Lesson 26: Decision Trees

Data Science with Python – BSc Course

Data Science Program

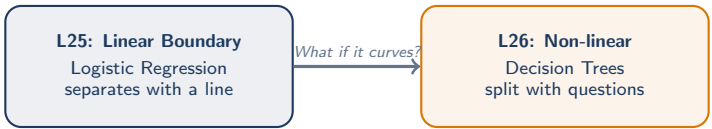
BSc Course

45 Minutes

Previously on L25...

Logistic regression draws a linear decision boundary.

The sigmoid function squashes any input to a probability between 0 and 1. But there is a catch: the boundary is always a straight line (or hyperplane).



Today: An algorithm that asks yes/no questions – no linearity assumed.

From straight lines to flexible splits: decision trees handle what logistic regression cannot

Learning Objectives

The Problem: Logistic regression assumes linear decision boundaries. What if the relationship between features and class is more complex?

After this lesson, you will be able to:

1. Build decision tree classifiers with sklearn
2. Understand splitting criteria (Gini impurity, Entropy)
3. Apply Random Forest for better generalization
4. Interpret feature importance from tree-based models

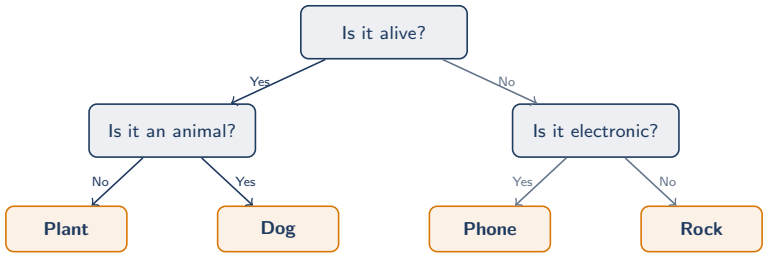
Decision trees: the most interpretable ML algorithm – every prediction has a clear explanation

The Game of Twenty Questions

A game you already know how to play:

With just 20 yes/no questions, you can identify any single object out of **1,048,576** possibilities. Why? Because $2^{20} = 1{,}048{,}576$.

Each question cuts the remaining options in half.



Decision trees work the same way – each node asks a question about a feature.

Twenty questions: the intuition behind every decision tree algorithm

What IS a Decision Tree?

A flowchart of yes/no questions about your data.

- Each question splits the data into two groups
- You follow the path of answers until you reach a prediction
- Every prediction has a clear, traceable explanation

Analogy – Doctor's Diagnosis:

- "Fever?" $\rightarrow$ Yes $\rightarrow$ "Cough?" $\rightarrow$ Yes $\rightarrow$ "Likely flu"
- Each question narrows down the possibilities

Why Trees Are Popular:

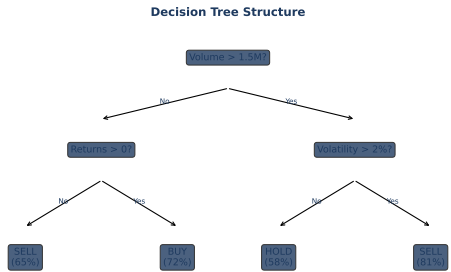
- **Interpretable:** explain WHY each prediction was made
- **No scaling needed:** unlike logistic regression or SVM
- **Mixed data:** handle numerical and categorical features

Decision trees: the most interpretable ML algorithm – every prediction has a clear explanation

How Trees Split: The Core Idea

Tree Terminology:

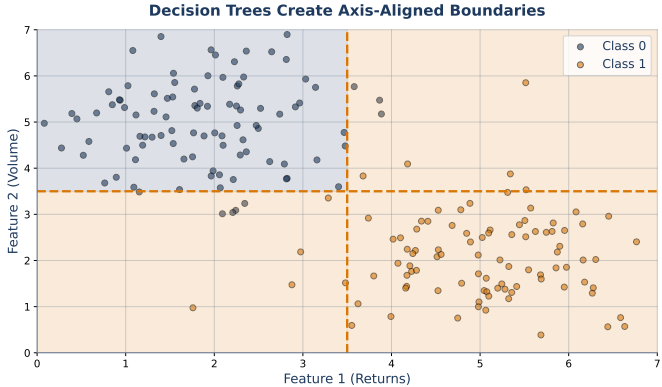
- **Root:** first question **Leaf:** final prediction **Depth:** number of levels
- **Internal node:** a split question (e.g., "Volatility > 0.02?")
- Each split makes groups more "pure" – algorithm tries all features and thresholds



