Decision Trees

The City College of New York
CSc 59929 – Introduction to Machine Learning
Fall 2017 – Erik K. Grimmelmann, Ph.D.

Decision Trees

- Decision Trees classify instances by sorting them down the tree from the **root** to some **leaf** node which provides the classification of the instance.
- Each **node** in the tree specifies examination of some **feature** of the instance and each **branch** from that node corresponds to one of the possible values (or range of values) of this feature.

Decision Tree Classification

- An instance is classified by starting at the root node, testing the feature specified by this node, and then moving down the tree branch corresponding to the value (or range of values) of the feature corresponding to the node for the given instance. This process is then repeated for the subtree rooted at the new node until there are no remaining features to be examined.

Decision Tree Learning

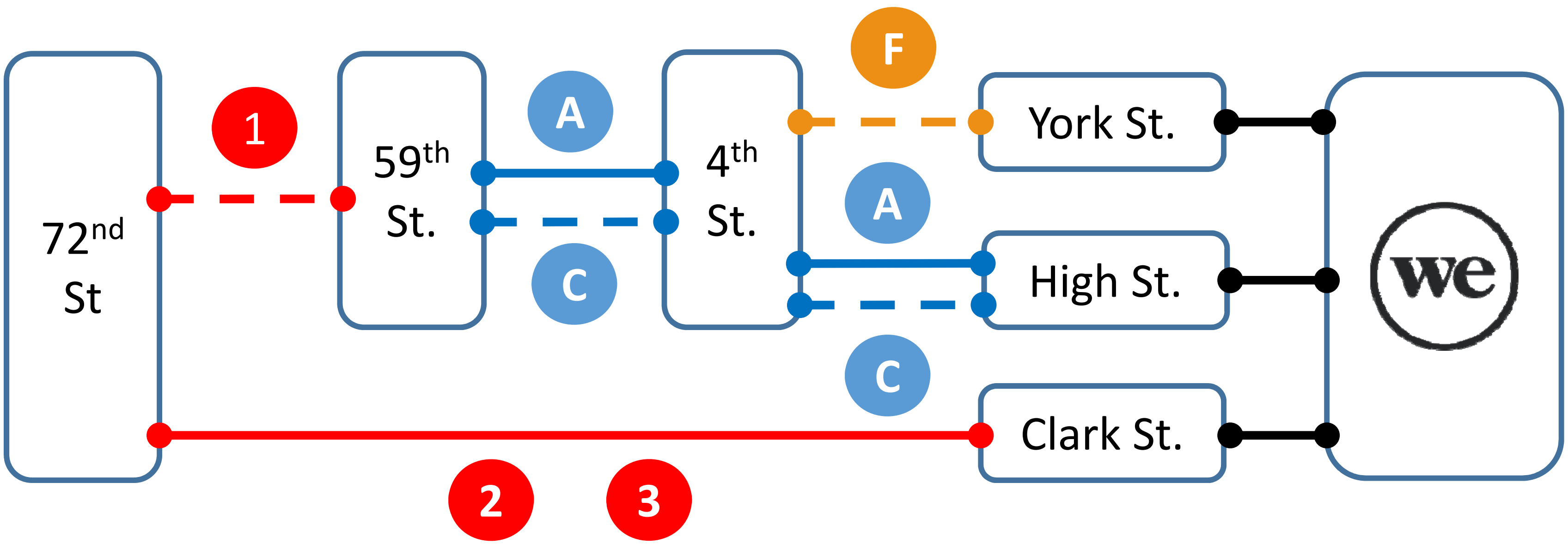
- Training a decision tree consists of determining which feature to assign to each node and which value (or range of values) of that feature to assign to each branch.
- Most decision tree learning algorithms utilize a top-down, **greedy** search through the space of possible decision trees.

Greedy Algorithms

- A **greedy algorithm** makes the locally optimal choice at each stage with the hope of finding a global optimal solution (or sometime close to it).

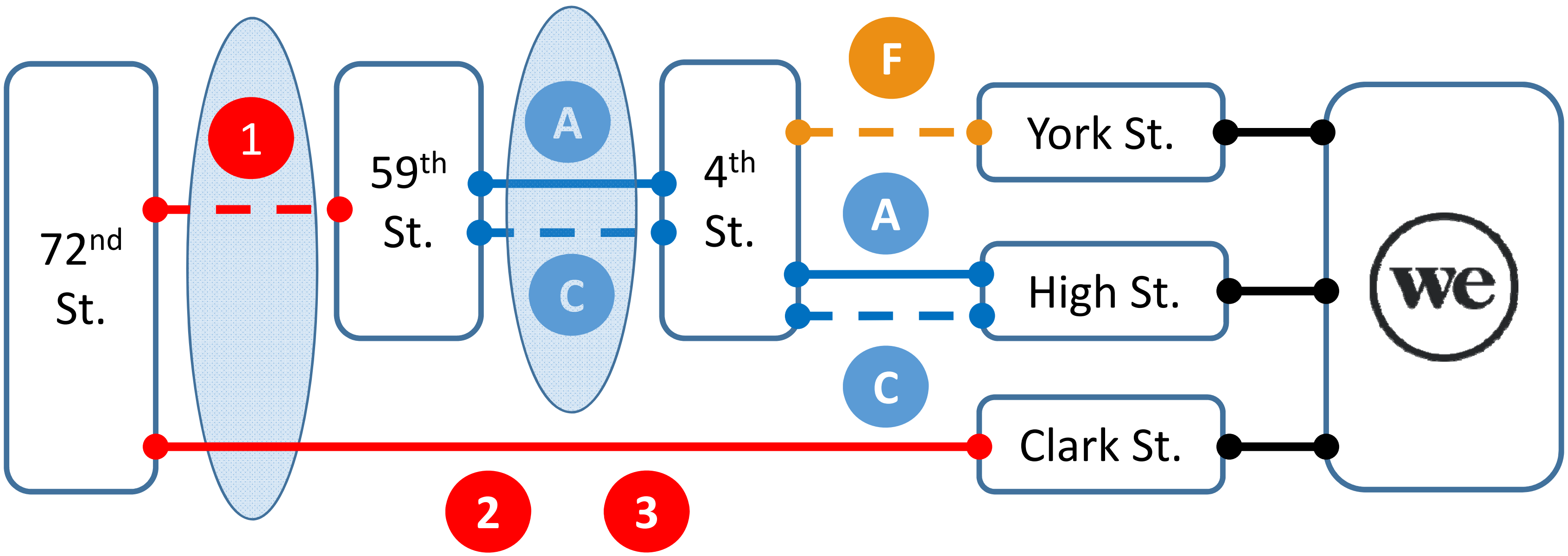
Subway Problem: Choosing My Route

- My commute from 72nd St & Broadway to the WeWork in Dumbo Heights.



