L04: Random Forests

Full Lecture: Ensemble Learning, Variance Reduction, and Fraud Detection

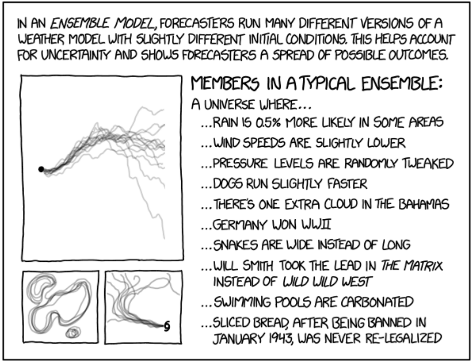
Methods and Algorithms

MSc Data Science

The Ensemble Approach

The cartoon captures the absurdity – and power – of combining models.

- One model is fragile; many models are robust
- Ensembles exploit the **wisdom of crowds**: independent errors cancel
- Random Forests, boosting, and stacking all follow this principle
- The key insight: **diversity** among models matters more than individual accuracy


IN AN *ENSEMBLE MODEL*, FORECASTERS RUN MANY DIFFERENT VERSIONS OF A WEATHER MODEL WITH SLIGHTLY DIFFERENT INITIAL CONDITIONS. THIS HELPS ACCOUNT FOR UNCERTAINTY AND SHOWS FORECASTERS A SPREAD OF POSSIBLE OUTCOMES.
MEMBERS IN A TYPICAL ENSEMBLE:
A UNIVERSE WHERE...
...RAIN IS 0.5% MORE LIKELY IN SOME AREAS
...WIND SPEEDS ARE SLIGHTLY LOWER
...PRESSURE LEVELS ARE RANDOMLY TWEAKED
...DOGS RUN SLIGHTLY FASTER
...THERE'S ONE EXTRA CLOUD IN THE BAHAMAS
...GERMANY WON WWII
...SNAKES ARE WIDE INSTEAD OF LONG
...WILL SMITH TOOK THE LEAD IN *THE MATRIX* INSTEAD OF *WILD WILD WEST*
...SWIMMING POOLS ARE CARBONATED
...SLICED BREAD, AFTER BEING BANNED IN JANUARY 1943, WAS NEVER RE-LEGALIZED

XKCD #1885 by Randall Munroe (CC BY-NC 2.5)

Learning Objectives

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

1. **Derive** the variance reduction formula for ensemble averaging (Analyze)
2. **Evaluate** RF vs. boosting through the bias-variance lens (Evaluate)
3. **Analyze** feature importance: MDI, permutation, SHAP (Analyze)
4. **Apply** ensembles to fraud detection with class imbalance (Apply)
5. **Compare** RF, AdaBoost, gradient boosting, XGBoost (Evaluate)

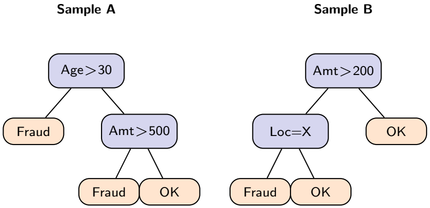
Bloom's Taxonomy Levels

- Analyze – break down, derive, compare components
- Evaluate – judge tradeoffs, assess suitability
- Apply – use methods on real data problems

L04 covers ensemble methods from bagging to boosting with finance applications.

Why Would a Single Tree Overfit Every Dataset It Touches?

- A fully grown tree has **low bias** (fits training data perfectly)
- But **high variance**: small data changes produce entirely different trees
- Bias-variance decomposition: $\text{MSE} = \text{Bias}^2 + \text{Variance} + \text{Noise}$
- Trees are **unstable learners** – ideal building blocks for ensembles



Different training samples ⇒ completely different tree structures.

Insight

High variance is a *feature*, not a bug – it means trees are sensitive enough to be improved by averaging.

Unstable learners benefit most from ensemble methods (Breiman, 1996).

How Does a Tree Choose the Best Split? (Gini Impurity)

Gini Impurity measures how often a random sample would be misclassified:

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

Binary case: $G = 2p(1 - p)$

Worked example (fraud data):

- Node: 800 legit, 200 fraud $\Rightarrow p_{\text{fraud}} = 0.2$
- $G = 1 - (0.8^2 + 0.2^2) = 1 - 0.68 = 0.32$
- Best split minimizes weighted Gini of children

Split evaluation:

- Left child: 50 fraud / 100 total $\Rightarrow G_L = 0.50$
- Right child: 150 fraud / 900 total $\Rightarrow G_R = 0.28$
- Weighted: $\frac{100}{1000}(0.50) + \frac{900}{1000}(0.28) = 0.30$
- Gain: $0.32 - 0.30 = 0.02$

Properties:

- $G = 0$: pure node (one class only)
- $G = 0.5$: maximum impurity (binary)
- Computationally cheaper than entropy

Insight

Gini impurity is a *concave* function of class proportions – every split on a mixed node reduces impurity.

