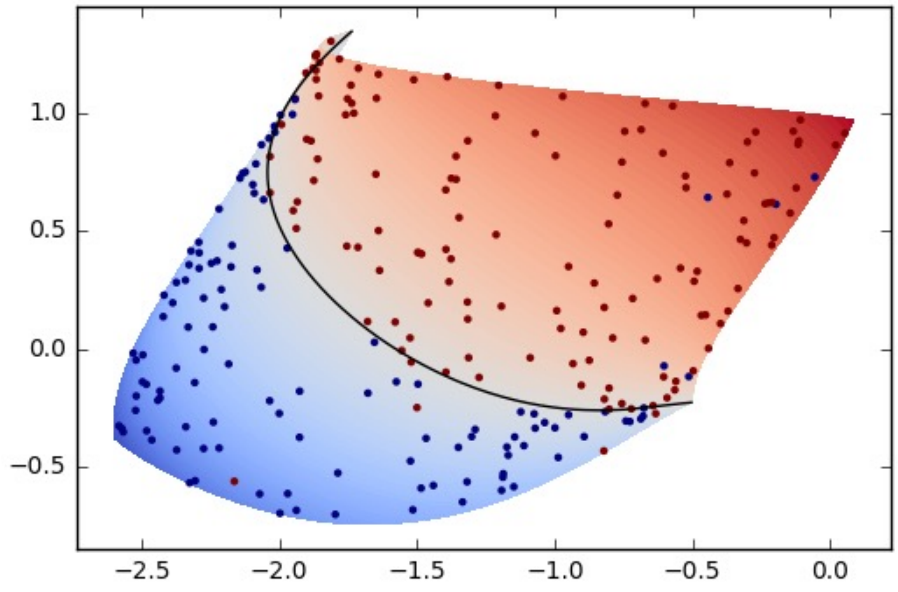
Lecture 19: Decision Trees 2



CS178 Spring 2023

Gavin Kerrigan

Some slides adapted from Padhraic Smyth, Alex Ihler

Announcements/Reminders

- HW4 released later today
  - Due in 2 weeks (Friday 5/26)

- Project team formation due in this Friday (5/19)
  - 103/162 students still not in groups
  - Worth 10% of your project grade
  - Start working on project soon (due 6/12)

Recap of Decision Trees

Example of Learning a Decision Tree

Decision Trees in Code

A Decision Tree Classifier

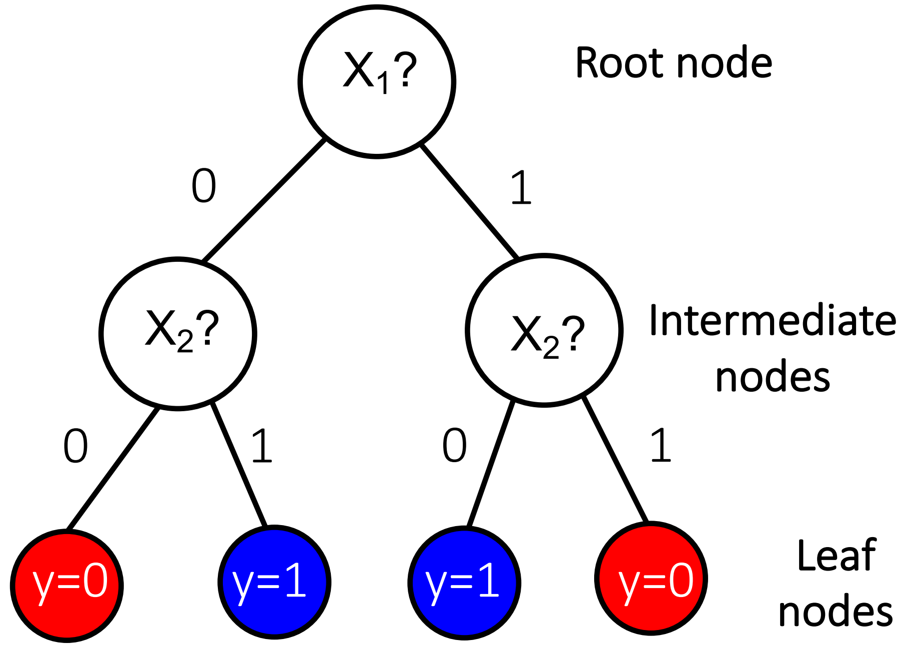
A simple binary classification problem

XOR Dataset

x_1	x_2	y
0	0	0
0	1	1
1	0	1
1	1	0

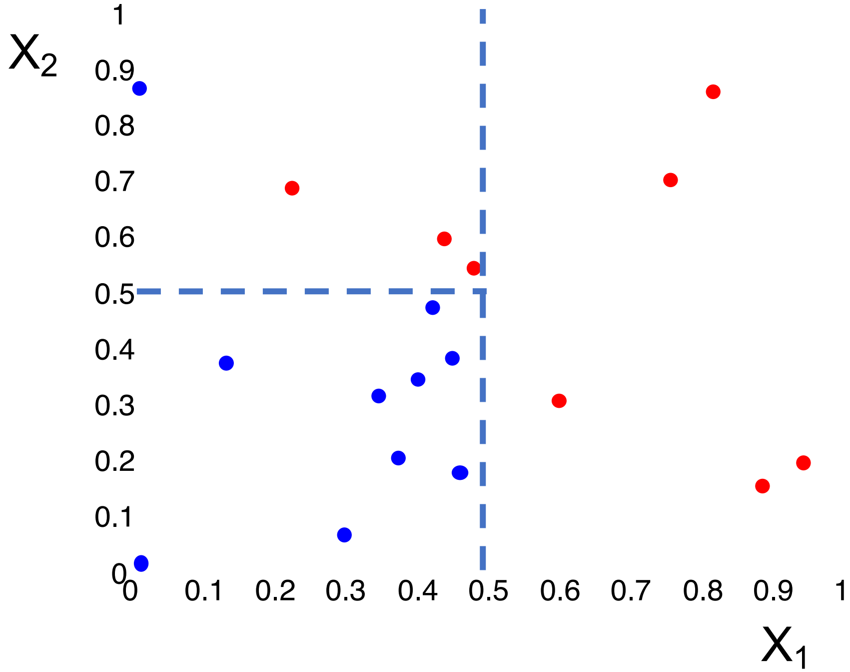
$y = XOR(x_1, x_2)$

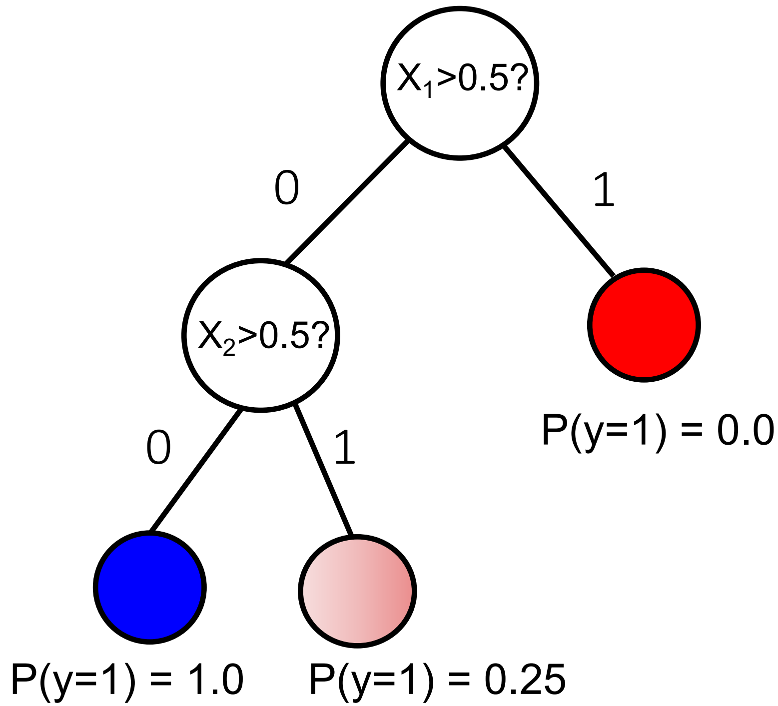
Can represent a Boolean function as a decision tree



The y prediction at each leaf node depends on the path to that node

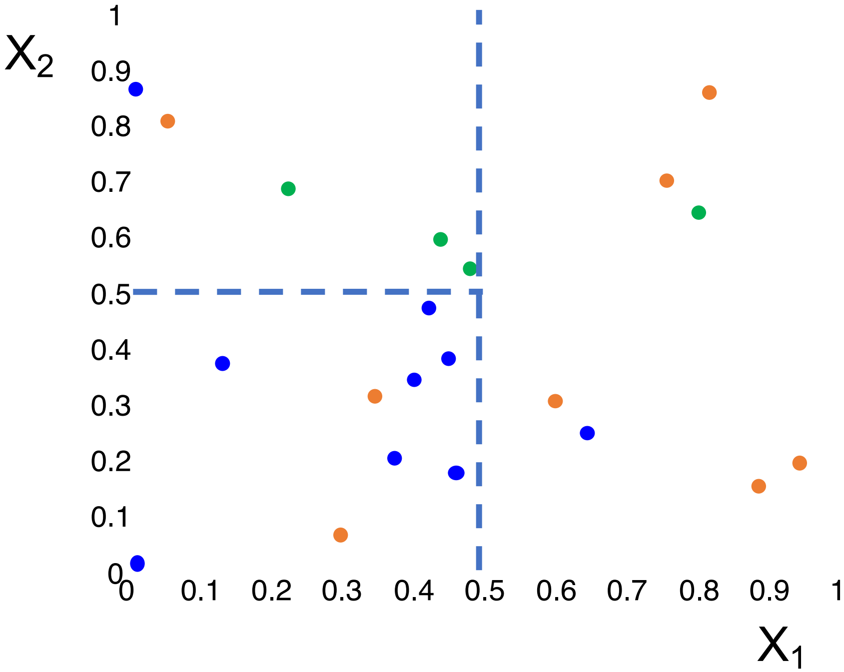
Decision Tree Model in 2 Dimensions

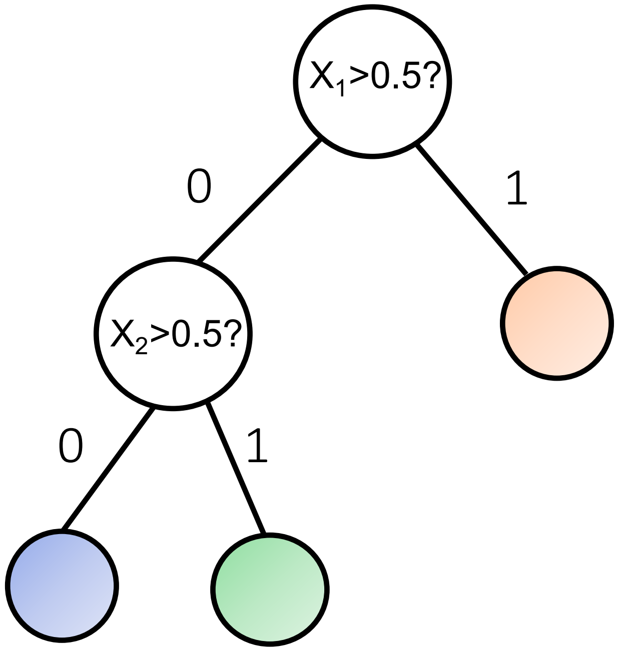




More generally, can predict P(y|path) at any leaf (or node)

Generalizing to K>2 Classes





At each leaf node:
vector of probabilities = relative frequencies of class labels
e.g., top left region: P(y=green) = 0.6, P(y=blue) = 0.2, ...

Learning Decision Trees

- Construct (or “grow”) trees in a top-down fashion

- Should a node be a leaf node?
  - If so: what should we predict?
  - If not: how should we further split the data?

- Leaf nodes: compute class probabilities, pick majority class

- Non-leaf nodes: pick a feature and a split
  - Greedy: “score” all possible features and splits
  - Score function measures uncertainty about labels after split
    - How much easier is our prediction task after we divide the data?

