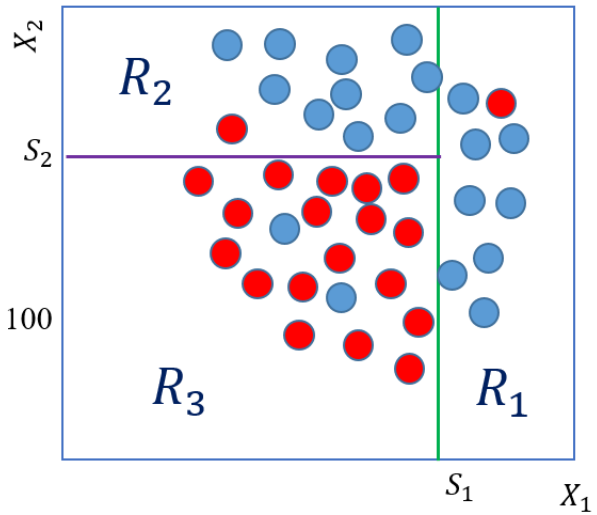
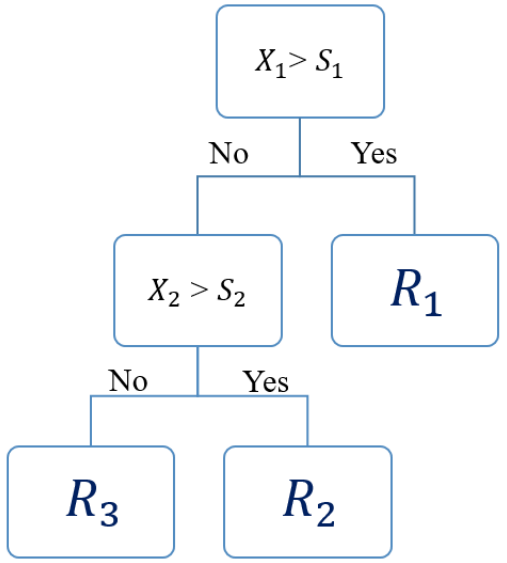
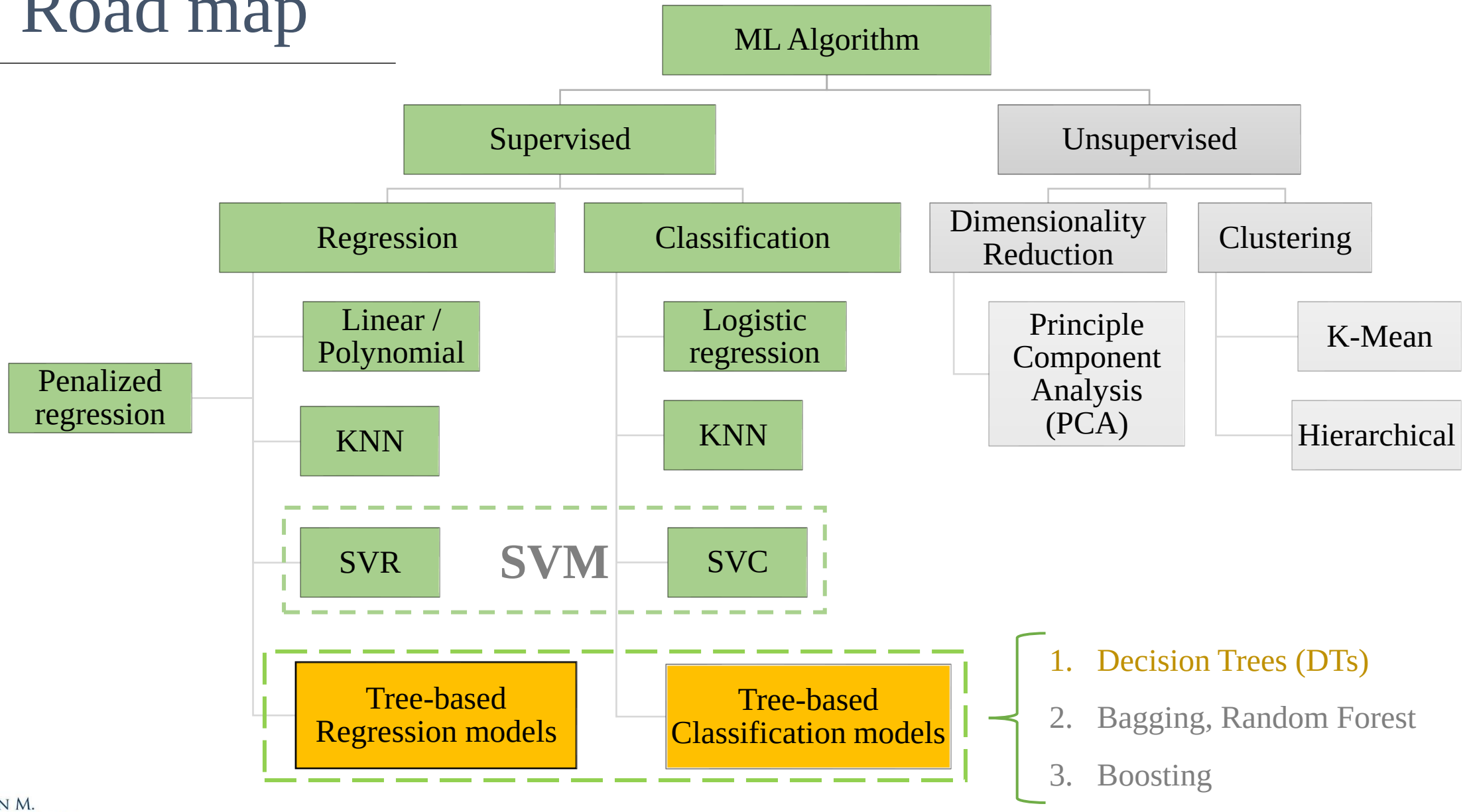
Class -16
Decision Trees (DTs)

Prof. Pedram Jahangiry

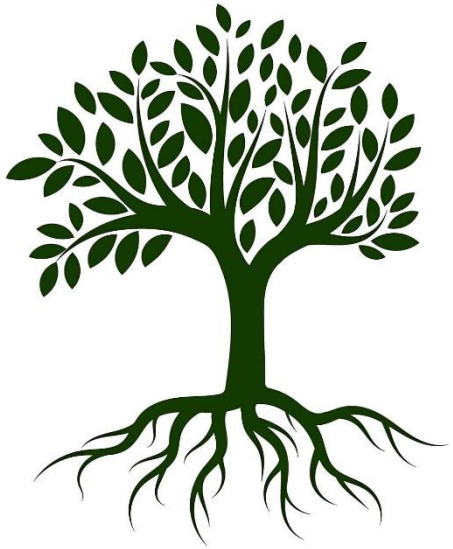




Road map


ML Algorithm
Supervised
Unsupervised
Regression
Classification
Dimensionality Reduction
Clustering
Linear / Polynomial
Logistic regression
Principle Component Analysis (PCA)
K-Mean
Penalized regression
KNN
KNN
Hierarchical
SVR
SVM
SVC
Tree-based Regression models
Tree-based Classification models
1. Decision Trees (DTs)
2. Bagging, Random Forest
3. Boosting

