Session 8
Tree-Based Methods

PSYC 560
Heungsun Hwang

Tree-Based Methods

- These methods involve stratifying or segmenting the predictor space (the set of possible values for predictors) into a number of simple regions.
  - They can be applied to both regression and classification problems.

- 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.

Tree-Based Methods

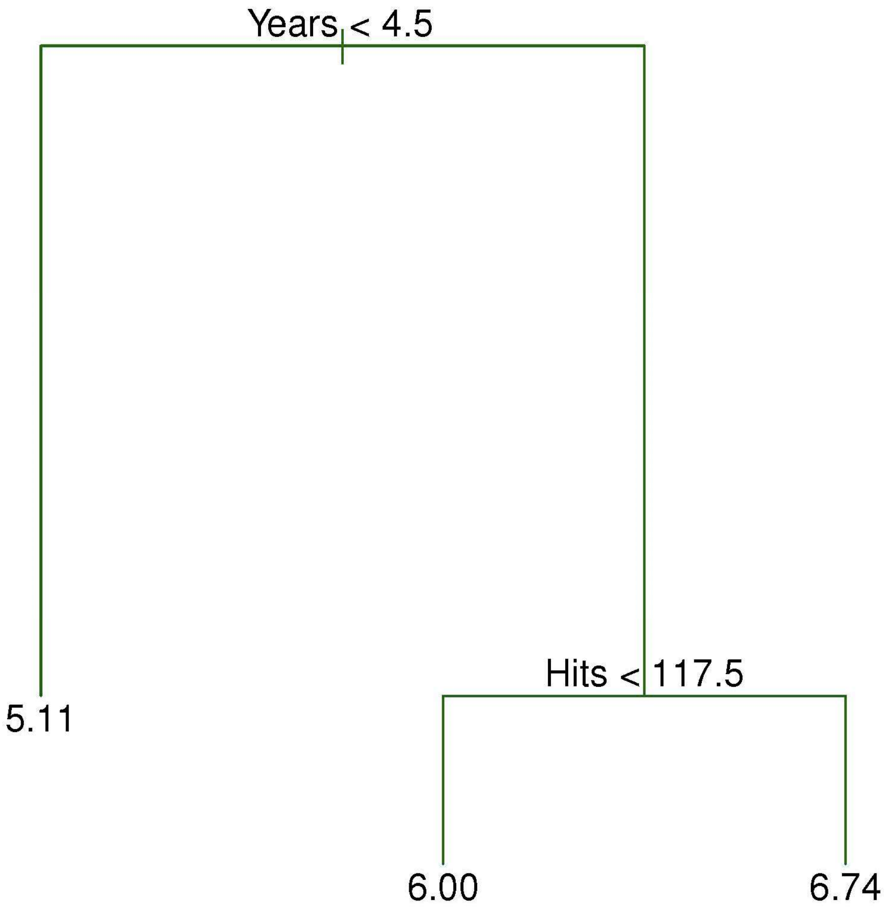
- Tree-based methods are simple and useful for interpretation.

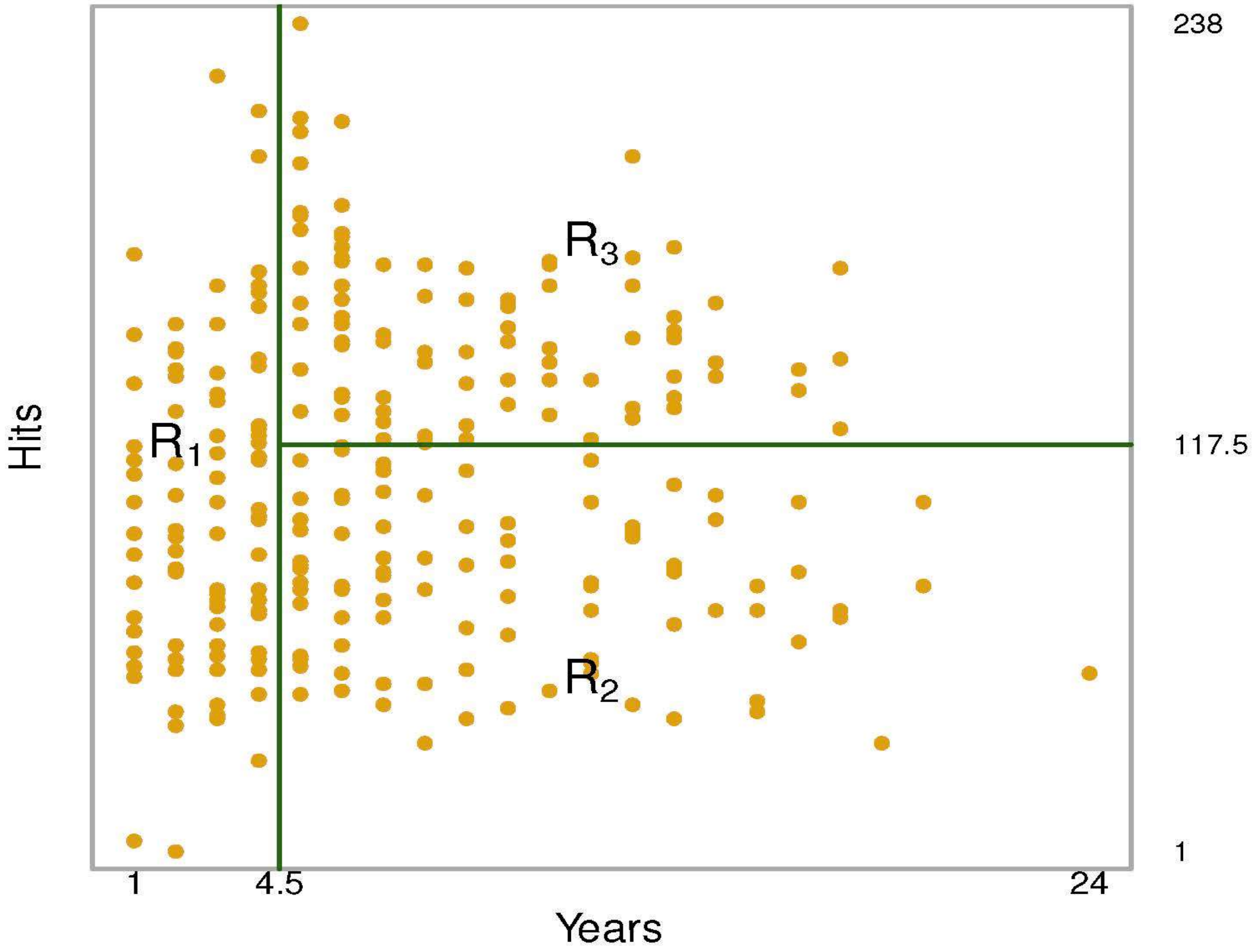
- However, they typically are not competitive with the best supervised learning approaches, such as regularization methods, in terms of prediction accuracy.

- We also discuss recent tree-based methods that aim to improve prediction accuracy.
  - Bagging, random forests, and boosting

Regression Trees

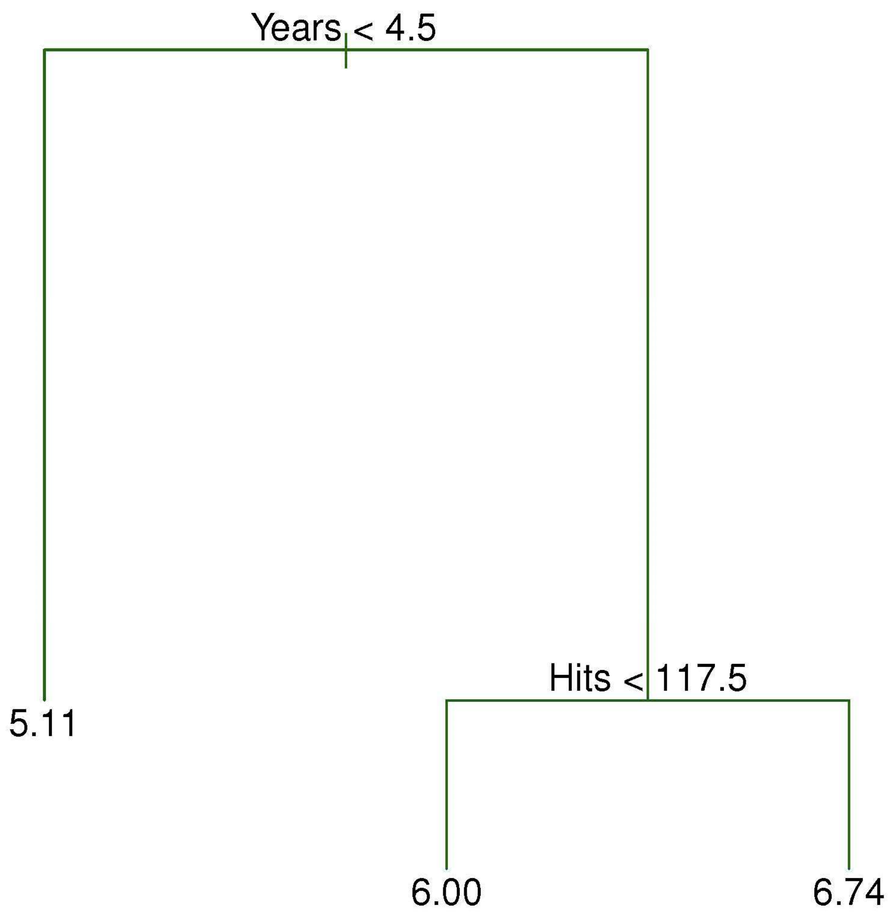
- The Hitters data (James et al.,2021, Figure 8.1)
  - Y = Salary (of major league baseball players)
  - X1 = Years (the number of years that he has played in the major leagues)
  - X2 = Hits (the number of hits that he made in the previous year)
  - N = 263





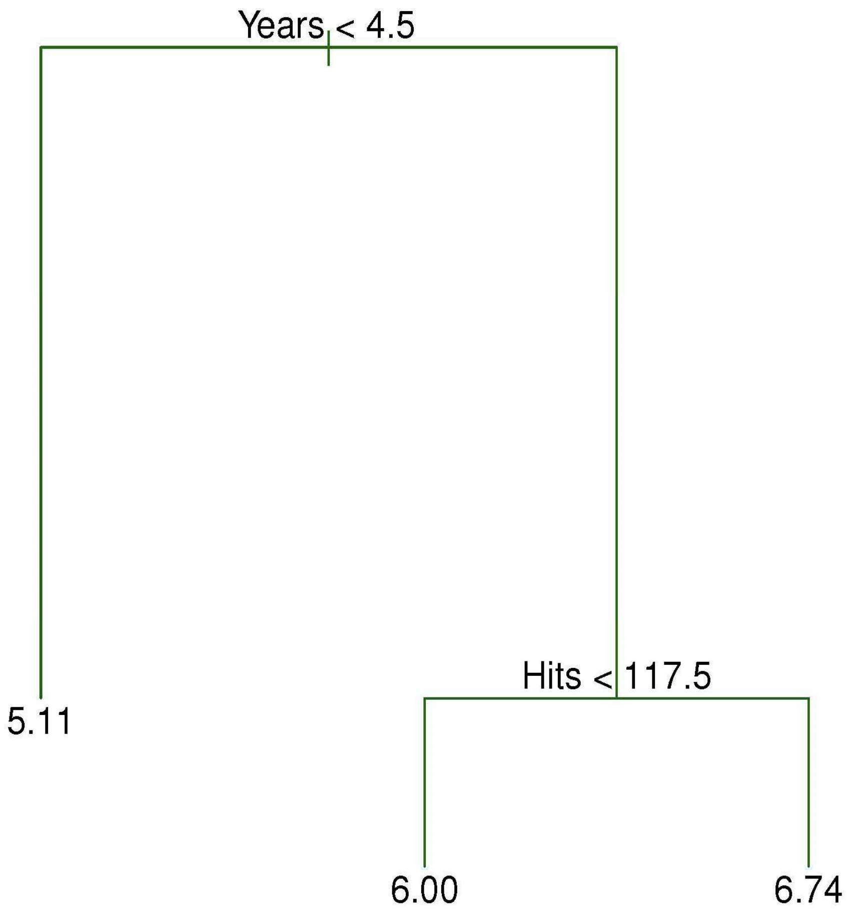
The tree segments the players into three regions of predictor space: R1 = {X | Years < 4.5}, R2 = {X | Years ≥ 4.5, Hits <117.5}, and R3 = {X | Years ≥ 4.5, Hits ≥ 117.5}. (James et al.,2021, Figure 8.2)

Regression Trees - Terminology



- The regions R1, R2, and R3 are called terminal nodes or leaves of the tree.
- The root node represents the entire sample which will further be divided into two or more nodes.
- Every subgroup in between is referred to as an internal node.
- The connections between nodes are called branches.