CART uses Gini by default; scikit-learn's DecisionTreeClassifier(criterion='gini').

What Does Entropy Add Beyond Gini?

Shannon Entropy:

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

Information Gain:

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

- Entropy is zero for pure nodes, maximal at uniform distribution
- IG selects the feature that reduces uncertainty the most
- C4.5 and ID3 algorithms use entropy; CART uses Gini
- **Regression trees:** use MSE reduction (variance reduction) as the splitting criterion instead of Gini/Entropy

Gini vs. Entropy Comparison:

Property	**Gini**	**Entropy**
Range (binary)	[0, 0.5]	[0, 1]
Computation	Faster	log needed
Sensitivity	Majority class	Rare classes
Algorithm	CART	C4.5, ID3

In practice, Gini and entropy produce **nearly identical** trees.

Insight

Entropy penalizes impure nodes slightly more than Gini – in fraud detection, this can surface rare-class splits earlier.

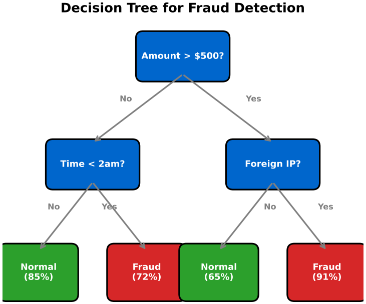
For binary classification, $H \approx 2G$ near $p = 0.5$; the choice rarely changes the final tree.

What Does a Trained Decision Tree Actually Look Like?

What you see: A tree trained on financial features, splitting recursively on the most informative thresholds.

Key pattern: Each internal node tests one feature; each leaf assigns a class or value. Deeper trees fit noise.

Takeaway: A single tree is interpretable but brittle – small data shifts change the entire structure.



Decision Tree for Fraud Detection

Amount > $500?

No

Yes

Time < 2am?

Foreign IP?

No

Yes

No

Yes

Normal (85%)

Fraud (72%)

Normal (65%)

Fraud (91%)

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

Insight

Interpretability is the single tree's greatest strength – and its fragility is the motivation for ensembles.

Pruning (max_depth, min_samples_leaf) trades bias for variance reduction.

Why Does Averaging Multiple Models Reduce Variance?

Key derivation. For B models with variance σ^2 and pairwise correlation ρ:

$$\text{Var}\left(\frac{1}{B}\sum_{b=1}^{B} X_b\right) = \rho\sigma^2 + \frac{1-\rho}{B}\,\sigma^2$$

Two extreme cases:

- $\rho = 1$ (identical models): Var $= \sigma^2$ – no improvement
- $\rho = 0$ (independent models): Var $= \sigma^2/B$ – perfect scaling

The **goal of bagging**: reduce ρ by training on different bootstrap samples while keeping σ^2 large (deep trees).

Derivation sketch:

- $\text{Var}(\bar{X}) = \frac{1}{B^2}\sum_{i,j}\text{Cov}(X_i, X_j)$
- Diagonal terms: $B \cdot \sigma^2$
- Off-diagonal: $B(B-1)\rho\sigma^2$
- Combine: $\frac{1}{B^2}[B\sigma^2 + B(B-1)\rho\sigma^2]$
- Simplify to the variance formula above

Practical implication:
With $B = 500$ and $\rho = 0.05$:
Var $\approx 0.05\sigma^2 + \frac{0.95}{500}\sigma^2 \approx 0.052\sigma^2$

Insight

The first term $\rho\sigma^2$ cannot be reduced by adding more trees – only *decorrelation* can shrink it.

This is why Random Forests add feature randomization on top of bagging.

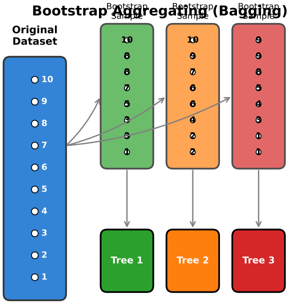
How Does Bootstrap Sampling Create Diversity?

What you see: Bootstrap resampling draws n samples with replacement from n observations.

Key pattern: Each bootstrap sample contains ∼63.2% unique observations; the remaining ∼36.8% are left out (OOB).

Takeaway: Bootstrap creates model diversity through data perturbation – each tree sees a different "view" of the training set.

