L04: Random Forests

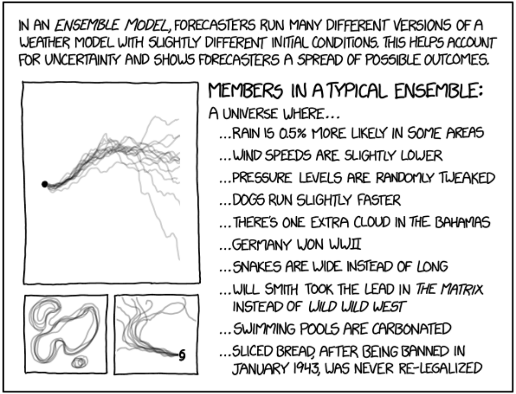
Ensemble Learning for Robust Predictions

Methods and Algorithms

Spring 2026

Outline

Three zones: Introduction, Core Content (PMSP), and Wrap-Up



When one model is not enough, why not use all of them?

The Wisdom of Crowds

Combining opinions beats relying on one expert

- **Voting and polling:** election forecasts aggregate many polls, not just one
- **Audience lifelines:** "Ask the Audience" on quiz shows beats "Phone a Friend"
- **Medical diagnosis:** second opinions reduce misdiagnosis rates significantly

The core insight:

- Individual predictions are noisy and biased in different directions
- Averaging many independent estimates cancels out individual errors
- The crowd is smarter than any single member

Francis Galton (1907): crowd's median estimate of an ox's weight was within 1% of the true value

From One Tree to a Forest

One person's opinion vs. the wisdom of the crowd

- A single decision tree is like asking one analyst for a recommendation
- Small changes in data can completely change that analyst's conclusion
- The result: unstable, high-variance predictions you cannot rely on

The forest solution

- Train many trees, each on a slightly different version of the data
- Combine their predictions: stable, robust, reliable
- In fraud detection: one tree may miss a pattern; 500 trees will catch it

---

Random Forests trade a small increase in bias for a large reduction in variance

Why Banks Use Ensembles

Regulatory and business requirements

- **Accuracy:** fraudulent transactions cost the banking industry billions annually
- **Interpretability:** regulators (Basel, EBA) demand explanations for every decision
- **Robustness:** models must perform consistently across different market conditions

Why Random Forests fit the bill

- Built-in feature importance satisfies explainability requirements
- Ensemble averaging resists overfitting to noisy transaction data
- No feature scaling needed – handles mixed data types naturally

---

Random Forests remain a top choice in banking for fraud, credit scoring, and AML

Learning Objectives

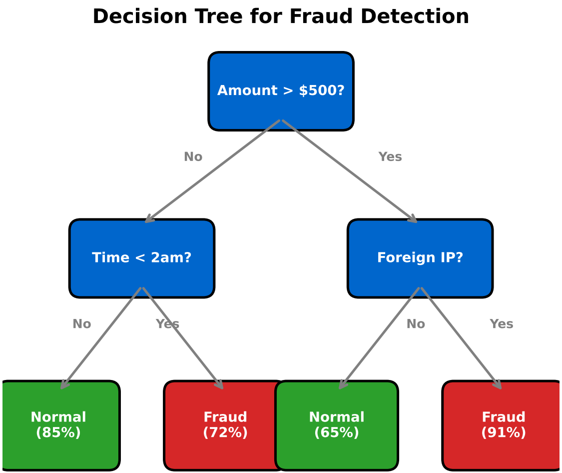
By the end of this lecture, you will be able to:

1. **Analyze** how bootstrap aggregating reduces prediction variance through tree decorrelation
2. **Evaluate** Random Forest hyperparameters for optimal bias-variance tradeoff
3. **Compare** feature importance methods (MDI, permutation, SHAP) for model interpretation
4. **Critique** ensemble methods for regulatory compliance in fraud detection

Finance Application: Fraud detection with interpretable feature importance

---

Bloom's Level 4–5: Analyze, Evaluate, Compare, Critique — MSc-level objectives



https://github.com/Digital-AI-Finance/methods-algorithms/tree/master/slides/L04_Random_Forests/01_decision_tree

