Classification: Decision Tree

Penyusun Modul: Chairul Aulia

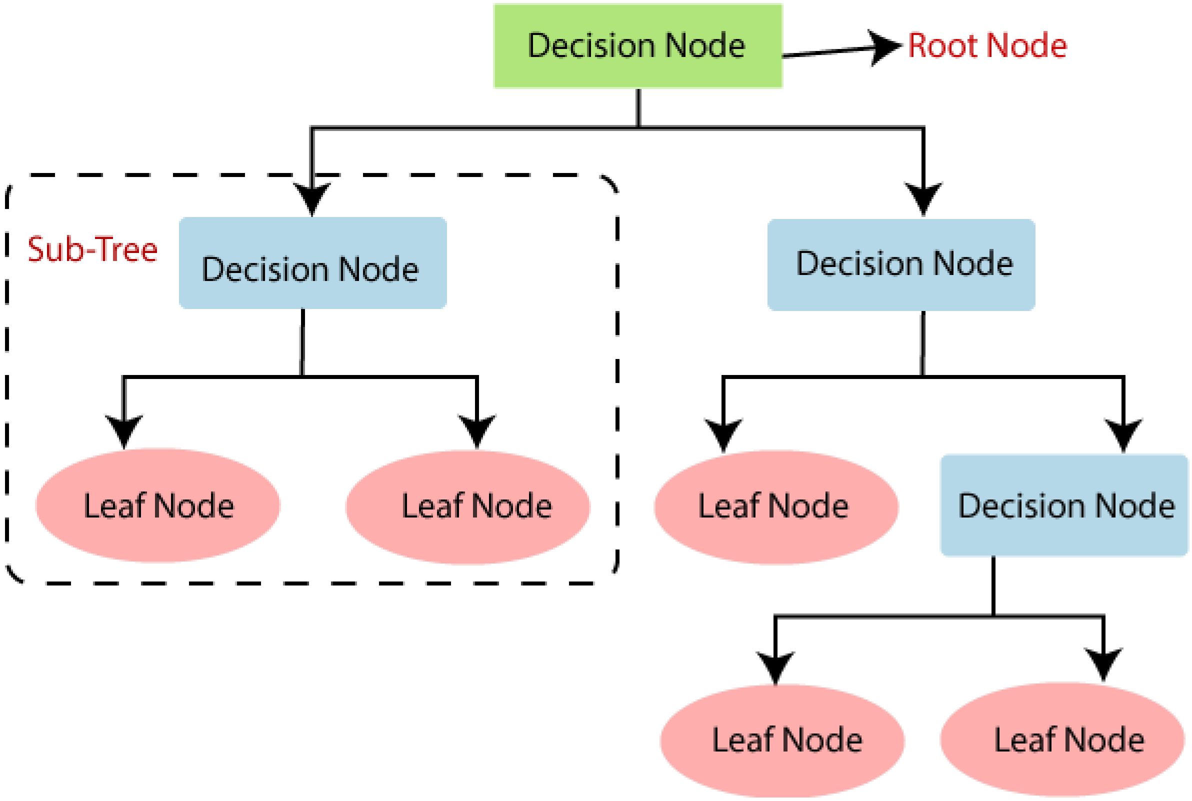
Editor: Citra Chairunnisa

Decision Tree

Decision Tree is a **Supervised learning technique** that can be used for both classification and Regression problems, but mostly it is preferred for solving **Classification** problems.

It is a tree-structured classifier, where

- **internal nodes represent the features of a dataset,**
- **branches represent the decision rules** and
- **each leaf node represents the outcome.**



https://www.javatpoint.com/machine-learning-decision-tree-classification-algorithm

How to construct a decision tree?

- There are 2 popular tree building-algorithm out there: **ID3** and **Classification and Regression Tree (CART)**. The main difference between these two models is the cost function that they use. The **cost function** decides which question to ask and how each node being split.
- The Decision Tree algorithm intuition is as follows:
  1. For each attribute in the dataset, the Decision-Tree algorithm forms a node. The most important attribute is placed at the root node.
  2. For evaluating the task in hand, we start at the root node and we work our way down the tree by following the corresponding node that meets our condition or decision.
  3. This process continues until a leaf node is reached. It contains the prediction or the outcome of the Decision Tree.

