Decision Trees: A Complete Introduction With Examples

Shubham Koli · Follow

10 min read · Feb 27

Decision Trees

A decision tree is a **non-parametric** supervised learning algorithm. It has a hierarchical, tree structure, which consists of a root node, branches, internal nodes and leaf nodes.

Decision Trees are the foundation for many classical machine learning algorithms like **Random Forests, Bagging**, and **Boosted** Decision Trees. His idea was to represent data as a tree where each internal node denotes a test on an attribute (basically a condition), each branch represents an outcome of the test, and each leaf node (terminal node) holds a class label.

Types of Decision Trees

1. **CART** (Classification and Regression Trees) → uses ***Gini Index(Classification)*** as metric.

2. **ID3** (Iterative Dichotomiser 3) → uses ***Entropy function*** and ***Information gain*** as metrics.

Before learning more about decision trees let's get familiar with some of the terminologies.

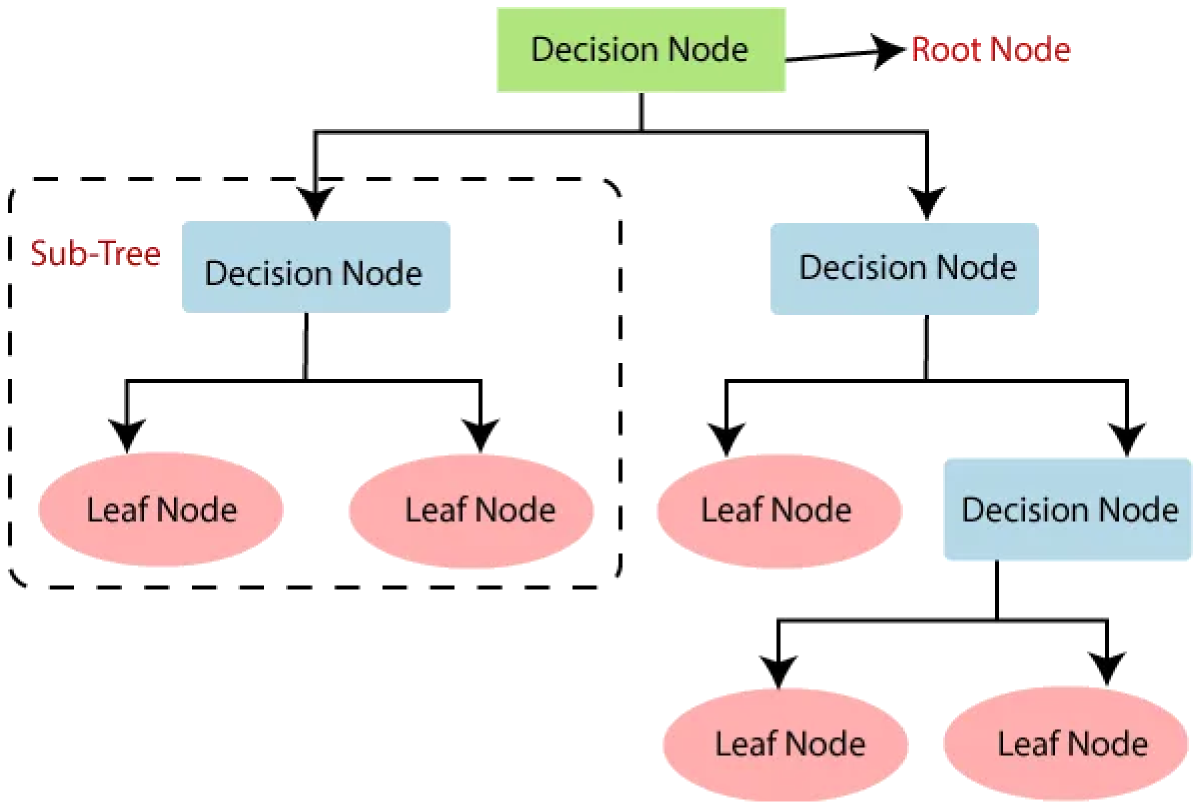
Root Nodes — It is the node present at the beginning of a decision tree from this node the population starts dividing according to various features.

Decision Nodes — the nodes we get after splitting the root nodes are called Decision Node

Leaf Nodes — the nodes where further splitting is not possible are called leaf nodes or terminal nodes

Branch/Sub-tree — just like a small portion of a graph is called sub-graph similarly a sub-section of this decision tree is called sub-tree.

Pruning — is nothing but cutting down some nodes to stop overfitting.



Why Use Decision Trees?

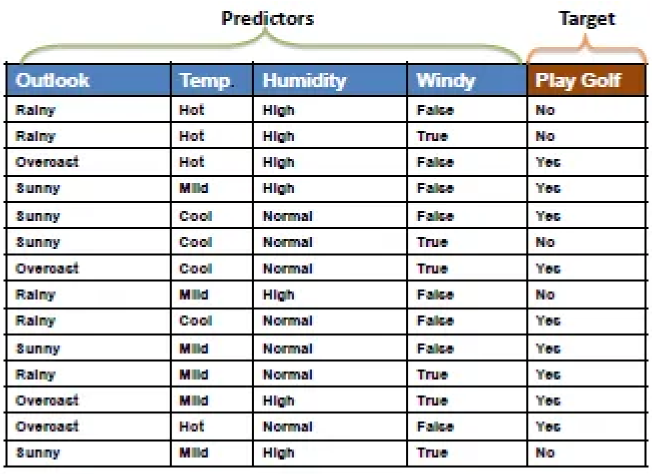
There are various algorithms in Machine learning. Below are the two reasons for using the Decision tree:

