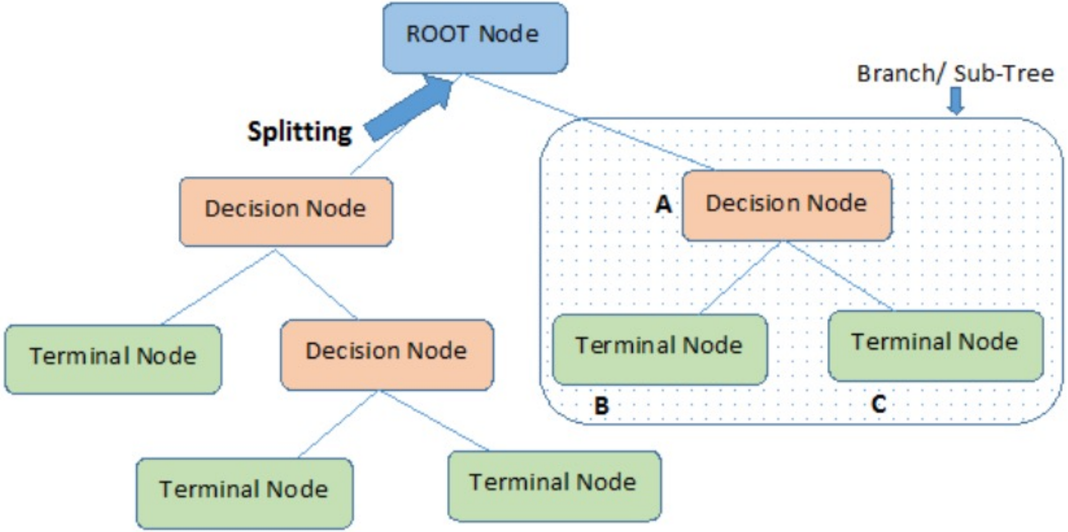
Decision Tree

Decision Tree

- **Decision Tree** is a machine learning algorithm that can be used for classification or regression tasks.
- Decision Tree algorithm uses a tree-like where each internal node represents a test on an attribute, each branch represents the outcome of the test, and each leaf node represents a class label as a decision made.
- There are two main types of decision trees: classification trees and regression trees.
  - Classification trees are used for categorical target variables.
  - Regression trees are used for numerical target variables.

Decision Tree 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.
- ***Leaf Nodes (Terminal Nodes)*** – the nodes where further splitting is not possible.
- ***Sub-tree (Branch)*** – a small portion or sub-section of a decision tree.
- ***Pruning*** – cutting down some nodes to stop overfitting.

Source: https://www.analyticsvidhya.com/blog/2021/08/decision-tree-algorithm/

How Decision Tree Work?

- Decision Trees build a model of decisions and their possible consequences.
- They divide the data into smaller subgroups based on the values of different attributes.
- The tree is built recursively, with each split being based on the attribute that provides the most information gain.

Building a Decision Tree Model

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

- Suppose that we have the data consists of 14 days conditions, based on features **{'Outlook', 'Temperature', 'Humidity', 'Wind'}**
- We need to classify whether person will **play tennis or not.**
- **But how do we select which feature to split on first?** We need that feature which **separates the Target({'Play Tennis'}) best.**
- But how do we know which feature splits the target best, for that we need **impurity**.

Impurity Criterion

Impurity measures the homogeneity of features. The most common way to measure impurity is **“Gini-index” and “Entropy”.**

The **Gini index** is a measure of inequality in samples. It has a value between 0 and 1.
The Gini index of value 0 means samples are perfectly homogeneous and all elements are similar, whereas, the Gini index of value 1 means maximal inequality among elements.

Gini Index

$$I_G = 1 - \sum_{j=1}^{c} p_j^2$$

p_j: proportion of the samples that belongs to class c for a particular node

Entropy

$$I_H = -\sum_{j=1}^{c} p_j log_2(p_j)$$

p_j: proportion of the samples that belongs to class c for a particular node.