Regression Trees - Interpretation



- 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 seems to play little role in his salary. But among players who have been in the major leagues for longer than 4.5 years, the number of hits does affect salary, and players who made more hits tend to have higher salaries.

Regression Trees - Prediction

- Prediction of observations in regression trees
  - We divide the predictor space (the set of possible values for P predictors) into R distinct and non-overlapping regions, R_r (r = 1,…,R).
  - For every observation that falls into the region R_r, we make the same prediction, which is simply the mean of the response values for the training observations in R_r.

Regression Trees: Recursive Binary Splitting

- We construct the regions R_j via recursive binary splitting.

- This approach is top-down because it begins at the top of the tree (at which all observations belong to a single region) and then successively splits the predictor space. Each split is indicated via two new branches further down on the tree.

- It is greedy because at each step of the tree-building process, the best split is made at that particular step, considering all predictors and all possible cutpoint values for each of the predictors.

Regression Trees: Recursive Binary Splitting

- To perform recursive binary splitting, we first select the predictor X$_p$ and the cutpoint s such that splitting the predictor space into the two regions $\{X|Xp < s\}$ and $\{X|X_p \geq s\}$ leads to the greatest possible reduction in RSS

$$\mathrm{RSS} = \sum_{r=1}^{R} \sum_{i \in Rr} \left(y_i - \bar{y}_{R_r}\right)^2$$

- $\bar{y}_{R_r}$ is the mean response for the training observations within the *r*th region.

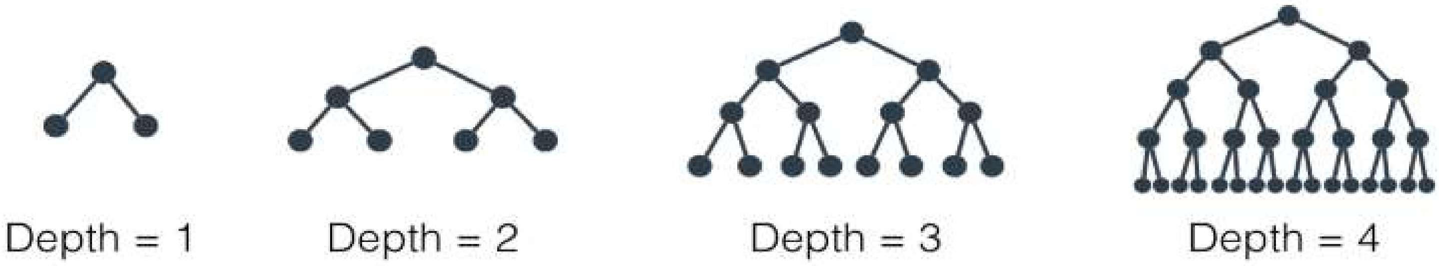
Regression Trees: Recursive Binary Splitting

- We consider all predictors and all possible values of the cutpoint s for each of the predictors, and choose the predictor and cutpoint such that the resulting tree has the lowest RSS.

- We repeat the process, looking for the best predictor and best cutpoint 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.

Regression Trees: Recursive Binary Splitting

- The process continues until a stopping criterion is reached.
  - Maximum depth: The maximum depth of a tree is the largest length between the root node to a leaf. The root node has a depth of 0. (e.g., The R tree package = 31)



Maximum depth of a decision tree

  - Minimum number of observations per leaf, e.g., each region must contain at least 5 observations.

https://julienbeaulieu.gitbook.io/wiki/sciences/machine-learning/decision-trees

Regression Trees: Tree Pruning

- This recursive binary splitting process may produce good predictions on the training set, but is likely to overfit the data, leading to poor test set performance.

- This is because the resulting tree might be too complex. A smaller tree with fewer splits (i.e., fewer regions) might lead to lower variance and better interpretation at the cost of a little bias.

- A popular strategy for achieving a smaller tree is to grow a very large tree, say T_0, and then prune it back to obtain a subtree T.
  - Cost complexity pruning

Regression Trees: Tree Pruning

- In cost complexity pruning, a sequence of trees are indexed by a nonnegative tuning parameter α.

- For each value of α there corresponds a subtree T ⊂ T_0 such that

$$\sum_{m=1}^{|T|} \sum_{i:xi\in Rm} \left(y_i - \bar{y}_{R_r}\right)^2 + \alpha C \qquad \text{Eq. (1)}$$

is as small as possible. Here C indicates the number of terminal nodes of the tree *T*, and R_m is the rectangle (i.e., the subset of predictor space) corresponding to the *m*th terminal node.

Regression Trees: Tree Pruning

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

- When $\alpha = 0$, $T = T_0$. As α increases, there is a price to pay for having a tree with many terminal nodes, and so the quantity Eq. (1) will tend to be minimized for a smaller subtree.

- We select an optimal value of α using cross validation.

- We then obtain the subtree corresponding to the optimal value of α.

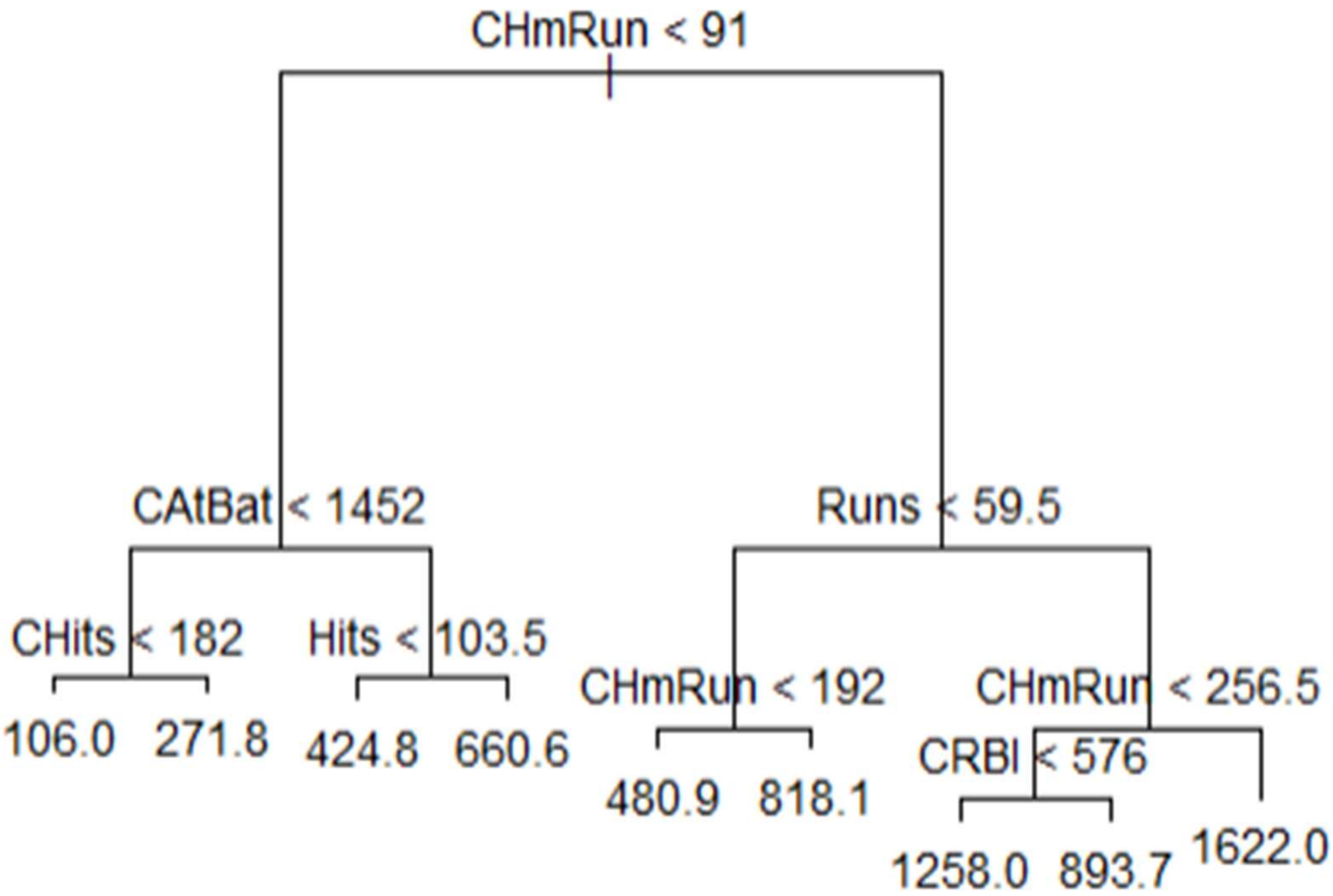
Example: Regression Trees

- We randomly divided the Hitters data in half, yielding 132 observations in the training set and 131 observations in the test set.
  - We used 19 predictors
- We then built a large regression tree on the training data.
- Finally, we performed 5-fold cross validation to determine the value of α.
  - A subtree with 8 terminal nodes

Example: Regression Tree

```
Regression tree:
tree(formula = Salary ~ ., data = mydata)
Variables actually used in tree construction:
[1] "CHmRun" "CAtBat" "CHits"  "Hits"   "Runs"   "CRBI"
Number of terminal nodes:  9
Residual mean deviance:  57830 = 7113000 / 123
Distribution of residuals:
    Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
-1087.00   -63.12   -12.80     0.00    75.63   837.80
```

