Random Forest

- uses bagging on decorellated decision trees for regression or classification.
- decision trees are low bias + high variance models and can thus benefit from bagging (moreover, they are quick to train)

- an ensemble method

bagging

① bootstrap a sample of the original dataset size from the dataset
② train a model, M_i
③ repeat for N models
④ regression: avg. results ; classification: take majority vote

$$\mathrm{Var}[\bar{f}(x)] = \mathrm{Var}\left[\frac{1}{N}\sum_{i=1}^{N} f_i(x)\right] = \frac{1}{N^2}\left\{\sum_i \overbrace{\mathrm{Var}[f_i(x)]}^{=\sigma^2} + 2\sum_{i<j} \overbrace{\mathrm{corr}(f_i(x), f_j(x))}^{=\rho} \sqrt{\mathrm{Var}[f_i(x)]}\sqrt{\mathrm{Var}[f_j(x)]}\right\}$$

$$= \frac{1}{N^2}\left\{N\sigma^2 + 2\binom{N}{2}\rho\sigma^2\right\} = \frac{\sigma^2}{N} + \frac{N-1}{N}\rho\sigma^2 \rightsquigarrow$$

$\Rightarrow$ increasing N decreases variance to a limiting value of $\rho\sigma^2$ ($\rho \in [0,1]$); better than the variance of a single model σ^2

bias remains unchanged:

$$\text{Bias}[\bar{f}(x)] = \text{Bias}\left[\frac{1}{N}\sum_i f_i(x)\right] = E\left[\frac{1}{N}\sum f_i(x)\right] - f_{\text{true}}(x)$$

$$= E[f_1(x)] - f_{\text{true}}(x)$$

$$= \text{Bias}[f_1(x)]$$

building a decision tree

- algo is a greedy process where we intend to reduce squared loss (for regression) and 0-1 loss for misclassification
  - recursively partition the feature space w/ binary splits on a certain feature, at a certain value
  - the feature, and feature value are found by greedily minimizing squared loss ($\hat{y}_R = \bar{y}_R$) for classification and gini index / cross-entropy for classification
    - gini index or cross entropy tend to result in more "pure" nodes rather than the misclassification rate, which would correspond to minimizing 0-1 loss

- stop building tree at a certain stopping criterion (eg. # data points in leaves)

RF Algo

repeat N times:

1. Bootstrap sample
2. build decision tree where you only consider a randomly chosen subset of features at each node along which to split. This reduces correlation amongst the trees (introduces randomness, & avoids all trees splitting on very predictive features), thus reducing variance.

3. Average preds in the end (regression), & the majority vote (classification)

OOB estimate

- each bootstrap contains $\approx 2/3 |D|$, so that for each data point, i, it is not used to train $\approx 1/3$ of the trains
- let $y^{(i)}_{oob}$ = mean prediction from $x^{(i)}$ over all trees not trained w/ i.
- $MSE_{oob} = \frac{\sum (y^{(i)} - y^{(i)}_{oob})^2}{n}$ ← mean; could technically be $< n$ if there are data points used in training every tree

Variable importance

$$\approx FI_j^t = \frac{\sum_{s:\, j \text{ is the splitting feature}} (\Delta RSS)_s}{RSS_j^t - RSS_0^t} \Rightarrow FI_j = \langle FI_j^t \rangle_t$$

(FI_j^t: t = tree t, j = feature j)

- Feature importance is roughly the decrease in the RSS for feature j for tree t, divided by total decrease in RSS (then averaged over all trees)
- similarly, use Δ(purity measure) for categoricals

Pros

- typically great performance
- easily parallelizable (just train each tree on a different core)
- difficult to over-fit
- sometimes slow

Cons

- not a regularized model (can lead to issues w/ high dims)
  - can fix w/ feature selection
- difficult to interpret