- *Decision Trees usually mimic human thinking ability while making a decision, so it is easy to understand.*
- *The logic behind the decision tree can be easily understood because it shows a tree-like structure.*

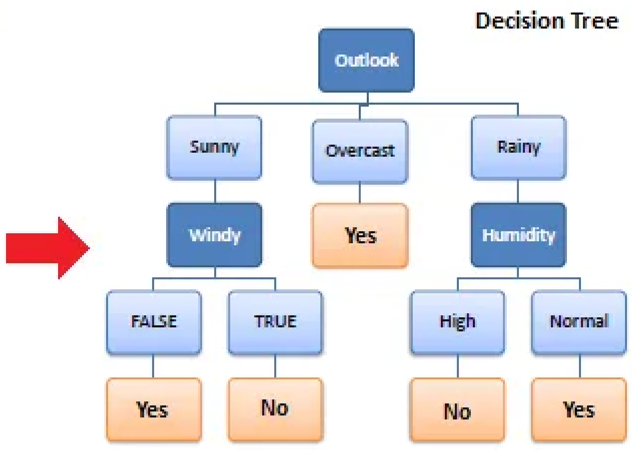
Example of a decision tree

Let’s understand decision trees with the help of an example.

Decision tree builds classification or regression models in the form of a tree structure. It breaks down a dataset into smaller and smaller subsets while at the same time an associated decision tree is incrementally developed. The final result is a tree with **decision nodes** and **leaf nodes**. A decision node (e.g., Outlook) has two or more branches (e.g., Sunny, Overcast and Rainy). Leaf node (e.g., Play) represents a classification or decision. The topmost decision node in a tree which corresponds to the best predictor called **root node**. Decision trees can handle both categorical and numerical data.



Predictors				Target
Outlook	Temp	Humidity	Windy	Play Golf
Rainy	Hot	High	False	No
Rainy	Hot	High	True	No
Overcast	Hot	High	False	Yes
Sunny	Mild	High	False	Yes
Sunny	Cool	Normal	False	Yes
Sunny	Cool	Normal	True	No
Overcast	Cool	Normal	True	Yes
Rainy	Mild	High	False	No
Rainy	Cool	Normal	False	Yes
Sunny	Mild	Normal	False	Yes
Rainy	Mild	Normal	True	Yes
Overcast	Mild	High	True	Yes
Overcast	Hot	Normal	False	Yes
Sunny	Mild	High	True	No



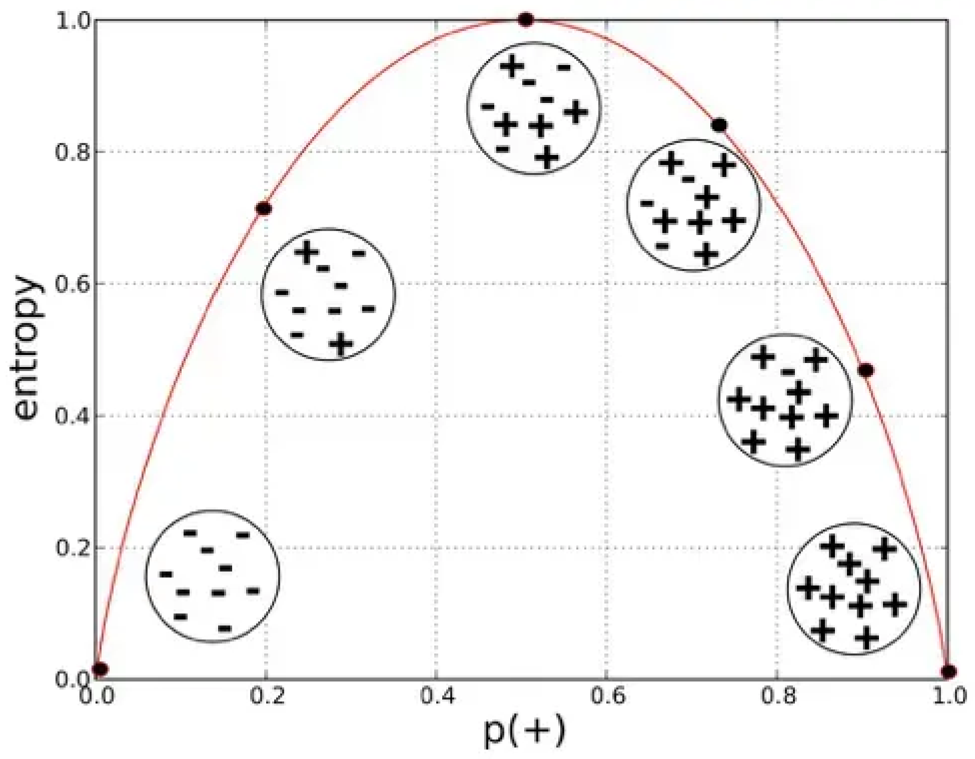
> *Decision trees are upside down which means the root is at the top and then this root is split into various several nodes. Decision trees are nothing but a bunch of if-else statements in layman terms. It checks if the condition is true and if it is then it goes to the next node attached to that decision.*

Entropy

Entropy is a metric to measure the impurity in a given attribute. It specifies randomness in data.

Entropy comes from information theory. The higher the entropy the more the information content.

ID3 algorithm uses entropy to calculate the homogeneity of a sample. If the sample is completely homogeneous the entropy is zero and if the sample is an equally divided it has entropy of one.



H(X) = — Σ (pi * log2 pi)

H(X) = — Σ (pi * log2 pi)

> *Where,*
>
> *X = Total number of samples*
>
> pi *is the probability of class i*

Gini Index:

- Gini index is a measure of impurity or purity used while creating a decision tree in the **CART**(*Classification and Regression Tree*) algorithm.
- An attribute with the low Gini index should be preferred as compared to the high Gini index.
- It only creates binary splits, and the CART algorithm uses the Gini index to create binary splits.
- Gini index can be calculated using the below formula:

$$Gini = 1 - \sum_{i=1}^{n} (p_i)^2$$

Where,

pi is the probability of a particular element belonging to a specific class.

