# Applied Statistics and Data Analysis

# 8. Tree-based methods

Paolo Vidoni

Department of Economics and Statistics
University of Udine
via Tomadini 30/a - Udine

paolo.vidoni@uniud.it

Partly based on Chapter 8 of *An Introduction to Statistical Learning: with Applications in R* by G. James et al.

# Table of contents

# Summary

- Introduction to tree-based methods
- Regression trees
- Classification trees
- Ensemble methods: Bagging, Random forests, Boosting

# Introduction to tree-based methods

- **Tree-based methods**, also called **decision tree methods**, can be applied to both regression and classification problems.
- The basic idea is very simple: the $p$-dimensional predictor space is stratified or segmented into a number of simple sub-regions. This is performed with a suitable recursive partitioning procedure.
- The set of subsequent splitting rules, used to segment the predictor space, can be summarized in a tree-type graphical representation.
- At each **internal node** in the tree, a binary test is applied to one regressor variable. Depending on the result, the corresponding left or right **branch** of the tree is considered.
- The **terminal nodes** or **leaves** are at the bottom of the tree and they specify the sub-regions that are used to make a prediction for a given observation: the mean (in case of numerical response) or the mode (in case of factor response) of the training observations belonging to each region are typically considered.

# Example: the spam data set

## Data from *Data Analysis and Graphics Using R - An Example-Based Approach* by J. Maindonald and W.J. Braun

The spam data set, described before, contains data on $4601$ email items, of which $1813$ were identified as spam (UCI Machine Learning Repository `http://archive.ics.uci.edu/ml`).

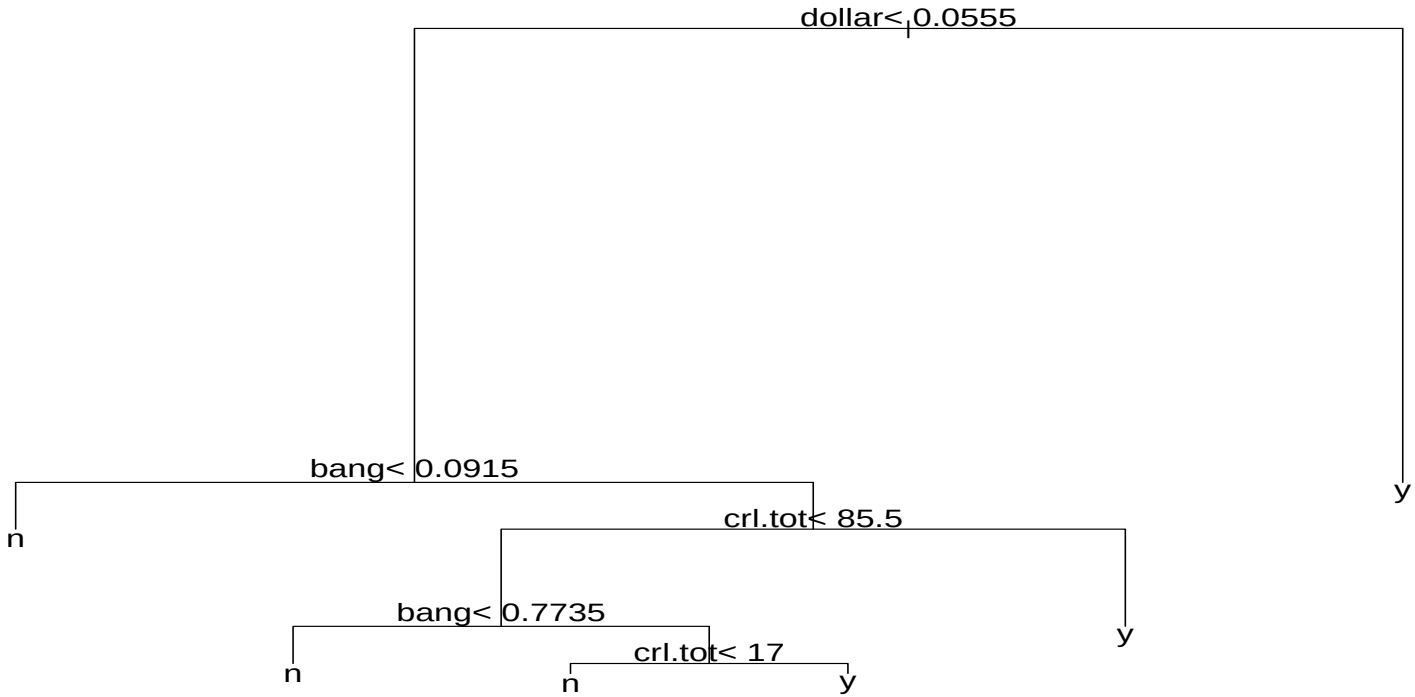
There are $57$ explanatory variables; a reduced version with $6$ explanatory variables is taken into account:

- `crl.tot` : total length of words in capitals
- `dollar` : number of occurrences of the \$ symbol
- `bang` : number of occurrences of the ! symbol
- `n000` : number of occurrences of the string '000'
- `make` : number of occurrences of the word 'make'
- `money` : number of occurrences of the word 'money'

The response variable is:

- `yesno`: a factor with two levels: `n` not spam, `y` spam

A simple classification tree is given below.




The top split assigns observations having $\texttt{dollar} \geq 0.0555$ to the right branch. The prediction for these kind of messages is `yesno=y`.

Items with $\texttt{dollar} < 0.0555$ are assigned to the left branch, and that group is further subdivided by `bang` and `crl.tot`; five additional terminal nodes are specified, with the associated predictions.

- Tree-based methods are useful with *extensive data* and uncertainty about the form in which explanatory variables should enter the model.
- In large data set, it is possible to detect relatively complex forms of structure, that are hard to find using conventional regression modeling.
- They may be useful for initial data exploration and the presentation of the results is effective and easy to explain.
- They are based on a methodology which is rather different from that of regression methods. It is fairly easy to use and it can be applied to a wide class of problems.
- Although they are not competitive with the best regression and classification techniques, they form the basis of more performing ensemble methods.

The most important problems for which tree-based methods may be used are:

- regression with a continuous response variable;
- classification with a binary response variable;
- classification with responses which can have one of several possible ordered outcomes;
- classification with responses which can have one of several possible unordered outcomes.

The first two problems are considered in the following.

The case of classification for binary responses is particularly important, for which tree-based methods are a further useful alternative to logistic regression.

There are several algorithm for specifying regression and classification trees. The focus here is on the so called **CART** approach introduced in the book *Classification and Regression Trees* by L. Breiman *et al.*, and well-known in the statistical framework.

# Table of contents

# The basic idea

- Given a training set $(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_n, y_n)$, the process of building a regression tree is based on the following steps:
  - divide the set of possible values for the regressors $X_1, \ldots, X_p$ (the predictor space) into $J$ non-overlapping regions $R_1, \ldots, R_J$;
  - the mean of the response values for the training observations that belong to $R_j$, $j = 1, \ldots, J$, is considered as point predictor for every (new) observation that falls into $R_j$.
- For ease of interpretation, the regions are defined as high-dimensional rectangles or boxes. Then, the objective is to find boxes $R_1, \ldots, R_J$ that minimize the **Residual Sum of Squares (RSS)**

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

  where $\widehat{y}_{R_j}$ is the mean response for training observations in $R_j$.
- Since this optimization problem is computationally infeasible, a simple *top-down*, *greedy approach* is considered.