Algorithm BuildTree: Greedy training of a decision tree classifier

Input: Labeled dataset D = { (x_i, y_i) }, i = 1,.. n
Output: A decision tree with parameters θ

Create a new node

Compute P = ClassProbabilityVector(D)

if LeafCondition(D, P, node) **then**
    node = leaf node   % declare node to be a leaf node

else
    t_j = FindBestSplit(D)   % threshold value for best split, for some feature x_j
    $D_L = \{ (x_i, y_i) : x_{ij} \leq t \}$
    $D_R = \{ (x_i, y_i) : x_{ij} > t \}$
    Set left and right children to trees from BuildTree(D_L) and BuildTree(D_R)

end if

ClassProbabilityVector?

The class probability vector at a node
= relative frequencies of the class labels at that node

Example with 4 classes:

12 datapoints at a node, with:

Num(class 1) = 6 => P(class 1 | node) = 6/12 = 0.5

Num(class 2) = 4 => P(class 2 | node) = 4/12 = 0.333

Num(class 3) = 2 => P(class 3 | node) = 2/12 = 0.167

Num(class 4) = 0 => P(class 4 | node) = 0/12 = 0

LeafCondition?

Typically declare a node to be a leaf if any of the following are true

1. All labels at the node belong to the same class
   (no point in splitting any further)

2. The node is at the maxDepth for the tree
   (controls complexity, can prevent the tree from getting too large)

3. The number of examples at a leaf node < n_{min}
   (if we split any further we may be fitting noise and overfit)

Hyperparameters: maxDepth, n_{min}

Gini Index

The Gini index is a measure of the variance of a set of class labels

Definition:
Given a set of probabilities of class labels p_1, p_2, ... p_K

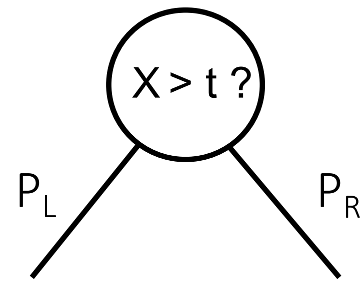
Gini index = $\Sigma_k\, (p_k)\, (1 - p_k)$

$= \Sigma_k\, (p_k)\; - \Sigma_k\, (p_k\, p_k)$

$= 1 - \Sigma_k\, (p_k)^2$

Intuition: if one p_k=1 and all others = 0, then Gini index = 1 – 1 = 0,
i.e., we have zero uncertainty about the class labels

Using the Gini Index at Nodes



Fraction of datapoints that go left and right

D_L = left subset of data

Class probabilities in D_L

G_L = Gini index of D_L

D_R = right subset of data

Class probabilities in D_R

G_R = Gini index of D_R

Gini(t) = weighted average = $P_L\, G_L + P_R\, G_R$

We want the feature x and split t that minimizes Gini(t)

FindBestSplit?

Given the subset of data D we are considering at the current node:

Compute the Gini index of every possible split of D

- i.e. for every feature, and every threshold, compute the Gini index of splitting on that (feature, threshold) pair

Best split is whichever (feature, threshold) tuple results in the lowest Gini index

Questions?

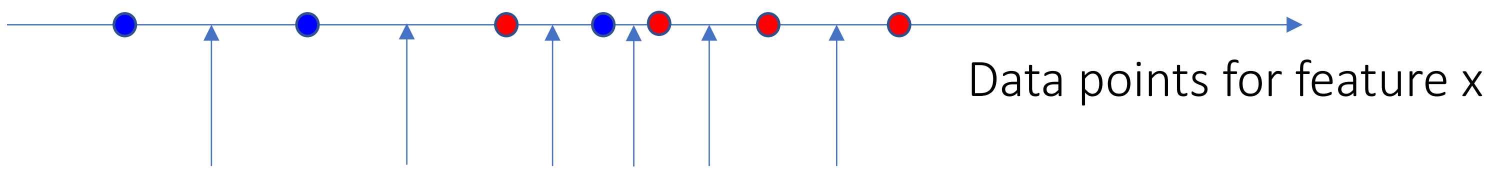
Recap of Decision Trees

Example of Learning a Decision Tree

Decision Trees in Code

Possible Threshold Values

Two classes, red and blue

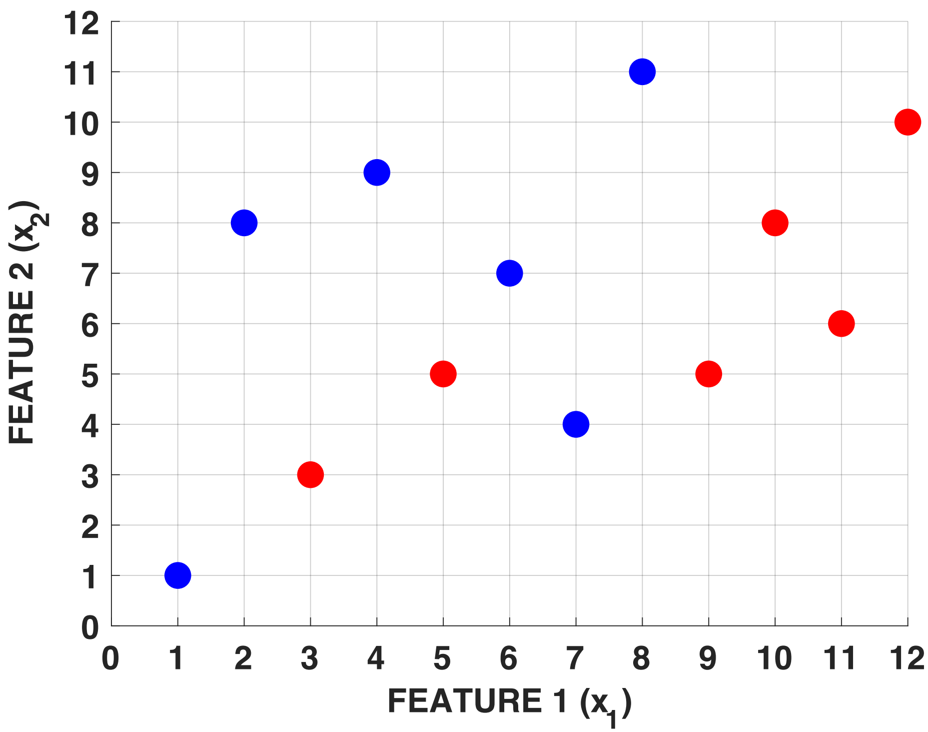


Note that the Gini index does not change between data points since it only depends on numbers of each label to the left and right of it

So we only need to evaluate thresholds between data points:
e.g., can pick thresholds halfway between each pair of data points

If we have n datapoints, we have n-1 thresholds t to consider per feature
(as shown above: n = 7, number of thresholds = 6)

Example



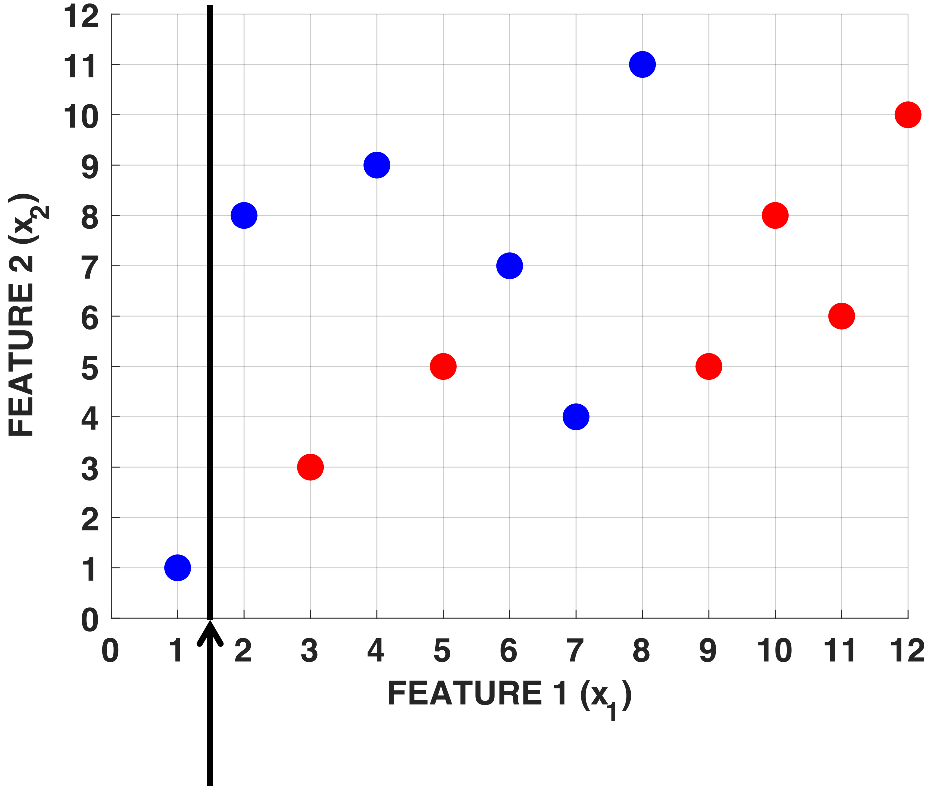
Two class problem

P(class = red) = P(class = blue) = 6/12

Gini index = $1 - (0.5)^2 - (0.5)^2 = 0.5$

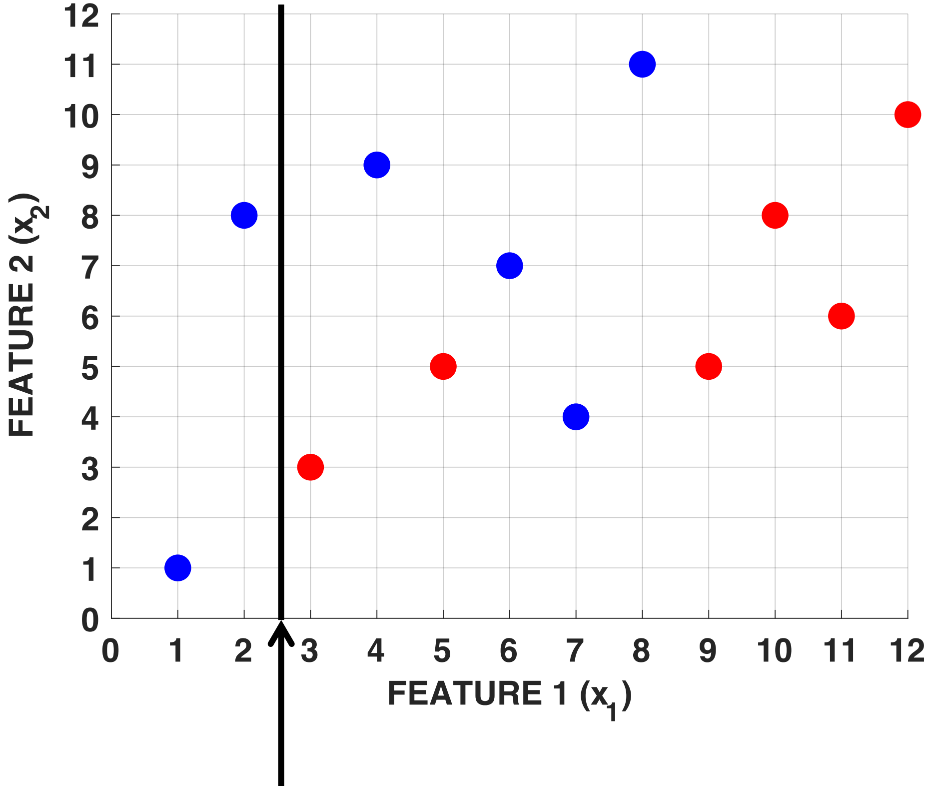
We would like to split the data with thresholds, to find subsets with lower Gini index values

Evaluating the Gini Index of Different Thresholds for Feature 1



Threshold t = 1.5: $x_1 > 1.5$?
$P_L = 1/12$, $P_R = 11/12$
$G_L = 0.0$, $G_R = 0.496$
Gini(t) = $(P_L\, G_L + P_R\, G_R) = 0.455$

