International Journal of Scientific Research in Engineering and Management (IJSREM)
Volume: 09 Issue: 12 | Dec - 2025 SJIF Rating: 8.586 ISSN: 2582-3930

A Comparative Study of XGBoost and Random Forest for Modern Machine Learning Applications

,

Er. Gundeep Kaur[1] , Dr. Rachna Rana[2], Mr. Sachin Sharma[3],Er. Manpreet Kaur[4]

Er. Gundeep Kaur , Assistant Professor, Computer Science& Engineering ,Ludhiana Group Of Colleges, Chaukimann, India
Dr. Rachna Rana, Assistant Professor, Computer Science& Engineering ,Ludhiana Group Of Colleges, Chaukimann, India
Mr. Sachin Sharma, Assistant Professor, Computer Science& Engineering ,Ludhiana Group Of Colleges, Chaukimann, India
Er. Manpreet Kaur , Assistant Professor, Computer Science& Engineering ,Ludhiana Group Of Colleges, Chaukimann, India

Abstract - Selecting the appropriate ensemble method is critical for predictive tasks in modern data science. This paper provides a comparative study between two widely used tree-ensemble algorithms: **Random Forest** (bagging) and **XGBoost** (gradient boosting). We analyze algorithmic design, computational characteristics, hyperparameter sensitivity, handling of missing/sparse data, interpretability, and typical application scenarios. Empirical patterns reported in the literature and practical guidelines are synthesized to help practitioners choose between these approaches. The study highlights XGBoost's superior predictive performance in many benchmark scenarios and Random Forest's robustness and simplicity for rapid development. [1]

Keywords: Random Forest, XGBoost, Ensemble Learning, Predictive Performance, Hyperparameter Sensitivity, Sparse Data Handling

1.Introduction

Ensemble learning is central to many high-performance supervised learning tasks. Random Forest (RF) and eXtreme Gradient Boosting (XGBoost) represent two dominant ensemble paradigms: bagging and boosting, respectively. RF builds many independent trees trained on bootstrap samples and aggregates predictions, offering stability and low tuning burden. XGBoost builds trees sequentially by optimizing a differentiable loss using gradient information and includes engineering optimizations and regularization, yielding strong predictive power on complex datasets.[13] Choosing between them requires understanding tradeoffs in accuracy, computational cost, and interpretability. [2]

2. Background and Related Work

Random Forest was formalized by Breiman and remains a standard baseline for classification and regression problems. Its strengths include variance reduction via bagging and simple measures of feature importance. XGBoost (2016) brought system-level optimizations (sparsity-aware split finding, weighted quantile sketch, cache-aware layout) and algorithmic regularization (L1/L2), rapidly becoming a de-facto choice for competition-grade models. Since then, additional GBDT implementations (LightGBM, CatBoost) improved speed, categorical handling, and robustness; benchmarking studies compare these gradient boosting variants empirically. [2]

3. Methodology

This is a conceptual and literature-driven comparison organized around four axes:

1. **Algorithmic architecture** — learning type, tree construction, regularization.

2. **Computational characteristics** — training time, memory footprint, parallelism and distributed capabilities.

3. **Predictive behavior** — accuracy, variance/bias tradeoff, sensitivity to hyperparameters, and overfitting tendencies.

4. **Practical aspects** — missing data handling, categorical features, interpretability, and typical domains of success.

Empirical results are drawn from recent benchmark and domain studies summarized in the literature (no new dataset experiments were performed here). Representative literature is cited throughout. [14]

4. Algorithm Overviews

4.1 Random Forest (RF)

- **Type:** Bagging ensemble of decision trees.[7]

- **Training:** Each tree trained independently on a bootstrap sample; at each split a random subset of features is considered. Predictions aggregate via majority vote (classification) or averaging (regression).[15]

- **Hyperparameters (typical):** number of trees (n_estimators), max features per split (max_features), tree depth (max_depth), min samples per leaf.[9]

- **Strengths:** Robust to overfitting, low tuning effort, natural variable importance measures, works well on noisy/small-to-medium datasets.

- **Weaknesses:** May be outperformed by tuned boosting on large, complex datasets; does not natively include strong regularization like boosting frameworks. [2]

-

4.2 XGBoost

- **Type:** Gradient boosting decision trees (GBDT) with system and algorithmic optimizations.[7]

- **Training:** Trees built sequentially; at each iteration a tree fits the negative gradient of the loss. Includes regularization terms (L1/L2), shrinkage (learning rate), row/column subsampling, and specialized split finding. XGBoost implements sparsity-aware tree learning and optimizations for cache/memory.[15]

- **Hyperparameters (typical):** learning rate (eta), n_estimators, max_depth, subsample, colsample_bytree, regularization parameters (lambda, alpha).