*This is the the definition of entropy for all non-empty classes ($p \neq 0$). The entropy is 0 if all samples at a node belong to the same class.

Entropy is the amount of information is needed to accurately describe the samples. So if the sample is homogeneous, which means all the elements are similar then Entropy is 0, else if the sample is equally divided then entropy is a maximum of 1.

Building a Decision Tree Model

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

The data consists of 14 data, 4 attributes/features and 2 classes (YES/NO).

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

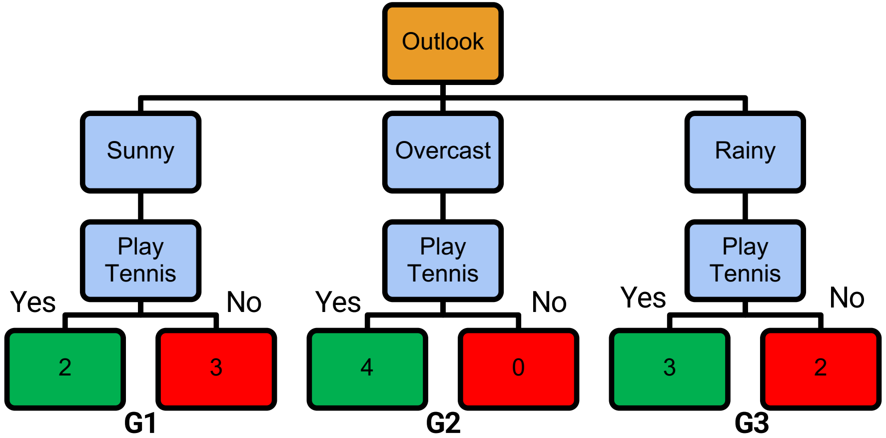
Steps:

1) Calculate the Gini Index (GI) for each attribute.
2) Determine root based on GI value. The root is the attribute with the lowest GI value.
3) Repeat steps 1 and 2 for the next level in the tree until the GI value = 0.

Building a Decision Tree Model

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

Outlook	Play Tennis?
Sunny	No
Sunny	No
Overcast	Yes
Rain	Yes
Rain	Yes
Rain	No
Overcast	Yes
Sunny	No
Sunny	Yes
Rain	Yes
Sunny	Yes
Overcast	Yes
Overcast	Yes
Rain	No

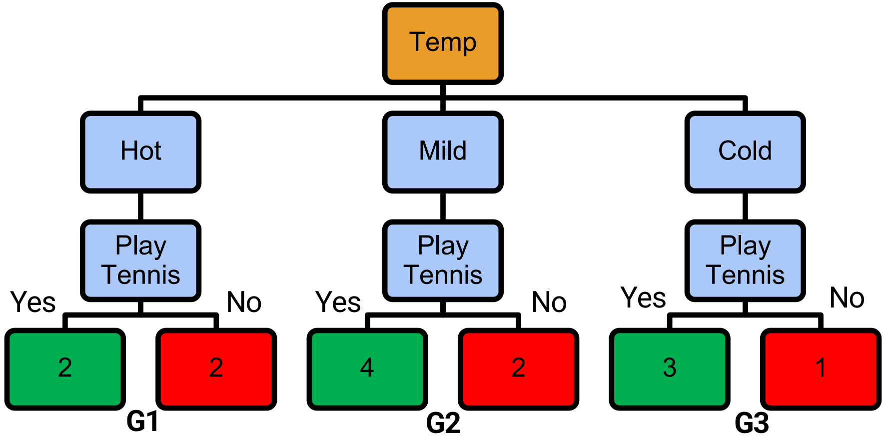


G1 (Sunny) = 1- $(2/5)^2$- $(3/5)^2$ = 12/25 = 0.48 ,
G2 (Overcast) = 1- $(4/4)^2$- $(0/4)^2$ = 0
G3 (Rainy) = 1- $(3/5)^2$- $(2/5)^2$ = 12/25 = 0.48 ,
Now calculating the **weighted average of G1, G2 and G3.**
G.I (Outlook|Play Tennis) = 5/14 * G1 + 4/14 * G2 + 5/14 * G3
= **0.342**

