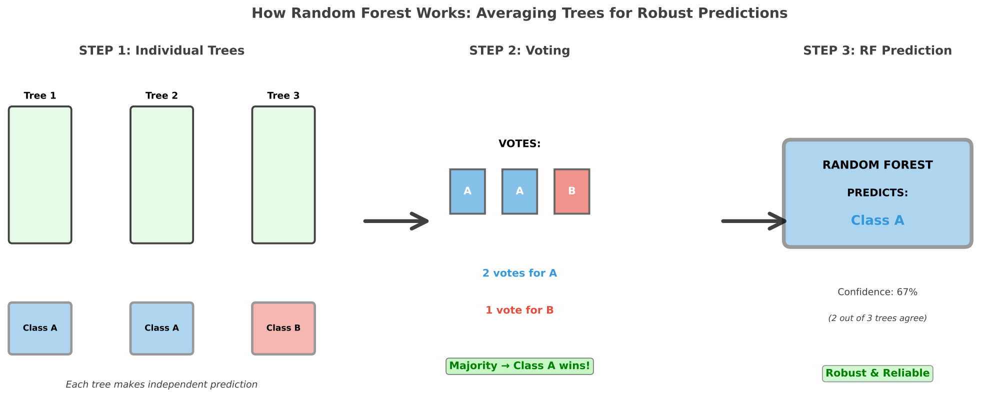
Introduction to Random Forest

Ensemble Learning for Robust Predictions

October 29, 2025

The Random Forest Concept



Core idea: Multiple trees vote, majority wins - creates robust predictions through averaging

The Wisdom of Crowds

Key Insight
Many weak predictors can combine to form one strong predictor.

Example: Guess the Weight

- 100 people guess ox weight
- Individual guesses: Often far off
- Average of guesses: Remarkably accurate!

Why?

- Errors cancel out
- Overestimates + underestimates $\rightarrow$ balanced
- Collective intelligence

Machine Learning Analog

- Train many models
- Each makes mistakes
- Different mistakes (uncorrelated)
- Average predictions
- $\rightarrow$ Better than any single model

Random Forest
Apply wisdom of crowds to decision trees:

- Train 100 decision trees
- Each tree votes
- Majority wins
- Result: Robust predictions

Random Forest: Harness collective intelligence of many decision trees

Quick Recap: Decision Trees

What They Are
Hierarchical IF-THEN rules:

- Ask questions about features
- Split data recursively
- Create decision regions
- Majority vote in leaf

Strengths

- Interpretable
- Handle non-linear patterns
- No preprocessing needed

The Problem
Single trees suffer from:

- **High variance**: Small data change → completely different tree
- **Overfitting**: Grow until perfect fit on training data
- **Instability**: Unpredictable performance

Solution?
Don't use one tree. Use many!
→ Random Forest

Single decision trees are powerful but unstable: Random Forest fixes this

The Problem: Single Trees are Unstable

High Variance Problem
Train same algorithm on slightly different data:

- Different bootstrap sample
- Different random subset
- Different initialization

Result: **Completely different tree!**

Consequences

- Unpredictable performance
- Sensitive to data changes
- Unreliable predictions
- Hard to trust

Why This Happens
Decision trees:

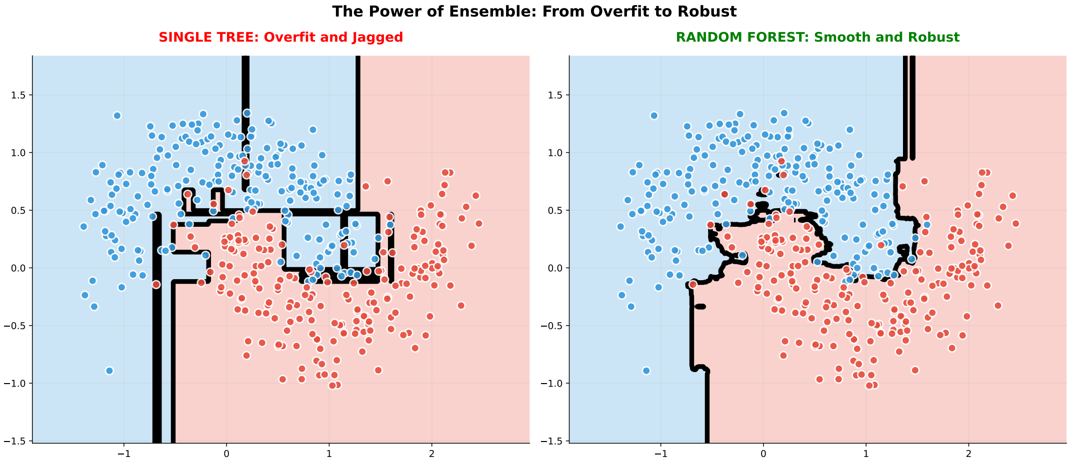
- Greedy algorithm (no look-ahead)
- Small data change $\rightarrow$ different first split
- Cascades down entire tree
- Creates completely different structure