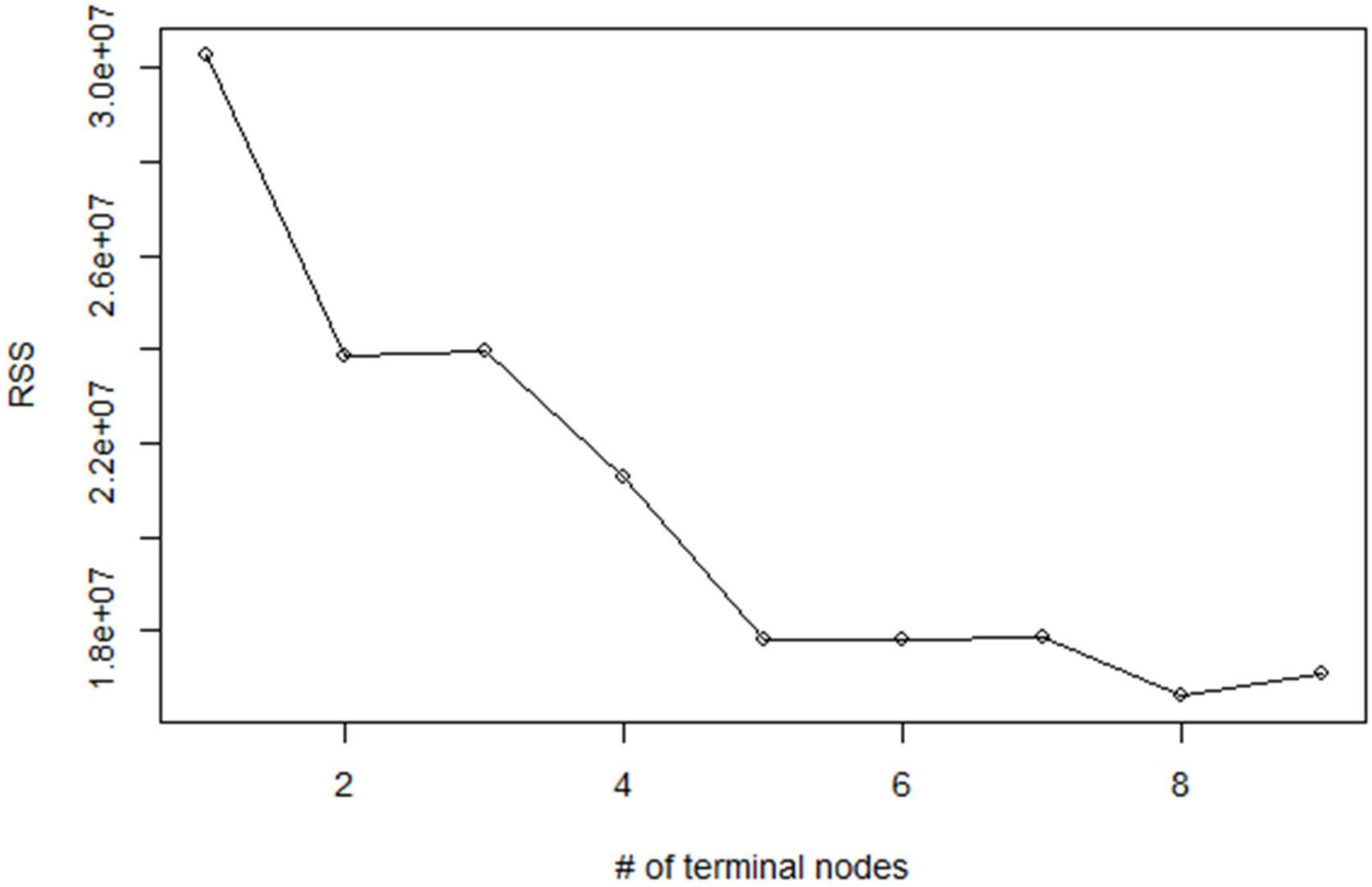
Example: Regression Trees



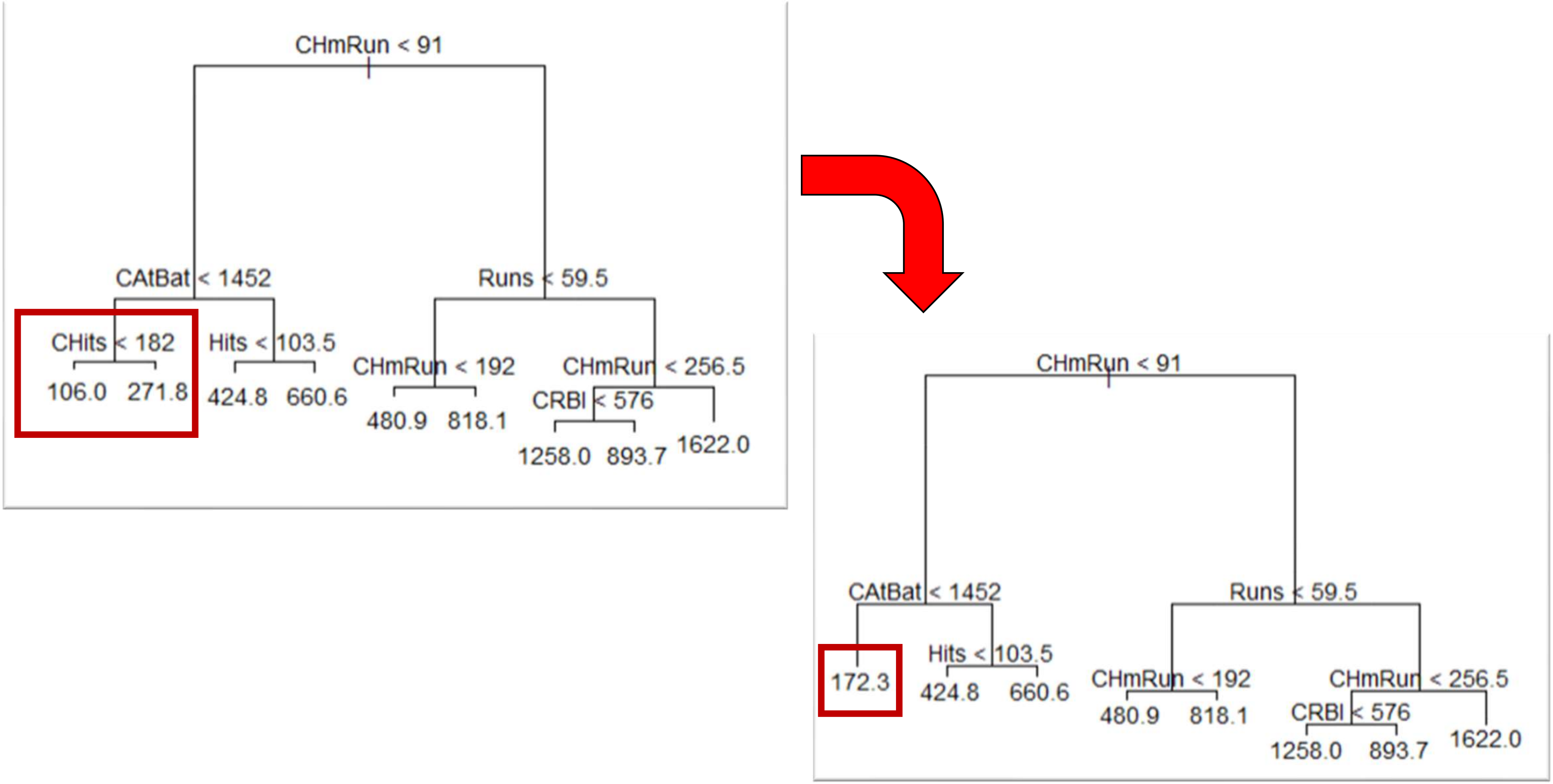
Example: Regression Trees

```
node), split, n, deviance, yval
      * denotes terminal node

 1) root 132 29870000  544.7
   2) CHmRun < 91 89  4948000  338.6
     4) CAtBat < 1452 50   507800  172.3
       8) CHits < 182 30    35160  106.0 *
       9) CHits > 182 20   143200  271.8 *
     5) CAtBat > 1452 39  1286000  551.7
      10) Hits < 103.5 18   423700  424.8 *
      11) Hits > 103.5 21   323900  660.6 *
   3) CHmRun > 91 43 13320000  971.3
     6) Runs < 59.5 18  1658000  593.3
      12) CHmRun < 192 12   484900  480.9 *
      13) CHmRun > 192 6   718100  818.1 *
     7) Runs > 59.5 25  7235000 1243.0
      14) CHmRun < 256.5 17  1886000 1065.0
        28) CRBI < 576 8   725200 1258.0 *
        29) CRBI > 576 9   598000  893.7 *
      15) CHmRun > 256.5 8  3661000 1622.0 *
```

Example: Regression Trees



Example: Regression Trees



Example: Regression Trees

```
MSE_original(test):  120203
MSE_pruned(test)  :  118489.8
```

Classification Trees

- A classification tree is very similar to a regression tree, except that it is used to predict a nominal response rather than a quantitative one.

- For a classification tree, we predict that each observation belongs to the most commonly occurring class of training observations in the region to which it belongs.

- We also use recursive binary splitting to grow a classification tree.

Classification Trees

- RSS cannot be used as a criterion for making binary splits. A natural alternative to RSS is the classification error rate, which is the fraction of the training observations in a region that do not belong to the most common class.

$$E = 1 - \max(\hat{p}_{rk}),$$

where $\hat{p}_{rk}$ is the proportion of training observations in the *r*th region that are from the *k*th class.

- However, classification error is not sufficiently sensitive for tree-growing.

Classification Trees

- In practice, two other criteria are preferable – the Gini index and entropy.

- The Gini index is defined by

$$G = \sum_{k=1}^{K} \hat{p}_{rk}(1 - \hat{p}_{rk})\,.$$

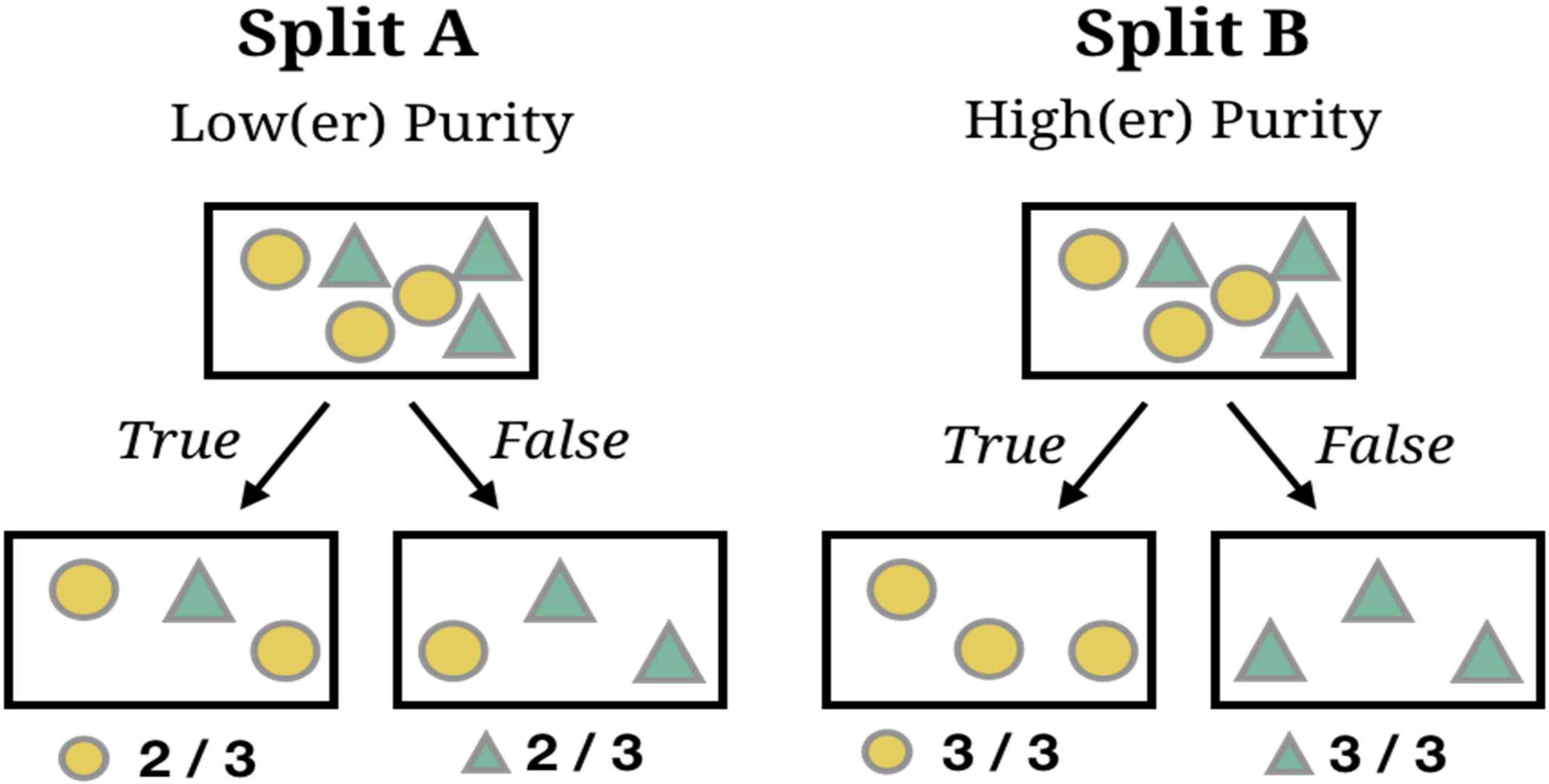
- This is a measure of total variance across the *K* classes. It takes a small value if all of the $\hat{p}_{rk}$'s are close to 0 or 1. Thus, the Gini index is referred to as a measure of node purity – a small value indicates that a node contains predominantly observations from a single class.

Classification Trees

- The entropy is defined by

$$D = -\sum_{k=1}^{K} \hat{p}_{rk} \log(\hat{p}_{rk})\,.$$

  - It takes a small value if all of the $\hat{p}_{rk}$'s are close to 0 or 1. Thus, like the Gini index, the entropy will take on a small value if a node is pure.



Split A

Low(er) Purity

True

False

● 2 / 3

▲ 2 / 3

Split B

High(er) Purity

True

False

● 3 / 3

▲ 3 / 3

https://therbootcamp.github.io/AML_2020AMLD/_sessions/Prediction/Prediction.html#13

Classification Trees

- When building a classification tree, either the Gini index or the entropy is typically used to evaluate the quality of a particular split.
  - Our goal is to have these metrics reach 0 where it will be minimally impure and maximally pure falling into one category.

- Any of the three criteria might be used when pruning the tree, but the classification error rate is preferable if the prediction accuracy of the final pruned tree is the goal.