Building a Decision Tree Model

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

Temperature	Play Tennis?
Hot	No
Hot	No
Hot	Yes
Mild	Yes
Cool	Yes
Cool	No
Cool	Yes
Mild	No
Cool	Yes
Mild	Yes
Mild	Yes
Mild	Yes
Hot	Yes
Mild	No

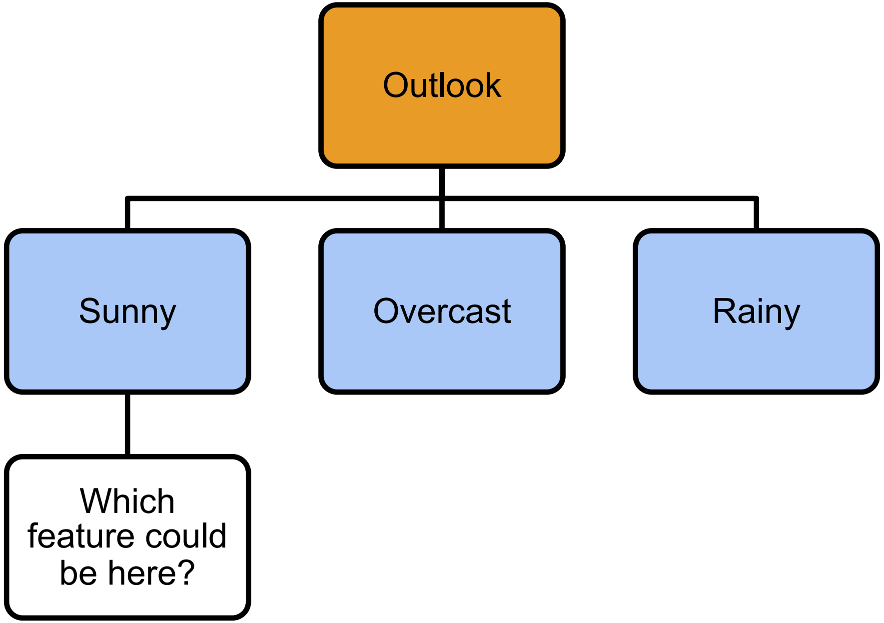


G1 (Hot) = 1- $(2/4)^2$- $(2/4)^2$ = 0.5
G2 (Mild) = 1- $(4/6)^2$- $(2/6)^2$ = 0.444
G3 (Cold) = 1- $(3/4)^2$- $(1/4)^2$ = 0.375
Now calculating the **weighted average of G1, G2 and G3.**
G.I (Temp|Play Tennis) = 4/14*G1 + 6/14*G2 + 4/14*G3
= **0.440**

Determine Root

- **Gini Index Outlook = 0.342**
- Gini index Temperature = 0.440
- Gini index Humidity = 0.367
- Gini index Wind = 0.82

Out of all, **G.I (Outlook|Play Tennis)** is the lowest, so this attribute is chosen as the root node.



Determine Decision Nodes

- Rearranging rows our table.
- **As done previously, we will calculate G.I for {'Temperature', 'Humidity', 'Wind'}.**

