Module: Adv Machine Learning

Live Session-1

Agenda:
Decision Trees
Classification
Regression

Hierarchical Clustering

- *K*-means is an objective-based approach that requires us to pre-specify the number of clusters *K*

- The answer it gives is somewhat random: it depends on the random initialization we started with

- Hierarchical clustering is an alternative approach that does not require a pre-specified choice of *K*, and which provides a deterministic answer (no randomness)

- We'll focus on bottom-up or agglomerative hierarchical clustering.

DECISION TREE

Decision Tree:

- It is one of the most popular algorithms in Machine Learning. It can be used for classification mainly, but can also be used regression.

- A decision tree is a structure that includes a root node, branches, and leaf nodes.

- Each internal node denotes a test on an attribute, each branch denotes the outcome of a test, and each leaf node holds a class label.

Algorithm:

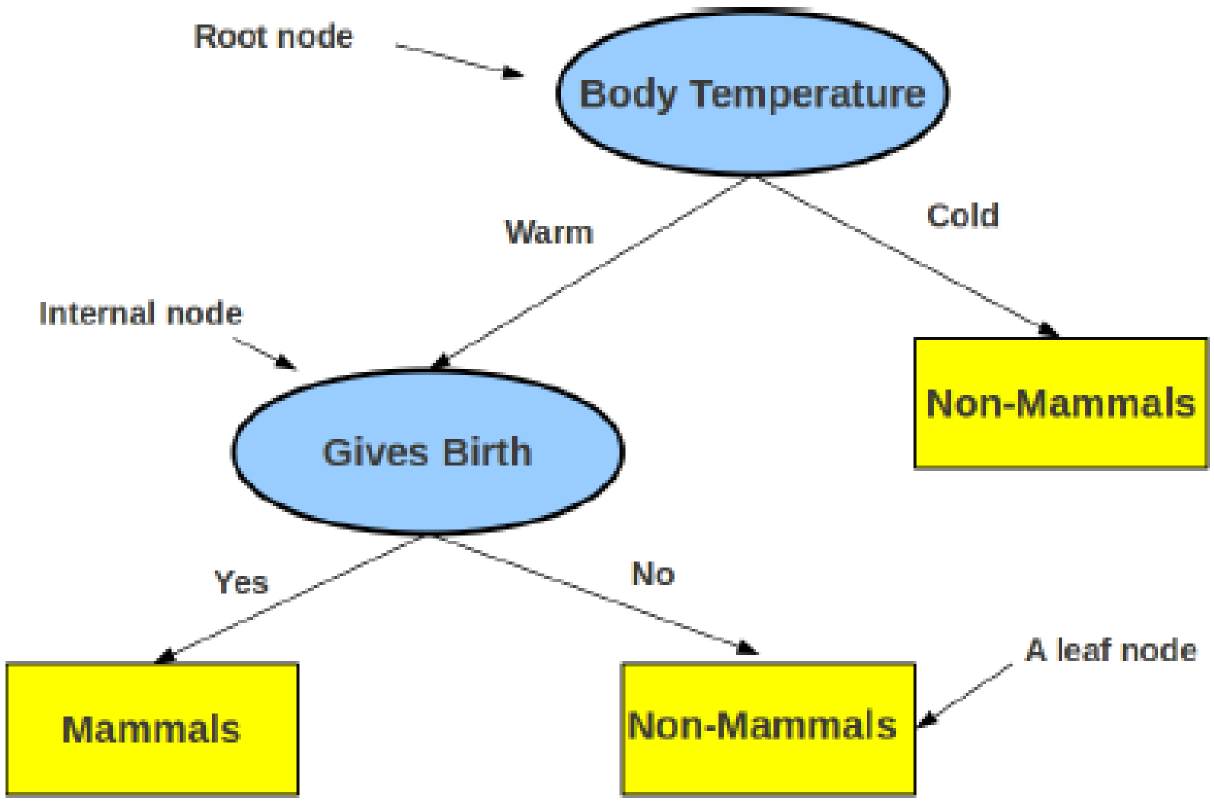
I. At the beginning, the whole training set is considered as the root.

II. Feature values need to be categorical. If the values are continuous then they are discretized prior to building the model.