Attribute Selection

There are different attributes selection measure to identify the attribute which can be considered as the root node at each level. There are 2 popular attribute selection measures. They are as follows:

- **Information gain** in ID3 algorithm
- **Gini index** in CART algorithm

ID3 (Iterative Dichotomiser) Algorithm

The ID3 (Iterative Dichotomiser) Decision Tree algorithm uses entropy to calculate information gain.

Entropy measures the impurity in the given dataset. In Physics and Mathematics, entropy is referred to as the randomness or uncertainty of a random variable X. In information theory, it refers to the impurity in a group of examples.
Information gain is the decrease in entropy. Information gain computes the difference between entropy before split and average entropy after split of the dataset based on given attribute values.

So, by calculating decrease in **entropy measure** of each attribute we can calculate their information gain. The attribute with the highest information gain is chosen as the splitting attribute at the node.

Entropy

Entropy, also called as Shannon Entropy is denoted by H(S) for a finite set S, is the measure of the amount of uncertainty or randomness in data.

$$H(S) = \sum_{x \in X} p(x) \log_2 \frac{1}{p(x)}$$

Highest entropy is when there’s no way of determining what the outcome.
Consider a coin which has heads on both sides. Since we know beforehand that it’ll always be heads, this event has **no randomness** and its **entropy is zero**.

In other words, lower values imply less uncertainty while higher values imply high uncertainty.

Information Gain

Information Gain denoted by IG(S,A) for a set S is the effective change in entropy after deciding on a particular attribute A. It measures the relative change in entropy with respect to the independent variables.

$$IG(S,A) = H(S) - H(S,A)$$

$$IG(S,A) = H(S) - \sum_{i=0}^{n} P(x) * H(x)$$

where IG(S, A) is the information gain by applying feature A. H(S) is the Entropy of the entire set, while the second term calculates the Entropy after applying the feature A, where P(x) is the probability of event x.

ID3 Algorithm

Let’s understand the ID3 algorithm with the help of a simple example below.
Consider a piece of data collected over the course of 14 days where the features are Outlook, Temperature, Humidity, Wind and the outcome variable is whether Golf was played on the day.

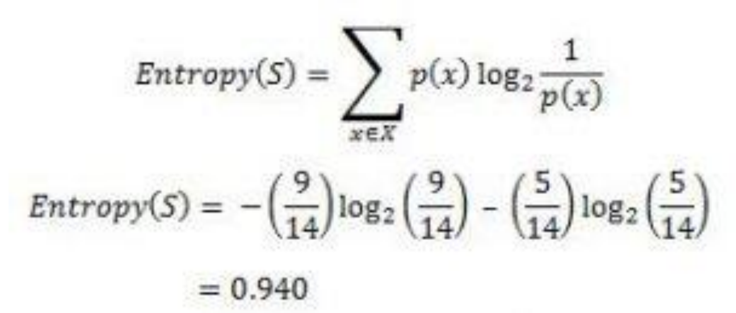
Day	Outlook	Temperature	Humidity	Wind	Play Golf
D1	Sunny	Hot	High	Weak	No
D2	Sunny	Hot	High	Strong	No
D3	Overcast	Hot	High	Weak	Yes
D4	Rain	Mild	High	Weak	Yes
D5	Rain	Cool	Normal	Weak	Yes
D6	Rain	Cool	Normal	Strong	No
D7	Overcast	Cool	Normal	Strong	Yes
D8	Sunny	Mild	High	Weak	No
D9	Sunny	Cool	Normal	Weak	Yes
D10	Rain	Mild	Normal	Weak	Yes
D11	Sunny	Mild	Normal	Strong	Yes
D12	Overcast	Mild	High	Strong	Yes
D13	Overcast	Hot	Normal	Weak	Yes
D14	Rain	Mild	High	Strong	No

ID3 Algorithm will perform following tasks recursively

1. Create root node for the tree
2. If all examples are positive, return leaf node ‘positive’
3. Else if all examples are negative, return leaf node ‘negative’
4. Calculate the entropy of current state H(S)
5. For each attribute, calculate the entropy with respect to the attribute ‘x’ denoted by H(S, x)
6. Select the attribute which has maximum value of IG(S, x)
7. Remove the attribute that offers highest IG from the set of attributes
8. Repeat until we run out of all attributes, or the decision tree has all leaf nodes.