# Recursive binary splitting

- Initially, all observations are supposed to belong to a single region.
- The tree-building process is based on **recursive binary splitting**. The first split is obtained by selecting the regressor $X_j$ and the cutpoint $s$ such that the corresponding regions

$$R_1(j,s) = \{\mathbf{x} : x_j < s\}, \quad R_2(j,s) = \{\mathbf{x} : x_j \geq s\}$$

produces the lowest RSS

$$\sum_{i:\mathbf{x}_i \in R_1(j,s)} (y_i - \widehat{y}_{R_1})^2 + \sum_{i:\mathbf{x}_i \in R_2(j,s)} (y_i - \widehat{y}_{R_2})^2.$$

The process is then repeated by splitting one of these two regions and it continues recursively until a stopping criterion is reached.

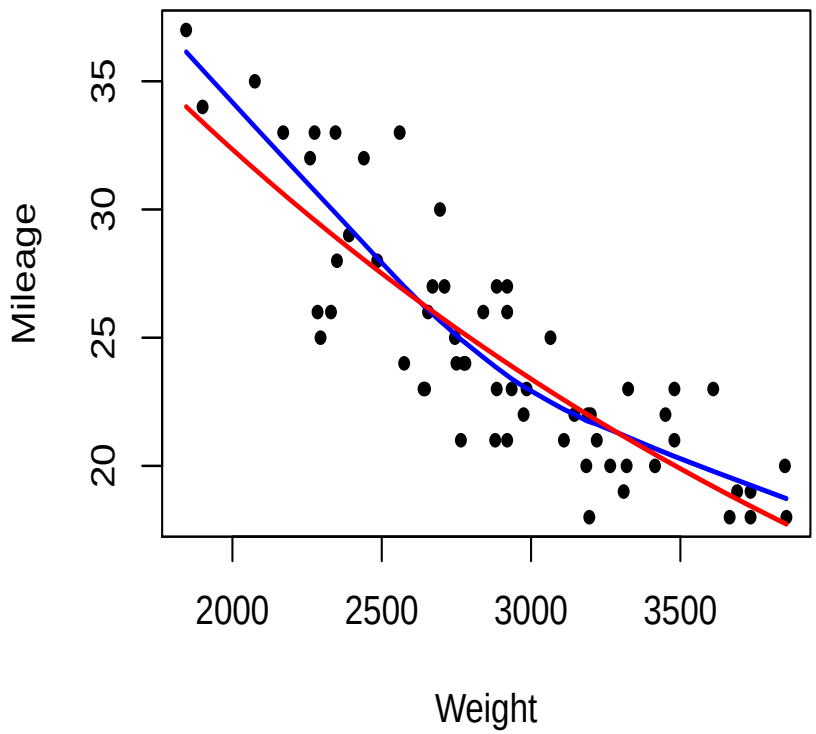
- The recursive binary splitting usually stops when no region contains more than a low number (generally five) of observations or when the splitting does not reduce the RSS by more than some threshold.
- For categorical regressors, possible splits are defined by factor levels.

# Example: automobile data

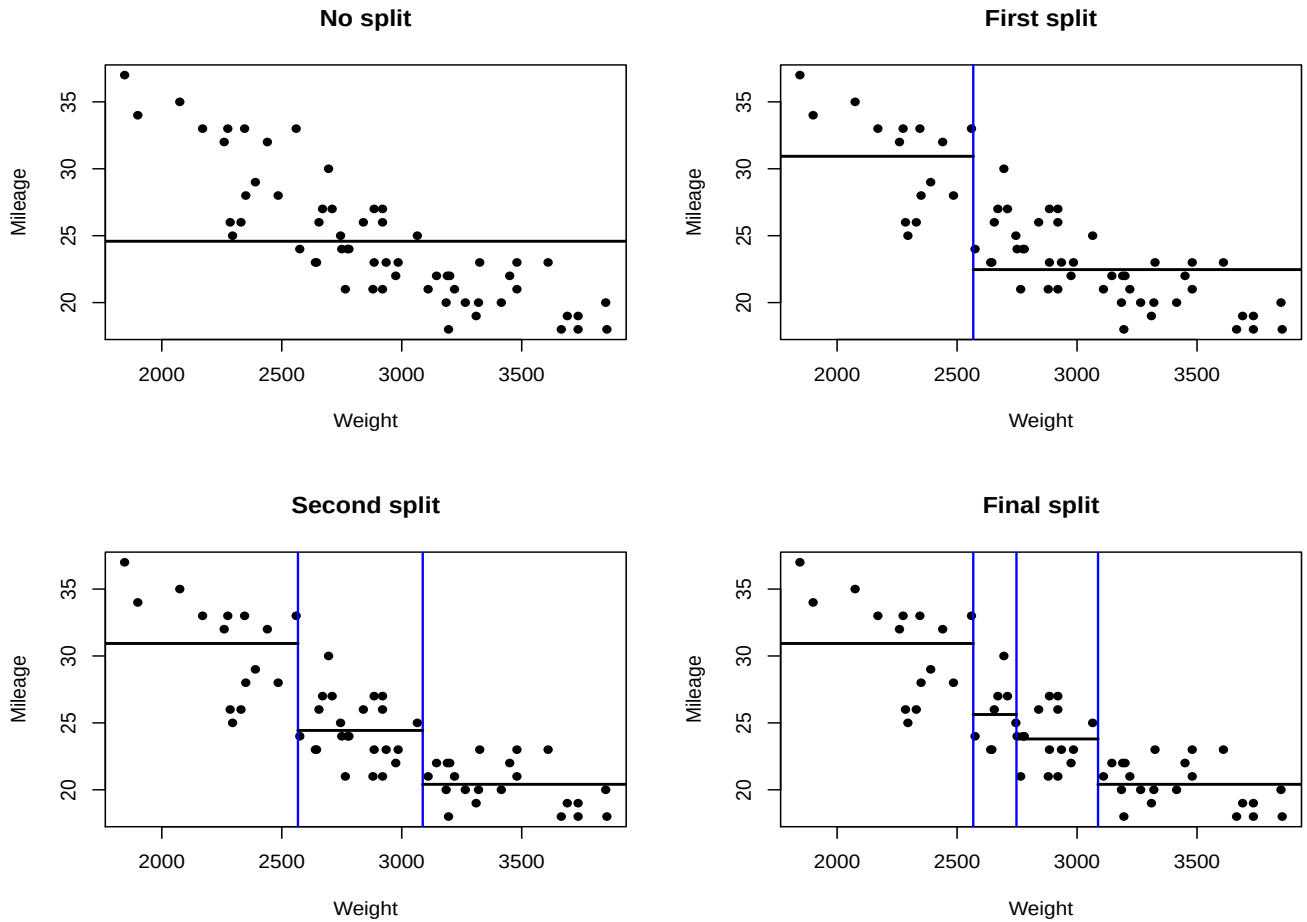
Data set with $n = 60$ observations on eight variables regarding makes of cars taken from the April, 1990 issue of *Consumer Reports*.