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

Determine Decision Nodes

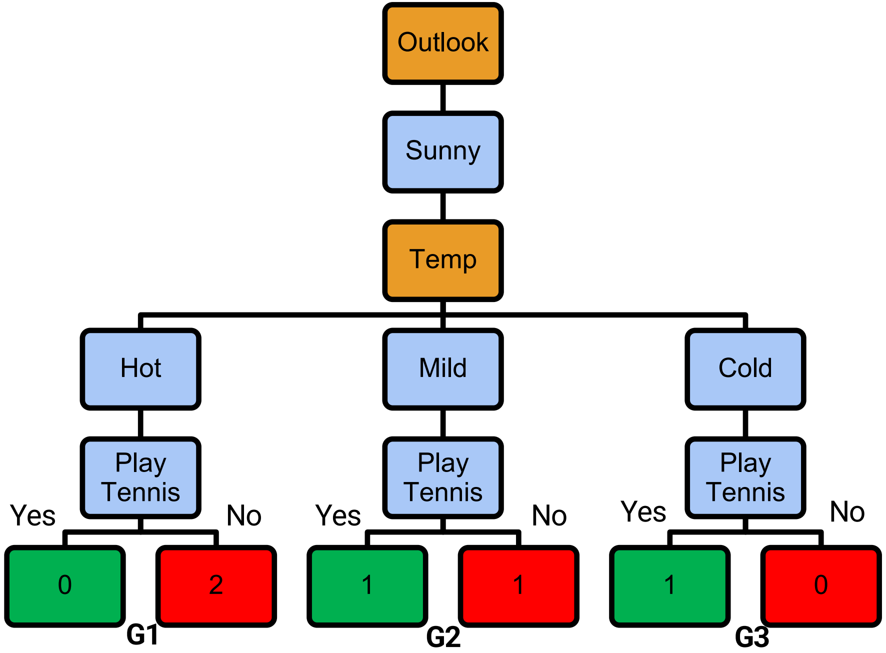
Splitting on ‘**Temperature**’

- $G1 = 1 - (0/2)^2 - (2/2)^2 = 0$
- $G2 = 1 - (1/2)^2 - (1/2)^2 = 0.5$
- $G3 = 1 - (1/1)^2 - (0/1)^2 = 0$

G.I(Temperature|Sunny)

= 2/5*G1 + 2/5*G2 + 1/5*G3

= **0.2**



Determine Decision Nodes

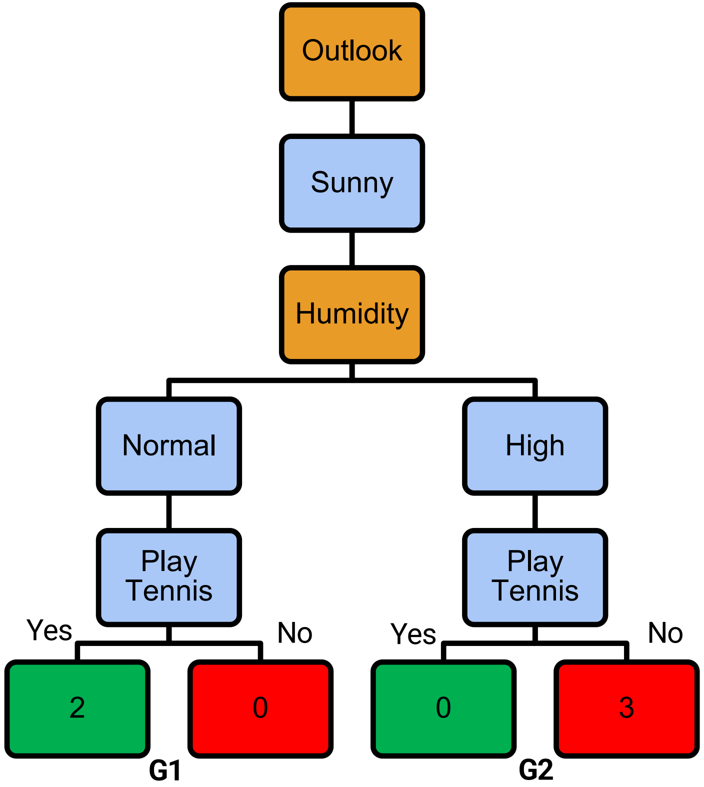
Splitting on ‘**Humidity**’

- $G1 = 1 - (2/2)^2 - (0/2)^2 = 0$
- $G2 = 1 - (3/3)^2 - (0/3)^2 = 0$

G.I (Humidity|Sunny)

