Tree-Based Method*

- we divide the predictor space (set of possible value $x_1, \dots x_p$) into J distinct and non overlapping regions $A_1, A_2, \dots A_J$. For every observation that falls into A_j, we make the same prediction, which is the mean of the response value for the training observation in A_j. we can give any shape but use high-dimensional rectangle to minimize $\sum_{j=1}^{J} \sum_{i \in A_j} (y_i - \hat{y}_{A_j})^2$. In order to perform recursive binary splitting, we consider all predictors $x_1, \dots x_j$ and all possible values of the cutpoint s for each of the predictors, and then choose the predictor and cutpoint such that the resulting tree has the lowest RSS. $\{X \mid x_j < s\}$ and $\{X \mid x_j > s\}$. Now we repeat by splitting one of the identified regions, and so forth till no regions contains more than S observations.
- we can built a large tree and then prune back.
  1. Use recursive binary splitting to grow a large tree on the training data, stopping only when each terminal node has fewer than some minimum number of observation
  2. Apply cost complexity pruning to the large tree in order to obtain a sequence of best subtree, as a function of α:
  3. Use K-fold cross validation to choose α. That is, divide the training observation into K folds: For each $k = 1, \dots K$:
     (a) Repeat Step 1 and 2 on all but the kth fold of the training data.
     (b) Evaluate the mean squared prediction error on the data in the left-out kth fold, as a function of α. Average the results for each value of α and pick α to minimize the average error
  4. Return the subtree from Step 2 that corresponds to the chosen value of α.

$\sum_{m=1}^{|T|} \sum_{i: x_i \in R_m} (y_i - \hat{y}_{R_m})^2 + \alpha |T|$

$|T|$ is the number of terminal node.

- For ~~prediction~~ classification we use the most commonly ~~used~~ occurring class. But we can't use RSS, but classification error rate $E = 1 - \max_k (\hat{p}_{mk})$ (often not sensitive enough)
  - Gini index $G = \sum_{k=1}^{K} \hat{p}_{mk} (1 - \hat{p}_{mk})$: measure of total variance, is small if $\hat{p}_{mk}$ close to 0 or 1. (node purity)
  - Entropy : $D = -\sum_{k=1}^{K} \hat{p}_{mk} \log \hat{p}_{mk}$

- linear regression of the form $f(x) = \beta_0 + \sum_{j=1}^{p} x_j \beta_j$ but tree of the form $f(x) = \sum_{m=1}^{M} c_m \cdot 1(x \in R_m)$. Hence if there is a highly non-linear and complex relationship between the features and the response, tree are better.
- Ensemble method (Bagging, Random forests, Boosting, Bayesian Additive Regression Trees) use weak learners to build better model.

- Bootstrap aggregation, or bagging is a general-purpose procedure for reducing the variance of a statistical learning method. we generate B different bootstrapped training data sets. we then train our method on the bth bootstrapped training set in order to get $\hat{f}^{*b}(x)$, and finally average all predictions
  → we grow deep trees but do not prune them → they have high variance but low bias. Average the B trees reduces the variance.

$\hat{f}_{bag}(x) = \frac{1}{B} \sum_{b=1}^{B} \hat{f}^{*b}(x)$

One can show that on average, each bagged tree makes use of around two-thirds of the observations. The remaining one-third of the observations not used to fit a given bagged tree are referred to as the out-of-bag observations. In order to obtain a single prediction for the ith observations, we can average (or majority) them. We can then compute a OOB MSE or classification error rate. It is a correct estimate of the test error for bagged model.
- In the case of bagging regression trees, we can record the total amount that the RSS (or Gini) is decreased due to splits over a given predictors, averaged over all B trees, large value = important predictor for classifi

Random forest provides an improvement over bagged trees by way of a small tweak that decorrelate the trees. As in bagging, we build a number of decision trees on bootstrapped training samples. But when building these decision trees, each time a split is considered, a random sample of m predictors is chosen as split candidates from the full set of p predictors. The split is allowed to use only one of these m predictors. $m \approx \sqrt{p}$, small m when we have a large number of correlated predictors.

Boosting trees are grown sequentially: each tree is grown using information from previously grown trees. Boosting does not involve bootstrap sampling: instead each tree is fit on a modified version of the original dataset.

```
1. Set $\hat{f}(x) = 0$ and $r_i = y_i$ for all i in the training set.
2. For b = 1, 2, ... B repeat:
    (a) Fit a tree $\hat{f}^b$ with d splits (d+1 terminal nodes) to the training data (X, r)
    (b) Update $\hat{f}$ by adding in a shrunken version of the new tree
        $\hat{f}(x) \leftarrow \hat{f}(x) + \lambda \hat{f}^b(x)$
    (c) update residuals
        $r_i \leftarrow r_i - \lambda \hat{f}^b(x_i)$
3. output the boosted model
    $\hat{f}(x) = \sum_{b=1}^{B} \lambda \hat{f}^b(x)$
```

- Use CV to select B.
- λ is learning rate - small λ may require large B.
- the number d of split controls the complexity

Bayesian Additive regression Trees is related to both approaches: each tree is constructed in a random manner (bagging, rf) and each tree tries to capture signal not yet accounted for by the current model (boosting). $\hat{f}_k^b(x)$ represents the prediction at x for the k^{th} regression tree used in the b iteration.

```
1. Let $\hat{f}_1^1(x) = \dots \hat{f}_K^1(x) = \frac{1}{nK} \sum_{i=1}^{n} y_i$
2. compute $\hat{f}^1(x) = \sum_{k=1}^{K} \hat{f}_k^1(x) = \frac{1}{n} \sum_{i=1}^{n} y_i$
3. For b = 2, ..., B
    (a) for k = 1, 2, ... K
        i. For i = 1, ... n compute the current partial residual
            $r_i = y_i - \sum_{k'<k} \hat{f}_{k'}^b(x_i) - \sum_{k'>k} \hat{f}_{k'}^{b-1}(x_i)$
        ii. Fit a new tree, $\hat{f}_k^b(x)$, to $r_i$, by randomly perturbing the kth tree from the previous iteration $\hat{f}_k^{b-1}(x)$. Perturbation that improve fit are favored
    (b) compute $\hat{f}^b(x) = \sum_{k=1}^{K} \hat{f}_k^b(x)$
4. compute the mean after L burn-in samples $\hat{f}(x) = \frac{1}{B-L} \sum_{b=L+1}^{B} \hat{f}^b(x)$
```

- change structure → add or prune a branch
- we may change the prediction in each terminal node of the tree.

- bagging → tree tends to be similar → can get caught in local optima
- random forest → more thorough exploration of model space relative to bagging

In BART, we use only the original data, and we grow tree successively. However each tree is perturbed in order to avoid local optima and achieve a more thorough exploration of space. We try to improve the fit to the current partial residual by slightly modifying the tree obtained. This guard against overfitting since it limits how "hard" we fit the data in each iteration