Data Science Engineering Methods and Tools

Lecture 2

Northeastern University

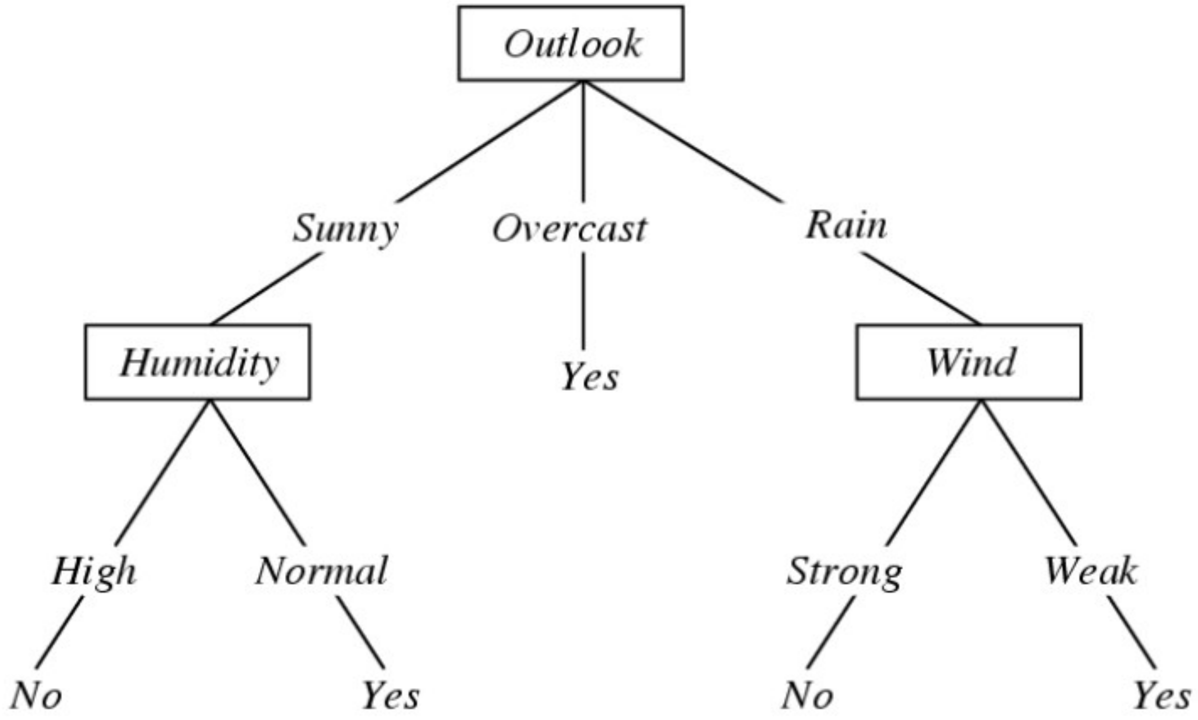
College of Engineering

INFO 6105 – Spring 2024

Abdolreza Mosaddegh

Decision Tree

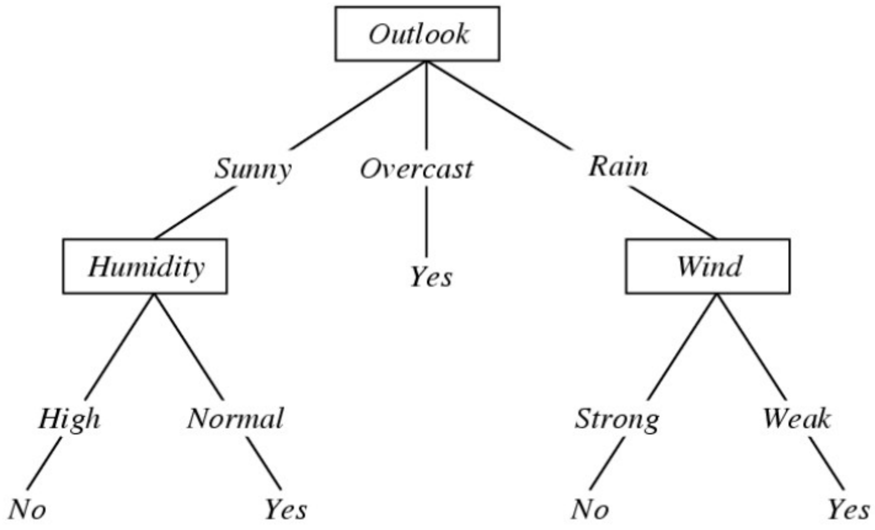
- A decision tree is a hierarchical classification model that uses a tree structure and can be used to support decisions
- Each internal node represent a test on one attribute (feature)
- Each branch from a node represents a possible outcome of the test
- Each leaf node represents a class label



Is weather condition suitable for playing tennis?

Decision Tree - Supervised Learning

- Columns denote features
- Rows denote labeled instances
- Class label denotes whether a tennis game was played

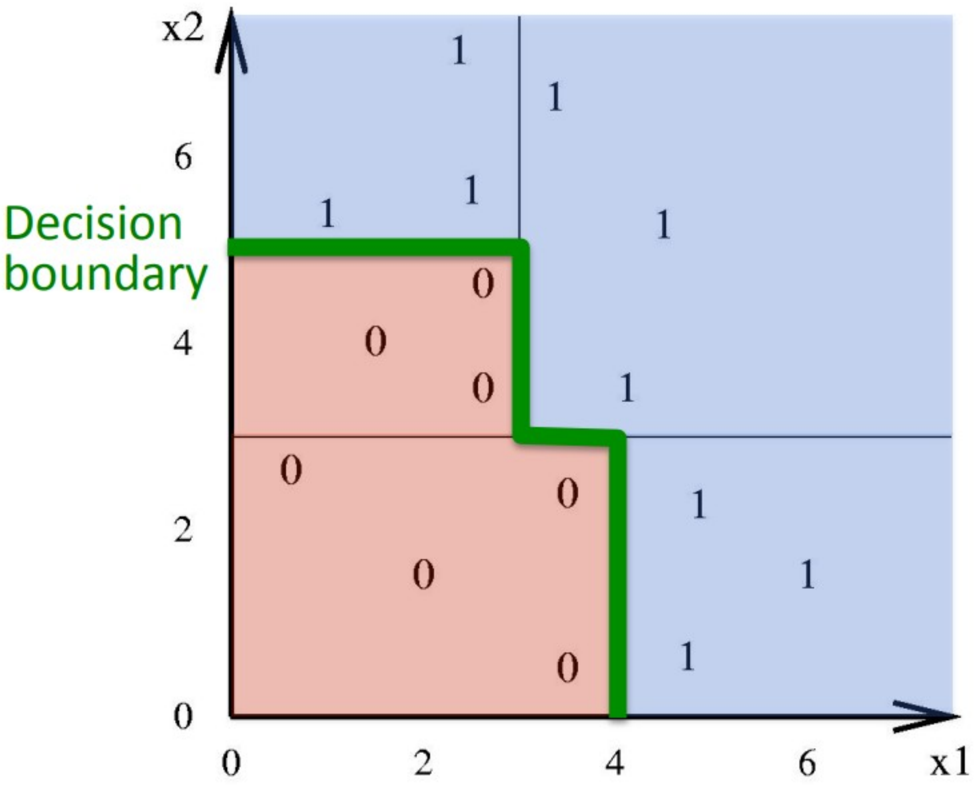


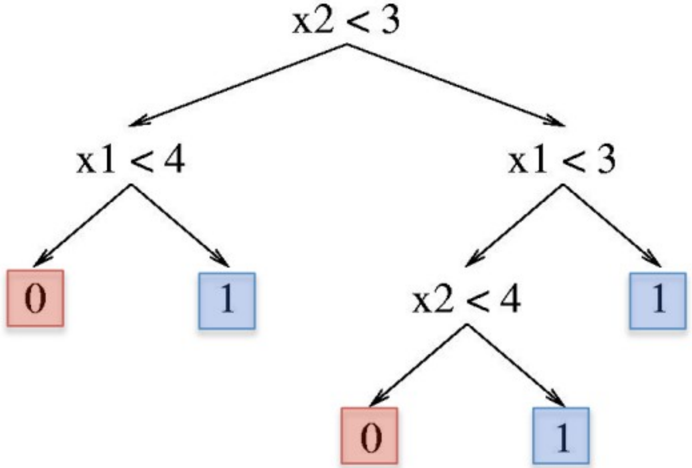
Features				Label
Outlook	Temperature	Humidity	Wind	Play Tennis
Sunny	Hot	High	Weak	No
Sunny	Hot	High	Strong	No
Overcast	Hot	High	Weak	Yes
Rain	Mild	High	Weak	Yes
Rain	Cool	Normal	Weak	Yes
Rain	Cool	Normal	Strong	No
Overcast	Cool	Normal	Strong	Yes
Sunny	Mild	High	Weak	No
Sunny	Cool	Normal	Weak	Yes
Rain	Mild	Normal	Weak	Yes
Sunny	Mild	Normal	Strong	Yes
Overcast	Mild	High	Strong	Yes
Overcast	Hot	Normal	Weak	Yes
Rain	Mild	High	Strong	No

Prediction: Rainy and Strong Wind > Play Tennis or Not?

Decision Tree as a Non-linear Classifier

Decision trees divide the feature space into axis-parallel rectangles

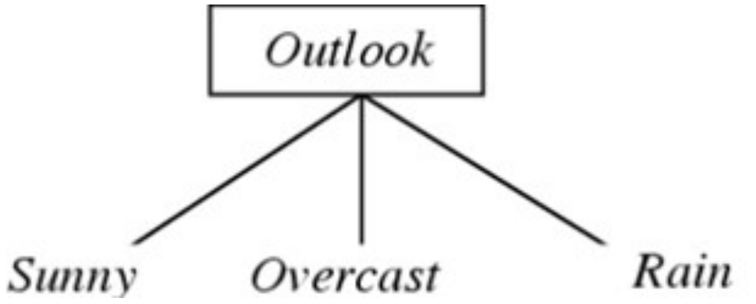




Splitting Categorical Variables in D-Tree

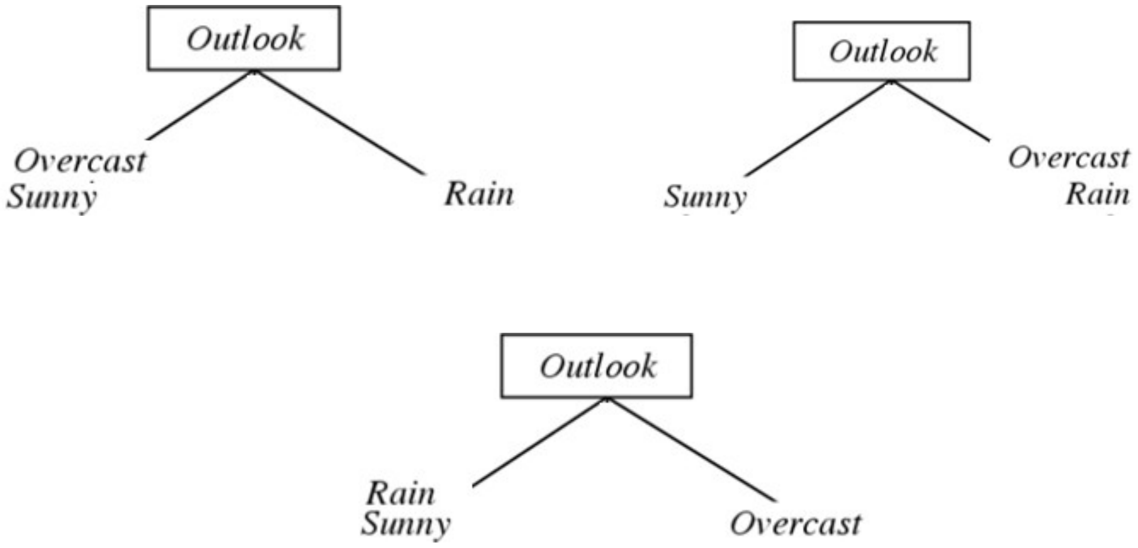
Multi-way split:

Use as many partitions as distinct values.