III. Records are distributed recursively on the basis of attribute values.

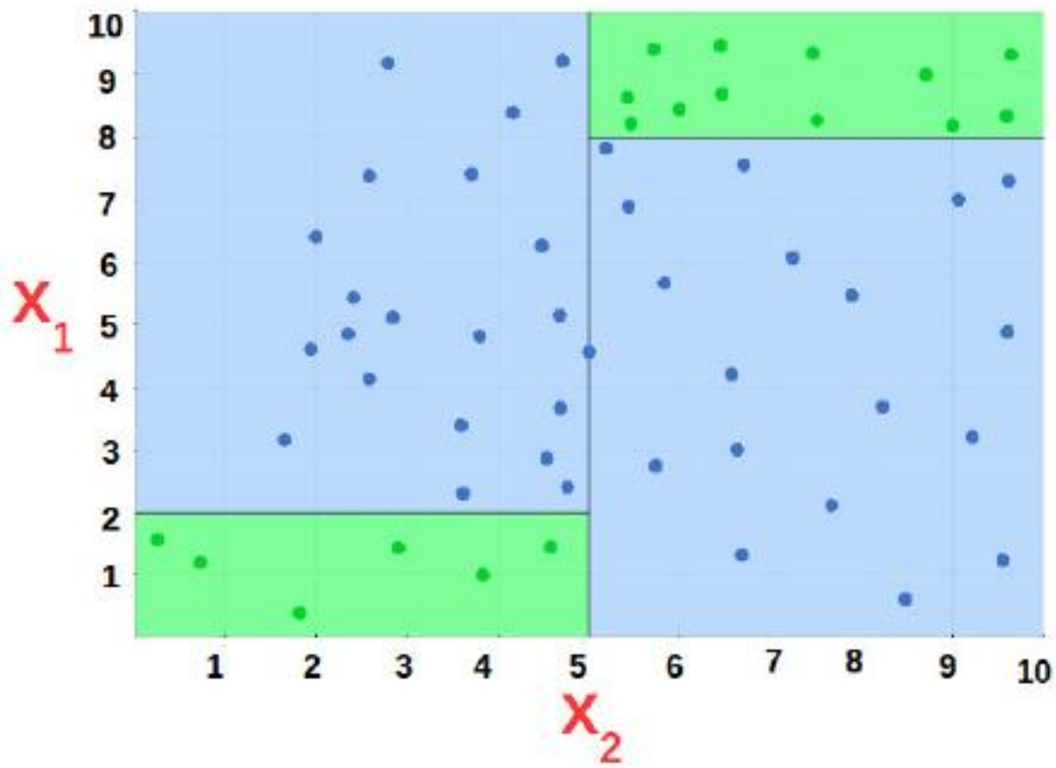
IV. Order to placing attributes as root or internal node of the tree is done by using some statistical approach.

Algorithm:

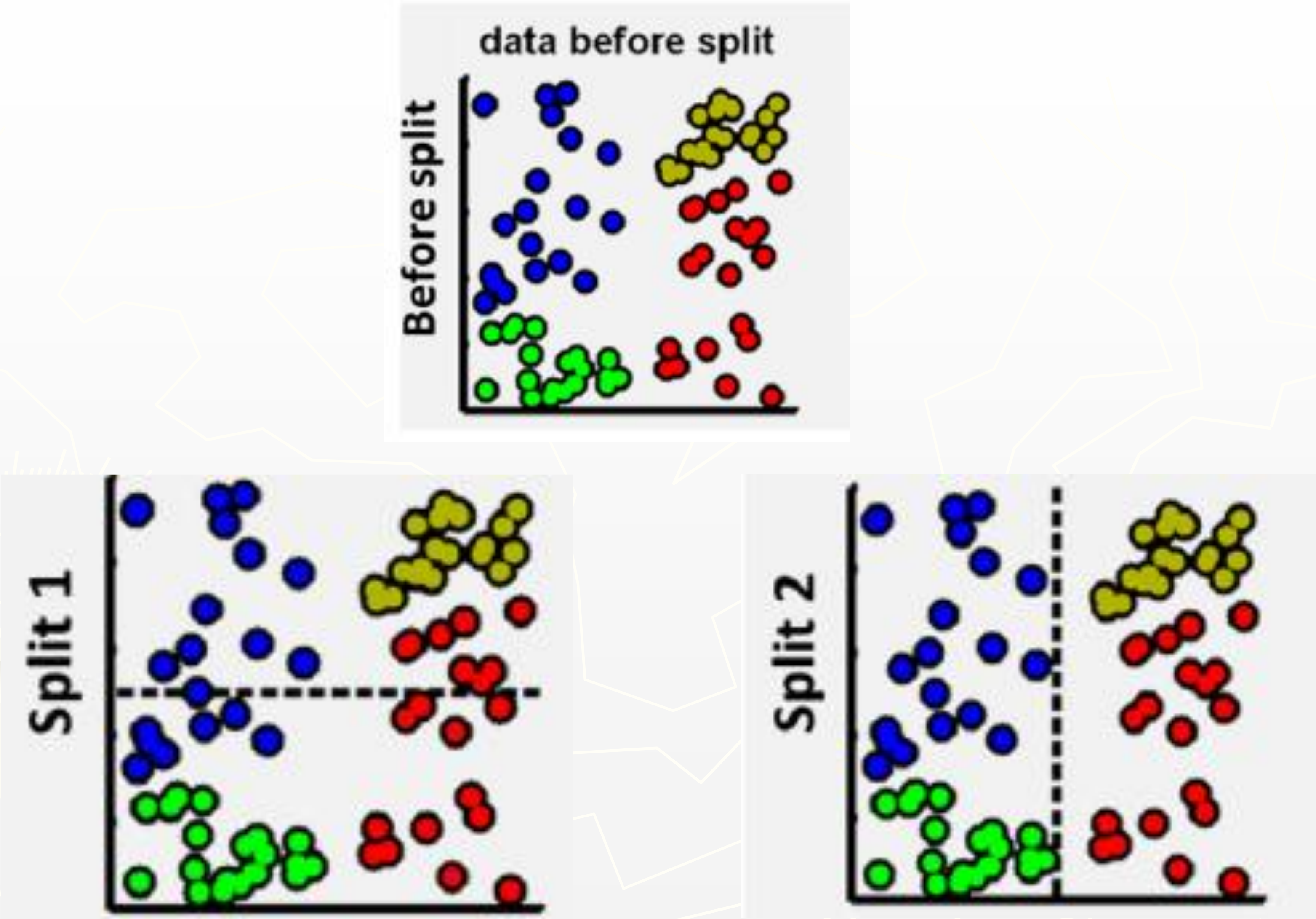


Breiman, Leo; Friedman, J. H.; Olshen, R. A.; Stone, C. J. (1984). Classication and regression trees

Feature importance



Ordering of features



Machine learning by Tom M. Mitchell

Entropy:

- Entropy is a measure of randomness/uncertainty of a set
- Assume our data is a set S of examples with C many classes
- p_c is the probability that a random element of S belongs to class c
  - .. basically, the fraction of elements of S belonging to class c
- Probability vector $p = [p_1, p_2, \ldots, p_C]$ is the class distribution of the set S
- Entropy of the set S

$$H(S) = -\sum_{c \in C} p_c \log_2 p_c$$

- If a set S of examples (or any subset of it) has..
  - Some dominant classes $\Longrightarrow$ small entropy of the class distribution
  - Equiprobable classes $\Longrightarrow$ high entropy of the class distribution
- We can assess informativeness of each feature by looking at how much it reduces the entropy of the class distribution

Entropy:

- Let's assume each element of S has a set of features
- Information Gain (IG) on knowing the value of some feature 'F'