Evaluating the Gini Index of Different Thresholds for Feature 1



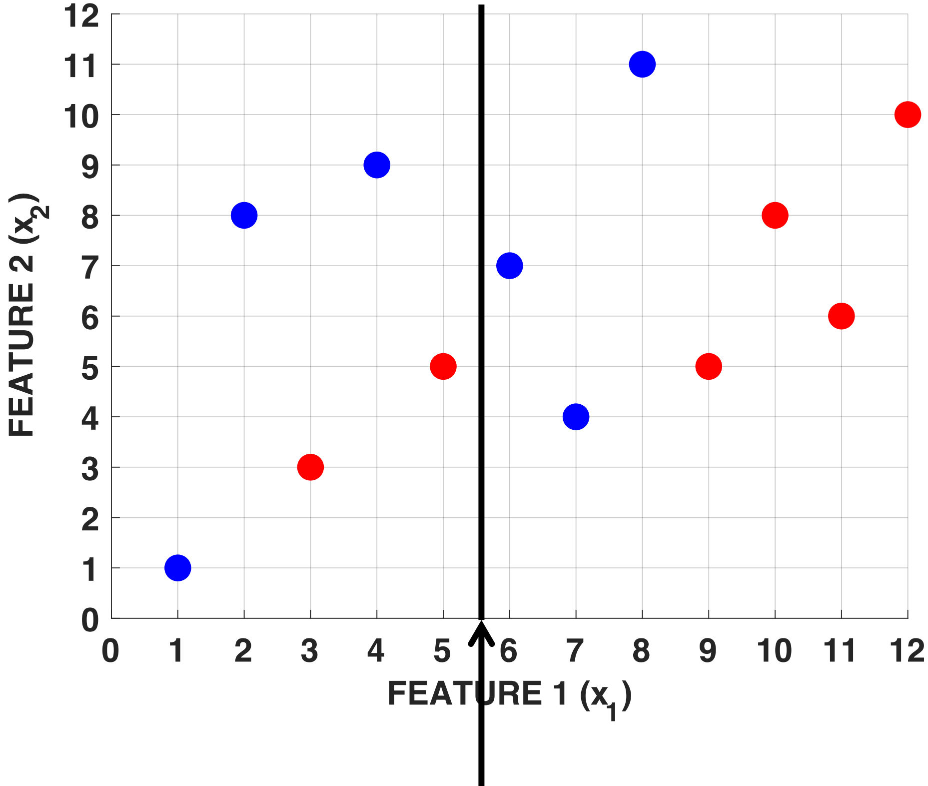
Threshold $t = 2.5$: $x_1 > 2.5$?

$P_L = 2/12$, $P_R = 10/12$

$G_L = 0.0$, $G_R = 0.48$

$\text{Gini}(t) = (P_L G_L + P_R G_R) = 0.4$

Evaluating the Gini Index of Different Thresholds for Feature 1



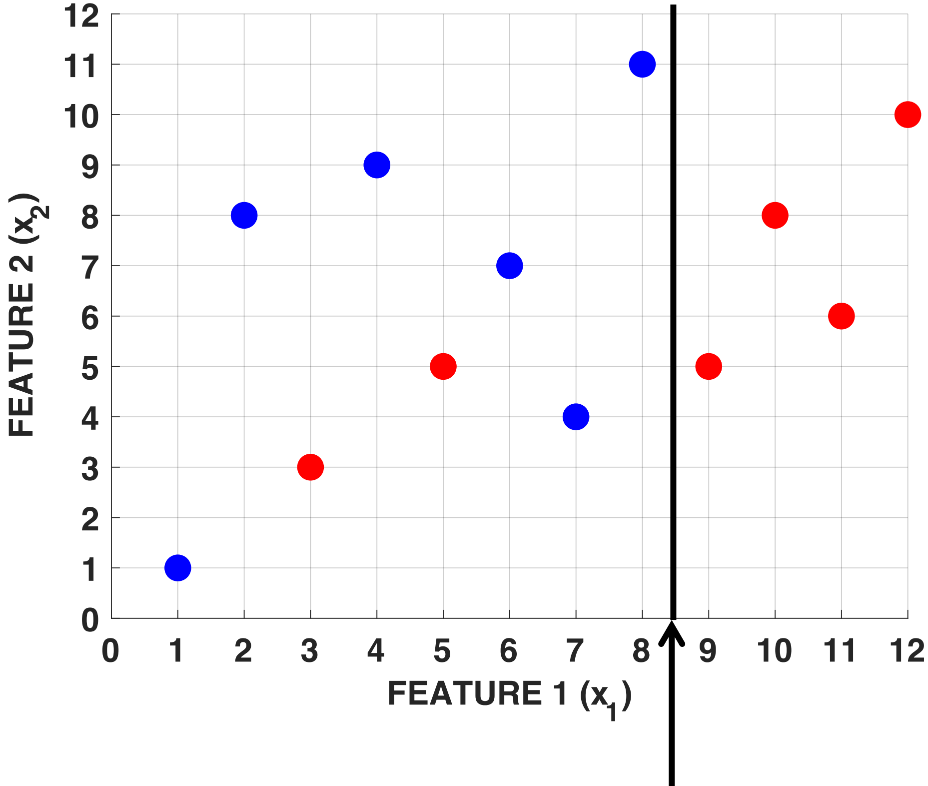
Threshold t = 5.5: $x_1 > 5.5$?

$P_L = 5/12$, $P_R = 8/12$

$G_L = 0.48$, $G_R = 0.49$

Gini(t) = $(P_L\, G_L + P_R\, G_R) = 0.486$

Evaluating the Gini Index of Different Thresholds for Feature 1



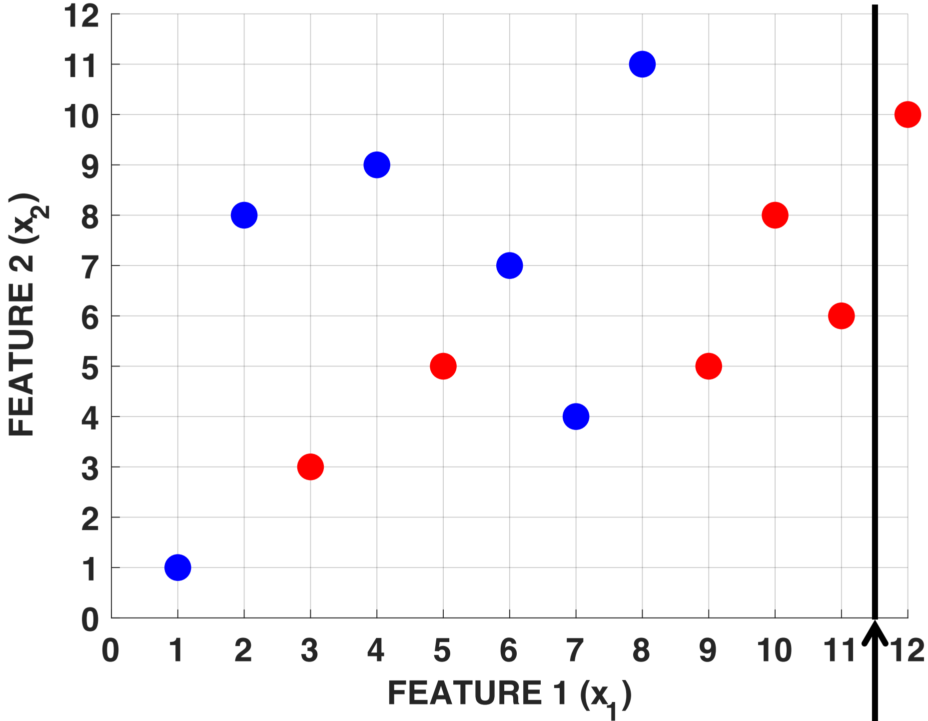
Threshold t = 8.5: $x_1 > 8.5$?

$P_L = 8/12$, $P_R = 4/12$

$G_L = 0.375$, $G_R = 0.0$

$\text{Gini}(t) = (P_L\, G_L + P_R\, G_R) = 0.25$

Evaluating the Gini Index of Different Thresholds for Feature 1



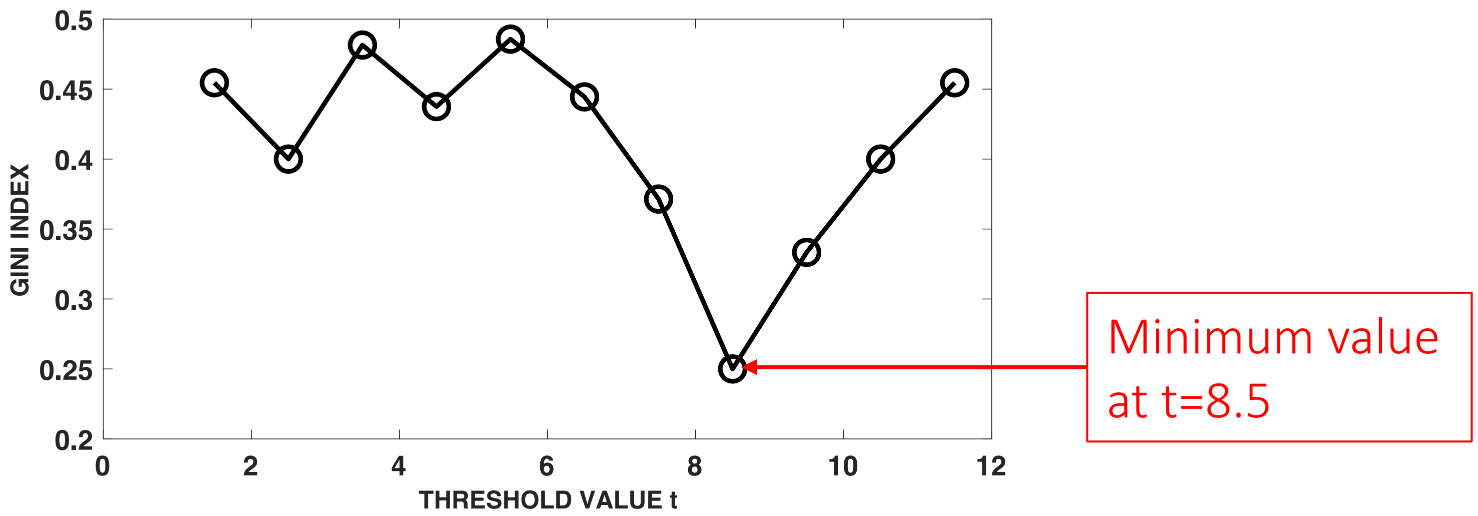
Threshold t = 11.5: $x_1 > 11.5$?

