Class 10 – Tree based methods, Bagging and Random Forest (Regression)

Pedram Jahangiry

Fall 2019

JON M.
HUNTSMAN
SCHOOL OF BUSINESS
UtahStateUniversity
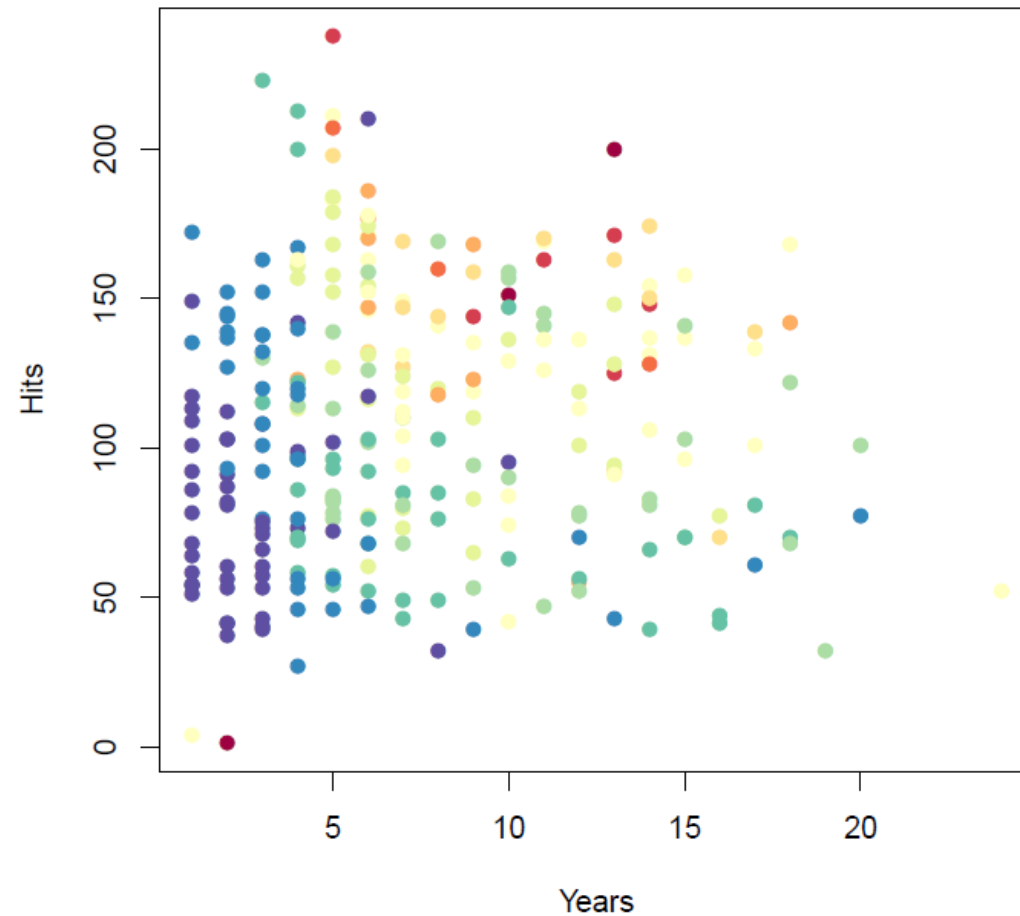
Tree-Based methods

- Here we describe *tree-based* methods for regression and classification.

- These involve *stratifying* or *segmenting* the predictor space into a number of simple regions.

- Since the set of splitting rules used to segment the predictor space can be summarized in a tree, these types of approaches are known as *decision-tree* methods.

Pros and Cons

- Tree-based methods are simple and useful for interpretation.

- However they typically are not competitive with the best supervised learning approaches in terms of prediction accuracy.

- Hence we also discuss *bagging*, *random forests*, and *boosting*. These methods grow multiple trees which are then combined to yield a single consensus prediction.

- Combining a large number of trees can often result in dramatic improvements in prediction accuracy, at the expense of some loss interpretation.
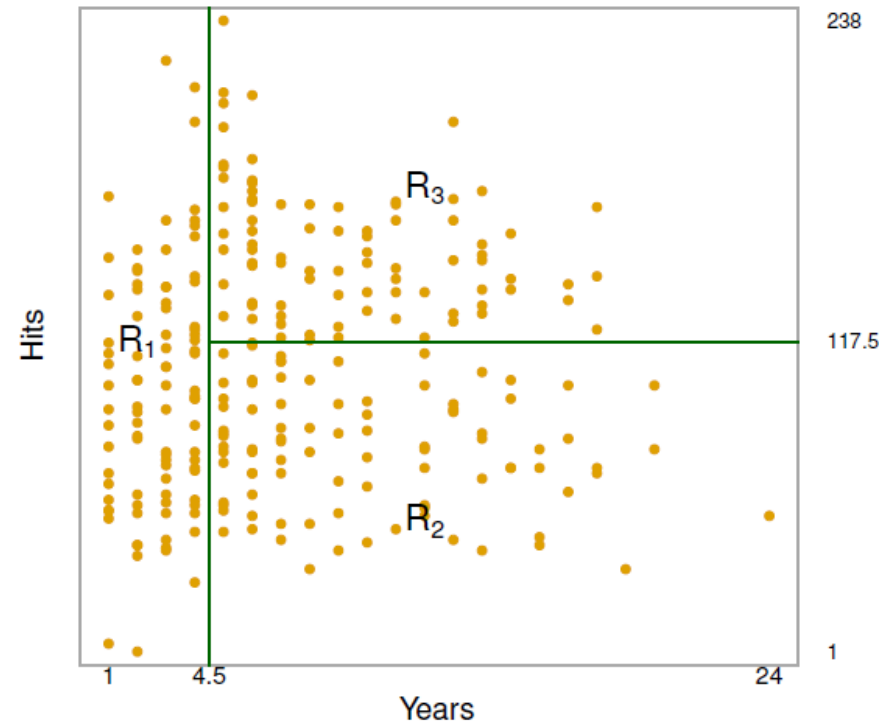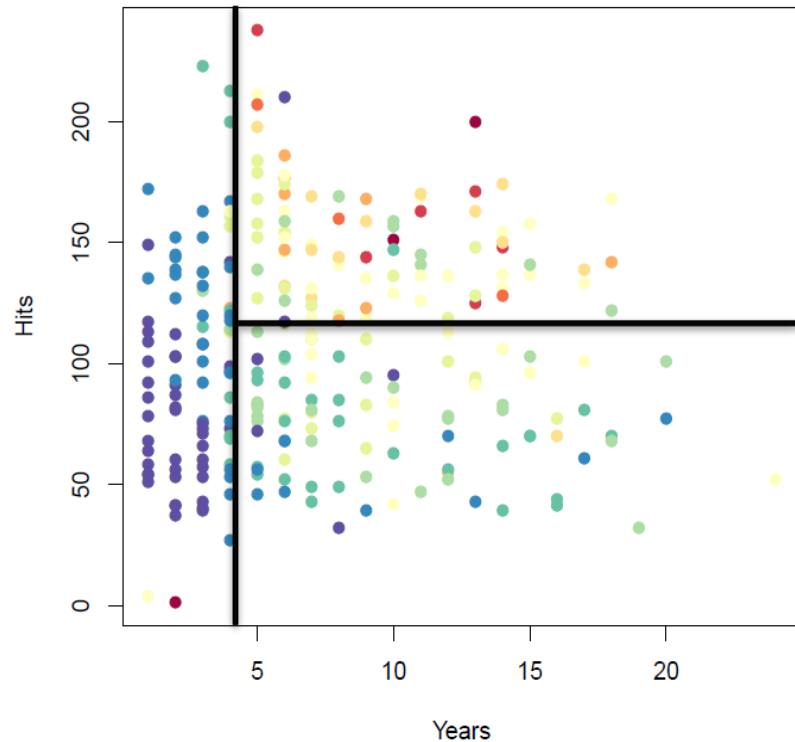
3