Analogy
Like asking one person to guess:

- Individuals vary widely
- Each has their own biases
- One person $=$ unreliable

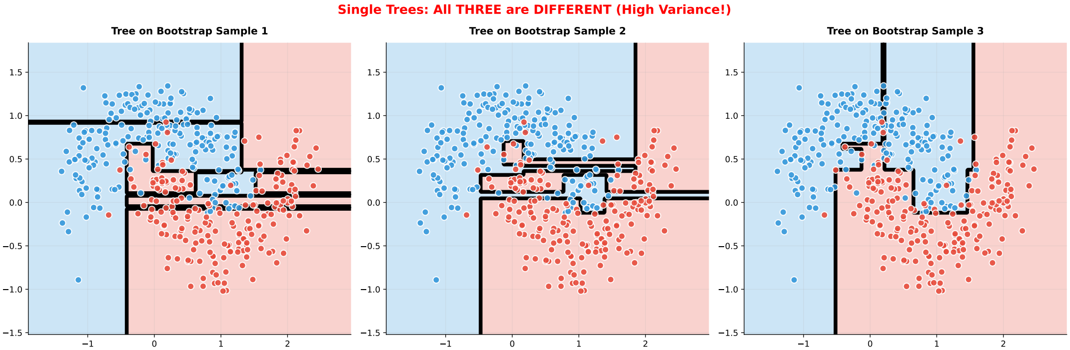
Solution: Ask many people!

Single decision tree: Powerful but unstable, high variance is the key problem



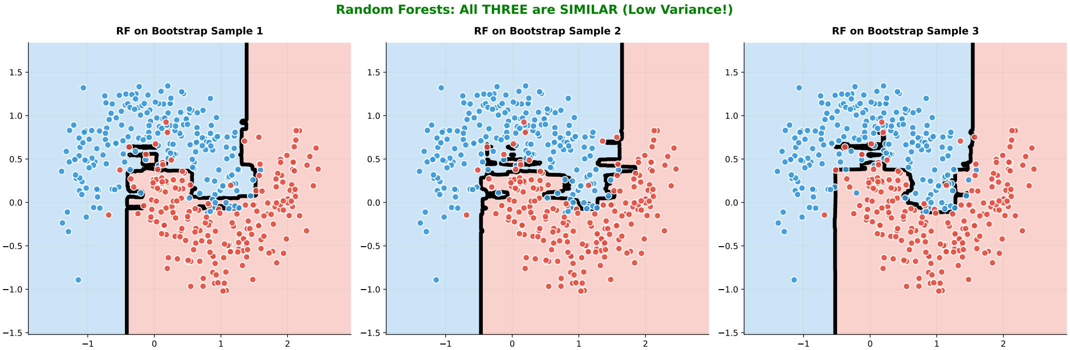
Left: Single tree creates jagged, overfit boundary — Right: Random Forest creates smooth, robust boundary

Single Trees: UNSTABLE (All Different)



3 different bootstrap samples: All three trees produce DRAMATICALLY DIFFERENT boundaries

Random Forests: STABLE (All Similar)



Same 3 bootstrap samples: All three RFs produce NEARLY IDENTICAL boundaries - Low variance!

The Random Forest Idea

Core Concept

Train many decision trees:

- Typically 100-500 trees
- Each tree different
- Combine predictions

Two Sources of Randomness

1. **Bagging**: Different data
2. **Feature randomness**: Different features

Both create diversity!