$P_L = 11/12$, $P_R = 1/12$

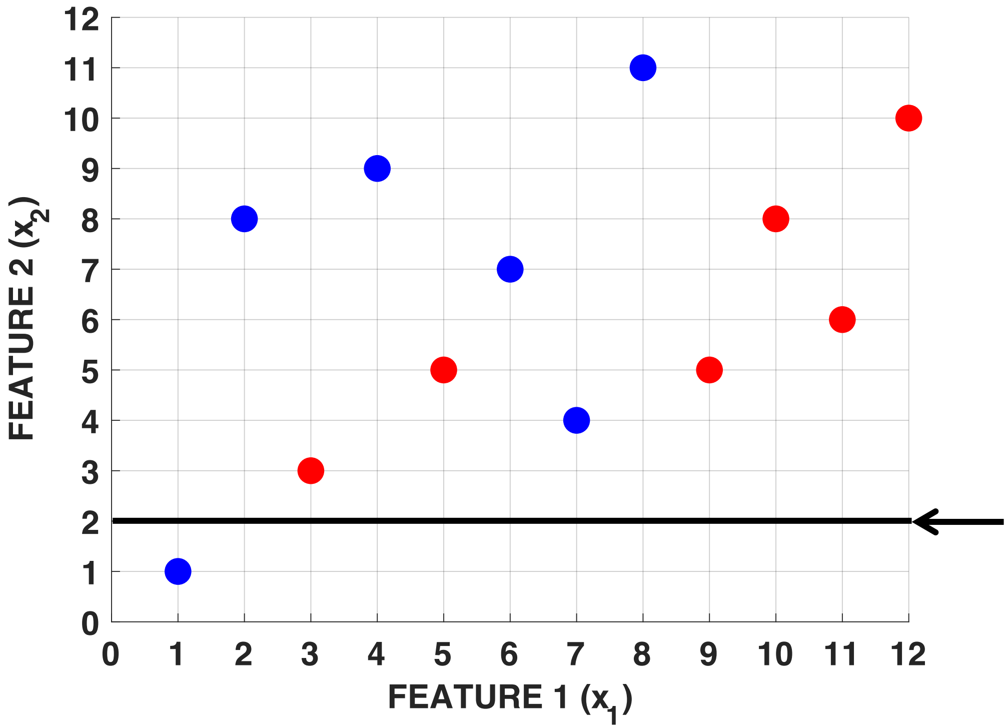
$G_L = 0.496$, $G_R = 0.0$

$\text{Gini}(t) = (P_L G_L + P_R G_R) = 0.455$

Gini Index versus Threshold Values for Feature 1

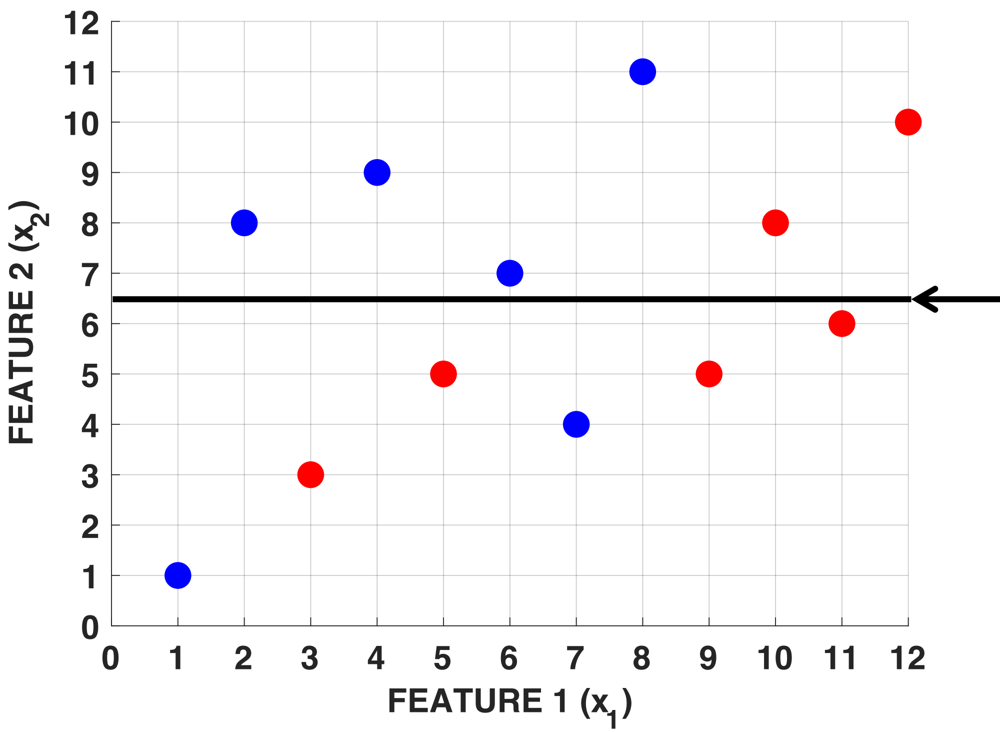


Evaluating the Gini Index of Different Thresholds for Feature 2



Threshold $t = 2$: $x_2 > 2$?
$P_L = 1/12$, $P_R = 11/12$
$G_L = 0.0$, $G_R = 0.496$
$\text{Gini}(t) = (P_L\, G_L + P_R\, G_R) = 0.455$

Evaluating the Gini Index of Different Thresholds for Feature 2



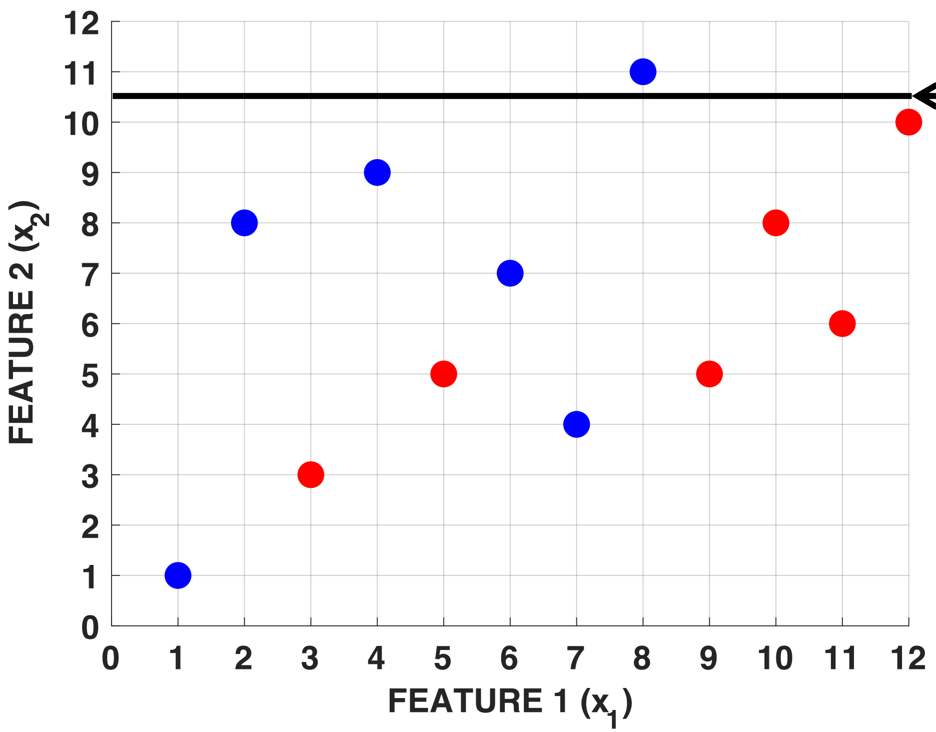
Threshold $t = 6.5$: $x_2 > 6.5$?

$P_L = 6/12$, $P_R = 6/12$

$G_L = 0.444$, $G_R = 0.444$

$\text{Gini}(t) = (P_L G_L + P_R G_R) = 0.444$

Evaluating the Gini Index of Different Thresholds for Feature 2



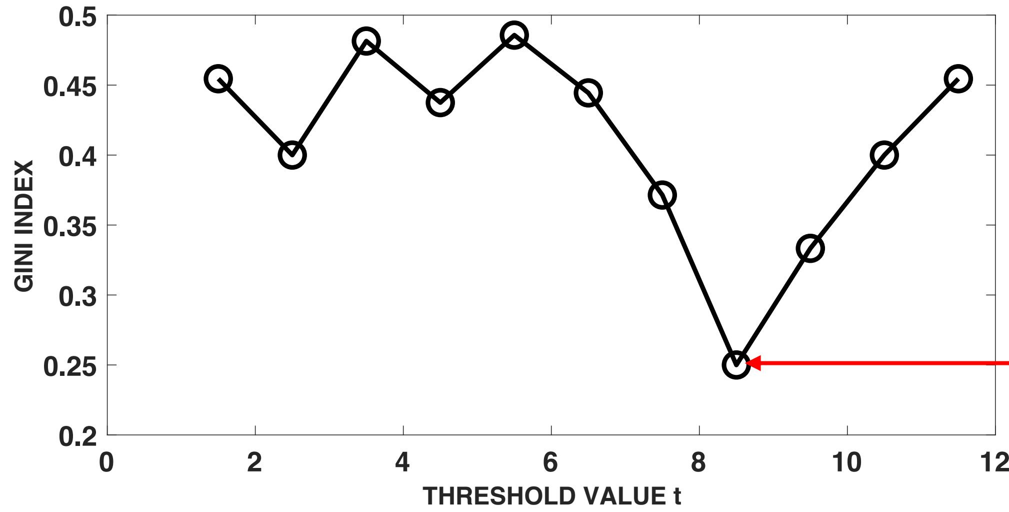
Threshold t = 10.5: $x_2 > 10.5$?

$P_L = 11/12$, $P_R = 1/12$

$G_L = 0.496$, $G_R = 0.0$

$\text{Gini}(t) = (P_L\, G_L + P_R\, G_R) = 0.455$

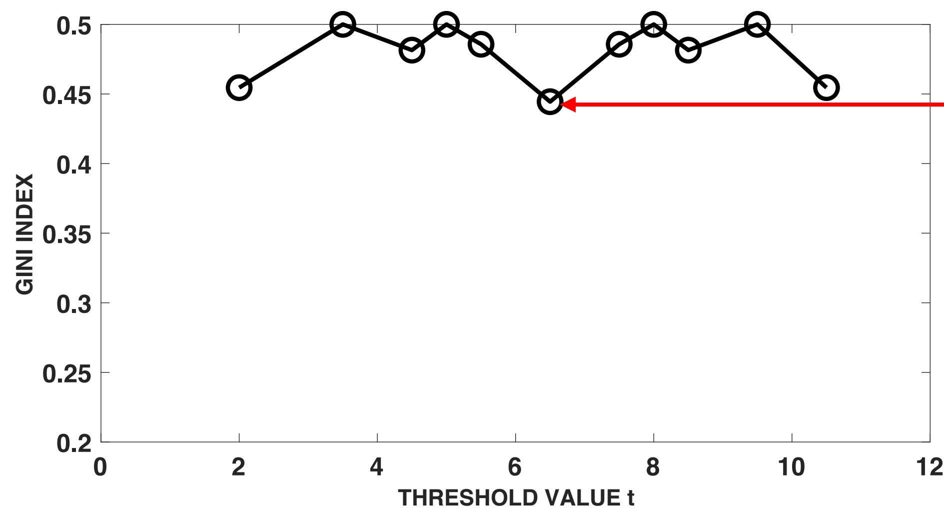
Gini Index versus Threshold Values for Feature 1