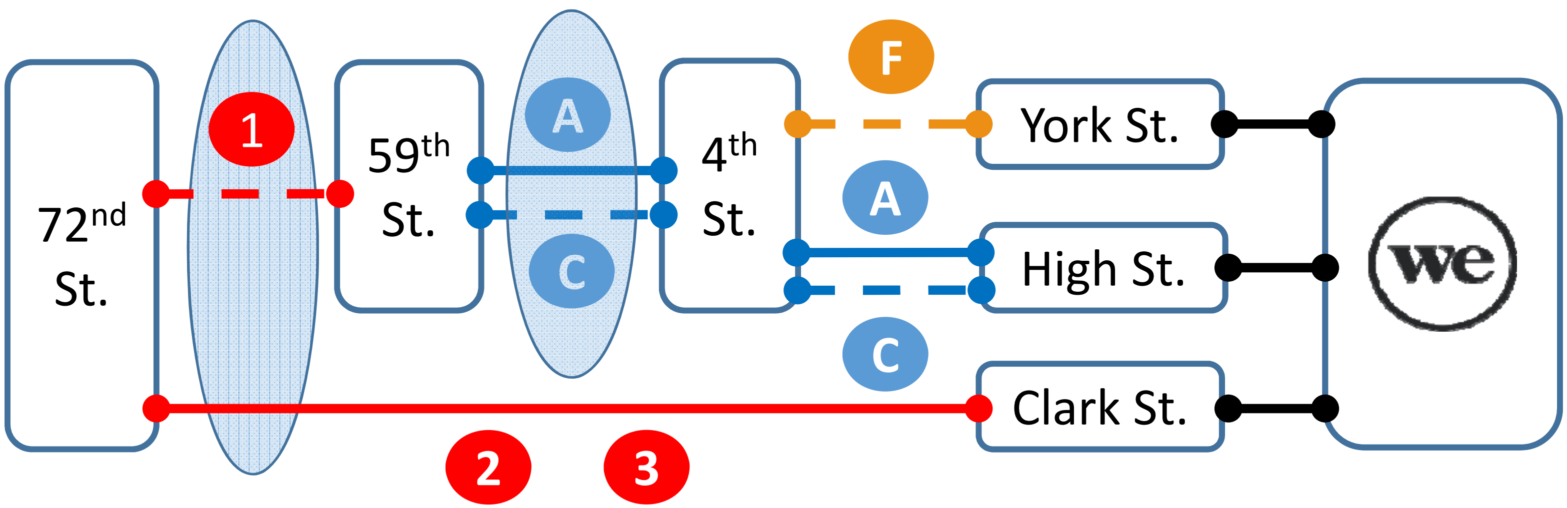
Greedy Algorithm for Choosing My Route

- At each node where there's a choice, take the first train that arrives.
- If two arrive simultaneously, take the express.
- Don't switch trains on the same line.



Greedy Algorithm Drawbacks

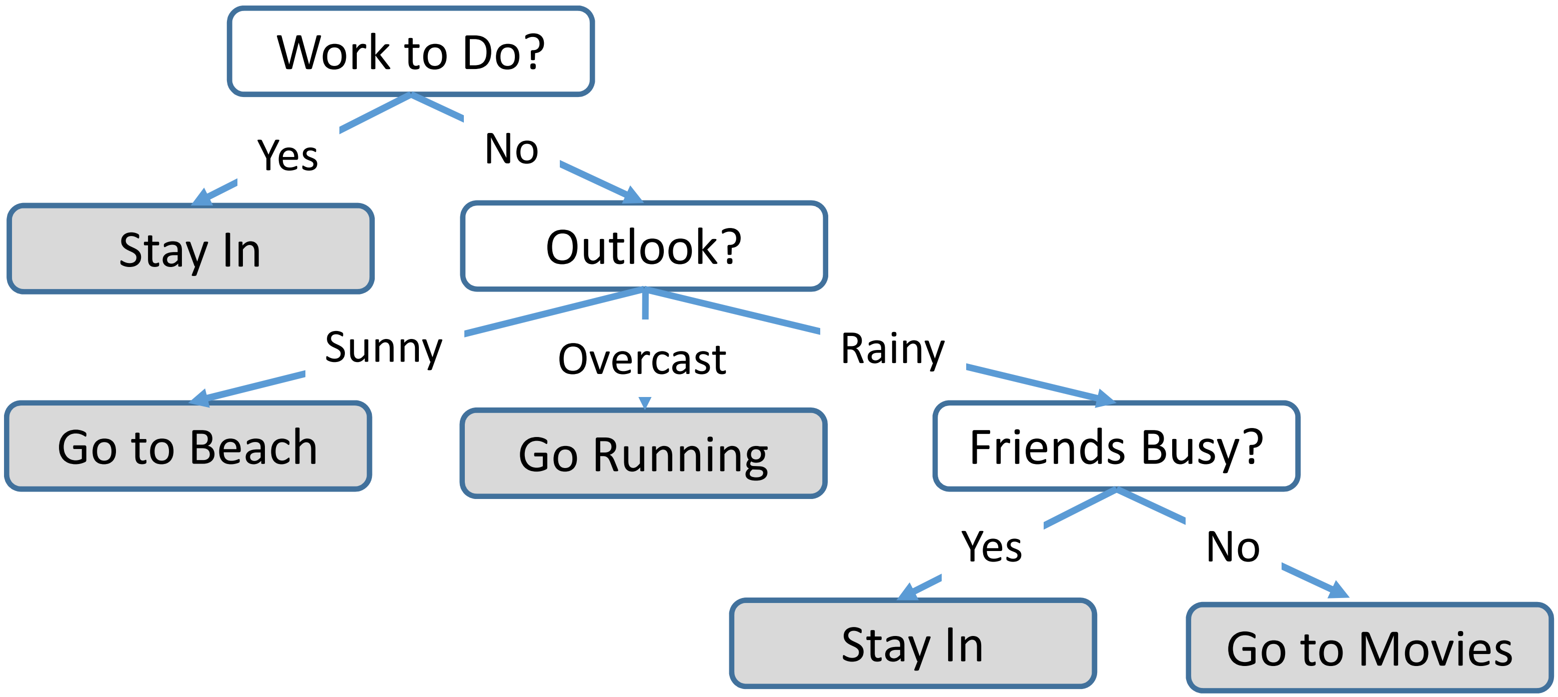
- I’ll never take the F although York St. is the closest of the three Dumbo stations to the office in WeWork.
- I’ll take a C even when the displays or my subway app tells me that an A will be arriving very shorty.