- $P(\text{not selected}) = (1 - 1/n)^n \to 1/e \approx 0.368$
- Duplicates force trees to emphasize different patterns
- OOB samples become free validation data



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

Insight

Bootstrap is a "poor man's" way to approximate drawing from the true data-generating distribution.

Efron (1979) introduced bootstrap; Breiman (1996) applied it to tree ensembles.

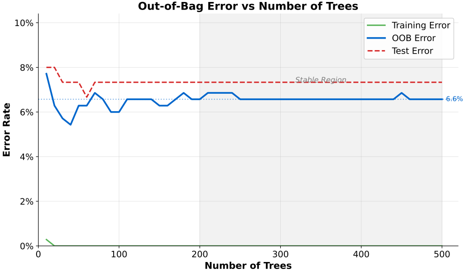
What Happens to the 36.8% of Samples Left Out?

What you see: OOB error converges as the number of trees grows, providing a built-in cross-validation estimate.

Key pattern: Each observation is OOB for ∼36.8% of trees. Aggregate their predictions to get an unbiased error estimate.

Takeaway: OOB error eliminates the need for a separate validation set – especially valuable when data is scarce.

- OOB error ≈ leave-one-out CV error
- No data “wasted” on validation
- Monitored automatically in scikit-learn with `oob_score=True`



https://github.com/Digital-AI-Finance/methods-algorithms/tree/master/slides/L04_Random_Forests/04_oob_error

Insight

OOB error is slightly pessimistic (each tree trained on ∼63% of data), making it a conservative estimate.

OOB error is one of Random Forests’ most elegant properties – free, unbiased validation.

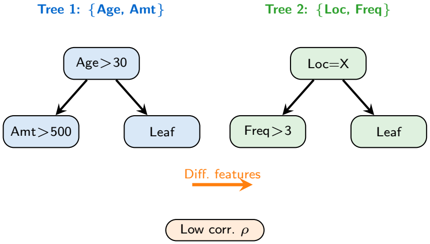
What Is the One Trick That Makes Random Forests Better Than Bagging?

Feature randomization. At each split, consider only a random subset of features:

- Classification: $m = \lfloor \sqrt{p} \rfloor$
- Regression: $m = \lfloor p/3 \rfloor$
- This forces trees to use **different features**, reducing correlation ρ

Why it works:

- Without randomization, every tree splits on the same dominant feature first
- Correlated trees ⇒ correlated errors ⇒ averaging helps less
- Feature subsampling **decorrelates** the trees



Insight

The “random” in Random Forests refers to feature subsampling at each split – not random data (that is bagging).

Breiman (2001): “Random Forests.” Machine Learning, 45(1), 5–32.

How Does Feature Randomization Reduce Tree Correlation?

Recall the ensemble variance formula:

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

Effect of feature randomization on ρ:

- Bagging alone: $\rho \approx 0.5$–0.9 (trees still correlated)
- RF with $m = \sqrt{p}$: $\rho \approx 0.05$–0.2
- Smaller $m \Rightarrow$ lower ρ, but higher per-tree variance σ^2

The sweet spot:

- As $\rho \to 0$: Var $\to \sigma^2/B$ (ideal)
- Diminishing returns beyond $B \approx 500$ trees
- The $\rho\sigma^2$ floor is the irreducible ensemble variance

Method	ρ	B	**Var**
Single tree	1.0	1	σ^2
Bagging	0.6	500	$0.601\sigma^2$
RF ($\sqrt{p}$)	0.1	500	$0.102\sigma^2$
RF ($\sqrt{p}$)	0.1	1000	$0.101\sigma^2$

Note: going from 500 to 1000 trees barely helps once ρ dominates. The real gain comes from **reducing** ρ.

Insight

Decorrelation (reducing ρ) matters far more than adding trees (increasing B) – this is RF's core innovation.

In practice, tune max_features before n_estimators.

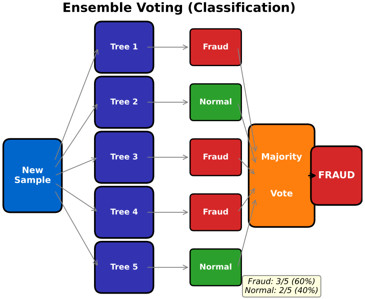
How Do 500 Trees Vote on a Single Prediction?

What you see: Individual tree predictions and the ensemble's aggregated vote for a single observation.

Key pattern: Individual trees disagree, but the **majority vote** converges to the correct class as B grows.

