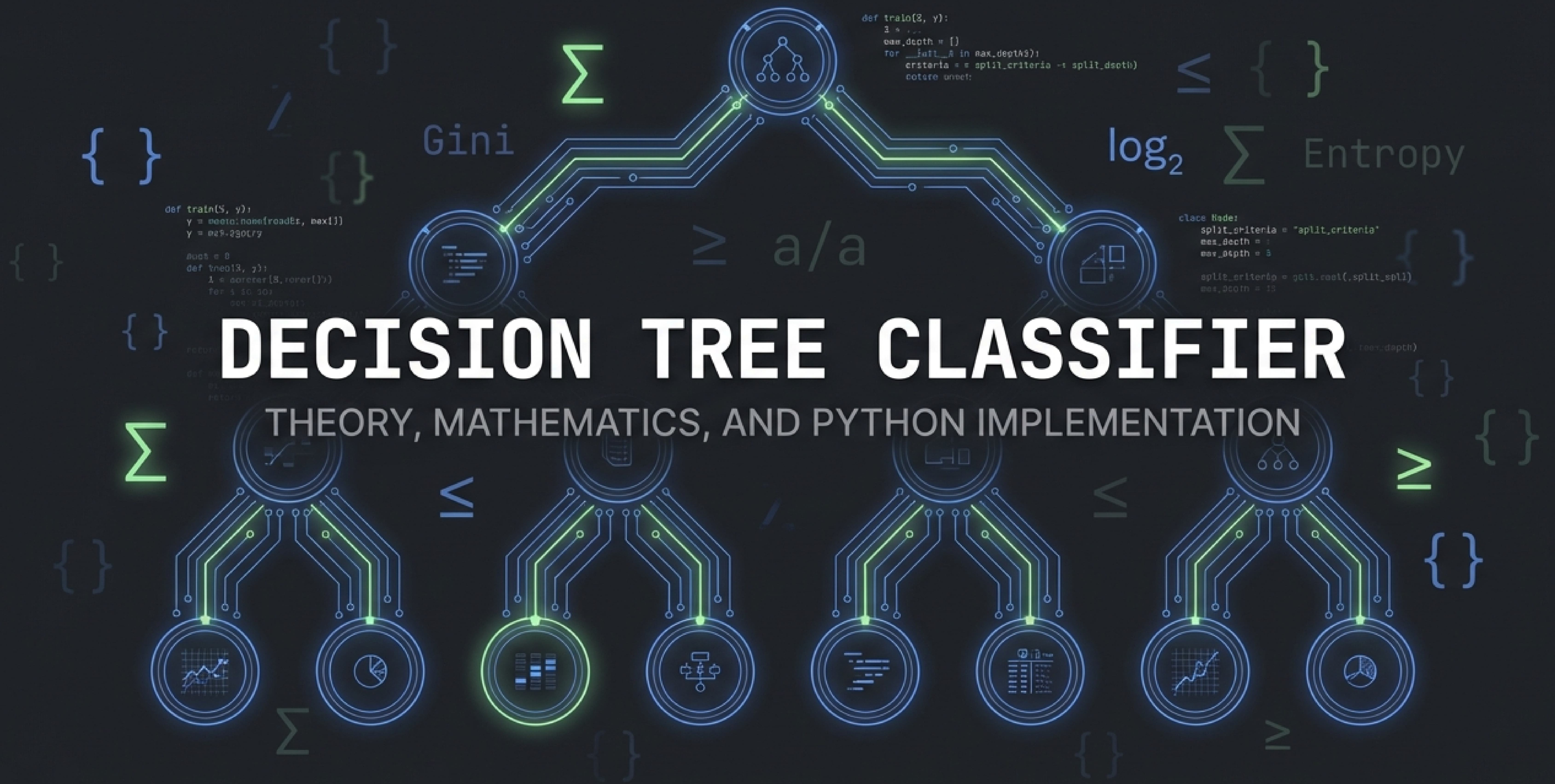
DECISION TREE CLASSIFIER

THEORY, MATHEMATICS, AND PYTHON IMPLEMENTATION



NotebookLM

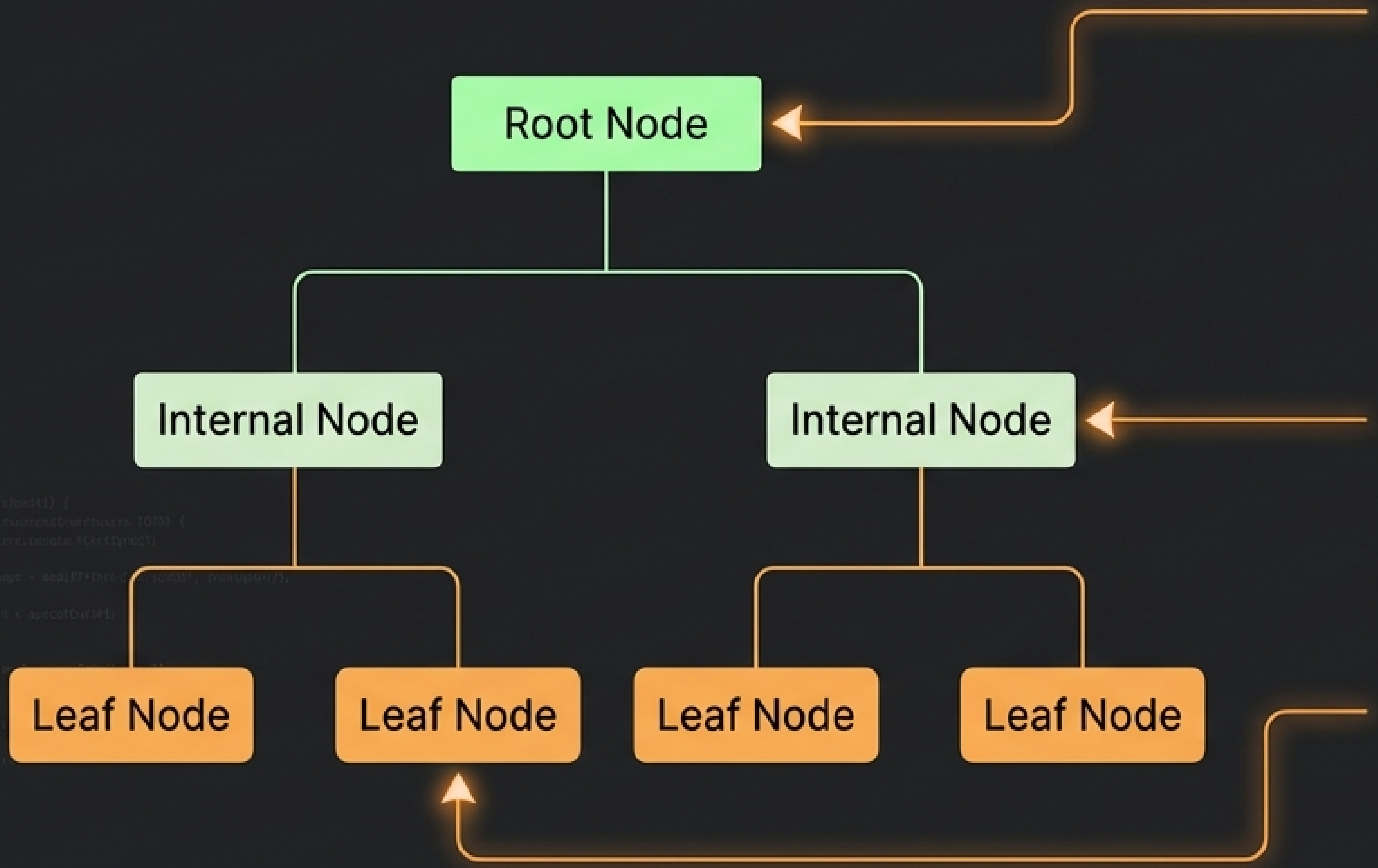
MIMICKING HUMAN LOGIC

A Decision Tree is a supervised machine learning algorithm that uses a hierarchical, flowchart-like structure to make predictions. Unlike “black box” models, it explicitly maps out decision paths.

>>> ML.explain(model)

NotebookLM

ANATOMY OF THE STRUCTURE



Root Node: The starting point containing the entire dataset (N).

Internal Node: A decision point where data splits based on a specific feature.