Decision trees are the building blocks of Random Forests – each tree partitions the feature space recursively

The Overfitting Problem

Why single decision trees are unreliable

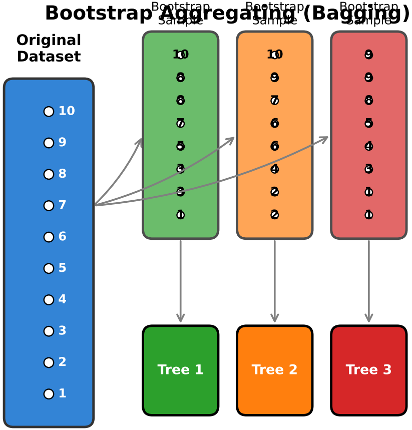
- **High variance:** small changes in training data produce completely different trees
- **Sensitivity:** adding or removing a few samples can change every split in the tree
- **Overfitting:** deep trees memorize noise instead of learning generalizable patterns

The consequences in practice

- A fraud detection tree trained on Monday's data may fail on Tuesday's transactions
- Unstable predictions undermine trust in the model and regulatory confidence
- We need a method that keeps the flexibility of trees but eliminates the instability

---

The bias-variance tradeoff: single trees have low bias but dangerously high variance



Each tree trained on ~63% unique samples (with replacement)

https://github.com/Digital-AI-Finance/methods-algorithms/tree/master/slides/L04_Random_Forests/03_bootstrap

Key idea: draw B bootstrap samples (with replacement), train one tree on each, average predictions.

Bagging: sampling with replacement creates diverse training sets – each tree sees roughly 63.2% of original data

Why does averaging trees reduce variance?

$$\mathrm{Var}(\bar{f}) = \rho\,\sigma^2 + \frac{1-\rho}{B}\,\sigma^2$$

- ρ = average pairwise correlation between trees
- σ^2 = variance of a single tree
- B = number of trees in the ensemble

Two levers for variance reduction:

- **Increase B:** more trees shrink the second term toward zero
- **Decrease ρ:** decorrelating trees shrinks the dominant first term

Key insight: once B is large enough, reducing ρ is the only way to improve further.

This formula is the theoretical foundation of Random Forests – decorrelation is the key innovation

The Random Forest Innovation

Two sources of randomness

- **Bootstrap sampling:** each tree trains on a different random subset of observations
- **Feature randomization:** at each split, consider only m randomly chosen features

How many features per split?

- Classification: $m \approx \sqrt{p}$ (e.g., 10 features from 100)
- Regression: $m \approx p/3$ (e.g., 33 features from 100)

Why this works: Feature subsampling prevents dominant predictors from appearing in every tree, which **decorrelates** the trees and reduces ρ in the variance formula.

Breiman (2001): the combination of bagging + feature randomization is what makes Random Forests so effective

Gini Impurity and Information Gain

Gini Impurity (default in scikit-learn):

$$G = 1 - \sum_{k=1}^{K} p_k^2$$

Entropy (information-theoretic alternative):

$$H = -\sum_{k=1}^{K} p_k \log_2(p_k)$$

Information Gain from splitting on feature A:

$$\text{IG}(A) = H(\text{parent}) - \sum_j \frac{n_j}{n} H(\text{child}_j)$$

- Gini and Entropy produce nearly identical splits in practice
- The tree greedily picks the feature and threshold that maximizes IG at each node

Both criteria measure impurity – the tree splits to create purer child nodes at each step

