Classification and Regression Tree (CART) Based Methods

Objectives

- Stratify or segment the feature space $(X_1, X_2, \dots, X_P)$ into simple regions $(R_1, R_2, \dots, R_J)$ for prediction or classification purposes of the objects with respect to the region R_j within which an object i falls.
- Explain the splitting rules and concept of decision trees, which are used to predict the mean, mode or class membership of the object in each feature region.
- Introduce enhancements of improve the prediction accuracy and MSE by producing multiple trees and then use a consensus prediction: *bagging*, *random forests*, and *boosting*.

Regression Trees

- Example with two key variables and the dependent variable on the logarithmic scale:

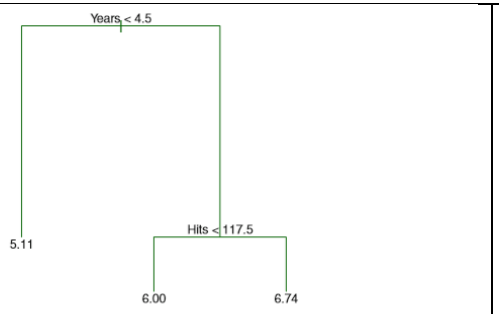


FIGURE 8.1. *For the* `Hitters` *data, a regression tree for predicting the log salary of a baseball player, based on the number of years that he has played in the major leagues and the number of hits that he made in the previous year. At a given internal node, the label (of the form $X_j < t_k$) indicates the left-hand branch emanating from that split, and the right-hand branch corresponds to $X_j \geq t_k$. For instance, the split at the top of the tree results in two large branches. The left-hand branch corresponds to* `Years<4.5`*, and the right-hand branch corresponds to* `Years>=4.5`*. The tree has two internal nodes and three terminal nodes, or leaves. The number in each leaf is the mean of the response for the observations that fall there.*

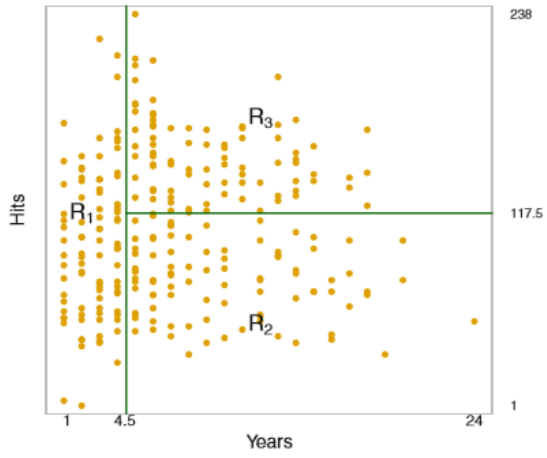


FIGURE 8.2. *The three-region partition for the* `Hitters` *data set from the regression tree illustrated in Figure 8.1.*

- Terminology:

  - The terminal nodes are called ***leaves***.
  - The nodes where the tree splits are called ***internal nodes***.
  - Segments of trees are called ***branches***.
- Interpretation:
  - The top node denotes the split of the data at a partition value c for the most important variable X_p discriminating between different salary levels.
  - A node heading a sub-branch splits a regions binary according to further important variable.
- Underlying idea:
  - Divide the P-dimensional feature space $(X_1, X_2, \dots, X_P)$ into J non-overlapping rectangular regions $(R_1, R_2, \dots, R_J)$.
  - Assign to each object i with its particular set of feature values $(x_{i1}, x_{i2}, \dots, x_{iP})$ to that region into which it falls and use a region's characteristic (e.g., mean, median or dominant class) as predictor $\hat{y}_{R_j}$ for y_i.
- In regression CART the objective is to find the regions $(R_1, R_2, \dots, R_J)$ for which the RSS is at its minimum:

$$RSS = \min_{R_1, R_2, \dots, R_J} \sum_{j=1}^{J} \sum_{i \in R_j} \left(y_i - \hat{y}_{R_j}\right)^2$$

- It is computational infeasible to evaluate all possible partitions into J regions. Therefore, a ***top-down***, ***greedy***[1] search strategy is engaged (remember stepwise forward regression analysis):
  - Each existing region R_j is ***recursively*** and ***binary*** split by one feature X_p at a time to constitute two new sub-regions $R_{j[1]}$ and $R_{j[2]}$ within it.