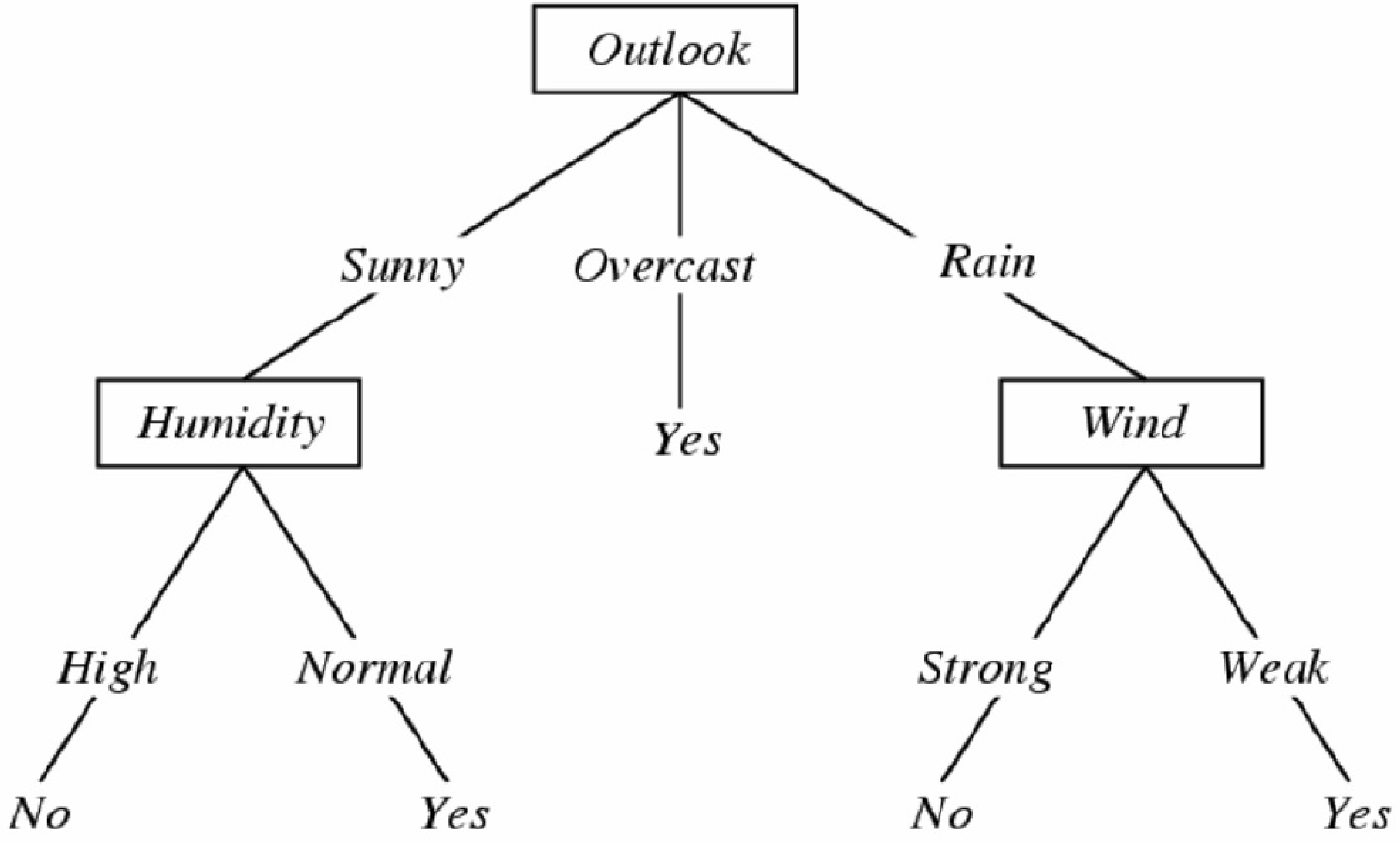
$$IG(S,F) = H(S) - \sum_{f \in F} \frac{|S_f|}{|S|} H(S_f)$$

- S_f denotes the subset of elements of S for which feature F has value f
- $IG(S,F)$ = entropy of S minus the weighted sum of entropy of its children
- $IG(S,F)$: Increase in our certainty about S once we know the value of F
- $IG(S,F)$ denotes the no. of bits saved while encoding S once we know the value of the feature F

Entropy by example

day	outlook	temperature	humidity	wind	play
1	sunny	hot	high	weak	no
2	sunny	hot	high	strong	no
3	overcast	hot	high	weak	yes
4	rain	mild	high	weak	yes
5	rain	cool	normal	weak	yes
6	rain	cool	normal	strong	no
7	overcast	cool	normal	strong	yes
8	sunny	mild	high	weak	no
9	sunny	cool	normal	weak	yes
10	rain	mild	normal	weak	yes
11	sunny	mild	normal	strong	yes
12	overcast	mild	high	strong	yes
13	overcast	hot	normal	weak	yes
14	rain	mild	high	strong	no

Tree


Outlook
Sunny
Overcast
Rain
Humidity
Yes
Wind
High
Normal
Strong
Weak
No
Yes
No
Yes

Entropy continued:

Entropy using the frequency table of one attribute:

$$E(S) = \sum_{i=1}^{c} - p_i \log_2 p_i$$

Play Golf	
Yes	No
9	5

Entropy(PlayGolf) = Entropy (5,9)

= Entropy (0.36, 0.64)

= - (0.36 $\log_2$ 0.36) - (0.64 $\log_2$ 0.64)

= 0.94

Sources: Saed Sayad: Decision Tree

Entropy cont.

Entropy using the frequency table of two attributes:

$$E(T, X) = \sum_{c \in X} P(c)E(c)$$

		Play Golf		
		Yes	No	
Outlook	Sunny	3	2	5
	Overcast	4	0	4
	Rainy	2	3	5
				14

E(PlayGolf, Outlook) = **P**(Sunny)***E**(3,2) + **P**(Overcast)***E**(4,0) + **P**(Rainy)***E**(2,3)

= (5/14)*0.971 + (4/14)*0.0 + (5/14)*0.971

= 0.693

Sources: Saed Sayad: Decision Tree

Information Gain:

The information gain is based on the decrease in entropy after a dataset is split on an attribute. Constructing a decision tree is all about finding attribute that returns the highest information gain (i.e., the most homogeneous branches).