Tree based Regression

Baseball Salary is color-coded from low (blue, green) to high (yellow, red)

Tree based Regression

- Overall, the tree stratifies or segments the players into three regions of predictor space: $R_1 = \{X \mid \texttt{Years} < 4.5\}$, $R_2 = \{X \mid \texttt{Years} >= 4.5, \texttt{Hits} < 117.5\}$, and $R_3 = \{X \mid \texttt{Years} >= 4.5, \texttt{Hits} >= 117.5\}$.

Tree-Based methods

- In keeping with the *tree* analogy, the regions R_1, R_2, and R_3 are known as *terminal nodes*

- Decision trees are typically drawn *upside down*, in the sense that the leaves are at the bottom of the tree.

- The points along the tree where the predictor space is split are referred to as *internal nodes*

- In the hitters tree, the two internal nodes are indicated by the text `Years`<4.5 and `Hits`<117.5.

Interpretation of Results

- Years is the most important factor in determining Salary, and players with less experience earn lower salaries than more experienced players.

- Given that a player is less experienced, the number of Hits that he made in the previous year seems to play little role in his Salary.

- But among players who have been in the major leagues for five or more years, the number of Hits made in the previous year does affect Salary, and players who made more Hits last year tend to have higher salaries.

- Surely an over-simplification, but compared to a regression model, it is easy to display, interpret and explain
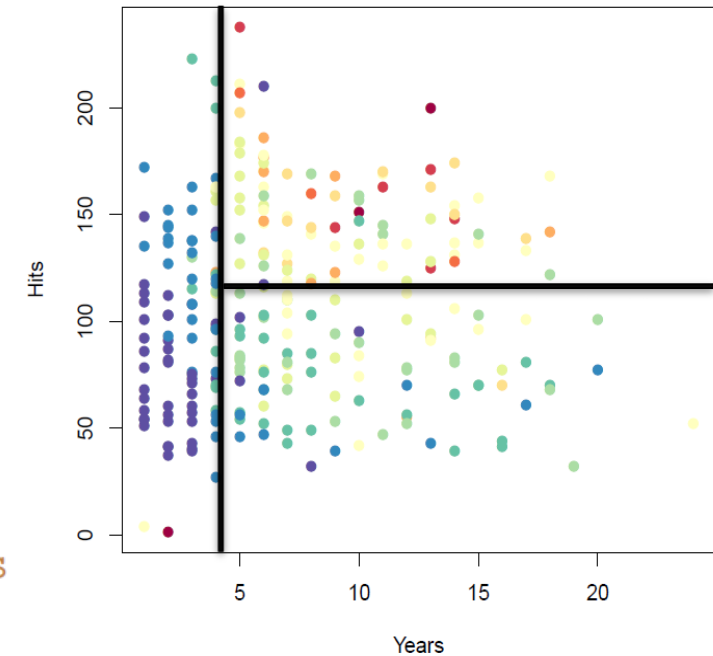
Tree building process
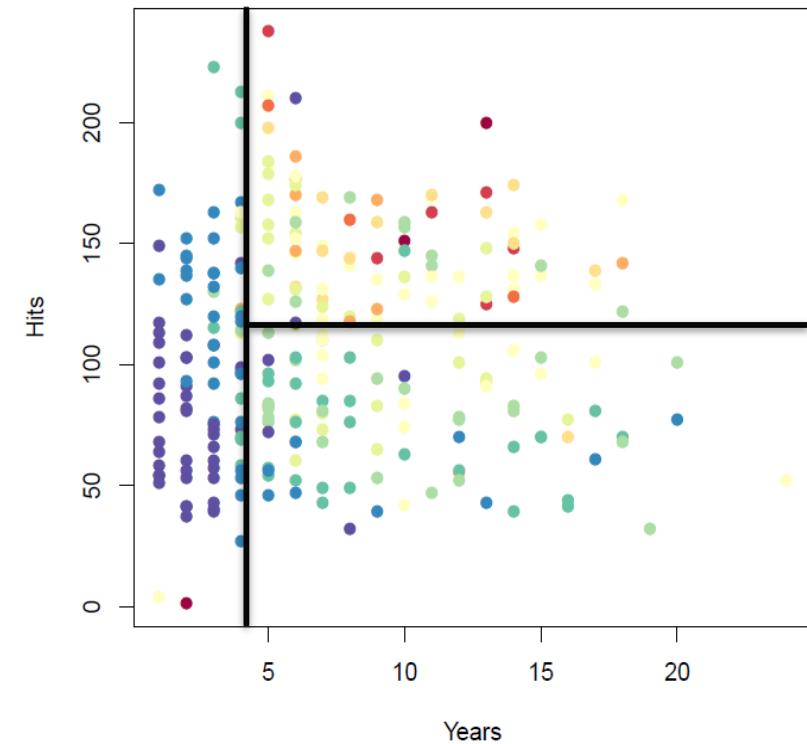
1. We divide the predictor space — that is, the set of possible values for $X_1, X_2, \ldots, X_p$ — into J distinct and non-overlapping regions, $R_1, R_2, \ldots, R_J$.