[1] Greedy because the algorithm looks just at one splitting step rather than the RSS of a completed possible tree.

- The best splitting point $\hat{c}_{pj}$ for X_p that minimizes the sum of the squared deviations in the newly generated sub-rectangles R_j, and $R_{j'}$, with $R_j = R_j \cup R_{j'}$, usually must be identified by a grid search algorithm.
- Repeat the evaluation of RSS for all X_p in all existing R_j and select the split point $\hat{c}_{pj}$ for which RSS minimized.
- The stepwise approach implies that at each step $J \times P$ calculations of $\hat{c}_{pj}$ and evaluations of the RSS need to be performed
- Stop splitting a R_j once its number of objects n_j drops below a threshold value.

Tree Pruning

- The stopping rule - the number of objects n_j drops below a threshold value - usually ***over-fits*** the training dataset and leads to an overly complex tree with little remaining degrees for freedom.
- Therefore, a smaller tree with fewer splits might lead to a reduction of the ***variance*** at the expense of a slight increase in the ***bias***.
- A possible strategy is to split a tree until the RSS or classification error shrinks only marginally. However, this criterion may miss potential subsequent splits which will lead to substantial improvements.
- Analogue to ***backwards*** stepwise regression a tree can pruned back at the leaves of a fully grown tree until an optimality criterion is satisfied.
- Technically, a constrained optimization is performed in which the hyper-parameter α controls the degree of pruning from a fully grown tree T_0 down to a tree T with $T \subset T_0$:

$$\min \sum_{j=1}^{|T_\alpha|} \sum_{x_i \in R_j} \left(y_i - \hat{y}_{R_j}\right)^2 + \underbrace{\alpha \cdot |T_\alpha|}_{penalty\ of\ complexity}$$

The number of leaves in a tree is denoted by $|T|$.

- An $\alpha = 0$ will lead to the full-grown tree T_0.
- For a given dataset and varying α the trees form subsets with $T_{\alpha_2} \subset T_{\alpha_1}$ for $\alpha_1 < \alpha_2$.
- K-fold cross-validation can be used to estimate the tuning parameter α:

Algorithm 8.1 *Building a Regression Tree*

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 observations.

2. Apply cost complexity pruning to the large tree in order to obtain a sequence of best subtrees, as a function of α.

3. Use K-fold cross-validation to choose α. That is, divide the training observations into K folds. For each $k = 1, \ldots, K$:

   (a) Repeat Steps 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.

- Example:

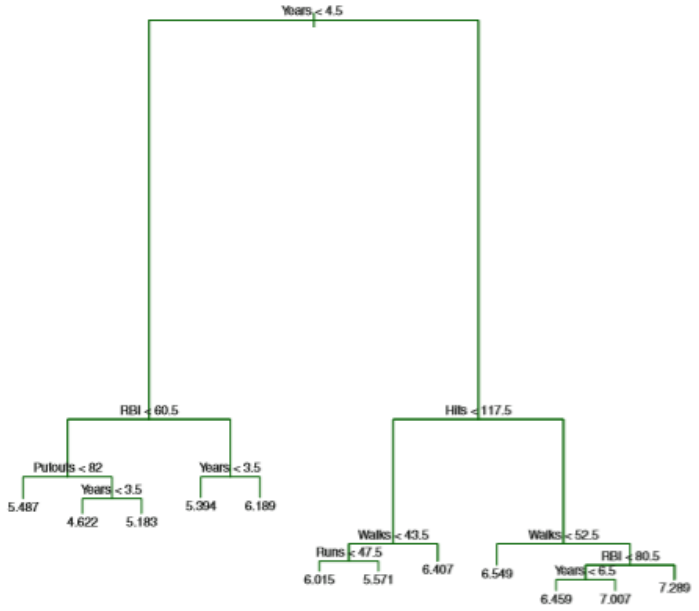


FIGURE 8.4. *Regression tree analysis for the* `Hitters` *data. The unpruned tree that results from top-down greedy splitting on the training data is shown.*