Prediction

Classification:

- Each tree votes for class
- Majority wins

Regression:

- Each tree predicts value
- Average all predictions

Result

- Smoother boundaries
- Lower variance
- More robust
- Better generalization

Random Forest = Ensemble of diverse decision trees

Mechanism 1: Bagging (Bootstrap Aggregating)

How It Works

For each tree:

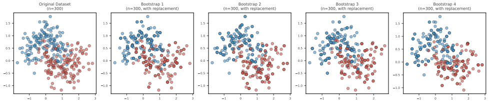
1. Sample n points with replacement
2. Train tree on this bootstrap sample
3. Save tree

Repeat 100-500 times.

Bootstrap Sampling

- Same size as original data
- Some points repeated
- Some points left out ("out-of-bag")
- Different sample → different tree

Why It Reduces Variance



Different data → Diverse trees → Uncorrelated errors → Errors cancel out when averaged

Averaging reduces variance!

Bagging: Train on different bootstrap samples to create diversity

Mechanism 2: Feature Randomness

The Problem
With just bagging:

- Trees still too similar
- Strong features dominate
- Trees correlate
- Limited variance reduction

The Solution
At each split:

- Don't consider all features
- Randomly select subset
- Choose best split from subset only
- Typical: $\sqrt{p}$ features

Why It Helps

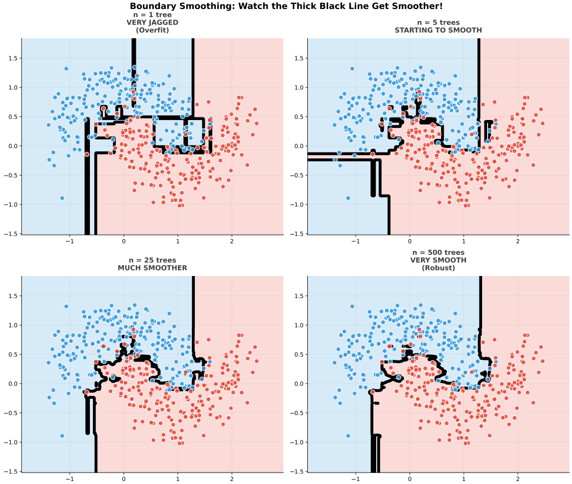
- Weak features get chance to split
- Trees use different features
- Trees become decorrelated
- Further variance reduction

Example
Dataset with 16 features:

- Without randomness: All trees split on Feature 1 first
- With randomness (4 random features): Trees split on Features 1, 5, 7, 12...
- Result: Diverse trees!

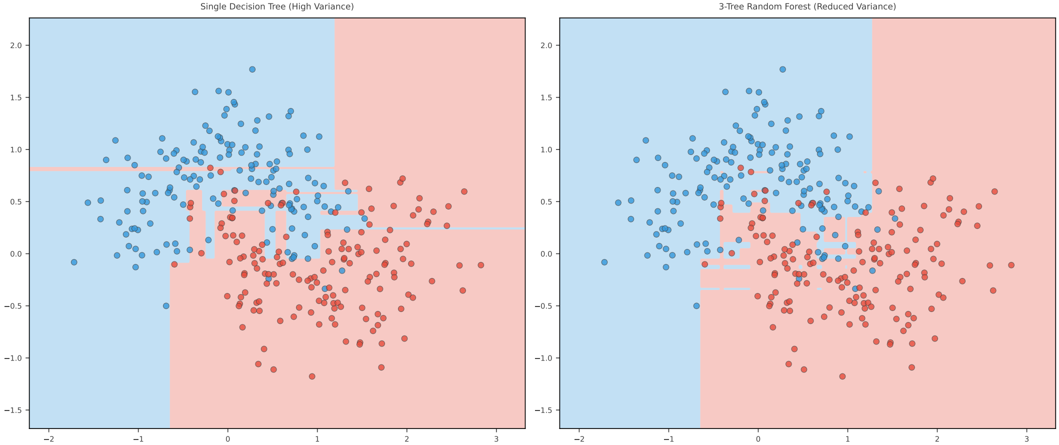
Feature randomness decorrelates trees beyond what bagging achieves