$= 2/5*G1 + 3/5*G2$

$= 0$

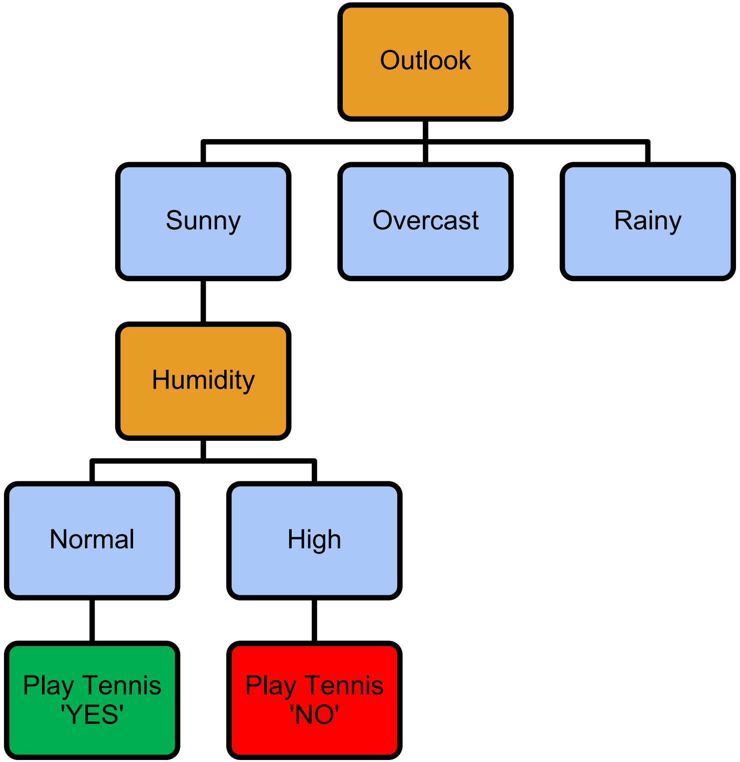


Determine Decision Nodes

- G.I (Temperature|Sunny) = 0.2
- **G.I (Humidity|Sunny) = 0**
- G.I (Wind|Sunny) = 0.466

Out of all, **G.I (Humidity|Sunny)** is the lowest, so Humidity is selected.

Now, the decision tree looks like..



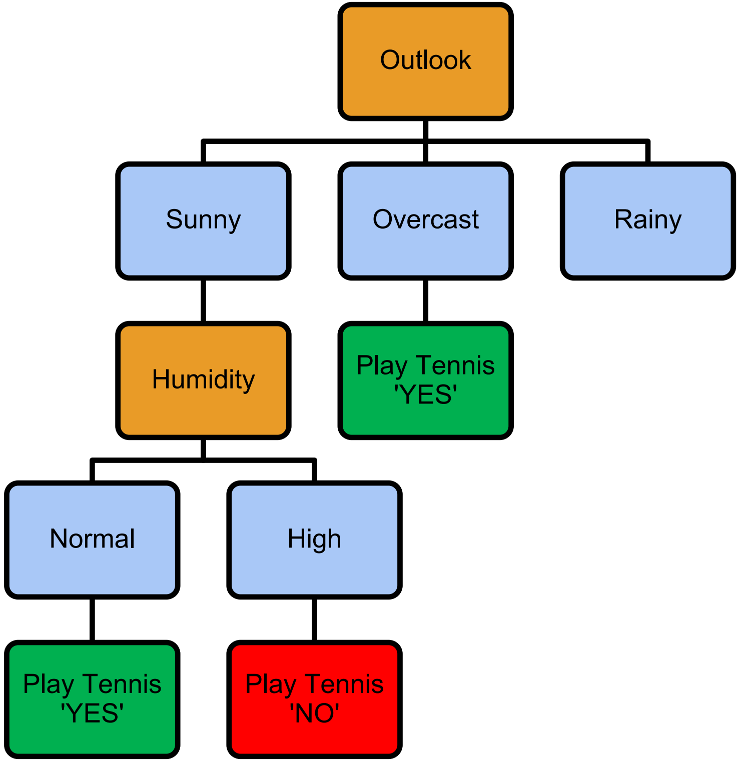
Determine Decision Nodes

- **Now we will try to split 'Overcast' to check if we can reduce G.I.**
- But we observe that whenever **'Outlook'** is **'Overcast' Play Tennis?** Is always **'YES'**.
- This means we do not need to split this node further.