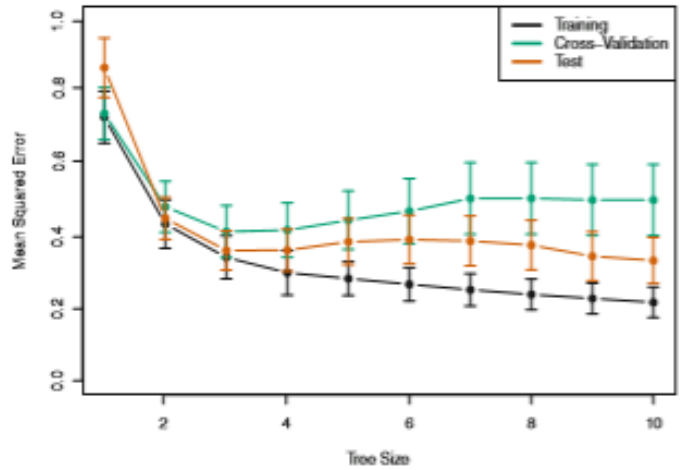


FIGURE 8.5. *Regression tree analysis for the* `Hitters` *data. The training, cross-validation, and test MSE are shown as a function of the number of terminal nodes in the pruned tree. Standard error bands are displayed. The minimum cross-validation error occurs at a tree size of three.*

Classification Trees

- For classifications the predicted value of $\hat{y}_i \in R_j$ needs to be revised. A common approach is to assign an object i to the class k for which the probability p_{kj} is the largest.
- The probability p_{kj} is estimated by the proportion of objects in R_j which belong to the k-th class:

$$\hat{p}_{kj} = \frac{1}{n_j} \cdot \sum_{x_i \in R_j} I(y_i = k)$$

- A natural decision rule to find the best classification tree is to minimize the total error rate. Using this all or nothing decision rule however is not sensitive enough to grow meaningful trees.

- Possible error rates within each region R_j are:
  - Misclassification error: $E_j = 1 - \max_k \hat{p}_{kj}$
  - Gini index: $G_j = \sum_{k=1}^{K} \hat{p}_{kj} \cdot \left(1 - \hat{p}_{kj}\right)$
  - Cross-entropy index: $D_j = -\sum_{k=1}^{K} \hat{p}_{kj} \cdot \ln \hat{p}_{kj}$
- Node purity in the region R_j, i.e., for one class k with probability $\hat{p}_{kj} \approx 1$ or $\hat{p}_{kj} \approx 0$, leads to small error values of E_j, G_j and D_j.
- Usually the total error rate is used to report the predictive quality of a tree, whereas the Gini or entropy indices are used to build a tree.

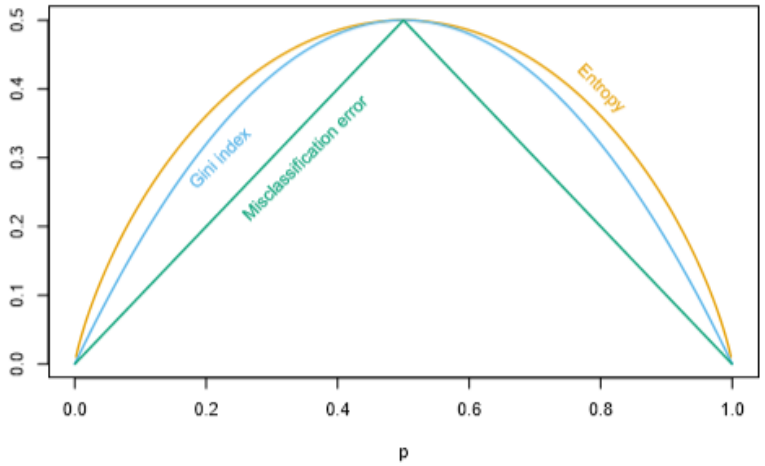


FIGURE 9.3. *Node impurity measures for two-class classification, as a function of the proportion p in class 2. Cross-entropy has been scaled to pass through* $(0.5, 0.5)$.

- The features X_j can also be ***categorical variables*** with a split distinguishing one set of factor levels against the remaining set of factor levels. Usually, the split is one factor level against all remaining factor levels.