Trees are interpretable: you can explain exactly why each prediction was made

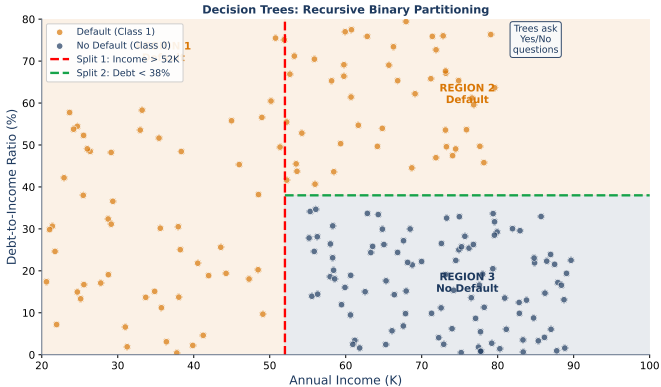
From Questions to Boundaries

Decision trees create **axis-aligned rectangular** decision regions – not curved, but many splits approximate complex shapes.



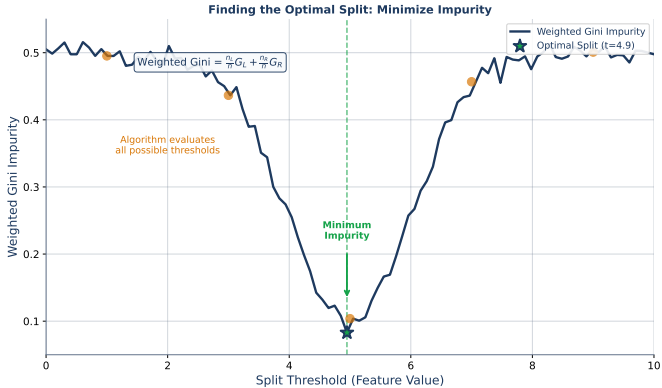
Each split is a horizontal or vertical line – many splits approximate any shape

Supplemental: Tree Intuition



Self-study: visual intuition for how trees partition feature space

Supplemental: The Splitting Process



Self-study: step-by-step view of recursive binary splitting

What Makes a "Good" Question?

Thought experiment:

A room has 50 dogs and 50 cats. You can ask one yes/no question. Which question separates them best?

- "Is it taller than 40cm?" $\rightarrow$ Maybe separates well
- "Does it have a collar?" $\rightarrow$ Probably not useful (both have collars)
- "Does it purr?" $\rightarrow$ Great split! Cats yes, dogs no

The best question creates the purest groups.

We need a way to *measure* purity. Two approaches:

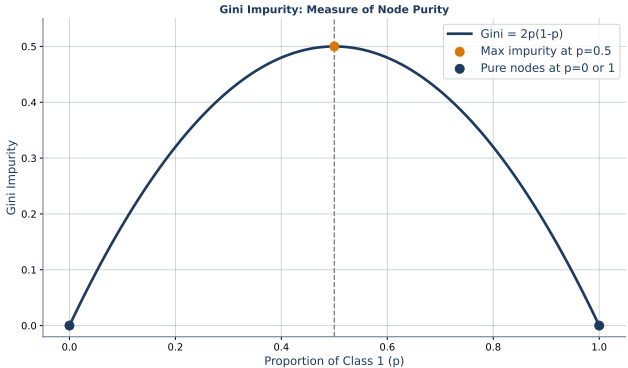
1. **Gini impurity** (sklearn default)
2. **Entropy** (information theory)

The algorithm tries every possible question and picks the one that maximizes purity gain

