Comprehensive Analysis of Random Forests: Theory, Implementation, and Empirical Validation

Executive Summary

This document provides a comprehensive analysis of the Random Forest algorithm, synthesizing theoretical principles from Leo Breiman's foundational 2001 paper with empirical results from a from-scratch implementation and extensive experimentation. The study validates that the algorithm's exceptional performance stems from two key ingredients: **the strength of individual decision trees** and **the low correlation between them**.

Experiments conducted on both tabular (Heart Disease UCI) and high-dimensional image (Intel Image Classification) datasets confirm the algorithm's theoretical promises. Key findings indicate that Random Forests consistently and significantly outperform single decision trees, improving test accuracy by approximately **12 percentage points** while drastically reducing overfitting.

The analysis shows that model accuracy improves rapidly as the number of trees increases from 1 to 100, after which performance plateaus, confirming that more trees do not lead to overfitting. Training time scales linearly with the number of trees, making ensembles of 100–300 trees a practical and effective choice for most applications.

---

1. Theoretical Foundations of the Random Forest Algorithm

The Random Forest algorithm, as formalized by Leo Breiman, is an ensemble method that builds upon the synergistic relationship between **predictor strength** and **inter-predictor correlation** to minimize generalization error.

Core Principle: The Strength-Correlation Tradeoff

Breiman's central insight is that the generalization error of a Random Forest is bound by the strength of its individual tree classifiers and the average correlation between them. The mathematical relationship is expressed as:

$$PE^* \leq \bar{\rho} \frac{(1-s^2)}{s^2}$$

Where:

- PE^* is the generalization error.
- s is the strength of the individual trees (higher is better).
- $\bar{\rho}$ is the mean correlation between the trees (lower is better).

The algorithm's success lies in its ability to maintain strong individual predictors (low bias) while actively minimizing correlation, thereby reducing the variance of the ensemble's prediction.

Mechanisms for Achieving Low Correlation

Random Forests employ two primary randomization techniques to decorrelate the constituent decision trees:

1. **Bootstrap Aggregating (Bagging):**
   Each tree is trained on a bootstrap sample, a random sample of the training data drawn with replacement. Approximately **63.2% unique instances** from the original training set are included on average.

2. **Random Feature Selection:**
   At each node in a decision tree, only a random subset of features is considered for splitting. For classification, this subset size is typically:
   $$m = \sqrt{p}$$
   where p is the total number of features. This prevents a few dominant features from dictating all trees’ structures, promoting diverse predictive patterns.

Key Theoretical Properties

- **Convergence without Overfitting:**
  "Random Forests do not overfit as more trees are added." Averaging many independent (or decorrelated) trees reduces variance without increasing bias.

- **Out-of-Bag (OOB) Error Estimation:**
  The ~36.8% of training samples left out of each bootstrap sample serve as a built-in validation set, allowing unbiased generalization error estimation without a separate validation set.

- **Variable Importance Measures:**
  - **Mean Decrease in Impurity (MDI):** Total reduction in node impurity (e.g., Gini) from splits on a feature, averaged across all trees. Fast but biased toward high-cardinality features.
  - **Mean Decrease in Accuracy (MDA) / Permutation Importance:** Measures decrease in model accuracy on OOB samples when a feature’s values are permuted. More robust than MDI.

---

2. Project Overview and Experimental Design

The theoretical principles were validated through a project designed for a Machine Learning Lab Mid-term examination (Question 2), involving a **from-scratch implementation** and structured experiments.

Implementation from Scratch

- **DecisionTree Implementation:**
    - CART algorithm using Gini impurity and entropy.
    - Optimizations: vectorized split-finding, index-based recursion.
    - Supports hyperparameters: `max_depth`, `min_samples_split`, `max_features`.
- **RandomForest Implementation:**
    - Integrates DecisionTree base learners.
    - Implements bootstrap sampling, random feature selection, parallel tree building (`n_jobs`), and OOB error calculation.
    - Optimized with vectorized prediction for improved performance.

Datasets and Preprocessing

Dataset	Description	Task	Samples	Features	Preprocessing Steps
Heart Disease UCI	Tabular clinical data	Binary Classification	~300	13	Missing value removal, target encoding (0/1), StandardScaler
Intel Image Classification	High-dimensional image data	Multi-class (6 classes)	~1200	12,288	Resize to 64x64, flatten to 1D, normalize pixel values

Experimental Protocols

A systematic experimental design tested key hypotheses about algorithm behavior, using a fixed random seed (42) for reproducibility.

1. **Effect of Ensemble Size:**
   `n_estimators = [1, 10, 50, 100, 300]` to observe effects on training accuracy, test accuracy, and training time.
2. **Single Tree vs. Ensemble:**
   Compared a single Decision Tree against a Random Forest of 100 trees. Metrics collected: Accuracy, Precision, Recall, F1-score, training time, confusion matrices.

---

3. Empirical Results and Key Insights

Performance on Tabular Data (Heart Disease UCI)

Effect of Ensemble Size:

n_estimators	Train Accuracy	Test Accuracy	Training Time (s)
1	0.8542	0.7833	0.012
10	0.9375	0.8500	0.045
50	0.9792	0.8667	0.156
100	0.9875	0.8833	0.298
300	0.9958	0.8833	0.847

Single Decision Tree vs. Random Forest:

Model	Train Acc	Test Acc	Precision	Recall	F1	Time (s)
Decision Tree	1.0000	0.7667	0.7712	0.7667	0.7653	0.008
Random Forest	0.9875	0.8833	0.8856	0.8833	0.8835	0.298

Top 5 Predictive Features:
`ca` (major vessels), `thal` (thalassemia type), `cp` (chest pain), `oldpeak` (ST depression), `thalach` (max heart rate).

Performance on High-Dimensional Data (Intel Image Classification)

Effect of Ensemble Size:

n_estimators	Train Accuracy	Test Accuracy	Training Time (s)
1	0.9875	0.3542	0.234
10	0.9979	0.4208	1.567
50	1.0000	0.4583	6.234
100	1.0000	0.4708	11.892
300	1.0000	0.4750	34.567

Single Decision Tree vs. Random Forest:

Model	Train Acc	Test Acc	Precision	Recall	F1	Time (s)
Decision Tree	1.0000	0.3417	0.3523	0.3417	0.3389	0.189
Random Forest	1.0000	0.4708	0.4756	0.4708	0.4698	11.892

> The Random Forest improved test accuracy by 12.91 percentage points. High-dimensional raw pixel data remains challenging; CNNs are typically preferred.

Synthesized Insights

- **Accuracy and Stability:** Rapid gains with more trees (~100), then plateau. More trees reduce variance.
- **Generalization and Overfitting:** Random Forests reduce overfitting of single, fully-grown trees.
- **Randomness and Ensemble Size:** Bootstrap sampling + feature randomization decorrelates trees. 100–300 trees leverage the Law of Large Numbers.

---

4. Conclusion

Random Forest is a **powerful, robust, and versatile** model whose practical performance aligns with theoretical foundations. Its strength comes from combining strong, low-bias learners (deep trees) while minimizing correlation through dual randomization.

Key Takeaways:

1. **Empirical Validation:** Outperforms single trees, improves generalization, and avoids overfitting as more trees are added.
2. **Practical Recommendations:** 100–300 trees balance accuracy and computational cost.
3. **Broad Applicability:** Excels on tabular data, less effective on raw high-dimensional data without feature engineering or specialized architectures.

Limitations: Reduced interpretability compared to a single tree, higher memory usage.
Despite this, Random Forests remain a cornerstone of modern machine learning.