Measures of a Decision Tree

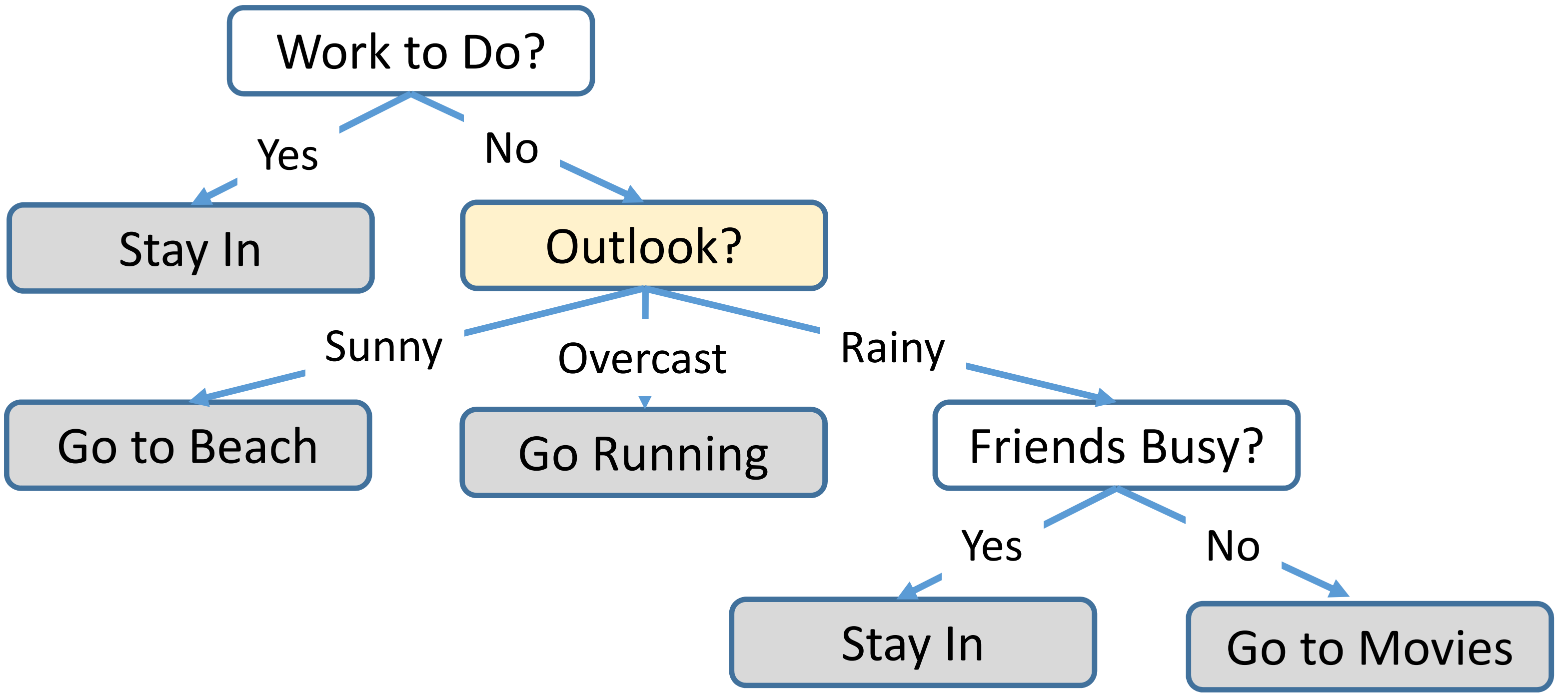
- The **depth** of a decision tree is the length of the longest path from the **root** of the tree to a **leaf**. The **size** of a decision tree is the number of nodes in the tree.