Topics



Part I

1. Decision Trees Definitions
2. Decision Trees criteria
   - MSE
   - Error Rate
   - Gini Index
   - Entropy

Part II

1. Regression Trees
2. Classification Trees

Part III

1. Pruning a tree
2. Hyperparameters

Part IV

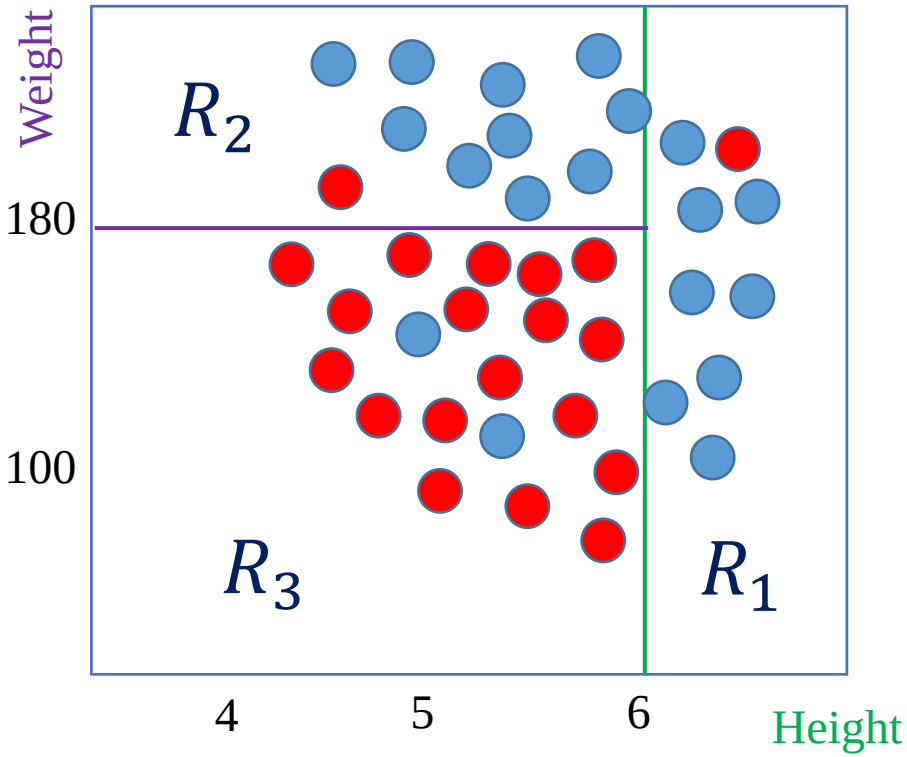
1. Pros and Cons
2. Applications in Finance

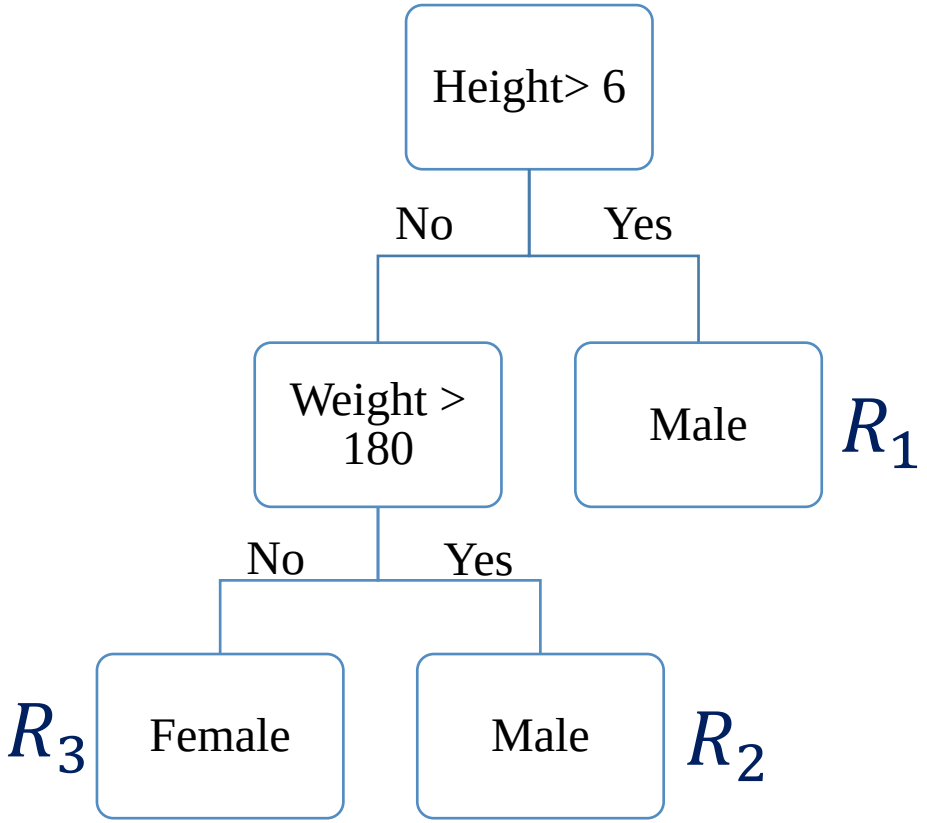
Part I

Decision Trees definitions and criteria

Decision Trees Definitions

- DTs are ML algorithms that progressively divide data sets into smaller data groups based on a descriptive feature, until they reach sets that are small enough to be described by some label.
- DTs apply a top-down approach to data, trying to group and label observations that are similar.

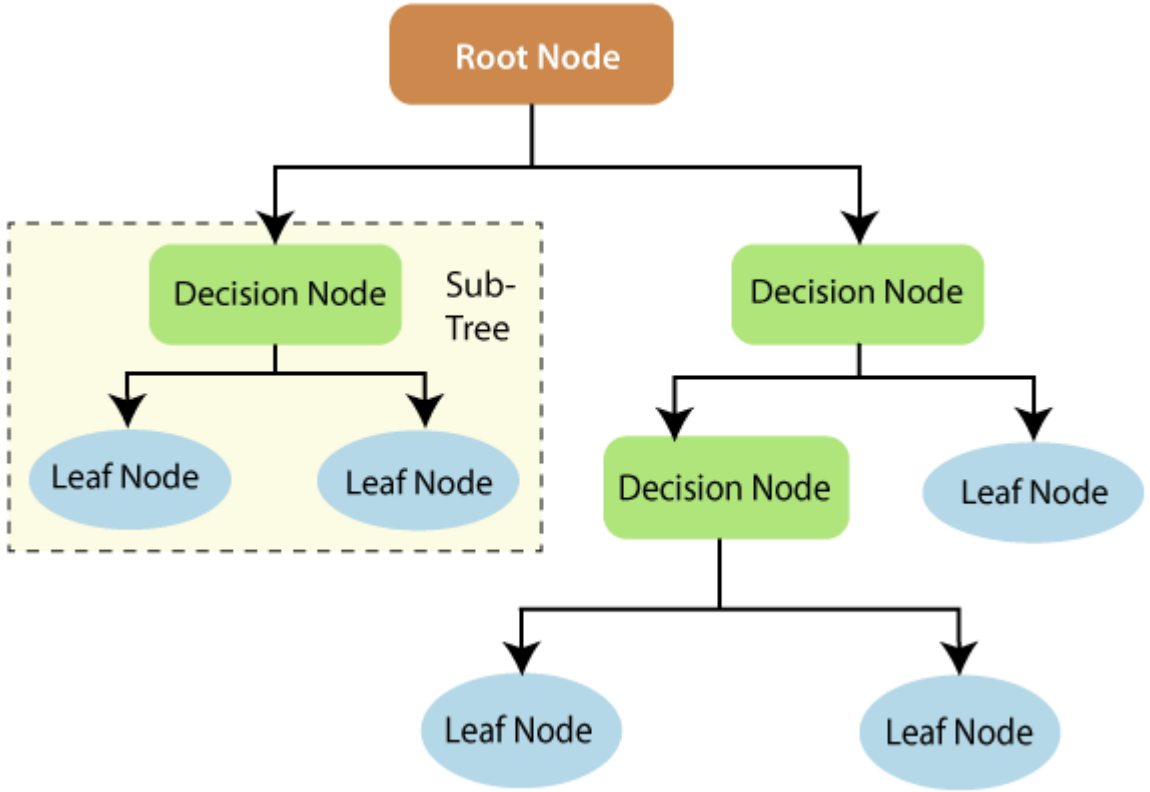




Decision Trees Definitions

- When the target variable consists of real numbers: regression trees
- When the target variable is categorical: classification trees
- Terminology:

  - ✓ Root node
  - ✓ Splitting
  - ✓ Branch
  - ✓ Decision node (internal node)
  - ✓ Leaf node (terminal node)
  - ✓ Sub-tree
  - ✓ Depth (level)
  - ✓ Pruning

