

MACHINE LEARNING

SUPERVISED

CHAP 4

CLASSIFICATION/ DECISION TREE

Jean Tabet

j.tabet@ul.edu.lb

➔ DECISION TREE

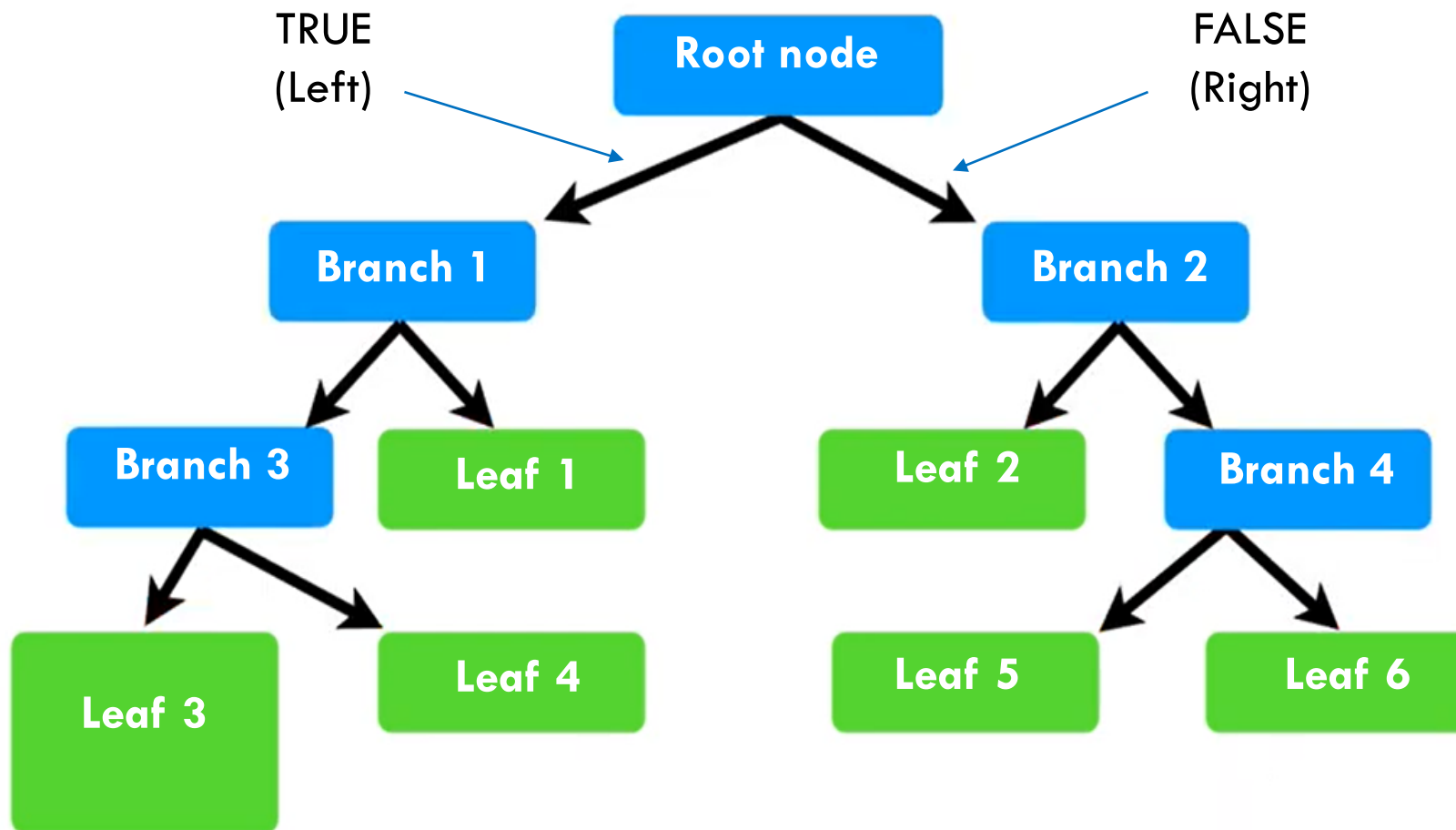
- Simple and Intuitive: Decision trees mimic human decision-making, making them easy to understand and interpret.
- Handles Different Data Types: both numerical and categorical data effectively.
- No Need for Feature Scaling: Unlike many algorithms, decision trees don't require normalization or standardization.
- Captures Non-linear Relationships: Able to model complex, non-linear patterns.
- Useful for Classification & Regression.
- Wide Real-world Applications: From medical diagnosis and fraud detection to customer segmentation and recommendation systems

➔ DECISION TREE

Applications

- Credit scoring (approve/reject loan)
- Medical diagnosis (predict disease based on symptoms)
- Spam detection (classify emails)
- Customer segmentation (marketing)