***Step 1*: Calculate entropy of the target**.

$$\begin{aligned}\text{Entropy(PlayGolf)} &= \text{Entropy}(5,9)\\ &= \text{Entropy}(0.36, 0.64)\\ &= -(0.36 \log_2 0.36) - (0.64 \log_2 0.64)\\ &= 0.94\end{aligned}$$

Sources: Saed Sayad: Decision Tree

Information Gain:

Step 2: The dataset is then split on the different attributes. The entropy for each branch is calculated. Then it is added proportionally, to get total entropy for the split. The resulting entropy is subtracted from the entropy before the split. The result is the Information Gain, or decrease in entropy.

		Play Golf	
		Yes	No
Outlook	Sunny	3	2
	Overcast	4	0
	Rainy	2	3
Gain = 0.247			

		Play Golf	
		Yes	No
Temp.	Hot	2	2
	Mild	4	2
	Cool	3	1
Gain = 0.029			

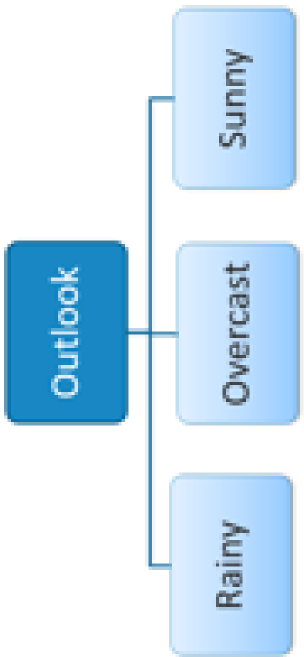
		Play Golf	
		Yes	No
Humidity	High	3	4
	Normal	6	1
Gain = 0.152			

		Play Golf	
		Yes	No
Windy	False	6	2
	True	3	3
Gain = 0.048			

Information Gain:

Step 3: Choose attribute with the largest information gain as the decision node, divide the dataset by its branches and repeat the same process on every branch

		Play Golf	
		Yes	No
Outlook	Sunny	3	2
	Overcast	4	0
	Rainy	2	3
Gain = 0.247			

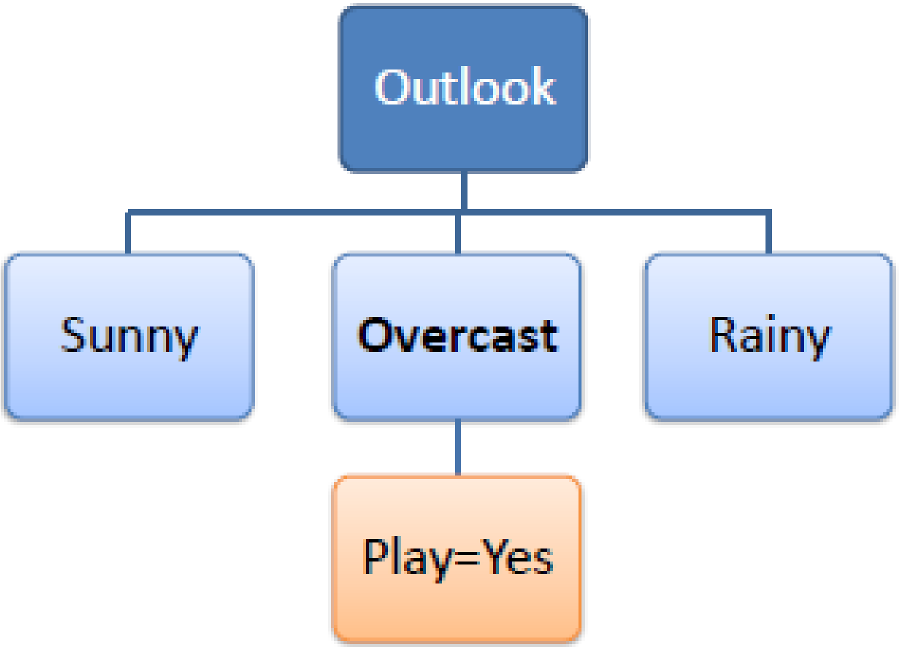


Outlook	Temp	Humidity	Windy	Play Golf
Sunny	Mild	High	FALSE	Yes
Sunny	Cool	Normal	FALSE	Yes
Sunny	Cool	Normal	TRUE	No
Sunny	Mild	Normal	FALSE	Yes
Sunny	Mild	High	TRUE	No
Overcast	Hot	High	FALSE	Yes
Overcast	Cool	Normal	TRUE	Yes
Overcast	Mild	High	TRUE	Yes
Overcast	Hot	Normal	FALSE	Yes
Rainy	Hot	High	FALSE	No
Rainy	Hot	High	TRUE	No
Rainy	Mild	High	FALSE	No
Rainy	Cool	Normal	FALSE	Yes
Rainy	Mild	Normal	TRUE	Yes

Information gain:

Step 4a: A branch with entropy of 0 is a leaf node.

