Lesson 26: Decision Trees

Data Science with Python – BSc Course

45 Minutes

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:

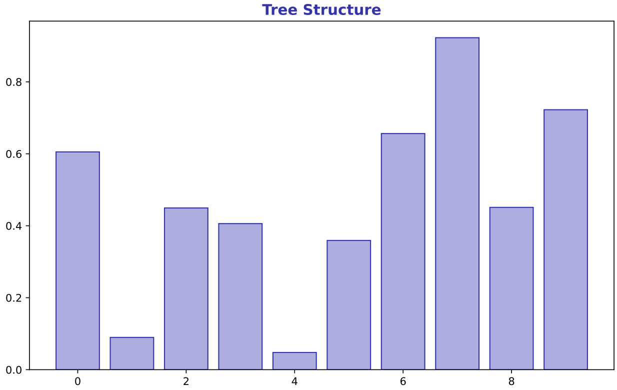
- Build decision tree classifiers
- Understand splitting criteria (Gini, Entropy)
- Apply Random Forest for better generalization
- Interpret feature importance from tree models

Finance Application: Rule-based credit scoring and trading signals

Tree Structure

If-Then Rules as a Tree

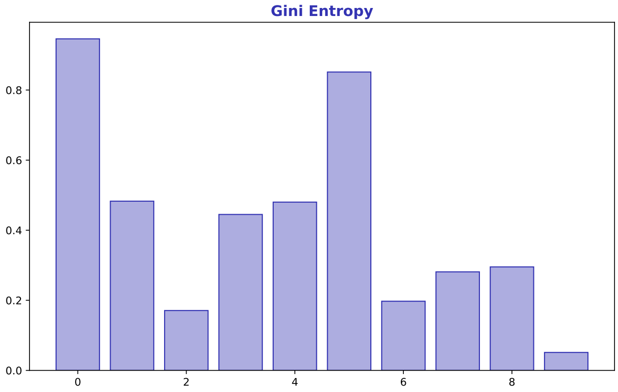
- Root node: first split on most informative feature
- Leaf nodes: final class predictions



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

How to Choose the Best Split?

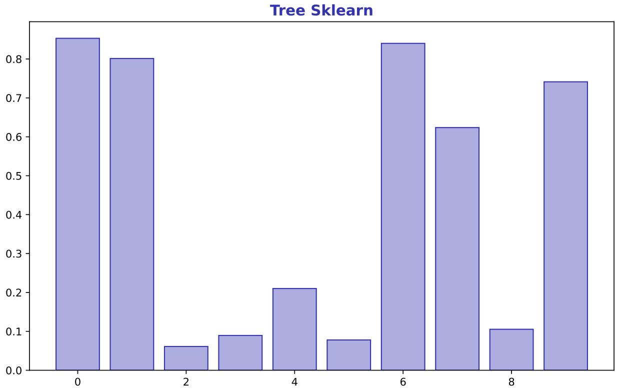
- Gini: $1 - \sum p_i^2$ – measures impurity (lower = purer)
- Entropy: $-\sum p_i \log p_i$ – information gain criterion



In practice: Gini and Entropy give similar results. sklearn default is Gini.

Building Trees in Python

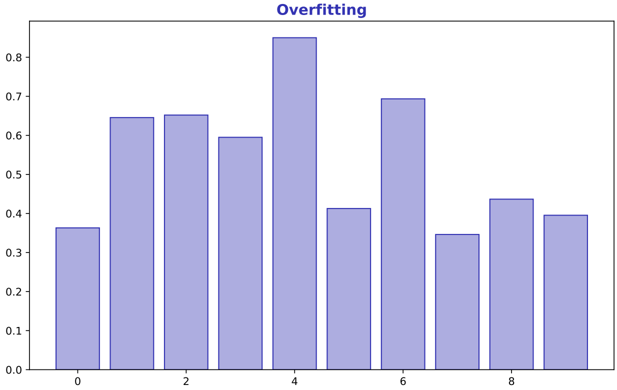
- `from sklearn.tree import DecisionTreeClassifier`
- Key params: `max_depth`, `min_samples_leaf`



Always set max_depth to prevent trees from memorizing training data

The Danger of Deep Trees

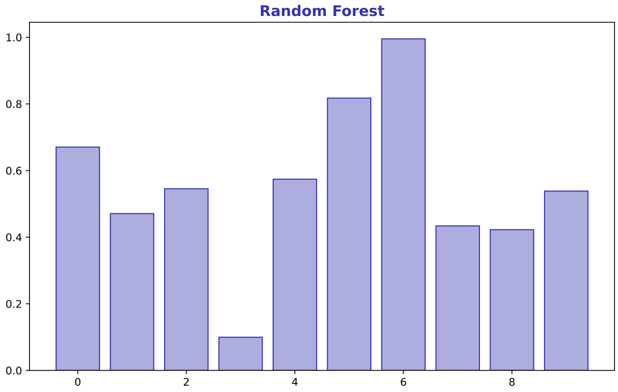
- Unlimited depth: tree can perfectly fit training data (100% accuracy)
- Test accuracy often much worse – memorization, not learning