→ DECISION TREE



→ DECISION TREE

Example

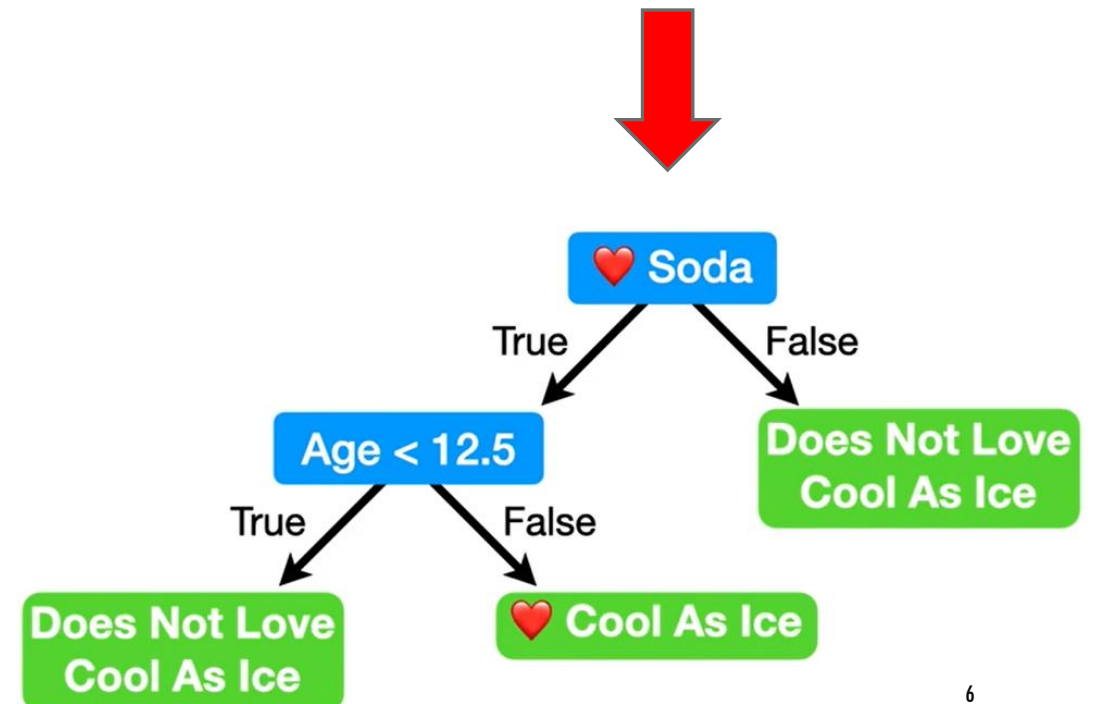
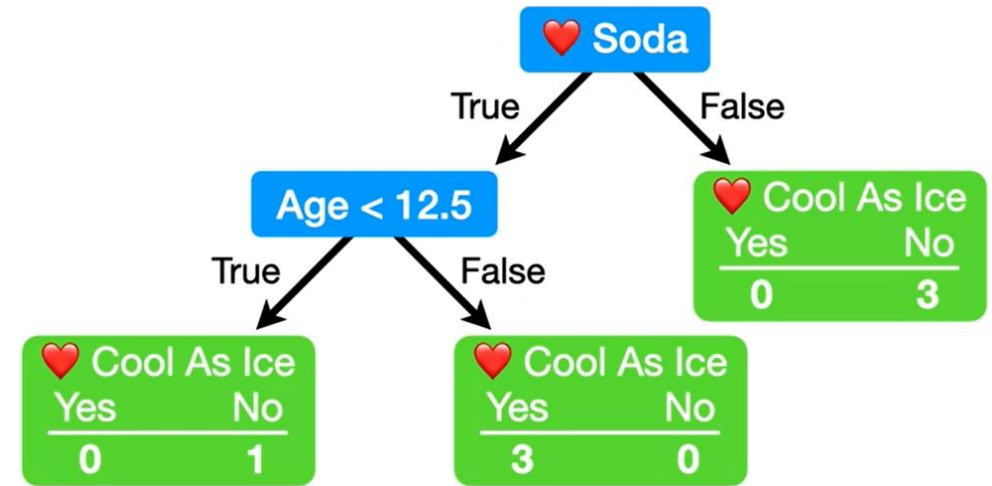
Classification of people if they like the movie Cool As Ice depending on the variables:

- Loving Popcorn
- Loving Soda
- Age

Loves Popcorn	Loves Soda	Age	Loves Cool As Ice
Yes	Yes	7	No
Yes	No	12	No
No	Yes	18	Yes
No	Yes	35	Yes
Yes	Yes	38	Yes
Yes	No	50	No
No	No	83	No

→ DECISION TREE

Loves Popcorn	Loves Soda	Age	Loves Cool As Ice
Yes	Yes	7	No
Yes	No	12	No
No	Yes	18	Yes
No	Yes	35	Yes
Yes	Yes	38	Yes
Yes	No	50	No
No	No	83	No

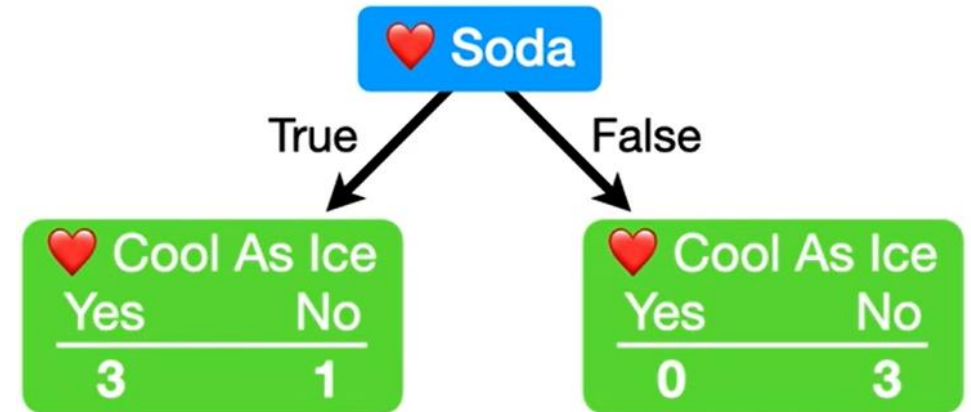
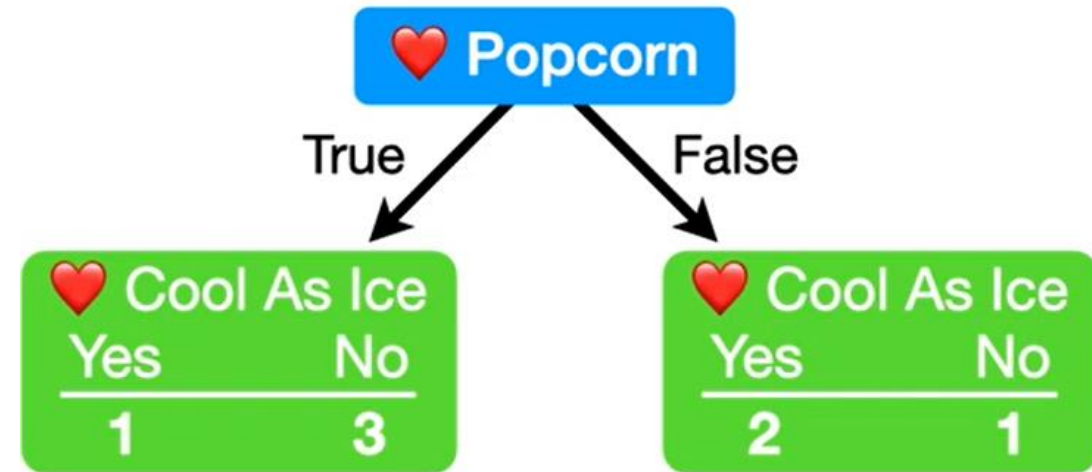