Calculating Information Gain

In the example, we can see in total there are 5 No's and 9 Yes's.



$$Entropy(S) = \sum_{x \in X} p(x) \log_2 \frac{1}{p(x)}$$

$$Entropy(S) = -\left(\frac{9}{14}\right)\log_2\left(\frac{9}{14}\right) - \left(\frac{5}{14}\right)\log_2\left(\frac{5}{14}\right)$$

$$= 0.940$$

Remember that the Entropy is 0 if all members belong to the same class (no uncertainty), and 1 when half of them belong to one class and other half belong to other class (high randomness/high uncertainty). Here it's 0.94 which means the distribution is fairly random. **Now the next step is to choose the attribute that gives us highest possible Information Gain** which we'll choose as the root node.

Information Gain for an Attribute

Let's start with 'Wind'. Amongst all the 14 examples we have 8 places where the wind is weak and 6 where the wind is Strong.

$$P(S_{weak}) = \frac{Number\ of\ Weak}{Total}$$
$$= \frac{8}{14}$$
$$P(S_{strong}) = \frac{Number\ of\ Strong}{Total}$$
$$= \frac{6}{14}$$

Out of the 8 Weak examples, 6 of them were 'Yes' for Play Golf and 2 of them were 'No' for 'Play Golf'

$$Entropy(S_{weak}) = -\left(\frac{6}{8}\right)\log_2\left(\frac{6}{8}\right) - \left(\frac{2}{8}\right)\log_2\left(\frac{2}{8}\right)$$
$$= 0.811$$

Similarly, out of 6 Strong examples, we have 3 examples where the outcome was 'Yes' for Play Golf and 3 where we had 'No' for Play Golf (this is perfect randomness)

$$Entropy(S_{strong}) = -\left(\frac{3}{6}\right)\log_2\left(\frac{3}{6}\right) - \left(\frac{3}{6}\right)\log_2\left(\frac{3}{6}\right)$$
$$= 1.000$$

Now calculate the information gain

$$IG(S, Wind) = H(S) - \sum_{i=0}^{n} P(x) * H(x)$$

$$IG(S, Wind) = H(S) - P(S_{weak}) * H(S_{weak}) - P(S_{strong}) * H(S_{strong})$$
$$= 0.940 - \left(\frac{8}{14}\right)(0.811) - \left(\frac{6}{14}\right)(1.00)$$
$$= 0.048$$

Information Gain for All Attributes

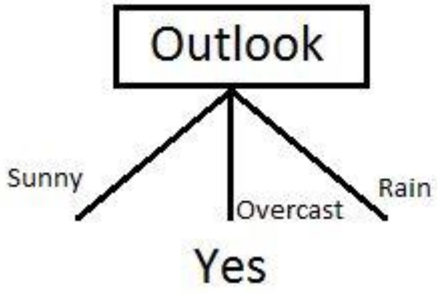
$$IG(S, Outlook) = 0.246$$

$$IG(S, Temperature) = 0.029$$

$$IG(S, Humidity) = 0.151$$

$$IG(S, Wind) = 0.048 \text{ (Previous example)}$$

We can clearly see that IG(S, Outlook) has the highest information gain of 0.246, **hence we chose Outlook attribute as the root node**. At this point, the decision tree looks like



Whenever the outlook is Overcast, Play Golf is always 'Yes', it's no coincidence by any chance, the simple tree resulted because of **the highest information gain is given by the attribute Outlook**. Now how do we proceed from this point? We can simply apply **recursion**, you might want to look at the algorithm steps described earlier.
Now that we've used Outlook, we've got three of them remaining **Humidity**, **Temperature**, and **Wind**. And, we had three possible values of Outlook: Sunny, Overcast, Rain. The Overcast node already ended up having leaf node 'Yes', so we're left with two subtrees to compute: **Sunny** and **Rain**.

Information Gain in All Subtrees

For Sunny Subtree, the table looks like this

Temperature	Humidity	Wind	Play Golf
Hot	High	Weak	No
Hot	High	Strong	No
Mild	High	Weak	No
Cool	Normal	Weak	Yes
Mild	Normal	Strong	Yes

We get **Humidity** as the highest information gain

For Rain Subtree, we will get **Wind** as the highest information gain (you can try calculate it by yourself ☺)