Boundary Smoothing Progression



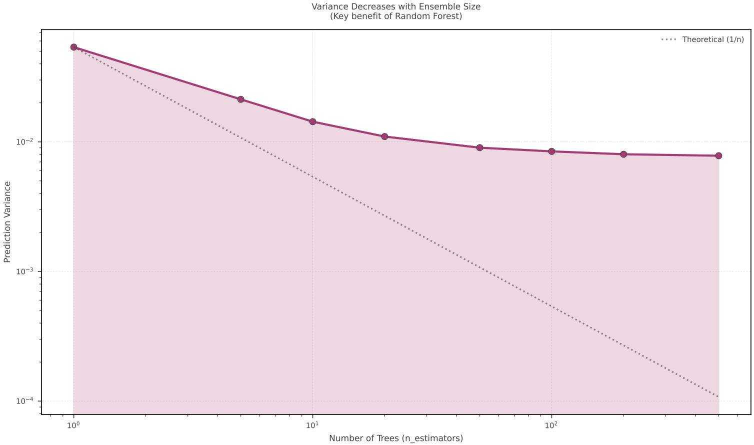
n=1 (very jagged) → n=5 (starting to smooth) → n=25 (much smoother) → n=500 (very smooth) — Watch thick black boundary line!

Single Tree vs Ensemble



Single tree creates complex regions, ensemble smooths to robust boundary

Variance Reduction in Action



Prediction variance decreases as more trees added, plateaus around 100-200 trees

How Trees Combine: Voting and Averaging

Classification: Majority Vote

- Tree 1: Class A
- Tree 2: Class B
- Tree 3: Class A
- Tree 4: Class A
- ...
- Tree 100: Class A

Result: 65 votes A, 35 votes B
$\rightarrow$ Predict Class A

Regression: Average

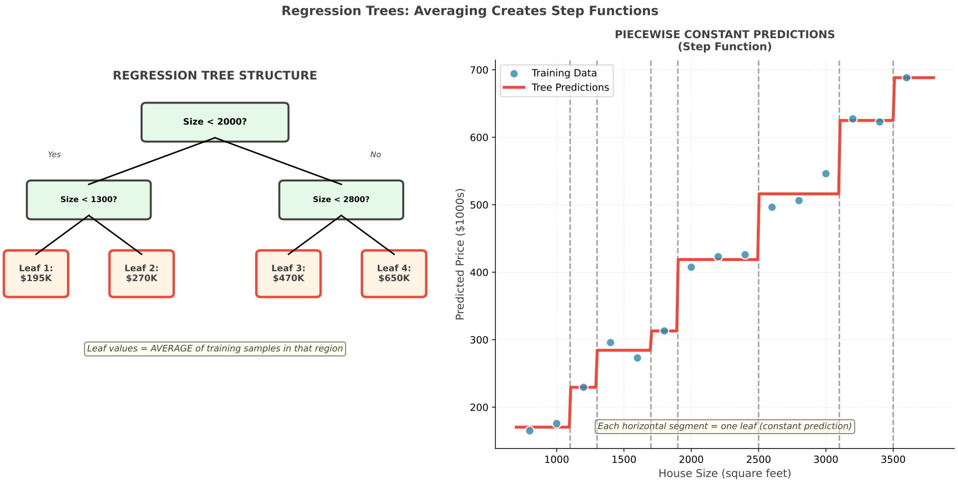
- Tree 1: 5.2
- Tree 2: 4.8
- Tree 3: 5.5
- Tree 4: 4.9
- ...
- Tree 100: 5.1

Result: Average $=$ 5.0
$\rightarrow$ Predict 5.0

Aggregation smooths out individual tree errors and creates robust predictions.

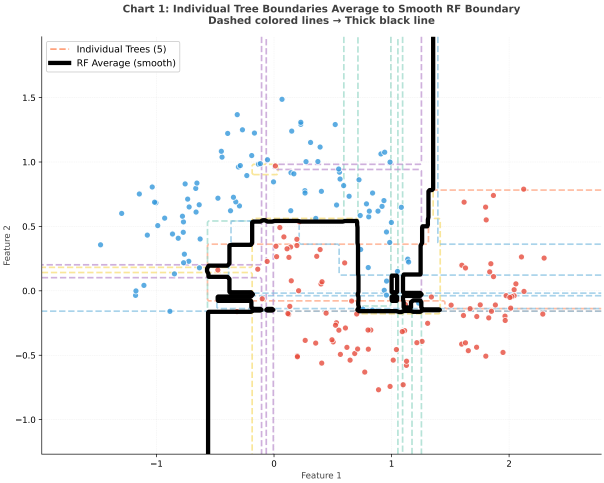
Simple aggregation (vote/average) is key to Random Forest power