Gini Impurity: Measuring Purity

Formula: Gini $= 1 - \sum p_i^2$ Pure $= 0$, $50/50 = 0.5$ (sklearn default)

Example: 80 buys, 20 sells ($p_{\text{buy}} = 0.8$): Gini $= 1 - 0.8^2 - 0.2^2 = 0.32$



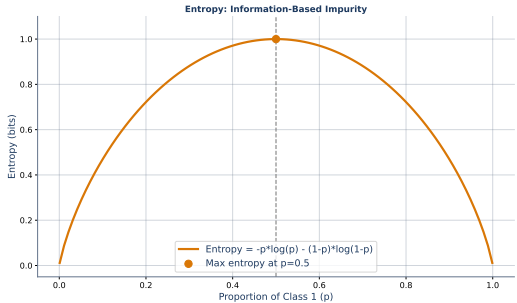
Lower Gini = purer node = better split. sklearn default: criterion='gini'

Entropy: Information Theory

Shannon, 1948: Entropy $= -\sum p_i \log_2 p_i$ Pure $= 0$, $50/50 = 1$ bit

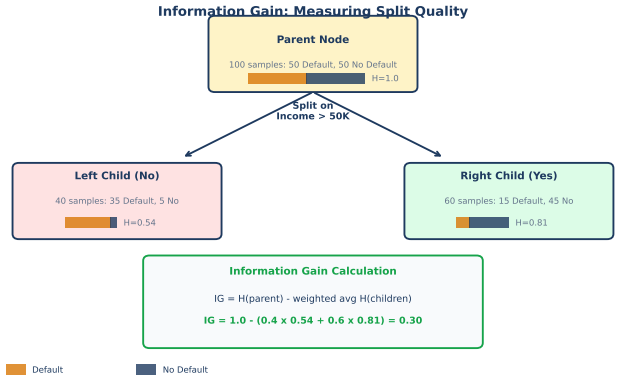
Information Gain $=$ Entropy$_{\text{before}}$ $-$ weighted Entropy$_{\text{after}}$

- Higher gain = better split (reduces uncertainty more)
- Usually gives same tree as Gini



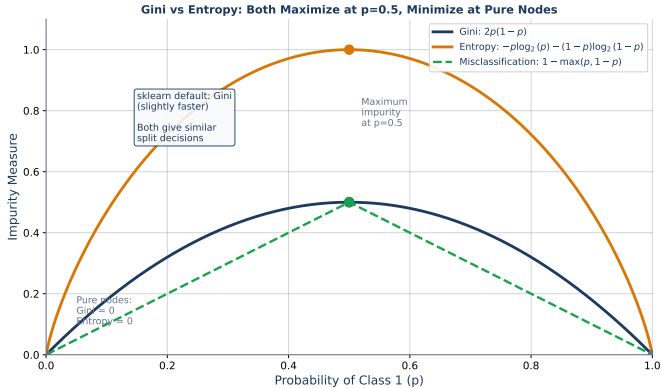
Gini and Entropy usually select the same splits. Use criterion='entropy' to compare.