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

Determine Decision Nodes

Now, the decision tree looks like..



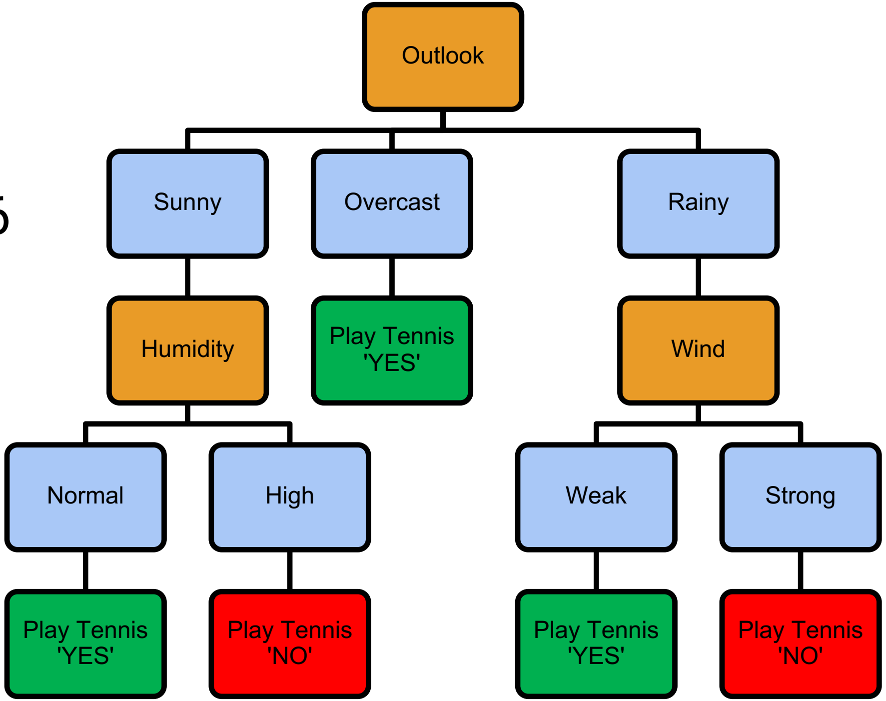
Determine Decision Nodes

Similarly, we calculate:

- G.I (Temperature|Rainy) = 0.466
- G.I (Humidity|Rainy) = 0.466
- **G.I (Wind|Rainy) = 0**

Out of all, **G.I (Wind|Rainy)** is the lowest, so Wind is selected.

Now, our final decision tree is..



What if your attribute is a numeric value?

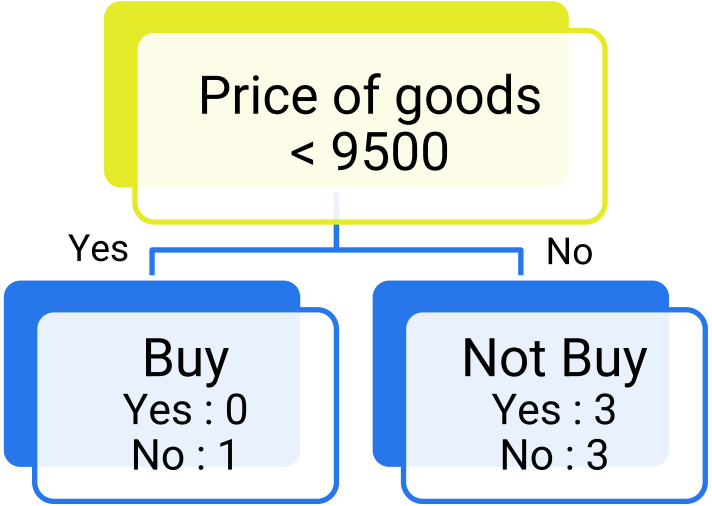
Primary needs?	Have you got paid?	Price of goods	Buy?
Yes	Yes	7000	No
Yes	No	12000	No
No	Yes	18000	Yes
No	Yes	35000	Yes
Yes	Yes	38000	Yes
Yes	No	50000	No
No	No	83000	No