Temp.	Humidity	Windy	Play Golf
Hot	High	FALSE	Yes
Cool	Normal	TRUE	Yes
Mild	High	TRUE	Yes
Hot	Normal	FALSE	Yes

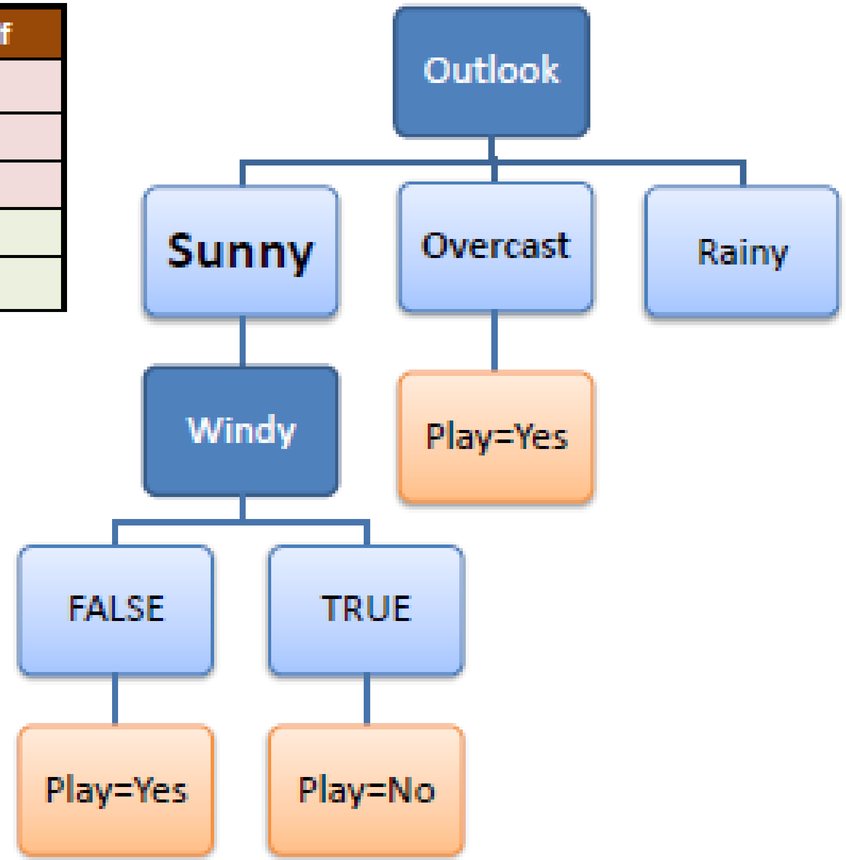


Sources: Saed Sayad: Decision Tree

Information gain:

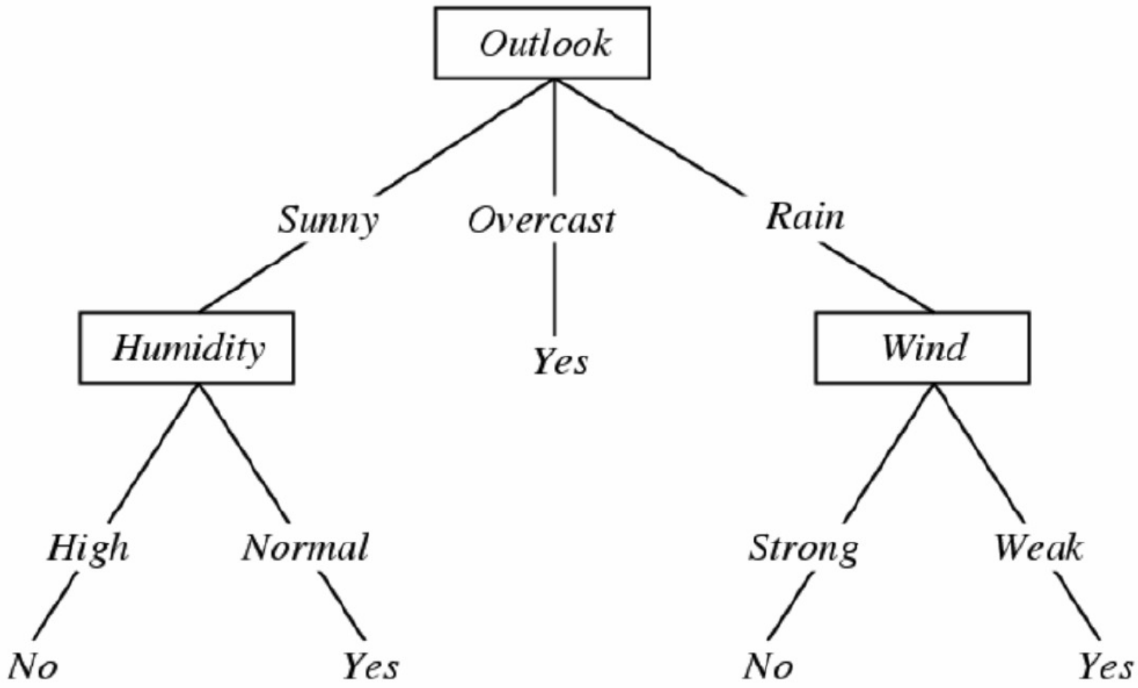
Step 4b: A branch with entropy more than 0 needs further splitting.