Minimum value of 0.25 at t=8.5

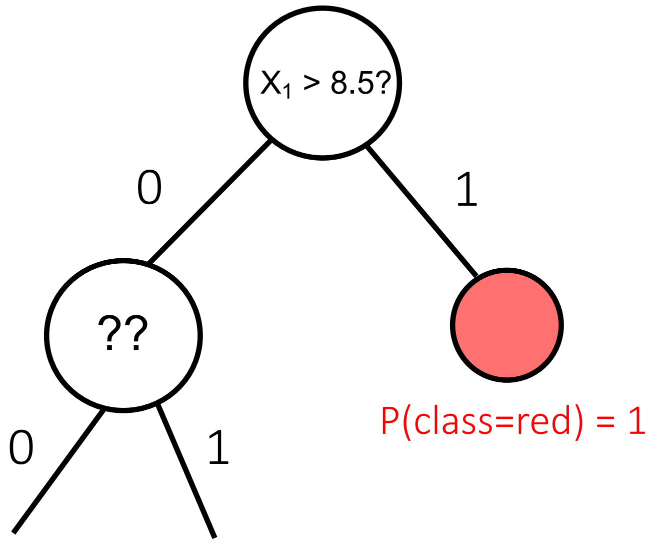
=> Split on Feature 1 at t = 8.5

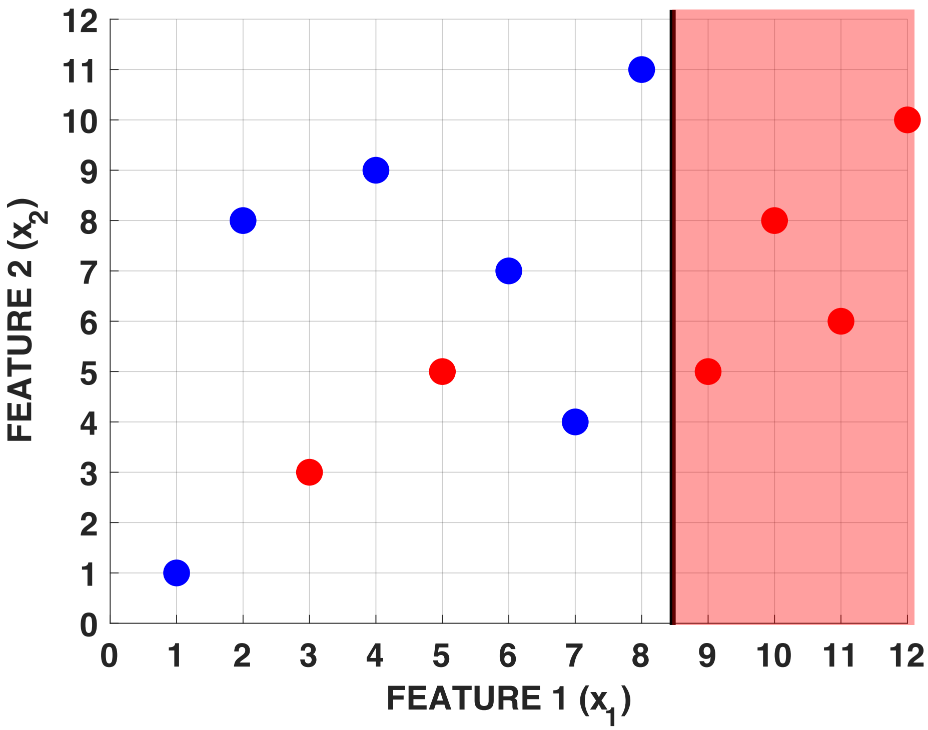
Gini Index versus Threshold Values for Feature 2



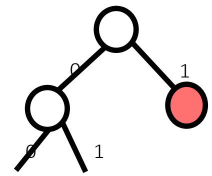
Minimum value of 0.44 at t=6.5

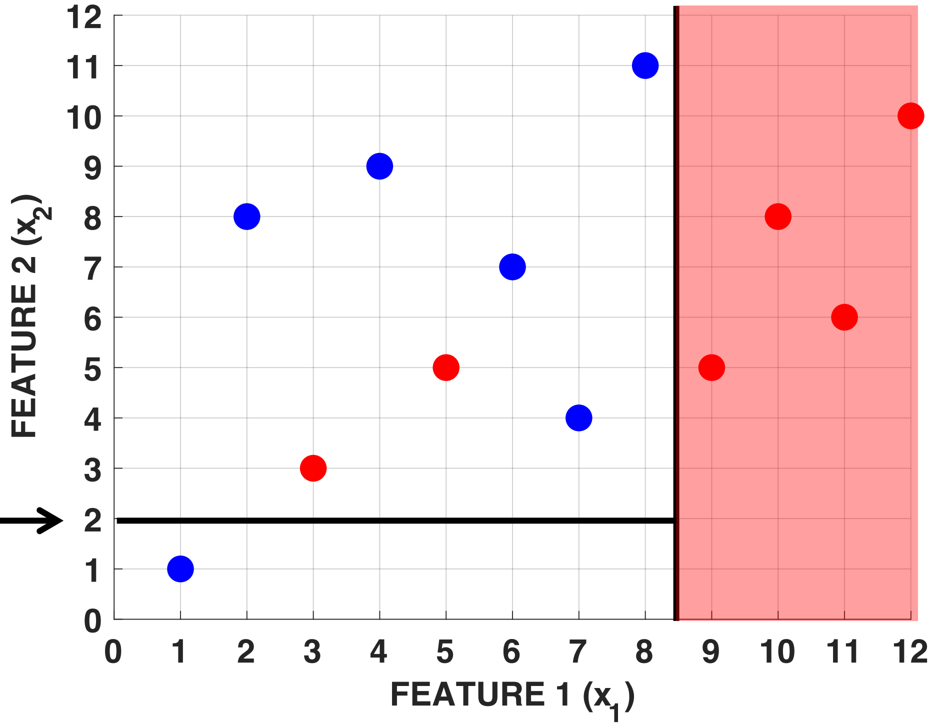
We now have a Root Node





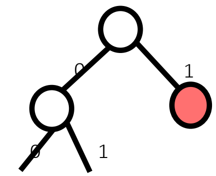
Finding a Threshold for Left Child Node





Threshold t = 2: $x_2 > 2$?
$P_L = 1/8$, $P_R = 7/8$
$G_L = 0.0$, $G_R = 0.408$
Gini(t) = $(P_L\, G_L + P_R\, G_R) = 0.357$

Finding a Threshold for Left Child Node

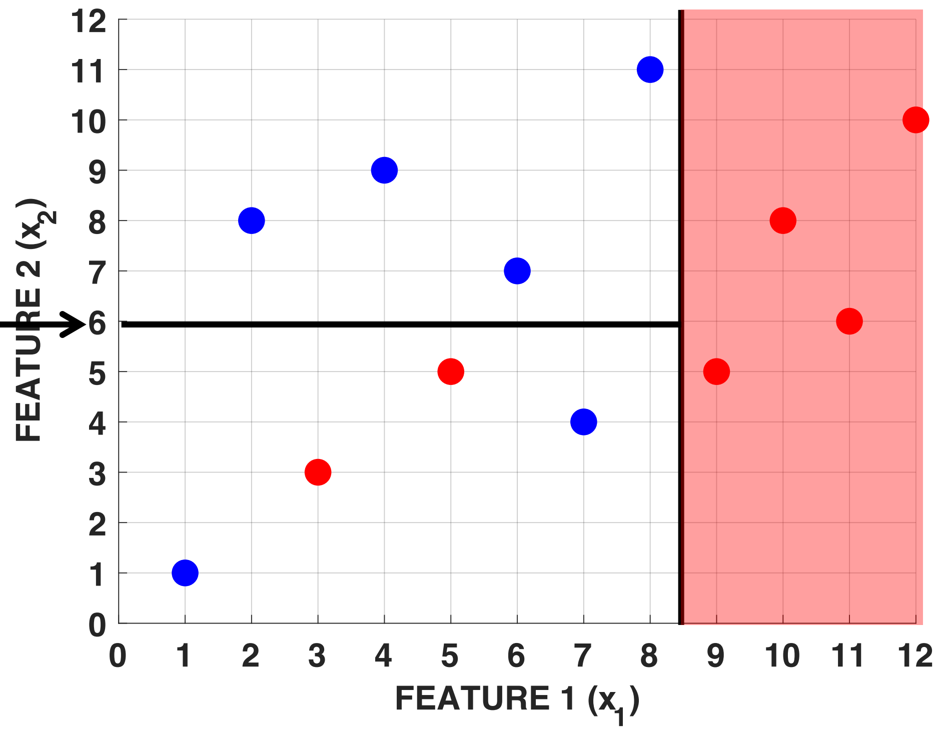


Threshold t = 6: $x_2 > 6$?

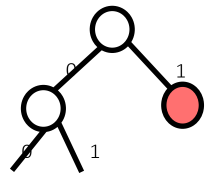
$P_L = 4/8$, $P_R = 4/8$

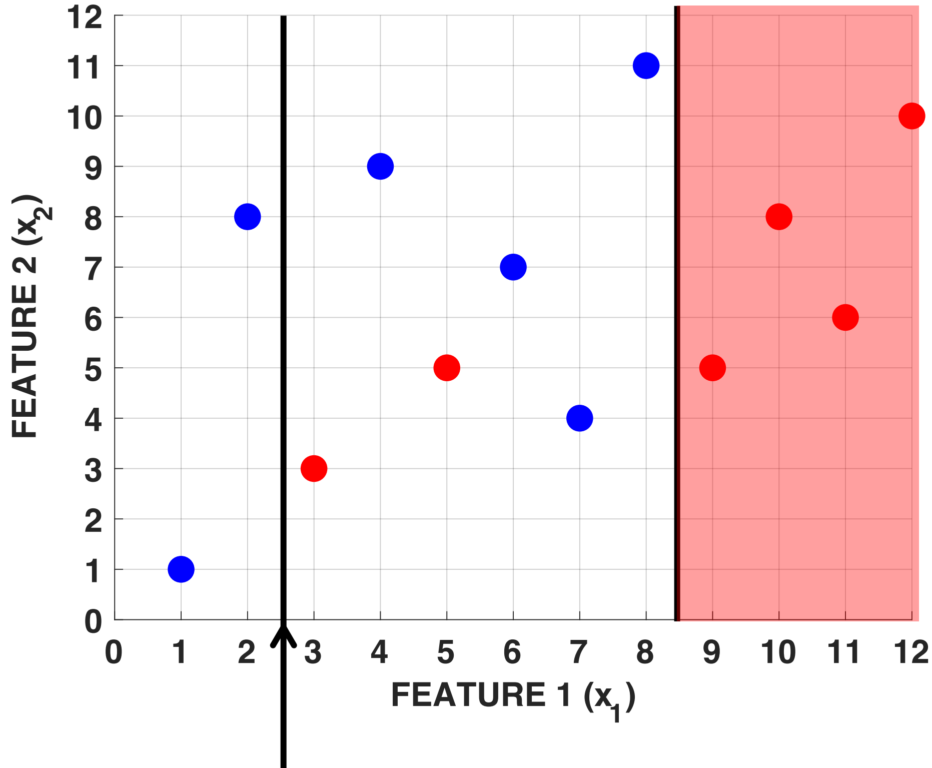
$G_L = 0.5$, $G_R = 0.0$

Gini(t) = $(P_L G_L + P_R G_R) = 0.25$



Finding a Threshold for Left Child Node





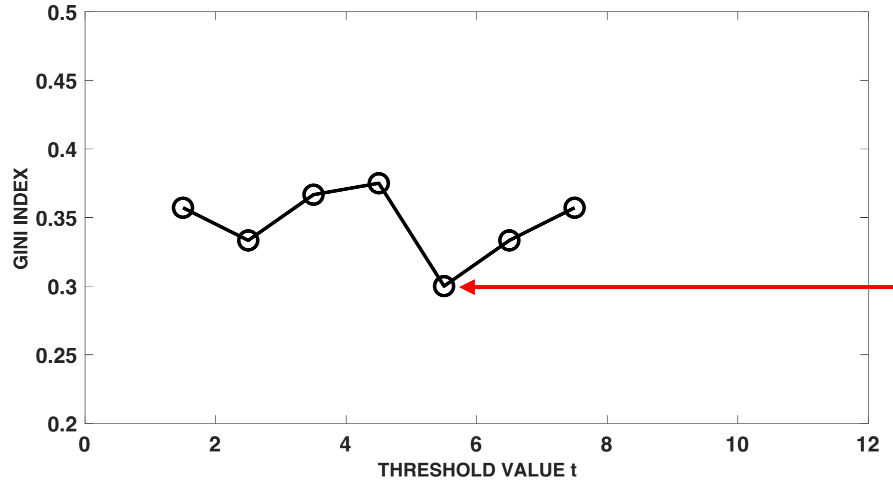
Threshold t = 2.5: $x_2 > 2.5$?