Gini Index for “Price of goods” attribute

The average of two prices

9500
15000
26500
36500
44000
66500

Price of goods	Buy?
7000	No
12000	No
18000	Yes
35000	Yes
38000	Yes
50000	No
83000	No



Gini Impurity
$= 1 - (0/1)^2 - (1/1)^2$
$= 0$

Gini Impurity
$= 1 - (3/6)^2 - (3/6)^2$
$= 0.5$

Total of Gini Impurity for Price of Goods < 9500
$= (1/7)*0 + (6/7)*0.5$
$= 0.43$

Gini Index for “Price of goods” attribute

Gini Index	The average of two prices
0.43	9500
0.34	15000
0.48	26500
0.48	36500
0.34	44000
0.43	66500

Price of goods	Buy?
7000	No
12000	No
18000	Yes
35000	Yes
38000	Yes
50000	No
83000	No

Choose the lowest Gini index: **Price of Goods < 15000** or **Price of Goods < 44000**

Hyperparameters

- Usually, real-world datasets have a large number of features, which will result in a large number of splits, which in turn gives a huge tree. Such trees take time to build and can lead to **overfitting**.
- There are many ways to tackle this problem through **hyperparameter tuning**.
- We can set the maximum depth of our decision tree using the ***max_depth*** parameter.
  - The more the value of ***max_depth***, the more complex your tree will be.
  - The training error will decrease if we increase the ***max_depth*** value but when our test data comes into the picture, we will get a very bad accuracy.
  - Therefore we need a value that will not overfit and underfit our data; for this, we can use GridSearchCV.
- Another way is to specify the minimum number of samples for each spilt using ***min_samples_split***.
  - For example, we can use a minimum of 10 samples to reach a decision. That means if a node has less than 10 samples then using this parameter, we can stop the further splitting of this node and make it a leaf node.
- There are more hyperparameters such as :
  - ***min_samples_leaf*** – represents the minimum number of samples required to be in the leaf node. The more you increase the number, the more is the possibility of overfitting.
  - ***max_features*** – it helps us decide what number of features to consider when looking for the best split.

Pruning

- It is another method that can help us avoid overfitting. It helps in improving the performance of the tree by cutting the nodes or sub-nodes which are not significant. It removes the branches which have very low importance.
- There are mainly 2 ways for pruning:
  - **Pre-pruning** – we can stop growing the tree earlier, which means we can prune/remove/cut a node if it has low importance **while growing** the tree.
  - **Post-pruning** – once our **tree is built to its depth**, we can start pruning the nodes based on their significance.

Decision Tree using Scikit-Learn

```
# Import Decision Tree Classifier dari sklearn
from sklearn.tree import DecisionTreeClassifier

# Import metric untuk memeriksa akurasi
from sklearn import metrics

dt = DecisionTreeClassifier(
      max_depth=None,
      min_samples_split = 2
    )
dt.fit(X_train, y_train)
y_pred = dt.predict(X_test)
score = metrics.accuracy_score(y_test, y_pred)
print(score)
```

Hyperparameters of Decision Tree

```
DecisionTreeClassifier(class_weight=None, criterion='entropy', max_depth=None,
                       max_features=None, max_leaf_nodes=None,
                       min_impurity_decrease=0.0, min_impurity_split=None,
                       min_samples_leaf=1, min_samples_split=2,
                       min_weight_fraction_leaf=0.0, presort=False,
                       random_state=None, splitter='best')
```