In the similar fashion, we compute the following values

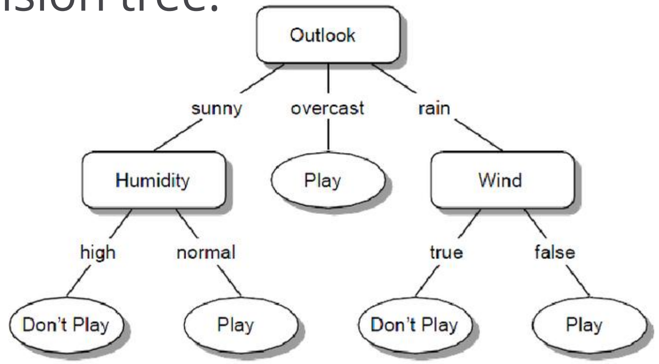
$$H(S_{sunny}) = \left(\frac{3}{5}\right)\log_2\left(\frac{3}{5}\right) - \left(\frac{2}{5}\right)\log_2\left(\frac{2}{5}\right) = 0.96$$

$$IG(S_{sunny}, Humidity) = 0.96$$

$$IG(S_{sunny}, Temperature) = 0.57$$

$$IG(S_{sunny}, Wind) = 0.019$$

Final decision tree:



Witten et al., 2011

CART Algorithm

Classification And Regression Trees (CART) algorithm is a classification algorithm for building a decision tree based on **Gini’s impurity index** as splitting criterion.

Gini Impurity

- Gini impurity is calculated using following formula

$$GiniIndex = 1 - \sum_j p_j^2$$

for j = 1 to number of classes

- Gini impurity measures the frequency at which any element of the dataset will be mislabelled when it is randomly labeled.
- The minimum value of the Gini Index is 0. This happens when the node is **pure**, this means that all the contained elements in the node are of one unique class. Therefore, this node will not be split again. Thus, the optimum split is chosen by the features with less Gini Index. Moreover, it gets the maximum value when the probability of the two classes are the same.

$$Gini_{min} = 1 - (1^2) = 0$$

$$Gini_{max} = 1 - (0.5^2 + 0.5^2) = 0.5$$

Gini Index (for Outlook feature)

Let’s understand the CART algorithm with the help of golf playing decision dataset.
Note that number of classes is 2 for ‘No’ = not playing golf and ‘Yes’ = playing golf.
Outlook consists of 3 values (Sunny, Overcast, Rain)

Outlook	Yes	No	Number of instances
Sunny	2	3	5
Overcast	4	0	4
Rain	3	2	5

Gini(Outlook=Sunny) = $1 – (2/5)^2 – (3/5)^2 = 1 – 0.16 – 0.36 = 0.48$
Gini(Outlook=Overcast) = $1 – (4/4)^2 – (0/4)^2 = 0$
Gini(Outlook=Rain) = $1 – (3/5)^2 – (2/5)^2 = 1 – 0.36 – 0.16 = 0.48$
Then, weighted sum of Gini indexes for outlook feature:
Gini(Outlook) = (5/14) x 0.48 + (4/14) x 0 + (5/14) x 0.48 = 0.171 + 0 + 0.171 = 0.342

Gini Index (for Humidity feature)

Humidity is a binary class feature and has 2 values (High and Normal)

Humidity	Yes	No	Number of instances
High	3	4	7
Normal	6	1	7

Gini(Humidity=High) = 1 – $(3/7)^2$ – $(4/7)^2$ = 1 – 0.183 – 0.326 = 0.489
Gini(Humidity=Normal) = 1 – $(6/7)^2$ – $(1/7)^2$ = 1 – 0.734 – 0.02 = 0.244
Weighted sum for humidity feature will be calculated next
Gini(Humidity) = (7/14) x 0.489 + (7/14) x 0.244 = 0.367

Your Turn
Can you calculate Gini index for the next features, i.e., Temperature and Wind?

Gini Index (for Temperature feature)

Temperature	Yes	No	Number of instances
Hot	2	2	4
Cool	3	1	4
Mild	4	2	6

Gini(Temp=Hot) = 1 – $(2/4)^2$ – $(2/4)^2$ = 0.5
Gini(Temp=Cool) = 1 – $(3/4)^2$ – $(1/4)^2$ = 1 – 0.5625 – 0.0625 = 0.375
Gini(Temp=Mild) = 1 – $(4/6)^2$ – $(2/6)^2$ = 1 – 0.444 – 0.111 = 0.445
We'll calculate weighted sum of gini index for temperature feature
Gini(Temp) = (4/14) x 0.5 + (4/14) x 0.375 + (6/14) x 0.445 = 0.142 + 0.107 + 0.190 = 0.439