Temp.	Humidity	Windy	Play Golf
Mild	High	FALSE	Yes
Cool	Normal	FALSE	Yes
Mild	Normal	FALSE	Yes
Cool	Normal	TRUE	No
Mild	High	TRUE	No



Step 5: The ID3 algorithm is run recursively on the non-leaf branches, until all data is classified.

Decision rules



R_1: IF (Outlook=Sunny) AND (Windy=FALSE) THEN Play=Yes

R_2: IF (Outlook=Sunny) AND (Windy=TRUE) THEN Play=No

R_3: IF (Outlook=Overcast) THEN Play=Yes

R_4: IF (Outlook=Rainy) AND (Humidity=High) THEN Play=No

R_5: IF (Outlook=Rain) AND (Humidity=Normal) THEN Play=Yes

Machine learning by Tom M. Mitchell

GINI Index:

Another decision tree algorithm CART uses the *Gini method* to create split points, including the *Gini Index (Gini Impurity) and Gini Gain.*

Definition of Gini Index: The probability of assigning a wrong label to a sample by picking the label randomly and is also used to measure feature importance in a tree.

$$GINI\ Index = 1 - \sum p_j^2$$

GINI Index:

		Yes	No	Total
Feature 2:	Sunny	3	2	5
Outlook	Overcast	4	0	4
	Rainy	3	2	5
	Total	10	4	

Gini (PlayTennis, Outlook=Sunny)
= 1-(⅖)^2 - (⅗)^2 =0.48

Gini (PlayTennis, Outlook=Overcast)
= 1-(4/4)^2 - (0/4)^2=0

Gini (PlayTennis, Outlook=Rainy)
= 1-(⅗)^2 - (⅖)^2 =0.48

The Gigi Index of Outlook (children node)
= 5/14 x 0.48 + 4/14 x 0 + 5/14 x 0.48 = 0.3429

Gini Gain = Gini (parent node) - Gini (children node)

= [1- (10/14)^2 -(4/14)^2] - 0.3429
= 0.4082 - 0.3429
= 0.065

GINI Index:

After calculating Gini Gain for every attribute, sklearn.tree.**DecisionTreeClassifier** will choose the attribute with the **largest Gini Gain** as the Root Node.

A branch with Gini of 0 is a leaf node, while a branch with Gini more than 0 needs further splitting.

Training Algorithm	**CART** (Classification and Regression Trees)	**ID3** (Iterative Dichotomiser 3)
Target(s)	Classification and Regression	Classification
Metric	Gini Index	Entropy function and Infomation Gain
Cost function (Based on what to split?)	Select its splits to achieve the subsets that minimize Gini Impurity	Yield the largest Information Gain for categorical targets

DT Regression?

DT Regression is similar to DT Classification, however we use **Mean Square Error** (MSE, default) or **Mean Absolute Error** (MAE) instead of *cross-entropy* or *Gini impurity* to determine splits.