To build a decision tree, we need to calculate two types of entropy using frequency tables as follows:

a) 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

b) 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

T is the output attribute,

X is the input attribute,

> *P(c) is the probability w.r.t the possible data point present at X, and*
>
> *E(c) is the entropy w.r.t 'True' pertaining to the possible data point.*

Information Gain

- We want to determine **which attribute** in a given set of training feature vectors is **most useful** for discriminating between the classes to be learned.
- **Information gain** tells us how important a given attribute of the feature vectors is.
- We will use it to decide the ordering of attributes in the nodes of a decision tree.

Information Gain = entropy (parent) — [average entropy (children)]

$$IG(T,A) = Entropy(T) - \sum_{v \epsilon A} \frac{|T_v|}{T} \cdot Entropy(T_v)$$

Step 1: Calculate entropy of the target.

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

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			

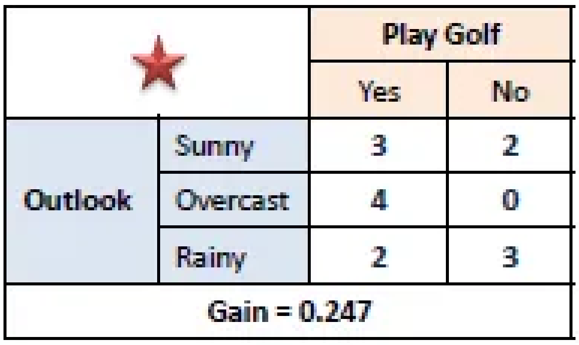
		Play Golf	
		Yes	No
Windy	False	6	2
	True	3	3
Gain = 0.048			

$$Gain(T,X) = Entropy(T) - Entropy(T,X)$$

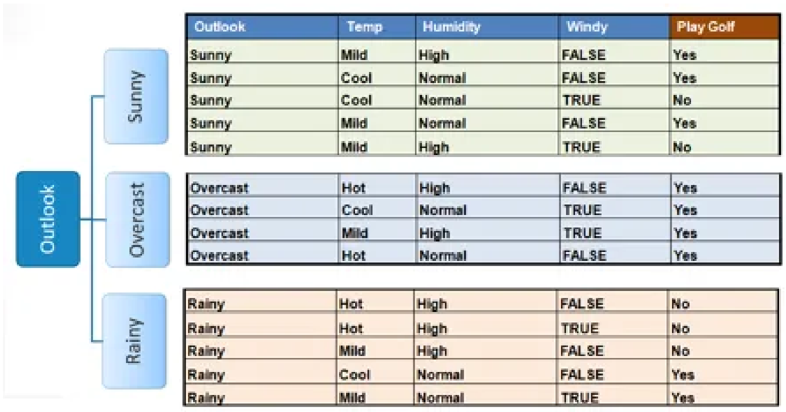
G(PlayGolf, Outlook) = **E**(PlayGolf) – **E**(PlayGolf, Outlook)

= 0.940 – 0.693 = 0.247

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.

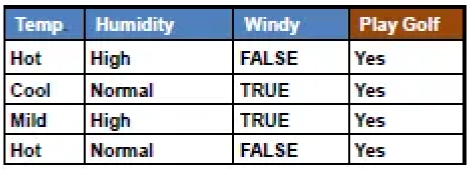


		Play Golf	
		Yes	No
Outlook	Sunny	3	2
	Overcast	4	0
	Rainy	2	3
Gain = 0.247			

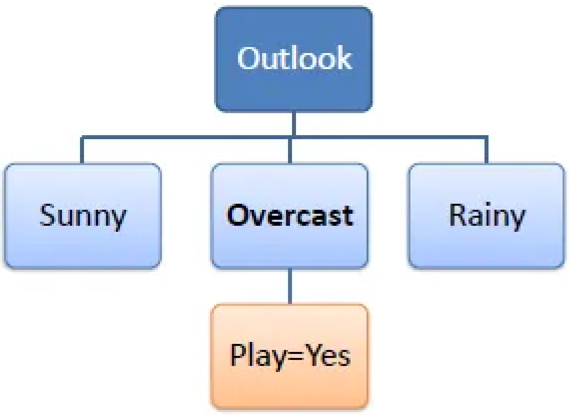


Outlook	Temp	Humidity	Windy	Play Golf
Sunny	Mild	High	FALSE	Yes
Sunny	Cool	Normal	FALSE	Yes
Sunny	Cool	Normal	TRUE	No
Sunny	Mild	Normal	FALSE	Yes
Sunny	Mild	High	TRUE	No
Overcast	Hot	High	FALSE	Yes
Overcast	Cool	Normal	TRUE	Yes
Overcast	Mild	High	TRUE	Yes
Overcast	Hot	Normal	FALSE	Yes
Rainy	Hot	High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

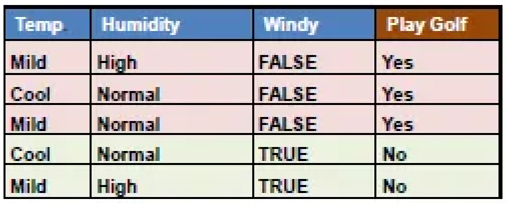
Step 4a : A branch with entropy of 0 is a leaf node.