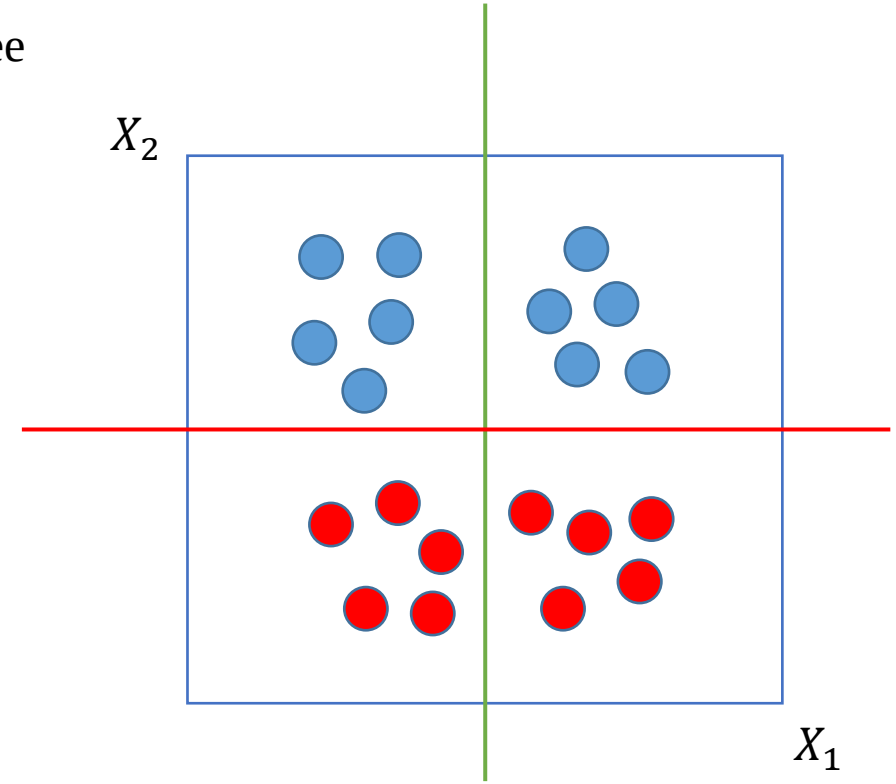


Decision Trees Criteria

- Which split adds the most information gain (minimum impurity)?
- Regression trees: MSE
- Classification trees:
  1. Error rate
  2. Entropy
  3. Gini Index

Control how a Decision Tree decides to split the data

They all measure **impurity**

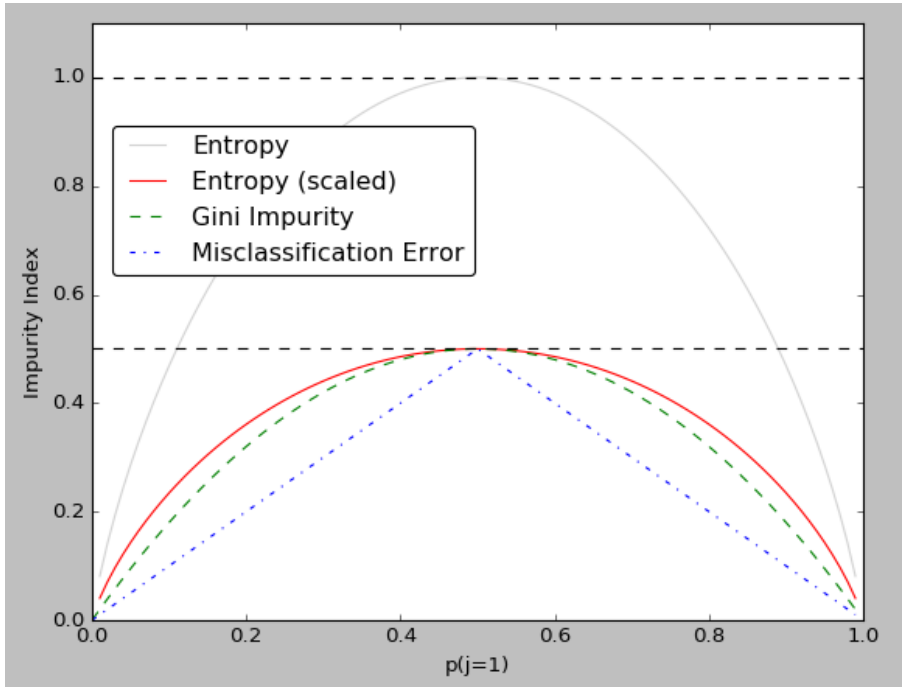


Decision Trees Criteria

- Entropy: Measures the impurity or randomness (uncertainty) in the data points
- Gini Index: Measure how often a randomly chosen element would be incorrectly labeled
- For both Entropy and Gini, 0 expresses all the elements belong to a specified class (pure)
- Different decision tree algorithms utilize different impurity metrics

$$entropy = -\sum_j p_j \log_2(p_j)$$

$$Gini = 1 - \sum_j p_j^2$$



Part II

Regression / Classification Trees!

How does a decision tree work?

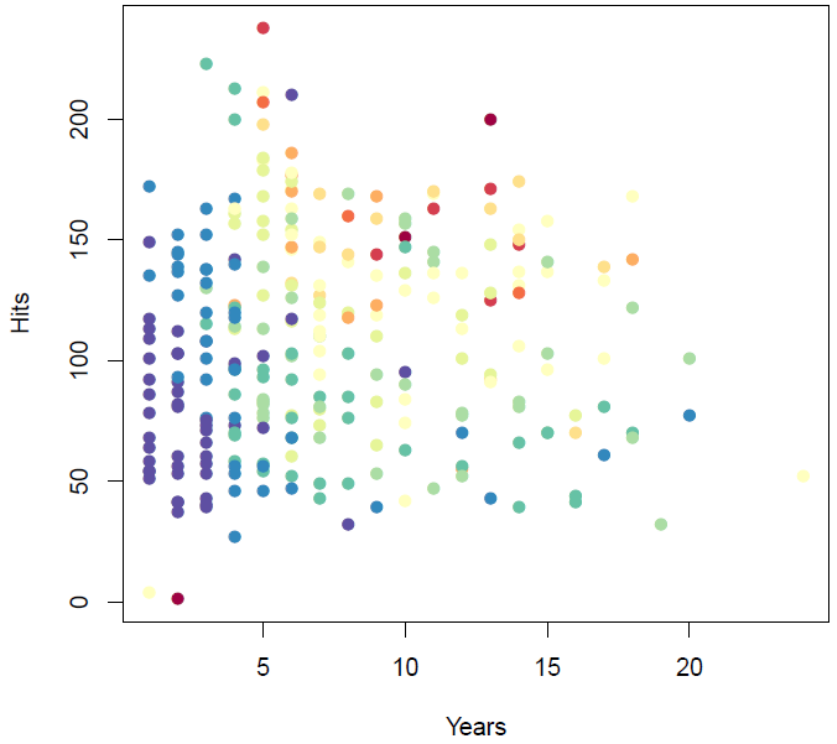
Regression Trees

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

- DTs apply a top-down approach to data, trying to group and label observations that are similar.