$P_L = 2/8$, $P_R = 6/8$

$G_L = 0.0$, $G_R = 0.44$

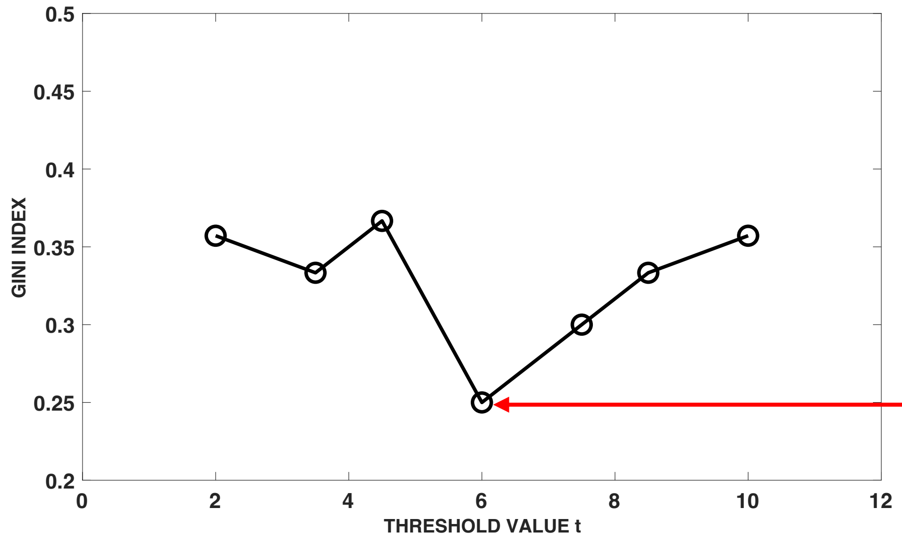
$\text{Gini}(t) = (P_L G_L + P_R G_R) = 0.357$

Gini Index versus Threshold Values for Feature 1



Minimum value of 0.3 at t=5.5

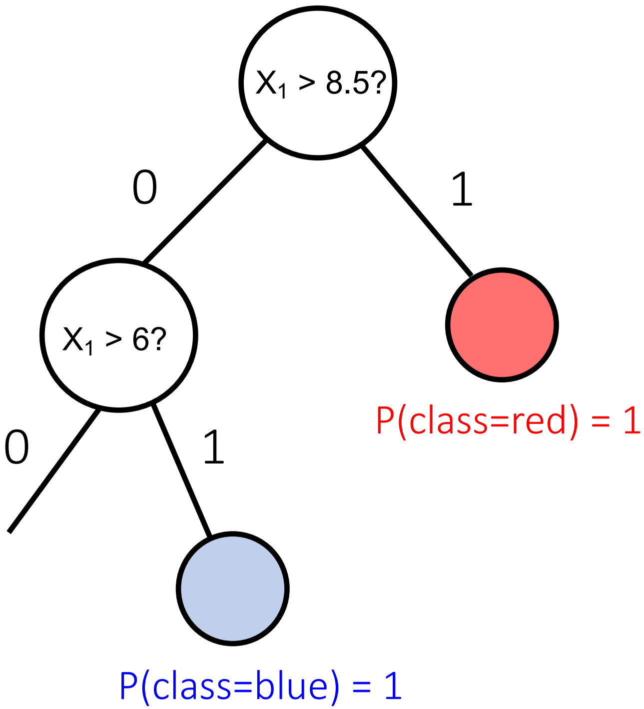
Gini Index versus Threshold Values for Feature 2

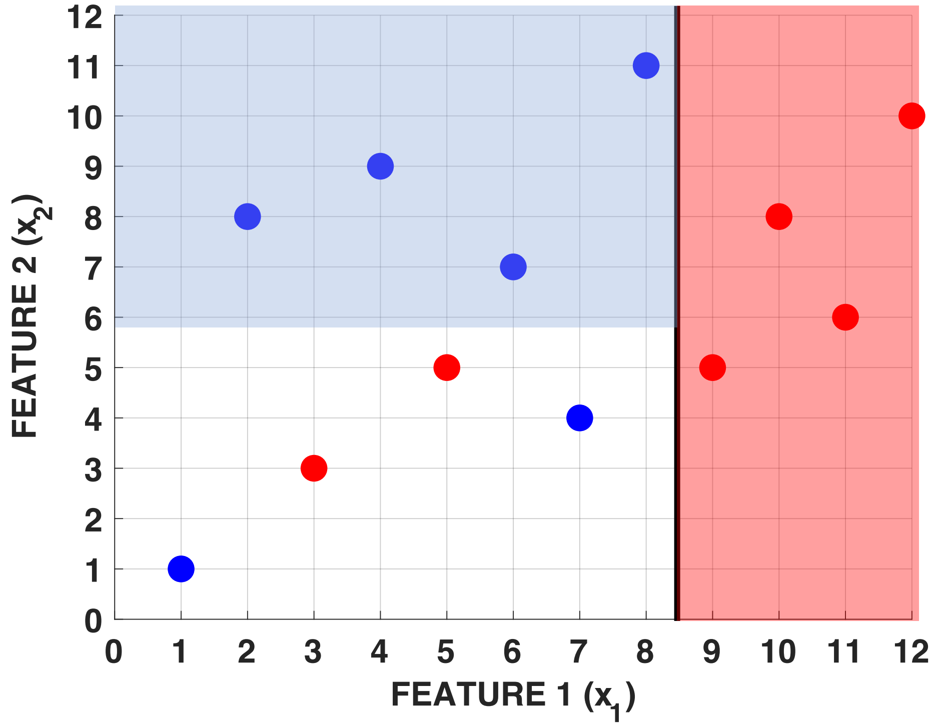


=> Split on Feature 2 at t = 6

Minimum value of 0.25 at t=6

We now add a Left Child Node





Additional Details on Tree-Growing

Features that are not real-valued?

Binary: simple, just 1 split to test
Categorical with M values: can create M binary features

Can compare their Gini values with thresholded real-valued features

Tree Size?

Typically: grow a large tree, e.g., depth 10 or more
Use validation to “prune” tree back
(Various heuristics used for pruning)
Select the pruned tree that has the highest accuracy on validation data

Questions?

Recap of Decision Trees

Example of Learning a Decision Tree

Decision Trees in Code

Decision Trees in Code

HW4 Problem 3: Implementing the decision tree learning algorithm
Basic object in a tree are *nodes*

```
class Node:
    """ A class representing a node in a decision tree.
    """

    def __init__(self, depth):
        self.depth = depth          # What level of the tree this node is at; depth=0 is the root node

        self.split_feature = None   # The index of the feature that this node splits, if any
        self.threshold = None       # The threshold used to split the feature

        self.left_child = None      # A node object (or None) representing the left-hand child of this node
        self.right_child = None     # A node object (or None) representing the right-hand child of this node

        self.probs = None           # A numpy array of length 2 representing [p(y=0), p(y=1)] at this node

    def __repr__(self):
        # Gives a nice looking representation if you call print on a node

        return f'DT Node: \n -| Depth: {self.depth}' \
               f'\n -| Split feature: {self.split_feature}' \
               f'\n -| Threshold: {self.threshold}' \
               f'\n -| Probs: {self.probs}'
```

Decision Trees in Code

```python
def build_tree(self, X, y, depth):
    """ Recursively builds the decision tree.
    """

    # Create a new node
    node = Node(depth)
    if depth == 0:
        self.root = node

    # Get the class probabilities for this node
    node.probs = self.class_prob_vector(y)

    # Check if this new node is a leaf node; otherwise, split it
    if self.leaf_condition(node):
        return node
    else:
        # Find which feature to split on and the splitting threshold
        split_idx, split_threshold = self.find_best_split(X, y)

        # Create left/right splits of data based on split_idx, split_threshold
        X_L, y_L = # TODO
        X_R, y_R = # TODO

        # Recursively create the left/right nodes
        node_L = # TODO
        node_R = # TODO

        # Fill in node information
        node.split_feature = # TODO
        node.threshold = # TODO
        node.left_child = # TODO
        node.right_child = # TODO

        return node
```

Decision Trees in Code

To make predictions: we traverse the tree until we hit a leaf node

```python
def _predict(self, x):
    """ Makes predictions on individual datapoints x.
    """
    current_node = self.root

    while True:
        if self.leaf_condition(current_node):
            # If we're at a leaf node, make a prediction based on the probabilities
            probs = current_node.probs
            y_hat = np.argmax(probs)
            return y_hat
        else:
            # Otherwise, traverse the tree based on the splits
            go_left = x[current_node.split_feature] <= current_node.threshold
            if go_left:
                current_node = current_node.left_child
            else:
                current_node = current_node.right_child
```

sklearn.tree.DecisionTreeClassifier

class `sklearn.tree.`**`DecisionTreeClassifier`**(**, criterion='gini', splitter='best', max_depth=None, min_samples_split=2, min_samples_leaf=1, min_weight_fraction_leaf=0.0, max_features=None, random_state=None, max_leaf_nodes=None, min_impurity_decrease=0.0, class_weight=None, ccp_alpha=0.0*) [source]

A decision tree classifier.

Read more in the User Guide.

Parameters::

criterion : *{"gini", "entropy", "log_loss"}, default="gini"*
The function to measure the quality of a split. Supported criteria are "gini" for the Gini impurity and "log_loss" and "entropy" both for the Shannon information gain, see Mathematical formulation.

splitter : *{"best", "random"}, default="best"*
The strategy used to choose the split at each node. Supported strategies are "best" to choose the best split and "random" to choose the best random split.

max_depth : *int, default=None*
The maximum depth of the tree. If None, then nodes are expanded until all leaves are pure or until all leaves contain less than min_samples_split samples.

min_samples_split : *int or float, default=2*
The minimum number of samples required to split an internal node:

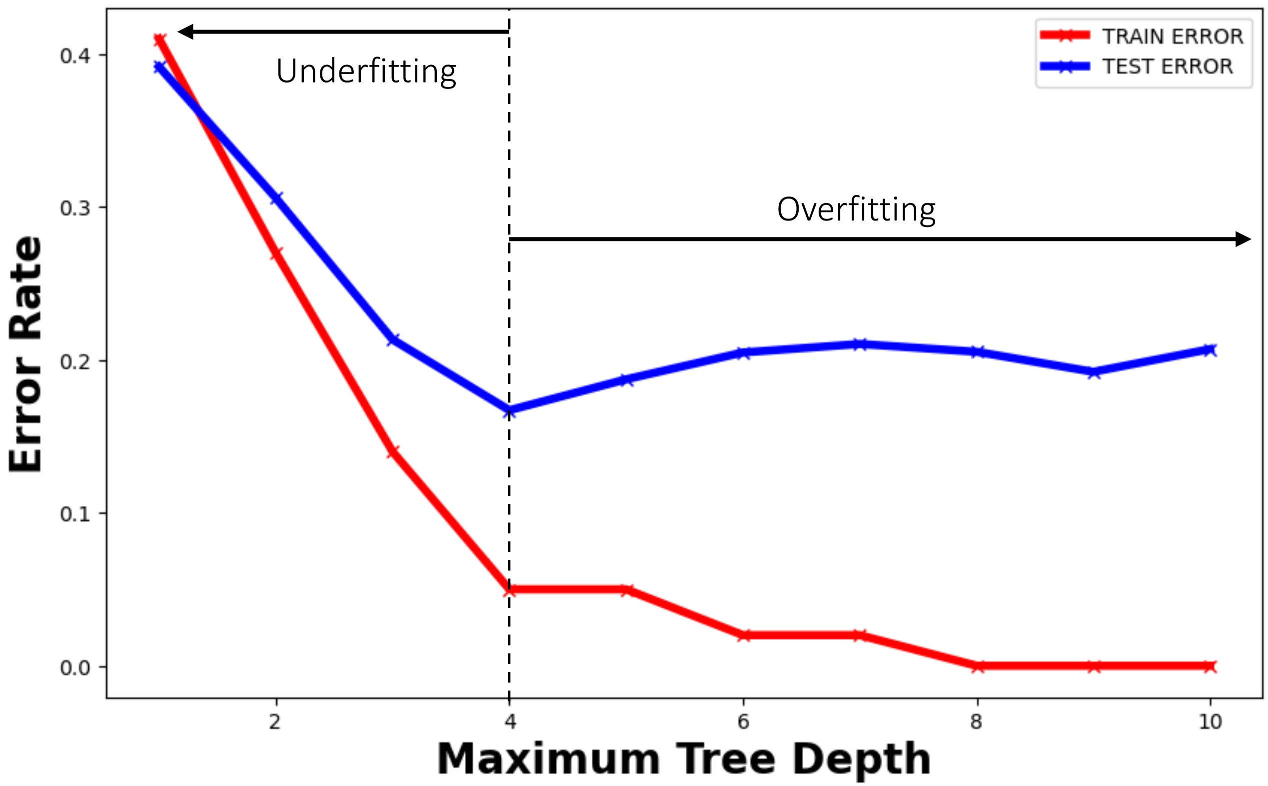
- If int, then consider `min_samples_split` as the minimum number.
- If float, then `min_samples_split` is a fraction and `ceil(min_samples_split * n_samples)` are the minimum number of samples for each split.