→ IMPURITY

The selection of the starting root node is based on the impurity of each variable in classifying the outcome. Impurity can be calculated using:

- Gini index
- Entropy
- Information gain

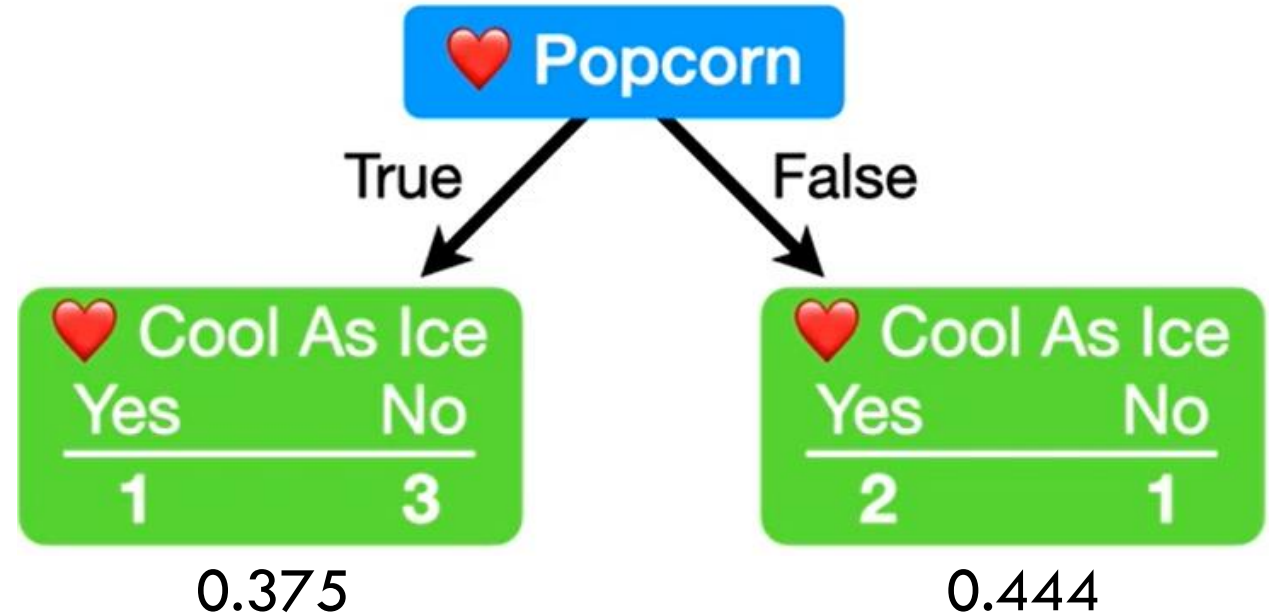
We focus here on **Gini impurity**



→ GINI IMPURITY

$$\text{Gini impurity} = 1 - \sum (p_i^2)$$

where p_i is the probability of class i .



$$\begin{aligned} \text{Gini Impurity for Leaf 1} &= 1 - (\text{the probability of "Yes"})^2 - (\text{the probability of "No"})^2 \\ &= 1 - \left(\frac{1}{1+3}\right)^2 - \left(\frac{3}{1+3}\right)^2 = 0.375 \end{aligned}$$

$$\begin{aligned} \text{Gini Impurity for Leaf 2} &= 1 - (\text{the probability of "Yes"})^2 - (\text{the probability of "No"})^2 \\ &= 1 - \left(\frac{2}{2+1}\right)^2 - \left(\frac{1}{2+1}\right)^2 = 0.444 \end{aligned}$$