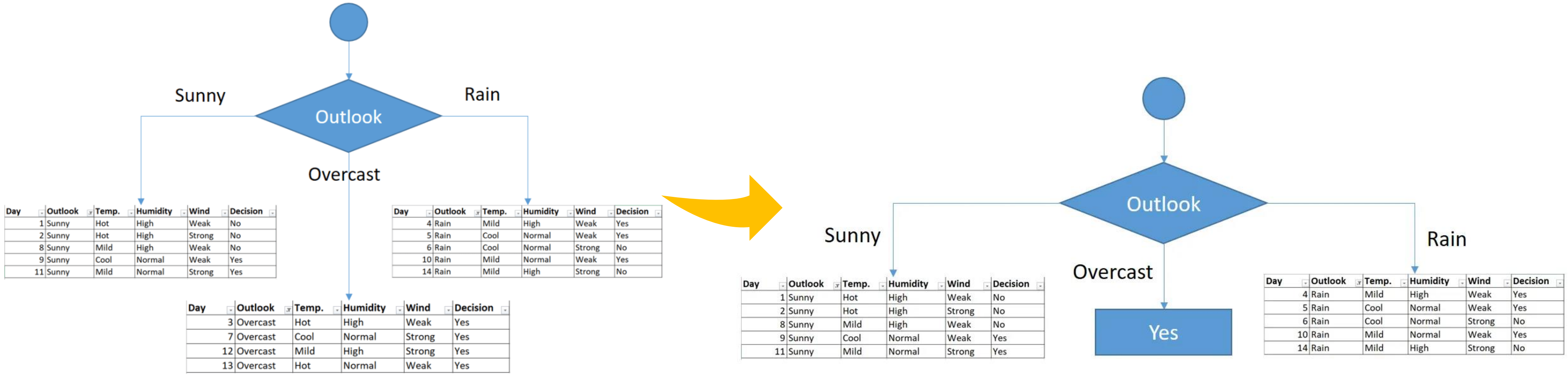
Gini Index (for Wind feature)

Wind	Yes	No	Number of instances
Weak	6	2	8
Strong	3	3	6

Gini(Wind=Weak) = 1 – $(6/8)^2$ – $(2/8)^2$ = 1 – 0.5625 – 0.062 = 0.375
Gini(Wind=Strong) = 1 – $(3/6)^2$ – $(3/6)^2$ = 1 – 0.25 – 0.25 = 0.5
Gini(Wind) = (8/14) x 0.375 + (6/14) x 0.5 = 0.428

Deciding Root Node

Gini (Outlook) = 0.342
Gini (Temperature) = 0.439
Gini (Humidity) = 0.367
Gini (Wind) = 0.428
We see Outlook has the lowest index and put it at the top of the tree



Day	Outlook	Temp.	Humidity	Wind	Decision
1	Sunny	Hot	High	Weak	No
2	Sunny	Hot	High	Strong	No
8	Sunny	Mild	High	Weak	No
9	Sunny	Cool	Normal	Weak	Yes
11	Sunny	Mild	Normal	Strong	Yes

Day	Outlook	Temp.	Humidity	Wind	Decision
4	Rain	Mild	High	Weak	Yes
5	Rain	Cool	Normal	Weak	Yes
6	Rain	Cool	Normal	Strong	No
10	Rain	Mild	Normal	Weak	Yes
14	Rain	Mild	High	Strong	No

Day	Outlook	Temp.	Humidity	Wind	Decision
3	Overcast	Hot	High	Weak	Yes
7	Overcast	Cool	Normal	Strong	Yes
12	Overcast	Mild	High	Strong	Yes
13	Overcast	Hot	Normal	Weak	Yes

Day	Outlook	Temp.	Humidity	Wind	Decision
1	Sunny	Hot	High	Weak	No
2	Sunny	Hot	High	Strong	No
8	Sunny	Mild	High	Weak	No
9	Sunny	Cool	Normal	Weak	Yes
11	Sunny	Mild	Normal	Strong	Yes

Day	Outlook	Temp.	Humidity	Wind	Decision
4	Rain	Mild	High	Weak	Yes
5	Rain	Cool	Normal	Weak	Yes
6	Rain	Cool	Normal	Strong	No
10	Rain	Mild	Normal	Weak	Yes
14	Rain	Mild	High	Strong	No