Regression Trees: Averaging for Continuous Targets



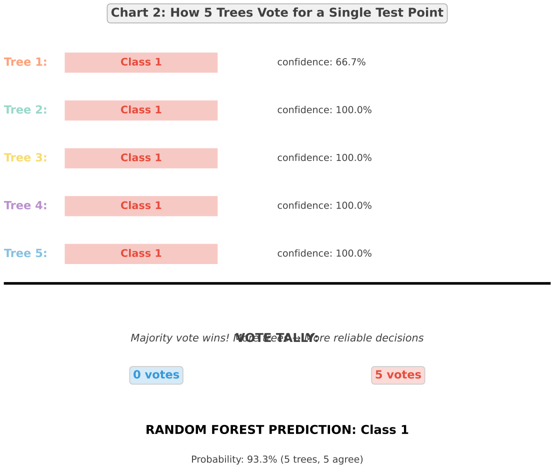
Regression trees average values in leaves, creating step functions — Random Forest averages these step functions for smoother predictions

How Averaging Works: Boundary Overlay



Individual tree boundaries (dashed, colored) average to smooth RF boundary (thick black)

How Averaging Works: Voting Mechanism



Step-by-step: How 5 trees vote for a single test point, majority wins

When to Use Random Forest

Advantages

- **Robust**: Low variance, stable
- **Accurate**: Often best performance
- **No overfitting**: Self-regulating
- **Handles complexity**: Non-linear patterns
- **Feature importance**: Built-in
- **Out-of-bag validation**: Free
- **Versatile**: Classification & regression

Best For

- Default choice for tabular data
- Complex non-linear relationships
- Need robust predictions
- Feature selection

Disadvantages

- **Less interpretable**: 100 trees hard to visualize
- **Slower**: Training many trees takes time
- **Memory**: Must store all trees
- **Prediction time**: Slower than single tree
- **Overkill**: For simple linear problems

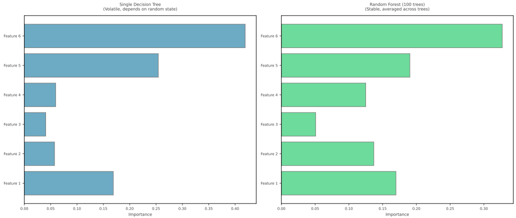
Not Ideal For

- Need explainability (use single tree)
- Real-time predictions (use linear model)
- Very large datasets (millions of rows)
- Simple linear relationships

Random Forest: Excellent default choice for most classification/regression tasks

Finding Innovations: Stable Feature Importance

RF Feature Importance is Reliable



Single tree importance: Volatile
RF importance: Stable and trustworthy

Innovation Discovery
Medical diagnosis example:

1. Blood Pressure: 32% importance
2. **Sleep Quality: 27% importance**
3. Age: 18% importance
4. Weight: 15% importance
5. Exercise: 8% importance

Innovation Insight
"Sleep quality was never in our screening protocol but RF shows it's the #2 predictor. Let's add sleep assessment to catch more cases early."

RF reveals hidden drivers!

Use RF feature importance to discover which factors truly drive outcomes

Rapid Discovery with Out-of-Bag Error

OOB Enables Fast Experimentation

- No separate validation set needed
- Test many ideas quickly
- Iterate rapidly
- Find best approach faster

Innovation Workflow

1. Try Feature Set 1 $\rightarrow$ OOB: 85% acc
2. Try Feature Set 2 $\rightarrow$ OOB: 87% acc
3. Try Feature Set 3 $\rightarrow$ OOB: 91% acc!
4. Try Interaction A$\times$B $\rightarrow$ OOB: 93% acc!

Discover best approach in minutes!

Innovation Example

E-commerce conversion optimization:

- Started with standard features
- OOB showed poor performance
- Tried adding time-of-day features
- OOB improved!
- Tried user journey interactions
- **OOB jumped to 94%!**

Innovation Discovered

"User journey path + time interaction was key predictor we never considered. Changed entire UX strategy."

OOB enables rapid innovation cycles!