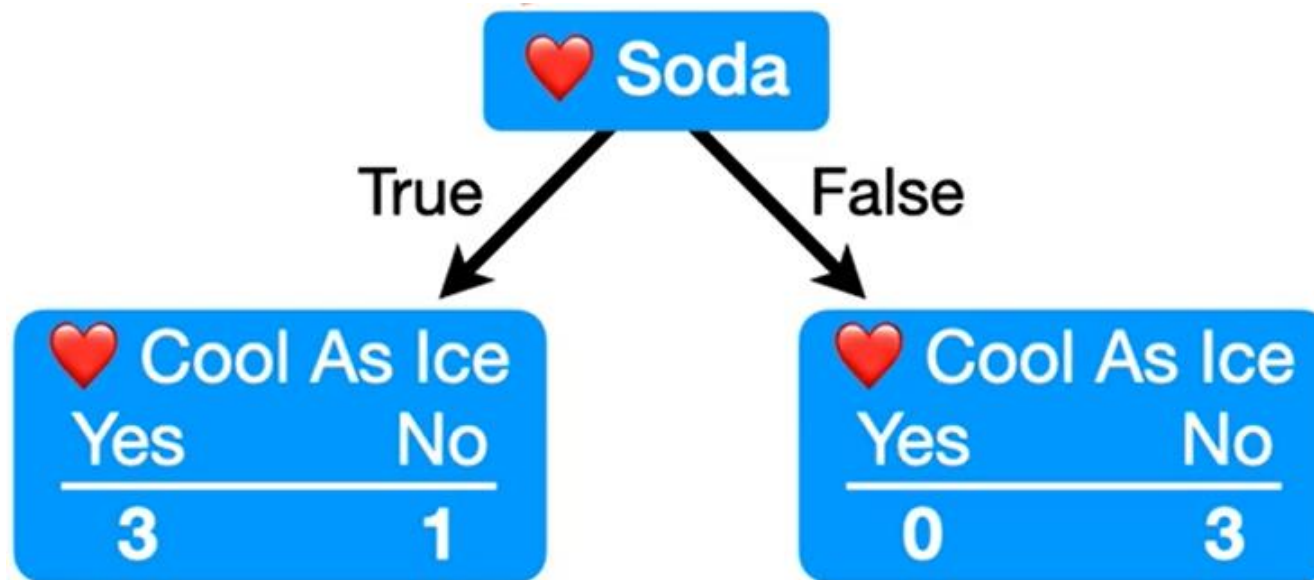
Gini Impurity = weighted average of **Gini Impurities** for the **Leaves**

$$\begin{aligned} \text{Total} &= \left(\frac{4}{4+3}\right) 0.375 + \left(\frac{3}{4+3}\right) 0.444 = 0.405 \end{aligned}$$

➔ GINI IMPURITY

$$\text{Gini impurity} = 1 - \sum (p_i^2)$$

where p_i is the probability of class i .



Gini index = 0.214

GINI IMPURITY FOR NUMERICAL DATA

Average
of each 2
consecutive
scores

	Age	Loves Cool As Ice	
	7	No	
9.5	12	No	Gini Impurity = 0.429
15	18	Yes	Gini Impurity = 0.343
26.5	35	Yes	Gini Impurity = 0.476
36.5	38	Yes	Gini Impurity = 0.476
44	50	No	Gini Impurity = 0.343
66.5	83	No	Gini Impurity = 0.429

$$\text{Gini impurity} = 1 - \sum (p_i^2)$$

These two candidate thresholds, **15** and **44**, are tied for the lowest **Impurity**, **0.343...**

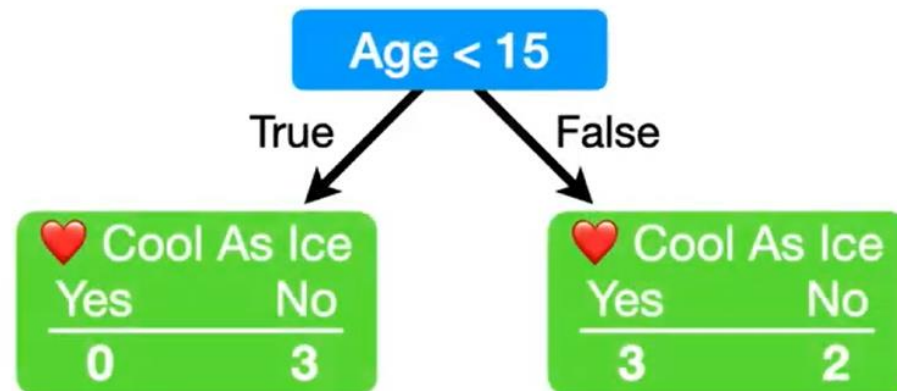
GINI IMPURITY

	Age	Loves Cool As Ice
	7	No
9.5	12	No
15	18	Yes
26.5	35	Yes
36.5	38	Yes
44	50	No
66.5	83	No

Gini Impurity = 0.343

...so we can pick either one. In this case, we'll pick **15**.