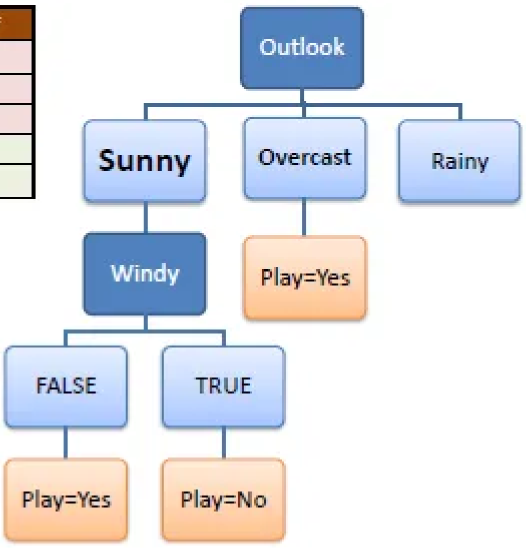
Temp	Humidity	Windy	Play Golf
Hot	High	FALSE	Yes
Cool	Normal	TRUE	Yes
Mild	High	TRUE	Yes
Hot	Normal	FALSE	Yes



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



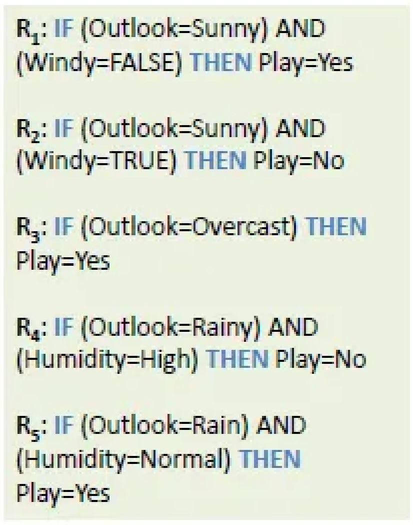
Step 5: The ID3 algorithm is run recursively on the non-leaf branches, until all data is classified.

Points to remember:

- A leaf node is the one that has no entropy, or when the entropy is zero. No further splitting is done on a leaf node.
- Only the branch that needs further splitting, i.e. when the entropy > 0 (when there's impurity) needs to undergo this splitting process.

Decision Tree to Decision Rules

A decision tree can easily be transformed to a set of rules by mapping from the root node to the leaf nodes one by one.



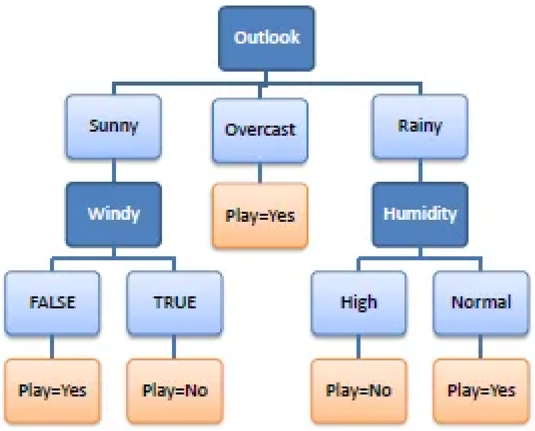
R_1: IF (Outlook=Sunny) AND (Windy=FALSE) THEN Play=Yes

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



Pruning: Getting an Optimal Decision tree

Pruning is a process of deleting the unnecessary nodes from a tree in order to get the optimal decision tree.

A too-large tree increases the risk of **overfitting**, and a small tree may not capture all the important features of the dataset. Therefore, a technique that decreases the size of the learning tree without reducing accuracy is known as Pruning. There are mainly two types of tree **pruning** technology used:

- **Cost Complexity Pruning**
- **Reduced Error Pruning.**

Applications of Decision Trees

Decision Tree is one of the basic and widely-used algorithms in the fields of Machine Learning. It's put into use across different areas in classification and regression modeling. Due to its ability to depict visualized output, one can easily draw insights from the modeling process flow. Here are a few examples wherein Decision Tree could be used,

- Business Management
- Customer Relationship Management
- Fraudulent Statement Detection
- Energy Consumption
- Healthcare Management
- Fault Diagnosis

Advantages:

1. Compared to other algorithms decision trees requires less effort for data preparation during pre-processing.
2. A decision tree does not require normalization of data.
3. A decision tree does not require scaling of data as well.
4. Missing values in the data also do NOT affect the process of building a decision tree to any considerable extent.

5. A Decision tree model is very intuitive and easy to explain to technical teams as well as stakeholders.

Disadvantage:

1. A small change in the data can cause a large change in the structure of the decision tree causing instability.
2. The decision tree contains lots of layers, which makes it complex.
3. Decision tree often involves higher time to train the model.
4. It may have an overfitting issue, which can be resolved using the **Random Forest algorithm.**

Decoding the Hyperparameters

Scikit-learn provides some functionalities or parameters that are to be used with a Decision Tree classifier to enhance the model's accuracy in accordance with the given data.

- **criterion**: This parameter is used to measure the quality of the split. The default value for this parameter is set to "Gini". If you want the measure to be calculated by entropy gain, you can change this parameter to "entropy".
- **splitter**: This parameter is used to choose the split at each node. If you want the sub-trees to have the best split, you can set this parameter to "best". We can also have a random split for which the value "random" is set.
- **max-depth**: This is an integer parameter through which we can limit the depth of the tree. The default value for this parameter is set to None.
- **min_samples_split**: This parameter is used to define the minimum number of samples required to split an internal node.
- **max_leaf_nodes**: The default value of max_leaf_nodes is set to None. This parameter is used to grow a tree with max_leaf_nodes in best-first fashion.