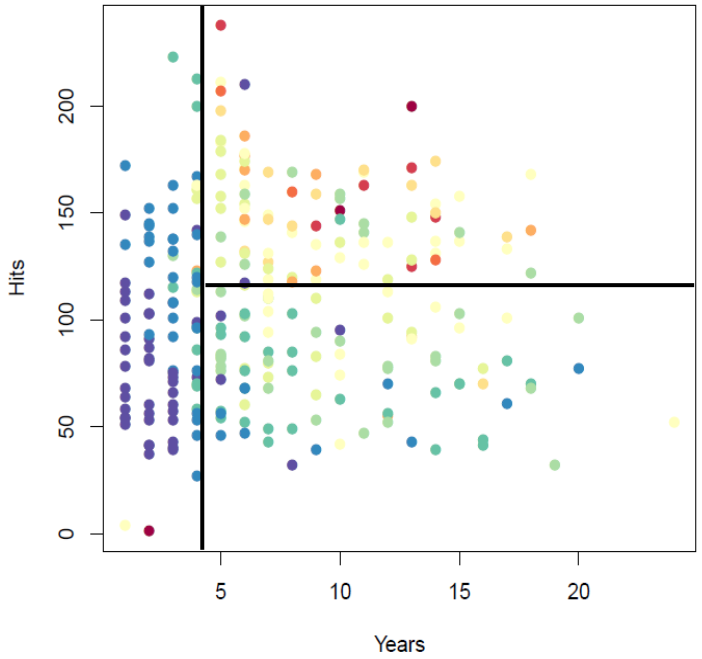
- The main questions in every decision-making process:

1. Which feature to start with?
2. Where to put the split (cut off)?



Interpreting the results

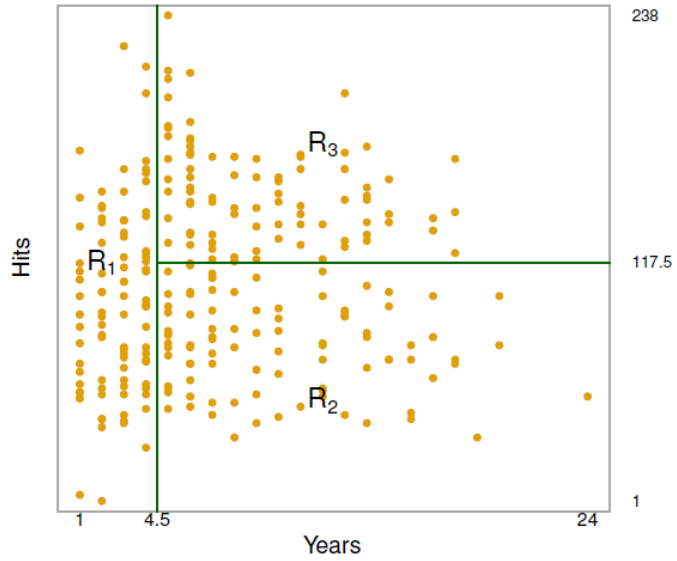
- Based on color-coded salary, it seems that years is the most important factor in determining salary.
- For less experienced players, the number of hits seems irrelevant.
- Among more experienced players thought, players with more hits tend to have higher salaries.
- As one can see, the model is very easy to display, interpret and explain.



Tree building process

- Divide the feature space into J distinct and non-overlapping regions.
- For every observation that falls into the region R_j, we make the same prediction, which is simply the mean of the target values for the training observations in R_j.
- The goal is to find rectangles $R_1, R_2, \ldots, R_j$ that minimize the RSS:

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



- Where $\hat{y}_{R_j}$ is the mean target for the training observations within the j^{th} rectangle.

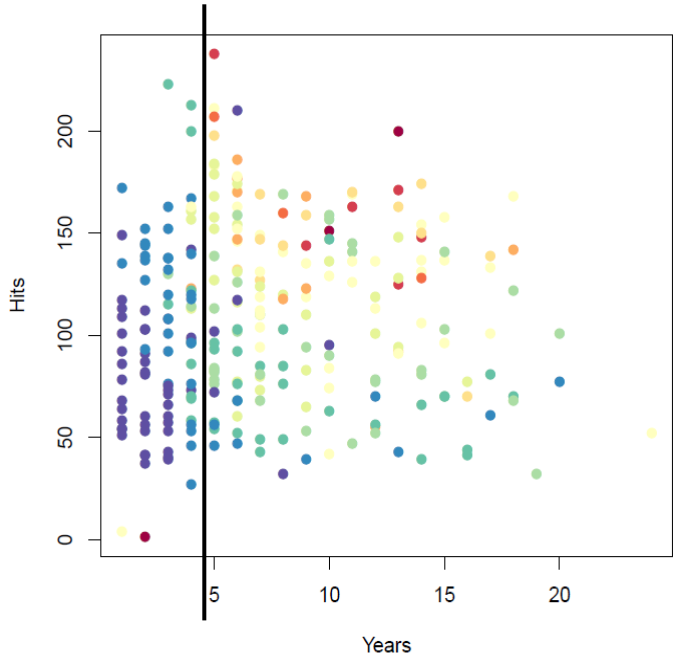
Tree building process: Recursive Binary Splitting

- How does the algorithm select X_j and the split s ?
- X_j and s are selected such that splitting the feature space into the regions $\{X|X_j < s\}$ and $\{X|X_j \geq s\}$ leads to the largest possible reduction in RSS.

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

- Seeking for the value of j and s that minimized the following equation:

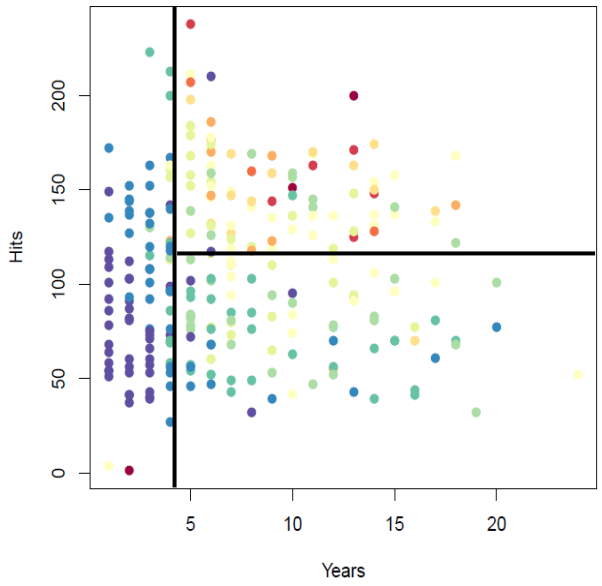
$$\sum_{i:\, x_i \in R_1(j,s)} (y_i - \hat{y}_{R_1})^2 + \sum_{i:\, x_i \in R_2(j,s)} (y_i - \hat{y}_{R_2})^2$$

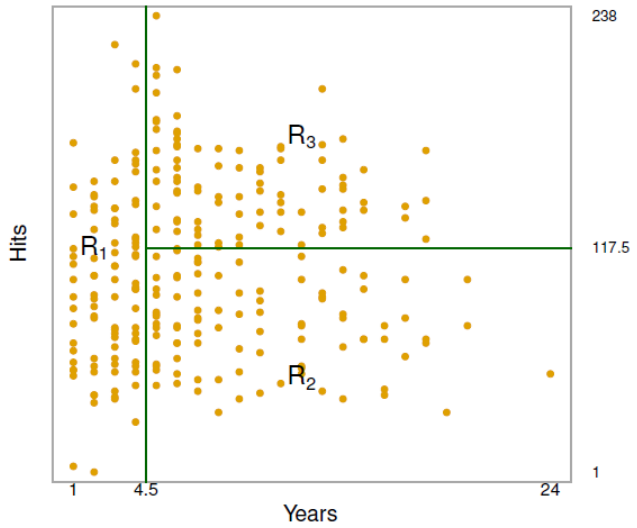


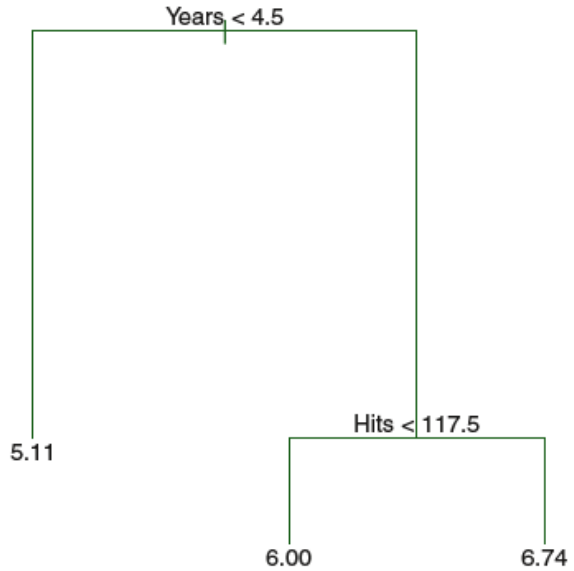
- The best split is made at that particular step, rather than looking ahead and picking a split that will lead to a better tree in some future step.