Binary split:

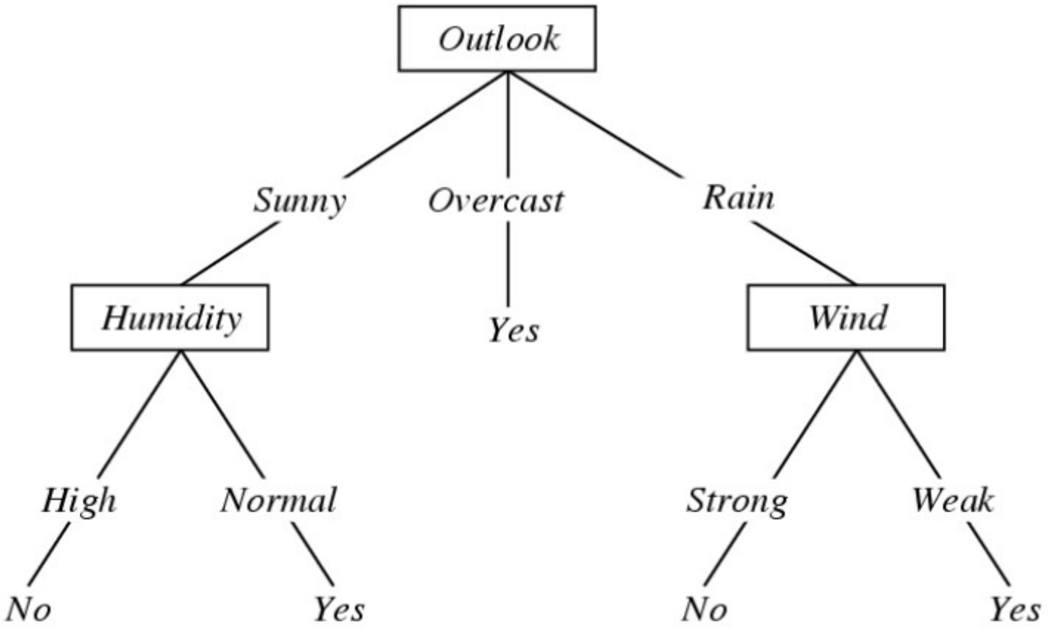
- Divides values into two subsets
- Need to find optimal partitioning.
- Preserve order property among attribute values

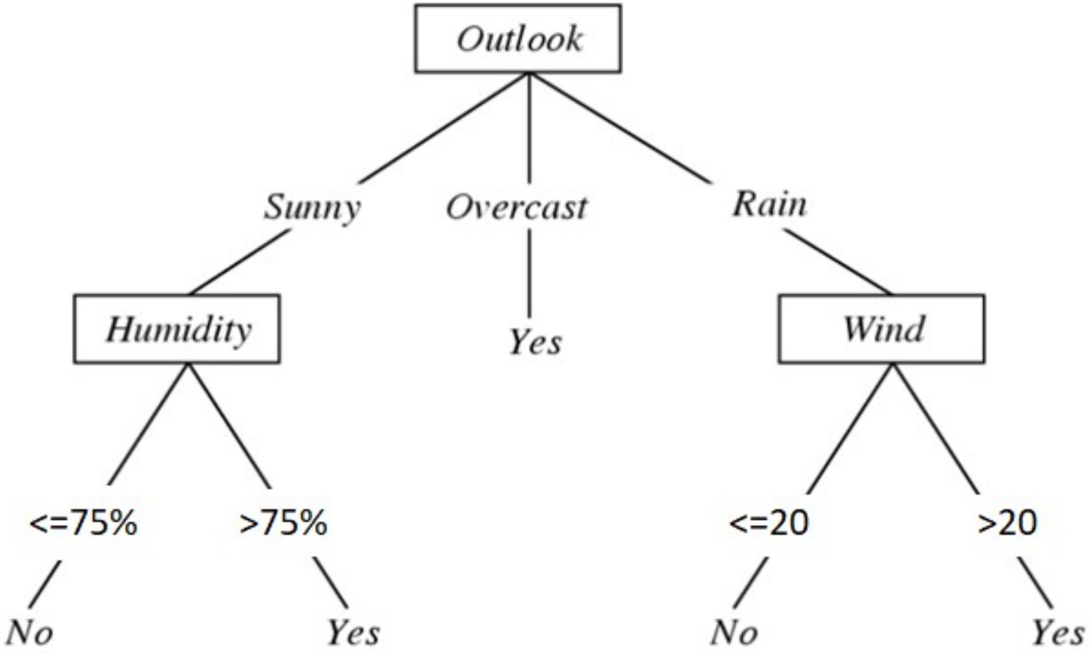


By considering performance issues, most common algorithms prefer binary split since there are too many possibilities for the next split in the multiway D-Trees

Discretizing Continuous Variables in D-Tree

- If features are continuous, internal nodes can test the value of a feature against a threshold.
- Heuristic approaches can be used for finding optimal thresholds.



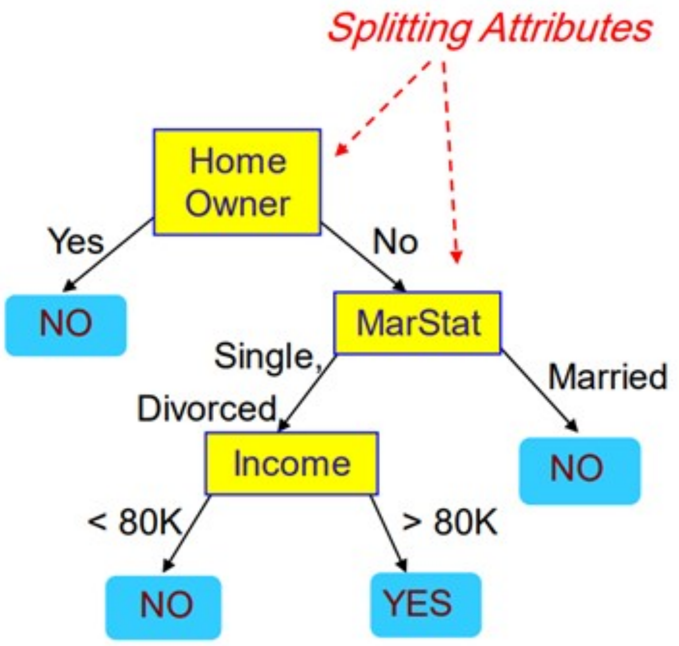


Making a Random D-Tree from Data

class

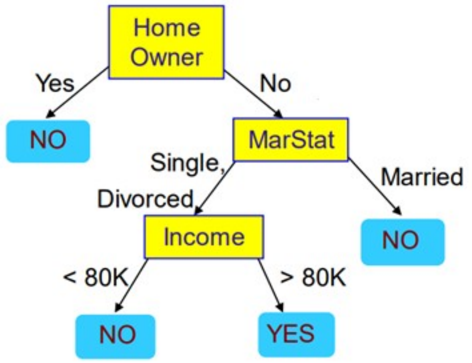
ID	Home Owner	Marital Status	Annual Income	Defaulted Borrower
1	Yes	Single	125K	No
2	No	Married	100K	No
3	No	Single	70K	No
4	Yes	Married	120K	No
5	No	Divorced	95K	Yes
6	No	Married	60K	No
7	Yes	Divorced	220K	No
8	No	Single	85K	Yes
9	No	Married	75K	No
10	No	Single	90K	Yes

Training Data



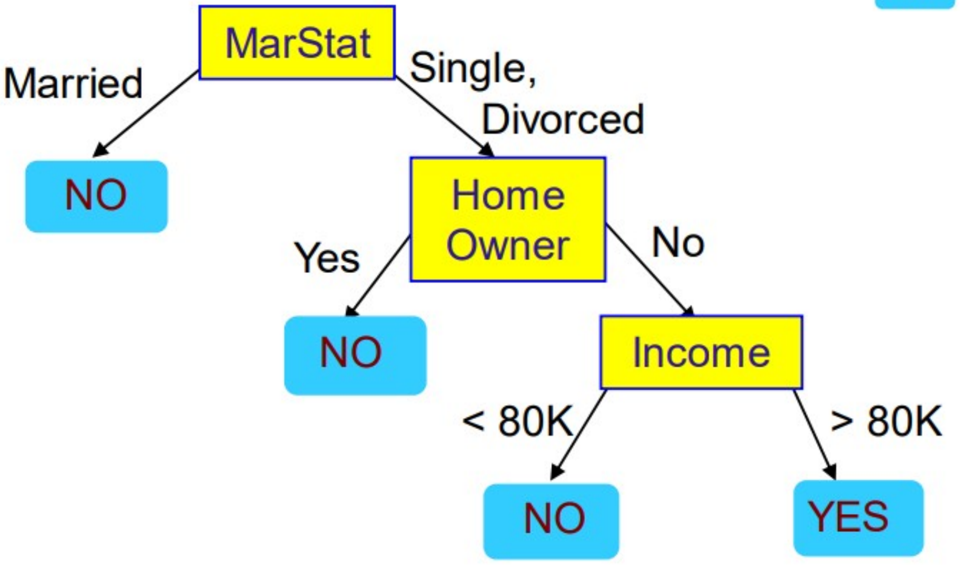
Model: Decision Tree

Another D-Tree with the Same



class

ID	Home Owner	Marital Status	Annual Income	Defaulted Borrower
1	Yes	Single	125K	No
2	No	Married	100K	No
3	No	Single	70K	No
4	Yes	Married	120K	No
5	No	Divorced	95K	Yes
6	No	Married	60K	No
7	Yes	Divorced	220K	No
8	No	Single	85K	Yes
9	No	Married	75K	No
10	No	Single	90K	Yes



There could be more than one tree that fits the same data!