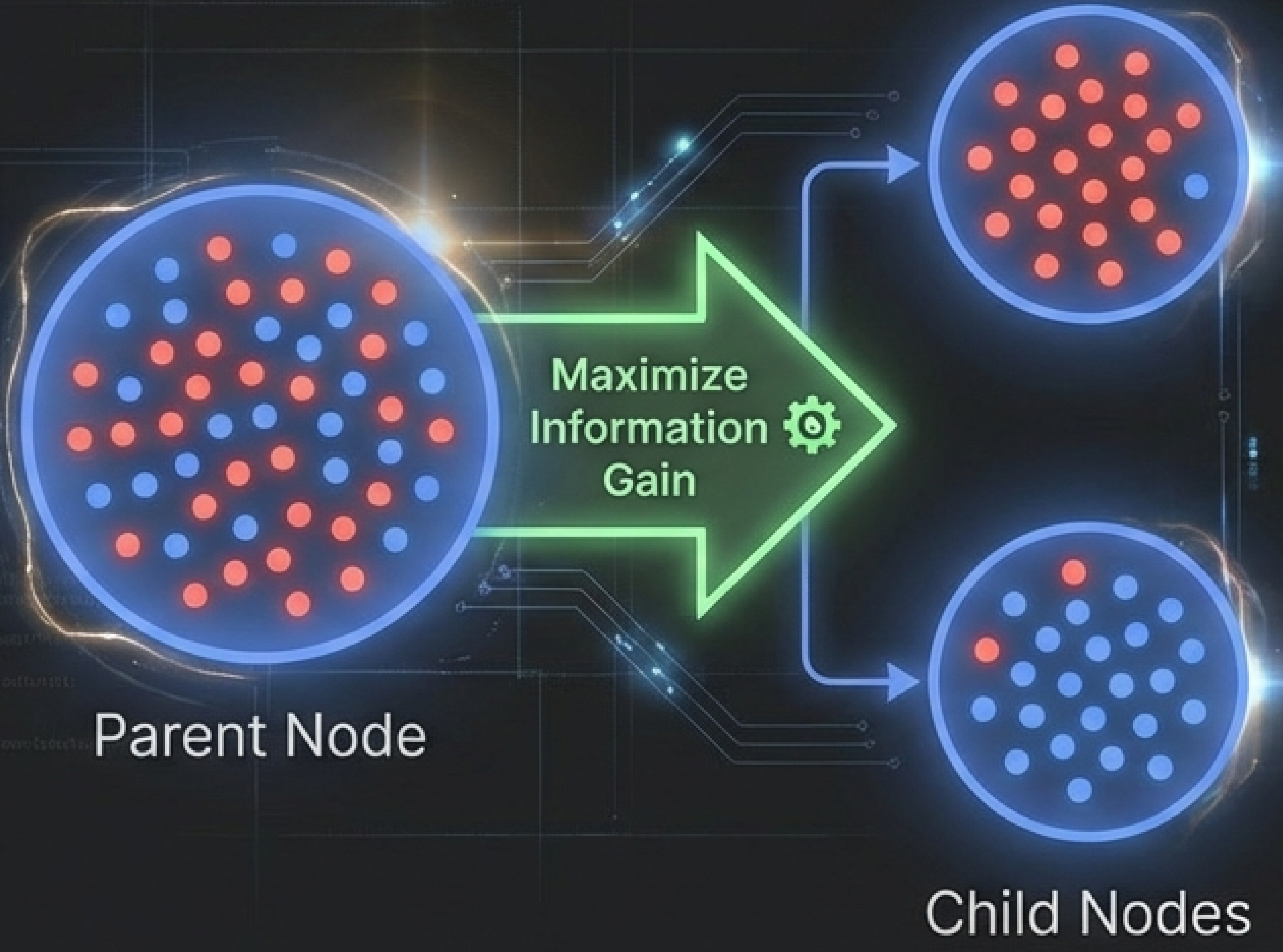
JetBrains Mono

Leaf Node: The final output or class prediction. No further splits occur here.