Decision for Sunny Sub Dataset

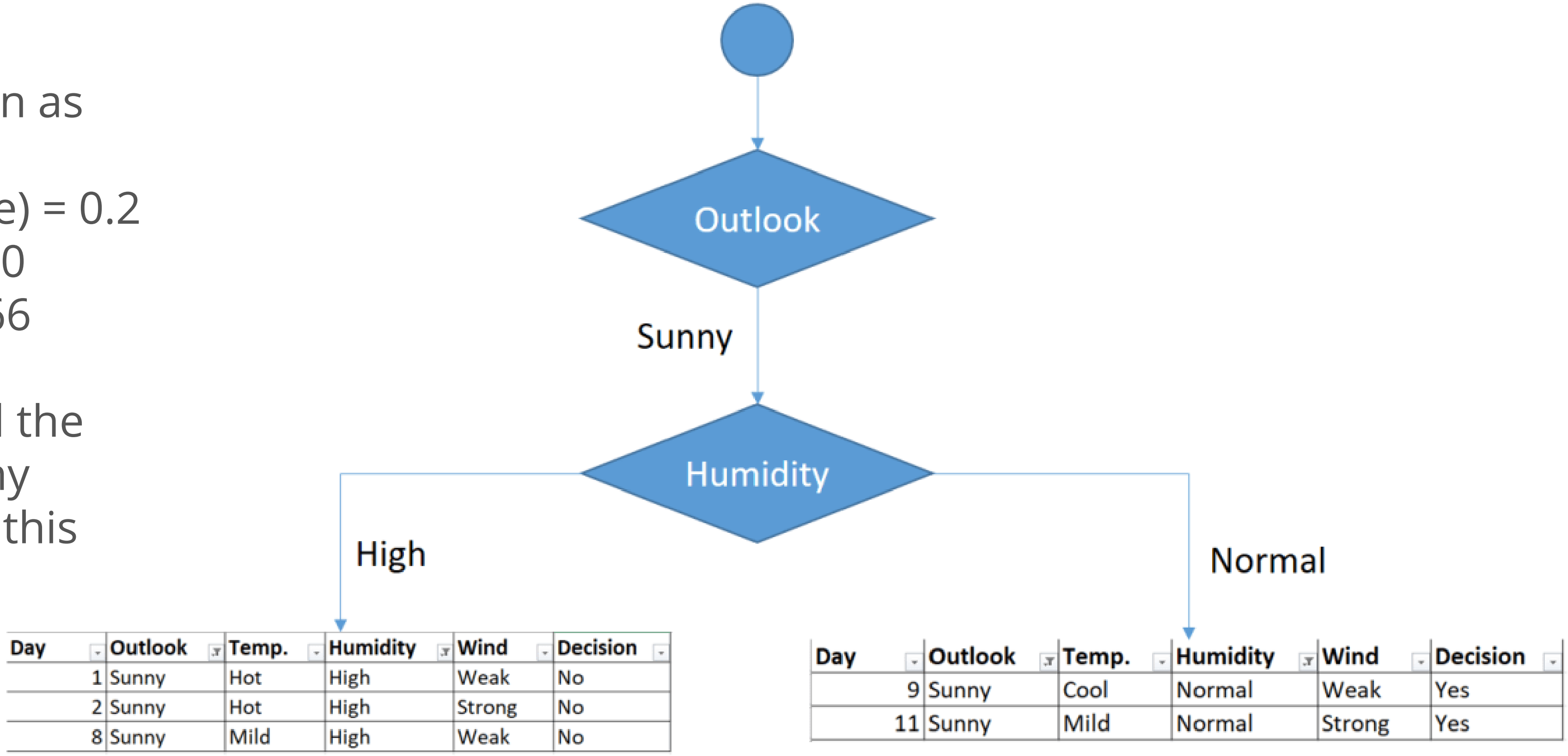
Next, we will apply same principles to the next sub datasets
This one is for Sunny Outlook

Day	Outlook	Temp.	Humidity	Wind	Decision
1	Sunny	Hot	High	Weak	No
2	Sunny	Hot	High	Strong	No
8	Sunny	Mild	High	Weak	No
9	Sunny	Cool	Normal	Weak	Yes
11	Sunny	Mild	Normal	Strong	Yes

Decision for Sunny Sub Dataset

With the same calculation as before you'll get
Gini (Sunny, Temperature) = 0.2
Gini (Sunny, Humidity) = 0
Gini (Sunny, Wind) = 0.466

We choose humidity and the decision tree of the sunny outlook will become like this



Day	Outlook	Temp.	Humidity	Wind	Decision
1	Sunny	Hot	High	Weak	No
2	Sunny	Hot	High	Strong	No
8	Sunny	Mild	High	Weak	No

Day	Outlook	Temp.	Humidity	Wind	Decision
9	Sunny	Cool	Normal	Weak	Yes
11	Sunny	Mild	Normal	Strong	Yes

As seen, decision is always no for high humidity and yes for normal humidity. So, this branch is over.