Sample Non-Binary Decision Tree



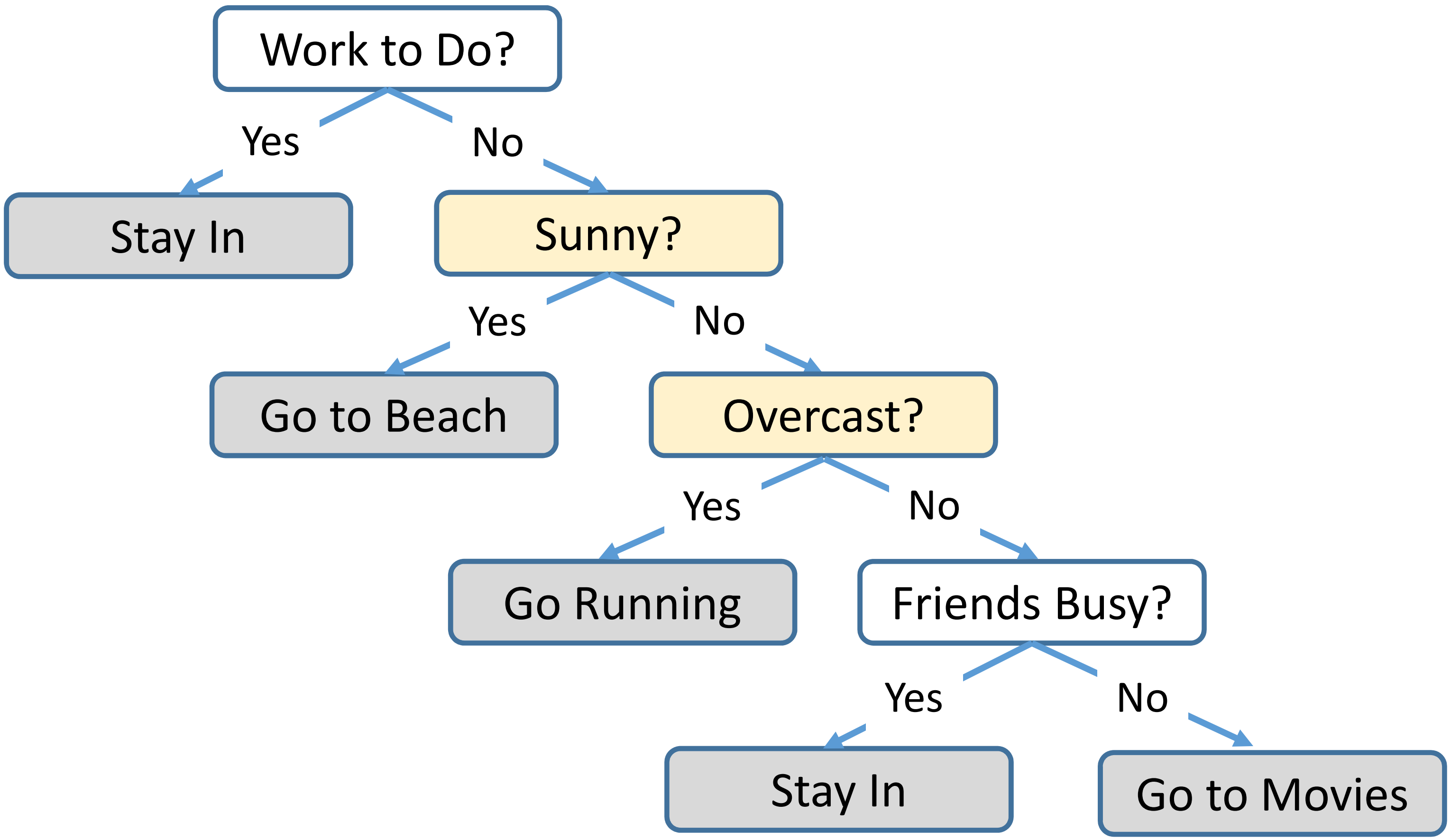
from Textbook

Non-Binary Decision Trees Can Always Be Converted to Binary Decision Trees

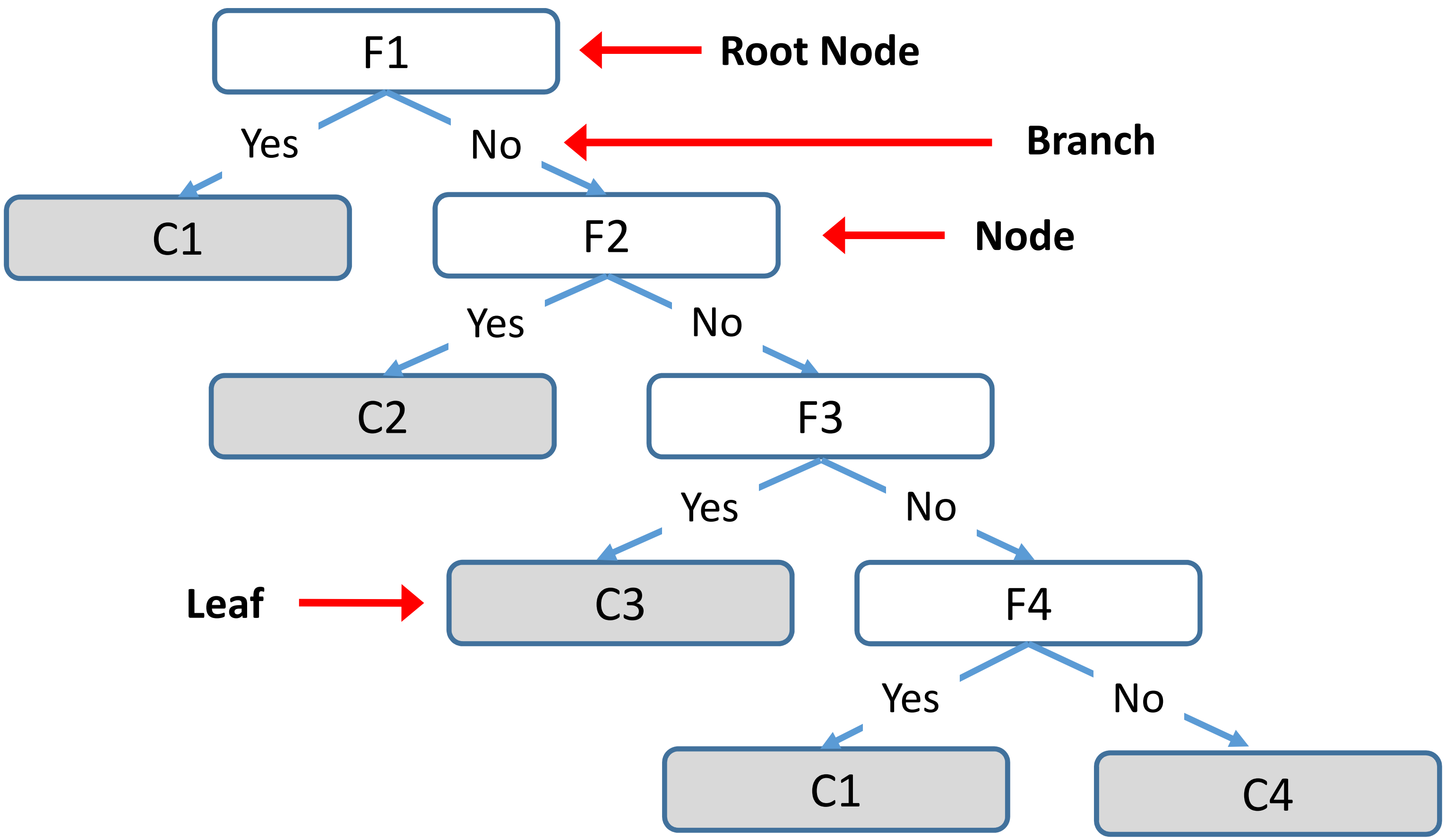


from Textbook

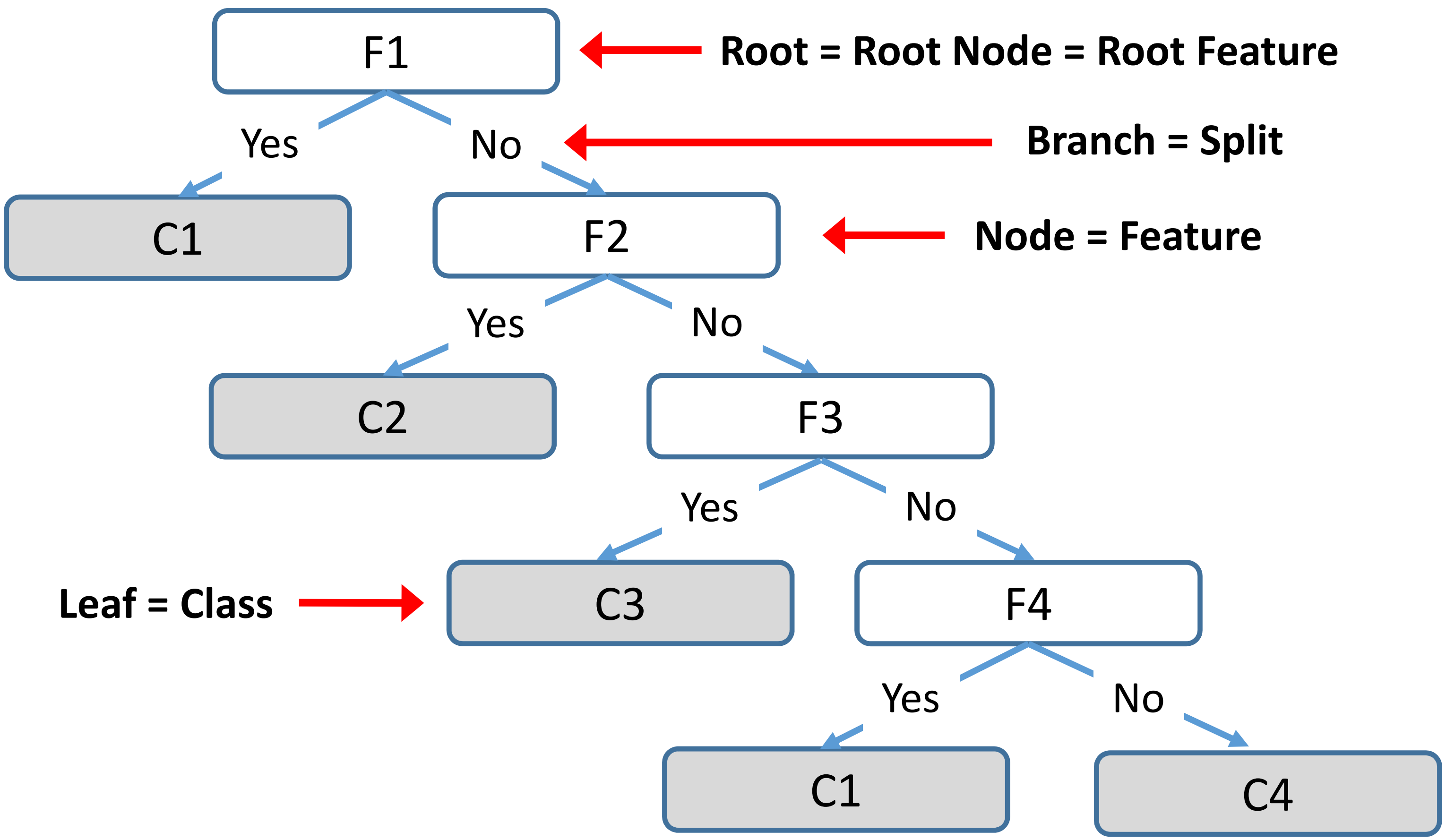
Sample Binary Decision Tree



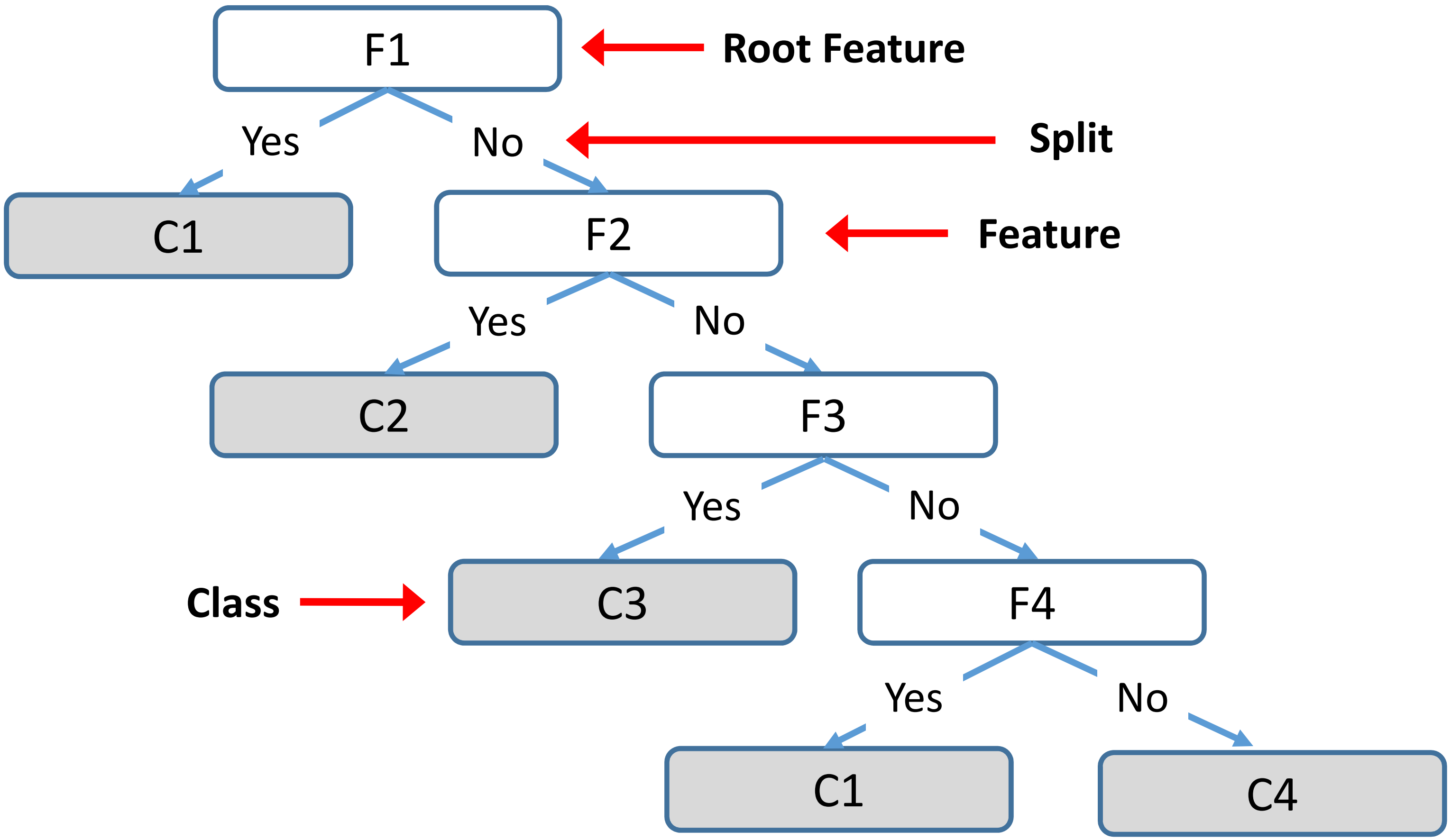
Sample Binary Decision Tree



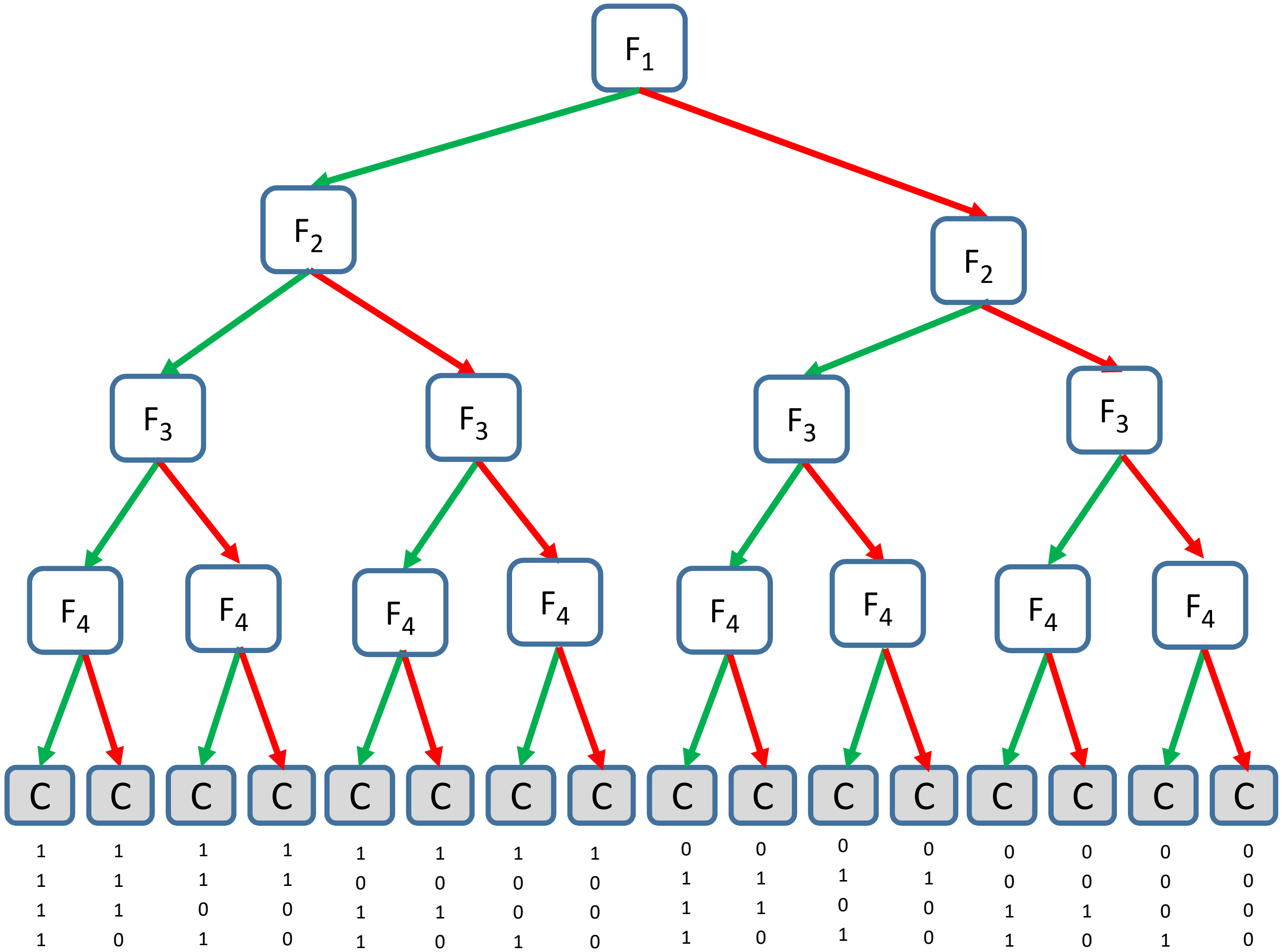
Sample Binary Decision Tree



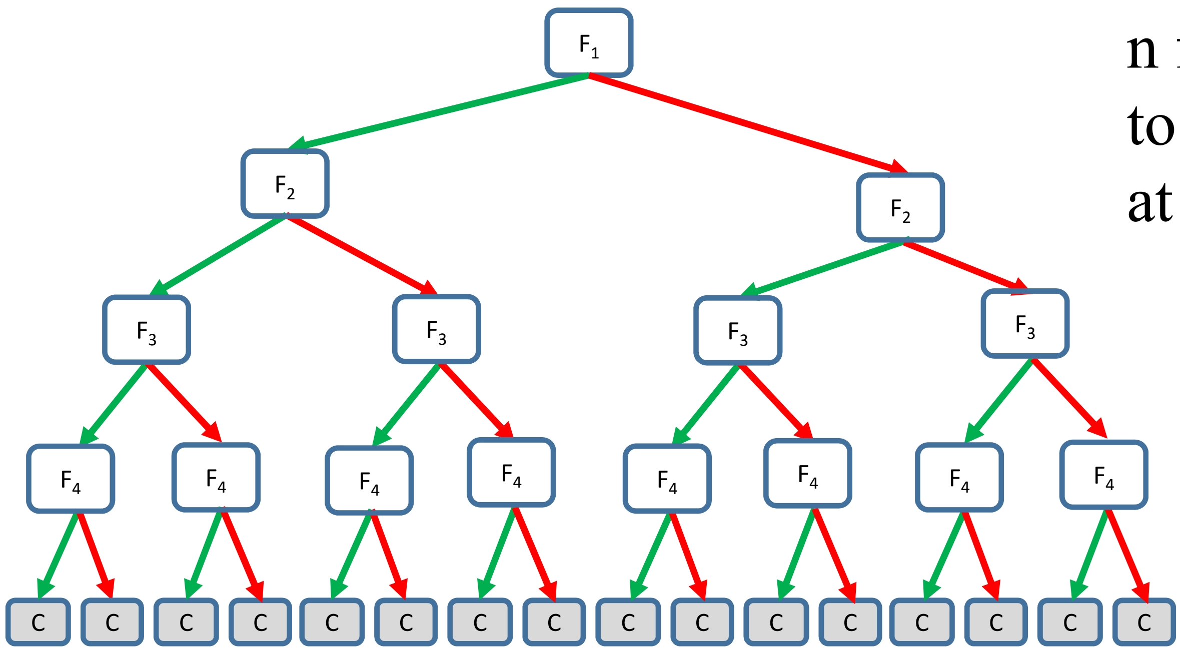
Sample Binary Decision Tree



Full Binary Decision Tree



Full Binary Decision Tree



n features can be used to classify samples into at most 2^n classes

Depth = d
Size = s = 2^{d+1}-1

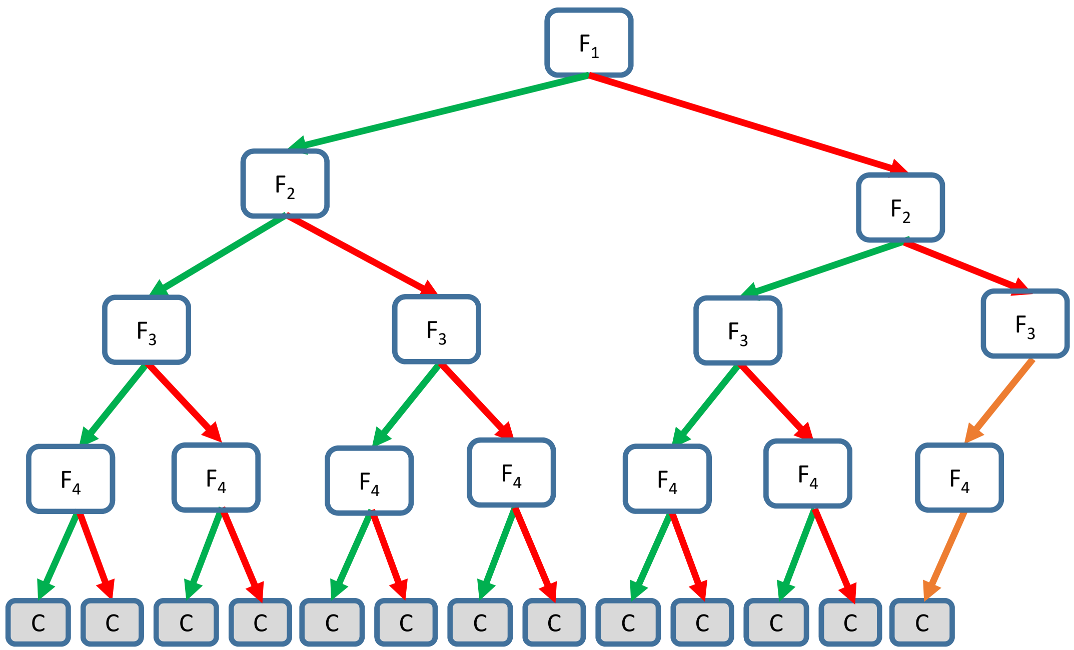
d =4
s= 2^4-1=31

Note that there is some ambiguity in the literature as to whether the depth of this tree is 4 or 5.

Overfitting is Often an Issue

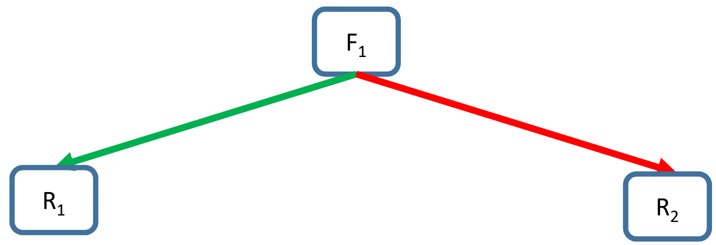
- n features can be used to classify samples into up to 2^n classes.
- There are many possible decision trees