Tree building process: Recursive Binary Splitting

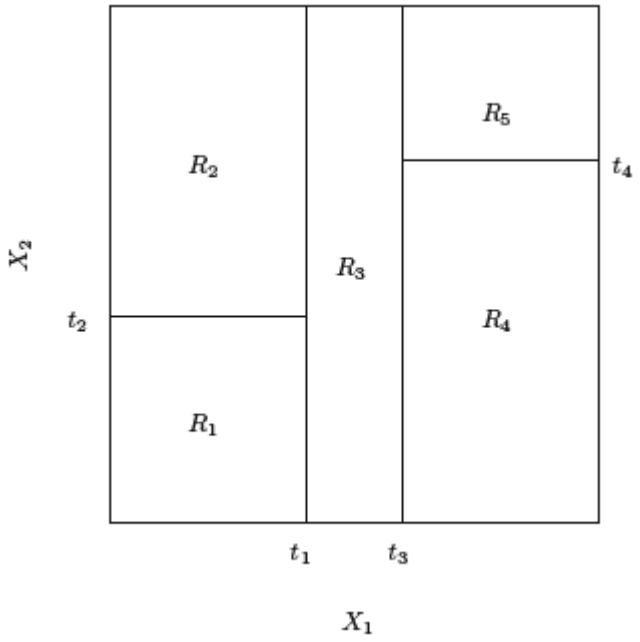
- Next, the algorithm repeats the process, looking for the best feature and best split in order to split the data further to minimize the RSS within each of the resulting regions.
- The process continues until a stopping criterion is reached; for instance, continues until no region contains more than a fixed number of observations.



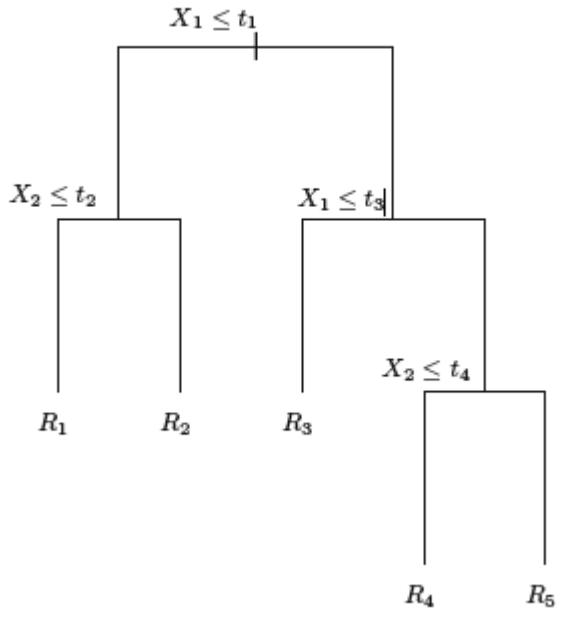




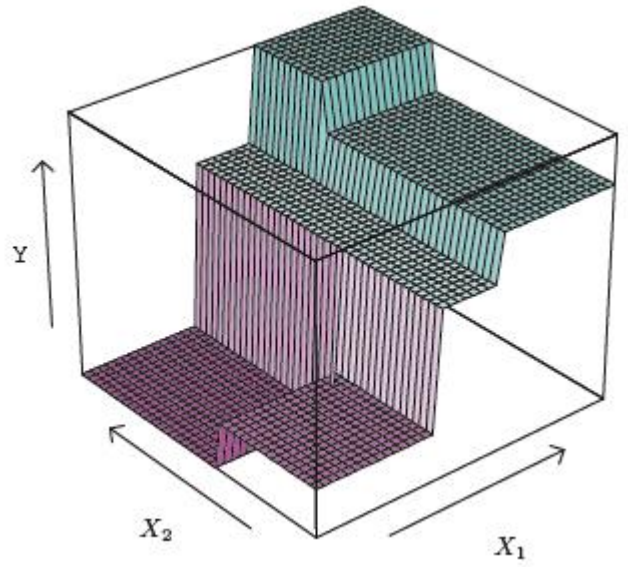
A Five-Region Example of Recursive Binary Splitting



The output of recursive binary splitting on a two-dimensional example

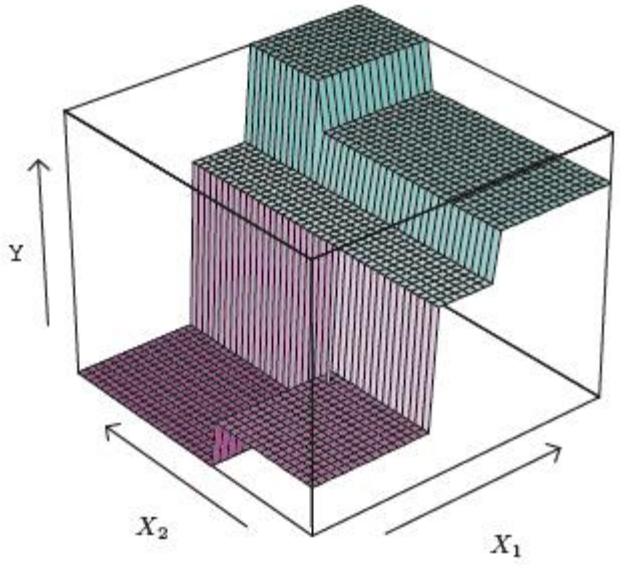


A tree corresponding to the partition in the left panel.



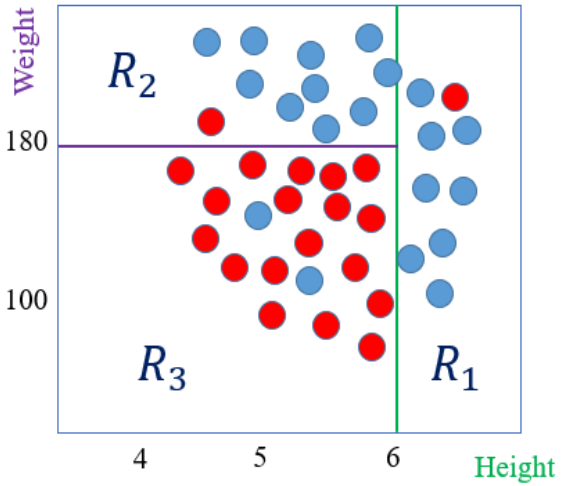
A perspective plot of the prediction surface corresponding to that tree.

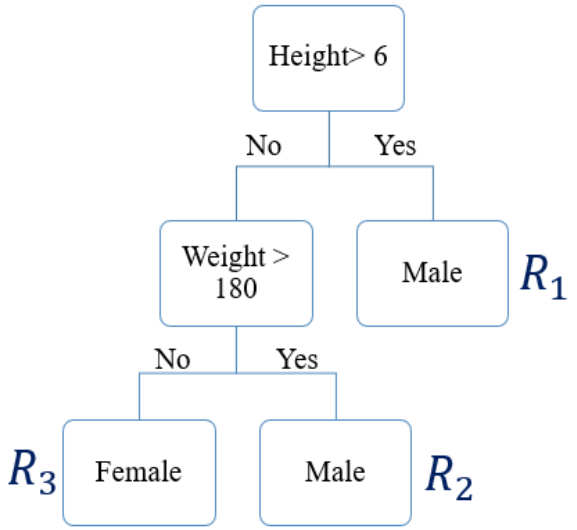
Overfitting?



Classification Trees

- Classification trees are very similar to regression trees, except that it is used to predict a qualitative response rather thana quantitative one.
- The prediction of the algorithm at each terminal node will be the category with the majority of data points i.e., the most commonly occurring class.