Supplemental: Information Gain in Detail



Self-study: how information gain is computed step by step

Supplemental: Gini vs Entropy Comparison

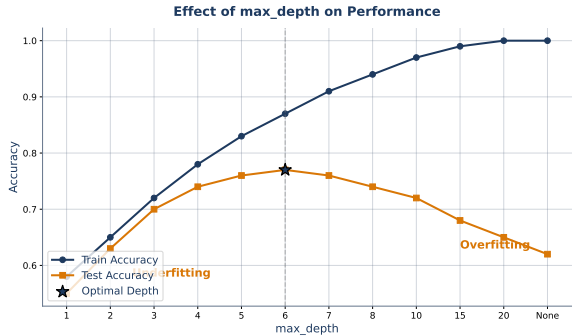


Self-study: side-by-side comparison of the two impurity measures

Controlling Tree Depth

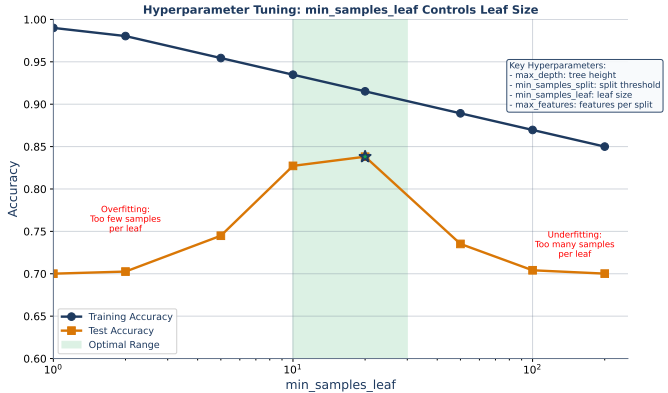
`max_depth` limits complexity

- Deeper trees capture more detail – but risk memorizing noise
- Always set `max_depth` to prevent uncontrolled growth



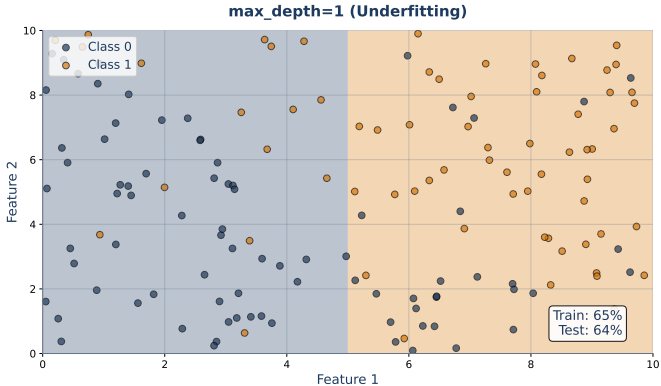
Always set max_depth – trees grow to full depth by default and will overfit

Supplemental: Hyperparameter Effects



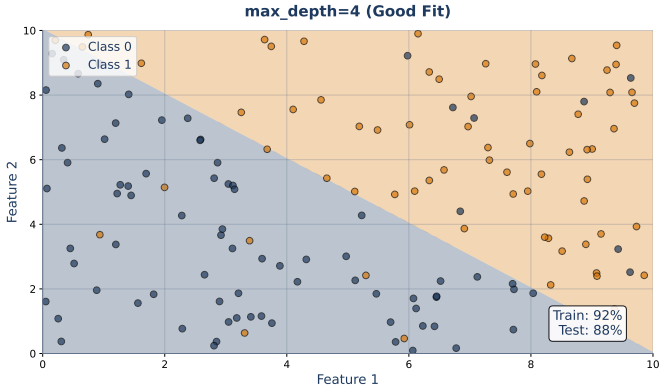
Self-study: how min_samples_leaf and min_samples_split affect tree complexity

Overfitting: Too Shallow



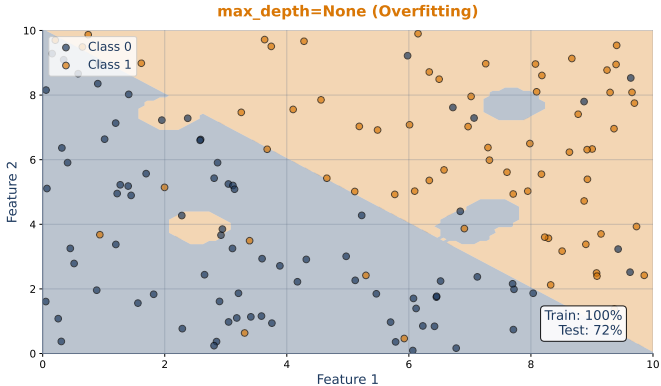
Too shallow: underfits – misses important patterns in the data

Overfitting: Just Right



Right depth: captures real patterns without memorizing noise

Overfitting: Too Deep



Too deep: memorizes training noise – poor generalization to new data

The Wisdom of Crowds

One tree is fragile. What if we trained 100 trees and let them vote?

Analogy:

- One expert might be wrong or biased
- Ask 100 experts, each seeing slightly different evidence
- Take the majority opinion – individual errors cancel out

This is exactly what Random Forest does.