Out-of-Bag (OOB) Error

Free cross-validation built into bagging

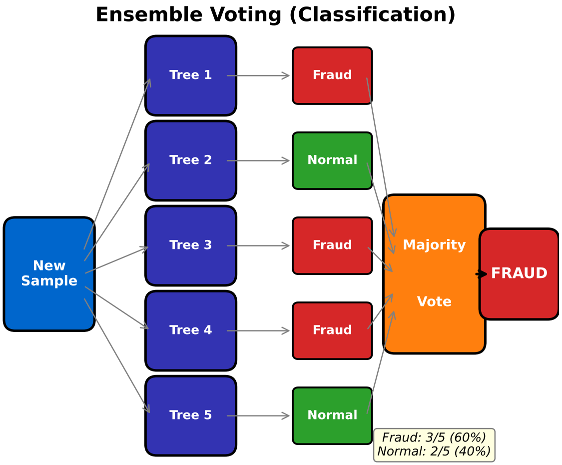
- Each bootstrap sample excludes roughly 36.8% of observations
- Probability of exclusion: $(1 - 1/n)^n \to 1/e \approx 0.368$ as $n \to \infty$
- For each observation, predict using *only* the trees that did not train on it

Why OOB error matters:

- **No separate validation set needed** – saves precious labeled data
- Closely approximates leave-one-out cross-validation accuracy
- Available at no extra computational cost during training

Practical use: Set `oob_score=True` in scikit-learn to monitor generalization performance during training without any additional code.

OOB error is one of the key practical advantages of Random Forests over other ensemble methods



https://github.com/Digital-AI-Finance/methods-algorithms/tree/master/slides/L04_Random_Forests/05_ensemble_voting

Classification: majority vote across all trees. Regression: average of all tree predictions.

Feature Importance Methods

Three approaches to understanding what the model learned:

- **MDI (Mean Decrease in Impurity):** sum of Gini/Entropy reductions from splits on each feature across all trees. Fast but biased toward high-cardinality features.
- **Permutation Importance:** randomly shuffle one feature, measure how much accuracy drops. Model-agnostic and unbiased, but slower.
- **SHAP Values:** game-theoretic approach assigning each feature a contribution to each individual prediction. Most informative but computationally expensive.

Recommendation: Use permutation importance for model selection, SHAP for regulatory explanations.

Feature importance is what makes Random Forests interpretable – critical for regulated industries

Fraud Detection with Random Forests

The class imbalance problem

- Fraudulent transactions are typically less than 1% of all transactions
- A naive model predicting "not fraud" achieves 99%+ accuracy – but catches zero fraud
- **Accuracy is the wrong metric** for imbalanced classification

Correct evaluation metrics:

- **Precision:** of flagged transactions, how many are actually fraud?
- **Recall:** of all fraud, how much did we catch?
- **AUC-PR:** area under the Precision-Recall curve (preferred over ROC for imbalanced data)

Use `class_weight='balanced'` in scikit-learn to upweight the minority class automatically.

In fraud detection, a missed fraud (false negative) is far more costly than a false alarm (false positive)

Handling Class Imbalance

Data-level strategies

- **SMOTE:** generate synthetic minority samples by interpolating between existing fraud cases
- **Undersampling:** reduce majority class to match minority – risks losing information
- **Threshold tuning:** adjust the classification threshold to favor recall over precision

Cost-sensitive learning

- Assign higher misclassification cost to false negatives (missed fraud)
- In banking: cost of missing one fraud case far exceeds cost of investigating a false alarm
- Random Forests support cost-sensitive learning via `class_weight` and `sample_weight`

Combine multiple strategies: SMOTE + threshold tuning + cost-sensitive weights for best results

Random Forests vs. Boosting

Property	Random Forest	Boosting
Training approach	Parallel (independent trees)	Sequential (correct errors)
Primarily reduces	Variance	Bias
Overfitting risk	Low	Higher (needs careful tuning)
Hyperparameter sensitivity	Low	High
State-of-the-art accuracy	Competitive	Often superior
Interpretability	Feature importance built-in	SHAP required

Modern boosting: XGBoost, LightGBM, and CatBoost dominate Kaggle competitions and production ML systems. Random Forests remain the robust, low-tuning baseline.

