University of Pittsburgh

ECE 2195: Special Topics – Computers Machine Learning

Decision Trees & Ensemble Methods

Mai Abdelhakim, PhD

Assistant Professor of ECE

Swanson School of Engineering

University of Pittsburgh

maia@pitt.edu

Decision Trees

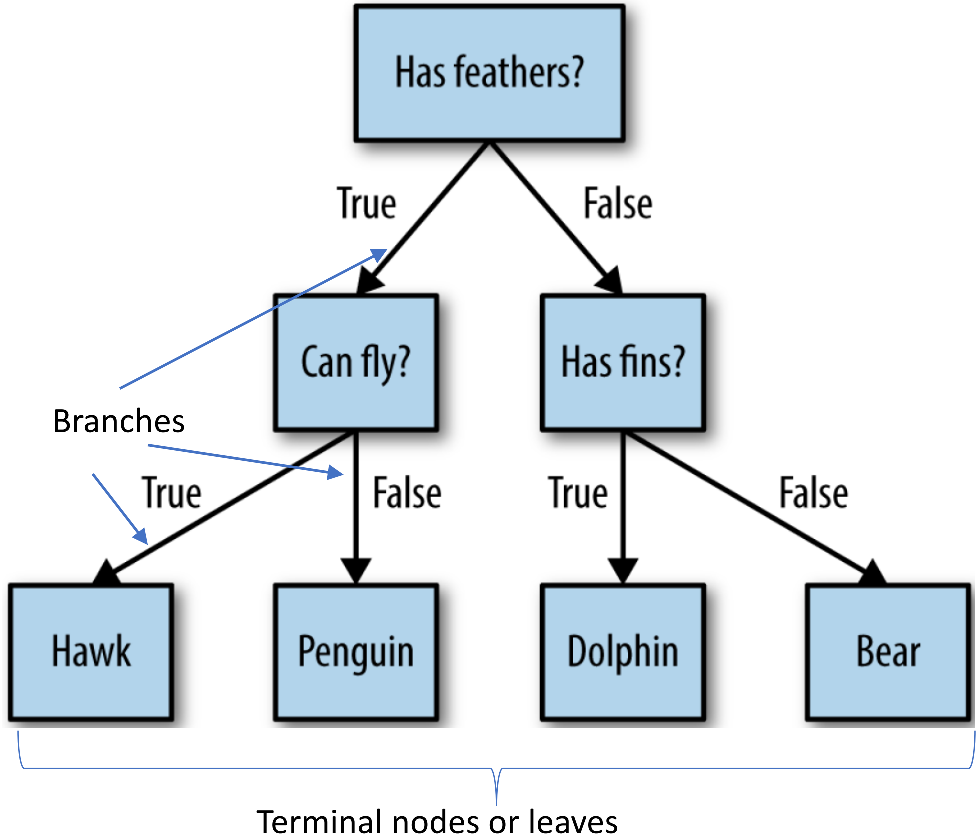
- For regression & classification

- Rely on **segmenting the feature space** into number of **regions** using set of **splitting rules**

- Rules are represented in a **tree**.

- **Tree also has**
  - Nodes:
    - **First rule at root**
    - **Terminal** nodes represent regions in feature space
    - **Internal** nodes are points of splits
  - Branches: segment that connect nodes

- **Tree depth** is maximum number of branches (from tree root) until a terminal node