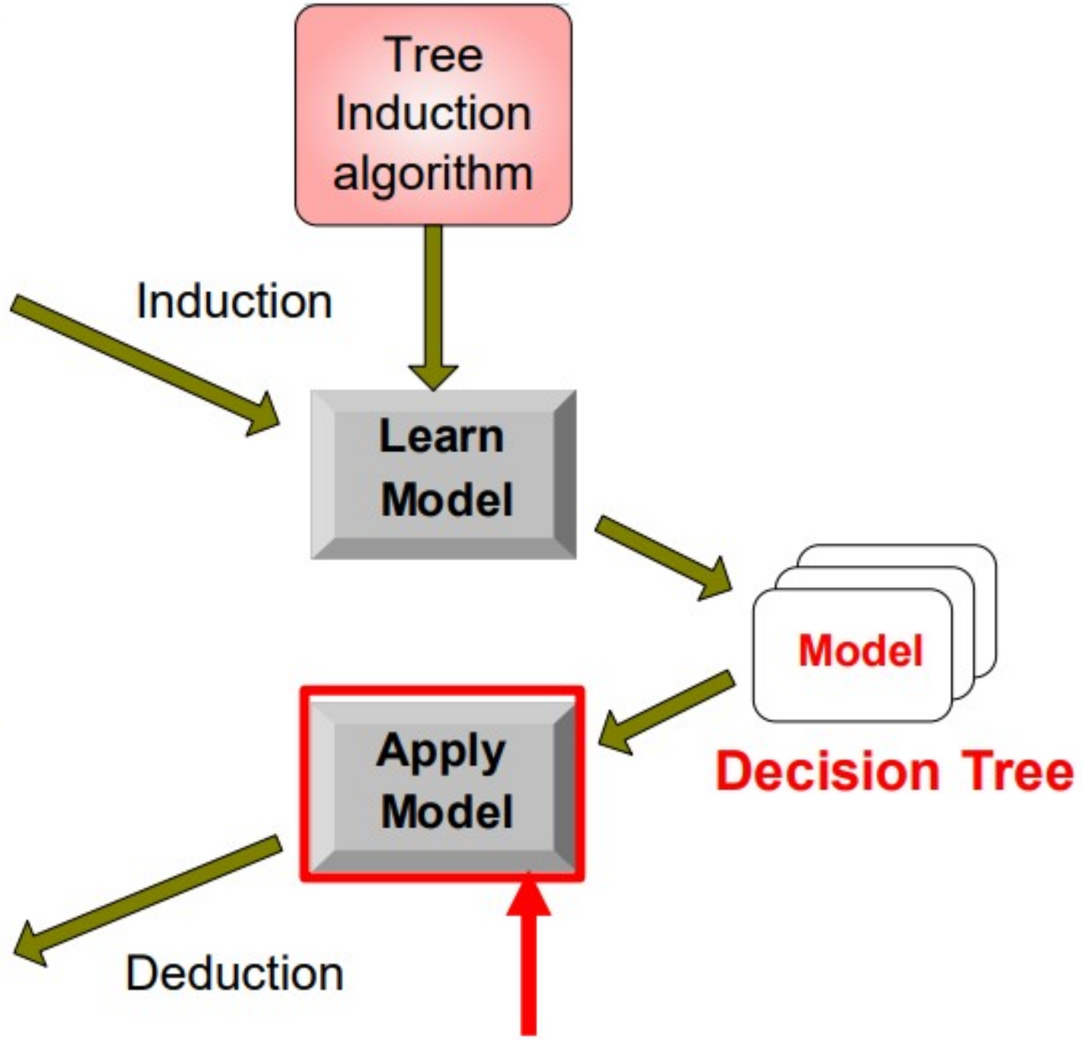
Which D-Tree is better?

Tid	Attrib1	Attrib2	Attrib3	Class
1	Yes	Single	125K	No
2	No	Married	100K	No
3	No	Single	70K	No
4	Yes	Married	120K	No
5	No	Divorced	95K	Yes
6	No	Married	60K	No
7	Yes	Divorced	220K	No
8	No	Single	85K	Yes
9	No	Married	75K	No
10	No	Single	90K	Yes

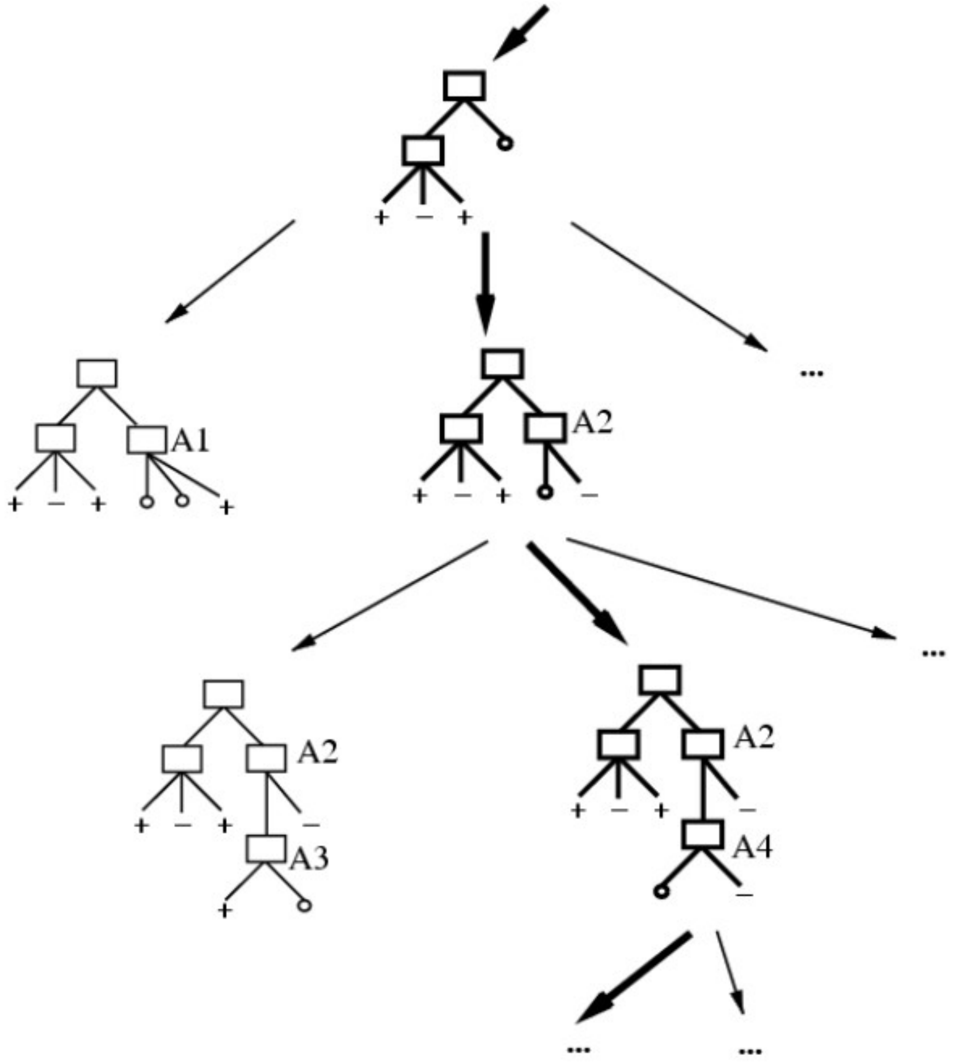
Training Set

Tid	Attrib1	Attrib2	Attrib3	Class
11	No	Single	55K	?
12	No	Married	80K	?
13	Yes	Divorced	110K	?
14	No	Single	95K	?
15	No	Divorced	67K	?

Test Set



Problem Statement: Which Tree?



Occam’s razor:

- In philosophy, Occam's razor is a problem-solving principle that recommends searching for explanations constructed with the smallest possible set of elements.

- Occam's razor has gained strong empirical support in statistics and machine learning.

- Based on Occam’s razor principle, the smallest decision tree that correctly classifies training examples is the best.

Smallest Tree

Learning the simplest (smallest) decision tree is an NP-complete problem [Hyafil & Rivest '76]

• Resort to a greedy heuristic:

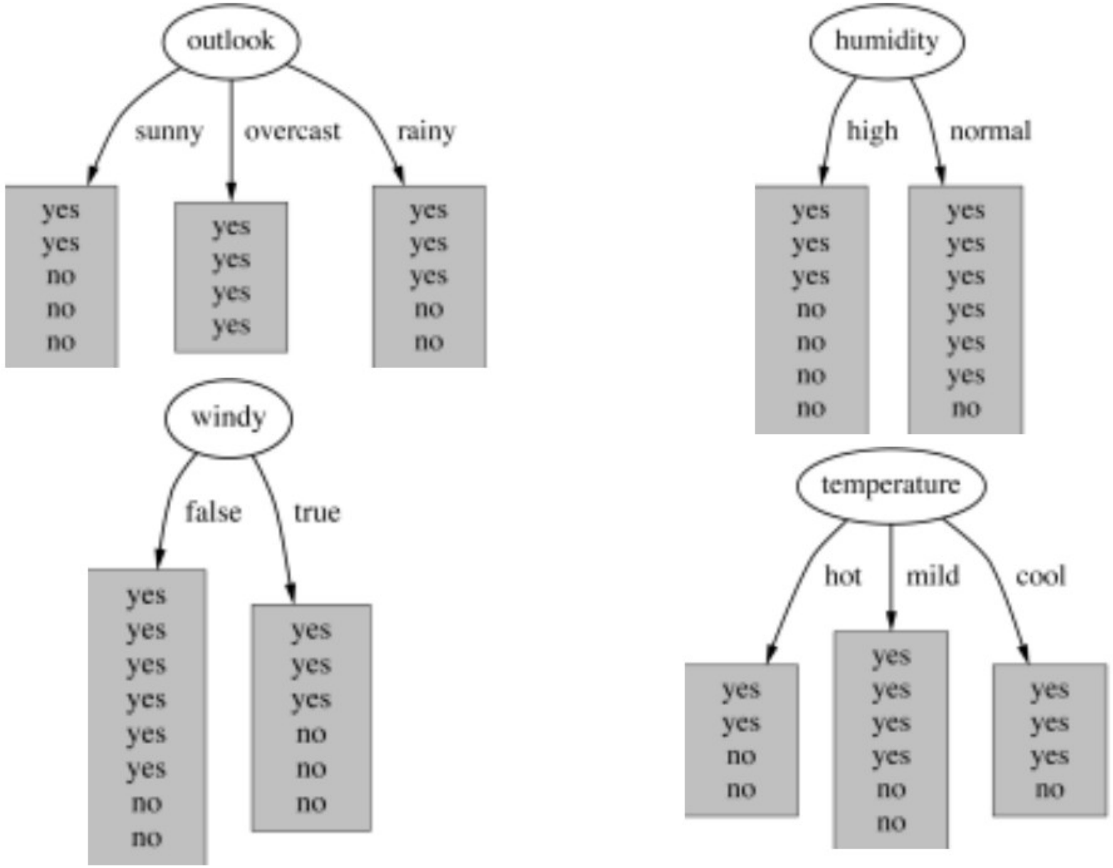
- Start from empty decision tree
- Split on next best attribute (feature)
- Recurse

- Greedy algorithms make a locally optimal choice at each stage that approximate a globally optimal solution in a reasonable amount of time.
- Sometimes, a greedy strategy does not produce an optimal solution, but just can yield locally optimal outcomes.

Which Attribute is the Best?