- Train many trees on different subsets of the data
- Each tree makes a prediction
- Final answer = majority vote (classification) or average (regression)

Result: more stable, more accurate, less overfitting.

Random Forest: the "wisdom of crowds" applied to machine learning

Random Forest: How It Works

Two Sources of Diversity:

1. Bagging (Bootstrap Aggregating)

- Each tree trains on a random sample (with replacement)
- Different data $\rightarrow$ different perspective

2. Feature Randomization

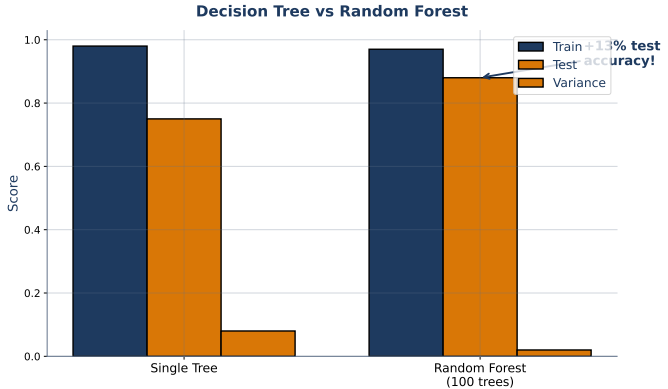
- At each split, consider only a random subset of features
- Prevents all trees from using the same "obvious" best feature
- Creates diverse trees that make uncorrelated errors

Same sklearn API:

```
from sklearn.ensemble import RandomForestClassifier
rf = RandomForestClassifier(n_estimators=100)
```

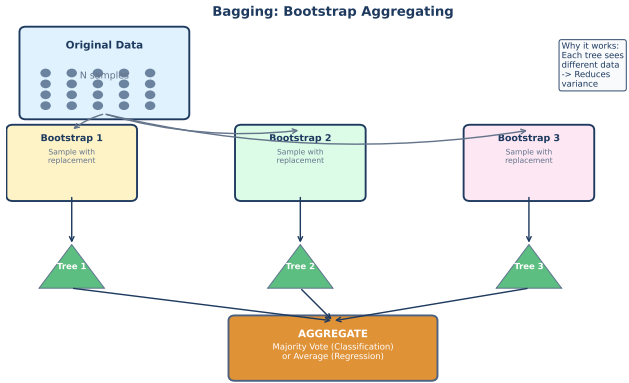
Two sources of randomness: random rows (bagging) + random features at each split

Forest vs Single Tree



Forest smooths out the jagged boundaries of a single tree – much more robust

Supplemental: Bagging Concept

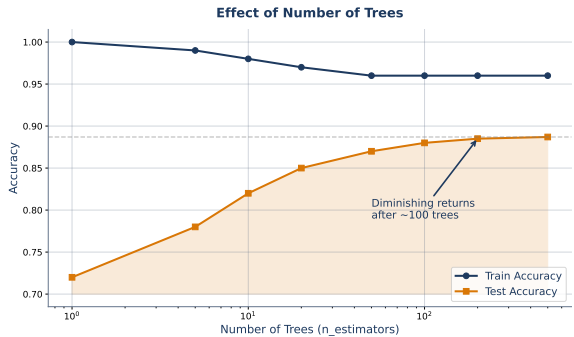


Self-study: how bootstrap samples create diversity among trees

How Many Trees? Diminishing Returns

Accuracy plateaus after ∼100–200 trees

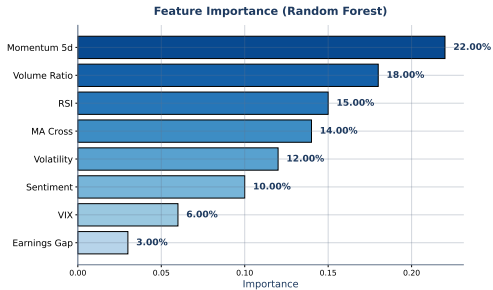
- More trees = more compute, little accuracy gain
- Default `n_estimators=100` is usually sufficient