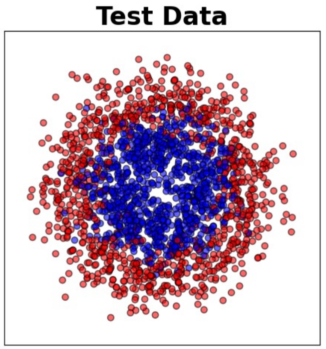
Changed in version 0.18: Added float values for fractions.

min_samples_leaf : *int or float, default=1*
The minimum number of samples required to be at a leaf node. A split point at any depth will only be considered if it leaves at least `min_samples_leaf` training samples in each of the left and right branches. This may have the effect of smoothing the model, especially in regression.

- If int, then consider `min_samples_leaf` as the minimum number.
- If float, then `min_samples_leaf` is a fraction and `ceil(min_samples_leaf * n_samples)` are the minimum number of samples for each node.

Error as a function of Tree Complexity

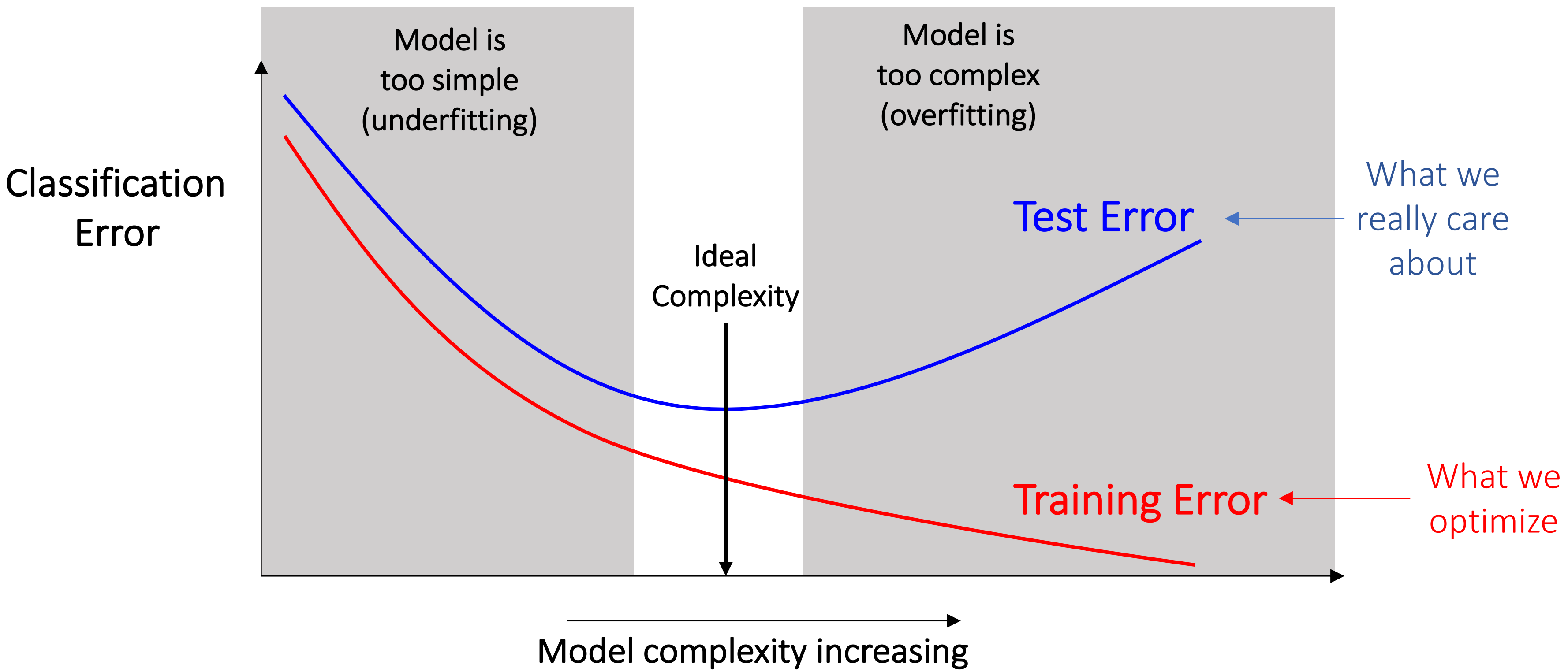




Complexity of decision tree model is increasing

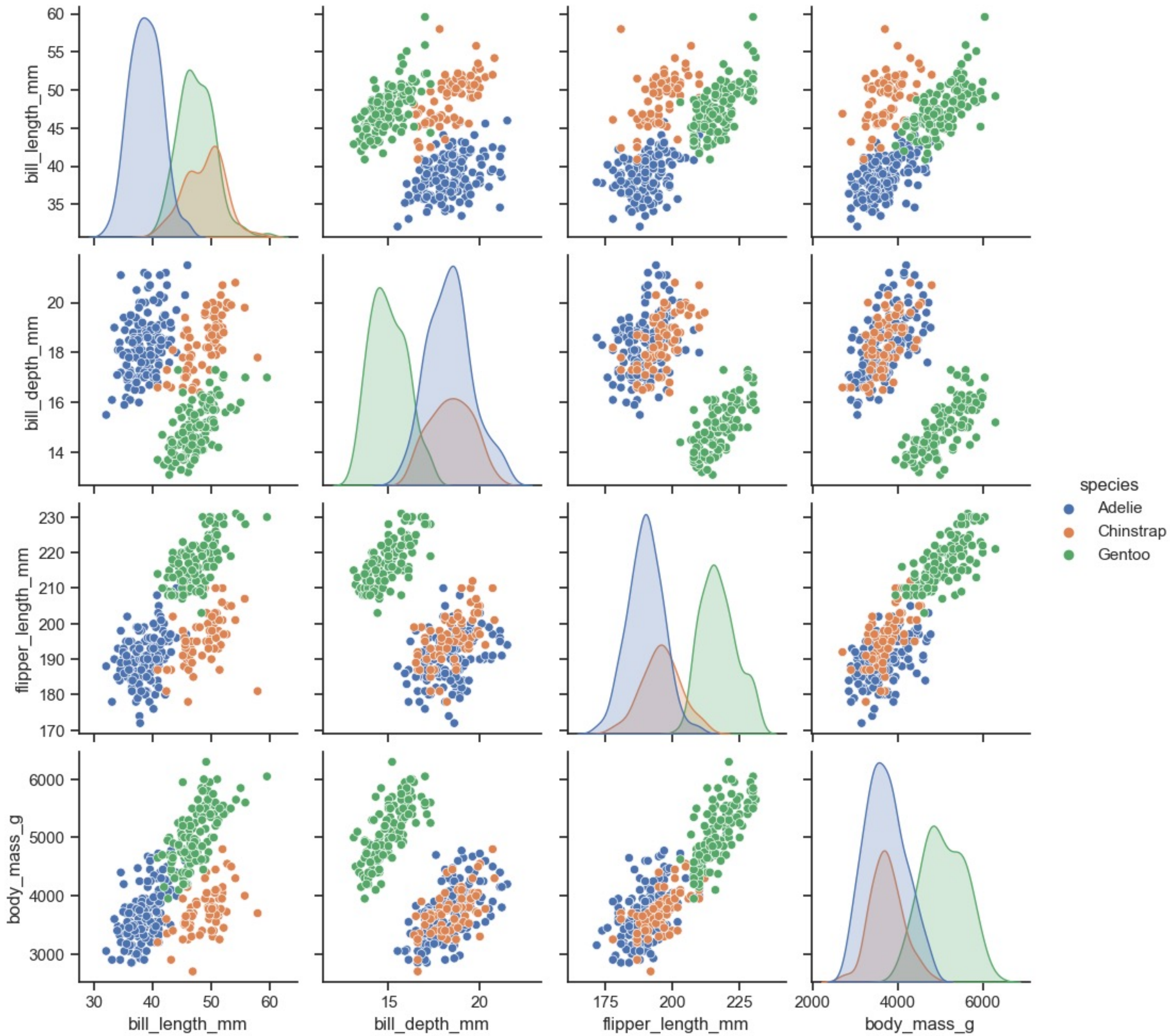
Trees trained on 100 datapoints with scikit-learn defaults, test error on 5000 datapoints