Python Implementation of Decision Tree

Step 1: Importing the Modules

The first and foremost step in building our decision tree model is to import the necessary packages and modules.

```
from sklearn.datasets import *
from sklearn.tree import DecisionTreeClassifier, DecisionTreeRegressor

import dtreeviz
import graphviz.backend as be
from IPython.display import Image, display_svg, SVG
import matplotlib.pyplot as plt
from sklearn.tree import plot_tree

import numpy as np
import warnings
warnings.filterwarnings('ignore')
%matplotlib inline
```

Step 2: Exploring the data

Next, we make our data ready by loading it from the datasets package using the load_iris() method.

```
iris = load_iris()

X_train = iris.data
y_train = iris.target
clas.fit(X_train, y_train)
```

Step 3: Create a decision tree classifier object & Fitting the Model

Here, we load the DecisionTreeClassifier in a variable named model, which was imported earlier from the sklearn package.This method is to fit the data by training the model on features and target.

```
decisiontree = DecisionTreeClassifier(  )
clas.fit(X_train, y_train)
```

Step 4: Making the Predictions

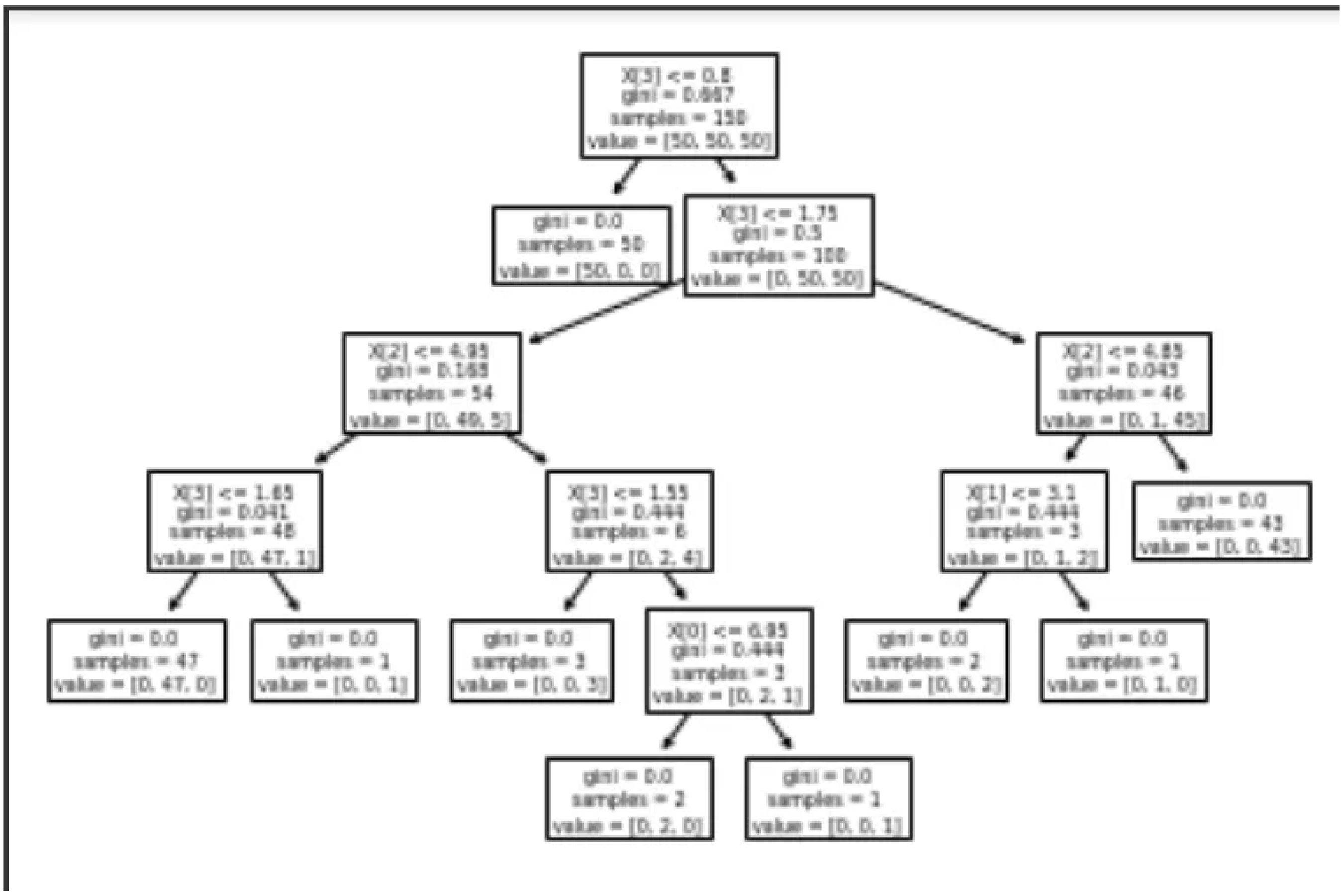
In this step, we take a sample observation and make a prediction. We create a new list comprising the flower sepal and petal dimensions. Further,

```
observation = [[ 5, 4, 3, 2]] # Predict observation's class
clas.predict(observation)
clas.predict_proba(observation)
```

```
Output:
array([1])
array([[0., 1., 0.]])
```