OOB error: Unique Random Forest advantage for fast discovery and experimentation

Real-World Applications

Kaggle Competitions

Random Forest frequently wins:

- Robust to overfitting
- Works well out-of-the-box
- Handles mixed data types

Finance

- Credit risk assessment
- Fraud detection
- Stock price movement
- Default prediction

Healthcare

- Disease diagnosis
- Patient outcome prediction
- Treatment recommendation
- Risk stratification

Other Domains

- Customer churn prediction
- Product recommendation
- Image classification (with features)
- Sensor data analysis

Random Forest is one of the most widely used ML algorithms in practice

Summary: Key Concepts

Core Idea
Wisdom of crowds: Many trees better than one

Two Mechanisms

1. **Bagging**: Bootstrap sampling creates data diversity
2. **Feature randomness**: Random feature subsets create split diversity

Aggregation
Vote (classification) or Average (regression)

Benefits

- Reduced variance
- Smooth boundaries
- No overfitting
- Robust predictions

Trade-offs

- Less interpretable
- Slower than single tree
- More memory

Random Forest is often the best starting point for supervised learning tasks.

Random Forest: Powerful ensemble method combining diversity and aggregation

Questions?

See appendix for mathematical details

Appendix A1: Variance Reduction Formula

Individual Tree Variance
Single tree has variance σ^2

Ensemble Variance (Uncorrelated)
If trees are independent:

$$\sigma^2_{\text{ensemble}} = \frac{\sigma^2}{n}$$

where n is number of trees.

Variance decreases as $1/n$!

Example: 100 trees $\rightarrow$ variance reduced by factor of 100

With Correlation
In reality, trees are correlated ($\rho > 0$):

$$\sigma^2_{\text{ensemble}} = \rho\sigma^2 + \frac{1-\rho}{n}\sigma^2$$

Two terms:

- $\rho\sigma^2$: Irreducible (correlation)
- $\frac{1-\rho}{n}\sigma^2$: Reducible (averaging)

Key Insight
Lower correlation $\rho \rightarrow$ Better variance reduction
This is why feature randomness matters!

Mathematics explains why diverse trees are crucial

Appendix A2: Bias-Variance Decomposition

Total Error

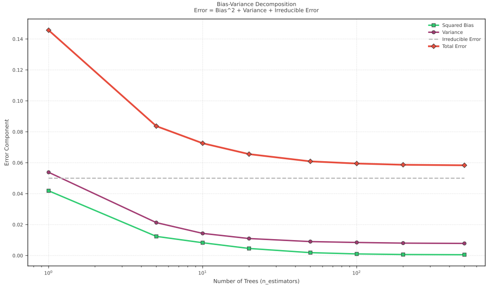
$$\text{Error} = \text{Bias}^2 + \text{Variance} + \text{Noise}$$

Random Forest Effect

- Bias: Stays low (trees are flexible)
- Variance: Decreases (averaging effect)
- Noise: Irreducible

Result: Lower total error!

Visualization



Variance drops while bias stays flat.

Random Forest reduces variance without increasing bias

Appendix A3: Out-of-Bag (OOB) Probability

Bootstrap Sampling
When sampling n points with replacement:
Each point has probability of being selected:

$$P(\text{selected}) = 1 - \left(1 - \frac{1}{n}\right)^n$$

For large n:

$$\lim_{n\to\infty} \left(1 - \frac{1}{n}\right)^n = \frac{1}{e}$$

Therefore:

$$P(\text{out-of-bag}) \approx 1 - \frac{1}{e} \approx 0.368$$

What This Means
Each tree leaves out 37% of data.

Free Validation
For each point:

- Use trees where it was OOB
- Make prediction
- Compare to true label
- Compute OOB error

OOB error ≈ Cross-validation error
No need for separate validation set!

OOB error: Unique Random Forest feature providing free validation

Appendix A4: Correlation Effect

Variance Reduction Depends on Correlation

$$\sigma^2_{\text{RF}} = \rho\sigma^2 + \frac{1-\rho}{n}\sigma^2$$

Two Extremes

If $\rho = 0$ (perfectly uncorrelated):

$$\sigma^2_{\text{RF}} = \frac{\sigma^2}{n}$$

If $\rho = 1$ (perfectly correlated):

$$\sigma^2_{\text{RF}} = \sigma^2$$

No reduction!

Why Feature Randomness Matters

Bagging alone: $\rho \approx 0.5$ (moderate correlation)
Bagging + Feature randomness: $\rho \approx 0.1$ (low correlation)

Example with 100 trees:
Just bagging:

$$\sigma^2 = 0.5\sigma^2 + 0.005\sigma^2 = 0.505\sigma^2$$

With feature randomness:

$$\sigma^2 = 0.1\sigma^2 + 0.009\sigma^2 = 0.109\sigma^2$$

Much better!