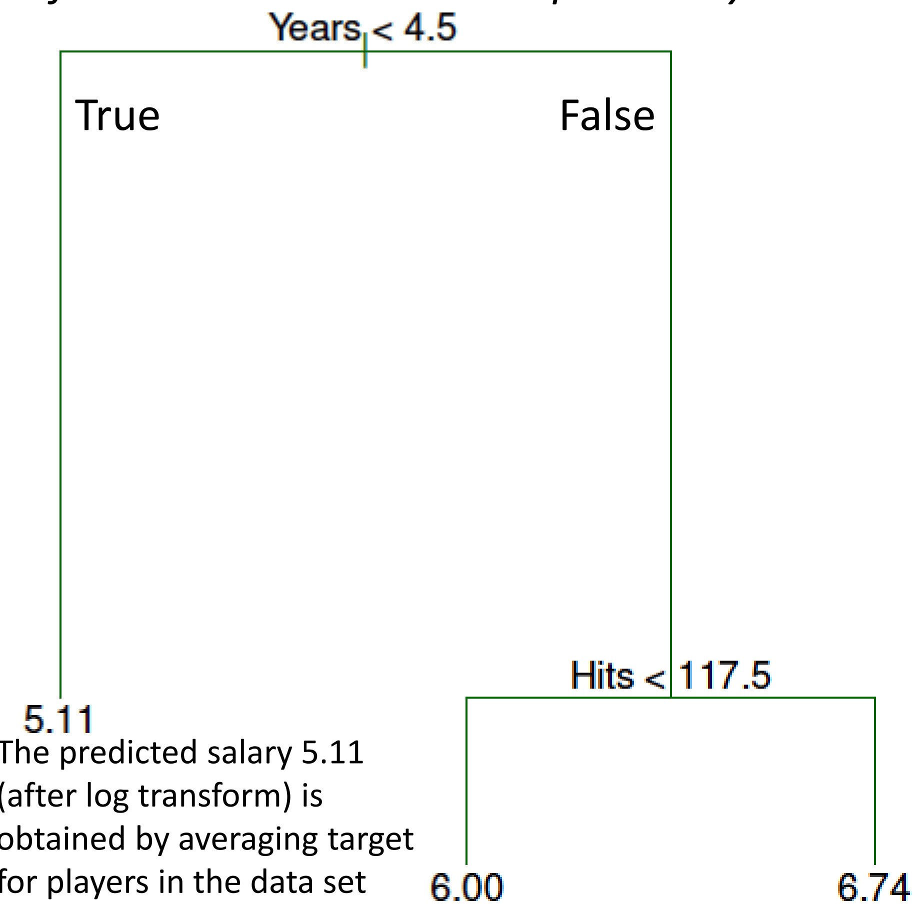
Regression Trees Example: Baseball Salary

Predicting the log salary (in $1000) of a baseball player, based on the number of years that he has played in the major leagues and the number of hits that he made in the previous year

Example: predict a baseball player log salary (in thousands of dollars) using two features

- Features:
  - **Years**: number of years the player is in a major league
  - **Hits**: number of hits the player made in the previous year

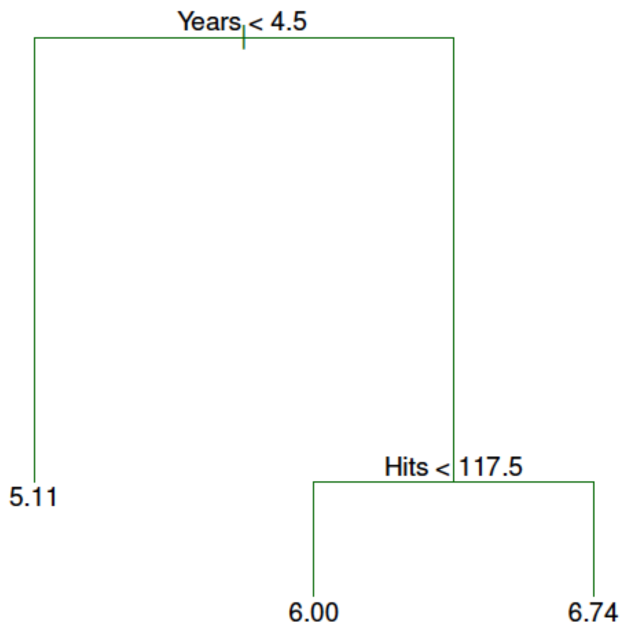
- Tree consists of series of spitting rules
  - Start with checking if Years < 4.5
  - If Years<4.5, the tree checks Hits