Step 5: Plot Graph

```
plot_tree(clas)
```

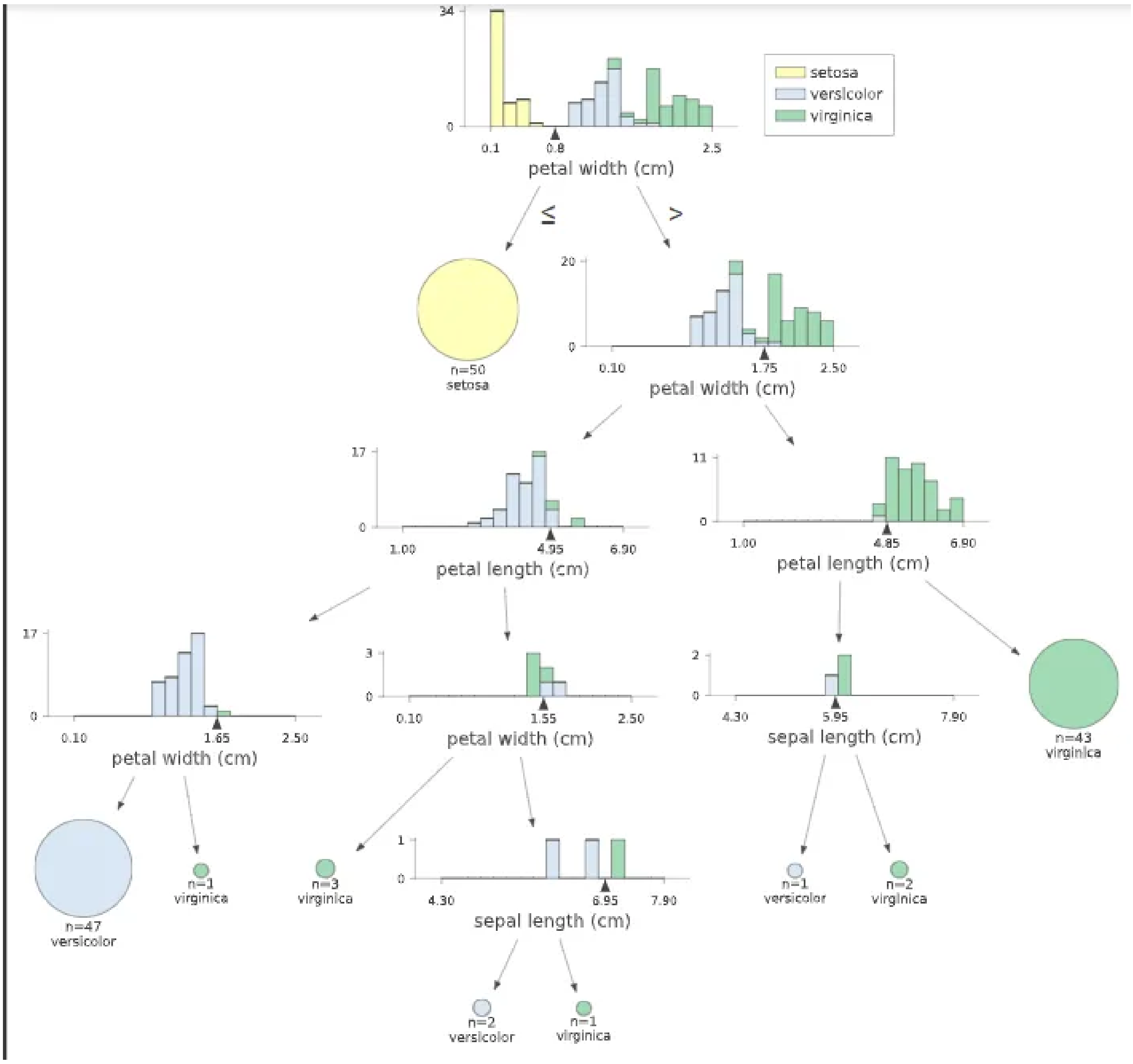


Step 6: Drawing the Graph

In the last step, we visualize the decision tree using an Image class that is to be imported from the dtreeviz package.

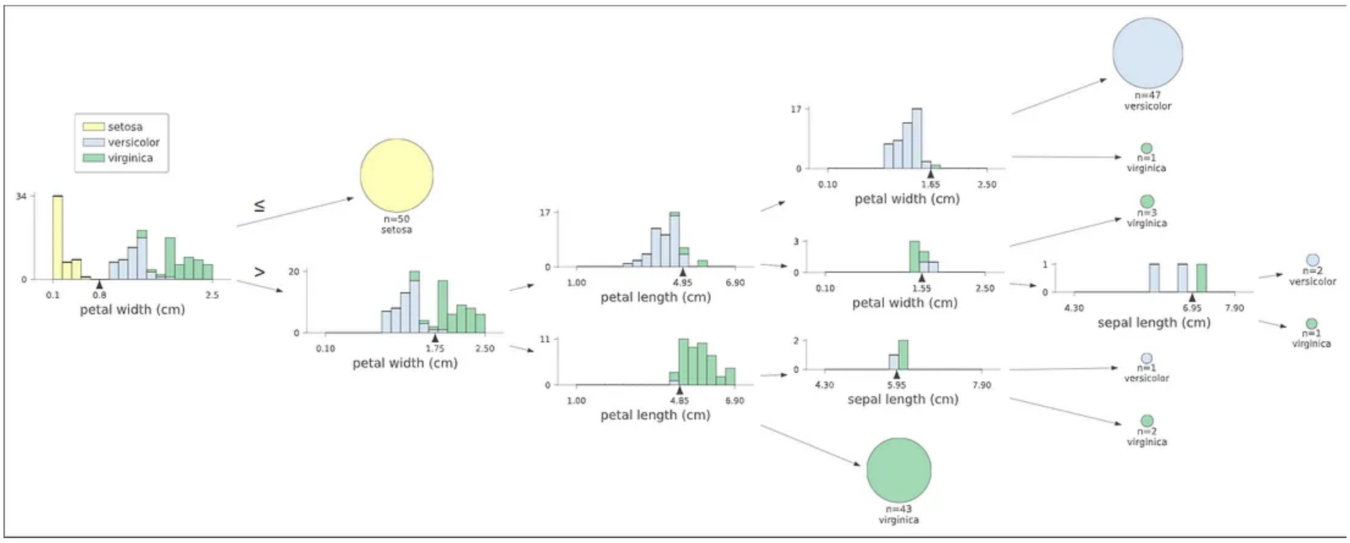
```
viz = dtreeviz.model(clas, X_train, y_train, feature_names=iris.feature_names,
                     class_names=["setosa", "versicolor", "virginica"])
viz
```

```
viz.view(scale=1)
```