2. For every observation that falls into the region R_j, we make the same prediction, which is simply the mean of the response values for the training observations in R_j.

Tree building process

- In theory, the regions could have any shape. However, we choose to divide the predictor space into high-dimensional rectangles, or *boxes*, for simplicity and for ease of interpretation of the resulting predictive model.
- The goal is to find boxes $R_1, \ldots, R_J$ that minimize the RSS, given by

$$\sum_{j=1}^{J} \sum_{i \in R_j} (y_i - \hat{y}_{R_j})^2,$$

where $\hat{y}_{R_j}$ is the mean response for the training observations within the jth box.

the best split is made at that particular step, rather than looking ahead and picking a split that will lead to a better tree in some future step.
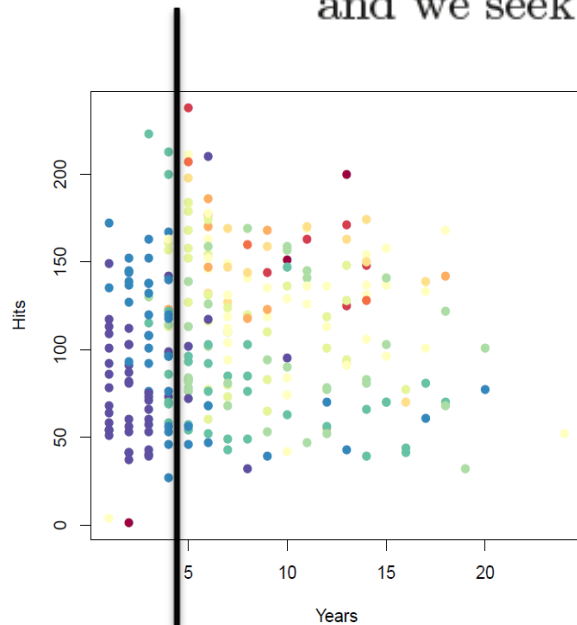
9

Tree building process (optimization)

- We first select the predictor X_j and the cutpoint s such that splitting the predictor space into the regions $\{X|X_j < s\}$ and $\{X|X_j \geq s\}$ leads to the greatest possible reduction in RSS.

$$R_1(j, s) = \{X|X_j < s\} \quad \text{and} \quad R_2(j, s) = \{X|X_j \geq s\},$$

and we seek the value of j and s that minimize the equation

$$\sum_{i:\ x_i \in R_1(j,s)} (y_i - \hat{y}_{R_1})^2 + \sum_{i:\ x_i \in R_2(j,s)} (y_i - \hat{y}_{R_2})^2,$$

Tree building process (optimization)

- Next, we repeat the process, looking for the best predictor and best cutpoint in order to split the data further so as to minimize the RSS within each of the resulting regions.

- However, this time, instead of splitting the entire predictor space, we split one of the two previously identified regions. We now have three regions.

- Again, we look to split one of these three regions further, so as to minimize the RSS. The process continues until a stopping criterion is reached; for instance, we may continue until no region contains more than five observations.
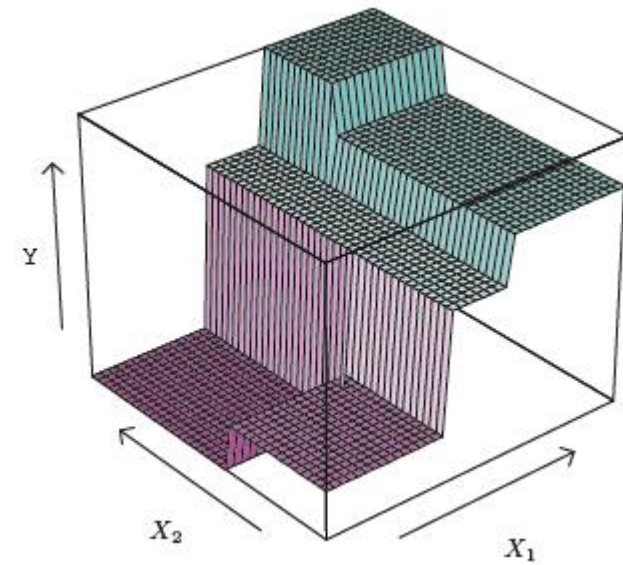
A five-region example of recursive binary splitting
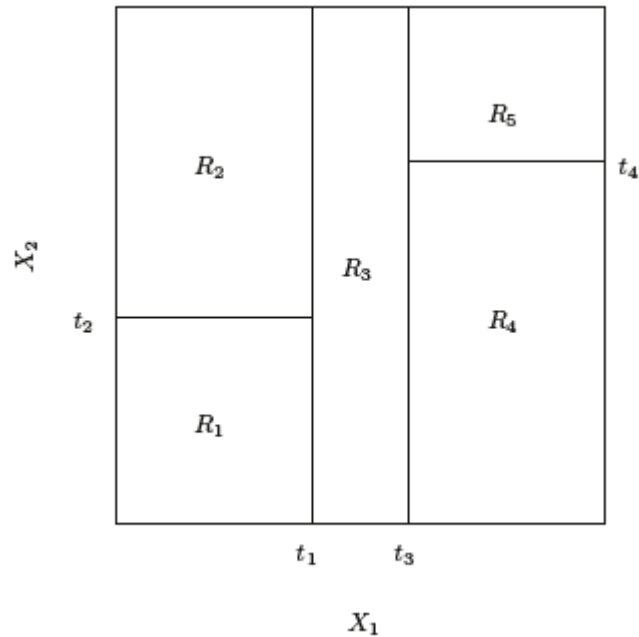
The output of recursive binary splitting on a two-dimensional example
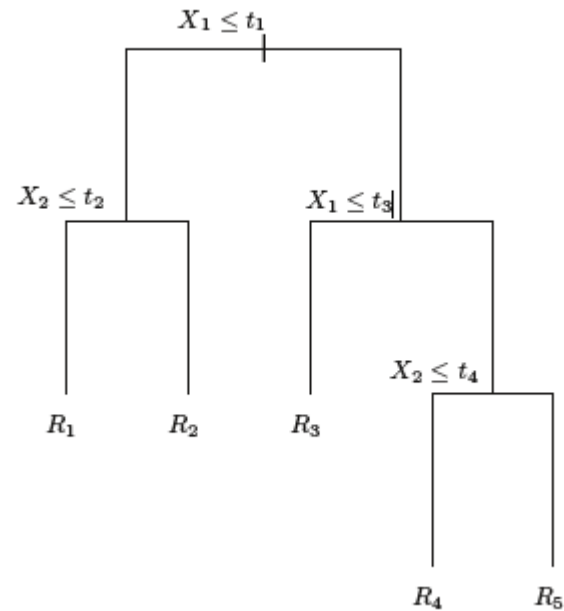
A tree corresponding to the partition in the left panel.