JetBrains Mono

`>>> ML.explain(model)`

THE SPLITTING LOGIC: INFORMATION GAIN



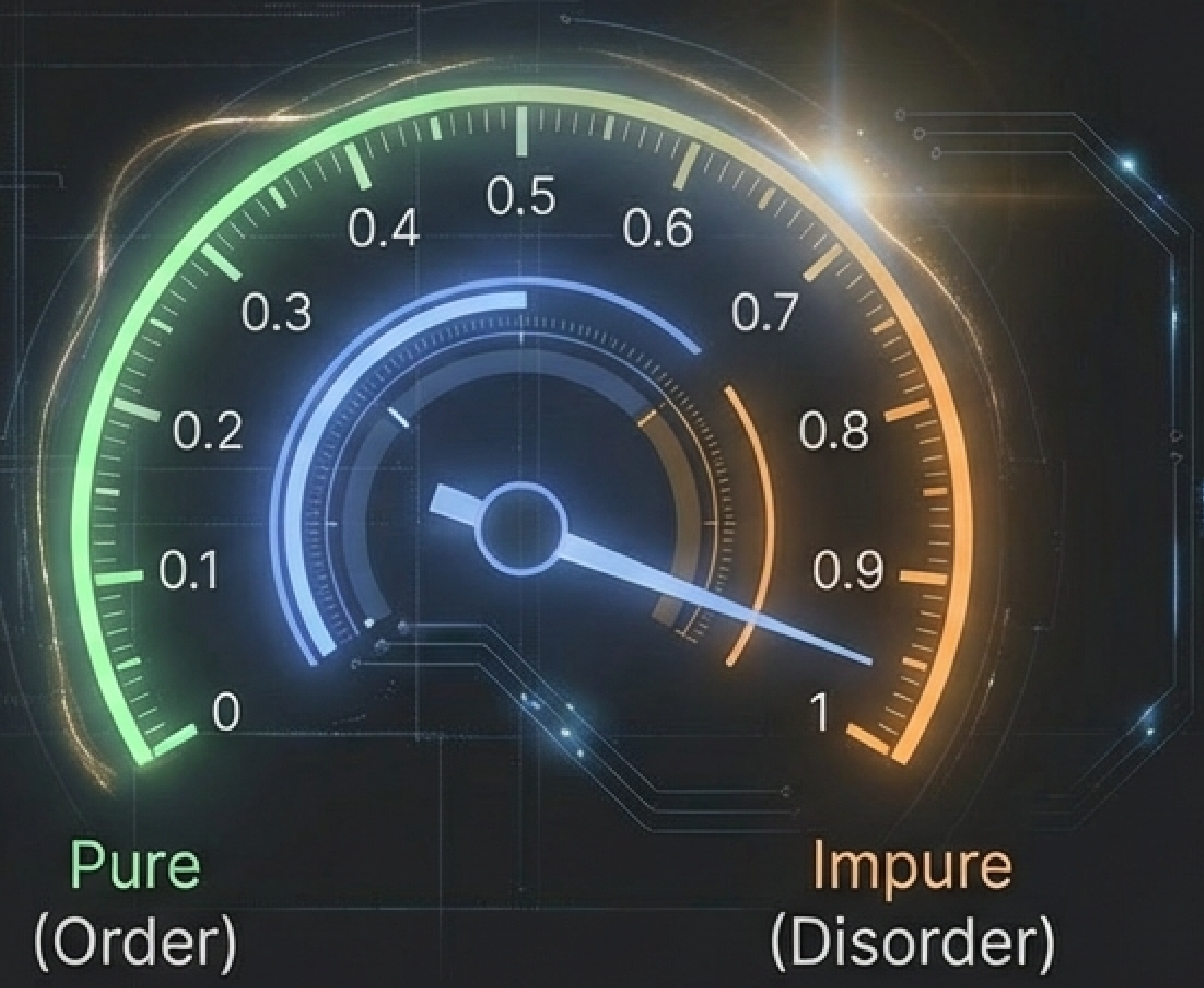
$$IG(root, child) = Impurity(root) - \sum \frac{n_i}{N} * Impurity(child_i)$$

- N = Total samples in parent
- n_i = Samples in child node
- **Goal:** Reduce impurity to create homogeneous groups.

JetBrains Mono

```
>>> ML.explain(model)
```

MEASURING IMPURITY: ENTROPY



Usage: Recommended for smaller datasets.

$$Entropy(s) = -\sum_i p_i * \log_2(p_i)$$

p_i = Probability of an output class.