The response `Mileage` (fuel consumption in miles per US gallon) and the regressor `Weight` (weight of the vehicle in pounds) are considered.

The scatterplot with the fitted smooth curve and the fitted regression curve (based on a simple linear regression for the log response) is given below



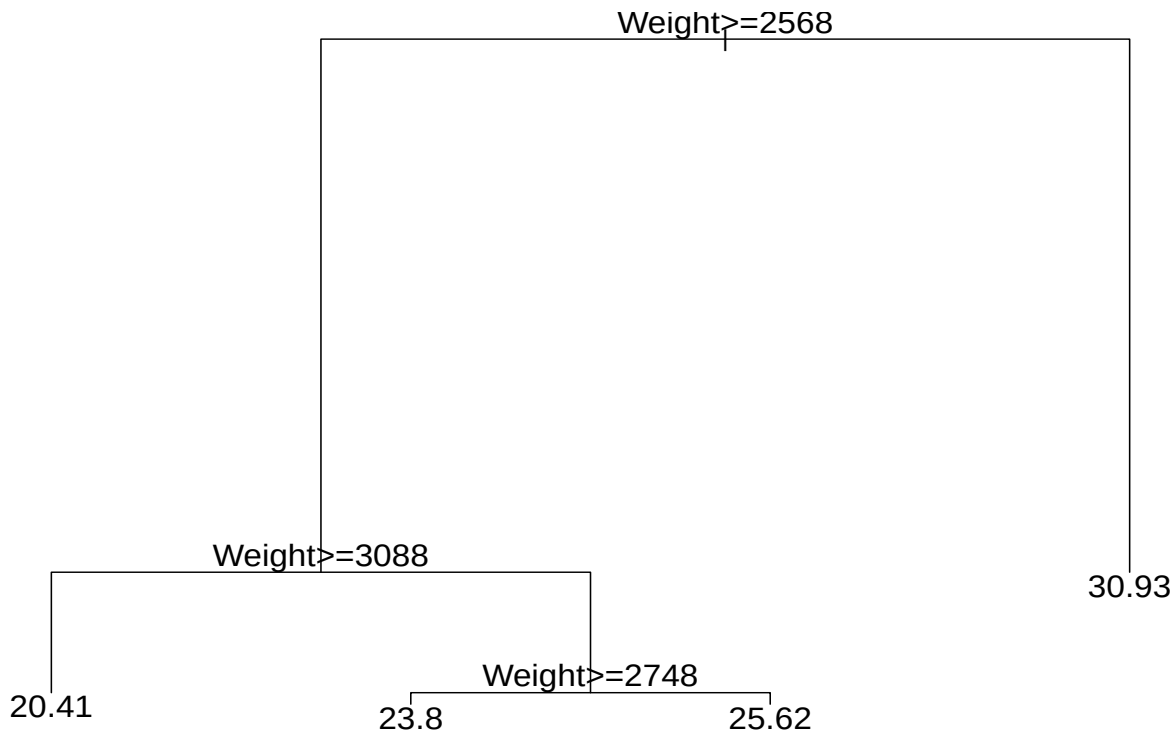
Partition of the one-dimensional regression space (blue line) by recursive binary splitting and point predictor for each box (**black line**).




In case of numerical predictors, regression trees specify an alternative, non-parametric approach to non-linear regression. The fitted regression curve is a step function.

The resulting fit of the tree-based regression model is not as good as for the other methods.

However, the associated graphical representation, using a regression (classification) tree, is appealingly simple.