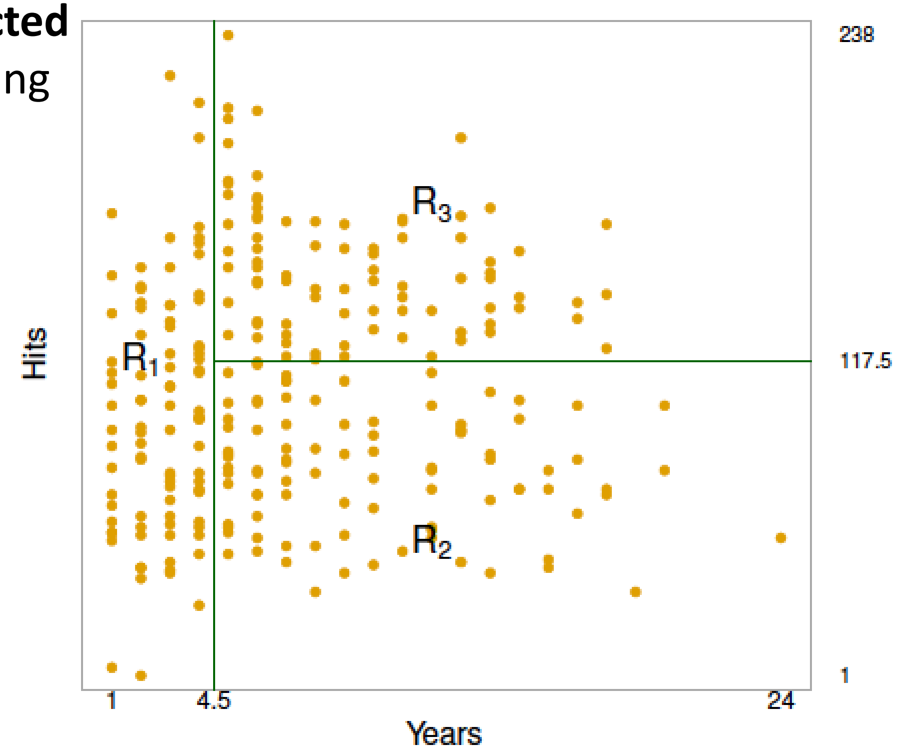


The predicted salary 5.11 (after log transform) is obtained by averaging target for players in the data set with Years <4.5

Regression Trees Example: Baseball Salary
The tree divides the feature space!

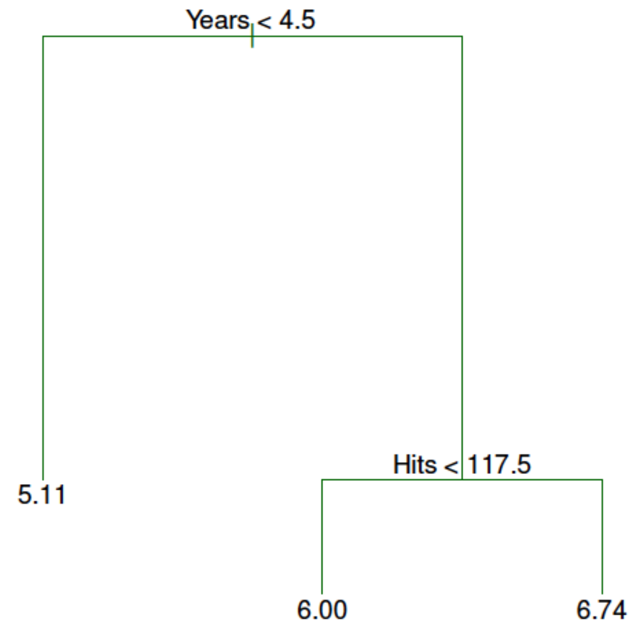
- The tree divides the feature space into regions
- If observation lies in region R_i, then **predicted response is the average response** of training observation in **that region**