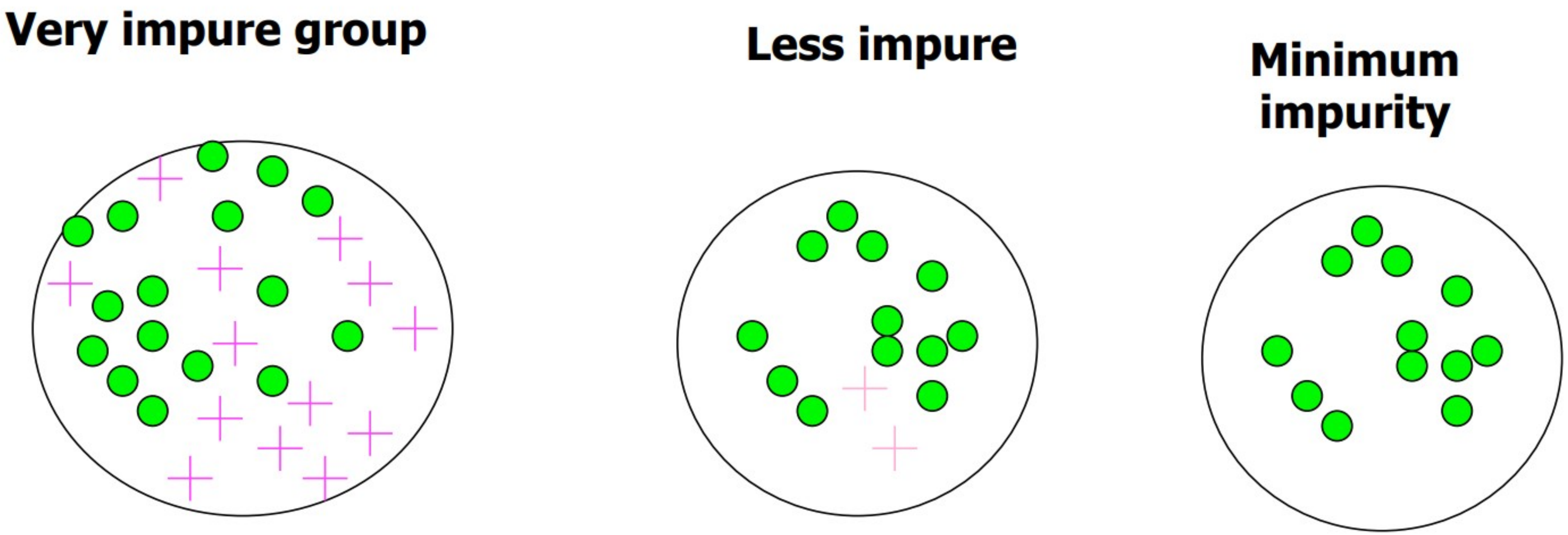
Heuristic:
choose the attribute that produces best classification (the “purest” nodes)

Outlook	Temperature	Humidity	Wind	Play Tennis
Features				*Label*
Sunny	Hot	High	Weak	No
Sunny	Hot	High	Strong	No
Overcast	Hot	High	Weak	Yes
Rain	Mild	High	Weak	Yes
Rain	Cool	Normal	Weak	Yes
Rain	Cool	Normal	Strong	No
Overcast	Cool	Normal	Strong	Yes
Sunny	Mild	High	Weak	No
Sunny	Cool	Normal	Weak	Yes
Rain	Mild	Normal	Weak	Yes
Sunny	Mild	Normal	Strong	Yes
Overcast	Mild	High	Strong	Yes
Overcast	Hot	Normal	Weak	Yes
Rain	Mild	High	Strong	No



Impurity as Loss Function

Impurity is presence of more than one class in a subset of data



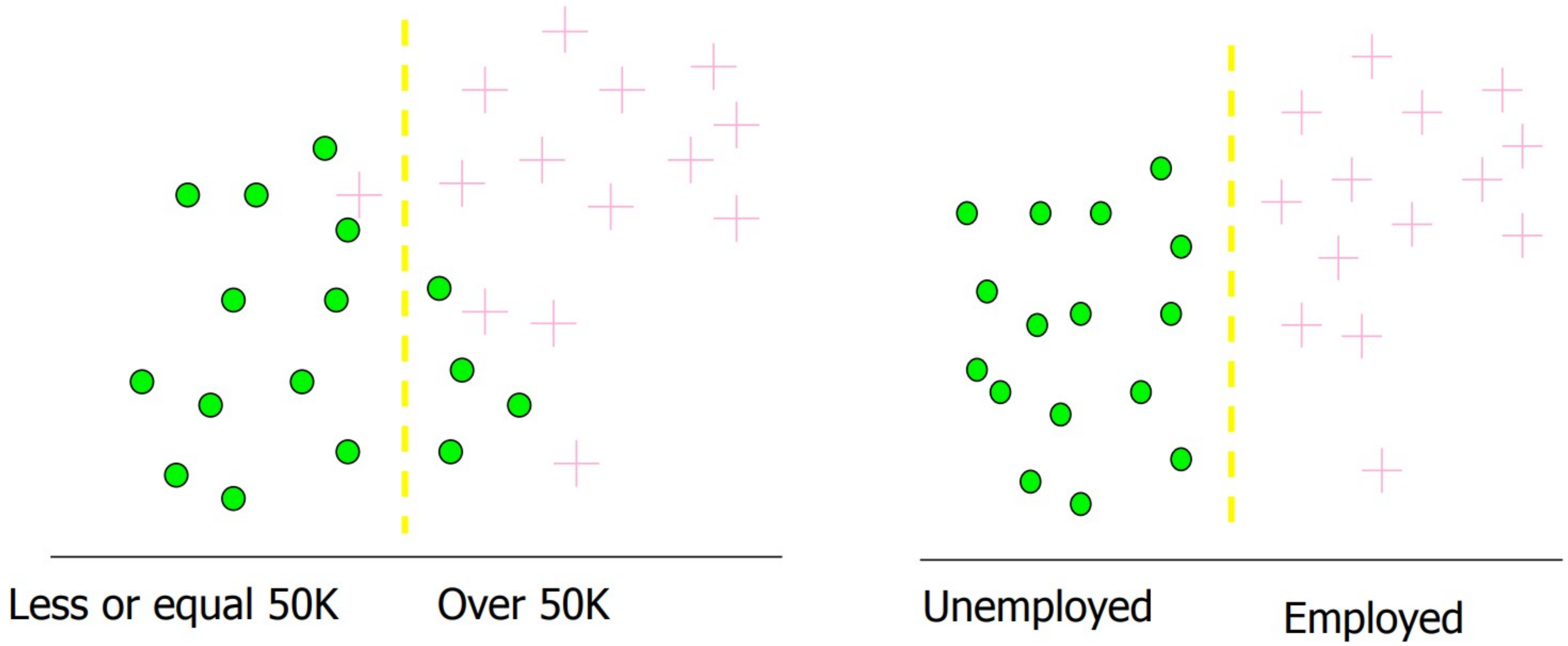
Impurity measures are used in Decision Trees as a loss function.
Learning process tries to find lowest impurity as possible

Gain

Which spilt decreases impurity more?

Gain = Impurity before spilt – Impurity after spilt

Bigger gain indicates more decrease in impurity after split



Splitting by Gain

1- Compute impurity measure (P) before splitting

2- Compute impurity measure (M) after splitting by any attribute

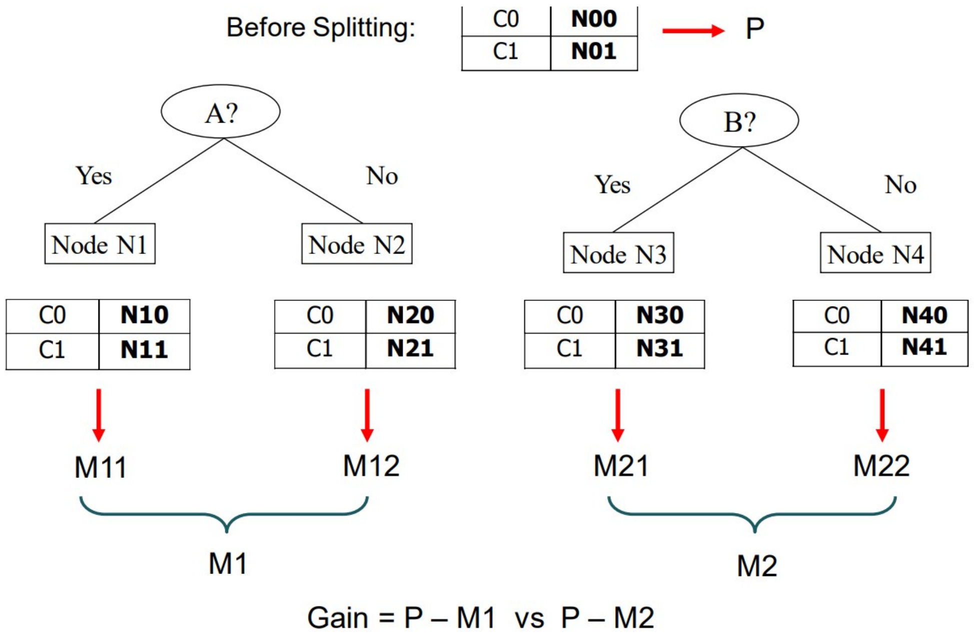
- Compute impurity measure of each child node
- Compute the average impurity of the children

3- Calculate gain as follows:

Gain = P – M

4- Choose the attribute test condition that produces the highest gain

This method needs a measure of impurity



Impurity Measures

For a given node t (p(j | t) is the relative frequency of class j at node t)

- Gini Index

$$GINI(t) = 1 - \sum_{j} [p(j \mid t)]^2$$

- Entropy

$$Entropy(t) = -\sum_{j} p(j \mid t) \log p(j \mid t)$$

- Misclassification error

$$Error(t) = 1 - \max_{i} P(i \mid t)$$

Gini Index

Gini Index for a given node t :

p(j | t) is the relative frequency of class j at node t

$$GINI(t) = 1 - \sum_j [p(j \mid t)]^2$$

The Gini coefficient measures the inequality among the values of a frequency distribution

– Maximum (1 - 1/Number_of_classes) when records are equally distributed among all classes, implying least interesting information

– Minimum (0) when all records belong to one class, implying most interesting information

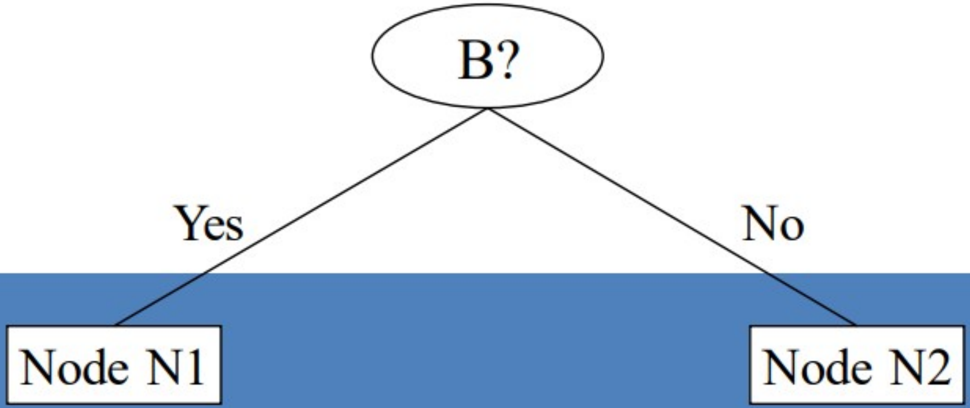
When a node p is split into k partitions :

$$GINI_{split} = \sum_{i=1}^{k} \frac{n_i}{n} GINI(i)$$

Selection using Information Gain

		Parent
Play Tennis	C1	6
Not Play Tennis	C2	6
	Gini = 0.500	

Gini = 1- (6/12)



Gini(N1)
$= 1 – (5/6)^2 – (1/6)^2$
$= 0.278$

Gini(N2)
$= 1 – (2/6)^2 – (4/6)^2$
$= 0.444$

	N1	N2
C1	5	2
C2	1	4
Gini=0.361		

Gini(Children)
$= 6/12 * 0.278 + 6/12 * 0.444 = .361$

Gain = 0.500 – 0.361 = 0.139

Classification Error

Classification error at a node t: $Error(t) = 1 - \max_i P(i \mid t)$

Measures misclassification error made by a node.