A perspective plot of the prediction surface corresponding to that tree.

Pruning a Tree

- The process described above may produce good predictions on the training set, but is likely to *overfit* the data, leading to poor test set performance. *Why?*

- A smaller tree with fewer splits (that is, fewer regions $R_1, \ldots, R_J$) might lead to lower variance and better interpretation at the cost of a little bias.

- This strategy will result in smaller trees, but is too *short-sighted:* a seemingly worthless split early on in the tree might be followed by a very good split — that is, a split that leads to a large reduction in RSS later on.

Pruning a Tree

- A better strategy is to grow a very large tree T_0, and then *prune* it back in order to obtain a *subtree*
- *Cost complexity pruning* — also known as *weakest link pruning* — is used to do this
- we consider a sequence of trees indexed by a nonnegative tuning parameter α. For each value of α there corresponds a subtree $T \subset T_0$ such that

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

is as small as possible. Here $|T|$ indicates the number of terminal nodes of the tree T, R_m is the rectangle (i.e. the subset of predictor space) corresponding to the mth terminal node, and $\hat{y}_{R_m}$ is the mean of the training observations in R_m.

Choosing the best subtree

- The tuning parameter α controls a trade-off between the subtree's complexity and its fit to the training data.

- We select an optimal value $\hat{\alpha}$ using cross-validation.

- We then return to the full data set and obtain the subtree corresponding to $\hat{\alpha}$.

Building a Regression Tree algorithm

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.
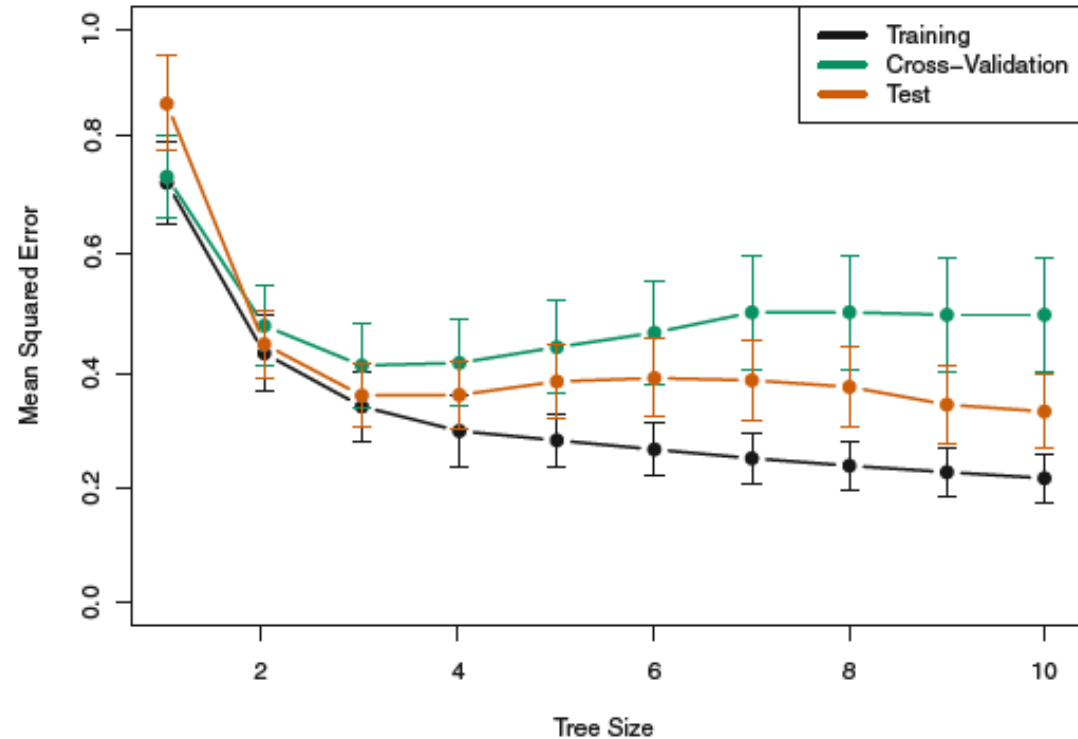
Salary example continued