Gini Impurity for Age < 15 = 0.343



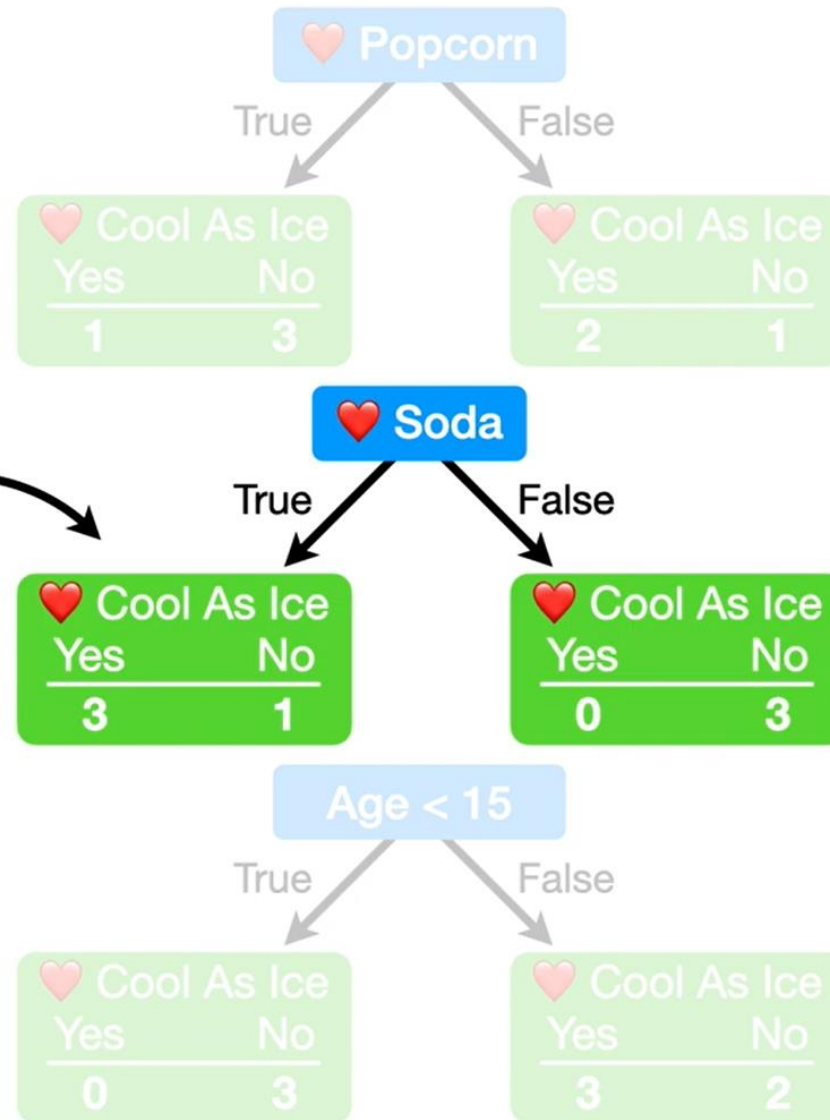
GINI IMPURITY

Gini Impurity for Loves Popcorn = 0.405

...we know that its
Leaves have the lowest
Impurity...

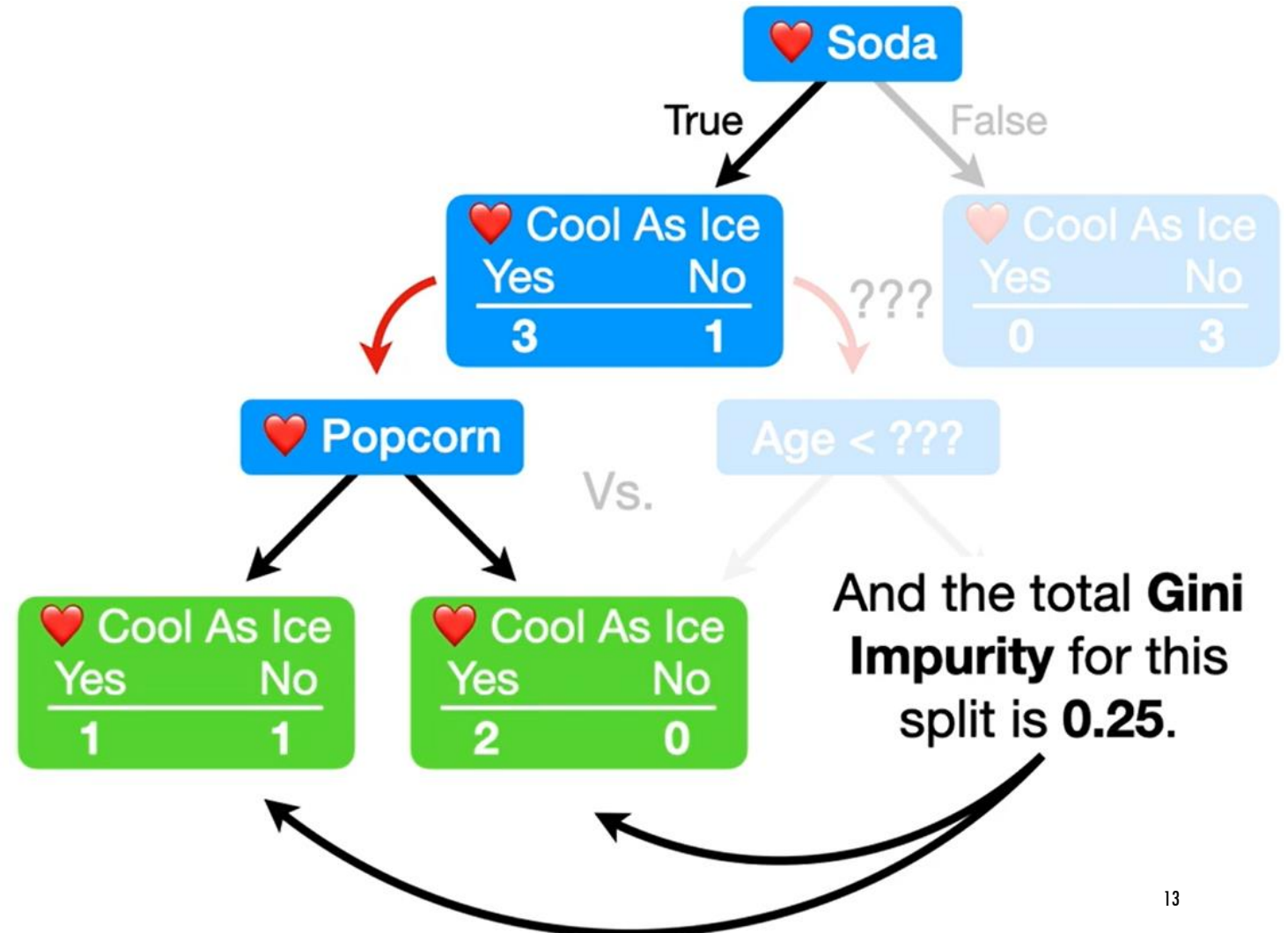
Gini Impurity for Loves Soda = 0.214

Gini Impurity for Age < 15 = 0.343



GINI IMPURITY

Loves Popcorn	Loves Soda	Age	Loves Cool As Ice
Yes	Yes	7	No
Yes	No	12	No
No	Yes	18	Yes
No	Yes	35	Yes
Yes	Yes	38	Yes
Yes	No	50	No
No	No	83	No



