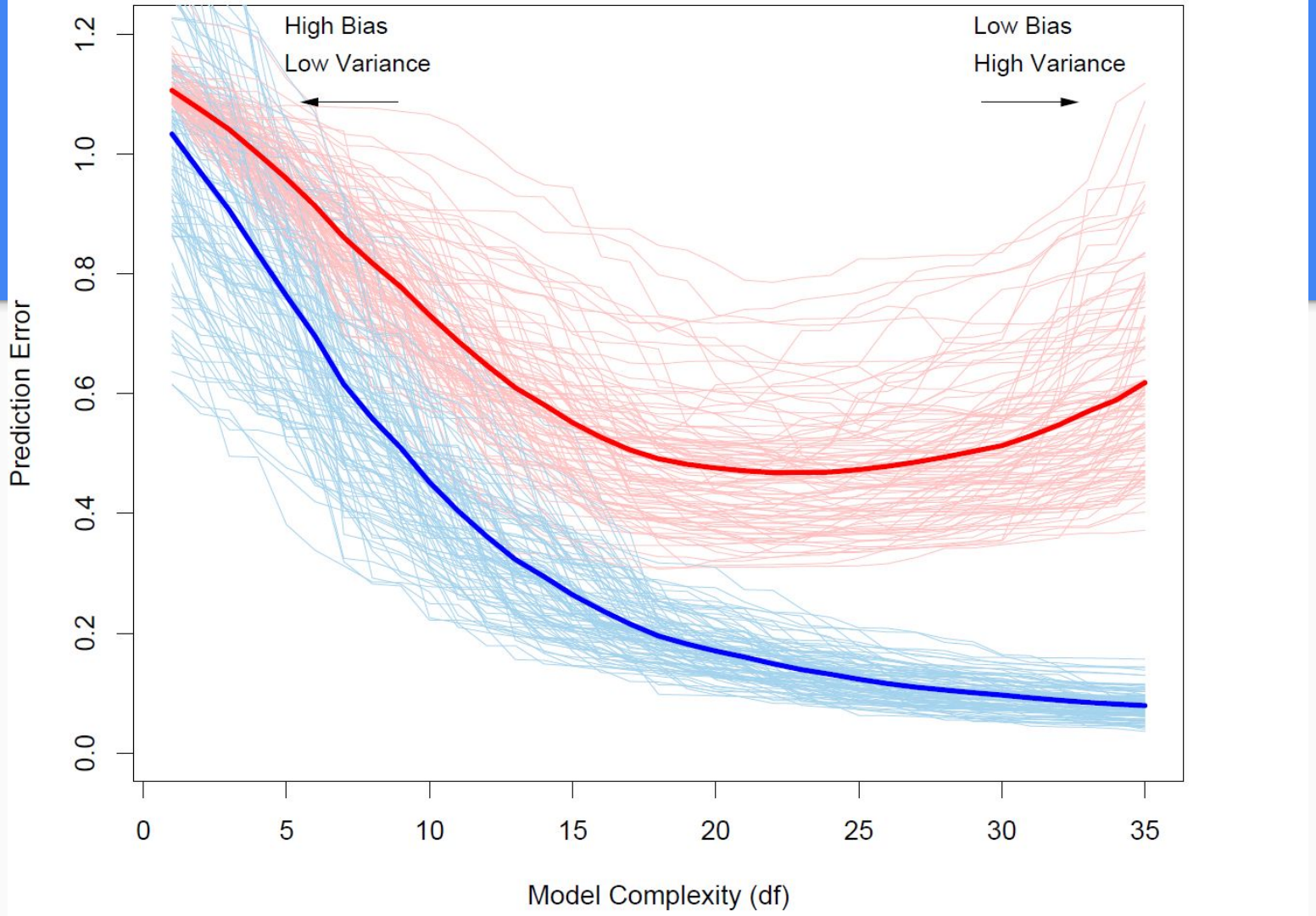
But also some **upsides**:

- Non-parametric
- Flexible

Idea Behind Ensemble Methods

Averaging over many trees **reduces variance** (adjusting for the tendency to overfit).

Randomizing predictors reduces issues with **correlated predictors.**


High Bias
Low Variance
Low Bias
High Variance
Prediction Error
0.0
0.2
0.4
0.6
0.8
1.0
1.2
0
5
10
15
20
25
30
35
Model Complexity (df)

Bagging, Random Forests, Extra Trees

Bagging randomizes over the **rows** of a dataset (by taking bootstrap samples of observations).

Random Forests randomizes over **rows AND columns** of a dataset (by also randomly choosing subsets of predictors to use).

Extra Trees goes one step further by randomizing splits (further reduce correlation between trees).

When to use which method

Remember: Goal is to find the best predictions.

There's nothing that says you can't **use all of them and choose the best one based on test error.**

The **ranger** package will include random forests and extra trees as part of its tuning process.

Out-of-bag Error

Bootstrapping results in a sample that looks like a new sample from the population but only contains about .632 of the original sample (there will be repeats).

The rest aren't used in building the model, and thus can possibly be used to estimate error.

Note: This is different from cross-validation because we aren't setting aside folds – an observation is not guaranteed to be in the test set once.

Out-of-bag Error

Calculating out-of-bag error:

1. Generate predictions for case i using models in which case i was not in training set.
2. Average over all such models to get test error for case i.
3. Repeat for all cases.

Bagging

Bagging can be done with any **base learner** (not just trees). For example, bagging with knn (sometimes called bnn).

Benefits of this vary (not much benefit for knn).

Computational Needs

Trees are generally **fast** to fit (especially with fewer variables to make splits on).

More trees increases computational needs, but generally should be quick.

Note: Easy to **parallelize** if needed (trees are not dependent on each other).