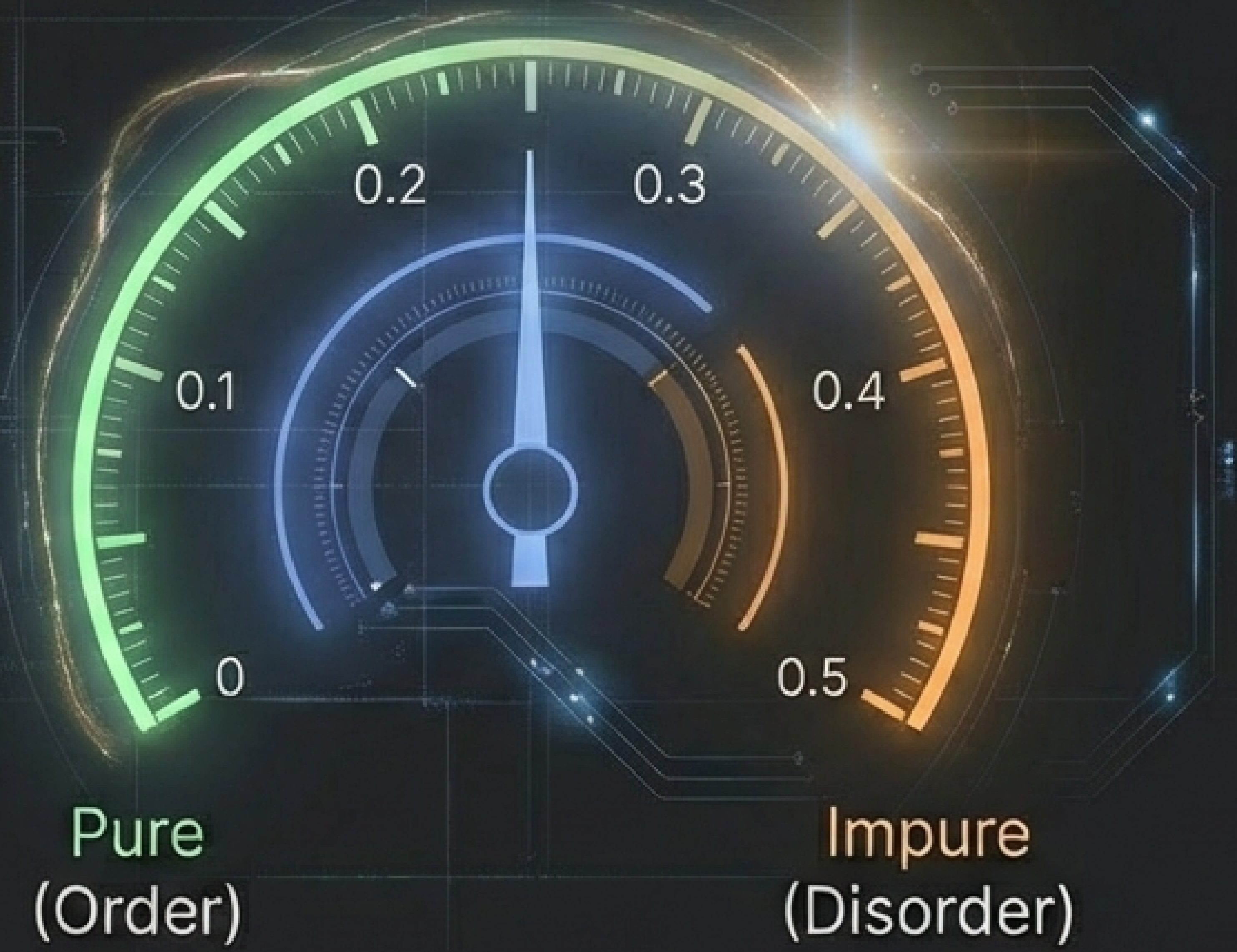
Entropy is a measure of randomness. If a bucket has only Red balls, Entropy = 0. If it's 50/50 Red and Blue, Entropy = 1.

JetBrains Mono

>>> ML.explain(model)

NotebookLM

MEASURING IMPURITY: GINI INDEX



Usage: Computationally faster (no logs). Standard for large datasets.