GINI IMPURITY

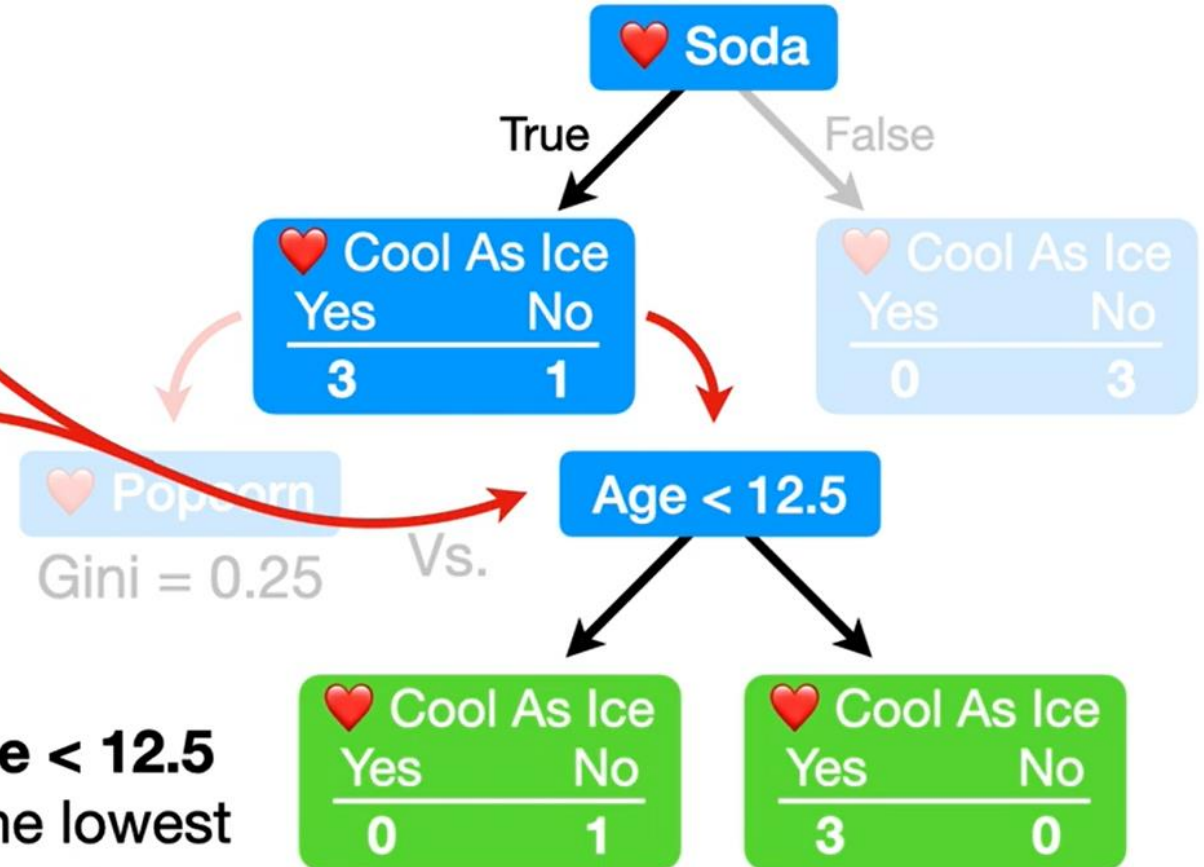
Loves Popcorn	Loves Soda	Age	Loves Cool As Ice
Yes	Yes	7	No
Yes	No	12	No
No	Yes	18	Yes
No	Yes	35	Yes
Yes	Yes	38	Yes
Yes	No	50	No
No	No	83	No