Takeaway:

- Classification: majority vote $\hat{y} = \text{mode}(\hat{y}_1, \dots, \hat{y}_B)$
- Regression: average $\hat{y} = \frac{1}{B} \sum_{b=1}^{B} \hat{y}_b$
- Probability: fraction voting for each class
- Confidence grows with ensemble size and tree diversity



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

Insight

The Condorcet Jury Theorem guarantees: if each tree is better than random and errors are independent, the ensemble converges to perfect accuracy.

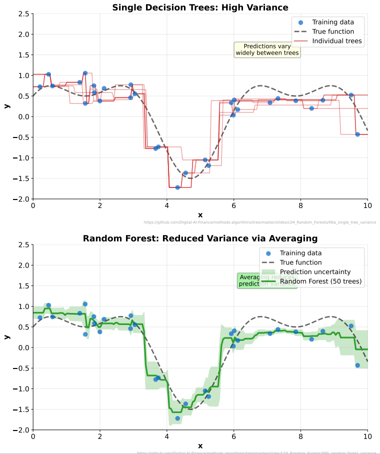
predict_proba() returns vote fractions – use these for ROC/AUC analysis.

Can You See the Variance Drop from One Tree to a Forest?

What you see: A single tree's jagged boundary (top) vs. the forest's smooth boundary (bottom).

Key pattern: The single tree overfits local noise; the forest averages it out.

Takeaway: Variance reduction made visible – averaging decorrelated trees smooths the prediction surface.



https://github.com/Digital-AI-Finance/methods-algorithms/tree/master/slides/L04_Random_Forests/06a_single_tree_variance

https://github.com/Digital-AI-Finance/methods-algorithms/tree/master/slides/L04_Random_Forests/06b_random_forest_variance

Insight

The forest does not learn a "better" tree – it averages *many different* trees to cancel out individual errors.

Top: single tree (high variance). Bottom: 500-tree forest (low variance, similar bias).

What Is the Random Forest Algorithm in Pseudocode?

```
Require: Training data $\{(\mathbf{x}_i, y_i)\}_{i=1}^n$, number of trees $B$, feature subset size $m$
for $b = 1$ to $B$ do
    Draw bootstrap sample $\mathcal{D}_b$ of size $n$
    Grow tree $T_b$ on $\mathcal{D}_b$:
        At each node, select $m$ features at random
        Find best split among these $m$ features
        Split and recurse until stopping criterion
    Store $T_b$
end for
Ensure: Predict new $\mathbf{x}$:
Classification: $\hat{y}$ = majority vote of $\{T_b(\mathbf{x})\}_{b=1}^B$
Regression: $\hat{y} = \frac{1}{B} \sum_{b=1}^B T_b(\mathbf{x})$
```

Complexity:

- Training: $O(B \cdot n \log n \cdot m)$
- Prediction: $O(B \cdot \text{depth})$
- Embarrassingly parallel – each tree independent

Default hyperparameters:

- $B = 500$ (n_estimators)
- $m = \lfloor \sqrt{p} \rfloor$ (max_features)
- No max_depth (fully grown)
- min_samples_leaf = 1

Insight

The algorithm is trivially parallelizable – training time scales linearly with cores, making RF fast even on large datasets.

scikit-learn: RandomForestClassifier(n_jobs=-1) uses all available CPU cores.

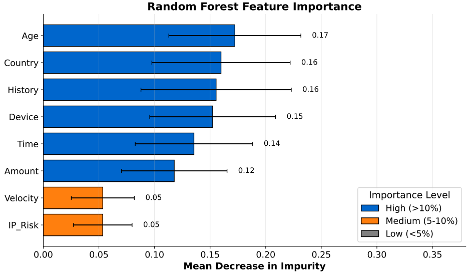
Which Features Matter Most? (Mean Decrease in Impurity)

What you see: Feature importance ranked by Mean Decrease in Impurity (MDI) across all trees.

Key pattern: MDI sums the Gini reduction from each feature across all splits in all trees:

$$\mathrm{MDI}_j = \frac{1}{B} \sum_{b=1}^{B} \sum_{t \in T_b} \Delta G_{t,j}$$

Takeaway: MDI is fast and built-in (feature_importances_ in scikit-learn), but it has known biases.



https://github.com/Digital-AI-Finance/methods-algorithms/tree/master/slides/L04_Random_Forests/02_feature_importance

Insight

MDI favors features used in many splits – not necessarily the features most predictive of the outcome.

Always cross-check MDI with permutation importance or SHAP for reliable conclusions.

Why Can MDI Be Misleading, and What Is Permutation Importance?