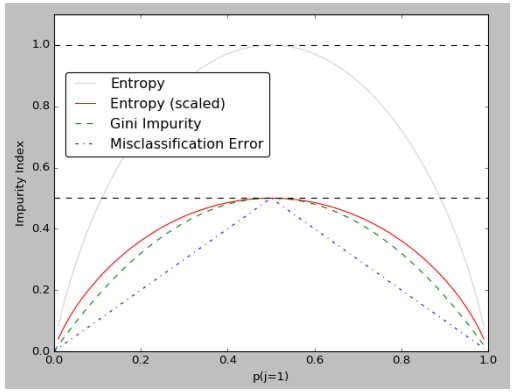




Classification Trees (details)

- Just as in the regression setting, the recursive binary splitting is used to grow a classification tree. However, instead of RSS we will be using one of the following impurity criteria:

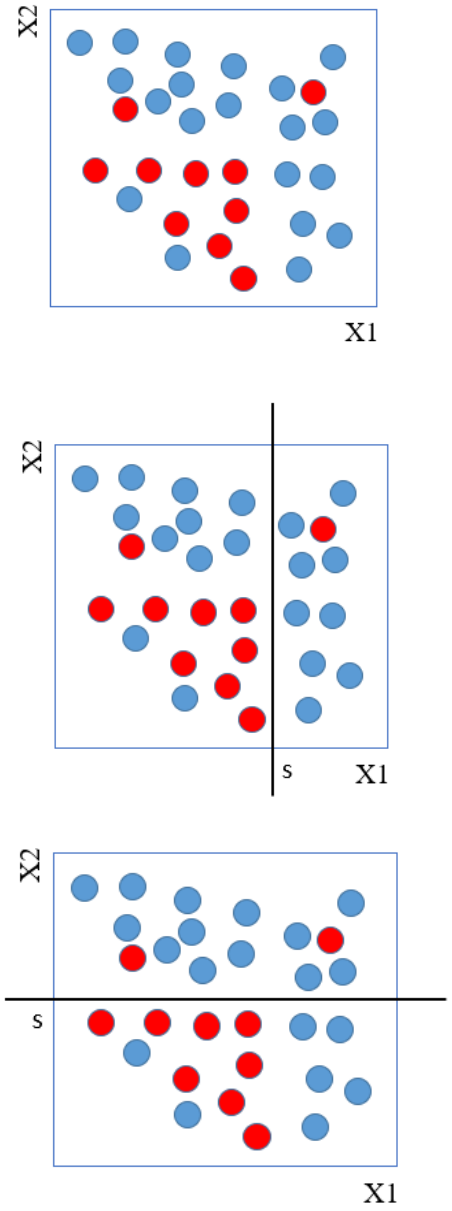
1. Classification error rate: $E = 1 - \max_k(\hat{p}_{mk})$
2. Gini index: $G = \sum_{k=1}^{K} \hat{p}_{mk}(1 - \hat{p}_{mk})$
3. Cross entropy: $D = -\sum_{k=1} \hat{p}_{mk} \log \hat{p}_{mk}$



- $\hat{p}_{mk}$ represents the proportion of training observations in the m^{th} region from the k^{th} class.
- Classification error rate is not sufficiently sensitive to node purity and in practice either Gini or Cross entropy is preferred.

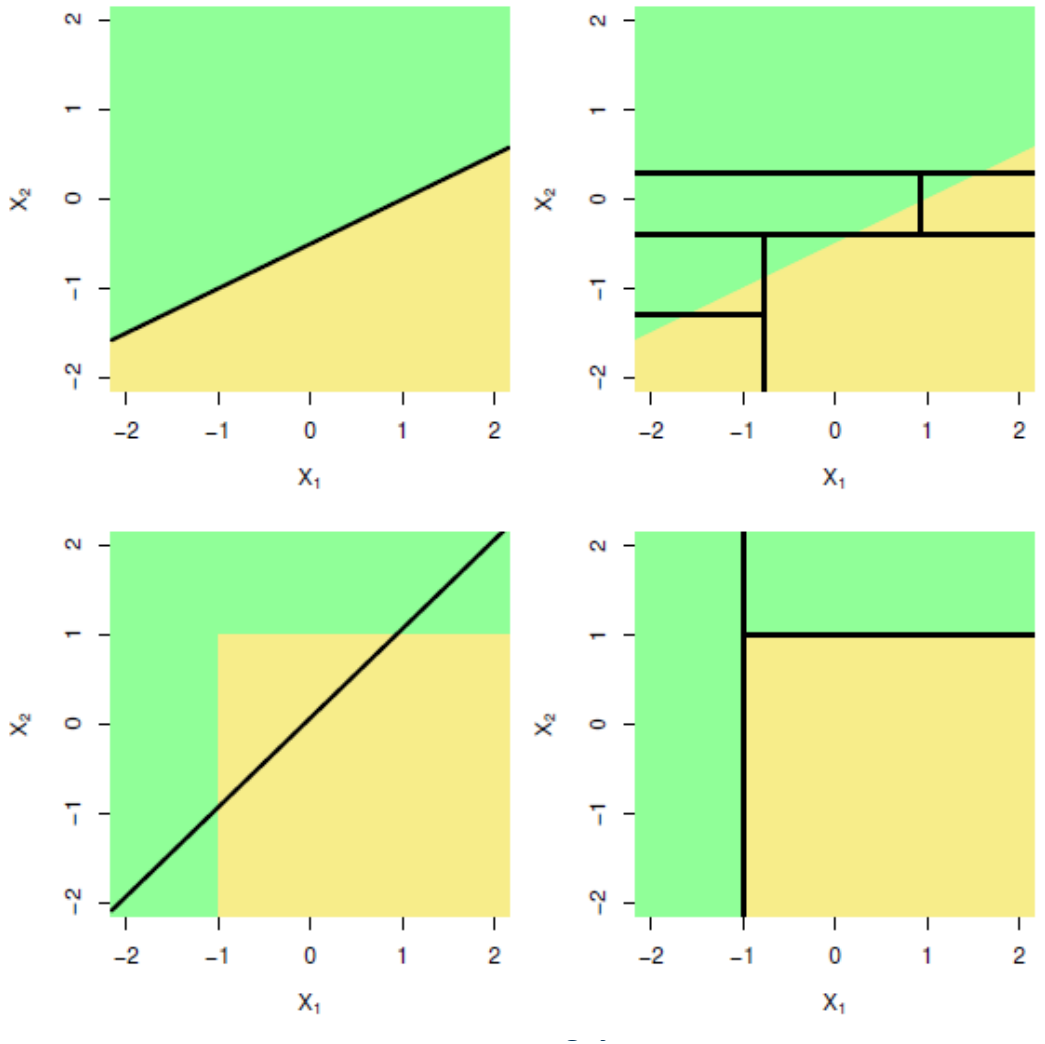
Decision Tree Metrics (Simple Example)

Node	*Gini* $1 - \sum_j p_j^2$	*Cross entropy* $-\sum_j p_j \log(p_i)$	*Error rate* $1 - \max(p_i)$
Entire training data before split	$\left\{1 - \left(\left(\frac{10}{30}\right)^2 + \left(\frac{20}{30}\right)^2\right)\right\} = 0.44$	$-\left\{\left(\frac{10}{30} log \frac{10}{30} + \frac{20}{30} log \frac{20}{30}\right)\right\} = 0.64$	$1 - \max\left\{\frac{10}{30}, \frac{20}{30}\right\} = 1 - \frac{20}{30} = 0.333$



Trees Versus Linear Models

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



Top Row: True linear boundary
Bottom row: true non-linear boundary.