Diminishing returns: 100 trees gives most of the benefit at a fraction of the cost

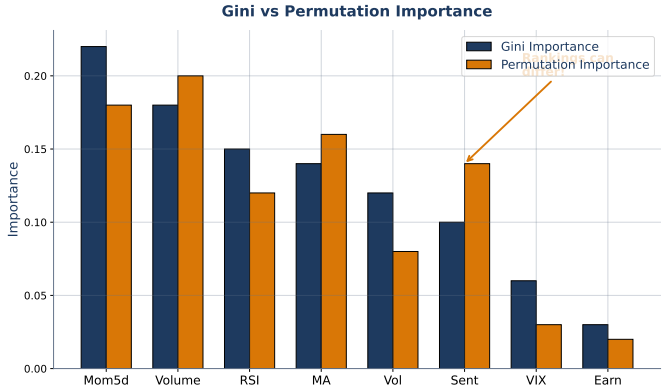
Feature Importance: Which Questions Matter?

1. **Gini-Based** – fast, computed during training (`model.feature_importances_`)
   - Sum of impurity reduction across all splits; biased toward high-cardinality
2. **Permutation** – shuffle one feature, measure accuracy drop
   - More reliable but slower; use for final interpretation



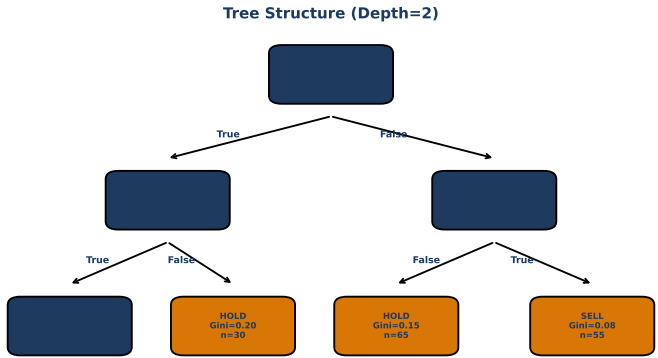
Quick exploration: Gini importance. Final interpretation: permutation importance.

Supplemental: Gini vs Permutation Importance



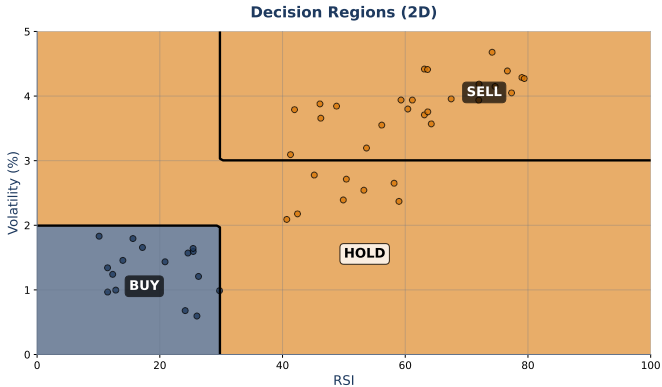
Self-study: permutation importance can reveal features masked by correlation

Supplemental: Tree Structure Visualization



Self-study: sklearn.tree.plot_tree() for inline tree visualization

Supplemental: Decision Surface



Self-study: visualization helps explain model decisions to non-technical stakeholders