– Maximum (1 - 1/Number_of_classes) when records are equally distributed among all classes, implying least interesting information

– Minimum (0) when all records belong to one class, implying most interesting information

$$Error(t) = 1 - \max_i P(i \mid t)$$

C1	**0**
C2	**6**

P(C1) = 0/6 = 0   P(C2) = 6/6 = 1

Error = 1 – max (0, 1) = 1 – 1 = 0

C1	**1**
C2	**5**

P(C1) = 1/6   P(C2) = 5/6

Error = 1 – max (1/6, 5/6) = 1 – 5/6 = 1/6

C1	**2**
C2	**4**

P(C1) = 2/6   P(C2) = 4/6

Error = 1 – max (2/6, 4/6) = 1 – 4/6 = 1/3

Entropy

Entropy at a given node t: $$Entropy(t) = -\sum_j p(j \mid t) \log p(j \mid t)$$

Entropy is commonly associated with a state of disorder or uncertainty

– Maximum (Log Number_of_classes) when records are equally distributed among all classes implying least information

– Minimum (0.0) when all records belong to one class, implying most information

Entropy based computations are quite similar to the GINI index computations

$$Entropy(t) = -\sum_j p(j \mid t) \log p(j \mid t)$$

C1	**0**
C2	**6**

P(C1) = 0/6 = 0   P(C2) = 6/6 = 1

Entropy = – 0 log 0 – 1 log 1 = – 0 – 0 = 0

C1	**1**
C2	**5**

P(C1) = 1/6   P(C2) = 5/6

Entropy = – (1/6) $\log_2$ (1/6) – (5/6) $\log_2$ (1/6) = 0.65

C1	**2**
C2	**4**

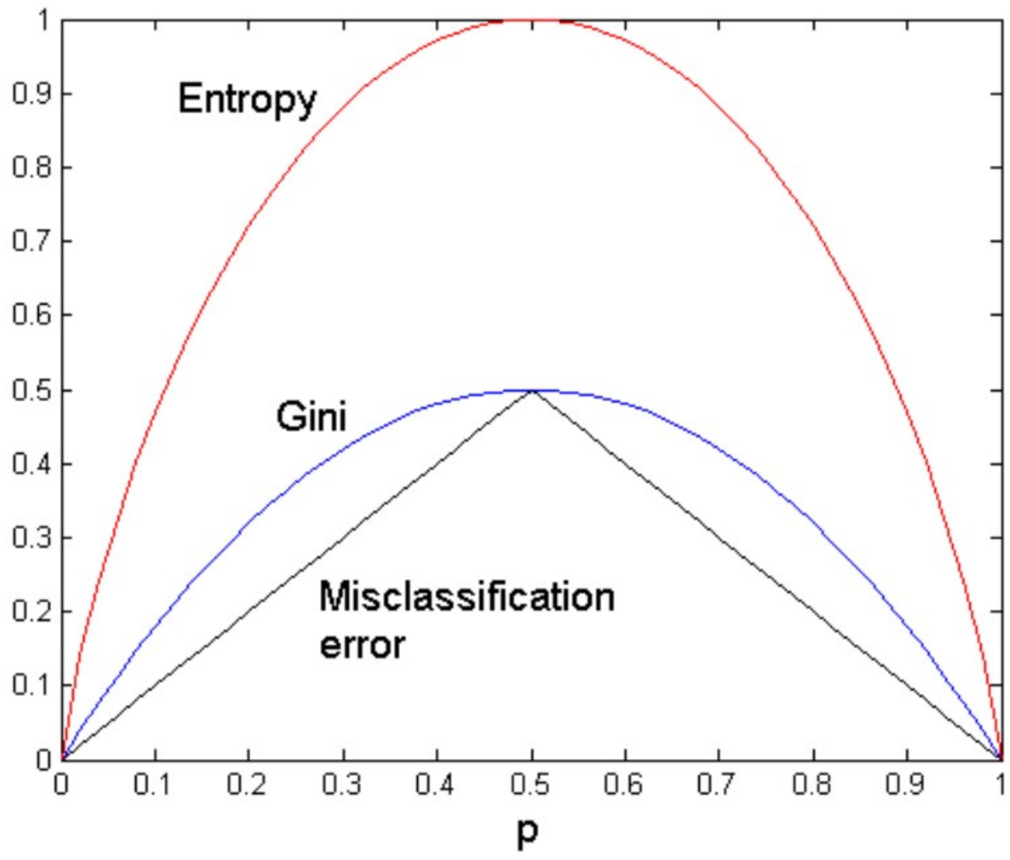
P(C1) = 2/6   P(C2) = 4/6

Entropy = – (2/6) $\log_2$ (2/6) – (4/6) $\log_2$ (4/6) = 0.92

Comparing Impurity Measures

Entropy and Gini are more sensitive to changes in the node probabilities than the misclassification error rate

For a 2-class problem



Information Gain

Information gain is the reduction in entropy produced from splitting.

Information Gain after splitting parent node(p) with n records:

$$GAIN_{split} = Entropy(p) - \left(\sum_{i=1}^{k} \frac{n_i}{n} Entropy(i) \right)$$

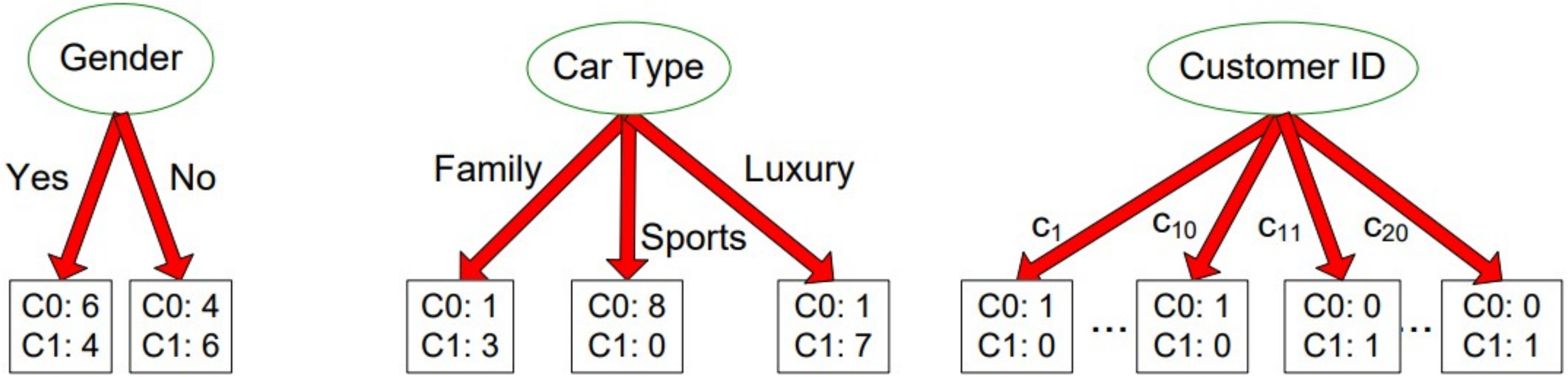
ni is number of records in partition i

– Choose the split that achieves most reduction in Entropy(maximizes GAIN)

– Used in the ID3 decision tree algorithm

Disadvantage of information gain

- It prefers attributes with large number of values that split the data into small, pure subsets $GAIN_{split} = Entropy(p) - \left(\sum_{i=1}^{k} \frac{n_i}{n} Entropy(i) \right)$
- Quinlan's gain ratio uses normalization to improve this



Customer ID has highest information gain because entropy for all the children is zero
information gain has the disadvantage that it prefers attributes with large number of values that split the data into small, pure subsets leads to overfitting to train dataset.

Gain Ratio (Quinlan's Gain Ratio)

- Adjusts Information Gain by the entropy of the partitioning (SplitINFO).
- Large number of small partitions is penalized
- Designed to overcome the disadvantage of Information Gain
- Used in C4.5 algorithm

$$GAIN_{split} = Entropy(p) - \left(\sum_{i=1}^{k} \frac{n_i}{n} Entropy(i) \right), SplitINFO = -\sum_{i=1}^{k} \frac{n_i}{n} \log_2 \frac{n_i}{n}$$

$$GainRATIO_{split} = \frac{GAIN_{Split}}{SplitINFO}$$

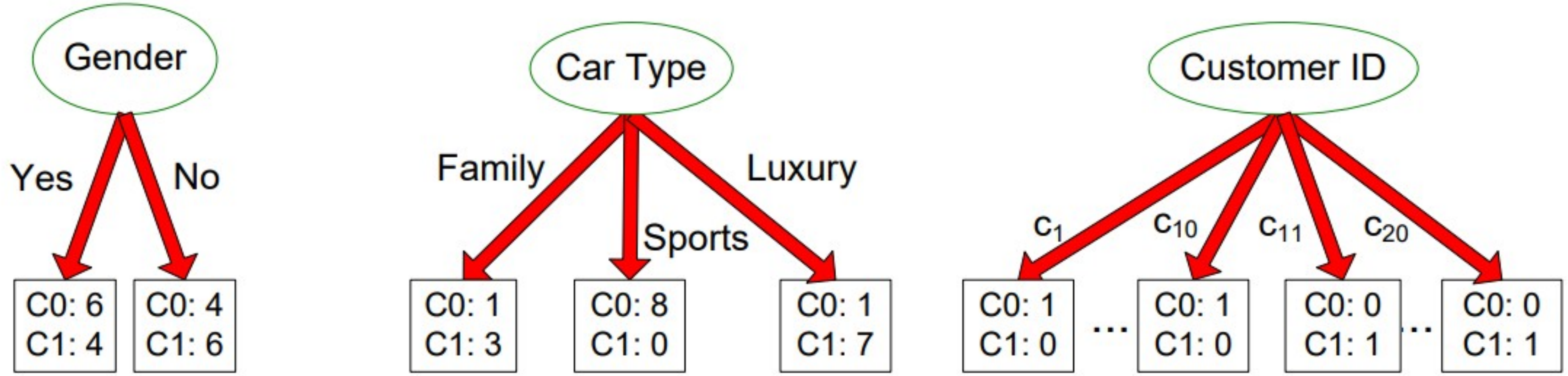
$$SplitINFO = -\sum_{i=1}^{k} \frac{n_i}{n} \log_2 \frac{n_i}{n}$$

Parent Node, p is split into k partitions

n_i is the number of records in partition i

$$GainRATIO_{split} = \frac{GAIN_{Split}}{SplitINFO}$$

$$GAIN_{split} = Entropy(p) - \left(\sum_{i=1}^{k} \frac{n_i}{n} Entropy(i) \right), SplitINFO = -\sum_{i=1}^{k} \frac{n_i}{n} \log_2 \frac{n_i}{n}$$



Gender: Entropy children = (10/20) * 0.97 + (10/20) * 0.97 = 0.97

➔ Information Gain or GAIN $_{split}$ = 1- 0.97 = 0.03

➔ SplitINFO = -(10/20)*log$_2$(10/20) -(10/20)*log$_2$(10/20) = 1 ➔ GainRATIO = 0.03/1.0 = 0.03

Car Type: Entropy children = (4/20) * 0.81 + (8/20) * 0.0 + (8/20) * 0.54 = 0.38

➔ Information Gain or GAIN $_{split}$ = 1- 0.38 = 0.62

➔ SplitINFO = -(4/20)*log$_2$(4/20) -(8/20)*log$_2$(8/20) -(8/20)*log$_2$(8/20) = 1.52 ➔ GainRATIO = 0.62/1.52 = **0.41**

Customer ID: Entropy children = (1/20) * 0.0 + … + (1/20) * 0.0 = 0.0

➔ Information Gain or GAIN $_{split}$ = 1- 0.0 = 1.0

➔ SplitINFO = $\underbrace{\text{-(1/20)*log}_2\text{(1/20) - … -(1/20)*log}_2\text{(1/20)}}_{\text{20 items}}$ = 4.32 ➔ GainRATIO = 1.0/4.32 = 0.23

Select the attribute with the *largest gain ratio* as the splitting criterion in the *decision tree*.