  - For N binary features there are 2^{2^N} possible binary decision trees.
  - If one or more of the features is a rational number there an infinite number of possible decision trees and that feature(s) can be used repeatedly.
- **Pruning** one way to reduce overfitting.

Compete Binary Decision Tree



- Full at every level except possibly the last
- All nodes are as far left as possible

Decision Stump



- Only One Decision Node, i.e.,
  - Depth = 1
  - Size = 3

Finding the Optimal Binary Decision Tree

- There are many possible decision trees
- How do we choose the optimal one?
- We start at the **tree root** and split the tree on the feature that results in the largest **information gain (IG)**.
- If we don't **prune** the tree, we continue down the tree, splitting it at each node until at stopping criterion is satisfied.

Finding the Optimal Binary Decision Tree

- How do we choose the optimal one?
- We start at the tree root and **split the tree on the feature that results in the largest information gain (IG).**
- If we don't prune the tree, we continue down the tree, **splitting it on the feature that results in the largest IG** until a stopping criterion is satisfied.

Notation

- f is the feature used to perform the split
- the **parent node** is the node at which the split is made
- m is the number of **child nodes** of the parent node
- for a **binary tree**, $m = 2$
- D_p is the dataset of the **parent node**
- D_j is the dataset of the jth **child node**
- N_p is the number of samples in the parent node
- N_j is the number of samples in the jth child node
- I is the **impurity** measure of a dataset
- IG is the **information gain** at a particular split in the tree, it is the difference between the impurity of the parent node and the sum of the impurities of the child nodes

Information Gain

Information gain at a particular split in the tree is the difference between the impurity of the parent node and the weighted sum of the impurities of the child nodes.

$$IG\left(D_p, f\right) = I\left(D_p\right) - \sum_{j=1}^{m} \frac{N_j}{N_p} I\left(D_j\right)$$