Part III

Pruning a tree

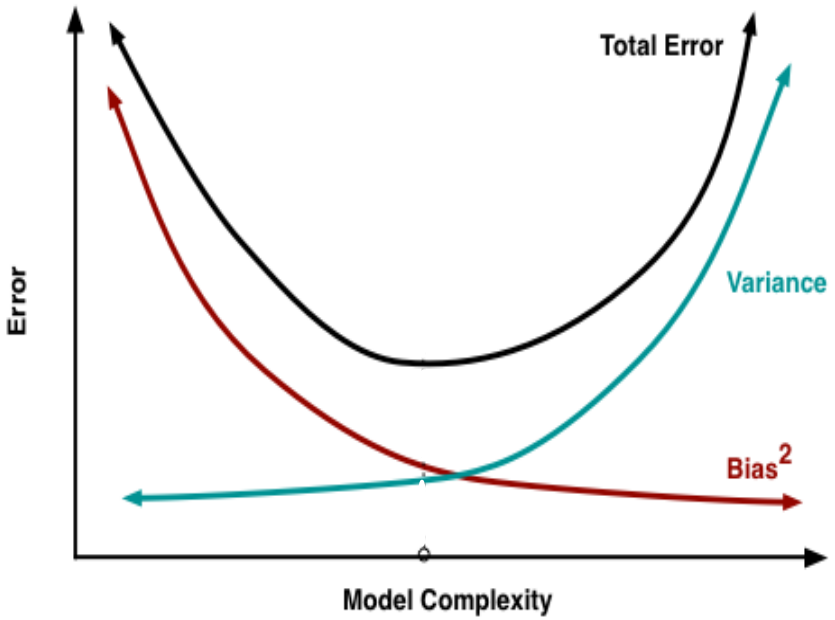
Tunning hyper parameters

Pruning a tree

- A smaller tree with fewer splits may lead to lower variance and better interpretation (but) at the cost of higher bias.
- Smaller trees are too short-sighted:

" a seemingly worthless split early on in the tree might be followed by a very good split, a split that leads to a large reduction in RSS/impurity index later on"

- A better strategy is to grow a very large tree, this may produce good predictions on the training set, but is likely to overfit the data, leading to poor test set performance.
- So, we need to prune it back in order to obtain a subtree.
- Cost complexity pruning is used to do this.



Cost complexity pruning (weakest link pruning)

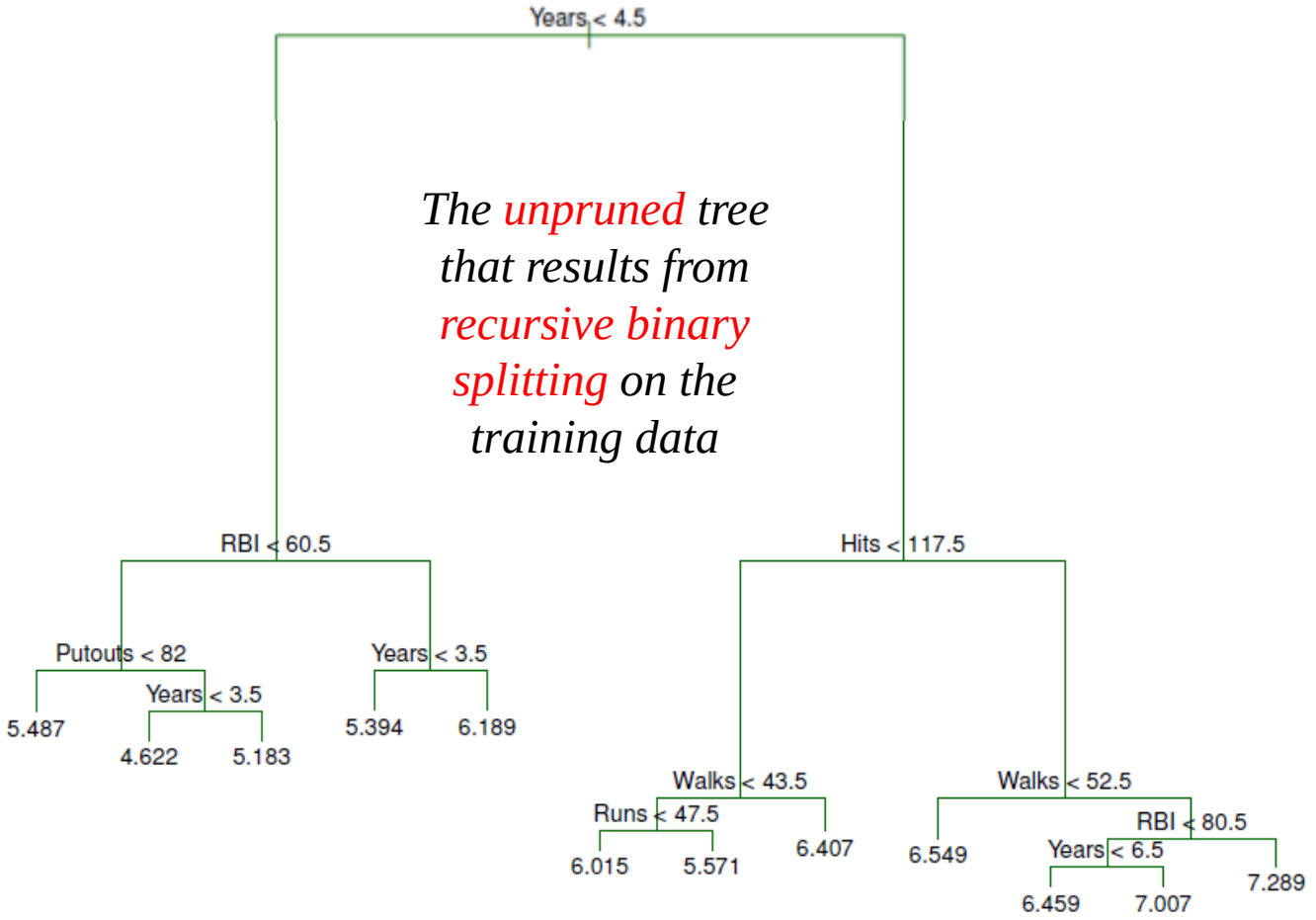
- Consider a sequence of trees indexed by a nonnegative tuning parameter α.
- For each value of α there corresponds a subtree $T \subset T_0$ such that the following objective function is minimized.

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

  - $|T|$ indicates the number of terminal nodes of the tree T
  - R_m is the rectangle corresponding to m^{th} terminal node and
  - $\hat{y}_{Rm}$ is the mean of the training observations in R_m

- α controls the bias variance trade off and is determined by cross validation.
- Lastly, we return to full data set and obtain the subtree corresponding to α

Salary example continued



The unpruned tree that results from recursive binary splitting on the training data