Common Mistakes with Trees

× **Not setting `max_depth`**

Trees grow to full depth by default and memorize training data

✓ Set `max_depth=3` to 8, tune via cross-validation

× **Trusting Gini importance blindly**

Biased toward high-cardinality features with many unique values

✓ Use `permutation_importance()` for reliable ranking

× **Single tree in production**

Unstable – small data changes produce very different trees

✓ Always use Random Forest or another ensemble

Three mistakes beginners make – and how to avoid them

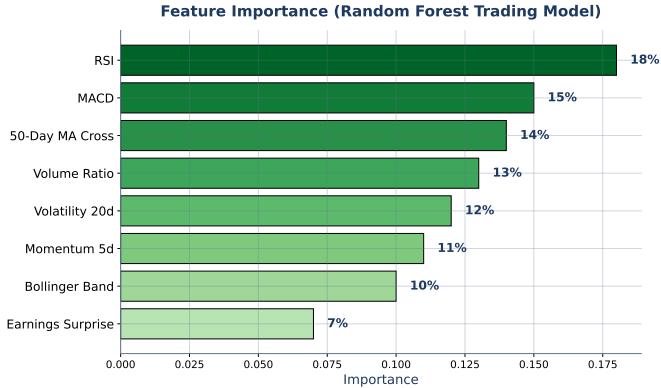
Finance: Trading Signal Classifier

Trees for rule-based strategies:

- "If PE < 15 AND debt_ratio < 0.5, predict BUY"
- Trees discover interpretable rules; importance shows which indicators matter

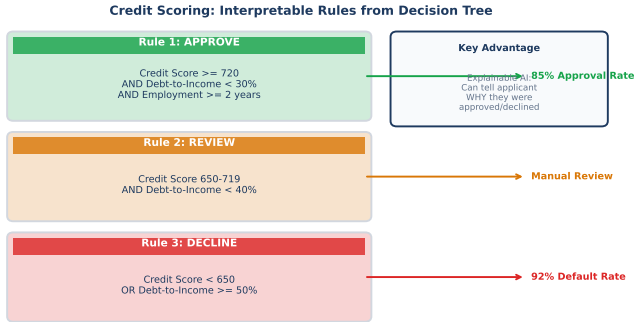
Trees naturally create rule-based strategies that traders can understand and audit

Supplemental: Trading Feature Importance



Self-study: which technical indicators drive trading signal predictions?

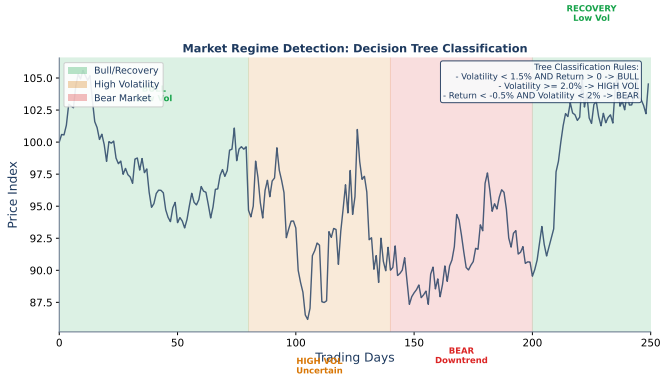
Supplemental: Credit Scoring Rules



Regulatory Compliance: Tree rules satisfy "right to explanation" requirements (GDPR, ECOA)

Self-study: decision trees for interpretable credit risk assessment

Supplemental: Regime Detection



Self-study: trees can identify market regimes from indicator combinations

Checkpoint: Pick the Right Tool

When logistic regression vs decision tree?

- **Logistic regression:** linear boundary, probability output, interpretable coefficients
- **Decision tree:** non-linear patterns, rule-based explanations, feature importance

When single tree vs Random Forest?

- **Single tree:** when you need a simple, fully explainable set of rules
- **Random Forest:** almost always – unless you specifically need single-tree interpretability

Rule of thumb: Start with Random Forest. If you need to explain individual predictions to a non-technical audience, consider a single shallow tree.

Random Forest is the safe default. Single trees are for explainability-critical applications.

Lesson Summary + Preview

Key Takeaways:

- Decision trees = yes/no questions that recursively split data
- Purity measures: Gini impurity (default) and Entropy (information theory)
- Control depth to avoid overfitting – always set `max_depth`
- Random Forest > single tree: wisdom of crowds reduces variance
- Feature importance reveals which variables drive predictions

Coming Next – L27: Classification Metrics

We have built classifiers (logistic regression, trees, forests). But how do we *measure* if they are any good?

Accuracy alone is not enough – especially when classes are imbalanced. Next lesson: precision, recall, F1, ROC curves, and the confusion matrix.

Trees = yes/no questions. Forest ¿ tree. Next: how to measure classifier quality beyond accuracy.