Stop splitting

How should the splitting procedure stop?

- If all the records belong to a same class in all nodes
- If all the records in any node have identical attribute values
- Early termination of process by settings some limits

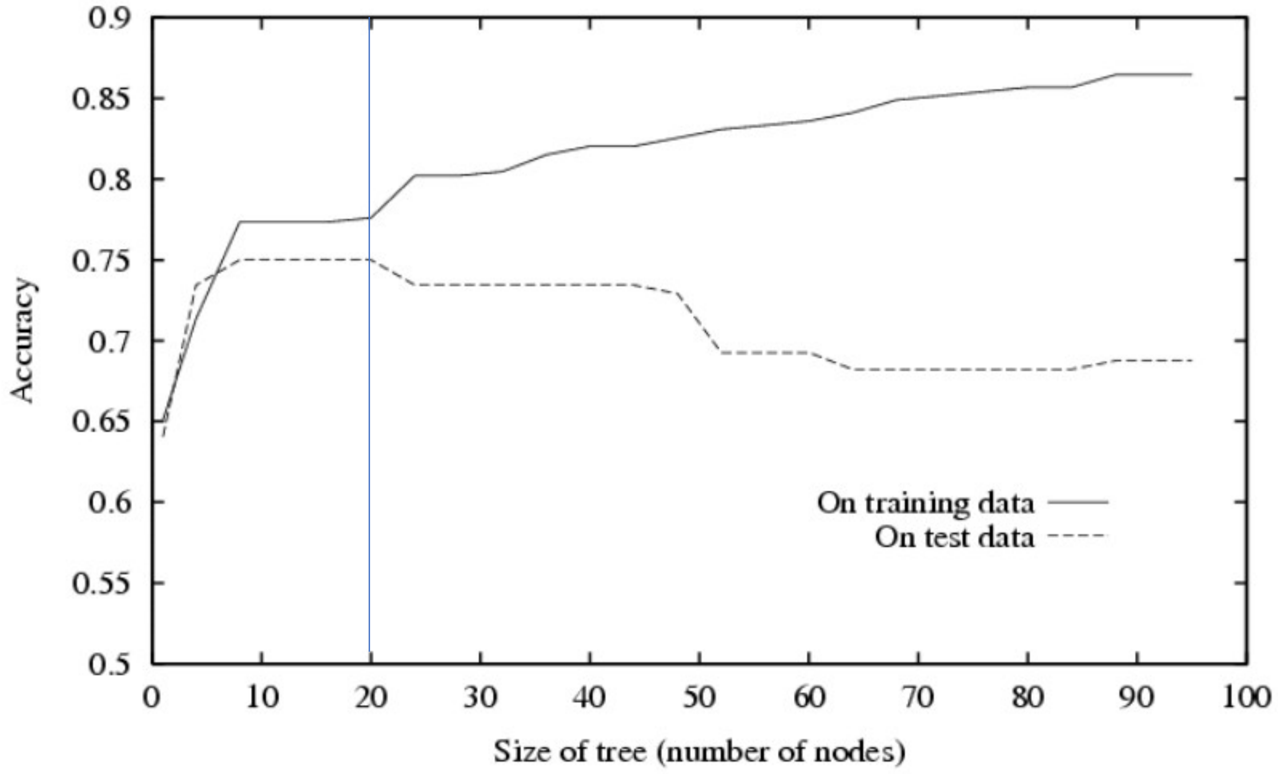
Growth of a decision tree leads to complexity of the model and overfitting to training data

Avoid overfitting

Two main approaches:

1. Reduce model complexity by stopping growth of the tree using constraints (during creation of tree)
2. Grow full tree, then prune to reduce model complexity

Performance of the tree degrades
by adding complexity after 20 nodes



Stop Growth (Pre-pruning) techniques

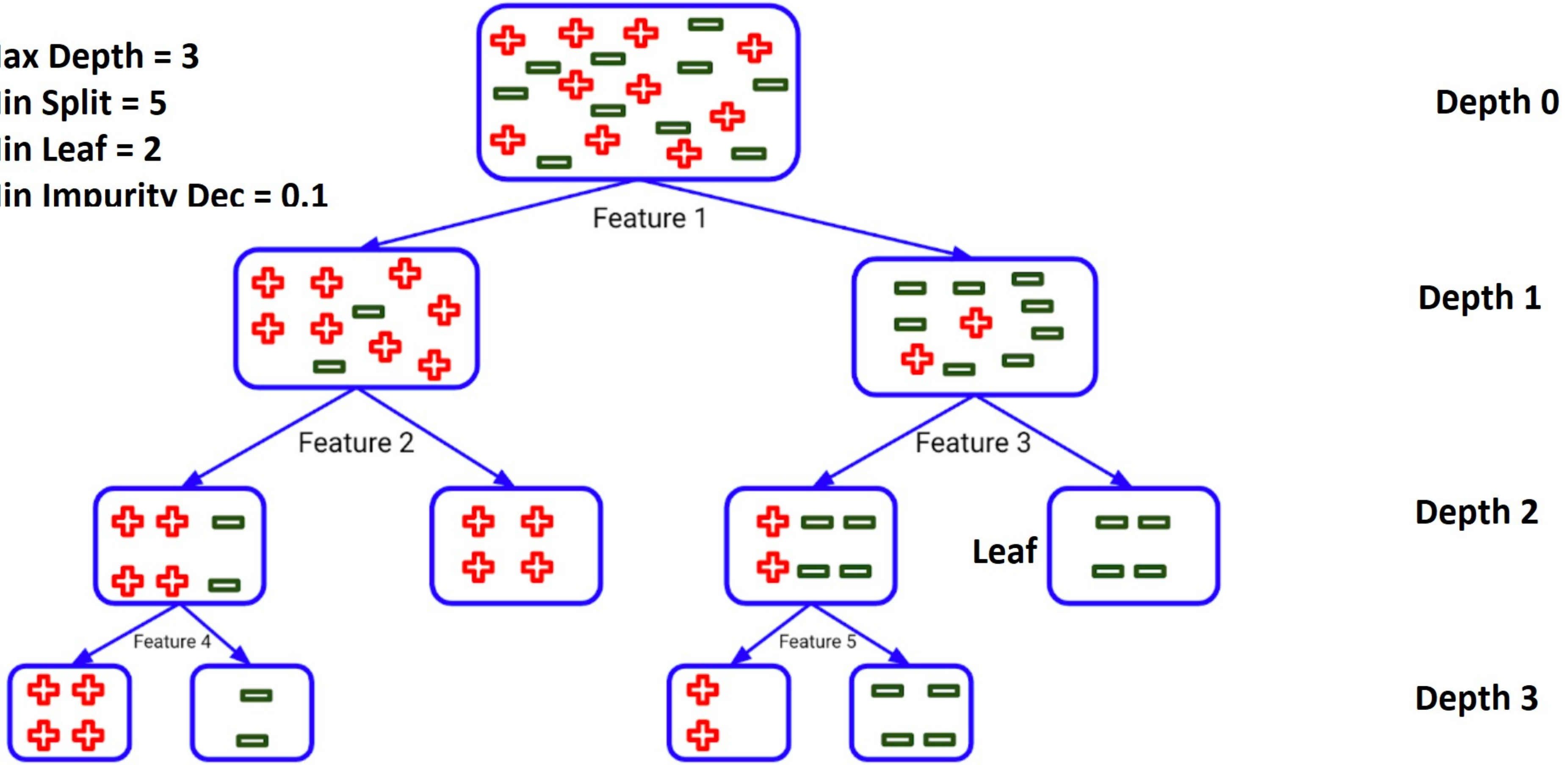
1. Pre pruning stops the growth of decision tree on an early stage (to avoid complexity of model).
2. Complexity of a ML model make it perfect for training data but limits the generalizability of the model thus the performance degrades on test / production phase.
3. We can limit the growth of trees by setting constraints with optimum values using a heuristic approach.
4. The validation data can be used to select best constraints (Hyper-parameters).
5. A hyperparameter is a parameter which specifies details of the learning process, in contrast to parameters which determine the model itself.

Some of hyperparameters can be used as a constraint in decision trees are as follows:

- max_depth: maximum depth of decision tree
- min_sample_split: The minimum number of samples required to split an internal node:
- min_samples_leaf: The minimum number of samples required to be at a leaf node
- min_impurity_decrease: The minimum decrease in impurity by spliting

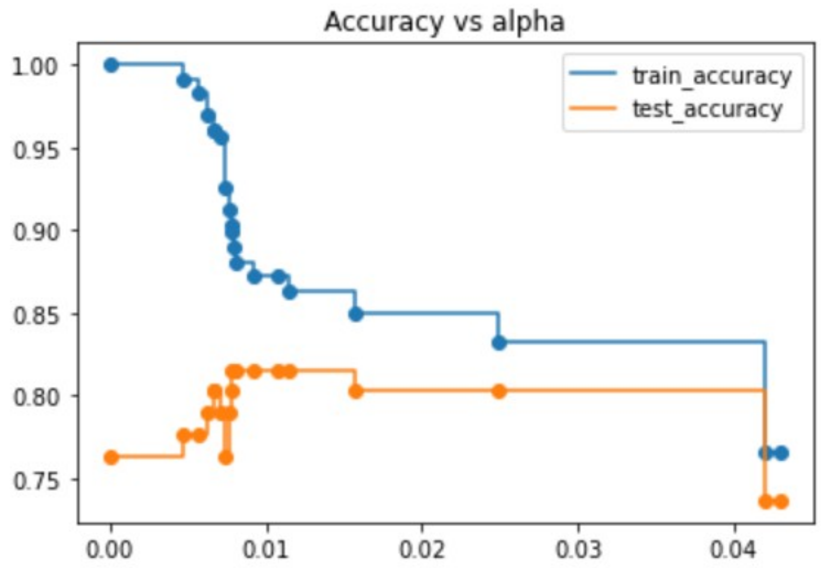
Example of Hyper-parameters

Max Depth = 3
Min Split = 5
Min Leaf = 2
Min Impurity Dec = 0.1



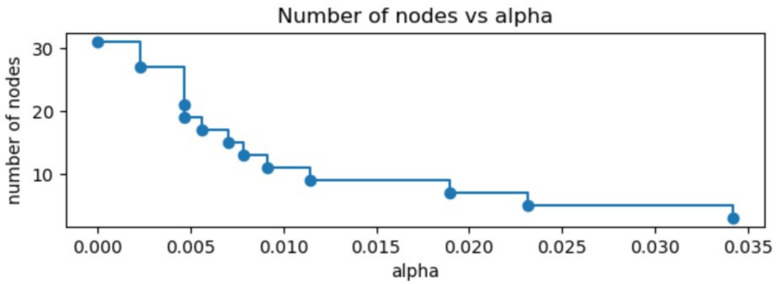
Post pruning techniques

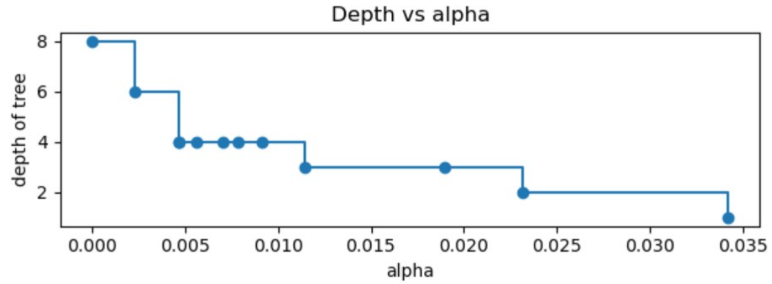
Cost complexity pruning (CCP) provides an option to control the size of a tree by pruning after creating the tree.