- **criterion** : *string, optional (default=”gini”)*
  The function to measure the quality of a split. Supported criteria are **“gini”** for the **Gini impurity** and **“entropy”** for the **information gain**.
- **max_depth** : *int or None, optional (default=None)*
  The maximum depth of the tree. If None, then nodes are expanded until all leaves are pure or until all leaves contain less than min_samples_split samples. **Higher the max_depth, more chances of overfitting.**
- **min_samples_split** : *int, float, optional (default=2)*
  The minimum number of samples required to split an internal node:
  - If **int**, then consider min_samples_split as the minimum number.
  - If **float**, then min_samples_split is a fraction and ceil(min_samples_split*n_samples) are the minimum number of samples for each split.

Hyperparameters of Decision Tree

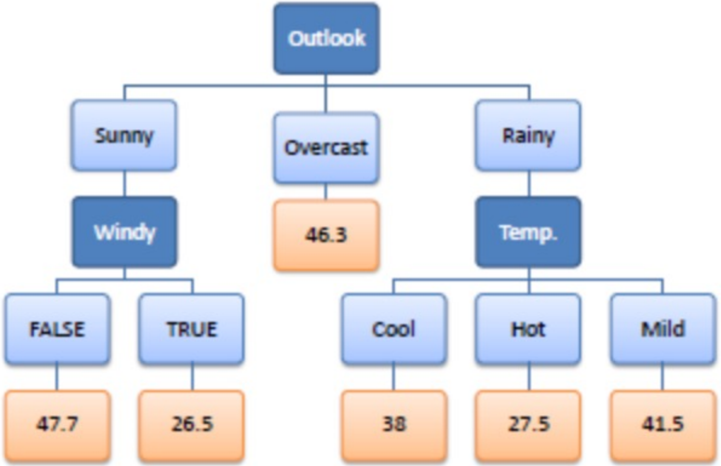
```
DecisionTreeClassifier(class_weight=None, criterion='entropy', max_depth=None,
                       max_features=None, max_leaf_nodes=None,
                       min_impurity_decrease=0.0, min_impurity_split=None,
                       min_samples_leaf=1, min_samples_split=2,
                       min_weight_fraction_leaf=0.0, presort=False,
                       random_state=None, splitter='best')
```

- **min_weight_fraction_leaf** : *float, optional (default=0.)*
  The minimum weighted fraction of the sum total of weights (of all the input samples) required to be at a leaf node. Samples have equal weight when sample_weight is not provided.
- **max_features** : *int, float, string or None, optional (default=None)*
  The number of features to consider when looking for the best split, If “auto”,
  then **max_features=sqrt(n_features)**

Decision Tree - Regression

- Decision tree builds regression model 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), each representing values for the attribute tested.
  - Leaf node (e.g., Hours Played) represents a decision on the numerical target.

Predictors				Target
Outlook	Temp.	Humidity	Windy	Hours Played
Rainy	Hot	High	False	25
Rainy	Hot	High	True	30
Overcast	Hot	High	False	46
Sunny	Mild	High	False	45
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	46
Rainy	Mild	Normal	True	48
Overcast	Mild	High	True	52
Overcast	Hot	Normal	False	44
Sunny	Mild	High	True	30



Source: https://www.saedsayad.com/decision_tree_reg.htm

Advantages of Decision Tree

- Decision Tree is easy to understand and interpret.
- It provide an explanation of the process carried out in the form of generated rules
- It can handle both categorical and numerical data.
- It can handle missing values and outliers.
- It can handle nonlinear relationships between the features and the target variable.

Limitations of Decision Tree

- Decision Tree can easily overfit the training data if not pruned properly.
- The process of building decision trees on numerical data is more complicated and may contain missing information.
- Decision trees can grow to be very complex on complex data.
- It may not perform well if the data is imbalanced.
- It can be sensitive to small changes in the data, leading to different tree structures.

Conclusion

- Decision Trees are a powerful and versatile algorithm for machine learning classification tasks.
- They have their limitations, but when used appropriately, they can provide valuable insights and accurate predictions.

THANK YOU