General Error-Complexity Tradeoff

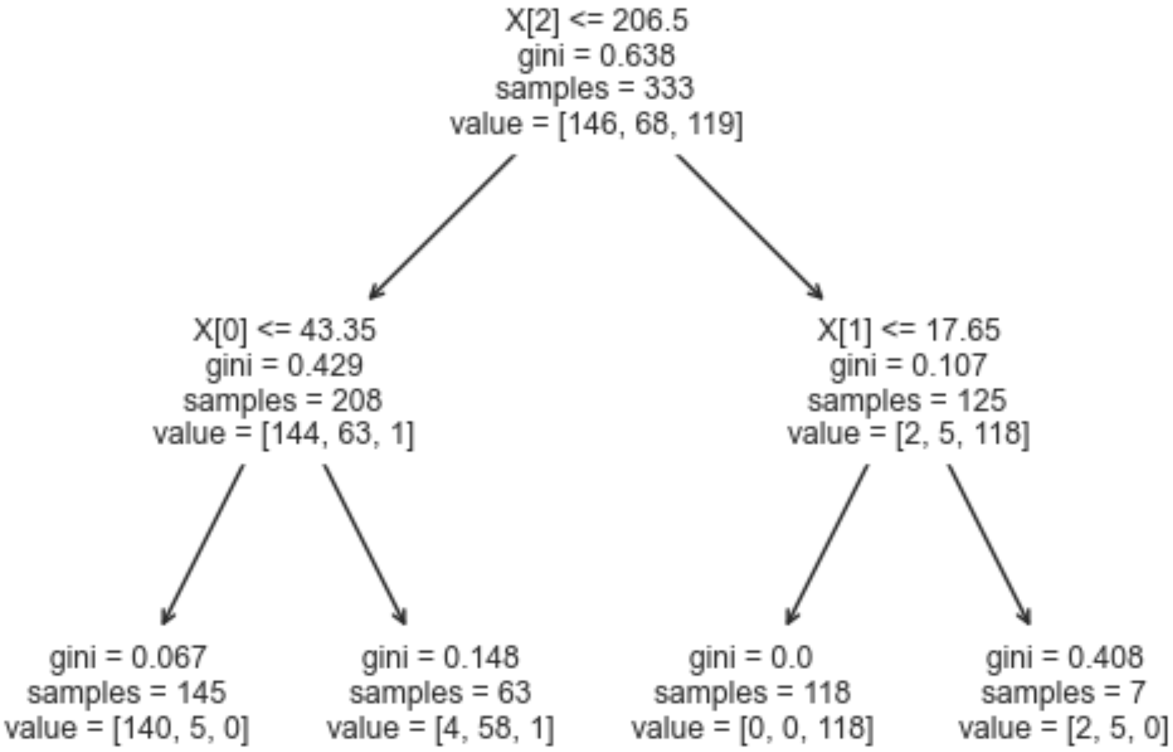


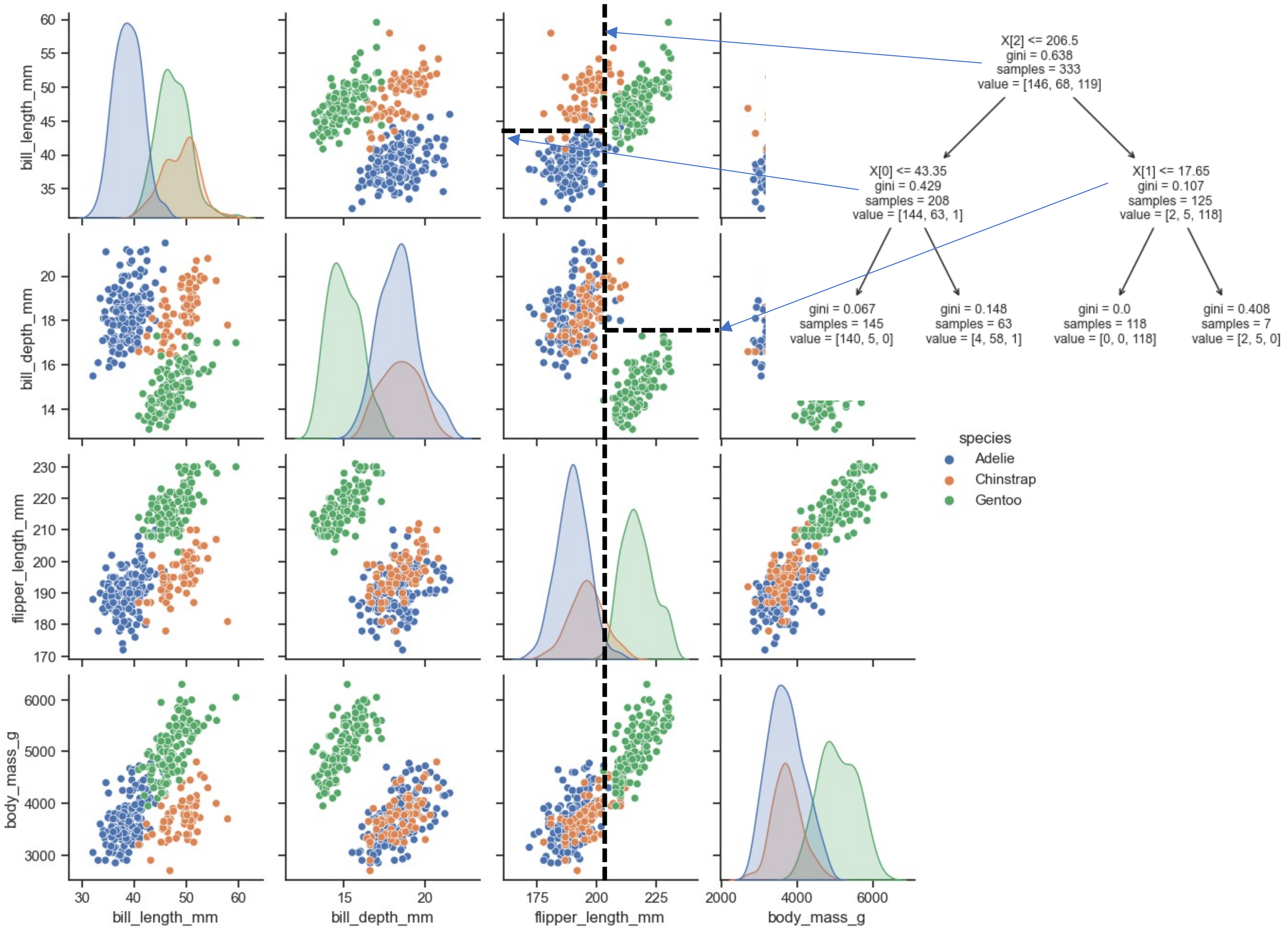
Penguin Dataset

(from earlier lectures)



Decision Tree for Penguin Data (MaxDepth=2)



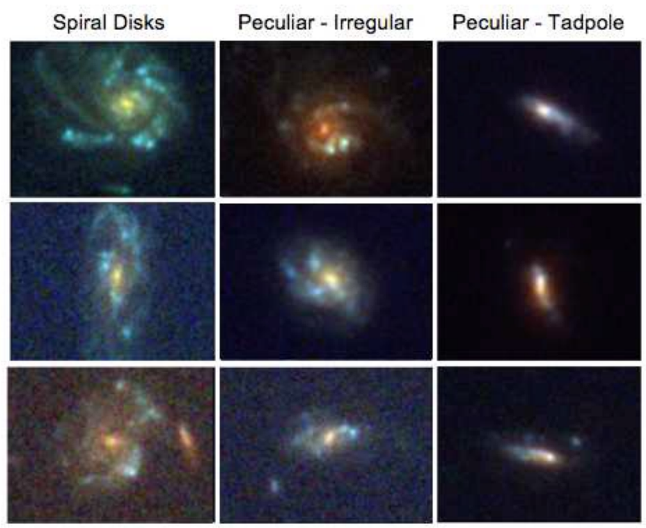


X[2] <= 206.5
gini = 0.638
samples = 333
value = [146, 68, 119]
X[0] <= 43.35
gini = 0.429
samples = 208
value = [144, 63, 1]
X[1] <= 17.65
gini = 0.107
samples = 125
value = [2, 5, 118]
gini = 0.067
samples = 145
value = [140, 5, 0]
gini = 0.148
samples = 63
value = [4, 58, 1]
gini = 0.0
samples = 118
value = [0, 0, 118]
gini = 0.408
samples = 7
value = [2, 5, 0]
species
Adelie
Chinstrap
Gentoo
bill_length_mm
bill_depth_mm
flipper_length_mm
body_mass_g

Decision Tree Classifiers in Astronomy

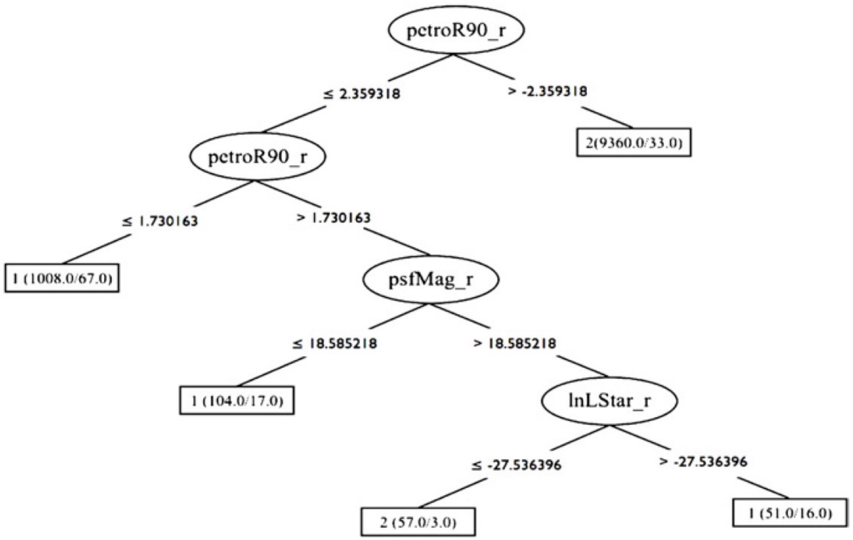
Digital telescopes produce images of billions of sky objects
Far too many images for human classification
Large-scale “sky catalogs” created automatically using machine learning

Different Classes of Galaxy Images



From Neichel at al, The Astronomy Journal, 2008

Decision Tree Classifier learned from 50,000 labeled images



From Vasconcellos at al, The Astronomy Journal, 2011

Decision Tree Classifiers in Medicine

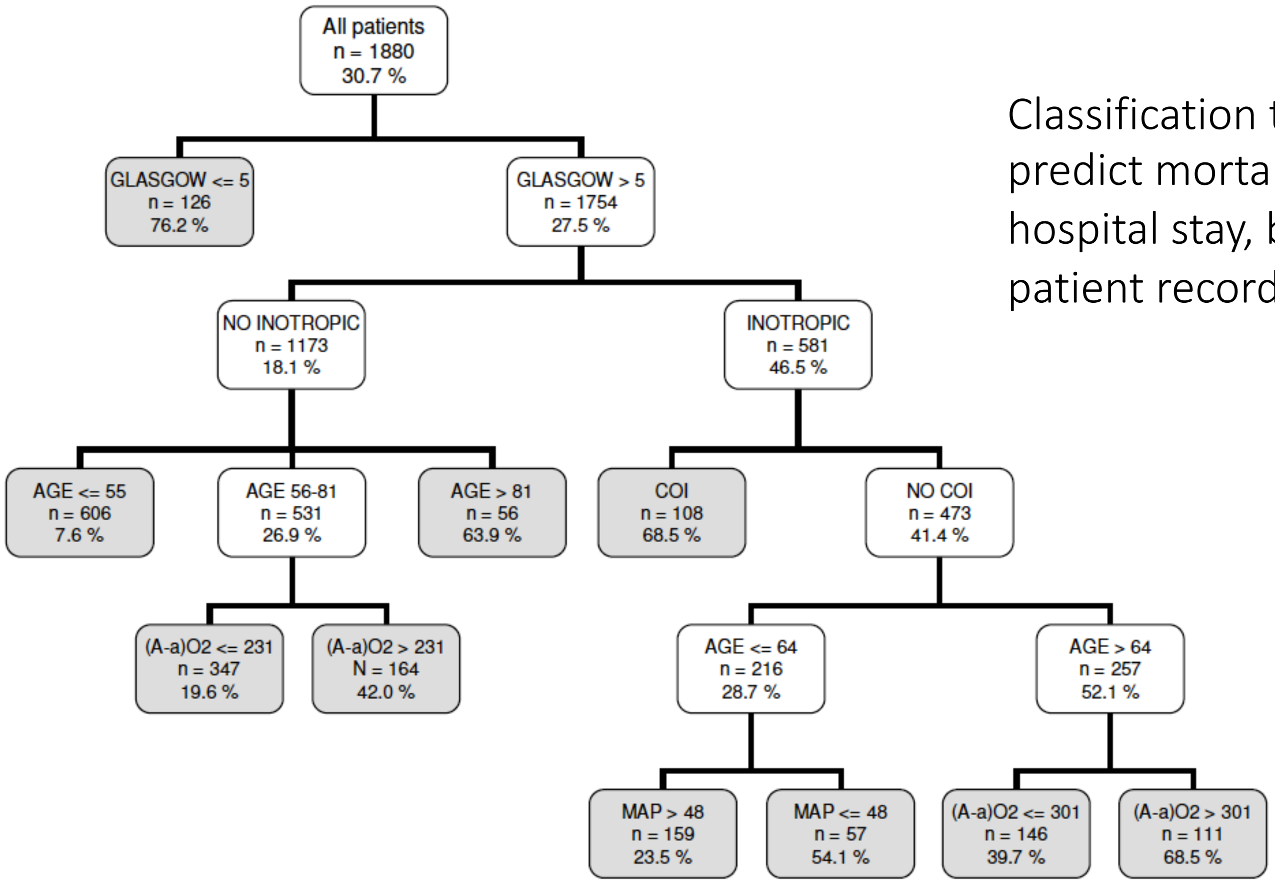


Figure 3
Classification tree by C4.5 algorithm. The gray squares denote terminal prognostic subgroups. INOT: Inotropic therapy; (A-a)O2 gradient: Alveolar-arterial oxygen gradient (mmHg); MV: Mechanical ventilation; COI: Chronic organ insufficiency; MAP: Mean arterial pressure.

Classification tree to predict mortality during hospital stay, built from patient records

From Trujillano et al, Stratification of the severity of critically-ill patients, BMC Medical Research Methodology, 2009

Strength/Weaknesses of Decision Tree Classifiers

- Strengths
  - Easy to interpret by humans (rule-like)
    - At least for small trees
  - Can easily handle categorical and binary features
  - Scale-invariant for real-valued variables

- Weaknesses
  - Axis-parallel linear decision boundaries

    -> has “bias” (in terms of bias-variance)

    -> not as flexible as other models
  - High variance
    - small change in data can cause a large change in tree
    - E.g., if a different root node is selected, tree may be quite different

Summary of Decision Tree Classifiers

- Decision tree classifiers
  - Flexible functional form (but linear/axis-parallel)
  - At each internal node, split on one variable
  - At leaves, produces vector of class probabilities

- Learning decision trees
  - Score all splits & pick best
    - Criterion used for splitting? Gini index
  - Stopping criteria

- Complexity depends on number of nodes/depth

Additional questions? outside after class