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

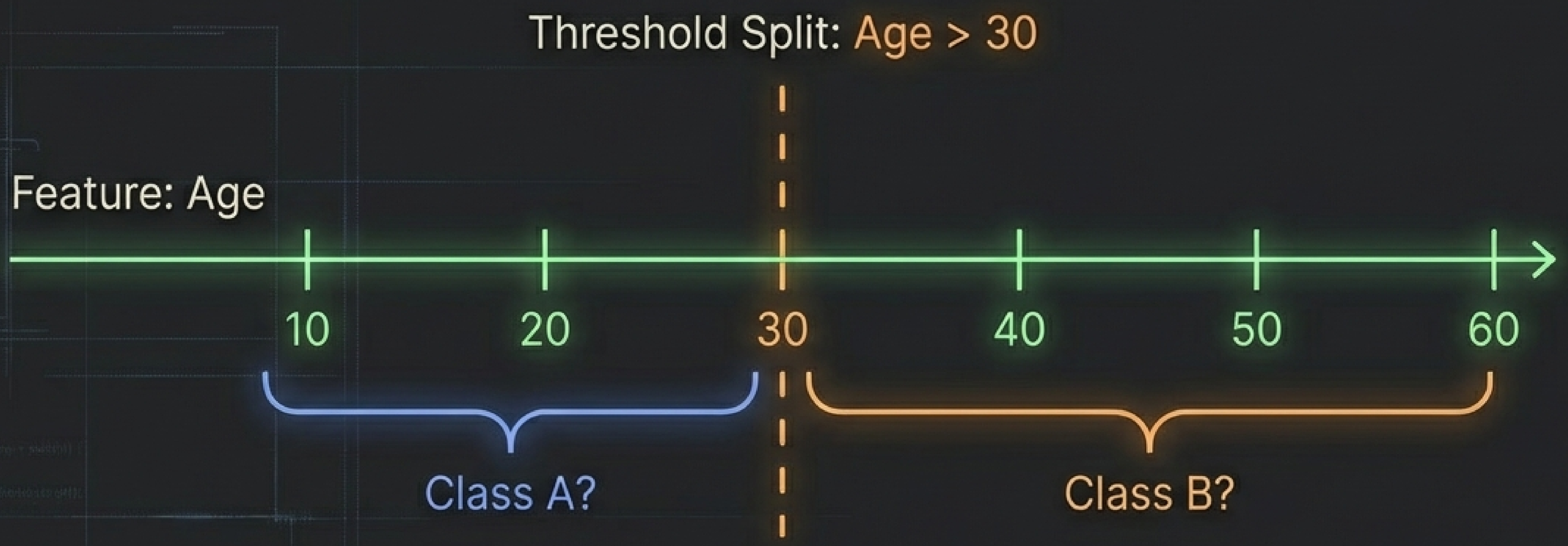
Comparison:

- Entropy: Range [0, 1]. Uses Logarithms (Slower).
- Gini: Range [0, 0.5]. Uses Squares (Faster).

JetBrains Mono

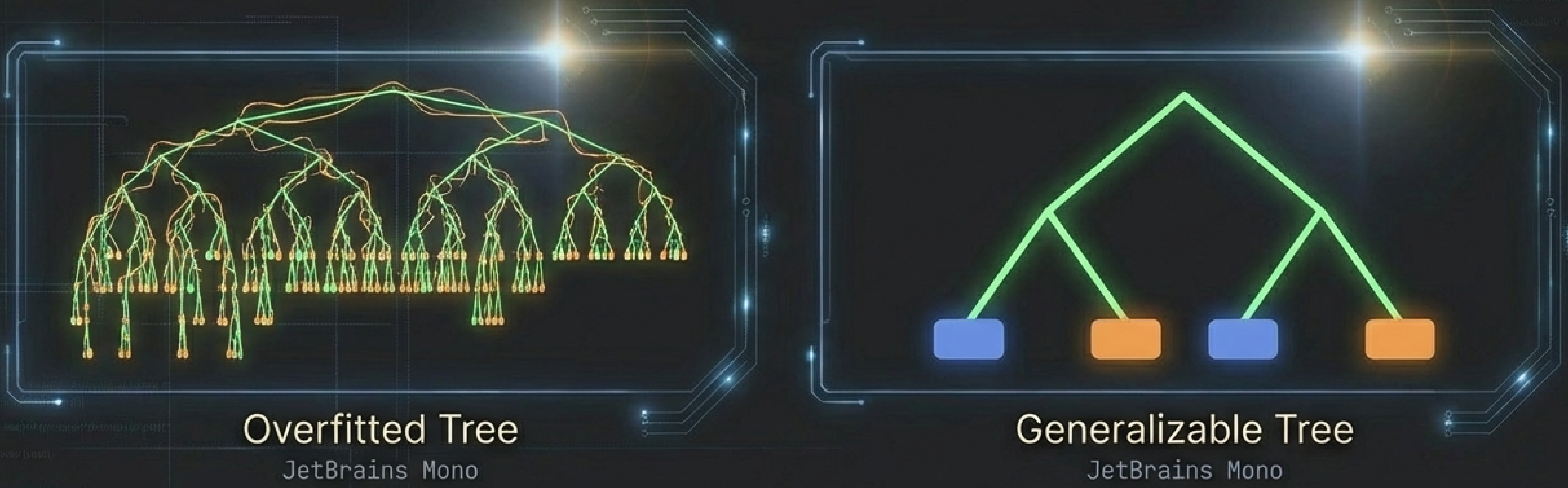
>>> ML.explain(model)

HANDLING CONTINUOUS VALUES



The algorithm converts continuous numbers into binary questions by testing different thresholds to find the highest Information Gain.