Finding the optimal α or T

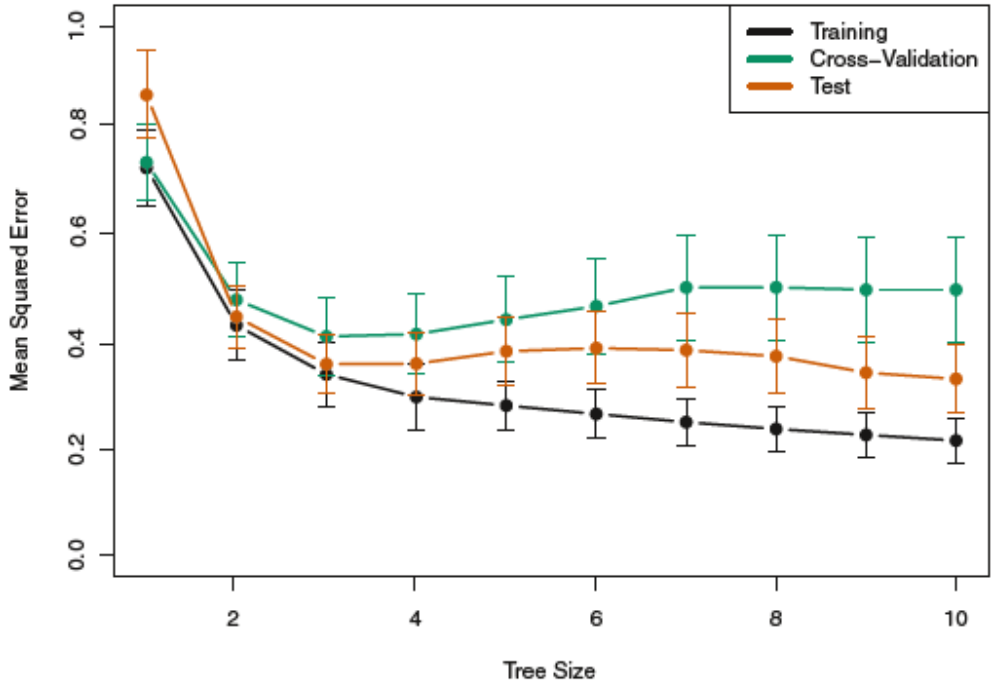
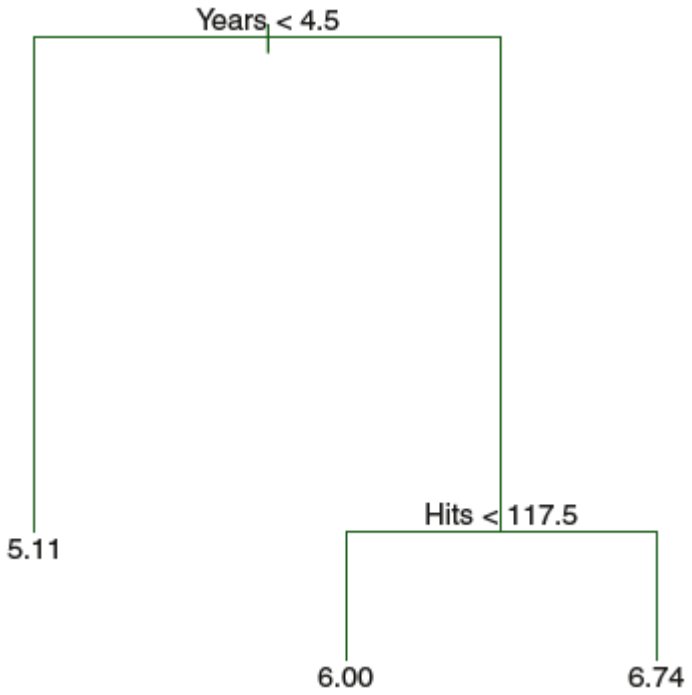
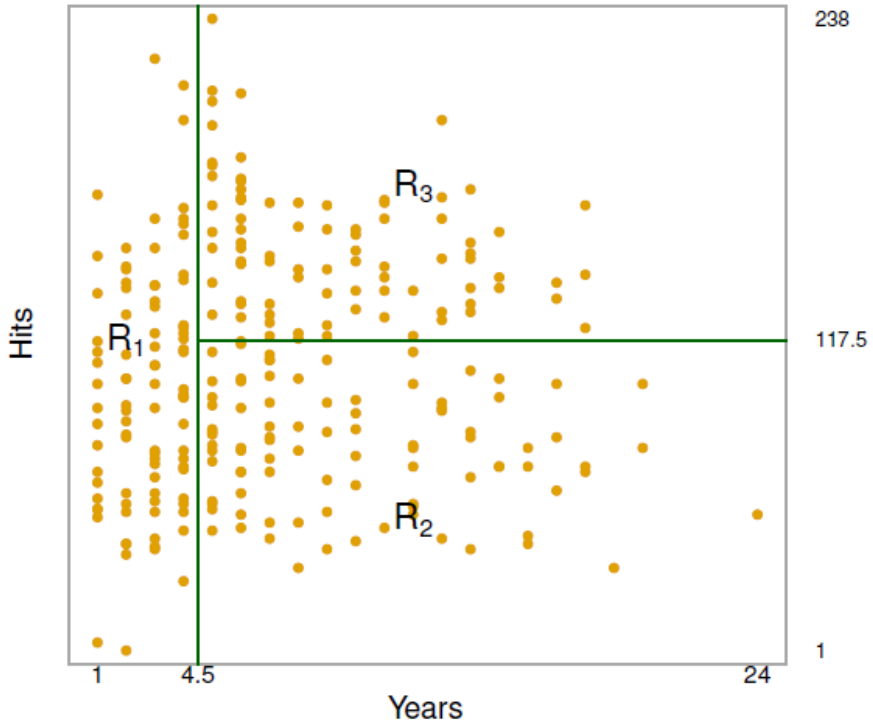


FIGURE 8.5. *Regression tree analysis for the* `Hitters` *data. The training, cross-validation, and test MSE are shown as a function of the number of terminal nodes in the pruned tree. Standard error bands are displayed. The minimum cross-validation error occurs at a tree size of three.*

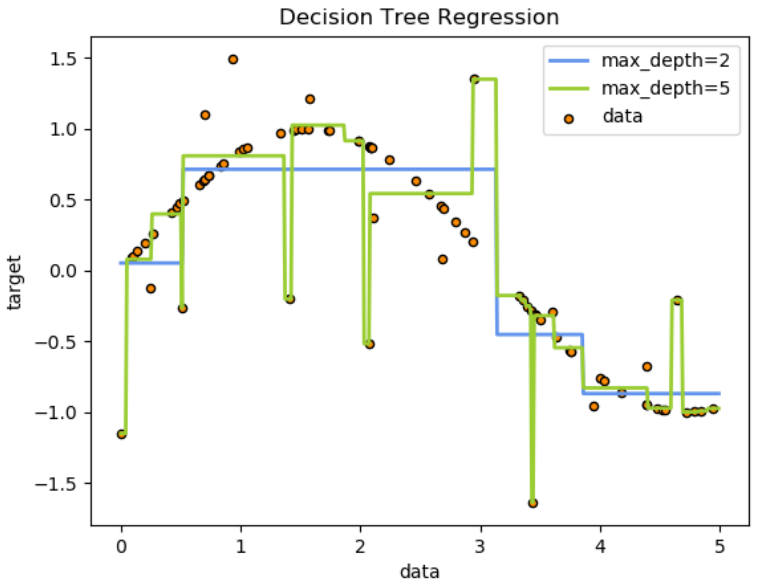
The optimal (pruned) tree





Other hyperparameters

- ✓ To avoid overfitting, regularization parameters can be added to the model such as:
  - Maximum depth of the tree
  - Minimum population at a node
  - Maximum number of decision nodes
  - Minimum impurity decrease (info gain)
  - Alpha (complexity parameter)
- ✓ Other hyperparameters are:
  - Criterion: gini, entropy
  - Splitter: best, random
  - Class weight: balanced, none



Part IV

Pros and Cons

Applications in finance

DTs' Pros and Cons

Pros:

- Easy to interpret and visualize
- Can easily handle categorical data without the need to create dummy variables
- Can easily capture Non-linear patterns
- Can handle data in its raw form (no preprocessing needed). Why?
- Has no assumptions about distribution because of the non-parametric nature of the algorithm

Cons:

- Poor level of predictive accuracy.
- Sensitive to noisy data. It can overfit noisy data. Small variations in data can result in the different decision tree*.

*This can be reduced by bagging and boosting algorithms.

DTs' Applications in finance

- Enhancing detection of fraud in financial statements,
- Generating consistent decision processes in equity and fixed-income selection
- Simplifying communication of investment strategies to clients.
- Portfolio allocation problems.

Parameter	Description	Default	Options
max_depth	The maximum number of levels: split the nodes until max_depth has been reached. All leaves are pure or contain fewer samples than min_samples_split.	None	int
max_features	Number of features to consider for a split.	None	None: all features int: # features float: fraction auto, sqrt: sqrt(n_features) log2: log2(n_features)
max_leaf_nodes	Split nodes until creating this many leaves.	None	None: unlimited int
min_impurity_decrease	Split node if impurity decreases by at least this value.	0	float
min_samples_leaf	A split will only be considered if there are at least min_samples_leaf training samples in each of the left and right branches.	1	int; float (as a percent of *N*)
min_samples_split	The minimum number of samples required to split an internal node.	2	int; float (percent of *N*)
min_weight_fraction_leaf	The minimum weighted fraction of the sum total of all sample weights needed at a leaf node. Samples have equal weight unless sample_weight is provided in the fit method.	0	

Appendix A

Scikit-Learn decision tree parameters.


class
DISMISSED