Low correlation between trees is crucial for variance reduction

Appendix A5: Feature Importance

How It's Computed

For each tree:

- Compute importance of each feature
- Based on impurity reduction

For Random Forest:

- Average importance across all trees
- Normalize to sum to 1

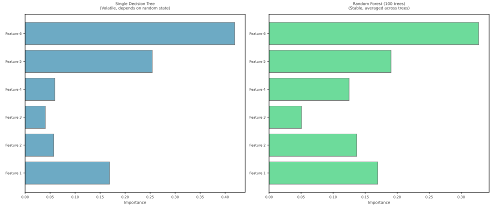
Formula

$$\text{Importance}_j = \frac{1}{T}\sum_{t=1}^{T}\sum_{k:\text{feature } j}\Delta I_k^t$$

where ΔI is impurity decrease at node k

Why It's Useful

- Feature selection
- Understanding model
- Domain insights
- More stable than single tree

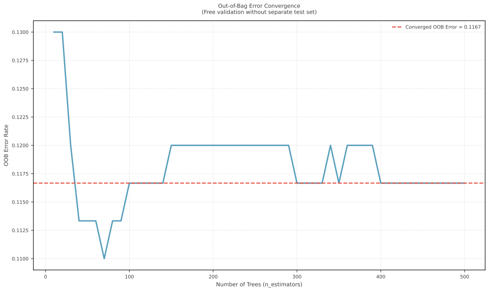


RF importance more stable and reliable.

Feature importance: Averaged across trees for stability

Appendix A6-A7: Out-of-Bag Error

OOB Error Concept



OOB error converges as more trees added.

How to Use

1. Train Random Forest
2. Compute OOB error automatically
3. Use OOB error to:
   - Assess model quality
   - Tune hyperparameters
   - Decide if more trees needed

No separate validation set required!

OOB vs Cross-Validation
Highly correlated, much faster.

OOB: Efficient alternative to cross-validation unique to Random Forest

Appendix A8-A9: Hyperparameter Tuning

Key Hyperparameters

`n_estimators` (number of trees)

- More is better (diminishing returns)
- Typical: 100-500
- Use OOB to decide

`max_features` (features per split)

- Classification: $\sqrt{p}$
- Regression: $p/3$
- Most important tuning parameter

`max_depth`

- Often: None (unlimited)
- Unlike single tree!

Tuning Strategy

1. Start with defaults
2. Tune `max_features` first (most impact)
3. Increase `n_estimators` until OOB stabilizes
4. Tune tree parameters if needed

Quick Recipe

For most problems:

- `n_estimators=100`
- `max_features='sqrt'`
- `max_depth=None`

Works well without tuning!

Random Forest often works well with default parameters

Appendix A10: How Feature Importance Works

In Decision Trees

Calculate for each feature:

1. Sum weighted impurity reductions
2. Normalize to 100%

$$\text{Importance}_j = \frac{\sum n_k \cdot \Delta I_k}{\text{Total}}$$

In Random Forest

1. Calculate importance in EACH tree
2. **Average across all trees**
3. Normalize final result

$$\text{RF Importance}_j = \frac{1}{T} \sum_{t=1}^{T} \text{Tree}_t \text{Importance}_j$$

Why RF is Better

Single tree importance:

- Changes with different data
- Unstable (high variance)
- Less reliable

Random Forest importance:

- **Averaged across 100-500 trees**
- Very stable
- Robust to data variations
- **Much more trustworthy**

Innovation Application

Use RF importance to:

- Discover hidden drivers
- Guide data collection
- Focus resources
- Find unexpected patterns

RF averaging makes feature importance reliable for discovery and innovation

Appendix A11: Further Learning

Key Concepts Covered

- Wisdom of crowds
- Bagging (bootstrap)
- Feature randomness
- Variance reduction
- Ensemble aggregation
- Out-of-bag error

Next Steps

- Implement with scikit-learn
- Try on real datasets
- Compare to single trees
- Explore feature importance
- Study gradient boosting (different ensemble method)

Resources

Papers:

- Breiman (2001): "Random Forests"
- Original RF paper

Books:

- "Introduction to Statistical Learning"
- "Elements of Statistical Learning"

Online:

- Scikit-learn Random Forest guide
- Kaggle tutorials

Random Forest: Foundation for modern ensemble methods