- This pruning technique adds complexity penalty to impurity.
- It is parameterized by the cost complexity parameter, ccp_alpha.
- Greater values of ccp_alpha increase the number of nodes pruned.
- When ccp_alpha is set to zero, the tree overfits.
- As alpha increases, more of the tree is pruned, thus creating a decision tree that generalizes better.

Cost complexity prunin



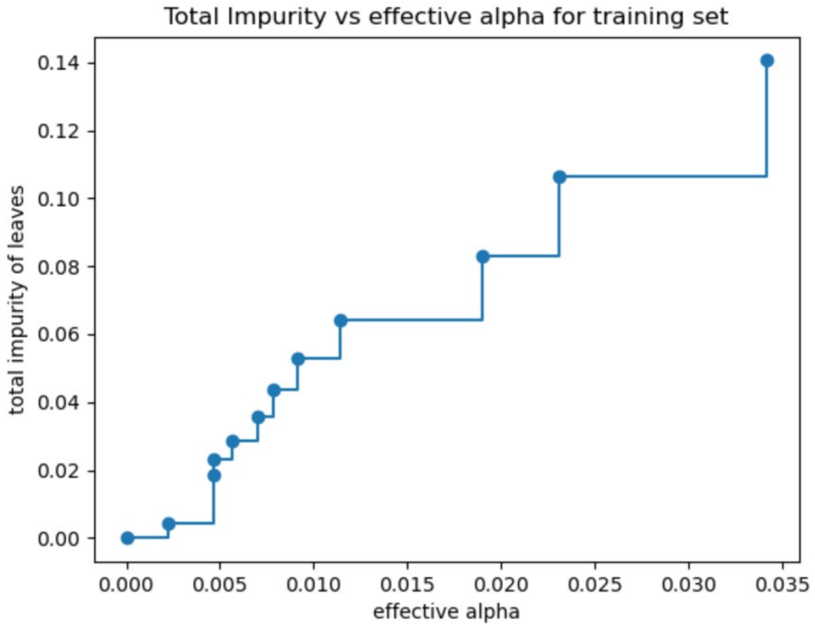


- Minimal cost complexity pruning recursively finds the node with the "weakest link".

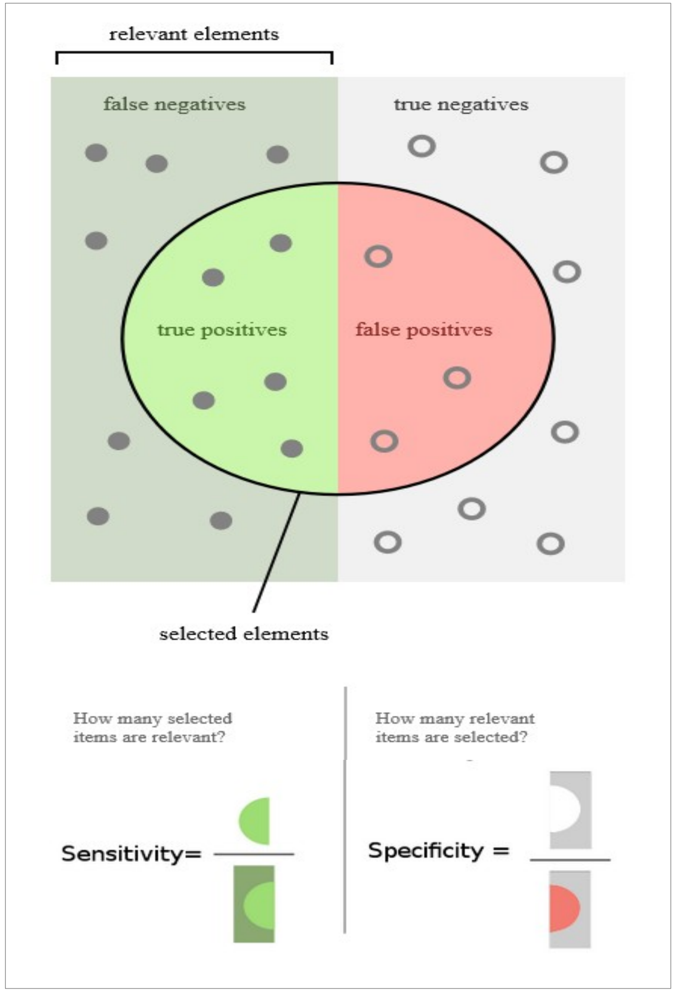
- The weakest link is characterized by an effective alpha, where the nodes with the smallest effective alpha are pruned first.

- Path of cost complexity pruning returns the effective alphas and the corresponding total leaf impurities at each step of the pruning process.

- As alpha increases, more of the tree is pruned, which increases the total impurity of its leaves.



Model Assessment

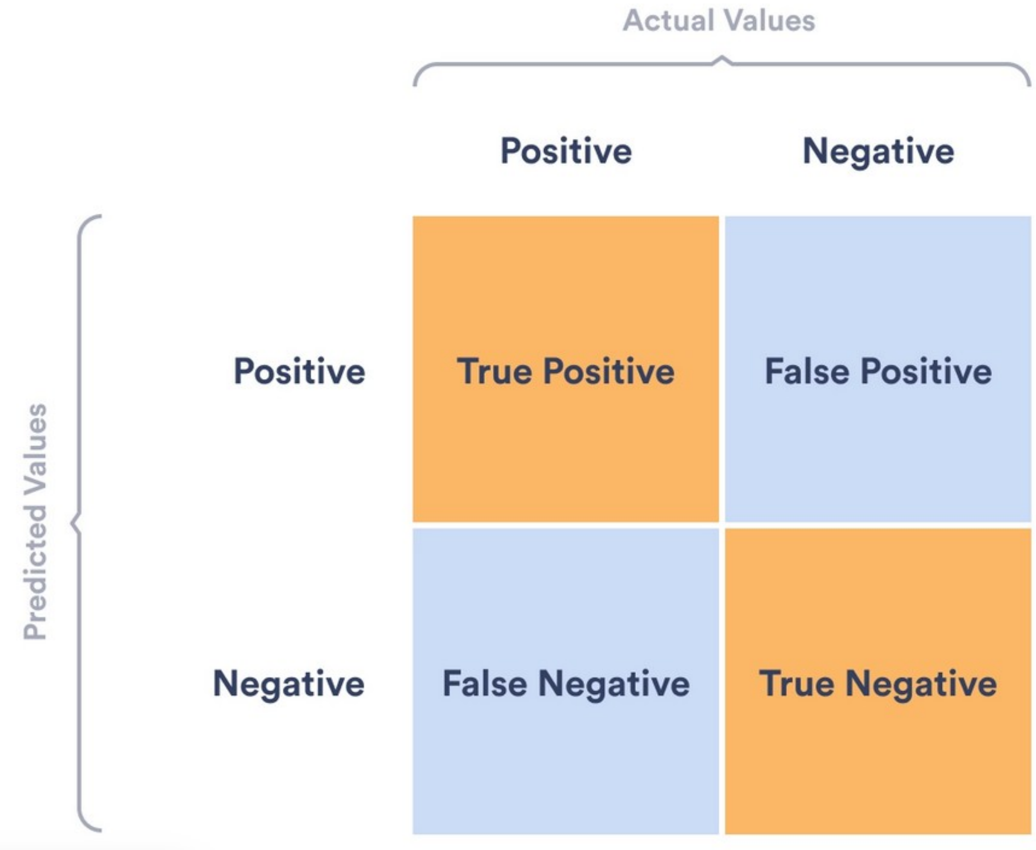


$$Accuracy = \frac{TP + TN}{TP + FP + FN + TN}$$

$$Sensitivity = \frac{TP}{TP + FN}$$

$$Specificity = \frac{TN}{TN + FP}$$

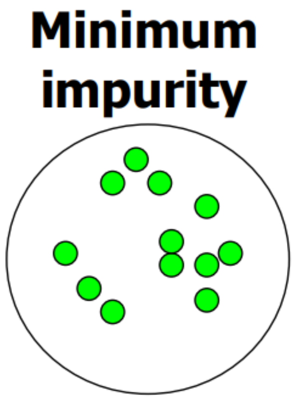
Confusion Matrix

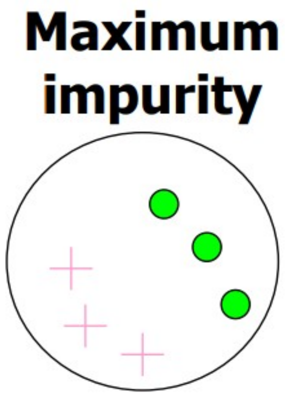


Preprocessing - Balancing Training Data

A dataset in which most of examples belong to a same class is not a good training set for learning (Low Impurity).

We need to balance such dataset using balancing techniques.





Right Assessment in Imbalanced Data

If there is a dataset with more than 90% percent of one class a model which classifies all samples into that class will have an excellent accuracy (more than 90%).

In such cases, Sensitivity (Recall) and Specificity measures should be calculated besides accuracy rate.

It is also possible to design a loss function that is penalizing wrong classification of the rare class more than wrong classifications of the abundant class.

Resampling from Imbalanced Data

Under-sampling:

Balances the dataset by reducing the size of large classes.

This method is used when large quantity of data is available for training

Over-sampling:

Balance dataset by increasing the size of rare classes.

Oversampling is used when the size of training data is limited.

New samples are generated using repetition, bootstrapping or SMOTE

Preprocessing - Normalizing Data

1. Loss function in Decision Tree is impurity
2. Despite many ML models this loss function (impurity) is not affected by distance between feature values, therefore, normalizing features (scaling) has no impact on performance of the model.
3. In white-box models (e.g., Decision Tree), not just prediction but interpretability of results is important, especially if the model is used in a semi-automated decision-making process.

2,3 -> Normalizing features is not recommended since it has no effect on performance but makes results less interpretable.