MDI bias:

- Favors **high-cardinality** features (many unique values $\Rightarrow$ more split opportunities)
- Inflated for correlated features (importance is "shared")
- Computed on training data – can reflect overfitting

Permutation importance (model-agnostic):

1. Compute baseline score on held-out data
2. Shuffle feature j's values randomly
3. Re-score; importance = drop in performance
4. Repeat for all features

	MDI	**Permutation**
Data used	Train	Test/OOB
Bias	High-card.	None
Cost	Free	$O(p \cdot B)$
Model	RF only	Any model
Correlations	Inflated	Shared

scikit-learn: `permutation_importance(model, X_test, y_test)`

Insight

Permutation importance measures what the model *actually uses*; MDI measures what the model *could use*.

Strobl et al. (2007) first demonstrated MDI bias; always prefer permutation importance for final reporting.

How Do SHAP Values Explain Individual Predictions?

Shapley value from cooperative game theory:

$$\phi_j = \sum_{S \subseteq N \setminus \{j\}} \frac{|S|!\,(p - |S| - 1)!}{p!} \left[f(S \cup \{j\}) - f(S) \right]$$

- ϕ_j = marginal contribution of feature j, averaged over all coalitions S
- Satisfies efficiency, symmetry, linearity, and null-player axioms
- $\sum_j \phi_j = f(x) - \mathbb{E}[f(X)]$ (predictions decompose exactly)

TreeSHAP (Lundberg et al., 2020):

- Exact Shapley values in $O(TLD^2)$ – polynomial, not exponential
- Exploits tree structure to enumerate coalitions efficiently
- Local (per-prediction) and global (averaged) explanations

Example: Transaction flagged as fraud
$\phi_{\text{amount}} = +0.32$
$\phi_{\text{location}} = +0.18$
$\phi_{\text{time}} = -0.05$
$\Rightarrow$ Amount and location drove the flag.

Insight

SHAP is the only feature attribution method with a *unique* solution satisfying all fairness axioms from game theory.

Lundberg & Lee (2017): "A Unified Approach to Interpreting Model Predictions." NeurIPS.

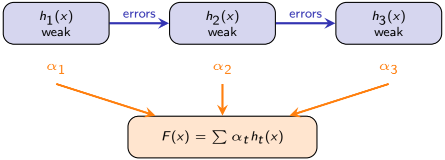
How Does Boosting Learn from Its Mistakes?

Core idea: Train weak learners **sequentially**, each focusing on the errors of the previous ensemble.

- **Bagging** (RF): parallel, reduces variance, full trees
- **Boosting**: sequential, reduces bias, shallow trees (stumps)
- Each new learner "corrects" the residual errors
- Final prediction: weighted sum of all weak learners

Three major algorithms:

- **AdaBoost** – reweights misclassified samples
- **Gradient Boosting** – fits residuals of a loss function
- **XGBoost** – regularized gradient boosting



Sequential: each learner corrects the combined errors of all predecessors.

Insight

Boosting reduces *bias* (makes weak learners strong); bagging reduces *variance* (stabilizes strong learners).

Schapire (1990) proved that weak learners can be "boosted" to arbitrary accuracy.

What Is AdaBoost's Weight Update Rule?

Learner weight (how much to trust learner t):

$$\alpha_t = \frac{1}{2} \ln \frac{1 - \epsilon_t}{\epsilon_t}$$

where $\epsilon_t =$ weighted error rate of h_t.

Sample weight update (Freund & Schapire notation):

$$w_i \leftarrow w_i \cdot \exp(-\alpha_t \, y_i \, h_t(x_i))$$

where $y_i \in \{-1, +1\}$ and $h_t(x_i)$ is the weak learner prediction.

- When $y_i \neq h_t(x_i)$, the exponent is positive, increasing the weight
- Correctly classified: weight **decreased**; misclassified: weight **increased**
- Next learner focuses on hard examples
- Normalize weights to sum to 1

Properties of α_t:

- $\epsilon_t = 0$ (perfect): $\alpha_t \to \infty$
- $\epsilon_t = 0.5$ (random): $\alpha_t = 0$ (ignored)
- $\epsilon_t > 0.5$: $\alpha_t < 0$ (inverted)

Final prediction:

$$H(x) = \text{sign}\left(\sum_{t=1}^{T} \alpha_t \, h_t(x)\right)$$

AdaBoost is equivalent to forward stagewise additive modeling with exponential loss.

Insight

AdaBoost's exponential loss makes it sensitive to outliers and label noise – a key limitation in noisy financial data.

Freund & Schapire (1997): "A Decision-Theoretic Generalization of On-Line Learning." JCSS.

How Does Gradient Boosting Minimize an Arbitrary Loss?