- Example of a classification tree:

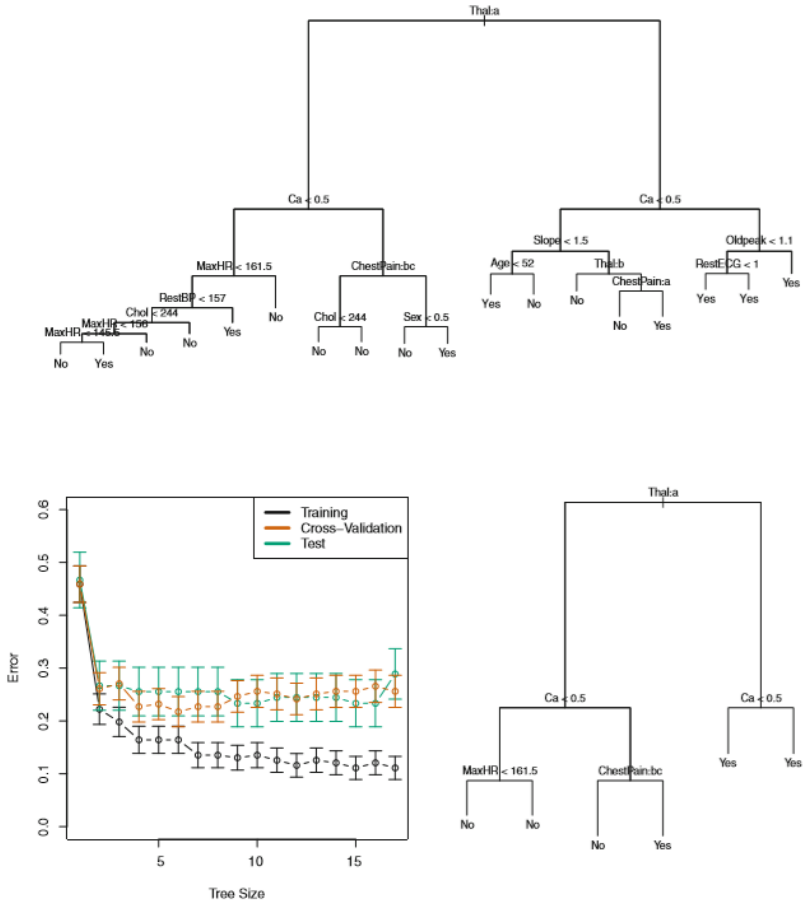


FIGURE 8.6. `Heart` *data.* Top: *The unpruned tree.* Bottom Left: *Cross -validation error, training, and test error, for different sizes of the pruned tree.* Bottom Right: *The pruned tree corresponding to the minimal cross-validation error.*

- Note: some splits, e.g., **`RestEGG < 1`** or **`Chol< 244`**, share the same class for all-or-nothing predictions but they different in their node purities. Thus, one branch discriminates better between the classes than

the other branch.
For these splits the overall error rate will not improve. However, allowing for a split with identical predictions will improve the Gini- and entropy-measures.

Advantages and Disadvantages of Trees

- Advantage: The outcome of trees is easier explain to “decision makers” than regression techniques.
- Advantage: The stepwise approach of decision trees has more in common with an assumed hierarchical lexicographic decision-making process of humans.
- Advantage: Moderately sized trees can be easily visualized and communicated.
- Advantage: Trees handle categorical variables well.
- Advantage: Trees will not consider irrelevant features.
- Disadvantage: Standard pruned trees usually do not achieve the same predictive accuracy than other regression or classification methods