Classification Trees

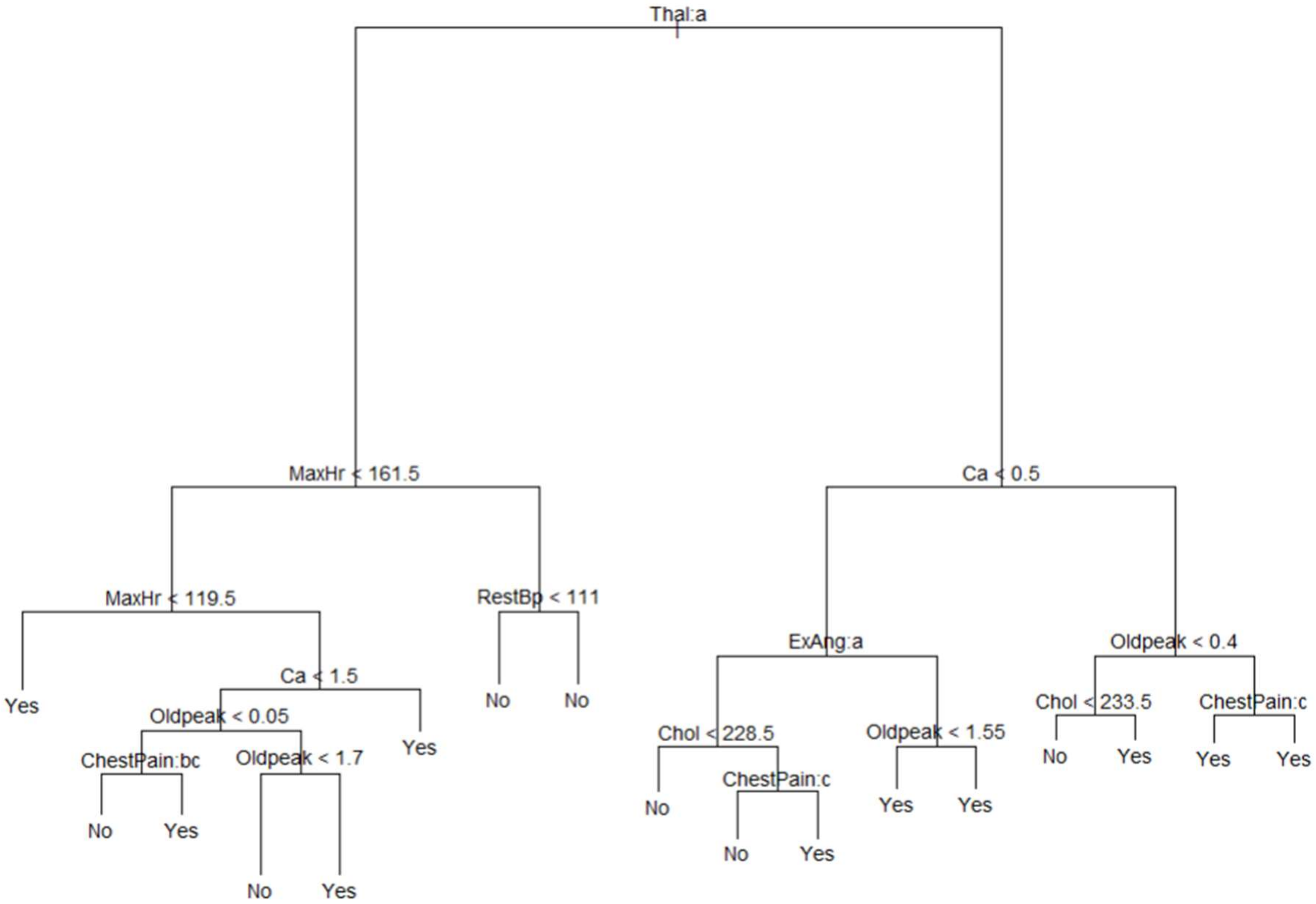
- The Heart data (Heart_training.csv & Heart_test.csv)
  - Y = HD (1 = heart disease, 0 = no heart disease).
  - X = 13 predictors, including Age, Sex, Chol (a cholesterol measurement), chest pain, etc.
  - Cross validation yields a tree with 4 terminal nodes.

* Janosi, Andras, Steinbrunn, William, Pfisterer, Matthias, Detrano, Robert & M.D., M.D.. (1988). Heart Disease. UCI Machine Learning Repository.

Example: Classification Tree

```
Classification tree:
tree(formula = HeartDisease ~ ., data = mydata2)
Variables actually used in tree construction:
[1] "Thal"      "MaxHr"     "Ca"        "Oldpeak"   "ChestPain"
"RestBp"    "ExAng"     "Chol"
Number of terminal nodes:  17
Residual mean deviance:  0.4774 = 90.71 / 190
Misclassification error rate: 0.1159 = 24 / 207
```

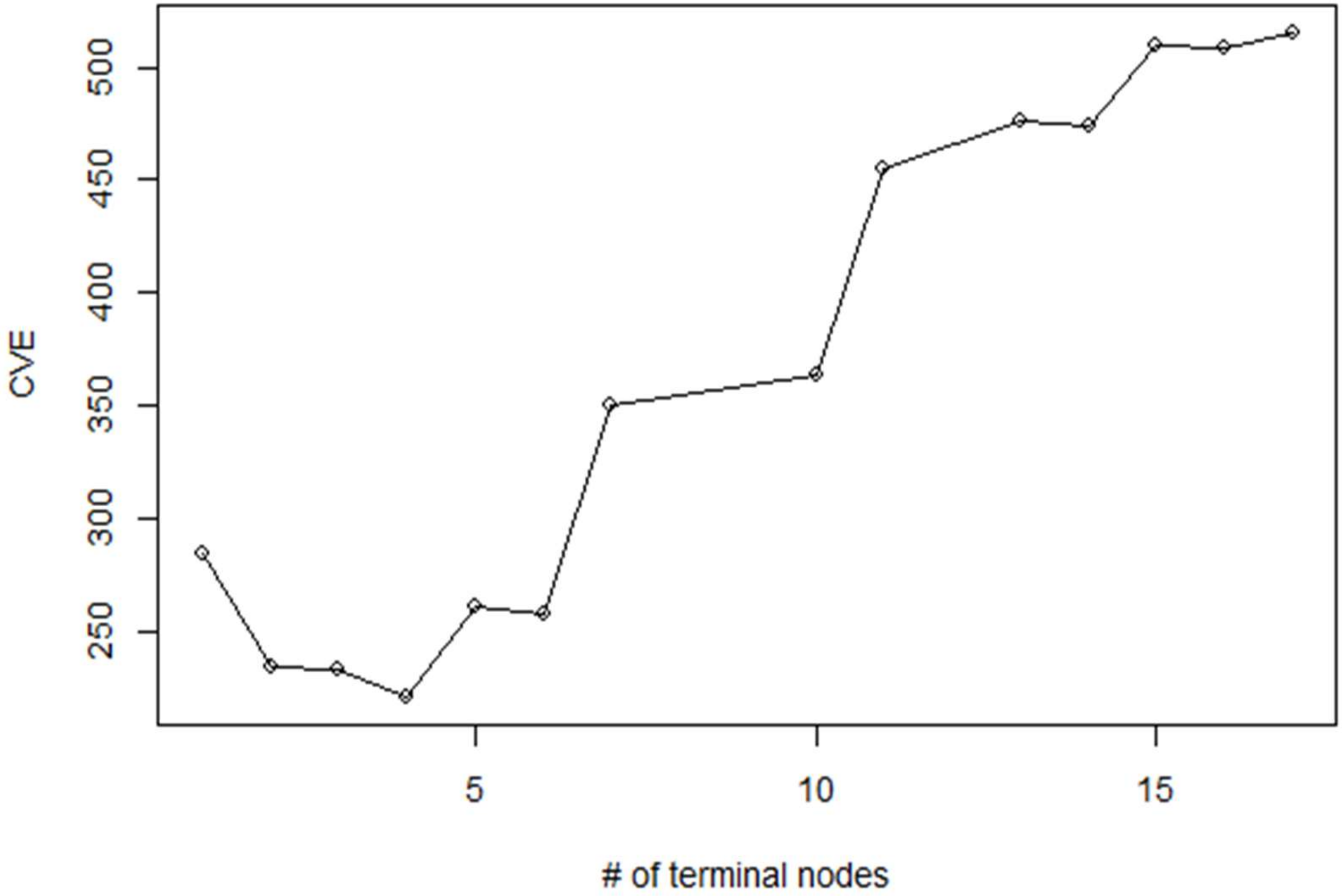
Example: Classification Tree



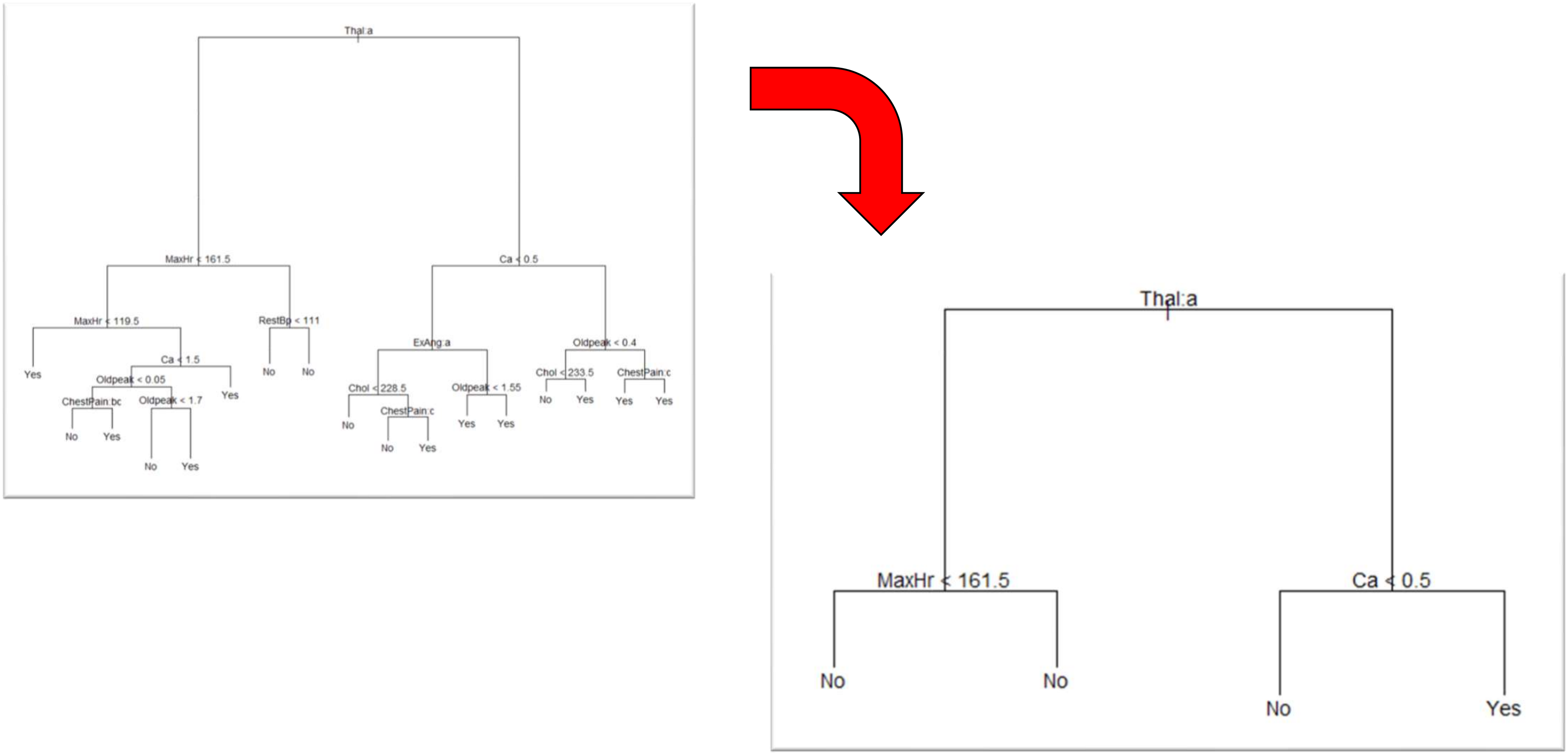
Example: Classification Tree

```
node), split, n, deviance, yval, (yprob)
      * denotes terminal node

 1) root 207 281.700 No ( 0.57971 0.42029 )
   2) Thal: 3 116 112.700 No ( 0.81034 0.18966 )
     4) MaxHr < 161.5 64  79.500 No ( 0.68750 0.31250 )
       8) MaxHr < 119.5 7   5.742 Yes ( 0.14286 0.85714 ) *
       9) MaxHr > 119.5 57  63.550 No ( 0.75439 0.24561 )
        18) Ca < 1.5 51  50.480 No ( 0.80392 0.19608 )
          36) Oldpeak < 0.05 19  25.010 No ( 0.63158 0.36842 )
            72) ChestPain: 2,3 9   6.279 No ( 0.88889 0.11111 ) *
            73) ChestPain: 1,4 10  13.460 Yes ( 0.40000 0.60000 ) *
          37) Oldpeak > 0.05 32  19.910 No ( 0.90625 0.09375 )
            74) Oldpeak < 1.7 27   0.000 No ( 1.00000 0.00000 ) *
            75) Oldpeak > 1.7 5   6.730 Yes ( 0.40000 0.60000 ) *
        19) Ca > 1.5 6   7.638 Yes ( 0.33333 0.66667 ) *
     5) MaxHr > 161.5 52  16.950 No ( 0.96154 0.03846 )
      10) RestBp < 111 6   7.638 No ( 0.66667 0.33333 ) *
      11) RestBp > 111 46   0.000 No ( 1.00000 0.00000 ) *
   3) Thal: 6,7 91 108.900 Yes ( 0.28571 0.71429 )
                  …
```