12.5

26.5

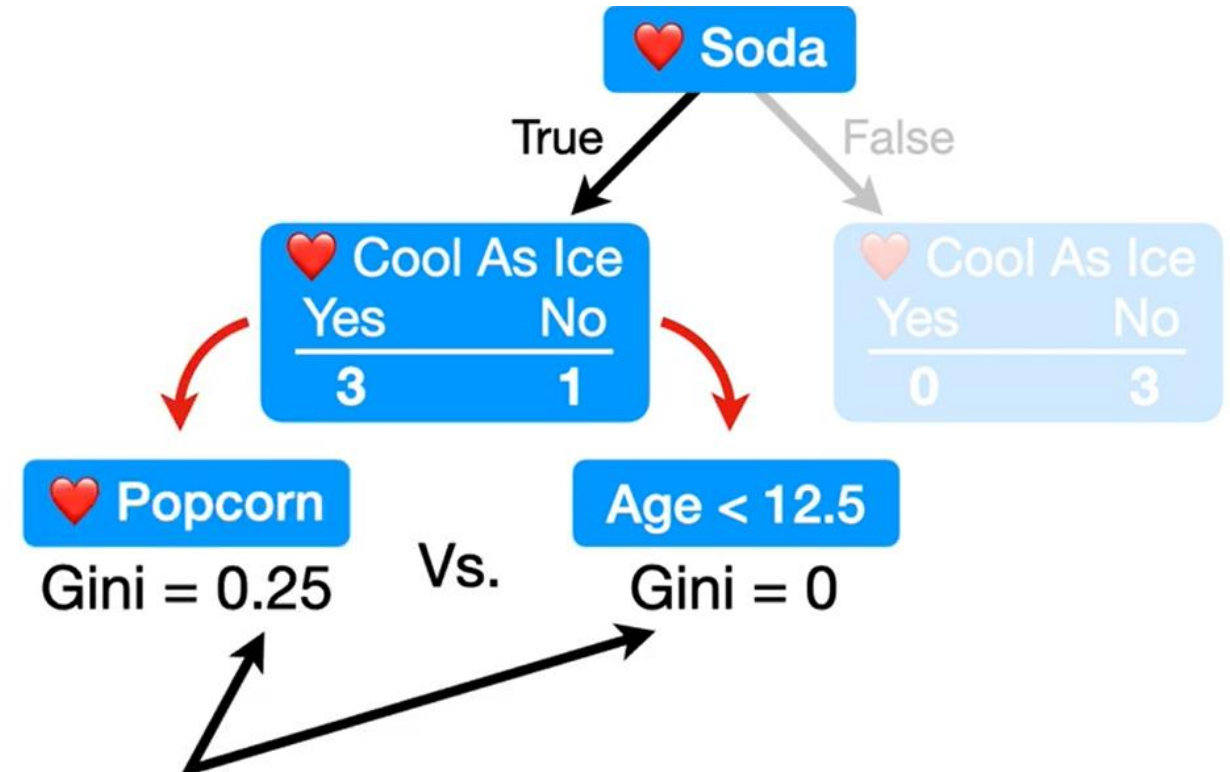
36.5

...and **Age < 12.5**
gives us the lowest
Impurity, 0...



GINI IMPURITY

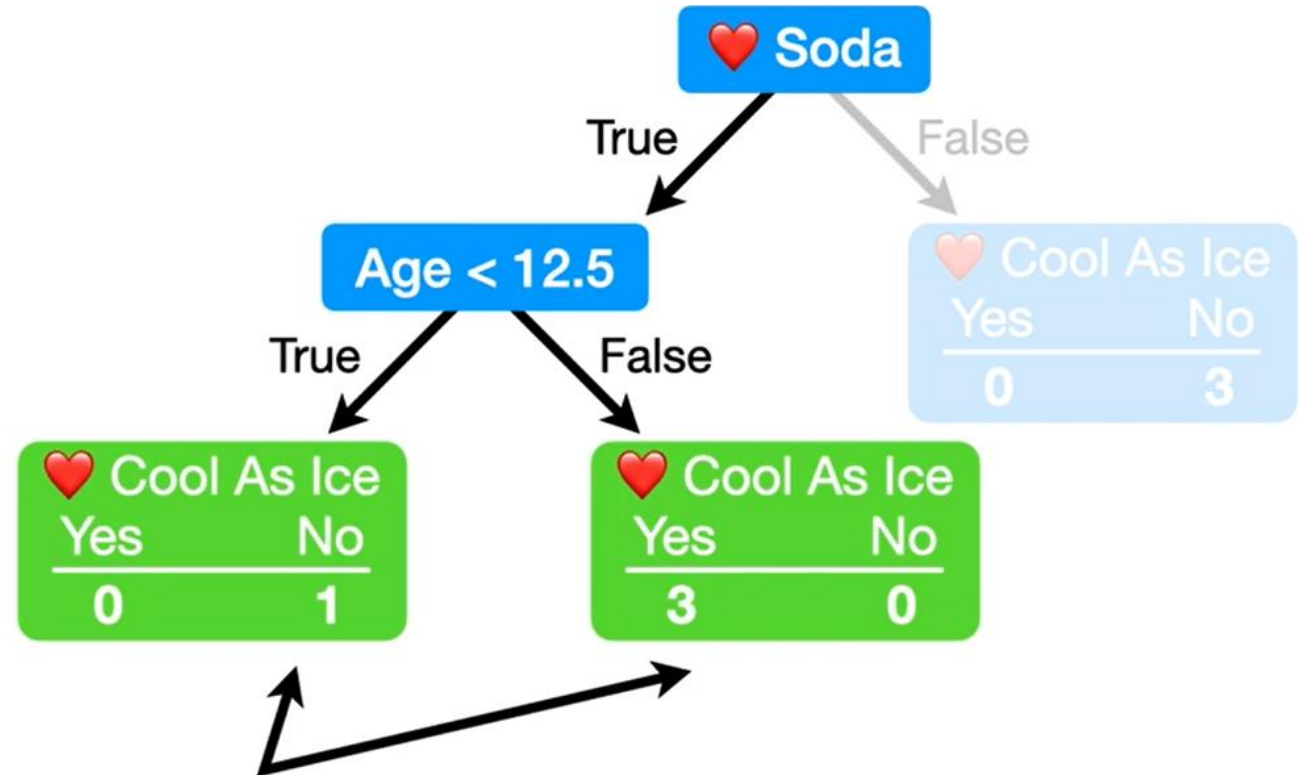
Loves Popcorn	Loves Soda	Age	Loves Cool As Ice
Yes	Yes	7	No
Yes	No	12	No
No	Yes	18	Yes
No	Yes	35	Yes
Yes	Yes	38	Yes
Yes	No	50	No
No	No	83	No



Now, because **0** is less than **0.25**, we will use **Age < 12.5** to split this **Node** into **Leaves**.