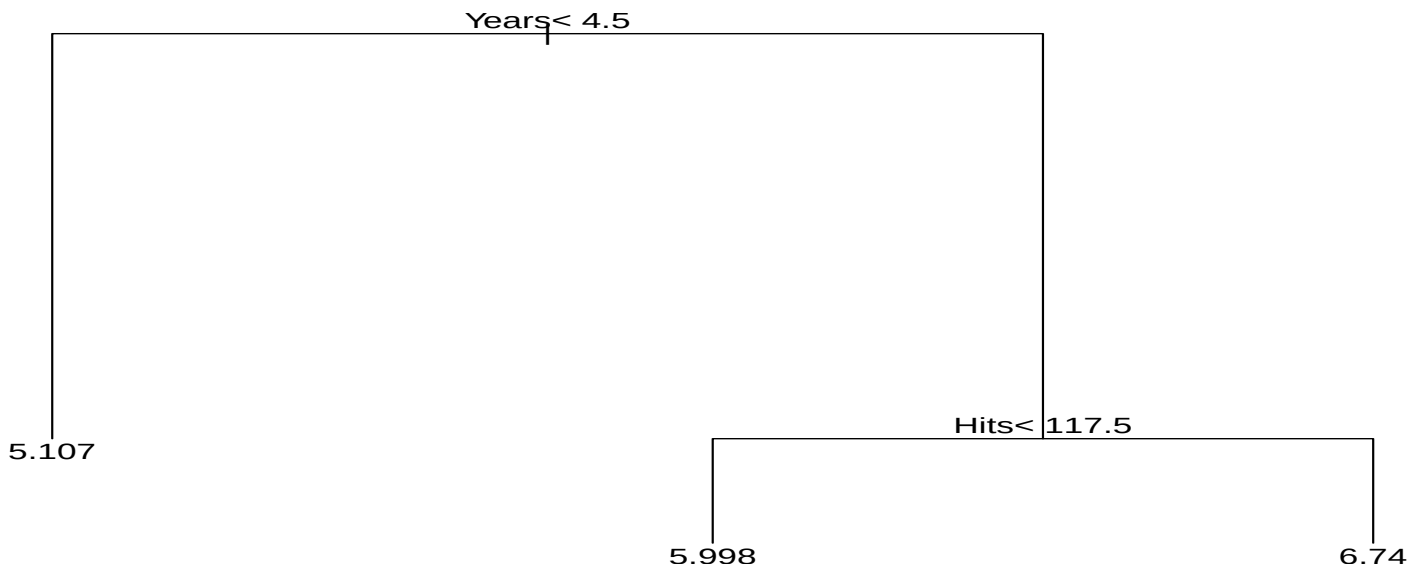
# Example: baseball data

**Data from *An Introduction to Statistical Learning: with Applications in R* by G. James et al.**

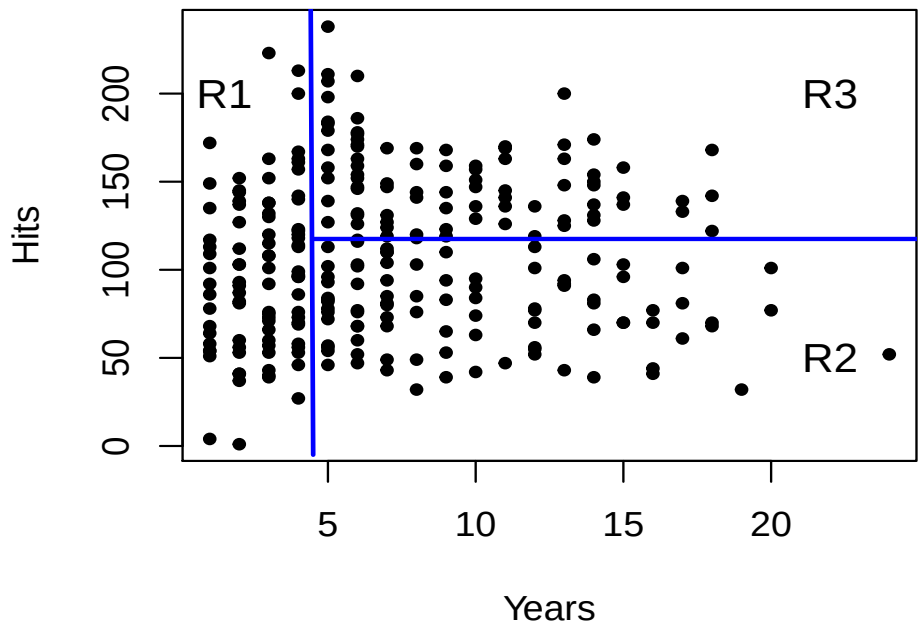
Data set with $n = 322$ observations on twenty variables regarding major league baseball players from the 1986 and 1987 seasons.

The response `Salary` (1987 annual salary in thousands of dollars) and the regressors `Years` (number of years in the major league) and `Hits` (number of hits in 1986) are considered.

A regression tree for predicting the log transformation of `Salary`, removing all the observations with missing values, is given below



This tree stratifies the players into three regions of the regressor space




The three regions, that correspond to the leaves of the tree, are $R_1 = \{\mathbf{x} : \texttt{Years} < 4.5\}$, $R_2 = \{\mathbf{x} : \texttt{Years} \geq 4.5, \texttt{Hits} < 117.5\}$ and $R_3 = \{\mathbf{x} : \texttt{Years} \geq 4.5, \texttt{Hits} \geq 117.5\}$.

The predicted salaries for these three groups of players are, respectively, $e^{5.107} = 165.174$, $e^{5.998} = 402.623$ and $e^{6.74} = 845.561$ thousands of dollars.

# Pruning the tree

- The process outlined above may produce a tree that might be too complex: poor test set predictive performance (overfitting).
- A smaller tree with fewer splits, and then fewer terminal nodes (leaves) $R_1, \ldots, R_J$, might present a lower variance (and a better interpretation) with a little increase in bias.
- An effective strategy is to define a very large tree and then to **prune** it back in order to obtain a suitable **subtree**. However, estimating the test error (for example using cross-validation) for every possible subtree would be too computationally demanding.
- An alternative is to consider **cost complexity pruning**, which is based on the following measure for the goodness of a tree

$$\sum_{j=1}^{J} \sum_{i \in R_j} (y_i - \widehat{y}_{R_j})^2 + c_p \cdot J = \text{RSS} + c_p \cdot J,$$

where $J$ is the number of terminal nodes and $c_p$ is a non-negative tuning parameter.

- The tuning parameter $c_p$ controls the trade-off between the goodness of fit (with respect to the training data) and the tree's complexity.
- When $c_p = 0$, the selected tree corresponds to that one (usually a large one) obtained by minimizing the training RSS.
- As $c_p$ increases, trees with many terminal nodes are progressively penalized; branches get pruned from the original large tree, obtaining a simpler subtree with less terminal nodes.
- The best value of $c_p$ can be chosen by a validation set or by a cross-validation procedure.
- The optimal tree, corresponding to the selected $c_p$, has the minimum cross-validated error rate, but sometimes it is still too complex.

  The **one standard deviation rule** is often employed: choose the smallest tree whose cross-validated error is less than minimum cross-validated error plus the associated standard deviation.

  This rule is conservative, since the predictive power will on average be slightly less than optimal.

The tree pruning process can be summarized in the following algorithm.

1. Use recursive binary splitting to specify a (large) tree based on the training data, stopping when no terminal node contains more than a low number of observations.
2. Apply cost complexity pruning to this large tree in order to obtain a sequence of best subtrees, as a function of suitable values for $c_p$.
3. Use a $K$-fold cross-validation procedure in order to chose the best value for $c_p$ (namely, the value minimizing the CV error rate or satisfying the *one standard deviation rule*).
4. Return the subtree from Step 2 that corresponds to this value of $c_p$.

Note that although the CV error rate is computed as a function of $c_p$, it can be represented as a function of $J$, the number of leaves (terminal nodes). As the fixed value of $c_p$ reduces, the size $J$ of the tree tends to increase.

# Example: baseball data

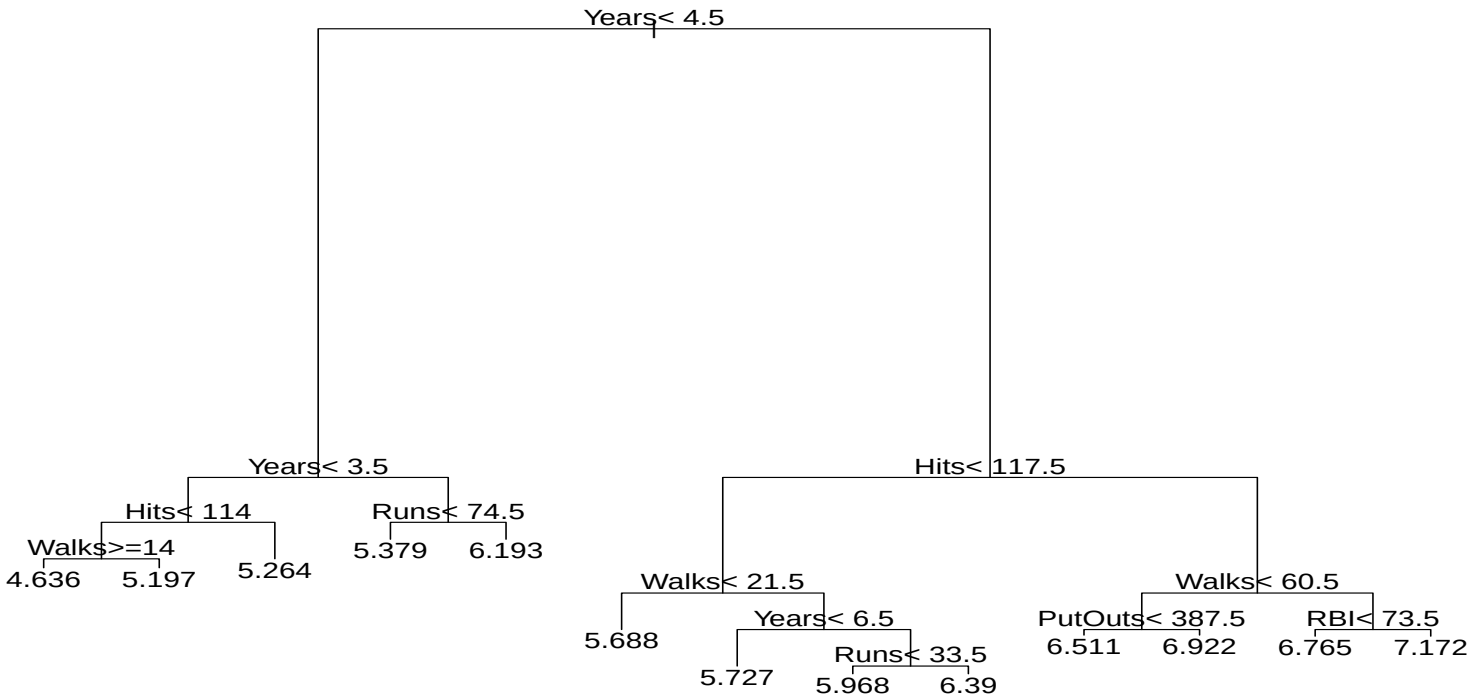
The baseball data set contains $n = 322$ observations on twenty variables regarding major league baseball players from the 1986 and 1987 seasons.