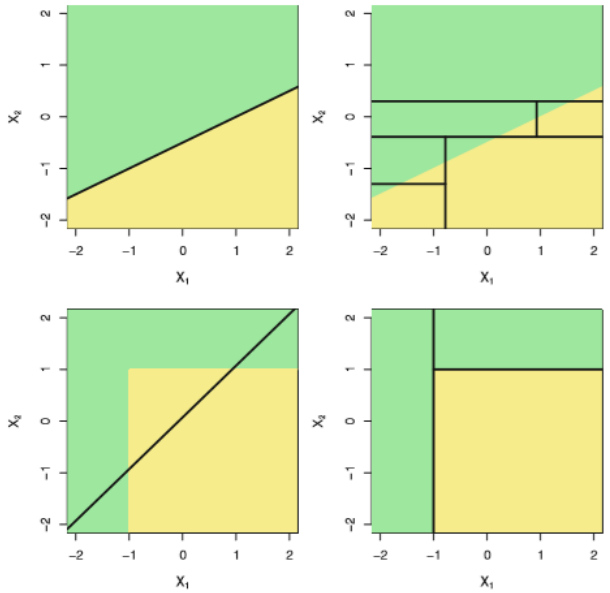


FIGURE 8.7. Top Row: *A two-dimensional classification example in which the true decision boundary is linear, and is indicated by the shaded regions. A classical approach that assumes a linear boundary (left) will outperform a decision tree that performs splits parallel to the axes (right).* Bottom Row: *Here the true decision boundary is non-linear. Here a linear model is unable to capture the true decision boundary (left), whereas a decision tree is successful (right).*

- Disadvantage: Trees can be very volatile from sample to sample, i.e., they exhibit a high variance. This leads to refined tree-building strategies aimed at reducing the variance.

Refined Tree Building Strategies

Bagging: Bootstrap aggregation of trees

- Decision trees usually suffer from high variance, which makes any prediction unreliable.
- It is well-known from the central limit theorem that the average of several ***independent*** and identically distributed sample observations will have variance smaller standard error.
  Proof for independent sample objects:

$$Var\left(\frac{1}{n}\cdot\sum_{i=1}^{n}X_i\right)=\frac{1}{n^2}\cdot\sum_{i=1}^{n}\underbrace{Var(X_i)}_{=\sigma^2}=\frac{1}{n^2}\cdot n\cdot\sigma^2=\frac{\sigma^2}{n}$$

- Bagging actually makes use of the variability of a trees, which were generated by different samples. The samples are obtained by bootstapping.
- Bagging estimates the predicted value for any combination of $(X_1, X_2, \dots, X_P)$ by averaging several sampled trees.
- A set of B sample trees is obtained using the bootstrap sampling approach (resampling of the sample observation until n samples are drawn) of the n training observations. This generates on sample tree $\hat{f}_b$ for each sample. Since the sample objects are different each sample tree $\hat{f}$ will be different.
- For metric prediction outcomes the predicted value will be the average of the predictions for the sample trees:

$$\bar{Y}|X_1, X_2, \dots, X_P=\frac{1}{B}\cdot\sum_{b=1}^{B}\hat{f}_b(X_1, X_2, \dots, X_P)$$

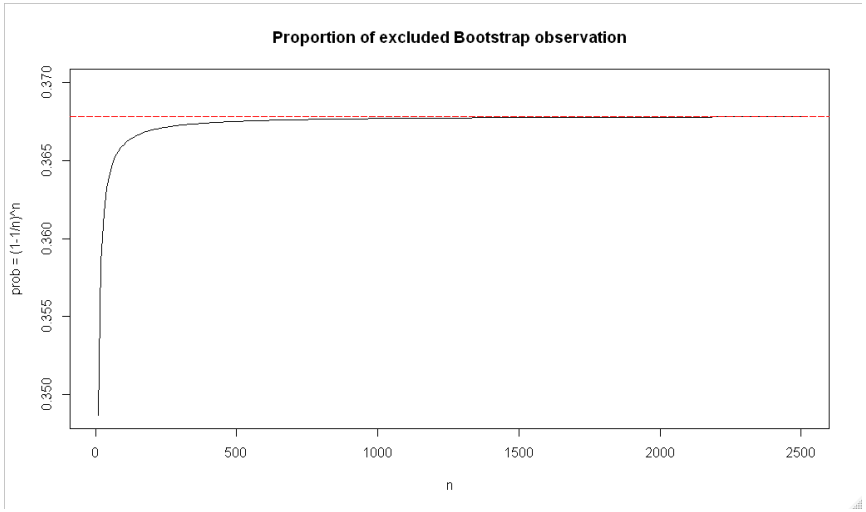
- For categorical outcomes the predicted class of each sample tree $\hat{f}_b$ is used and then the majority vote of the individual sample predictions determines the final class assignment.
- In bagging, the trees are usually grown deep, which reduces any bias, but increases the variance. The variance, however, shrinks subsequently by averaging a large number of bootstrapped sample trees.

- A disadvantage of bagged trees is that the multitude of generated trees can no longer be meaningful interpreted.