To change the visualization, you can pass parameters, such as changing the orientation to left-to-right:

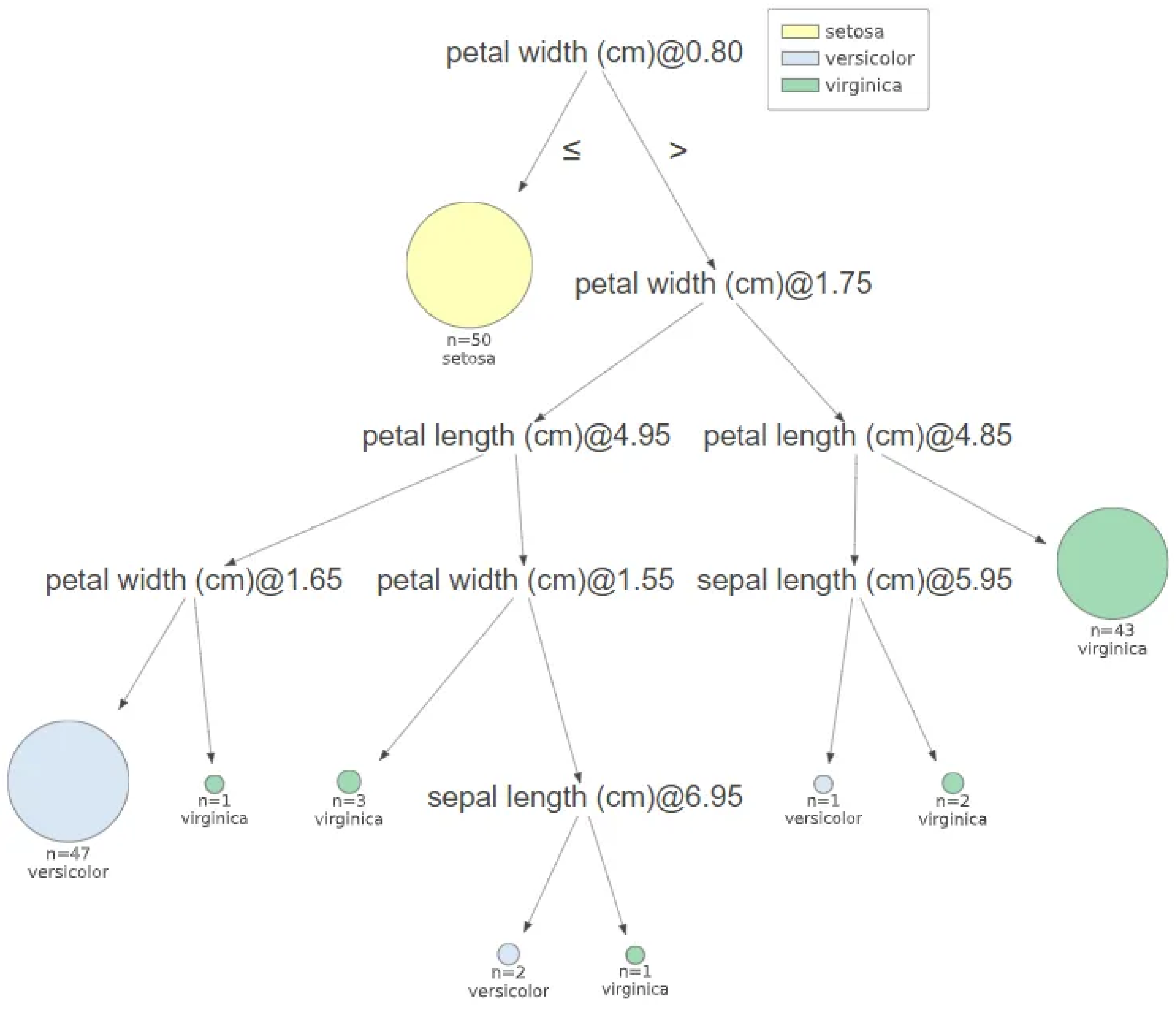
```
viz.view(orientation="LR")
```



Without Any graphs

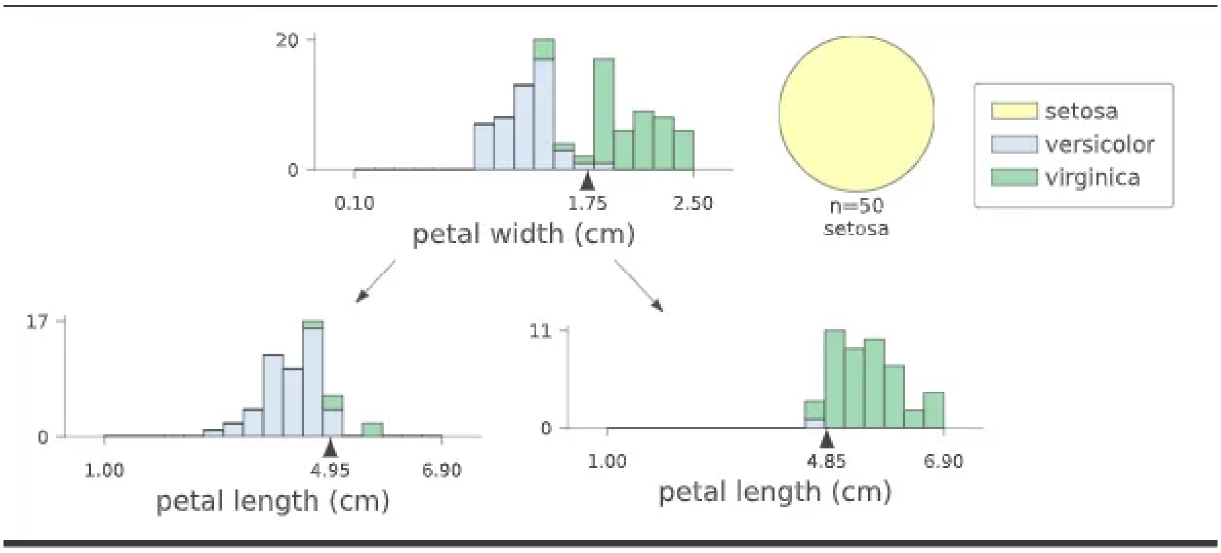
To visualize larger trees, you can reduce the amount of detail by turning off the fancy view:

```
viz.view(fancy=False)
```