Rule: Start with max_depth=3, increase only if underfitting

Ensemble of Trees

- Train many trees on random subsets of data and features
- Final prediction: majority vote (classification) or average (regression)

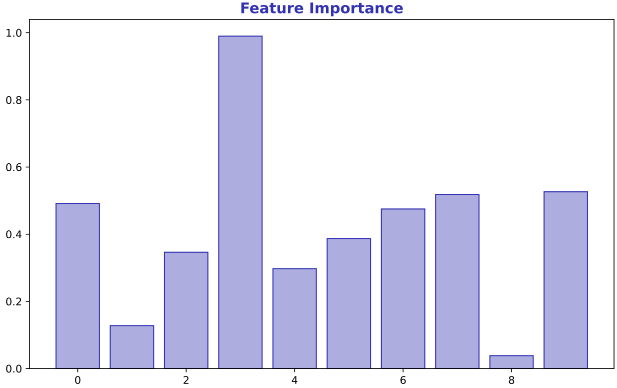


Random Forest = bagging + feature randomization. Much more robust than single tree.

Feature Importance

Which Features Matter Most?

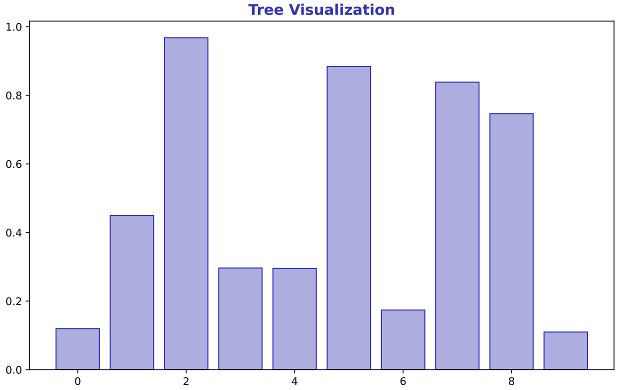
- Importance = total reduction in impurity from splits on that feature
- Normalized to sum to 1.0 across all features



Access via `model.feature_importances_` **– useful for feature selection**

Making Trees Interpretable

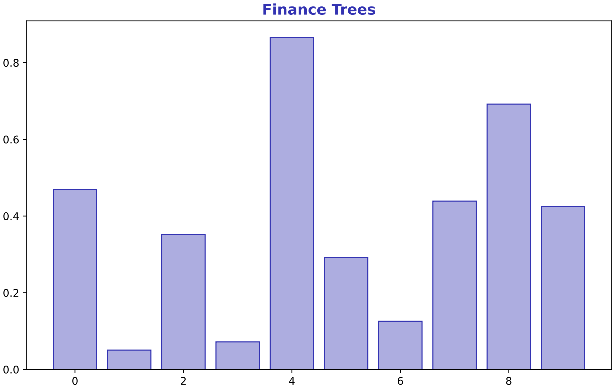
- `sklearn.tree.plot_tree()` for inline visualization
- Export to Graphviz for publication-quality diagrams



Visualization helps explain model decisions to non-technical stakeholders

Trading Rules from Trees

- Trees naturally create rule-based strategies
- Example: "If RSI $<$ 30 AND volume $>$ 2$\times$avg, then BUY"



Caution: Trees overfit easily on financial data – use cross-validation

Hands-On Exercise (25 min)

Task: Build a Trading Signal Classifier

1. Features: RSI, MACD, Bollinger Band position, volume ratio
2. Target: 1 if next-5-day return $> 2\%$, else 0
3. Train DecisionTree with max_depth=4 and RandomForest
4. Compare test accuracy – which generalizes better?
5. Plot feature importance for Random Forest

Deliverable: Tree visualization + feature importance bar chart.

Extension: Try different max_depth values – plot train vs test accuracy

Lesson Summary

Problem Solved: Decision trees capture non-linear relationships and produce interpretable rules.

Key Takeaways:

- Trees split on features that maximize information gain
- Control complexity via max_depth, min_samples_leaf
- Random Forest: many trees > one tree (reduces overfitting)
- Feature importance reveals which variables drive predictions

Next Lesson: Classification Metrics (L27) – beyond accuracy

Memory: Single tree overfits. Random Forest averages many trees for stability.