Out-of- Bag (OOB) Error Estimation

- Bagging has a further advantage that for each sample tree by default a test dataset will be available. Thus neither cross-validation needs to be applied nor objects withheld for a test dataset.
- Approximately $1/3$ of the observations are not included in the training bootstrap sample, because the probability of an object not being in a bootstrap sample of size n is:

$$\lim_{n\to\infty}\left(1-\frac{1}{n}\right)^{n} \approx 0.3678$$



- This will give approximately $B/3$ predictions for each observation, which are independent of the training samples. These are independent predictions are averaged to evaluate the performance of the bagged trees for a particular dataset.

- Comparison of bagging and out-of-bag model fit:

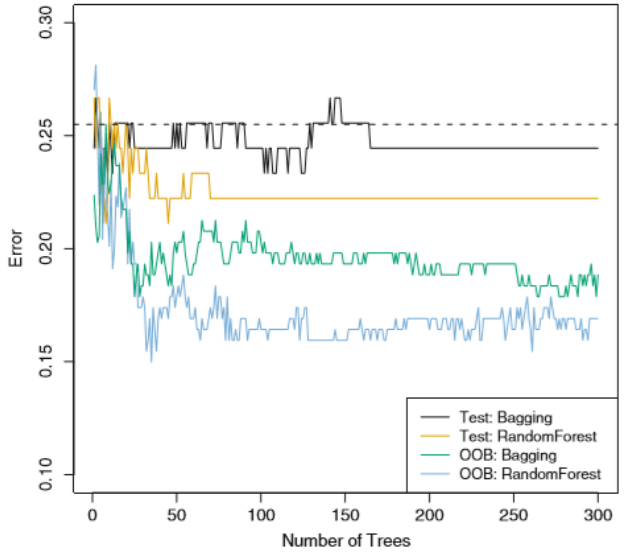


FIGURE 8.8. *Bagging and random forest results for the* `Heart` *data. The test error (black and orange) is shown as a function of* B*, the number of bootstrapped training sets used. Random forests were applied with* $m = \sqrt{p}$*. The dashed line indicates the test error resulting from a single classification tree. The green and blue traces show the OOB error, which in this case is considerably lower.*

Variable Importance Measures

- For each feature, the average reduction in the *deviance*, *RSS*, *cross-entropy* or *Gini* index over all sample trees can be averaged over every split of a given feature in each sample tree.
- Subsequently the features can be ranked according to their average reduction over all bagged sample trees.

Random Forest

- Many bootstrapped trees are similar to each other and therefore positively correlated.

only part of independent variables are considered for each tree

- For positively correlated trees the variance of their average prediction is inflated by the degree of intercorrelation.

$$Var\left(\frac{1}{n}\cdot\sum_{i=1}^{n}X_i\right)=\frac{1}{n^2}\cdot\begin{pmatrix}1 & 1 & \cdots & 1\end{pmatrix}\cdot\begin{pmatrix}\sigma_1^2 & \sigma_{12} & \cdots & \sigma_{1n}\\ \sigma_{21} & \sigma_2^2 & \cdots & \sigma_{2n}\\ \vdots & \vdots & \ddots & \vdots\\ \sigma_{n1} & \sigma_{n2} & \cdots & \sigma_n^2\end{pmatrix}\cdot\begin{pmatrix}1\\1\\\vdots\\1\end{pmatrix}$$

- To break the correlation among the sample trees up, only a random subset of features is considered at each split. Usually the set of randomly selected features is of size $m \approx \sqrt{P}$.