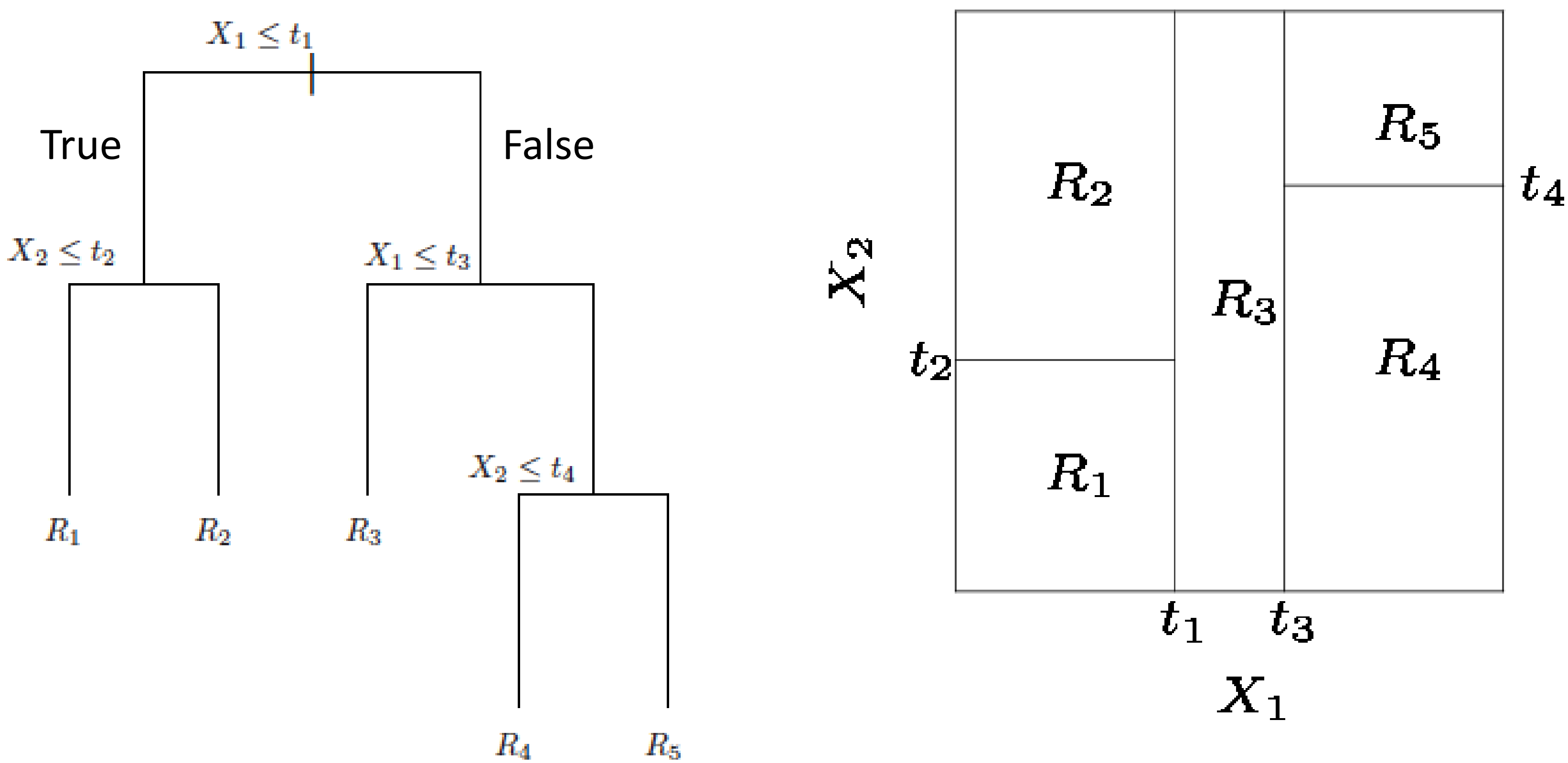


Tree Interpretation

- In this tree example, the following interpretations can be made
  - Years is the most important feature -- Appears at root of the tree
    - Player with less experience, gets lower salary
  - If a player is less experienced, Hits will not be a significant feature in determining the salary
  - If player is experienced, then "Hits" plays a key role in determining the salary
    - More hits, higher salary



Tree Divides the Feature Space into Rectangular Regions



Building a Regression Tree

- The goal is to divide the feature space into ***J* regions** $(R_1, R_2 \ldots . R_J)$, and find these region that minimizes the RSS
- The RSS is given by:

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

  - $\hat{y}_{R_j}$ is the mean response of training observation in region j
  - y_i is the response of the ith training example
- **Greedy approach**: try every possible set of partitions
  ➔ **Infeasible**
- **Instead,** we use **recursive binary splitting**
  - **Find one split at a time**

Recursive Binary Splitting

- Start with entire space, and iterate:

1. Each iteration select **predictor X_j and cutpoint s** such that splitting the space into regions with s leads to the **largest reduction in the RSS**

Region 1 : $X_j < s$

Region 2: $X_j \geq s$

$$R_1(j,s) = \{X|X_j < s\}$$

$$R_2(j,s) = \{X|X_j \geq s\}$$

   - Focus on **one split at a time** (instead of finding all splits at the same time like the greedy approach). Then find a best feature along with its splitting rule that minimizes RSS.