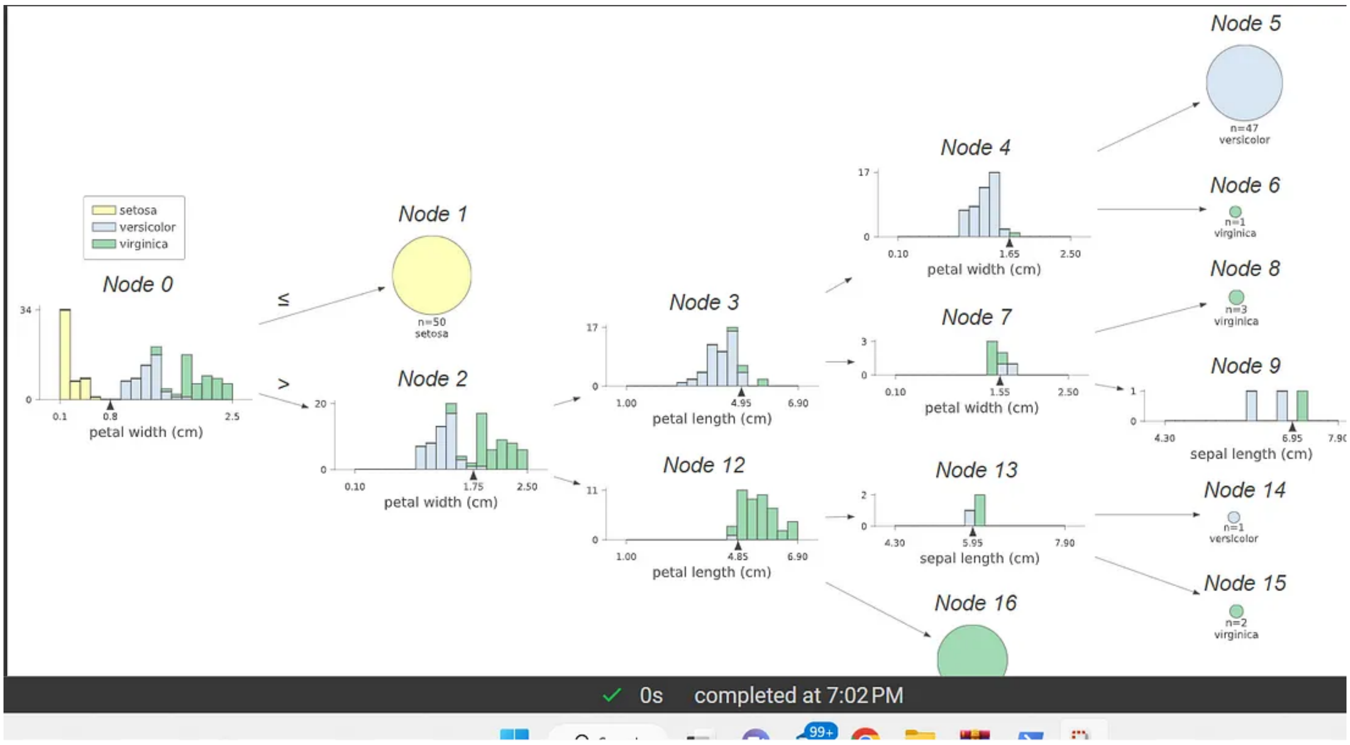
Another way to reduce the visualization size is to specify the tree depths of interest:

```
viz.view(depth_range_to_display=(1, 2)) # root is level 0
```



Show node number

```
viz.view(show_node_labels=True,orientation='LR')
```



Summary and Conclusion

In this article, we've discussed in-depth the Decision Tree algorithm. It's a supervised learning algorithm that can be used for both classification and regression. The primary goal of decision tree is to split the dataset as a tree based on a set of rules and conditions. We discussed the key components of a decision tree like the root node, leaf nodes, sub-trees, splitting, and pruning. Further, we've seen how a decision tree works and how strategic splitting is performed using popular algorithms like GINI, Information Gain. Furthermore, we used scikit-learn to code decision trees from scratch on the IRIS data set. Lastly, we discussed the advantages and disadvantages of using decision trees. There is still a lot more to learn, and this article will give you a quick-start to explore other advanced classification algorithms.

👉 **Here is a link to check out the Github.** 📁

End Notes:

> If you **liked** this **post**, share with your interest group, **friends** and colleagues. **Comment** down your thoughts, opinions and feedback below. I would **love** to hear from you. Do **follow me** for more such articles and **motivating** me 😀 .

It doesn't cost you anything to clap. 👏