Example: Classification Tree



Example: Classification Tree



Example: Classification Tree

Without Pruning

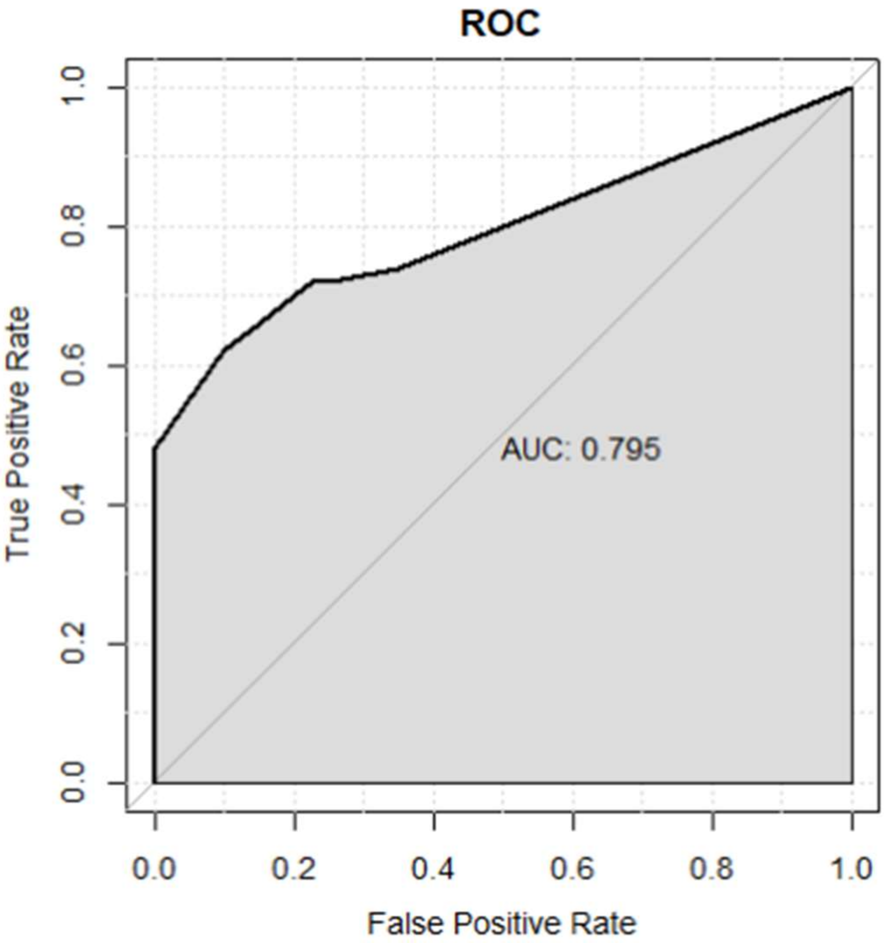
```
Confusion Matrix and Statistics

          Reference
Prediction No Yes
       No  31  14
       Yes  9  36

               Accuracy : 0.7444
                 95% CI : (0.6416, 0.8306)
    No Information Rate : 0.5556
    P-Value [Acc > NIR] : 0.0001664

                  Kappa : 0.4889

 Mcnemar's Test P-Value : 0.4042485

            Sensitivity : 0.7200
            Specificity : 0.7750
         Pos Pred Value : 0.8000
         Neg Pred Value : 0.6889
             Prevalence : 0.5556
         Detection Rate : 0.4000
   Detection Prevalence : 0.5000
      Balanced Accuracy : 0.7475

       'Positive' Class : Yes
```

With Pruning

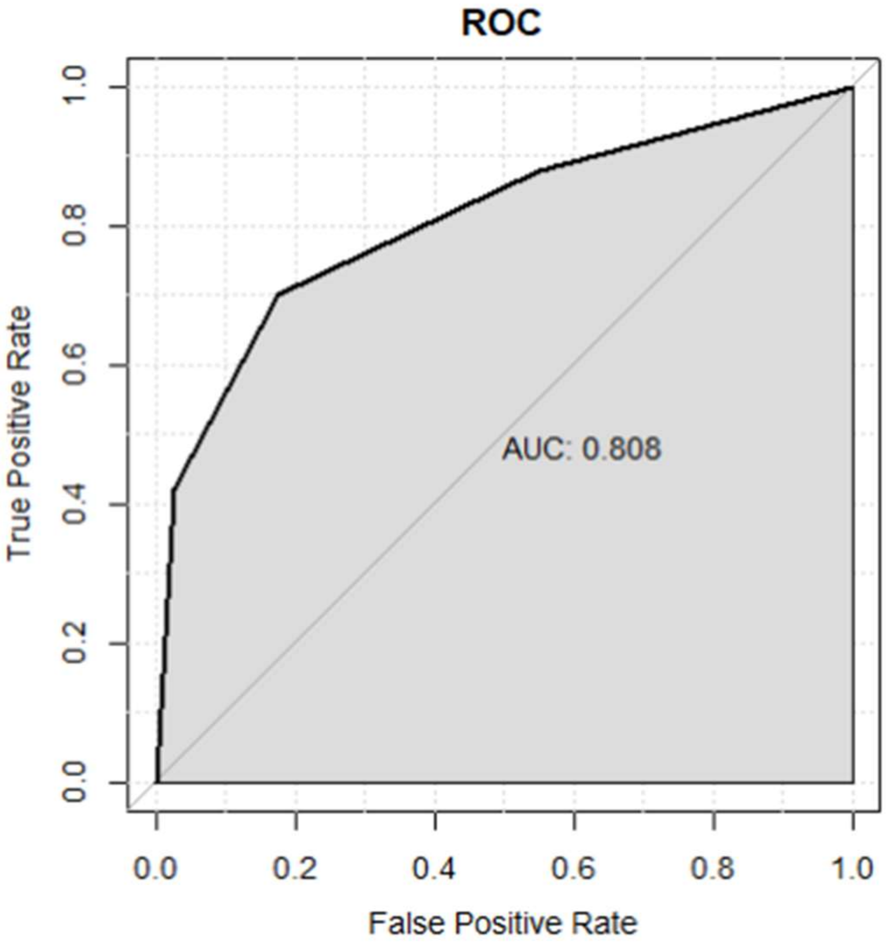
```
Confusion Matrix and Statistics

          Reference
Prediction No Yes
       No  39  29
       Yes  1  21

               Accuracy : 0.6667
                 95% CI : (0.5595, 0.7626)
    No Information Rate : 0.5556
    P-Value [Acc > NIR] : 0.02099

                  Kappa : 0.3692

 Mcnemar's Test P-Value : 8.244e-07

            Sensitivity : 0.4200
            Specificity : 0.9750
         Pos Pred Value : 0.9545
         Neg Pred Value : 0.5735
             Prevalence : 0.5556
         Detection Rate : 0.2333
   Detection Prevalence : 0.2444
      Balanced Accuracy : 0.6975

       'Positive' Class : Yes
```

Example: Classification Tree

Without Pruning



With Pruning



Advantages and Disadvantages of Trees

- Advantages:
  - Trees are easy to explain to people.
  - Some people believe that decision trees more closely mirror human decision-making than do the regression and classification approaches discussed in previous sessions.
  - Trees can be displayed graphically and easily interpreted even by non-experts (especially, if they are small).
  - Trees can easily handle nominal predictors without the need to create dummy variables.

Advantages and Disadvantages of Trees

- Disadvantages:
  - Trees generally do not have the same level of predictive accuracy as some other regression and classification approaches.
  - A tree can be very non-robust. A small change in the data can cause a large change in the final estimated tree.

- Using methods like bagging, random forests, and boosting, the predictive performance of trees can be substantially improved.
  - Also known as ensemble methods

Bagging

- Bootstrap aggregation, or bagging, is a general-purpose method for reducing the variance of a statistical method. It is particularly useful and frequently used in the context of decision trees.

- In general, we generate *B* bootstrapped training data sets and train our method on each bootstrapped set to get the resulting prediction $\hat{f}_b(x)$, and finally average all the predictions to obtain

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

Bagging

- To apply bagging to regression trees, we simply construct B regression trees using B bootstrapped training datasets and average the resulting predictions.

- These trees are grown deep and are not pruned. So, each individual tree has high variance, but low bias. Averaging all B trees reduces the variance.

- Bagging can give impressive improvements in prediction accuracy by combining a large number of trees into a single procedure.

Bagging

- When applying bagging to classification trees, for a given observation, we can record the class predicted by each of B classification trees and take a majority vote: the overall prediction is the most commonly occurring class among the B predictions.

- The number of trees (B) is not a critical parameter with bagging. Using a very large value of B will not lead to overfitting.

Bagging: Out-of-Bag Error

- In bagging, there is a very straightforward way of estimating the test error of a model, without the need to perform cross validation.

- It is known that on average, each bootstrapped training data set contains around 2/3 of all training observations. The remaining 1/3 of the observations not used are called the out-of-bag (OOB) observations.

- We can apply each tree to the OOB observations and average the predicted responses over B OOB datasets, leading to the OOB error.

Bagging: Out-of-Bag Error

- The OOB error is a valid estimate of the test error for the bagged model.

- If B is sufficiently large, the OOB error is virtually equivalent to the leave-one-out cross-validation error.

- The OOB approach for estimating the test error is particularly convenient when performing bagging on large data sets, for which cross validation would be computationally burdensome.

Bagging: Variable Importance

- Bagging typically results in improved prediction accuracy over using a single tree.

- However, when we bag a large number of trees, it is no longer possible to interpret the resulting model like using a single tree. So, it is no longer clear which variables are most important to the procedure.

- Bagging improves prediction at the expense of interpretability.

Bagging: Variable Importance

- But we can obtain an overall summary of the importance of each predictor.

- In the case of bagging regression trees, we can 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.

- In the case of bagging classification trees, similarly, we can sum up the total amount that the Gini index is decreased due to splits over a given predictor, averaged over all B trees.