- Find feature X_j and s that minimize

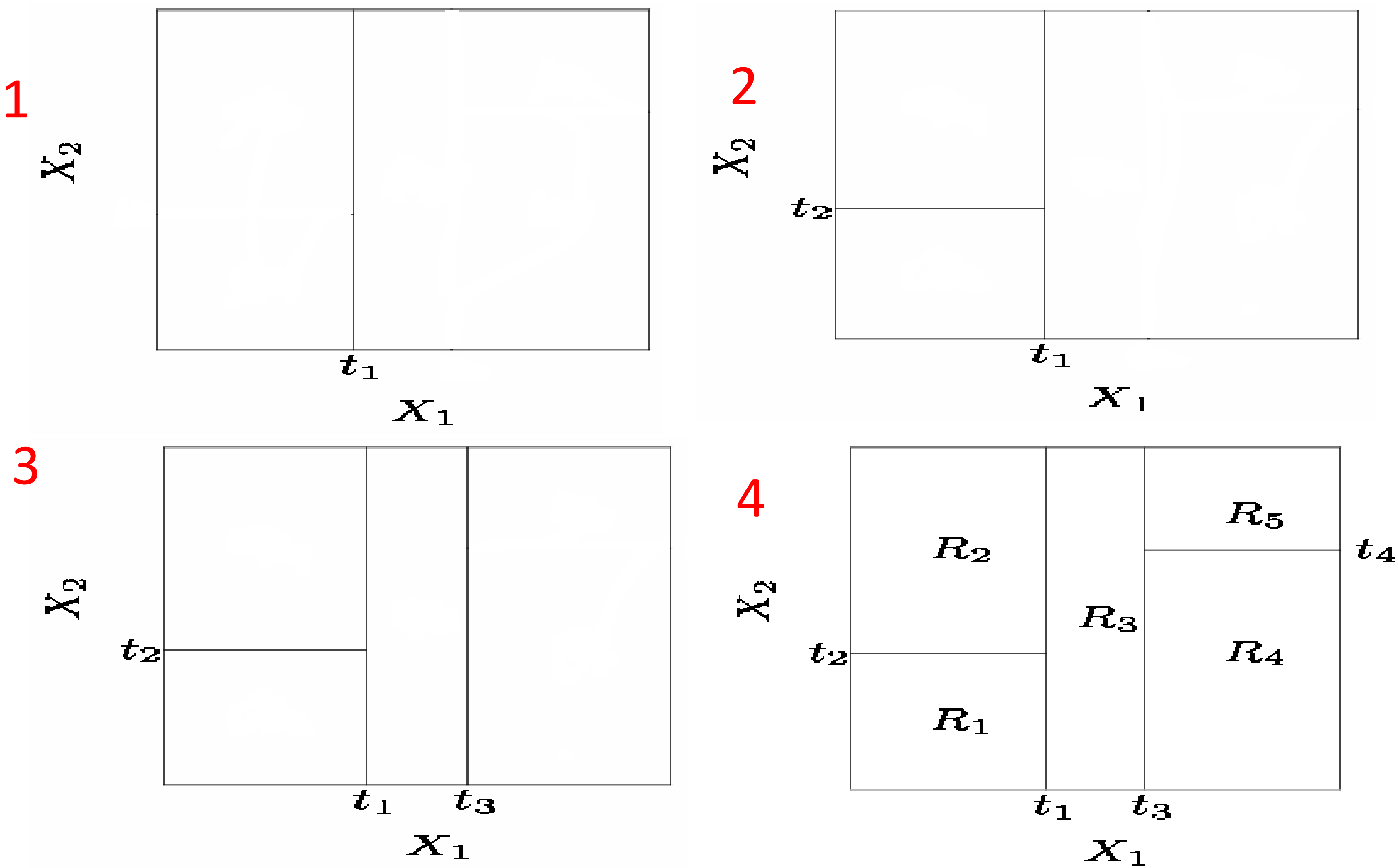
$$\sum_{i:\ x_i \in R_1(j,s)} (y_i - \hat{y}_{R_1})^2 + \sum_{i:\ x_i \in R_2(j,s)} (y_i - \hat{y}_{R_2})^2$$

2. **Repeat** the process with redefined regions: find the feature and cutpoint to split one of the regions already obtained (not entire space)

  - We choose the one with largest reduction in RSS

- In other words: in each iteration, we **split a region into two region**. In the next iteration, we focus on one of the regions then split it into two, and so on

- We repeat until a stopping criterion


1
X_2
t_1
X_1
2
X_2
t_2
t_1
X_1
3
X_2
t_2
t_1
t_3
X_1
4
X_2
R_2
R_5
t_4
R_3
R_4
t_2
R_1
t_1
t_3
X_1

Avoid Overfitting & Underfitting

- We can have **very long trees** ➔ Complex
  - Tree can grow until there is one sample (one one label) per region
- Short trees (fewer splits/regions) are easy to interpret ➔ have low variance, but may have high bias
- We want to achieve a good **bias variance trade-off**
- **Pre-pruning: Set a stopping criteria to prevent overfitting.**
  - Like **limit** on **depth** of tree, or number of **observations per region**, number of leaves (**regions**), threshold on reduction of **RSS**
- **Post-pruning (or just pruning)**: Grow a very long tree using training data, then **prune** it to obtain a subtree
  - Check if you remove a node, how well the pruned tree work on the validation set
  - Get the tree that minimizes average cross validation error

Classification Trees

- Decision trees can be used for classification
- Prediction: assign the observation to the **most commonly occurring class** in the region
- Similar to regression trees, **recursive binary splitting** is used to grow the tree
    - However, instead of using the RSS in regression we use other metrics