Additive update:

$$f_m(x) = f_{m-1}(x) + \eta \cdot h_m(x)$$

where h_m fits the **negative gradient** of the loss:

$$r_{im} = -\frac{\partial L(y_i, f(x_i))}{\partial f(x_i)}\bigg|_{f=f_{m-1}}$$

Squared loss: $r_{im} = y_i - f_{m-1}(x_i)$ (residuals)
Log loss: $r_{im} = y_i - \sigma(f_{m-1}(x_i))$
Learning rate $\eta \in (0, 1]$ shrinks each tree's contribution for regularization.

Algorithm:

1. Initialize $f_0(x) = \arg\min_\gamma \sum L(y_i, \gamma)$
2. For $m = 1, \ldots, M$:
   - Compute pseudo-residuals r_{im}
   - Fit tree h_m to $\{(x_i, r_{im})\}$
   - Update: $f_m = f_{m-1} + \eta \cdot h_m$
3. Output $f_M(x)$

Key hyperparameters:

- M: number of boosting rounds
- η: learning rate (0.01–0.3)
- Tree depth: usually 3–8

Insight

Gradient boosting performs *gradient descent in function space* – each tree is a step toward the loss minimum.

Friedman (2001): "Greedy Function Approximation: A Gradient Boosting Machine." Annals of Statistics.

What Makes XGBoost's Objective Function Special?

Regularized objective:

$$\mathcal{L} = \sum_{i=1}^{n} L(y_i, \hat{y}_i) + \sum_{k=1}^{K} \Omega(f_k)$$

where the regularization term is:

$$\Omega(f) = \gamma T + \tfrac{1}{2}\lambda \|w\|^2$$

- T = number of leaves, w = leaf weights
- γ penalizes tree complexity (pruning)
- λ penalizes large leaf values (L2)

Second-order Taylor expansion:

$$\mathcal{L}^{(t)} \approx \sum_i [g_i f_t(x_i) + \tfrac{1}{2} h_i f_t^2(x_i)] + \Omega(f_t)$$

where $g_i = \partial_{\hat{y}} L$, $h_i = \partial_{\hat{y}}^2 L$.

Optimal leaf weight:

$$w_j^* = -\frac{\sum_{i \in I_j} g_i}{\sum_{i \in I_j} h_i + \lambda}$$

Using second-order info gives faster convergence than first-order gradient boosting.

Insight

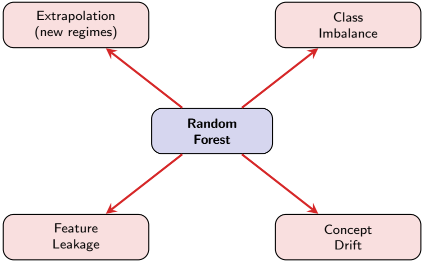
XGBoost's regularization prevents overfitting *structurally* (fewer leaves) and *numerically* (smaller weights) – crucial for noisy financial data.

Chen & Guestrin (2016): "XGBoost: A Scalable Tree Boosting System." KDD.

When Does a Forest Fail Silently?

Four failure modes that produce **no error message** but wrong predictions:

- **Extrapolation:** RF cannot predict outside training range (piecewise constant). New market regimes break the model.
- **Class imbalance:** 99.9% legitimate ⇒ "always predict legit" achieves 99.9% accuracy but catches zero fraud.
- **Feature leakage:** Future information in training features (e.g., post-transaction flags) inflates apparent accuracy.
- **Concept drift:** Fraud patterns evolve; a model trained in 2023 degrades by 2024.



All four produce high accuracy on historical data but fail in production.

Insight

Always evaluate on *temporally out-of-sample* data and monitor model performance continuously in production.

Silent failures are worse than crashes – they erode trust slowly and cause regulatory risk.

How Do You Choose the Right Hyperparameters?

Param	**Default**	**Guideline**
n_estimators	100	500+; more is rarely worse
max_depth	None	Limit to 10–20 for speed
min_samples_split	2	Increase to reduce overfit
max_features	$\sqrt{p}$	Try $\log_2 p$ for large p
class_weight	None	'balanced' for imbalance

- **Tuning order:** max_features → max_depth → n_estimators
- Use OOB score for quick evaluation
- Use cross-validation for final selection

Overfitting signals:

- Training accuracy $\gg$ test accuracy
- OOB error diverges from training error
- Feature importance dominated by noise features

Underfitting signals:

- Both training and test accuracy low
- Increase max_depth or n_estimators
- Check if max_features is too small

Rule of thumb: Start with defaults, increase n_estimators to 500, then tune max_features via grid search.

Insight

Random Forests are remarkably robust to hyperparameters – defaults often perform within 1–2% of the optimum.

Probst et al. (2019): "Hyperparameters and Tuning Strategies for Random Forest." WIREs Data Mining.