Example: Bagging

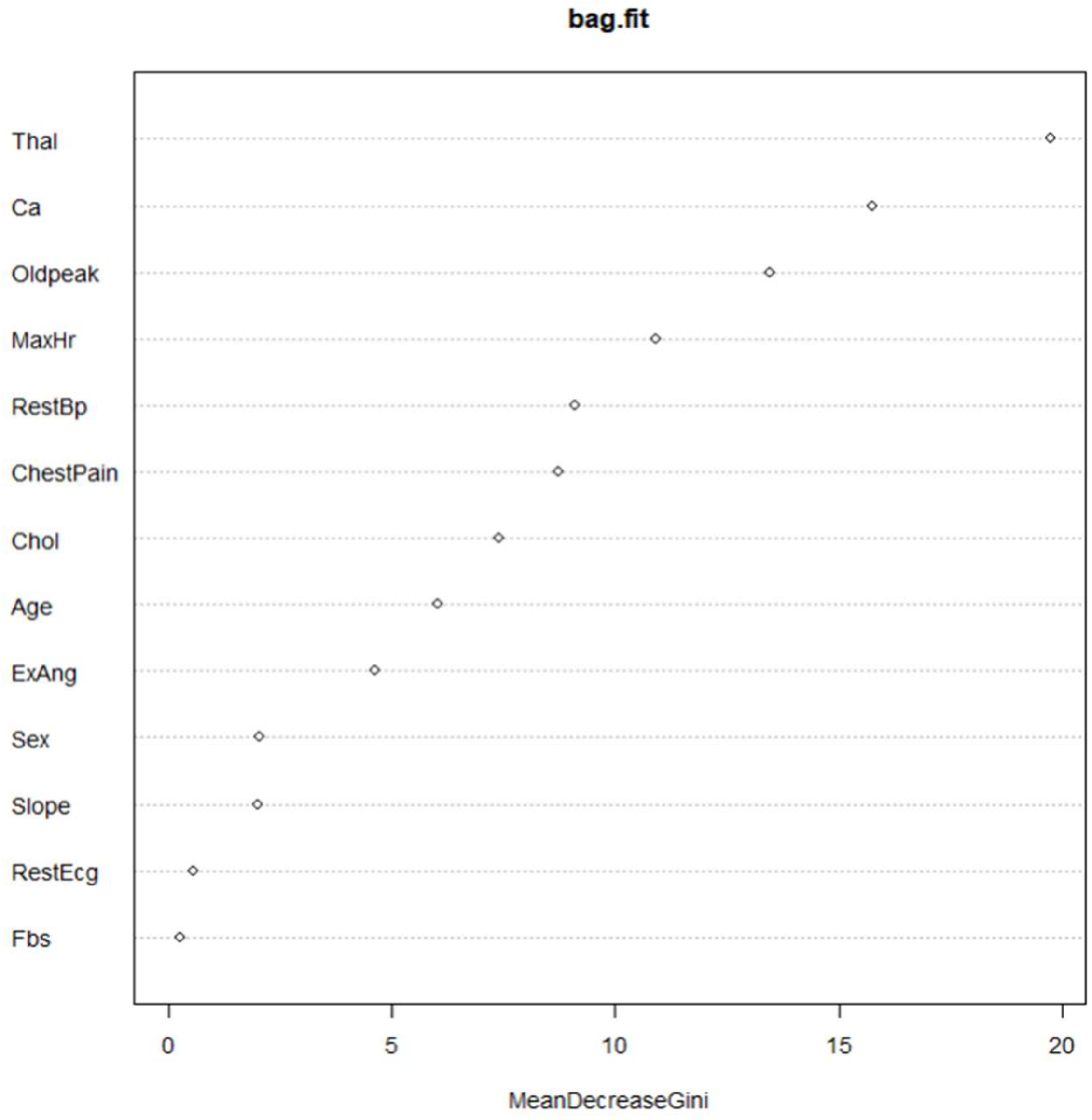
- The Heart data (Heart_training.csv & Heart_test.csv)
  - Y = HD (1 = heart disease, 0 = no heart disease).
  - X = 13 predictors, including Age, Sex, Chol (a cholesterol measurement), chest pain, etc.

* Janosi, Andras, Steinbrunn, William, Pfisterer, Matthias, Detrano, Robert & M.D., M.D.. (1988). Heart Disease. UCI Machine Learning Repository.

Example: Bagging

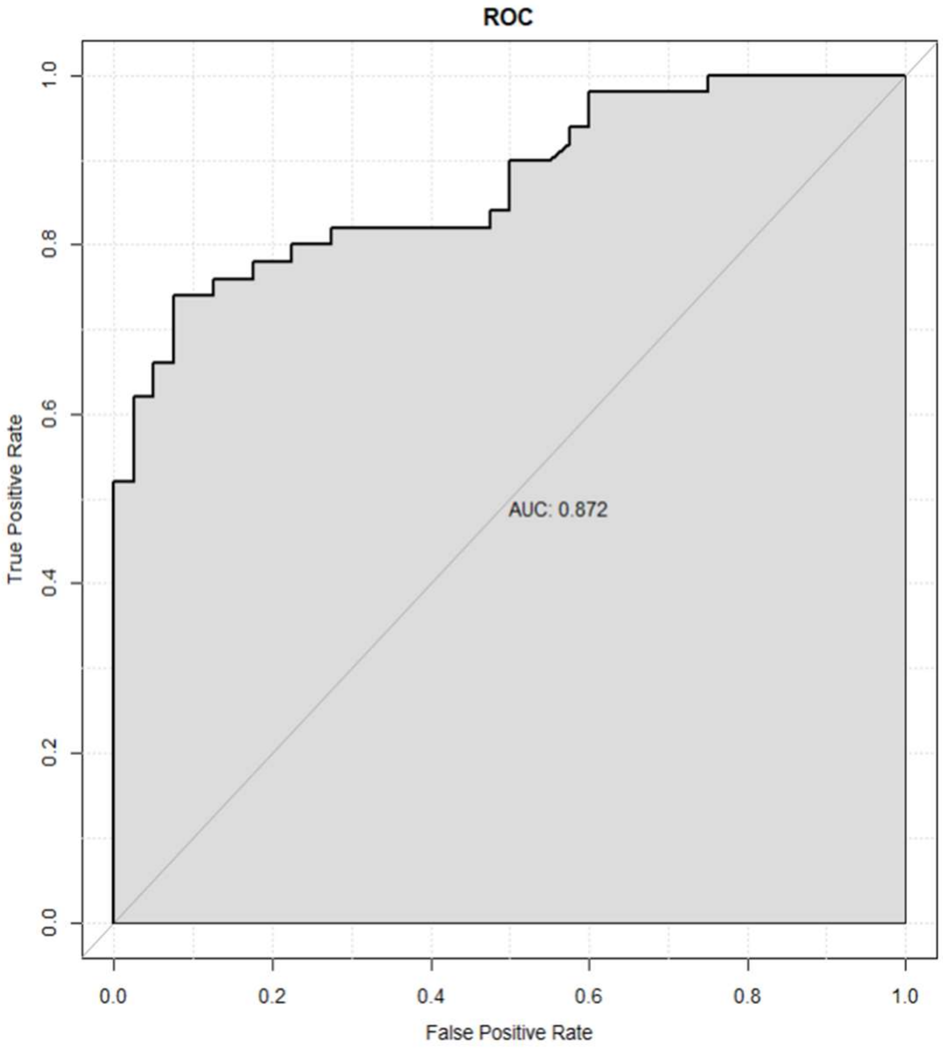
```
Call:
 randomForest(formula = HeartDisease ~ ., data = mydata2,
                          mtry = 13,importance = TRUE)

Type of random forest: classification
Number of trees: 500
No. of variables tried at each split: 13
OOB estimate of  error rate: 20.77%

Confusion matrix:
     No Yes class.error
No  101  19   0.1583333
Yes  24  63   0.2758621
```

Example: Bagging



Example: Bagging

```
Confusion Matrix and Statistics

          Reference
Prediction No Yes
       No  37  14
       Yes  3  36

               Accuracy : 0.8111
                 95% CI : (0.7149, 0.8859)
    No Information Rate : 0.5556
    P-Value [Acc > NIR] : 3.097e-07

                  Kappa : 0.6277

 Mcnemar's Test P-Value : 0.01529

            Sensitivity : 0.7200
            Specificity : 0.9250
         Pos Pred Value : 0.9231
         Neg Pred Value : 0.7255
             Prevalence : 0.5556
         Detection Rate : 0.4000
   Detection Prevalence : 0.4333
      Balanced Accuracy : 0.8225

       'Positive' Class : Yes
```



Random Forests

- Random forests provide an improvement over bagged trees by way of a small tweak that decorrelates the trees.

- As in bagging, we build a number of decision trees on bootstrapped training data sets. But when building these decision trees, each time a split in a tree 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 the m predictors.

- We may choose $m = \sqrt{P}$ (e.g., 4 out of 13 predictors) for classification trees, and $m = P/3$ for regression trees.

Random Forests

- Suppose that there is one very strong predictor in the data set, along with many other moderately strong predictors. Then, most of bagged trees will use this strong predictor in the top split. Consequently, all of the bagged trees will look quite similar to each other. Hence, the predictions from the bagged trees will be highly correlated.

- Unfortunately, averaging many highly correlated quantities does not lead to as large of a reduction in variance as averaging many uncorrelated quantities.

- This means that bagging may not lead to a substantial reduction in variance over a single tree in this setting.

Random Forests

- Random forests overcome this problem by forcing each split to consider only a subset of the predictors. Therefore, there might be a higher chance to consider other moderately strong predictors.

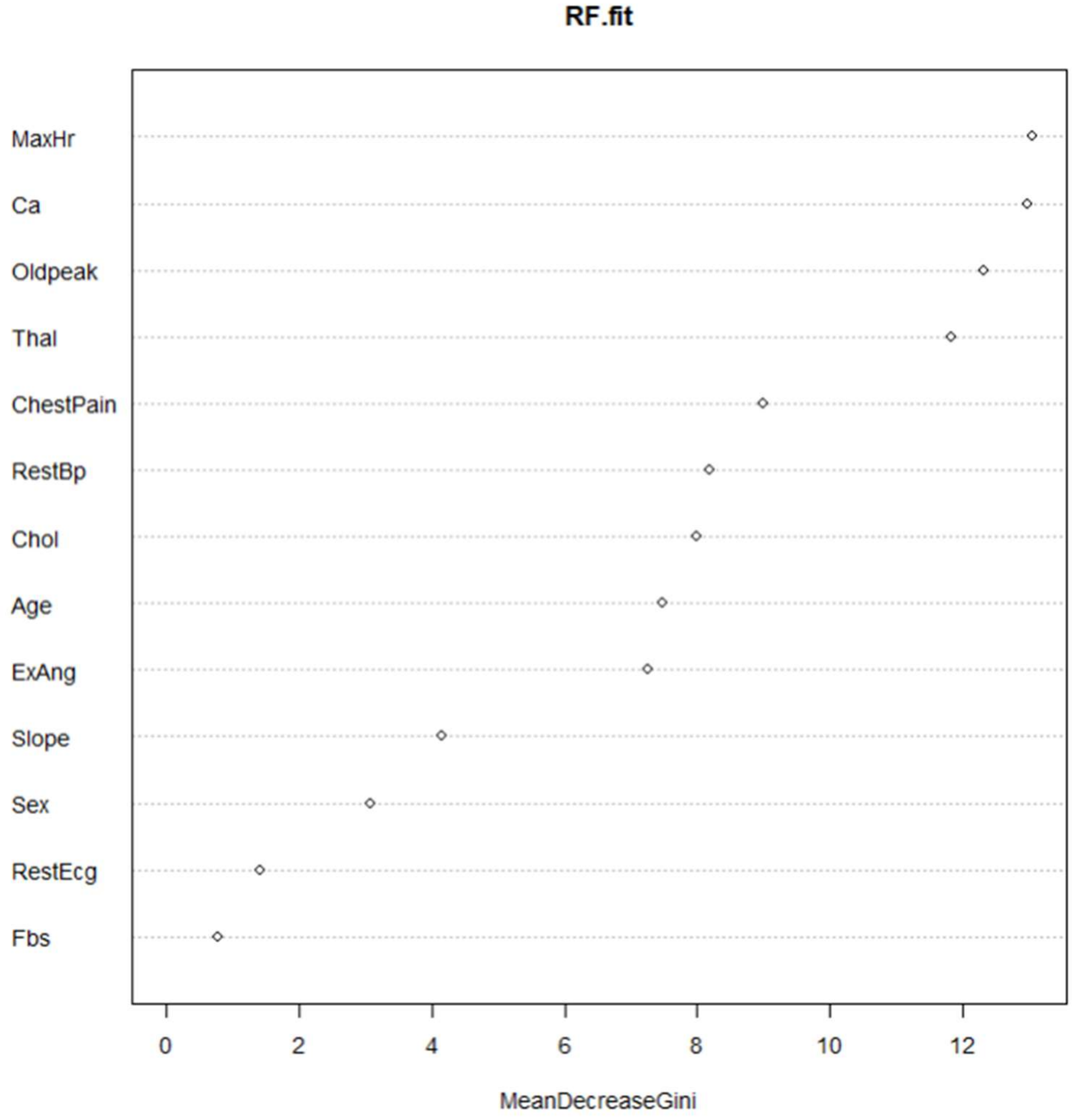
- This process can be considered decorrelating the trees, thereby making the average of the resulting trees less variable.

- If $m = P$, a random forest is equivalent to bagging. Using $m < P$ in a random forest is helpful when we have a large number of correlated predictors.