Metrics for Growing a Tree

- Let $\hat{p}_{mk}$ be the proportion of training observations in the ***m*th region** that are from the ***k*th class**

- **Classification error rate:**
  - **Training observations in a region that do not belong to the most common class**
    - **Error in region m is E: fraction of observations that do not belong to the common class**

$$E = 1 - \max_{k}(\hat{p}_{mk})$$

- Other metrics are common for growing a tree:
  - These metrics are: **Gini index** and **Entropy (Cross-entropy)**

Metrics for Growing a Tree – Gini Index

- **Gini index (*G*):** is a measure of a **node purity**
  - Nodes are pure when they contain observations that belongs to a single class
  - For region m

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

  - **When nodes are pure, the Gini index value is zero**
    - Example: consider K=2 (two classes), and all observations in region *m* belong to class 1. Then $\hat{p}_{m1} = 1$ and $\hat{p}_{m2} = 0$ ➔ In this case the Gini index G=0
  - **Gini index is maximal if classes are mixed**
    - Example: consider K=2 (two classes), and half observations in region *m* belong to class 1 and the other half belongs to class 2. Then $\hat{p}_{m1} = 0.5$ and $\hat{p}_{m2} = 0.5$➔ In this case the Gini index G=0.5 (maximum)

Metrics for Growing a Tree – Cross-Entropy

- Entropy is a measure of uncertainty
- Similar to the Gini metric, it measures node purity
- Cross entropy is defined as:

(note the log here is base 2)
$\text{Log}_2(p)=\ln(P)/\ln(2)$

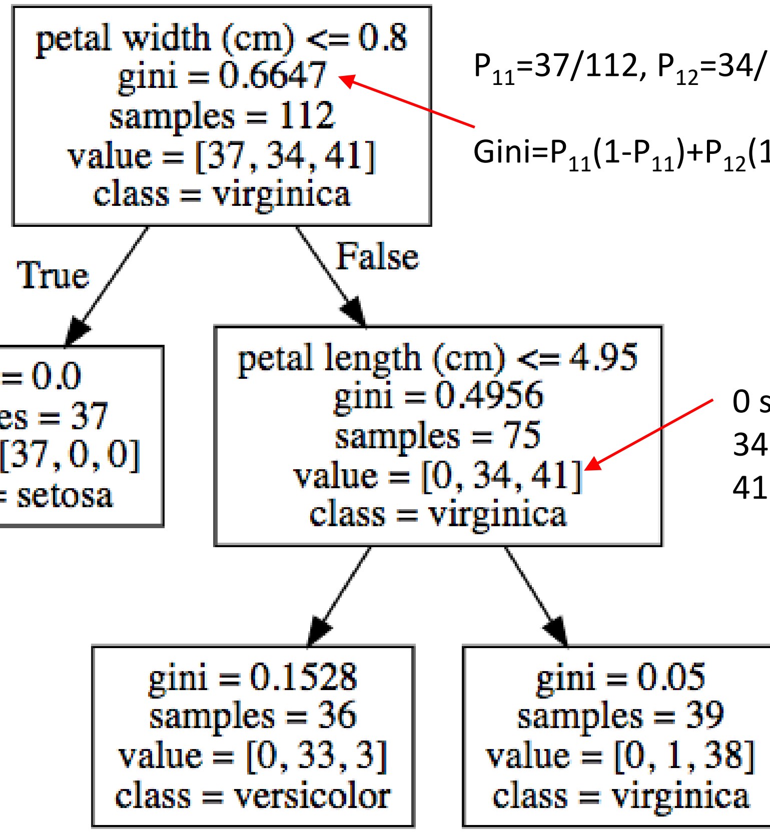
$$D = -\sum_{k=1}^{K} \hat{p}_{mk} \log \hat{p}_{mk}$$

  - D is close to **zero** when nodes are almost **pure**
  - D is maximum if classes are mixed.
- We search for splits the minimizes the metric (Gini or cross-entropy) in a recursive way as we did before