How Do Fraud Teams Use Feature Importance for Transaction Monitoring?

Worked case study:

Features: transaction amount, merchant category, time of day, customer age, distance from home, transaction frequency (last 24h), card-present flag.

Model: RF with 500 trees, class_weight='balanced', trained on 6 months of labeled data.

Top-5 by permutation importance:

1. Distance from home (ΔAUC $= 0.08$)
2. Transaction amount (ΔAUC $= 0.06$)
3. Frequency last 24h (ΔAUC $= 0.04$)
4. Time of day (ΔAUC $= 0.03$)
5. Card-present flag (ΔAUC $= 0.02$)

Actionable insights:

- **Distance:** Transactions far from home are the strongest fraud signal
- **Amount:** Large transactions need extra scrutiny, but amount alone is insufficient
- **Frequency:** Rapid-fire transactions suggest card testing
- **Time:** Late-night transactions correlate with fraud
- **Card-present:** CNP (card-not-present) transactions are riskier

These findings are presented to fraud analysts as SHAP waterfall plots for each flagged transaction.

Insight

Feature importance tells analysts *where to look*; SHAP tells them *why this specific transaction* was flagged.

Fraud teams combine model outputs with rule-based systems for final decisions.

What Happens When 99.9% of Transactions Are Legitimate?

The class imbalance problem:

- Fraud rate ≈ 0.1% ⇒ 1 fraud per 1000 transactions
- Classifier predicting "always legit" gets 99.9% accuracy
- **Accuracy is meaningless** for imbalanced data

Solutions:

- `class_weight='balanced'`: upweight minority class inversely proportional to frequency
- **Stratified bootstrap:** ensure each tree's sample has proportional fraud cases
- **SMOTE:** synthetic minority oversampling (creates new fraud examples by interpolation)
- **Cost-sensitive learning:** $C_{\mathrm{FN}} \gg C_{\mathrm{FP}}$ (missing fraud costs more than false alarms)

Precision-recall tradeoff:

- **Precision:** Of flagged transactions, how many are actually fraud?
- **Recall:** Of all fraud, how many did we catch?
- Lowering threshold ⇒ higher recall, lower precision
- Banks target **high recall** (catch fraud) at acceptable precision

Key metrics for imbalanced data:

- PR-AUC (precision-recall curve area)
- F1 or F_β score (weighted harmonic mean)
- Cost-weighted accuracy

Insight

In fraud detection, a false negative (missed fraud) costs 10–100x more than a false positive (unnecessary alert).

Use PR-AUC, not ROC-AUC, when the positive class is rare (Davis & Goadrich, 2006).

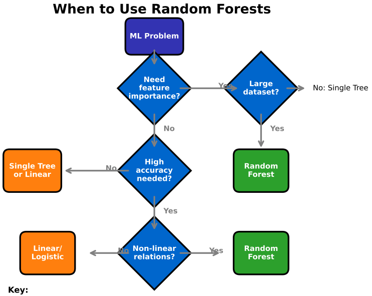
When Should You Choose RF Over Logistic Regression?

What you see: A decision flowchart guiding model selection based on data characteristics and requirements.

Key pattern:

- **Interpretability required?** ⇒ Logistic regression or single tree
- **Nonlinear relationships?** ⇒ RF or boosting
- **Many features, few samples?** ⇒ RF (implicit feature selection)
- **Maximum accuracy needed?** ⇒ XGBoost with tuning

Takeaway: No single model dominates; the right choice depends on the problem, data, and regulatory constraints.



Key:

Random Forest: Best for accuracy + feature importance

Alternative: When interpretability is paramount

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

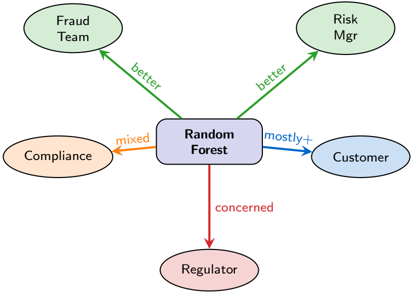
Insight

In regulated industries, the “best” model is often the one that is *explainable enough* to satisfy compliance – not the most accurate.

Many banks use RF internally but report logistic regression coefficients to regulators.

Who Wins and Who Loses When Ensembles Replace Scorecards?

- **Fraud Team:** Wins – better detection rates, fewer missed cases, faster adaptation
- **Risk Manager:** Wins – lower losses, better capital allocation, richer risk models
- **Compliance:** Mixed – improved outcomes but harder to audit 500-tree models
- **Customer:** Mostly wins – fewer false declines, but less transparent decisions
- **Regulator:** Concerned – demands explainability, adverse action reasons, fairness audits



Insight

The transition from scorecards to ensembles is a *governance* challenge as much as a technical one.