The logarithmic transformation of the response `Salary` (1987 annual salary in thousands of dollars) and the following regressors are taken into account

- `Years` : number of years in the major league
- `Hits` : number of hits in 1986
- `RBI` : number of runs batted in 1986
- `PutOuts` : number of put outs in 1986
- `Walks` : number of walks in 1986
- `Runs` : number of runs batted in in 1986

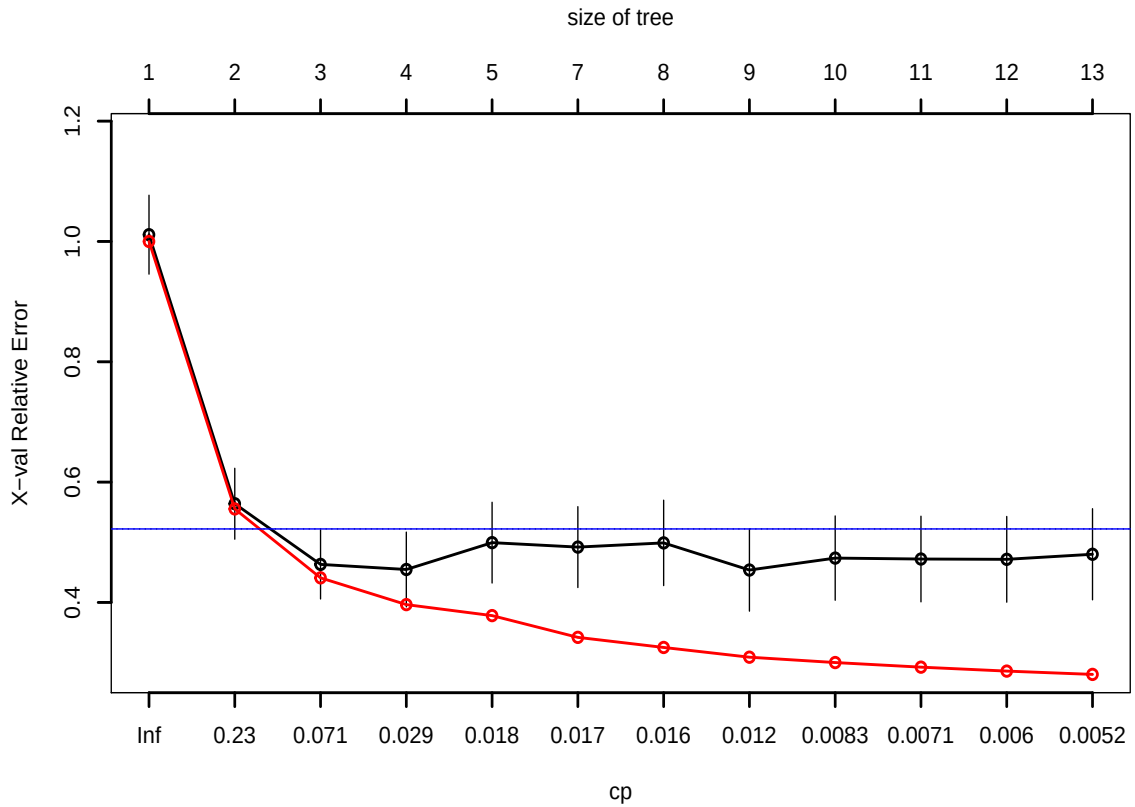
All the observations with missing values are removed and an unpruned, relatively large, regression tree is obtained by setting $c_p = 0.005$.

The result of fitting the regression tree on the training data is shown in the following tree representation.




This regression tree can be too complex for predicting the log-transform of a baseball player's `Salary`. A cost complexity pruning procedure is applied in order to reduce the number of leaves.

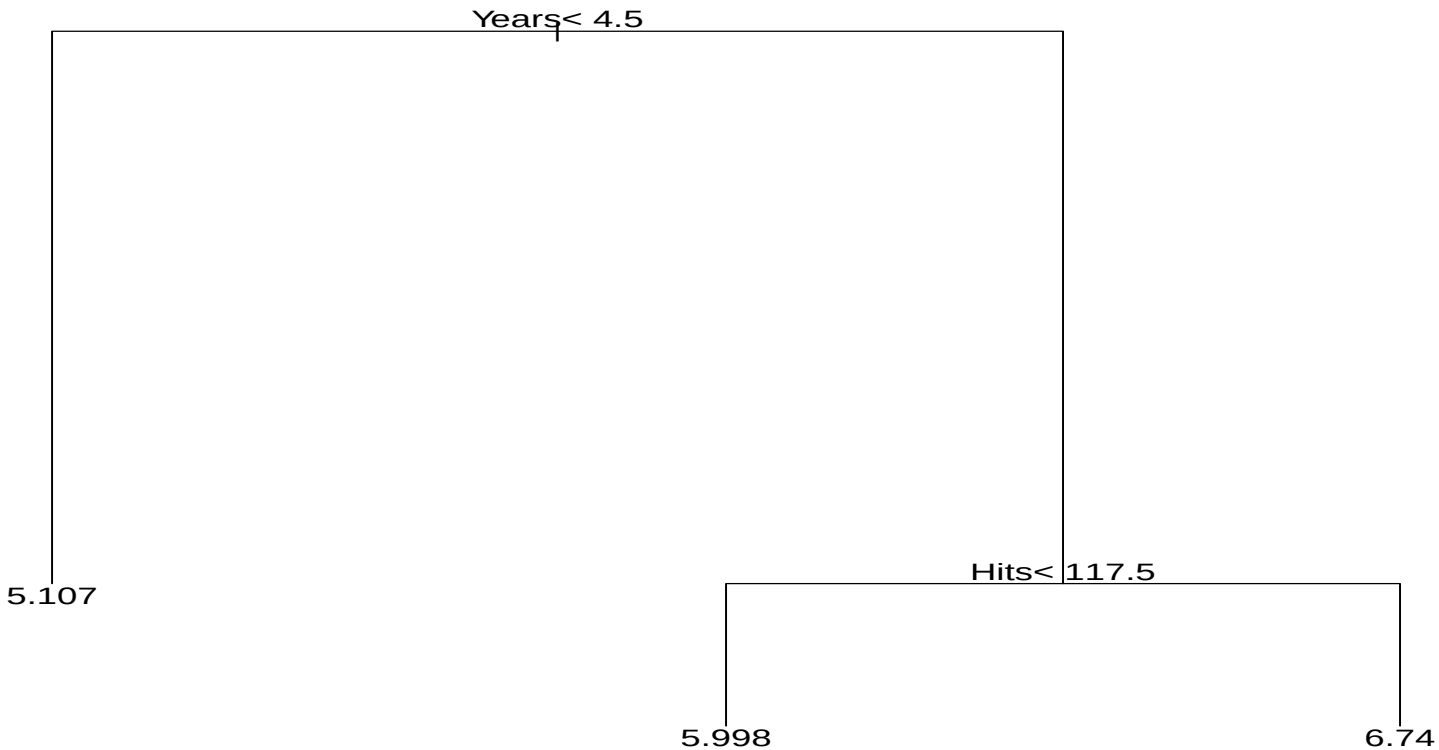
The CV estimate of the test error, with the associated standard error bars, are represented as function of $c_p$ and of the size of the tree (**black**). The training error curve is shown in red. The horizontal line is drawn at one standard deviation above the minimum of CV error.