Example: Iris Dataset

P_{11}=37/112, P_{12}=34/112, P_{13}=41/112

Gini=$P_{11}(1-P_{11})+P_{12}(1-P_{12})+P_{13}(1-P_{13})$=0.6647

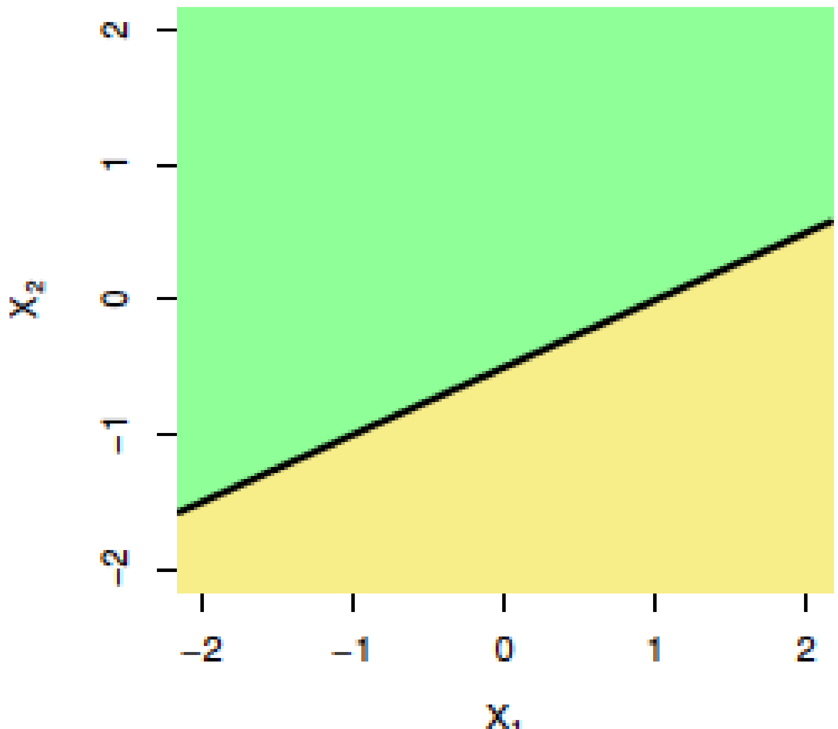


Gini index is 0 since all samples belong to a single class (setosa)

0 samples from class 1 (Setosa) 34 from class 2 (Versicolor), and 41 from class 3 (Virginica)