THE RISK OF OVERFITTING



Unpruned trees memorize training data, including noise. This results in 100% accuracy on training but failure on new, unseen data.

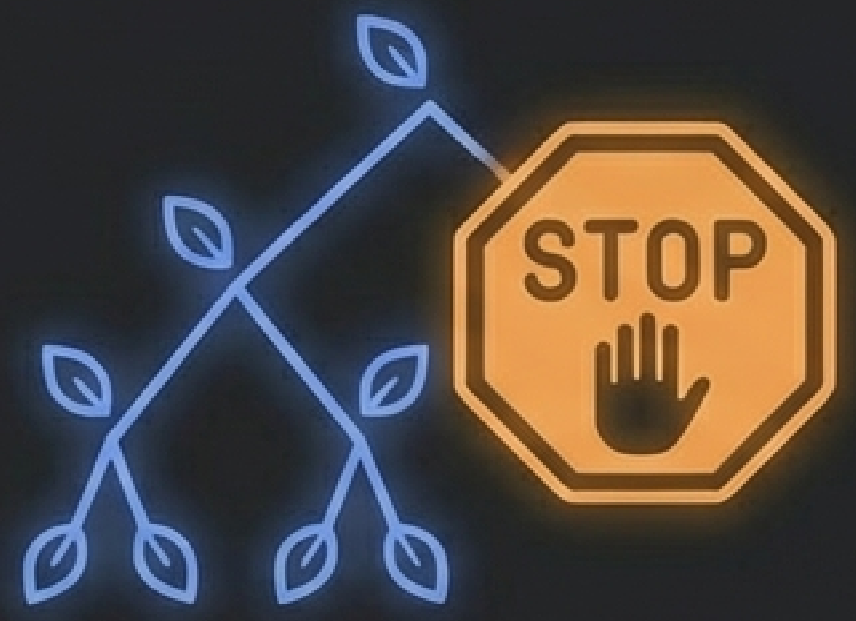
`>>> ML.explain(model)`

OPTIMIZATION: TREE PRUNING

Post-Pruning (Backward)

Grow full, then cut back. Best for small datasets.

Pre-Pruning (Early Stopping)



Stop growth when parameters (max_depth) are met. Best for large datasets.

>>> ML.explain(model)

IMPLEMENTATION: SETUP & DATA

CODE EDITOR

```python
import pandas as pd
from sklearn.datasets import load_iris
X, y = load_iris(return_X_y=True)
df = pd.DataFrame(X, columns=['sepal_len',
    'sepal_wid','petal_len','petal_wid'])

```

DATA PREVIEW

sepal_len	sepal_wid	petal_len	petal_wid
5.1	3.5	1.4	0.2
4.9	3.0	1.4	0.2
4.7	3.2	1.3	0.2
4.6	3.1	1.5	0.2
5.0	3.6	1.4	0.2

Dataset: 150 samples, 3 flower classes.