The minimum CV error occurs at a tree of size $4$, whereas the *one standard deviation rule* selects a tree of size $3$. Note that the training error points to the large original tree with $c_p = 0.005$ and size $13$.

The pruned tree, selected using the *one standard deviation rule*, has three terminal nodes.

It corresponds to that one obtained before with the regressor `Years` and `Hits` and a value $c_p = 0.1$.



# Table of contents

# Introduction to classifications trees

- A classification tree is similar to a regression tree. The main difference is that it is used to predict a qualitative response rather than a quantitative one.
- As for a regression tree, terminal nodes are used to obtain a prediction for a given observation: it is the most commonly occurring class (category) of training observations in the region to which it belongs.
- The interest is not only in the class prediction related to a particular terminal node but also in the class proportions among the training observations that fall into that region.
- Growing a classification tree is quite similar to the task of growing a regression tree.
- Recursive binary splitting is considered using a splitting criterion alternative to the RSS. A natural choice is the classification error rate or an alternative measure of node purity.

# Measures of node purity

- The general idea is that the best split should give nodes with a single kind of units predominating in each region.
- For the case of a qualitative response variable with $K \geq 2$ categories, indexed by $k$, let $\widehat{p}_{jk}$ be the observed proportion of training observations in the $j$-th region that are assigned to the $k$-th class.
- The **classification error rate** is the fraction of the training observations in the $j$-th region that do not belong to the most common class:

$$E = 1 - \max_k \{\widehat{p}_{jk}\}$$

- The **Gini index** is a measure of the total variance across the classes

$$G = \sum_{k=1}^{K} \widehat{p}_{jk}(1 - \widehat{p}_{jk})$$

- An alternative, similar index is the **cross-entropy** (**deviance**)

$$D = -\sum_{k=1}^{K} \widehat{p}_{jk} \log \widehat{p}_{jk}$$

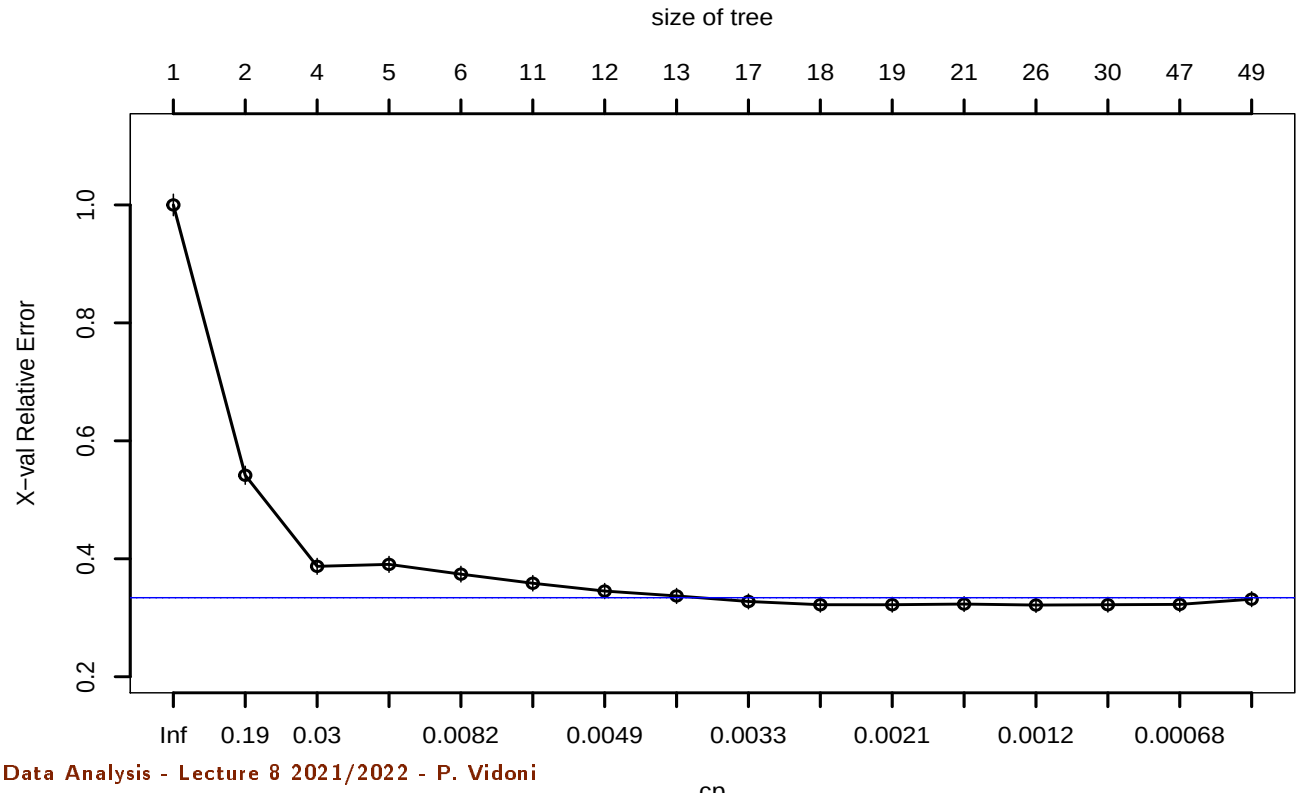
- These measures take small values when the observed proportions $\widehat{p}_{jk}$, $k = 1, \ldots, K$, are closed to zero or one, namely if the $j$-th node is pure.
- In a two-class problem, with $\widehat{p}_j$ the observed proportion of one class, the three indices correspond to $E = 1 - \max\{\widehat{p}_j, 1 - \widehat{p}_j\}$, $G = 2\widehat{p}_j(1 - \widehat{p}_j)$ and $D = -\widehat{p}_j \log(\widehat{p}_j) - (1 - \widehat{p}_j) \log(1 - \widehat{p}_j)$.
- The Gini index and the cross-entropy are quite similar numerically and, since they are more sensitive to node purity than the classification error rate, they are typically used for tree-growing.
- Any of these measures may be used when pruning the tree, for example using the cost complexity pruning approach.

  In this context, if the goal is the prediction accuracy of the final pruned tree, the classification error rate is preferable.
- As in the regression framework, also decision trees may involve factor regressors: in this case a split amounts to assigning some of the qualitative values to one branch and the remaining to the other one.