Both are ensemble methods but with fundamentally different strategies for improving predictions

From Bagging to Boosting

Bagging (Random Forests):

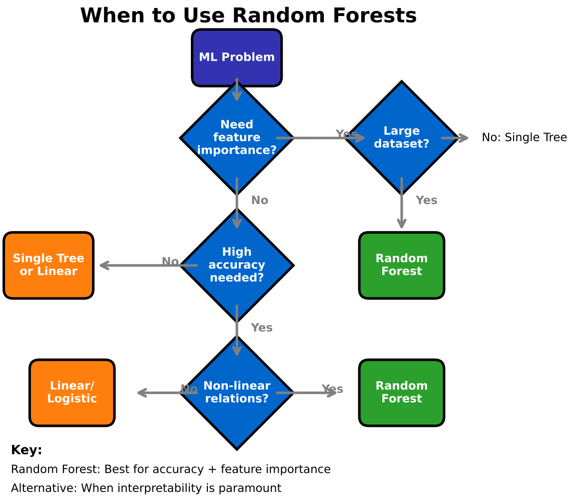
- Train trees **in parallel** on independent bootstrap samples
- Each tree votes equally – reduces variance by averaging
- Trees are deliberately decorrelated via feature randomization

Boosting (XGBoost, LightGBM):

- Train trees **sequentially** – each new tree focuses on previous errors
- Core idea: fit the next tree to the residuals of the current ensemble
- Later trees get lower weight to prevent overfitting (learning rate)

When to choose which: Use RF when you need a reliable baseline with minimal tuning. Use boosting when you need maximum predictive accuracy and can invest in hyperparameter optimization.

Bagging reduces variance (parallel, independent). Boosting reduces bias (sequential, adaptive).



https://github.com/Digital-AI-Finance/methods-algorithms/tree/master/slides/L04_Random_Forests/07_decision_flowchart

Use this flowchart to decide whether Random Forests are the right tool for your problem

Hands-on Exercise

Open the Colab Notebook

1. **Exercise 1:** Train a single decision tree on credit card transaction data and visualize its structure
2. **Exercise 2:** Build a Random Forest, compare OOB error to test error, and analyze feature importance
3. **Exercise 3:** Tune `n_estimators`, `max_depth`, and `max_features` using cross-validation

Challenge: Can you beat the single tree's AUC-PR by at least 10% with your tuned Random Forest? **Link:**

`https://colab.research.google.com/` – see course materials for notebook

Hands-on practice: apply the theory to real credit card fraud data with scikit-learn

Key Takeaways

What you should remember from this lecture:

1. **Ensemble = Wisdom of Crowds** – combining many weak learners produces a strong learner
2. **Bootstrap + Feature Randomization** – two sources of randomness decorrelate trees and reduce variance
3. **OOB Error for Free CV** – built-in validation without holding out data
4. **Feature Importance for Interpretation** – MDI, permutation, and SHAP make Random Forests explainable for regulators

Next lecture: PCA and t-SNE – dimensionality reduction for visualization and feature engineering

---

Random Forests: robust, interpretable, and production-ready – the reliable workhorse of applied ML

Closing Thought

"Pour the data into this pile of decision trees and see what comes out."

The beauty of Random Forests: sometimes the simplest ensemble approach is exactly what you need.

Adapted from XKCD #1838 "Machine Learning" by Randall Munroe (CC BY-NC 2.5)

References

- Breiman, L. (2001). *Random Forests*. Machine Learning, 45(1), 5–32.
- James, G., Witten, D., Hastie, T., & Tibshirani, R. (2021). *An Introduction to Statistical Learning*, 2nd ed., Chapter 8. https://www.statlearning.com/
- Hastie, T., Tibshirani, R., & Friedman, J. (2009). *The Elements of Statistical Learning*, 2nd ed., Chapter 15. https://hastie.su.domains/ElemStatLearn/
- Chen, T. & Guestrin, C. (2016). *XGBoost: A Scalable Tree Boosting System*. Proceedings of KDD 2016, 785–794.

Recommended reading: ISLR Chapter 8 for intuition, ESL Chapter 15 for mathematical depth