The *unpruned* tree that results from *recursive binary splitting* on the training data
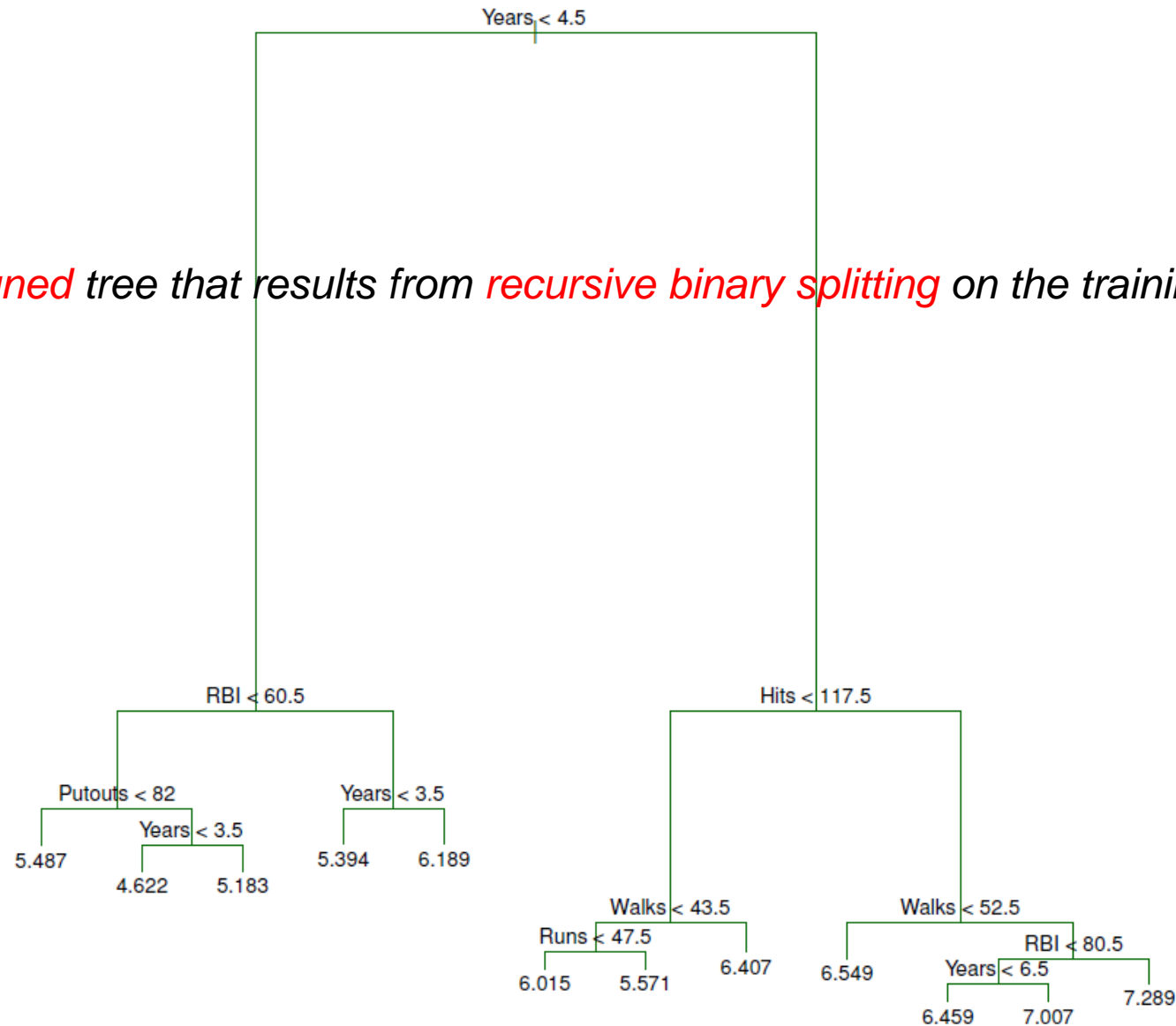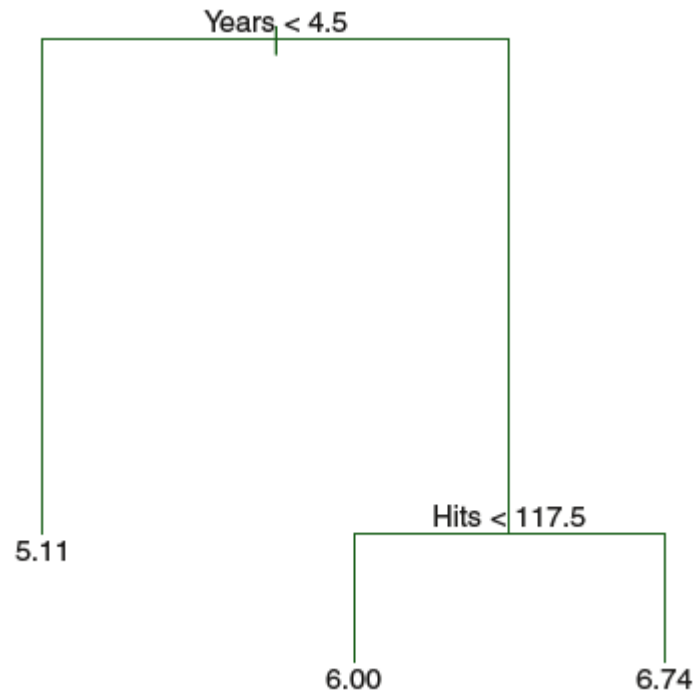
Finding the optimal α or T