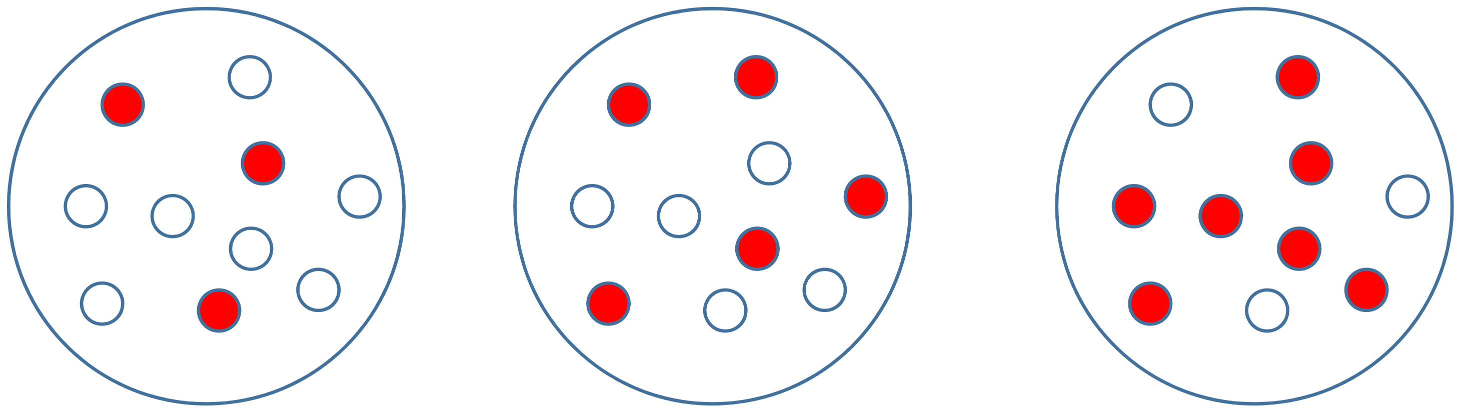
Impurity Measures

- I_G Gini impurity
- I_H Entropy
- I_E Classification Error

Impurity Measures

$n = 10, m = 2$

$p_1 = 0.3, p_2 = 0.7$ *$p_1 = 0.5, p_2 = 0.5$* *$p_1 = 0.7, p_2 = 0.3$*



Gini Impurity, I_G

The **Gini Impurity**, I_G, is a measure of how often a randomly chosen element from the set would be incorrectly labeled if were randomly labeled according to the distribution of labels in the subset.

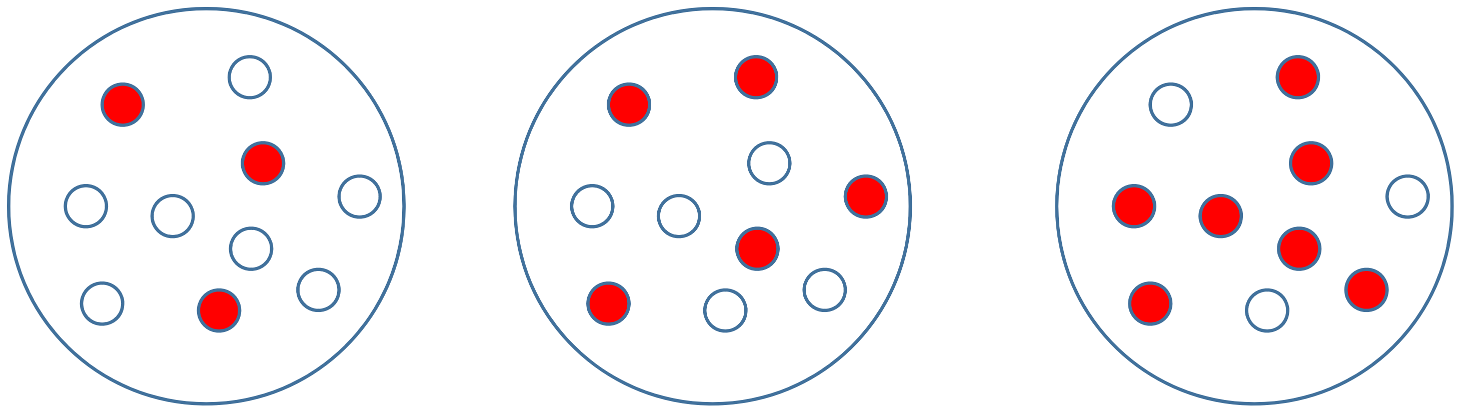
$$I_G(p) = \sum_{i=1}^{m} p_i(1-p_i) = \sum_{i=1}^{m}\left(p_i - p_i^2\right) = \sum_{i=1}^{m} p_i - \sum_{i=1}^{m} p_i^2 = 1 - \sum_{i=1}^{m} p_i^2 = \sum_{i \neq j} p_i p_j$$

Gini Impurity, I_G

$$I_G(p) = \sum_{i \neq j} p_i p_j$$

$n = 10, m = 2$

$p_1 = 0.3, p_2 = 0.7$ | $p_1 = 0.5, p_2 = 0.5$ | $p_1 = 0.7, p_2 = 0.3$



$I_G = 0.3*0.7 + 0.3*0.7 = 0.42$

$I_G = 0.5*0.5 + 0.5*0.5 = 0.5$

$I_G = 0.7*0.3 + 0.7*0.3 = 0.42$

Entropy, I_H

The **entropy**, I_H, is a measure of information content and of disorder.

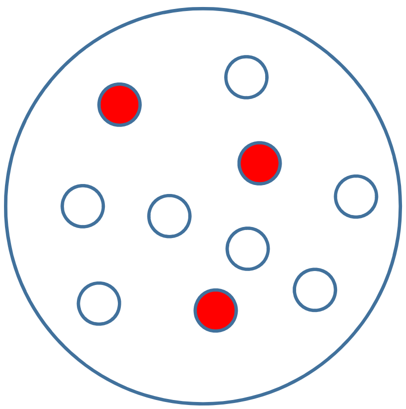
$$I_H(p) = -\sum_{i=1}^{m} p_i \log_2 p_i$$

Entropy, I_H

$$I_H(p) = -\sum_{i=1}^{m} p_i \log_2 p_i$$

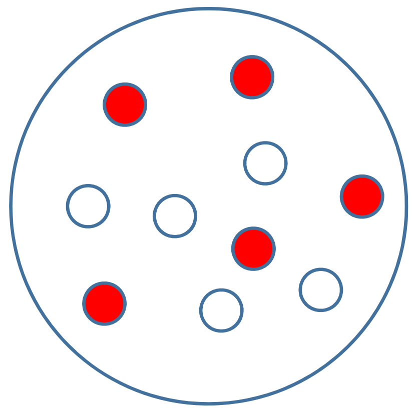
$n = 10, m = 2$

$p_1 = 0.3, p_2 = 0.7$



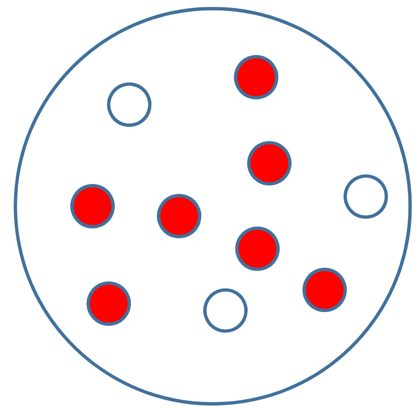
$I_H = -0.3*\log_2 0.3$
$-0.7*\log_2 0.7$
$= 0.8812$

$p_1 = 0.5, p_2 = 0.5$



$I_H = -0.5*\log_2 0.5$
$-0.5*\log_2 0.5$
$= 1.$

$p_1 = 0.7, p_2 = 0.3$



$I_H = -0.3*\log_2 0.3$
$-0.7*\log_2 0.7$
$= 0.8812$

Classification Error, I_E

The **classification error**, I_E, is a simple measure of order.