Dealing with missing values in D-tree

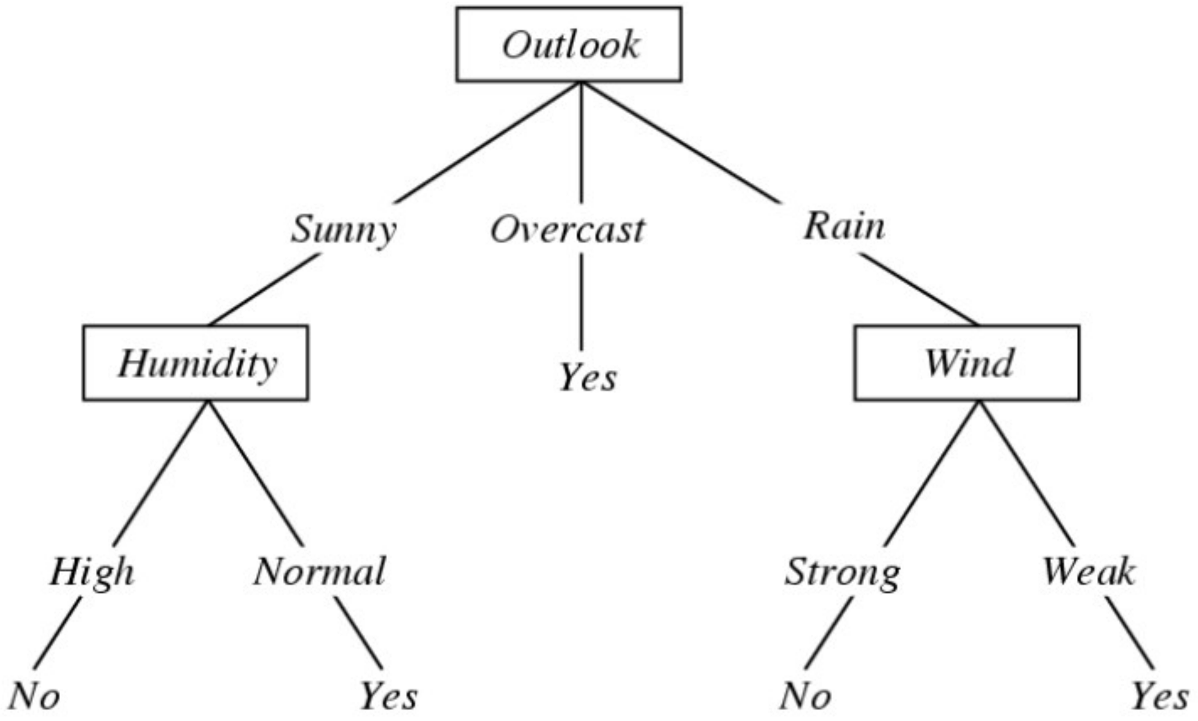
- All missing values assigned to a node with the biggest number of instances.

- Missing values are distributed to all children proportional with the number of instances from each child node.

- Missing values are distributed randomly to only one single child node.

- Missing values can be handled during data pre-processing phase.

Convert Decision trees to IF-THEN rules

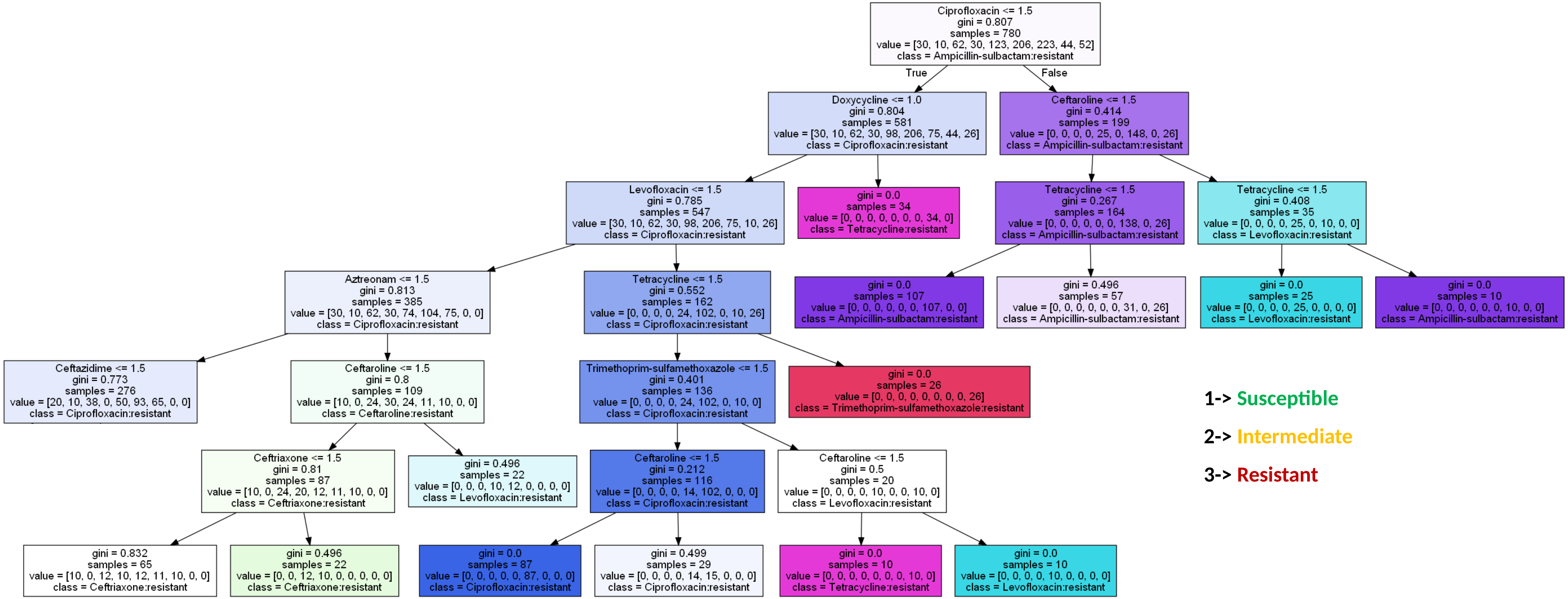


IF $(Outlook = Sunny)\ AND\ (Humidity = High)$
THEN $PlayTennis = No$

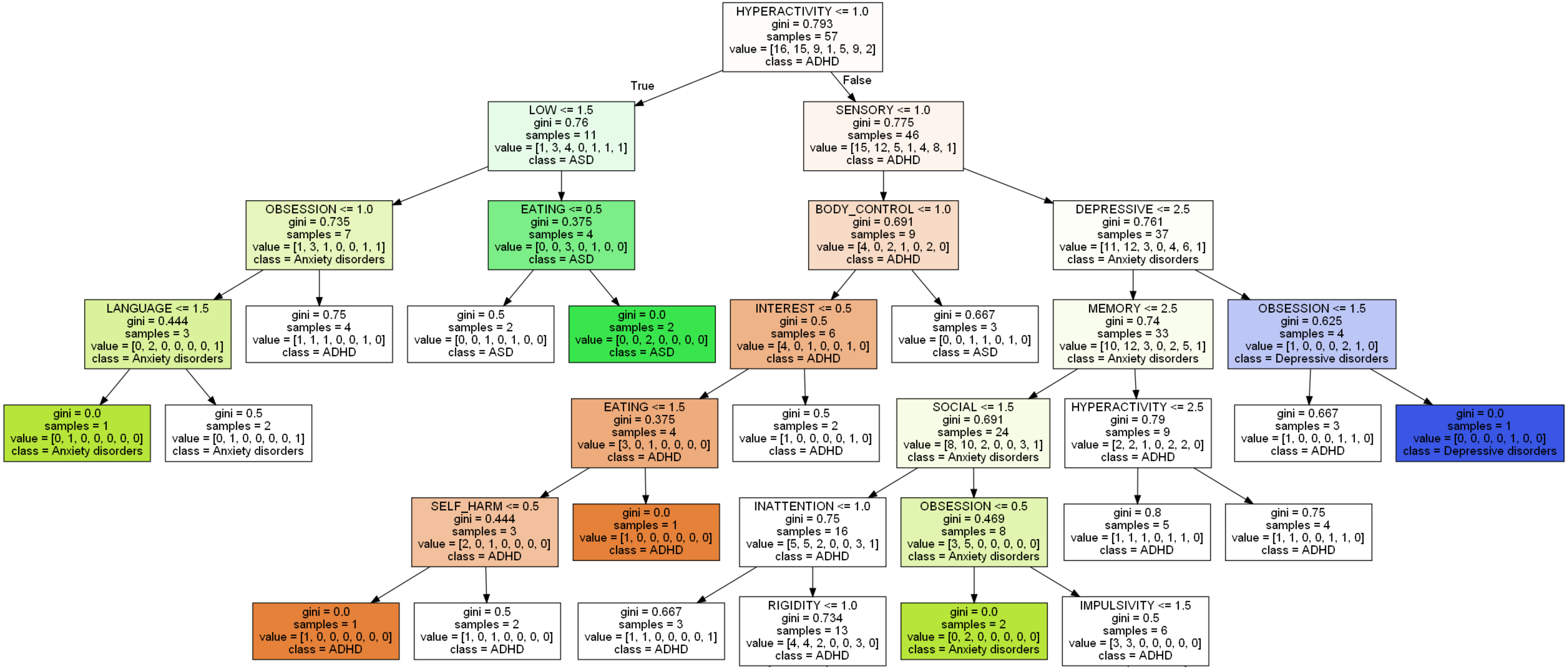
IF $(Outlook = Sunny)\ AND\ (Humidity = Normal)$
THEN $PlayTennis = Yes$

...

Application of Decision Tree



Application of Decision Tree



Most Common D-Tree Algorithms

- CART: Classification And Regression Tree is a non-incremental decision tree inducer for both classification and regression problems.
- ID3, C5: A “decision tree induction algorithm”, developed by Ross Quinlan (1979). ID3 stands for "Iterative Dichotomiser (version) 3". Later versions include C4 (1987), C4.5 (1993), and C5.
- SLIQ: Supervised Learning In Ques uses a fast sub-setting algorithm for determining splits for categorical attributes.
- SPRINT: Scalable PaRallelizable INndution of decision Trees is a scalable parallel classifier for data mining.
- VFDT: Very Fast Decision Trees learner reduces training time for large incremental data sets by subsampling the incoming data stream.
- EFDT : Extremely Fast Decision Tree learner is statistically more powerful than VFDT, allowing it to learn more detailed trees from less data.

D-Tree Advantages

- Handle non-Linear classification
- Interpretability of results
- Transparency in producing results (White-box model)
- Accuracy is reasonable for most uncomplex problems
- Model transparency supports semi-automated decision making suitable for many applications
- Decision trees are able to generate understandable rules.

D-Tree Disadvantages

- Accuracy is less than some advanced ML techniques (e.g., DNNs) for some complex classification / prediction problems
- Decision trees are prone to errors in classification problems with small number of training examples.
- The processes of growing and pruning of decisions tree are computationally expensive
- The Decision trees are prone to overfitting problem.
- A small change in the training data can lead to a completely different tree structure, making decision trees unstable

Assignment 2

Clean the dataset used in the first assignment in a way that is suitable for decision trees.

Create a binary decision tree using Gini index impurity measure over the cleaned data to predict Resistance (as the label) by Patient_Age, Bacteria, and Antimicrobial (as features).

Consider the following ranges of values for hyper-parameters:
max_depth = [3, 5, 10, 15]
min_sample_split= [5, 10, 30, 60]
min_samples_leaf= [3, 10, 20, 40]
min_impurity_decrease = [0.05, 0.01, 0.001, 0.0005]
ccp_alpha = [0.01, 0.001, 0.0005, 0.0001]

Spilt data into train, test, and validation (72%, 20%, 8%) and use validation data to select best hyper-parameter.
Calculate accuracy , specificity , and sensitivity of the best Dtree on test data using confusion matrix

Extract three rules from the decision tree.