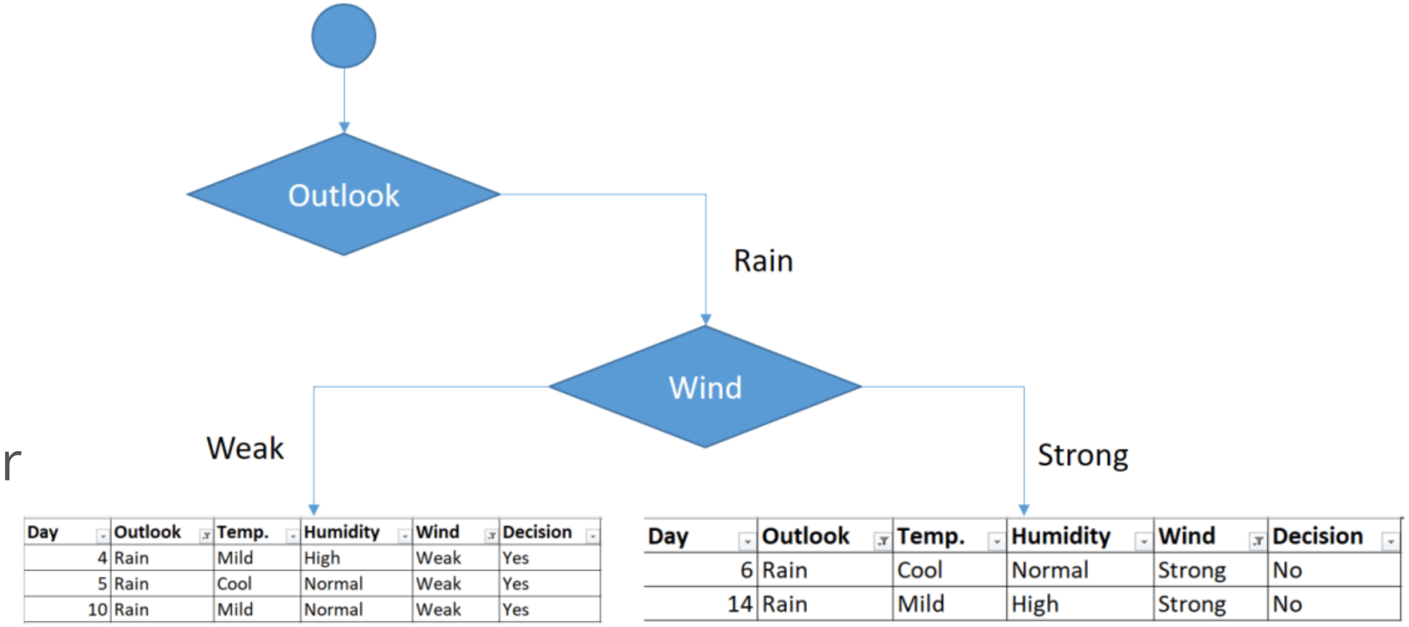
Decision for Rain Sub Dataset

Can you calculate the Gini indexes for the Rain Outlook branch?
Below is the dataset for Sunny Outlook

Day	Outlook	Temp.	Humidity	Wind	Decision
4	Rain	Mild	High	Weak	Yes
5	Rain	Cool	Normal	Weak	Yes
6	Rain	Cool	Normal	Strong	No
10	Rain	Mild	Normal	Weak	Yes
14	Rain	Mild	High	Strong	No

Decision for Rain Sub Dataset

With the same calculation as before you'll get
Gini (Rain, Temperature) = 4.66
Gini (Rain, Humidity) = 4.66
Gini (Rain, Wind) = 0