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.*

The optimal (pruned) Tree
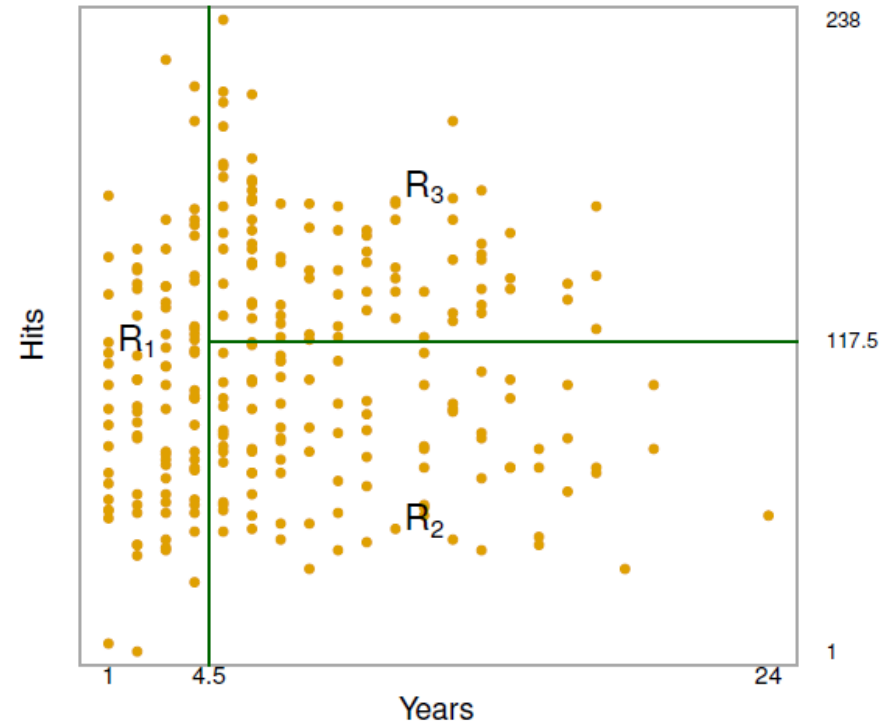
Advantages and Disadvantages of Trees

▲ Trees are very easy to explain to people. In fact, they are even easier to explain than linear regression!

▲ Some people believe that decision trees more closely mirror human decision-making than do the regression and classification approaches seen in previous chapters.

▲ Trees can be displayed graphically, and are easily interpreted even by a non-expert (especially if they are small).

▲ Trees can easily handle qualitative predictors without the need to create dummy variables.

▼ Unfortunately, trees generally do not have the same level of predictive accuracy as some of the other regression and classification approaches seen in this book.
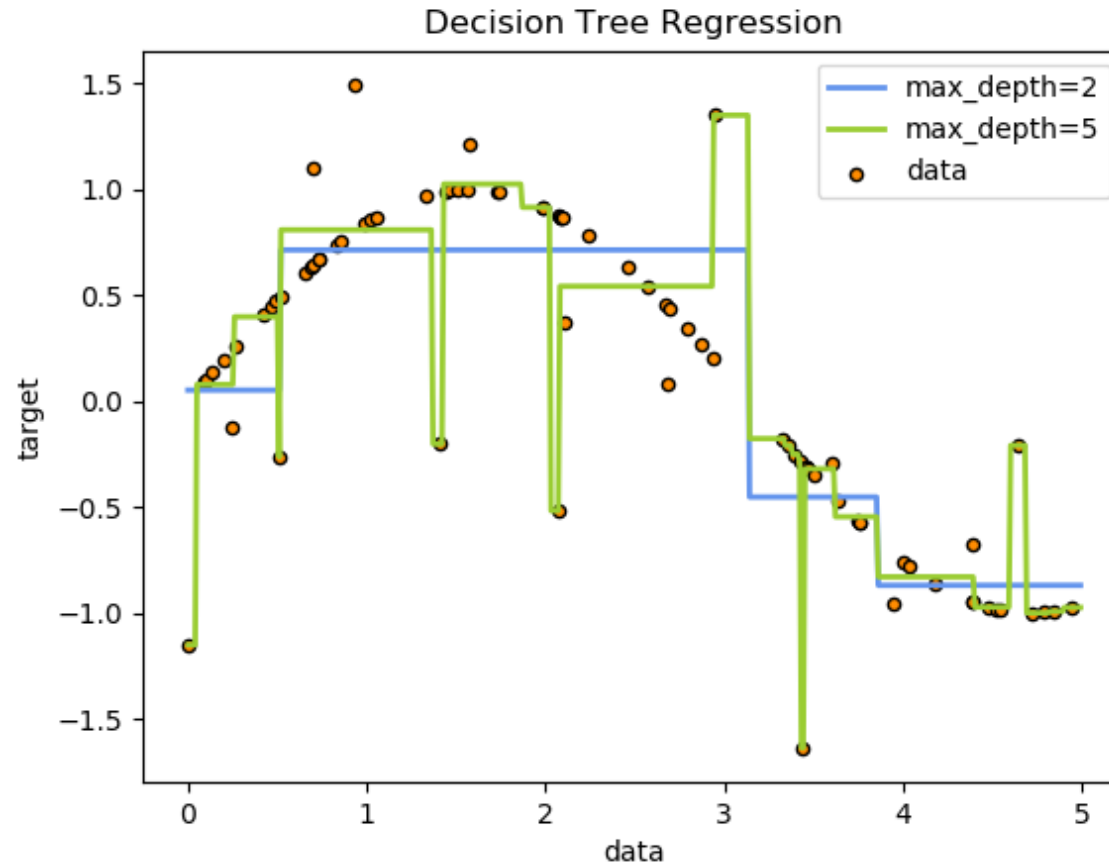
Trees Versus Linear Models