Example: Random Forests

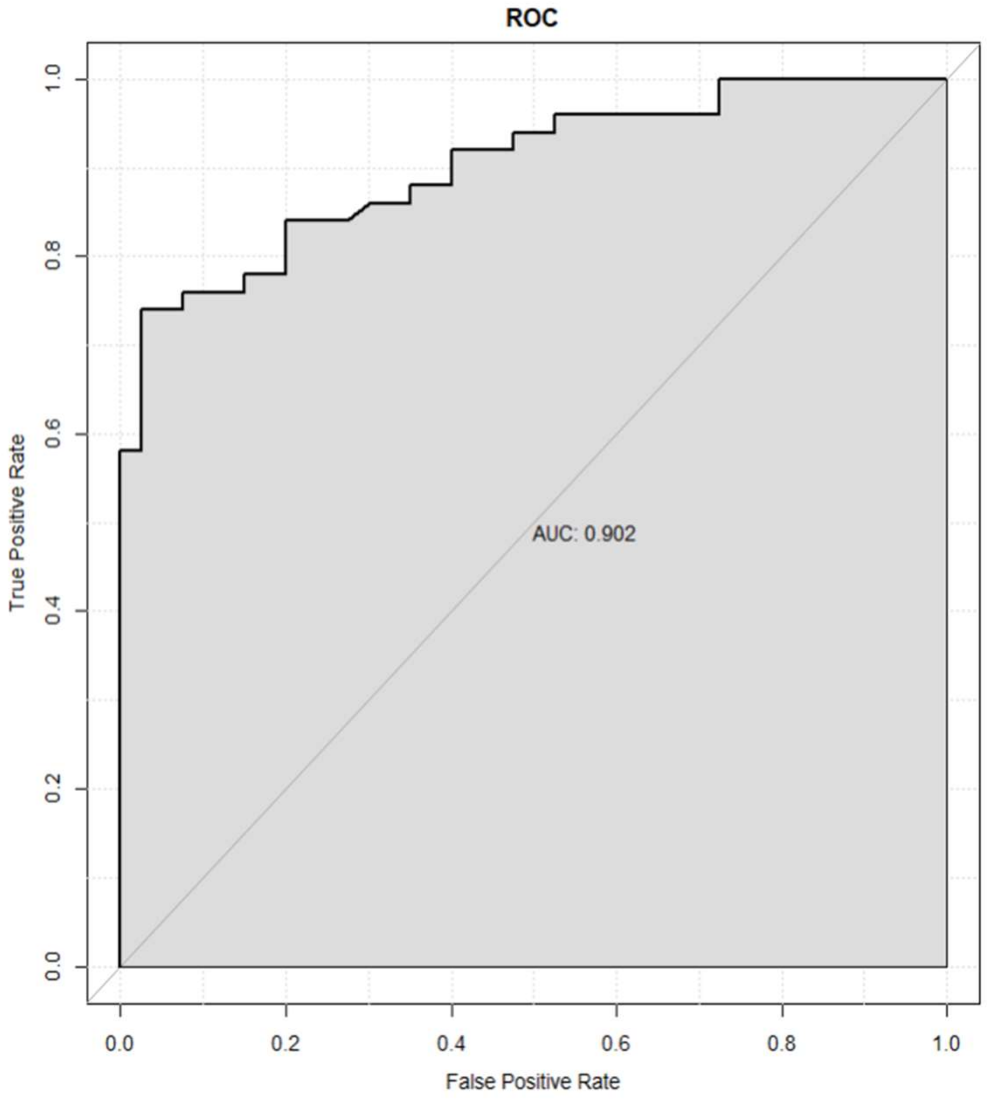
```
Call:
 randomForest(formula = HeartDisease ~ ., data = mydata2,
                importance = TRUE)

Type of random forest: classification
Number of trees: 500
No. of variables tried at each split: 3
OOB estimate of  error rate: 17.39%

Confusion matrix:
     No Yes class.error
No  105  15   0.1250000
Yes  21  66   0.2413793
```

Example: Random Forests



Example: Random Forests

```
Confusion Matrix and Statistics

          Reference
Prediction No Yes
       No  39  13
       Yes  1  37

               Accuracy : 0.8444
                 95% CI : (0.7528, 0.9123)
    No Information Rate : 0.5556
    P-Value [Acc > NIR] : 5.419e-09

                  Kappa : 0.6942

 Mcnemar's Test P-Value : 0.003283

            Sensitivity : 0.7400
            Specificity : 0.9750
         Pos Pred Value : 0.9737
         Neg Pred Value : 0.7500
             Prevalence : 0.5556
         Detection Rate : 0.4111
   Detection Prevalence : 0.4222
      Balanced Accuracy : 0.8575

       'Positive' Class : Yes
```



Boosting for Regression

- Like bagging, boosting is a general approach that can be applied to many statistical methods for regression or classification.

- Boosting also builds a decision tree on each of different training data sets and averages the resulting predictions.

- It grows trees sequentially: each tree is grown using information from previously grown trees.

- It does not involve bootstrap sampling; instead, each tree is fit on a modified version of the original data set.

Boosting for Regression

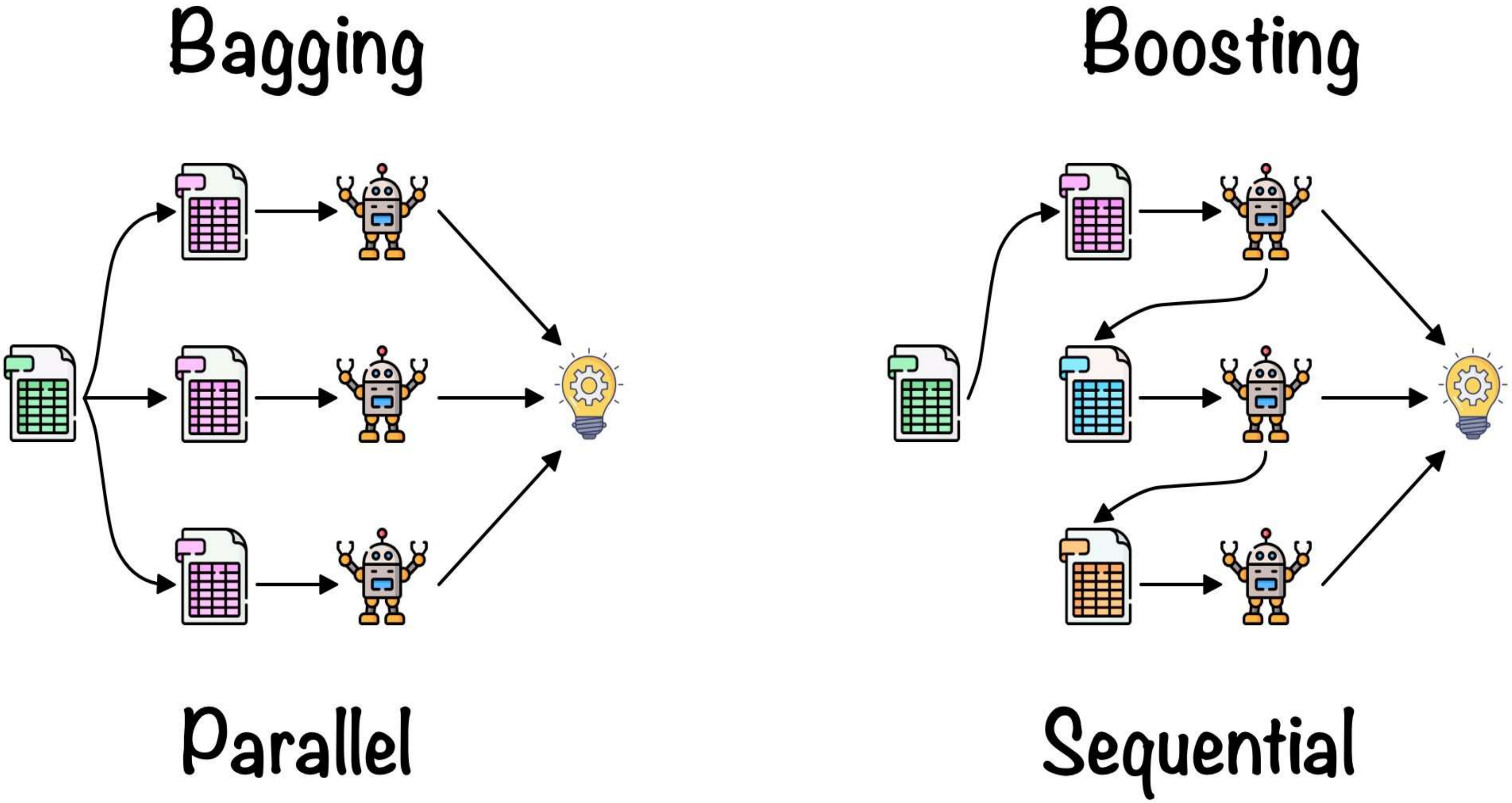
- In boosting, we start with fitting a small decision tree (e.g., with only one split) to the data and obtain the residuals of the data. Subsequently, we fit a new decision tree to the residuals and update the residuals. We continue the process B times and average the resulting predictions.

- Each of the B trees can be small, with just a few terminal nodes. By fitting small trees to the residuals, we slowly improve the predictions in areas where it does not perform well.

Boosting for Classification

- Boosting for classification is similar in spirit to boosting for regression but is a bit more complex. We will not go into detail here.
  - Refer to Hastie et al., (2009, Elements of Statistical Learning, Chapter 10).

- The R package gbm (gradient boosted models) handles a variety of regression and classification problems.



Bagging

Boosting

Parallel

Sequential

https://towardsdatascience.com/tagged/bagging

Boosting

- Boosting has three tuning parameters.

  - The number of trees (B). Unlike bagging and random forests, boosting can overfit if B is too large. We use cross validation to select B.

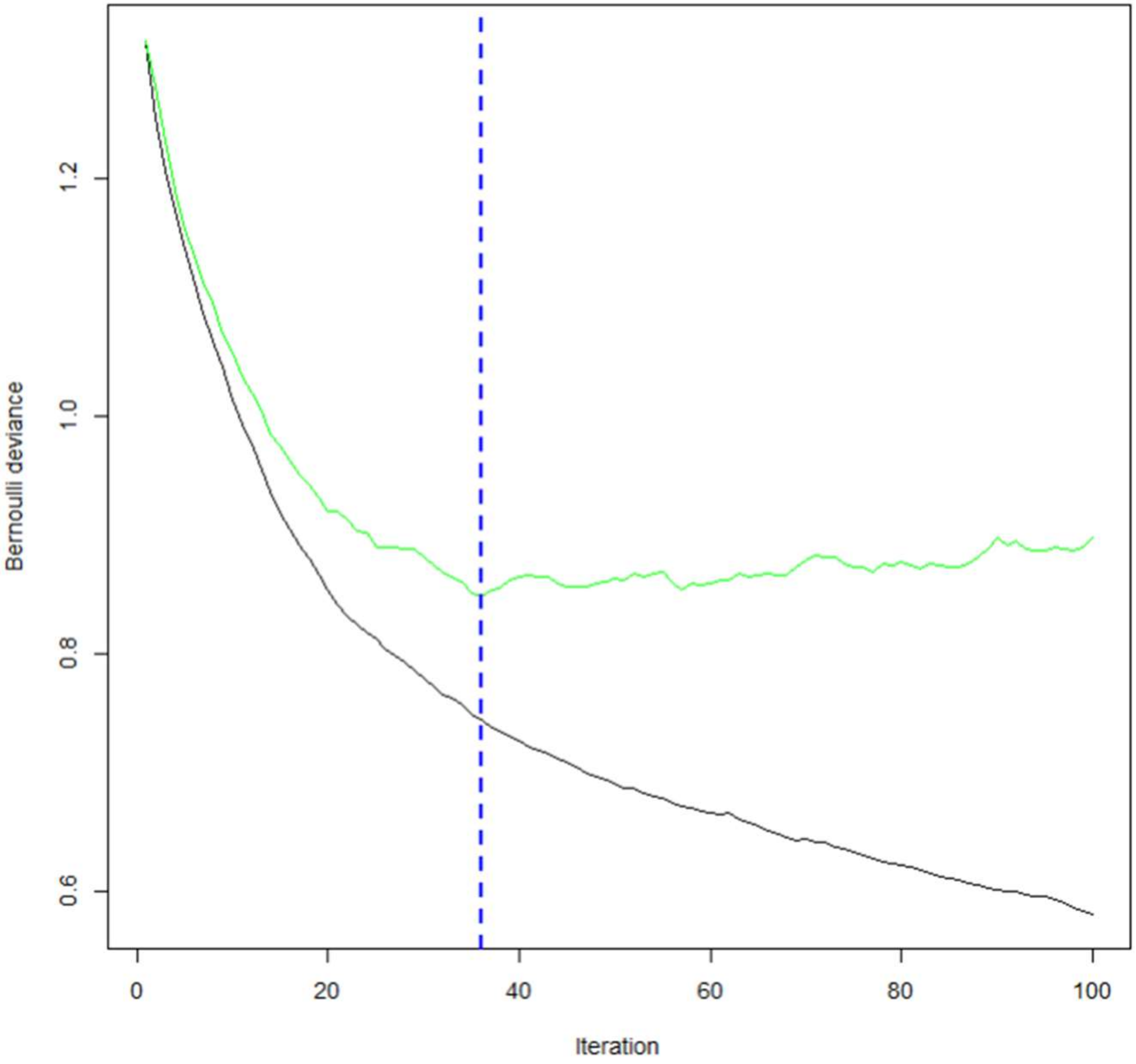
  - The shrinkage parameter (λ), a small positive number. This controls the rate at which boosting learns. Typical values are 0.01 or .001, and the right choice depends on the problem. Very small λ can require using a very large value of B to achieve good performance.