TRAINING THE MODEL

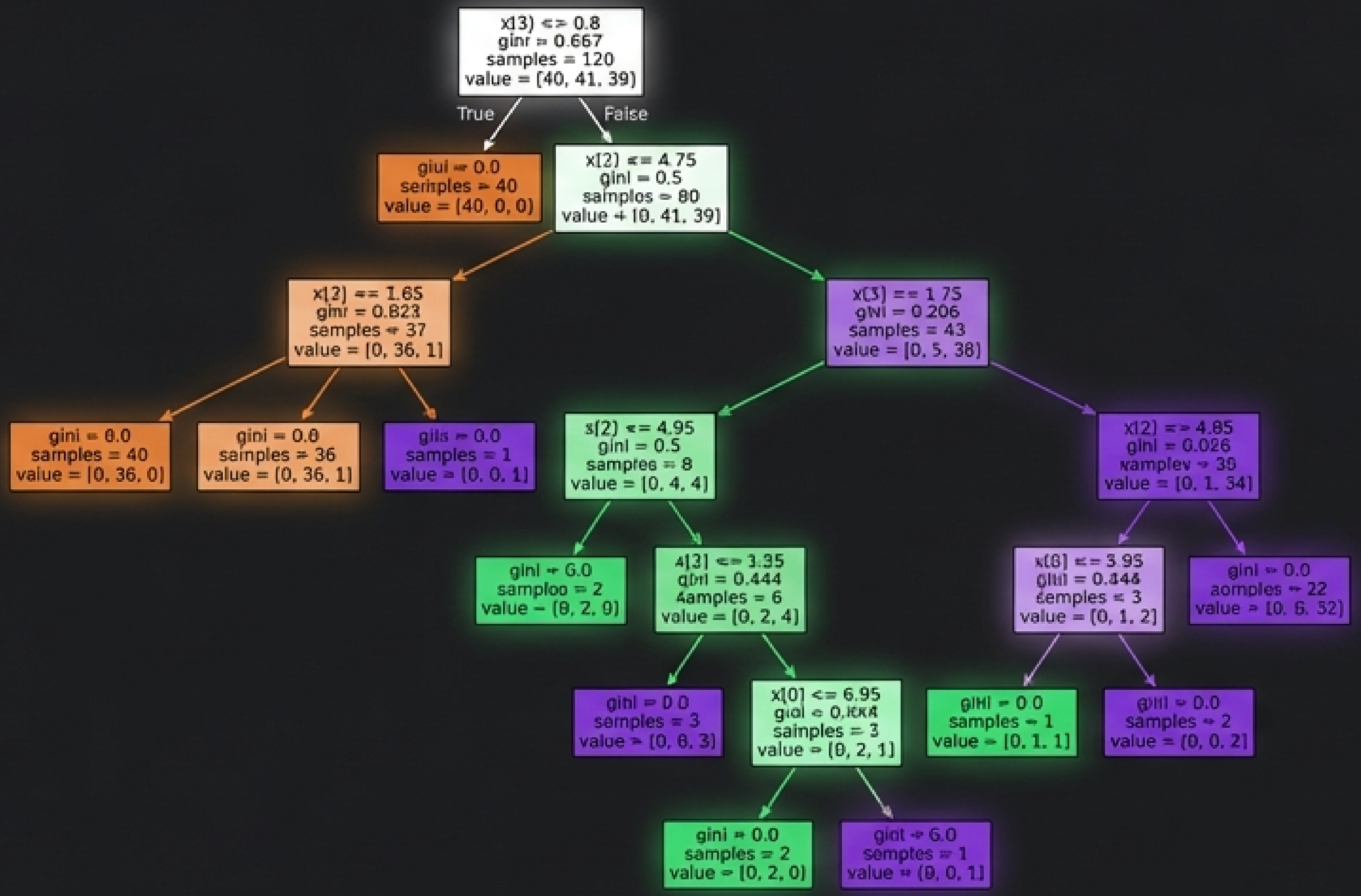
```
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier

# Split Data
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2)

# Initialize & Fit
model = DecisionTreeClassifier(criterion='gini')
model.fit(X_train, y_train)
```

The Math Happens Here. The algorithm is now calculating Gini indices to build the tree.

```
>> ML.explain(model)
```

VISUALIZING THE LOGIC

[code: tree.plot_tree(model, filled=True)]



This is the actual structure created by Python. Orange, Green, and Purple boxes represent the three different flower classes being separated.

DECODING THE OUTPUT

```
x[3] <= 0.8
gini = 0.667
samples = 120
value = [40, 41, 39]
```

Split Condition: Is Petal Width <= 0.8?

Impurity Score (High disorder)

Total samples in this node

Class distribution: 40 Setosa, 41 Versicolor, 39 Virginica

```
>> ML.explain(root_node)
```

PREDICTION & EVALUATION

```python
y_pred = model.predict(X_test)
print(accuracy_score(y_test, y_pred))
print(confusion_matrix(y_test, y_pred))
```

```
> Accuracy: 1.0
> Confusion Matrix:
[  10,   0,  0 ]
 [  0,   9,  0 ]
 [  0,   0, 11 ]
```

Diagonal values (10, 9, 11) indicate all 30 test samples were predicted correctly.

KEY TAKEAWAYS

1 **Interpretable**
2 Mimics human logic, transparent decision path.
3

1 **The Engine**
2 Driven by Information Gain (Entropy or Gini).
3

1 **Optimization**
2 Pruning is essential to prevent overfitting.
3

1 **Implementation**
2 Scikit-learn offers robust tools for training and visualization.