Successful adoption requires buy-in from all five stakeholders, not just data scientists.

Can Regulators Trust a Model Made of 500 Trees?

Regulatory landscape:

- **ECOA / Reg B (US):** Lenders must provide specific "adverse action reasons" for denial – hard with a forest
- **GDPR Article 22 (EU):** Right to "meaningful information about the logic involved" in automated decisions
- **EU AI Act:** High-risk AI systems (credit scoring) require transparency and human oversight

Bridging the gap with SHAP:

- SHAP provides per-decision explanations
- Top-k SHAP features $\Rightarrow$ adverse action reasons
- Global SHAP summary $\Rightarrow$ model documentation

Practical compliance workflow:

1. Train RF for optimal performance
2. Compute SHAP values for every prediction
3. Map top SHAP features to human-readable reason codes
4. Document model in a "model risk management" framework
5. Monitor for fairness across protected classes

SR 11-7 (Fed) requires banks to validate and document all models used in decision-making.

Insight

SHAP turns a "black box" into a "glass box" – the model is complex, but each decision is explainable.

The EU AI Act (2024) classifies credit scoring as "high-risk" – requiring explainability by law.

Key Takeaways

Mathematical Foundation

- Gini impurity and entropy guide tree splits; both yield similar results
- Variance of ensemble: $\rho\sigma^2 + \frac{(1-\rho)}{B}\sigma^2$ – decorrelation is key
- OOB error provides free, unbiased validation

Ensemble Methods Compared

- **Bagging/RF:** reduces variance via parallel, decorrelated trees
- **AdaBoost:** reduces bias via sequential reweighting
- **Gradient Boosting:** gradient descent in function space
- **XGBoost:** adds regularization and second-order optimization

Practical Application

- Feature importance: MDI → permutation → SHAP (increasing reliability)
- Fraud detection demands high recall, cost-sensitive learning, and PR-AUC evaluation
- Class imbalance: class_weight, SMOTE, stratified sampling
- SHAP bridges the gap between model complexity and regulatory explainability

Random Forests remain one of the strongest "out-of-the-box" classifiers two decades after Breiman (2001).

Until Next Time. . .

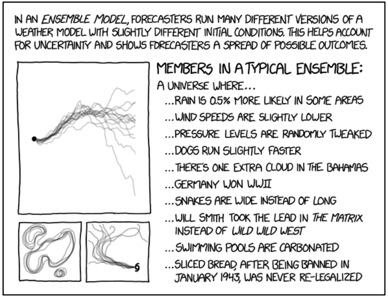
We opened with the absurdity of ensemble models (XKCD #1885).

Now you know **WHY** 500 trees beat one: **variance reduction through decorrelation**.

- One tree is interpretable but fragile
- Bagging stabilizes via bootstrap averaging
- Feature randomization decorrelates trees ($\rho \downarrow$)
- Boosting attacks bias where RF attacks variance
- SHAP makes the forest explainable for regulators

Next: L05 – PCA & t-SNE

From ensemble predictions to **dimensionality reduction** – how to visualize and compress high-dimensional financial data.


IN AN *ENSEMBLE MODEL*, FORECASTERS RUN MANY DIFFERENT VERSIONS OF A WEATHER MODEL WITH SLIGHTLY DIFFERENT INITIAL CONDITIONS. THIS HELPS ACCOUNT FOR UNCERTAINTY AND SHOWS FORECASTERS A SPREAD OF POSSIBLE OUTCOMES.
MEMBERS IN A TYPICAL ENSEMBLE:
A UNIVERSE WHERE...
...RAIN IS 0.5% MORE LIKELY IN SOME AREAS
...WIND SPEEDS ARE SLIGHTLY LOWER
...PRESSURE LEVELS ARE RANDOMLY TWEAKED
...DOGS RUN SLIGHTLY FASTER
...THERE'S ONE EXTRA CLOUD IN THE BAHAMAS
...GERMANY WON WWII
...SNAKES ARE WIDE INSTEAD OF LONG
...WILL SMITH TOOK THE LEAD IN *THE MATRIX* INSTEAD OF *WILD WILD WEST*
...SWIMMING POOLS ARE CARBONATED
...SLICED BREAD, AFTER BEING BANNED IN JANUARY 1943, WAS NEVER RE-LEGALIZED

"The real power of ensemble models is not that each tree is good – it is that each tree is wrong in a different way."

L05 preview: PCA finds directions of maximum variance; t-SNE preserves local structure for visualization.