# Example: the spam data set

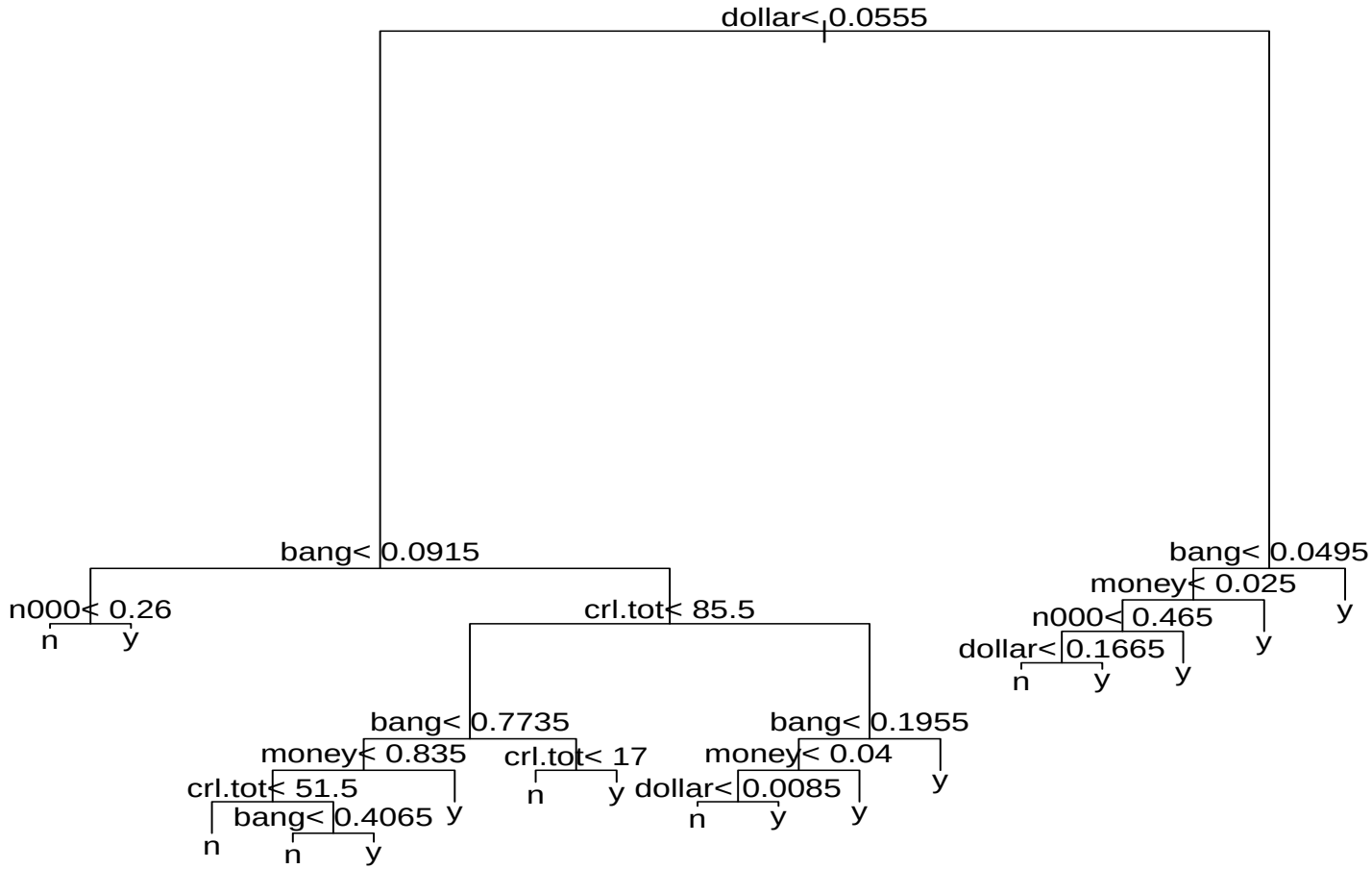
With regard to the spam data set considered before, a large tree is fitted to the training data by setting $c_p = 0.0005$. The resulting classifier is too complex, having $49$ leaves. A cost complexity pruning procedure is applied.

The following plot describes the CV classification error, corresponding to different values of $c_p$. The horizontal line is drawn at one standard deviation above the minimum of CV error.



The minimum CV error occurs at a tree of size $18$, whereas the *one standard deviation rule* selects a tree of size $17$. With this latter choice, $c_p = 0.0028$ and the classification error is $12.7\%$.

The final classifier is more complex than that one presented before, yet still interpretable.



# Advantages of tree-based methods

- Results are in an immediately useful form, since trees are easy to explain.
- Results do not change with monotone transformation of continuous predictors.
- Categorical predictors are easily handled, without using dummy variables.
- The methodology can be extended to handle missing values.
- Tree-based methods capture interactions and non-additive behavior.
- Both classification and regression are handled.

# Limitations of tree-based methods

- Large trees are hard to interpret, and they are used as *black boxes*.
- Continuous predictors are not treated as such, with some loss of information.
- Limited notions of what to look for may result in failure to find useful structures: they may obscure insights that are obvious from parametric models.
- Trees have usually high variance and small changes in data may result in different tree structures. An error in a top split is propagated down to all of the splits below.
- Trees often do not have the same predictive accuracy as some other regression and classification approaches.

# Table of contents

# Introduction to ensemble methods

- Ensemble methods apply multiple classification or regression procedures. They are based on iterated applications of simple classifiers or regression models.

  Trees or regression models are used as building blocks to construct more powerful prediction models.
- Ensemble methods are computationally intensive, but at the same time they provide *improved predictive performances*.
- There are several ensemble methods, the main ones are **Bagging**, **Boosting** and **Random forests**.

  The first two are very general and they can be built upon algorithms different from decision trees, whereas the latter one is mainly applied to tree-based methods.

# Bagging

- Decision and regression trees often have *high variance*.

  When a statistical method suffers from high variance, this means, for example, that if the training data are randomly split into two parts and the method is fitted to each part, the results could be quite different.
- **Bagging** (**bootstrap aggregation**) is a general method to reduce the variance of a statistical learning method. It is frequently use in the context of regression and decision trees.
- The basic idea is very simple: whenever $B$ *independent replications* of the training set are obtained and the method is applied to each replication, then the resulting variance of the average of the $B$ predictions will be reduced by a factor $B$.
- This is simply verified using the sample mean (averaging a set of $B$ observations reduces the variance by a factor $B$), but this is a quite general result.

- Since obtaining independent replications of the training set is usually impossible, bagging uses **nonparametric bootstrap** instead.

  A nonparametric bootstrap sample is obtained by sampling $n$ observations with replacement from the (single) $n$-dimensional training set.
- Given $B$ bootstrapped training sets, *regression trees* are fitted and the associated predictions $\widehat{f}_b^*(\mathbf{x})$, $b = 1, \ldots, B$, are obtained. Finally, an aggregated estimate is produced by averaging all the predictions

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

- The individual trees are grown deep, with no pruning. Each individual tree has a low bias and a large variance, but the final aggregation performs a significant variance reduction.
- Although it is particularly useful for regression and decision trees, bagging can improve predictions for many regression methods.