Left column: linear model; Right column: tree-based model

Top Row: True linear boundary
Bottom row: true non-linear
boundary.
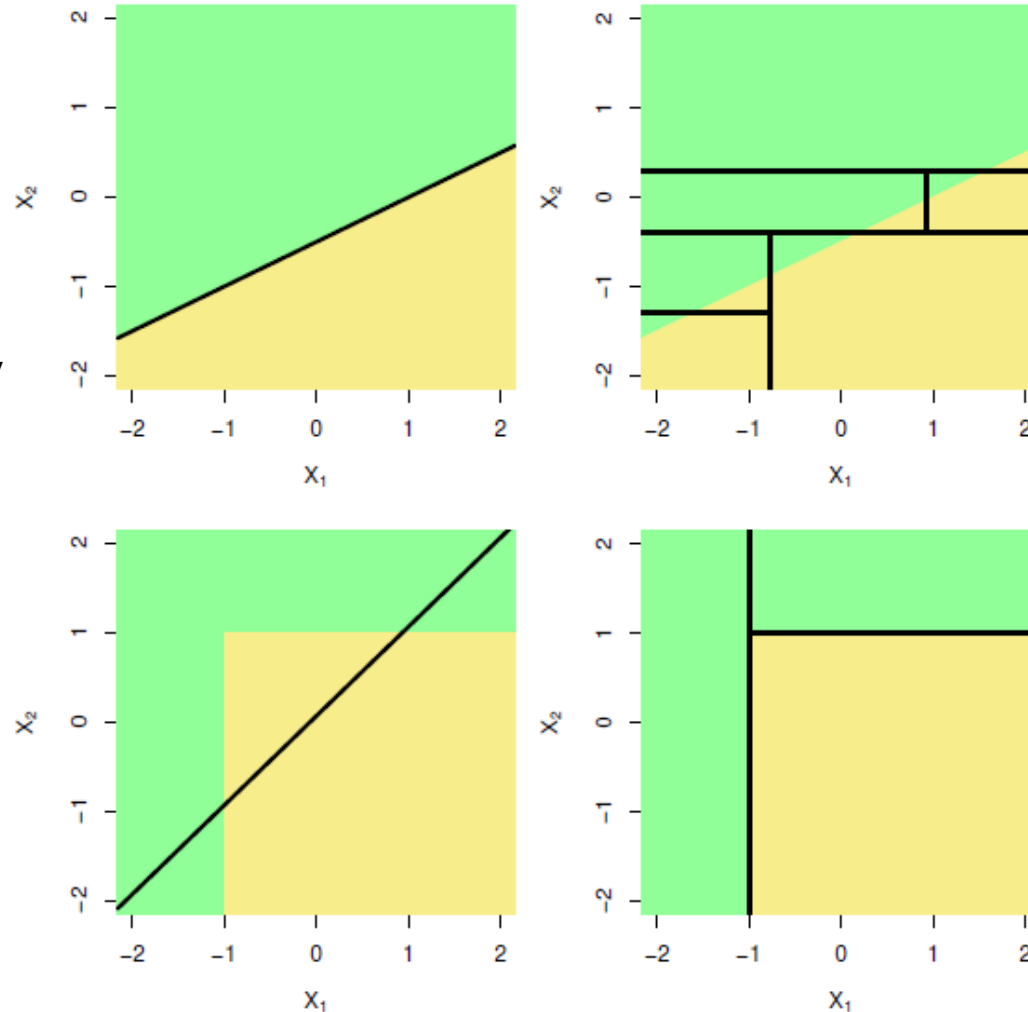
Decision Tree Regression Depth

Bootstrap Aggregation or Bagging

- *Bootstrap aggregation*, or *bagging*, is a general-purpose procedure for reducing the variance of a statistical learning method; we introduce it here because it is particularly useful and frequently used in the context of decision trees.

- Recall that given a set of n independent observations $Z_1, \ldots, Z_n$, each with variance σ^2, the variance of the mean $\bar{Z}$ of the observations is given by σ^2/n.

- In other words, *averaging a set of observations reduces variance*. Of course, this is not practical because we generally do not have access to multiple training sets.

Bagging Regressions

- Instead, we can bootstrap, by taking repeated samples from the (single) training data set.

- In this approach we generate B different bootstrapped training data sets. We then train our method on the bth bootstrapped training set in order to get $\hat{f}^{*b}(x)$, the prediction at a point x. We then average all the predictions to obtain

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

This is called *bagging*.

Out of Bag error estimation

- It turns out that there is a very straightforward way to estimate the test error of a bagged model.
- Recall that the key to bagging is that trees are repeatedly fit to bootstrapped subsets of the observations. 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* (OOB) observations.

- We can predict the response for the ith observation using each of the trees in which that observation was OOB. This will yield around $B/3$ predictions for the ith observation, which we average.

Random Forests

- *Random forests* provide an improvement over bagged trees by way of a small tweak that *decorrelates* the trees. This reduces the variance when we average 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 in a tree is considered, *a random selection of m predictors* is chosen as split candidates from the full set of p predictors. The split is allowed to use only one of those m predictors.

- A fresh selection of m predictors is taken at each split, and typically we choose $m \approx \sqrt{p}$ — that is, the number of predictors considered at each split is approximately equal to the square root of the total number of predictors

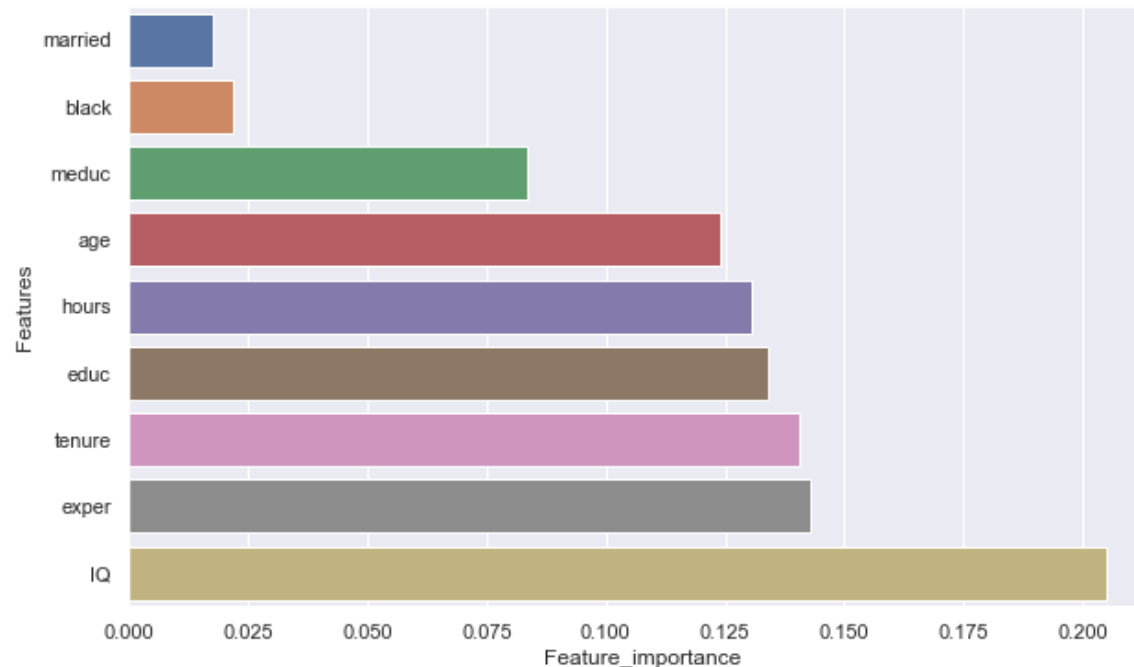# Why Random Forests vs Bagging?