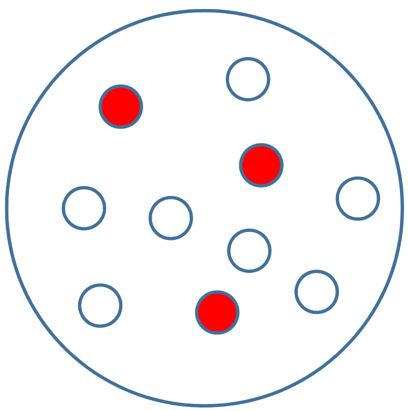
$$I_E(p) = 1 - \max\{p_i\}$$

Classification Error, I_E

$$I_E(p) = 1 - \max\{p_i\}$$

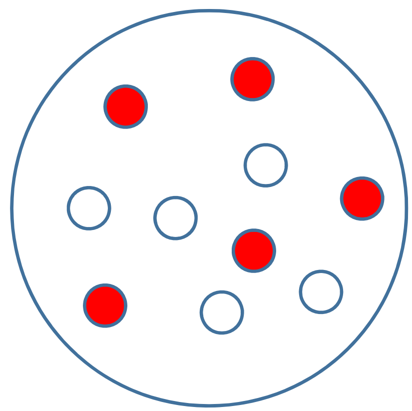
$n = 10, m = 2$

$p_1 = 0.3, p_2 = 0.7$



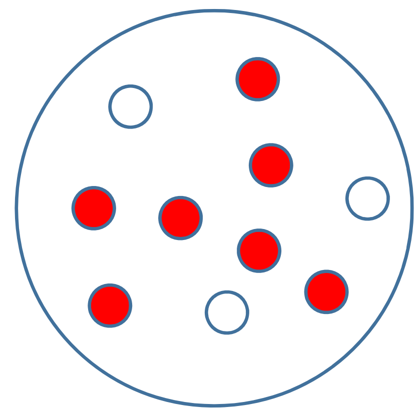
$I_E = 1 - max\{0.3, 0.7\}$
$= 0.3$

$p_1 = 0.5, p_2 = 0.5$



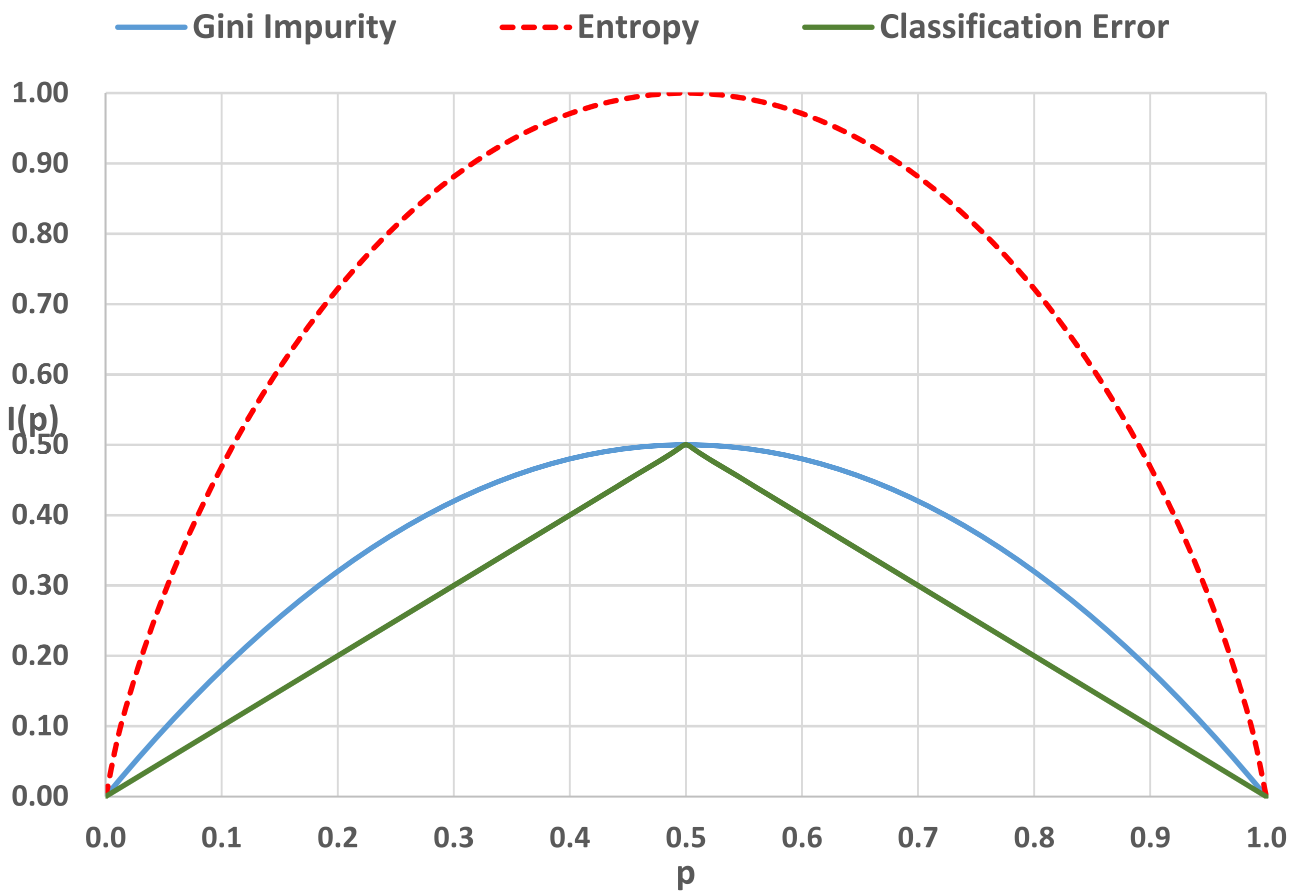
$I_E = 1 - max\{0.5, 0.5\}$
$= 0.5$

$p_1 = 0.7, p_2 = 0.3$



$I_E = 1 - max\{0.7, 0.3\}$
$= 0.3$

Impurity Measures



Impurity Measures

Overfitting is Often a Problem

- There are many possible decision trees
  - For N binary features there are 2^{2^N} possible binary decision trees.
  - If one of the features is a rational number there at an infinite number of possible decision trees.

Typical Stopping Criteria

- Only leaf nodes remain
- No further features remain to be examined
- Additional splitting fails to reduce the impurity by a specified amount
- A specified maximum tree depth has been reached
- A specified number of leaf nodes have been generated

Advantages of Decision Trees

- Efficient and Scalable Learning Algorithm
- Handles both Discrete and Continuous Features
- Robust against Monotonic Input Transformations
- Robust against Outliers
- Automatically Ignores Irrelevant Features; No Need for Feature Selection
- Results are Usually Interpretable

Scikit-learn Decision Tree Classifier Class

class sklearn.tree.DecisionTreeClassifier (…)

Parameter	Default		Parameter	Defaults
criterion	‘gini’		max_features	None
splitter	‘best’		random_state	None
max_depth	None		max_leaf_nodes	None
min_samples_split	2		min_impurity decrease	0.
min_samples_leaf	1		class_weight	None
min_weight_fraction_leaf	0.		presort	False

Scikit-learn Decision Tree Classifier Attributes

Attribute	Description
classes_	The class labels
feature_importances_	The feature importances
max_features_	The inferred value of max_features
n_classes_	The number of classes when fit is performed
n_features_	The number of features when fit is performed
n_outputs	The number of outputs when fit is performed
tree_	The underlying tree object

Scikit-learn Decision Tree Classifier Methods

Method	Description
apply(X[, check_input])	Returns the index of the leaf that each sample is predicted as.
decision_path(X[, check_input])	Return the decision path in the tree.
fit(X, y[, sample_weight, check_input, …})	Build a decision tree classifier from the training set (X, y).
get_params([deep])	Get parameters for this estimator.
predict(X[, check_input])	Predict class or regression value for X.
predict_log_proba(X)	Predict class log-probabilities of the input samples X.
predict_proba(X[, check_input])	Predict class probabilities of the input samples X.
score(X, y[, sample_weight])	Returns the mean accuracy on the given test data and labels.
set_params(**params)	Set the parameters of this estimator.