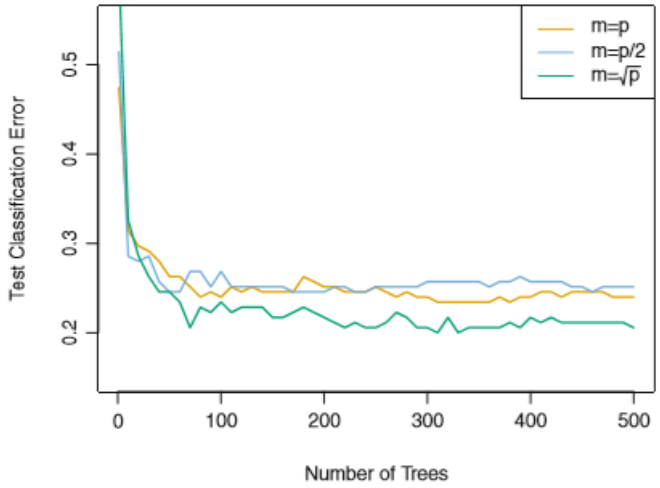


FIGURE 8.10. *Results from random forests for the 15-class gene expression data set with $p = 500$ predictors. The test error is displayed as a function of the number of trees. Each colored line corresponds to a different value of m, the number of predictors available for splitting at each interior tree node. Random forests ($m < p$) lead to a slight improvement over bagging ($m = p$). A single classification tree has an error rate of 45.7%.*

- Consequently, strong predictors will appear at varying levels of the sample trees and weaker features will have a chance to appear earlier in the tree. This will ***decorrelates the sample trees*** and makes the covariance matrix more diagonal.

Boosting

- Boosting grows a set of trees sequentially with information that is partially independent of previously generated trees.
- Its growths a final tree slowly and thus can avoid the problem of overfitting.
- This is accomplished by using residuals from the tree generated at the previous step. These residuals are uncorrelated with the information used in any of the previously generated trees.
  Recall from regression analysis: the residuals are uncorrelated with all exogenous variables in the regression model and they capture the unexplained information.
- Consequently, trees generated with boosting dependent heavily on the structure of the tree, which was generated at previous steps.
  To reduce the rate of learning (impact of the residuals) mixed predictions of trees from the previous step with the current step.

Algorithm 8.2 *Boosting for Regression Trees*

1. Set $\hat{f}(x) = 0$ and $r_i = y_i$ for all i in the training set.
2. For $b = 1, 2, \ldots, B$, 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:
   $$\hat{f}(x) \leftarrow \hat{f}(x) + \lambda \hat{f}^b(x). \tag{8.10}$$
   (c) Update the residuals,
   $$r_i \leftarrow r_i - \lambda \hat{f}^b(x_i). \tag{8.11}$$
3. Output the boosted model,
   $$\hat{f}(x) = \sum_{b=1}^{B} \lambda \hat{f}^b(x). \tag{8.12}$$

- Comparison of algorithms:

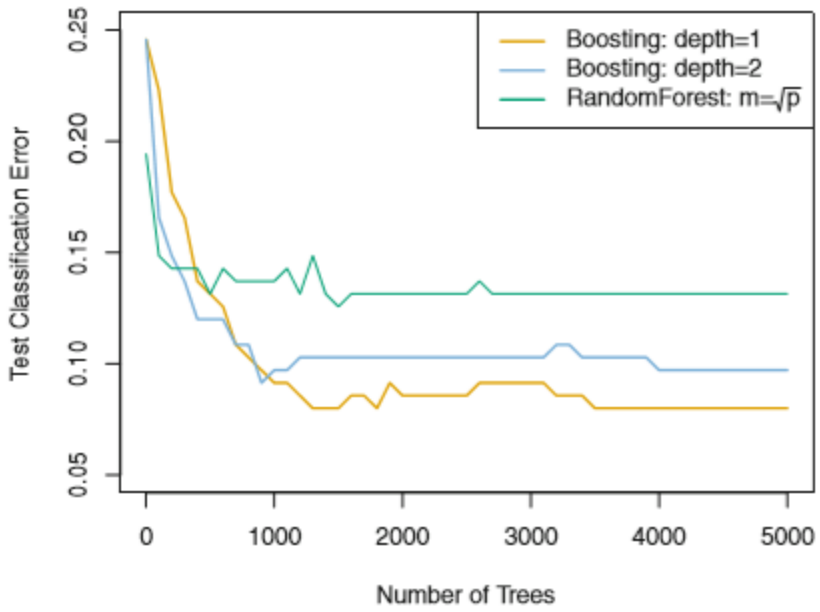


FIGURE 8.11. *Results from performing boosting and random forests on the 15-class gene expression data set in order to predict* cancer *versus* normal. *The test error is displayed as a function of the number of trees. For the two boosted models,* $\lambda = 0.01$. *Depth-1 trees slightly outperform depth-2 trees, and both outperform the random forest, although the standard errors are around 0.02, making none of these differences significant. The test error rate for a single tree is 24 %.*