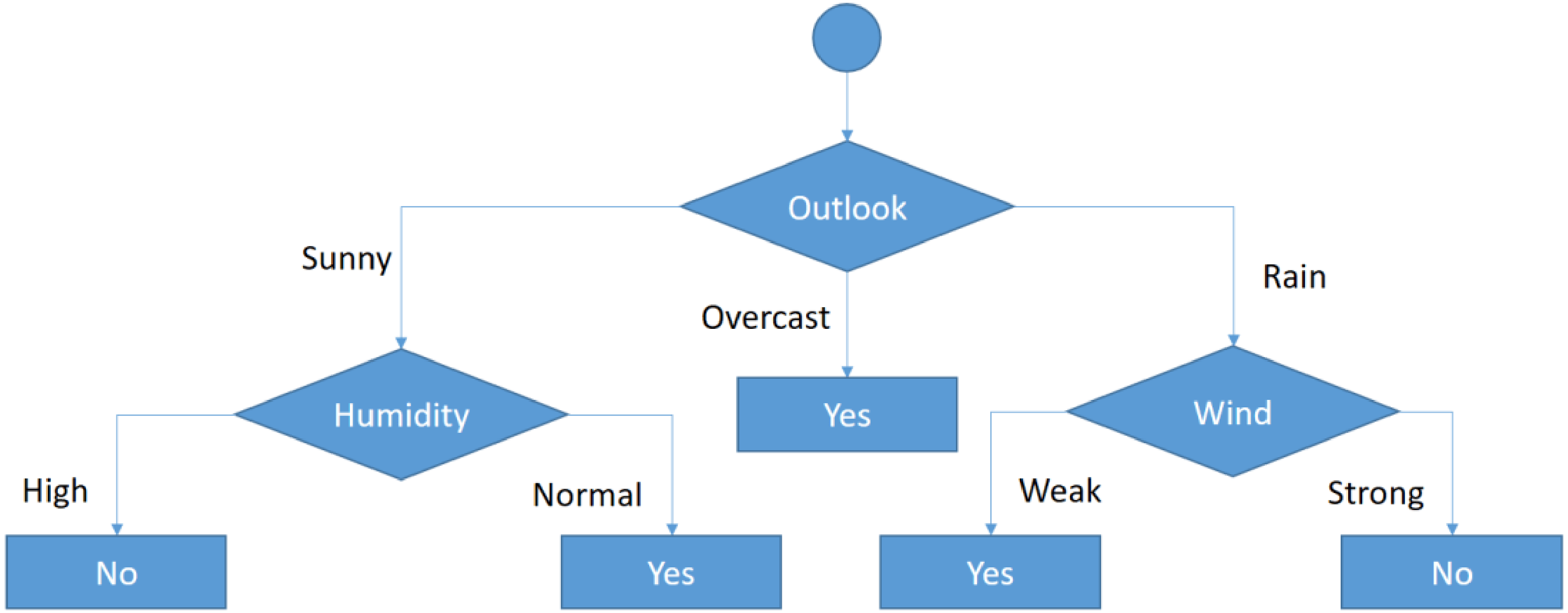


Day	Outlook	Temp.	Humidity	Wind	Decision
4	Rain	Mild	High	Weak	Yes
5	Rain	Cool	Normal	Weak	Yes
10	Rain	Mild	Normal	Weak	Yes

Day	Outlook	Temp.	Humidity	Wind	Decision
6	Rain	Cool	Normal	Strong	No
14	Rain	Mild	High	Strong	No

As seen, decision is always yes when wind is weak. On the other hand, decision is always no if wind is strong. This means that this branch is over.

Final Form of Decision Tree built by CART Algorithm



You might realize that we've created exactly the same tree as in ID3 example. This does not mean that ID3 and CART algorithms always produce same trees. This simple example fortunately generates the same tree. But it's not always the case. For more complex system, both algorithms might result in different trees.

Some of Decision Tree Advantages

1. It can capture nonlinear relationships: They can be used to classify non-linearly separable data.
2. Easy to understand, interpret, visualize.
3. Compared to other algorithms decision trees requires less effort for data preparation during pre-processing.
4. A decision tree does not require normalization of data.
5. A decision tree does not require scaling of data as well.
6. Missing values in the data also do NOT affect the process of building a decision tree to any considerable extent.
7. A Decision tree model is very intuitive and easy to explain to technical teams as well as stakeholders.

Some of Decision Tree Disadvantages

1. A small change in the data can cause a large change in the structure of the decision tree causing **instability**.
2. Adding a new data point can lead to re-generation of the overall tree and all nodes need to be recalculated and recreated.
3. It can’t be used in big data: If the size of data is too big, then one single tree may grow a lot of nodes which might result in complexity and leads to **overfitting**.

Exercise

Let’s build decision tree model to classify car acceptability!

We’ll be using **car.data** dataset

Your Turn

Let's build decision tree to classify survived and not survived passengers using all features in **Titanic_Train.csv** dataset

1. Target variable is the first column named **Survived**
2. Don't forget to split dataset into training and testing sets!
3. **Titanic_Train** consists of encoded features, so don't bother to preprocess the data, just start building the model ☺