→ GINI IMPURITY

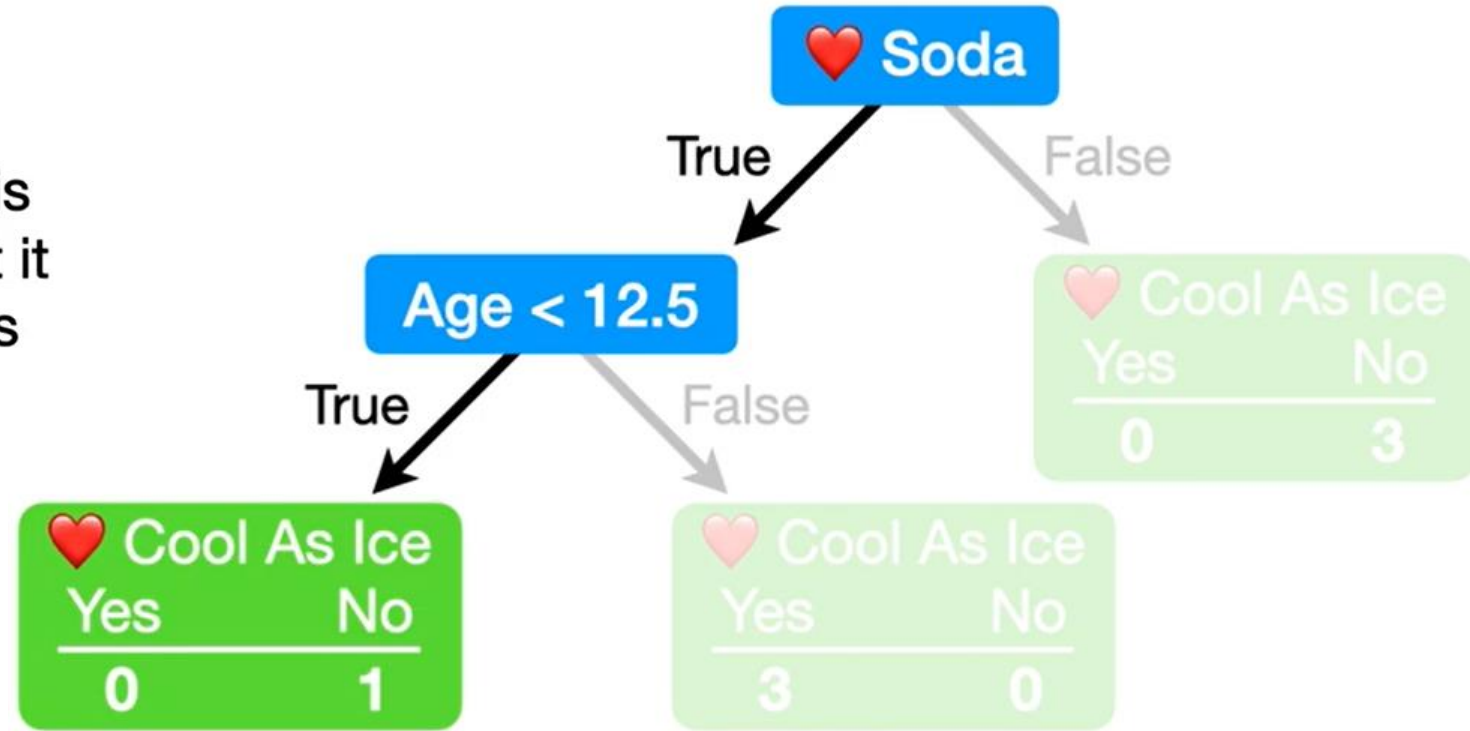
Loves Popcorn	Loves Soda	Age	Loves Cool As Ice
Yes	Yes	7	No
Yes	No	12	No
No	Yes	18	Yes
No	Yes	35	Yes
Yes	Yes	38	Yes
Yes	No	50	No
No	No	83	No



NOTE: These are **Leaves** because there is no reason to continue splitting these people into smaller groups.

➔ OVERFITTING RISK

Because so few people made it to this **Leaf**, it's hard to have confidence that it will do a great job making predictions with future data.



And it's possible that we have **Overfit** the data.

Solution: - Pruning
- Limits on the number of people in each leaf

➔ PRUNING

In decision trees, pruning is a critical step to reduce overfitting and improve generalization.

There are two main categories of pruning methods in machine learning:

- Pre-Pruning (Early Stopping)
- Post-Pruning (Simplifying After Full Growth)

➔ PRE-PRUNING (EARLY STOPPING)

This method stops the tree from growing too deep in the first place. Typical strategies:

- Maximum Depth (max_depth) → Limit how many levels the tree can grow.
 - Minimum Samples per Split (min_samples_split) → Minimum splitting samples
 - Minimum Samples per Leaf (min_samples_leaf) → Ensure each leaf has enough samples to avoid being too specific.
 - Maximum Features (max_features) → Number of features considered at each split.
- ✓ Advantage: Fast and prevents large trees.
- ✗ Disadvantage: Risk of underfitting if the tree is stopped too early.

➔ POST-PRUNING (AFTER FULL GROWTH)

The tree is grown fully, then unnecessary branches are removed. Main techniques:

- **Cost-Complexity Pruning (CCP)** → Used in CART (Classification and Regression Trees).
 - Adds a penalty term $\alpha \times \text{Tree Size (Nb of leaves)}$ to balance accuracy vs complexity.
 - Uses cross-validation to find the best α .
- **Reduced Error Pruning** → Iteratively remove nodes if doing so does not decrease validation accuracy.
- **Minimum Error Pruning** → Remove subtrees if they don't reduce prediction error significantly.



Advantage: Produces more optimal and balanced trees.



Disadvantage: Requires extra computation (grow full tree first + pruning phase). ²⁰

➔ DECISION TREE CLASSIFIER (PYTHON)

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

# Train/test split & evaluation
from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.metrics import classification_report, accuracy_score

# Example: Split your dataset (X = features, y = labels)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Define the base model
dt = DecisionTreeClassifier(random_state=42)
```

➔ TUNING HYPERPARAMETERS

```
# Define hyperparameter grid
param_grid = {
    'criterion': ['gini', 'entropy', 'log_loss'],
    'max_depth': [None, 3, 5, 10, 20],
    'min_samples_split': [2, 5, 10],
    'min_samples_leaf': [1, 2, 5],
    'max_features': [None, 'sqrt', 'log2'],
    'ccp_alpha': [0.0, 0.001, 0.01]
}

# Set up GridSearchCV
grid = GridSearchCV(estimator=dt, param_grid=param_grid,
                    cv=5, scoring='accuracy', n_jobs=-1)

# Fit
grid.fit(X_train, y_train)

# Evaluate
print("Best Parameters:", grid.best_params_)
y_pred = grid.predict(X_test)
print("Accuracy:", accuracy_score(y_test, y_pred))
print(classification_report(y_test, y_pred))
```