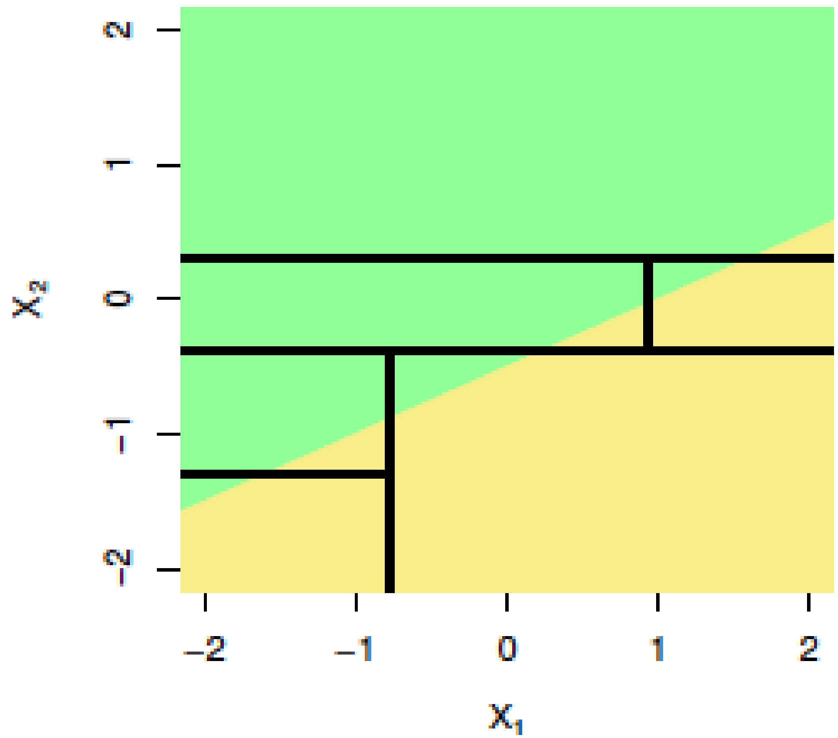
Example Tree vs Linear Model Boundaries

Linear model



Linear model is better in this example

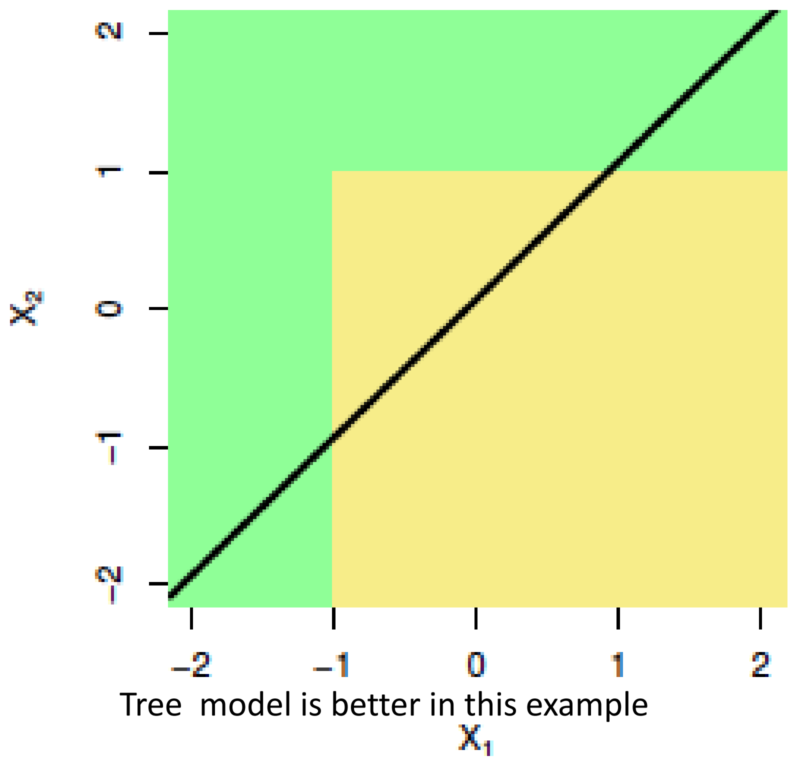
Tree model



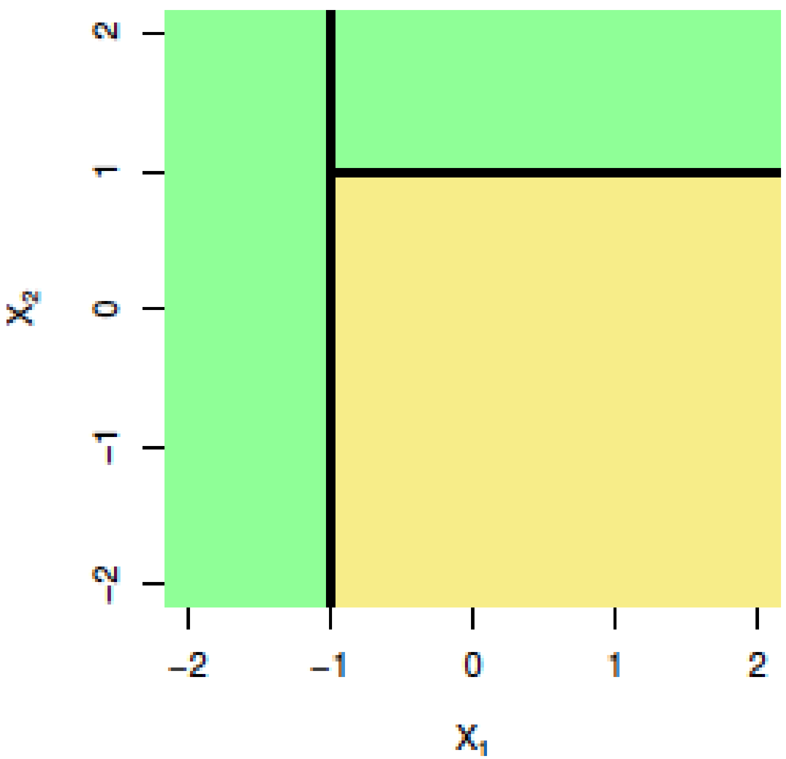
Tree splits the space into rectangular regions

Example Tree vs Linear Model Boundaries

Linear model



Tree model



Tree model is better in this example

Python

- Decision trees: For classification
-
  from sklearn.tree import **DecisionTreeClassifier**
  treeModel=DecisionTreeClassifier(max_depth=m, criterion='gini')
    - max_depth is depth of the tree
    - Default criterion is 'gini' (so no need to write it)

    - http://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeClassifier.html

- Decision trees: For regression, **DecisionTreeRegressor** is used
    - http://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeRegressor.html#sklearn.tree.DecisionTreeRegressor

Decision Trees: Advantages and Disadvantages

- **Very useful for interpretation,** but may **not have the same level of accuracy** compared to other approaches

- Can handle mixed types of features

- Low accuracy would be from overfitting the training data
  - If you have other training samples, maybe splitting rules are different

- Can be very accurate when we **combine multiple** trees together:
  - Helps in avoiding overfiting
  - Combining trees improves the **accuracy**, but becomes harder to **interpret**
  - Examples of approaches that do combining that are **random forest**, **boosting**