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$$

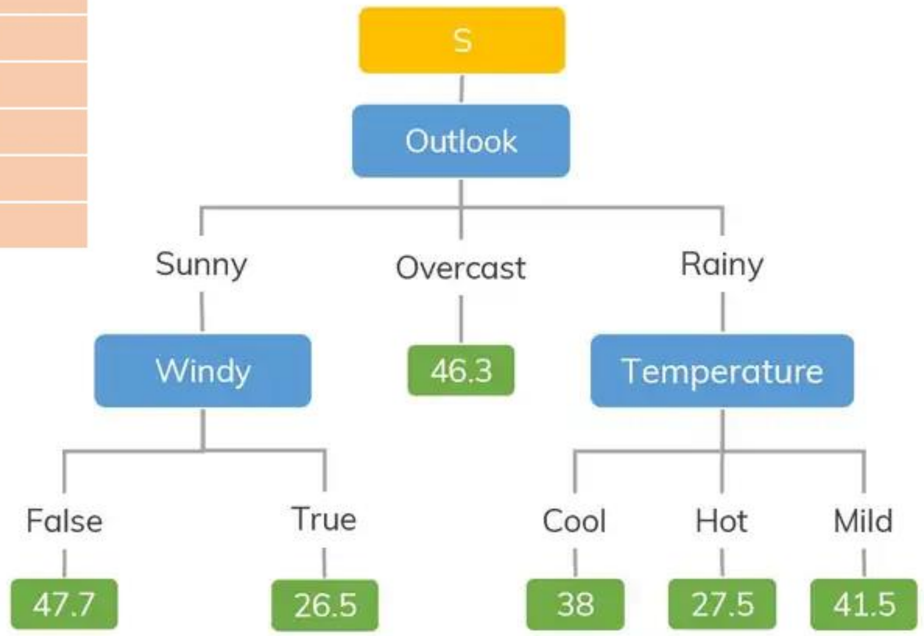
$$MAE = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i|$$

DT Regression: Example

Outlook	Temperature	Humidity	Windy	Hours Played
Rainy	Hot	High	False	26
Rainy	Hot	High	True	30
Overcast	Hot	High	False	48
Sunny	Mild	High	False	46
Sunny	Cool	Normal	False	52
Sunny	Cool	Normal	True	23
Overcast	Cool	Normal	True	43
Rainy	Mild	High	False	35
Rainy	Cool	Normal	False	38
Sunny	Mild	Normal	False	48
Rainy	Mild	Normal	True	48
Overcast	Mild	High	True	52
Overcast	Hot	Normal	False	44
Sunny	Mild	High	True	30

DT Regression: Example

Outlook	Temperature	Humidity	Windy	Hours Played
Rainy	Hot	High	False	26
Rainy	Hot	High	True	30
Overcast	Hot	High	False	48
Sunny	Mild	High	False	46
Sunny	Cool	Normal	False	52
Sunny	Cool	Normal	True	23
Overcast	Cool	Normal	True	43
Rainy	Mild	High	False	35
Rainy	Cool	Normal	False	38
Sunny	Mild	Normal	False	48
Rainy	Mild	Normal	True	48
Overcast	Mild	High	True	52
Overcast	Hot	Normal	False	44
Sunny	Mild	High	True	30



Algorithm: Step-1

- Calculate the **Standard Deviation** (*SD*) of the current node (let's say S, parent node) by using MSE or MAE.

$$SD(MSE) = \sqrt{(\frac{1}{n}\sum_{i=1}^{n}(y_i - \bar{y}_i)^2)}$$

Algorithm: Step-2

- Check the **stopping conditions** (we don't need to make any split at this node) to stop the split and this node becomes a leaf node. Otherwise, go to step 3.

Criteria:

- The minimum number of samples required to split an internal node, use min_samples_split in scikit-learn.
- The maximum depth of the tree, use max_depth in scikit-learn.
- Its *coefficient of variation* $\frac{SD(S)}{\bar{y}}$ is less than a certain threshold.

Algorithm: Step-3

- Calculate the **Standard Deviation Reduction** (SDR) after splitting node S on each attribute (for example, consider attribute O). The attribute w.r.t. the biggest SDR will be chosen!

$$\underbrace{SDR(S,O)}_{\text{Standard Deviation Reduction}} = \underbrace{SD(S)}_{\text{SD before split}} - \underbrace{\sum_j P(O_j|S) \times SD(S,O_j)}_{\text{weighted SD after split}}$$

where *j* number of different properties in O, and *P*(*O_j*) is the probability of *O_j* in O.

Note that, *SD*(*S*,*Oj*) means the SD of node *Oj* which is also a child of node S

Algorithm: Step-3..

- Calculate the **Standard Deviation Reduction** (SDR) after splitting node S on each attribute (for example, consider attribute O). The attribute w.r.t. the biggest SDR will be chosen!

$$\underbrace{SDR(S,O)}_{\text{Standard Deviation Reduction}} = \underbrace{SD(S)}_{\text{SD before split}} - \underbrace{\sum_j P(O_j|S) \times SD(S,O_j)}_{\text{weighted SD after split}}$$

Algorithm: Step-3..

$$\underbrace{SDR(S,O)}_{\text{Standard Deviation Reduction}} = \underbrace{SD(S)}_{\text{SD before split}} - \underbrace{\sum_j P(O_j|S) \times SD(S,O_j)}_{\text{weighted SD after split}}$$

Algorithm: Step-3..

Algorithm: Step-4

- After splitting, we have new child nodes. Each of them becomes a new parent node in the next step. Go back to step-1