- Suppose there is **one very strong feature** in the data set. When using "bagged" trees, most of the trees will use that feature as the top split, resulting in an ensemble of similar trees that are **highly correlated**.

Comparison of Random Forests and Bagging
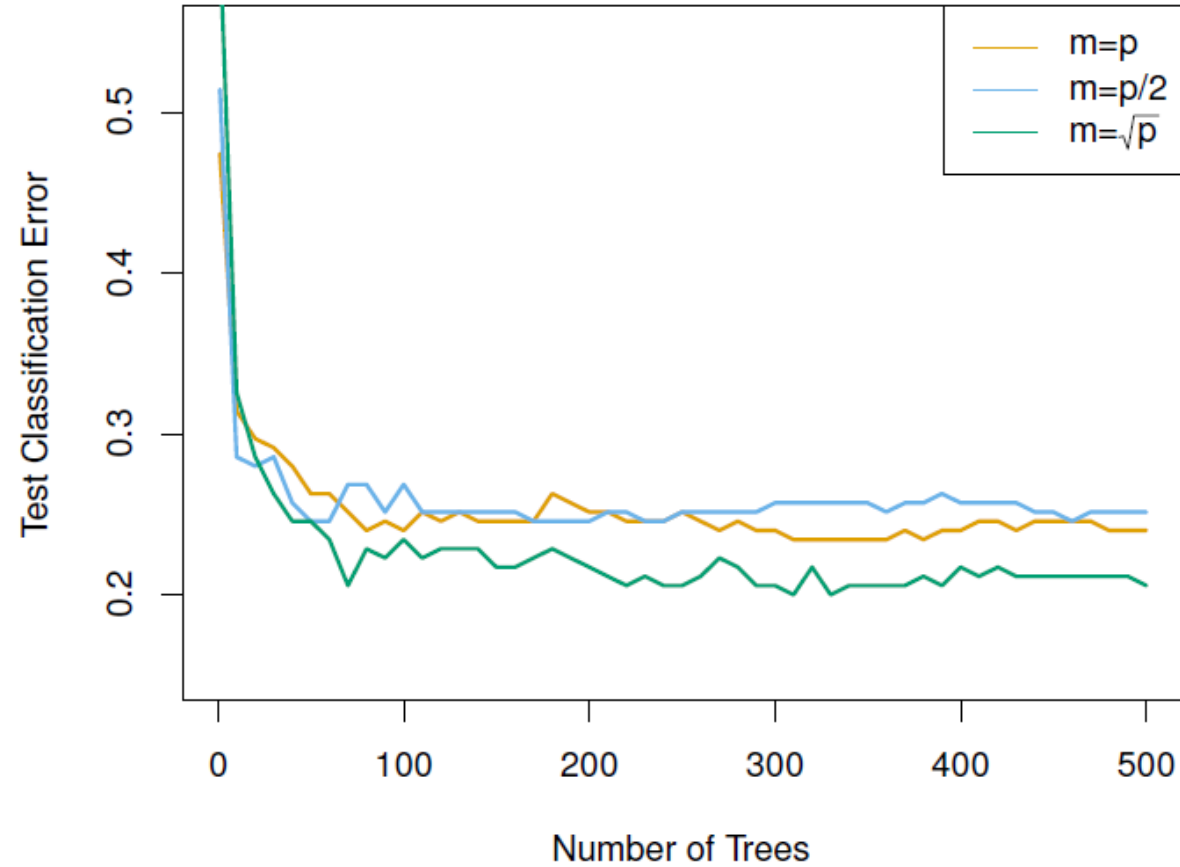
Variable importance measure

- For bagged/RF regression trees, we record the total amount that the RSS is decreased due to splits over a given predictor, averaged over all B trees. A large value indicates an important predictor.
- Similarly, for bagged/RF classification trees, we add up the total amount that the Gini index is decreased by splits over a given predictor, averaged over all B trees.

Take a way!

- Decision trees are simple and interpretable models for regression and classification

- However they are often not competitive with other methods in terms of prediction accuracy

- Bagging, random forests and boosting are good methods for improving the prediction accuracy of trees. They work by growing many trees on the training data and then combining the predictions of the resulting ensemble of trees.

- The latter two methods— random forests and boosting— are among the state-of-the-art methods for supervised learning. However their results can be difficult to interpret.

Random forest in python

- Find the Random forests Sklearn documentation here

- Blackbox version of Random forests in python:

```python
from sklearn.ensemble import RandomForestRegressor
```

```python
# Fitting Random Forests regression to the Training set

RF_regressor = RandomForestRegressor(n_estimators = 100, max_features='sqrt')
RF_regressor.fit(X_train, y_train)
```

```python
RandomForestRegressor(bootstrap=True, criterion='mse', max_depth=None,
                      max_features='sqrt', max_leaf_nodes=None,
                      min_impurity_decrease=0.0, min_impurity_split=None,
                      min_samples_leaf=1, min_samples_split=2,
                      min_weight_fraction_leaf=0.0, n_estimators=100,
                      n_jobs=None, oob_score=False, random_state=None,
                      verbose=0, warm_start=False)
```