- For *classification trees*, aggregation is performed by taking each observation of the test set, obtaining the class predicted by each individual tree and choosing the final classification by a *majority vote*.
- The number $B$ of trees is not very crucial, even a large value will not lead to overfit. The value $B = 500$ is often used, but the best choice may depend on the problem.
- In this framework, there is a very straightforward way to estimate the test error of the bagged model, without using a cross-validation or a validation set approach.
- It is based on a simple property of nonparametric bootstrap: each bootstrapped sample will include, on average, about two-thirds of the original observations, with the remaining one-third left out, and referred to as the **out-of-bag (OOB) observations**.
- It is possible to use the OOB observations as test data for the corresponding tree.

- More precisely, the response for the $i$-th observation is predicted using each tree in which the observation is OOB. An aggregated prediction for this observation is obtained by averaging the predicted responses (regression tree) or by taking a majority vote (classification tree).
- A single OOB prediction is obtained for each observation and then a final OOB prediction error can be obtained. The OOB error is equivalent to leave-one-out CV, but it is much quicker to obtain.
- The results of bagging, differently from decision trees, are often hard to interpret.
- However, it is possible to measure of the importance of each regressor by computing, for each bootstrapped tree, how much the RSS (or the Gini index) decreases by splitting over that predictor.

  The final measure is obtained, again, by averaging over the $B$ bootstrapped trees, with a large value denoting an important predictor.

# Random forests

- **Random forests** extend and improve bagging by means of a slight modification in the procedure that *decorrelates* the trees.
- It is well-known that averaging highly correlated quantities does not reduce the variance as averaging uncorrelated quantities.

  In case of bagging, if there are very strong regressors in the data set, most or all of the bagged trees will consider these regressors in the top splits, and the corresponding predictions will be highly correlated.
- As in bagging, decision or regression trees are built on the $B$ bootstrapped training samples.

  However, when building these trees, each time a split in a tree is considered, a random sample of only $m$ regressors (out of the original $p$) is considered.
- At each split a new random sample of $m$ predictors is taken and only one of those $m$ predictors can be used. A typical choice for the sample dimension is $m \approx \sqrt{p}$.

- This leads to individual trees that are often quite different from one another and thus they are, in some sense, *decorrelated*.

  This alleviates the main limitation of bagging, which is based on trees that can be highly correlated.
- The main difference between bagging and random forests is the choice of predictor subset size $m$. As a special case, if a random forest is built assuming $m = p$, a bagging procedure is obtained.
- Using a small value of $m$ will be usually helpful when there is a large number of correlated predictors.
- In many real data applications, especially for complex problems, random forests are often the best performing method for classification.

# Boosting

- Like bagging, **boosting** is a general approach that can be applied to many statistical learning methods, for regression and classification. Here, the focus is on regression and decision trees.
- Instead of fitting a single large tree model to the data, boosting involves combining, like bagging, a large number of decision trees.
- However, boosting works by creating a *sequence of trees* and each tree is fitted on a modified version of the original data set, so that no bootstrap resampling is involved.
- Each tree is grown using information from the previous trees in the sequence.
- The key thing is that boosting *learns slowly*: the next tree just modifies slightly the previous one in order to improve it where it does not perform well.

A boosting algorithm for regression trees is presented below. The algorithm for classification trees proceeds similarly, but it is slightly more complex.

1. Set $\widehat{f}(\mathbf{x}) = 0$ and $r_i = y_i$ for each $i$ in the training set.
2. For $b = 1, \ldots, B$, repeat:
   - **(a)** fit a tree $\widehat{f}_b(\mathbf{x})$ with $d$ splits to the training data $(\mathbf{x}_i, r_i)$, $i = 1, \ldots, n$;
   - **(b)** update $\widehat{f}(\mathrm{x})$ by adding a shrunken version of the new tree:

$$\widehat{f}(\mathbf{x}) \to \widehat{f}(\mathbf{x}) + \lambda \widehat{f}_b(\mathbf{x}).$$

   - **(c)** update the residuals:

$$r_i \to r_i - \lambda \widehat{f}_b(\mathbf{x}_i), \ i = 1, \ldots, n.$$

3. Output the boosted model

$$\widehat{f}(\mathbf{x}) = \sum_{b=1}^{B} \lambda \widehat{f}_b(\mathbf{x}).$$

- The algorithm has three tuning parameters:
  - the number of trees $B$: it requires more care than for bagging or random forests, since a large $B$ could lead to overfitting; it is best chosen via cross-validation;
  - the shrinkage parameter $\lambda > 0$: it should be positive and small, ensuring slow learning; typical values are $\lambda = 0.01$ or $\lambda = 0.001$;
  - the number $d \geq 1$ of splits in each tree; usually simple trees are used, with a small $d$, and even $d = 1$, that gives *decision stumps* (trees with a single split), works well in many cases.
- A tree is fitted sequentially using the current residuals, instead of $y_i$, $i = 1, \ldots, n$, as values for the response.
- Each of these trees is rather simple, having a small number $d$ of splits; the parameter $d$ is called the *interaction depth* of the boosted model ($d$ splits involve at most $d$ regressors).
- A small $\lambda$ can require a large value of $B$ in order to achieve a good performance.
- The boosting algorithm slowly improves the boosted model $\widehat{f}(\mathbf{x})$ (and the shrinkage parameter $\lambda$ further slows down the process) in areas where it does not perform well.

# Example: baseball data

The baseball data set is once again taken into account. The observations with missing values are removed, so that $n = 263$. The response is the logarithmic transformation of `Salary` and there are $p = 6$ predictors.

In addition to the regression tree with minimum CV error (size $18$) previously considered, bagging, random forests (with $m = 3$) and boosting (with $d = 3$) are applied.

In the following table, the $K$-fold cross-validation estimates (with $K = 10$) of the testMSE are presented

| Tree (size 3) | Bagging | Random forest ($m = 3$) | Boosting ($d = 3$) |
|---|---|---|---|
| $0.418$ | $0.257$ | $0.241$ | $0.281$ |

All the ensemble methods effectively improve over the simple regression tree and their performance is quite similar, since the standard error of the CV estimates is greater than $0.028$.

Moreover, the evaluation of the importance of the regressors states that `Years` and `Hits` are by far the two most important variables.

# Example: the spam data set

With regard to the spam data set already considered, a simple classification trees with size $20$ is specified using the C4.5 algorithm. This algorithm returns a pruned classification tree based on the cross-entropy node purity measure.

In this example, ensemble methods do not improve much over simple classification trees, as shown in the following table that gives the total accuracy rate, the true negative rate and the true positive rate estimated using $K$-fold cross-validation, with $K = 10$, or OOB data.

| | tot. accuracy | true negative | true positive |
|---|---|---|---|
| Trees | 0.87 | 0.94 | 0.77 |
| Boosting | 0.87 | 0.93 | 0.79 |
| Bagging | 0.88 | 0.92 | 0.81 |
| Random Forests | 0.88 | 0.95 | 0.78 |

More striking improvements are obtained with the full data set, with $57$ predictors.