  - The number of splits (d) in each tree. Often d = 1 works well.

Example: Boosting

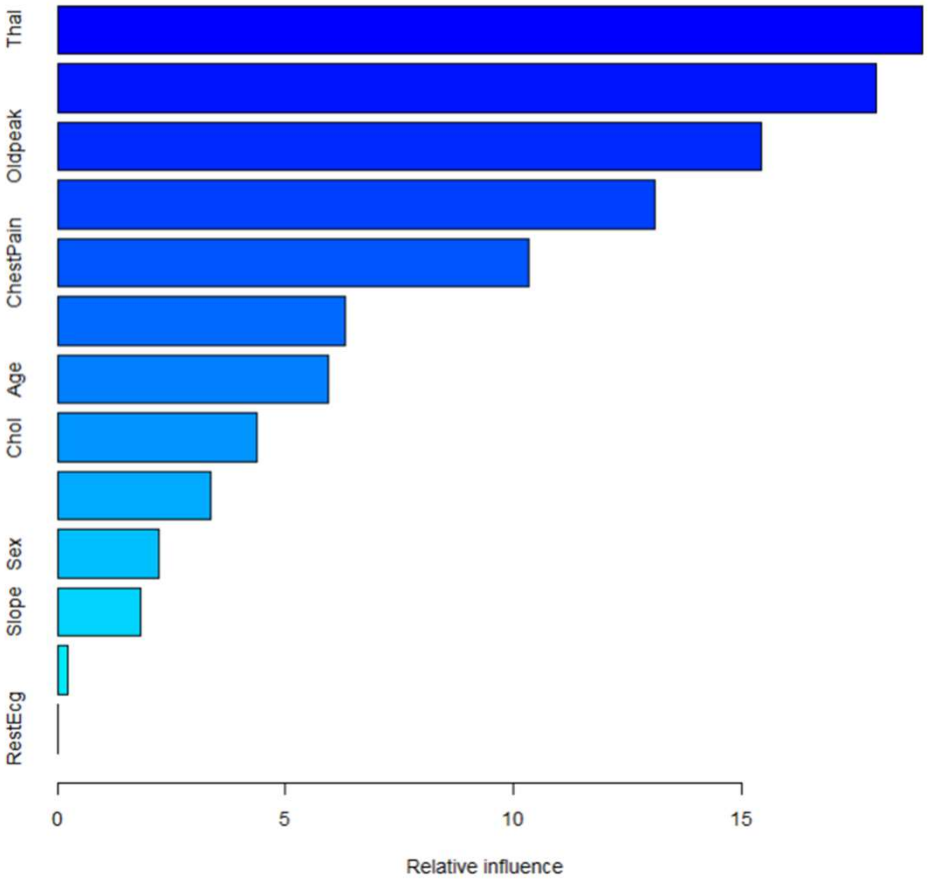
```
gbm(formula = HeartDisease ~ ., distribution = "bernoulli",
    data = mydata3, cv.folds = 5)

A gradient boosted model with bernoulli loss function.
100 iterations were performed.
The best cross-validation iteration was 36.
There were 13 predictors of which 11 had non-zero influence.
```

Example: Boosting



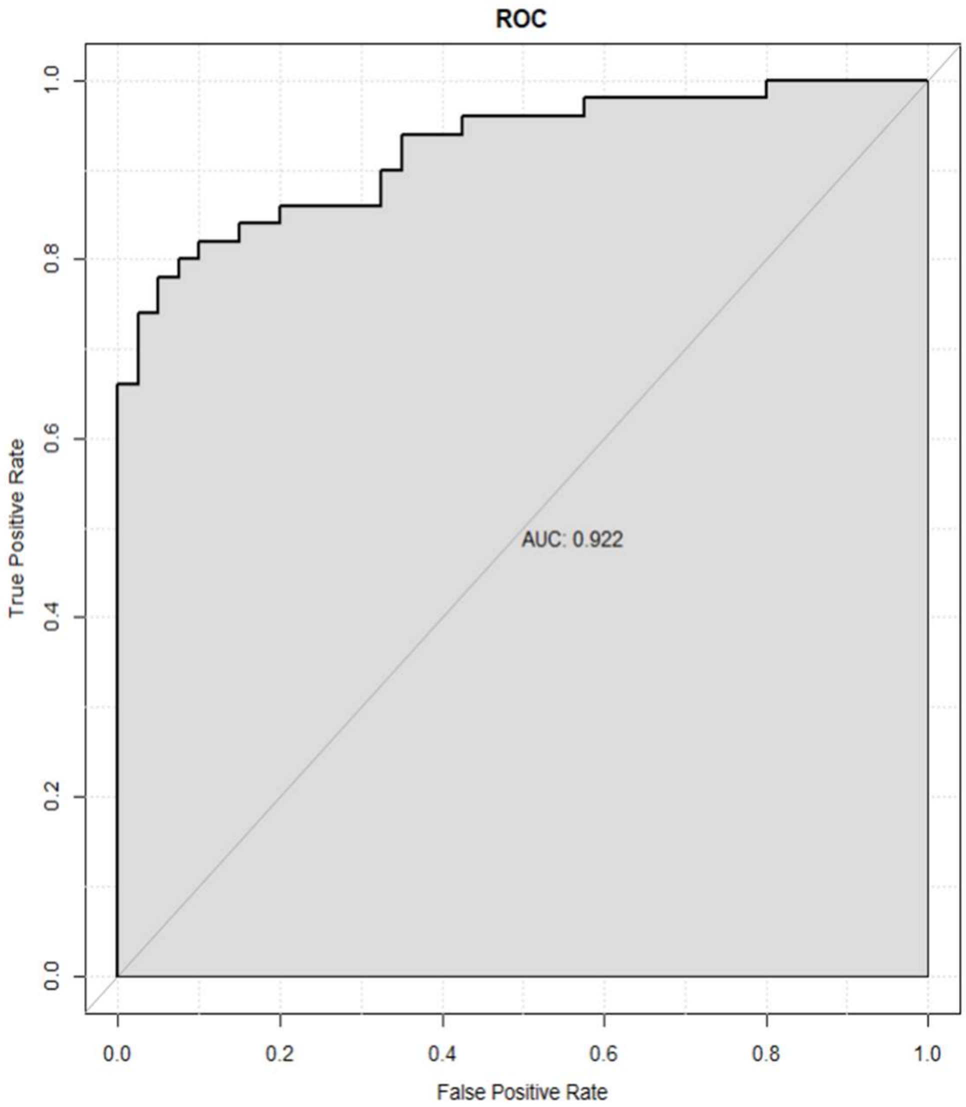
Example: Boosting

```
                var     rel.inf
Thal           Thal 18.9557929
Ca               Ca 17.9462722
Oldpeak     Oldpeak 15.4244102
MaxHr         MaxHr 13.0965419
ChestPain ChestPain 10.3311326
ExAng         ExAng  6.2973224
Age             Age  5.9277788
Chol           Chol  4.3763586
RestBp       RestBp  3.3601174
Sex             Sex  2.2291036
Slope         Slope  1.8145368
Fbs             Fbs  0.2406326
RestEcg     RestEcg  0.0000000
```

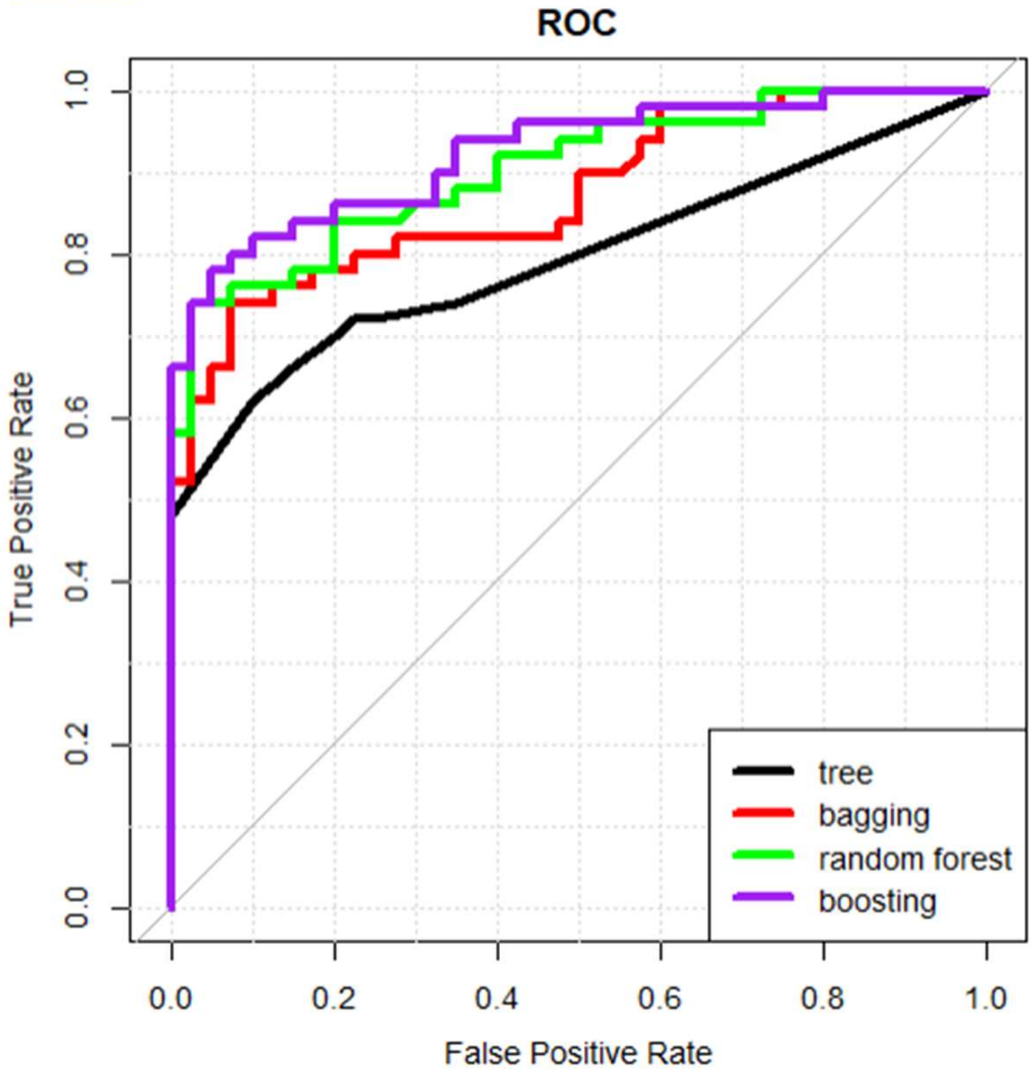


Example: Boosting

```
Confusion Matrix and Statistics

          Reference
Prediction No Yes
       No  39  14
       Yes  1  36

               Accuracy : 0.8333
                 95% CI : (0.74, 0.9036)
    No Information Rate : 0.5556
    P-Value [Acc > NIR] : 2.25e-08

                  Kappa : 0.6731

 Mcnemar's Test P-Value : 0.001946

            Sensitivity : 0.7200
            Specificity : 0.9750
         Pos Pred Value : 0.9730
         Neg Pred Value : 0.7358
             Prevalence : 0.5556
         Detection Rate : 0.4000
   Detection Prevalence : 0.4111
      Balanced Accuracy : 0.8475

       'Positive' Class : Yes
```



Example: Method Comparison



```
AUC(tree)          :  0.79525
AUC(bagging)       :  0.87225
AUC(random forest):  0.90175
AUC(boosting)      :  0.9225
```