- **Strengths:** High accuracy on many benchmarks, handles sparse data efficiently, strong regularization controls overfitting, scalable implementations.

- **Weaknesses:** More sensitive to hyperparameter tuning, less interpretable out-of-the-box relative to RF. [1]

5. Comparative Analysis

5.1 Algorithmic Structure & Learning Dynamics

- **RF (bagging):** Reduces variance by averaging many de-correlated trees; each tree is a high-variance learner but averaging reduces overfit.

- **XGBoost (boosting):** Reduces bias by sequentially fitting residuals; regularization and shrinkage are critical to prevent overfitting.[13] Boosting often attains better fit but requires careful regularization.[2][7]

5.2 Computational Complexity & Scalability

- **Training speed:** XGBoost's optimized C++ implementation and parallel tree construction (plus later variants like LightGBM with histogram-based splits) make boosting implementations faster for large

datasets per unit predictive power, but tuning increases wall-time. LightGBM and other GBDT frameworks further improved speed. [3]

- **Memory:** RF can be memory-heavy (many full trees stored), but XGBoost and LightGBM include memory optimizations and on-disk structures for very large datasets. [1]

5.3 Predictive Performance & Robustness

- **Accuracy:** Many comparative studies report that a properly tuned XGBoost (or other modern GBDT like LightGBM/CatBoost) typically outperforms Random Forest on tabular tasks, especially when feature engineering and hyperparameter tuning are applied. However, RF often provides competitive results with less tuning effort. [14]

- **Overfitting:** RF's averaging is a simple but effective guard against overfitting; XGBoost's explicit regularization (and careful tuning) addresses overfitting but misconfiguration can increase risk. [1][13]

5.4 Handling Missing & Categorical Data

- **Missing values:** XGBoost includes sparsity-aware split decisions (learns default directions for missing values). RF commonly requires imputation or modified implementations to handle missing values natively. [1]

- **Categorical features:** Native categorical handling is not present in classic RF/XGBoost — but implementations vary; CatBoost (and newer versions of other GBDTs) add native categorical support and reduce target leakage. For tabular use-cases with many categorical variables, CatBoost or LightGBM with proper encoding can outperform naive approaches. [4]

5.5 Interpretability & Diagnostics

- **Random Forest:** Easier to interpret: feature importance (Gini/permute), partial dependence plots, and decision path analysis are straightforward.[7]

- **XGBoost:** Feature importance and SHAP values are commonly used; SHAP provides consistent per-prediction attributions but is computation-heavy. For practitioners needing raw interpretability, RF often requires fewer auxiliary tools. [8][15]

6. Practical Guidelines & Recommendations

Based on the literature synthesis and practitioner guidelines:

- **Start simple:** If you need a fast baseline with minimal tuning, use Random Forest. It's robust for noisy and moderate datasets.[1][16][8]

- **For top predictive performance:** Use XGBoost (or LightGBM/CatBoost) with cross-validated hyperparameter tuning and early stopping. These models usually deliver superior accuracy on tabular data when tuned. [16][8]

- **For large, sparse datasets:** XGBoost and LightGBM are optimized for sparsity and scale. [1]

- **If categorical variables dominate:** Consider CatBoost or careful encoding with LightGBM. [4]

7. Figures & Tables

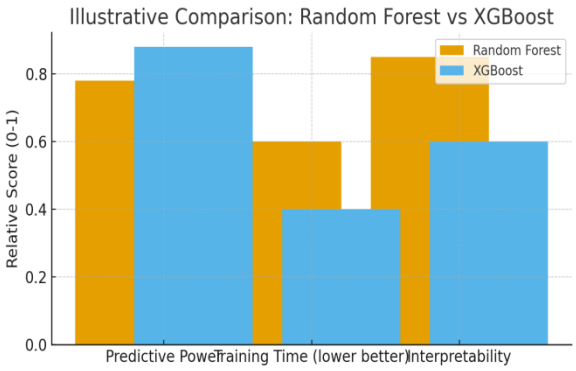

Figure 7.1 Comparison: RF vs XGBoost

Table 7.1 Algorithmic Structure [13][15]

Feature	Random Forest	XGBoost
Ensemble Type	Bagging	Boosting
Tree Dependency	Independent trees	Sequential trees
Regularization	Not explicit	Explicit L1 & L2
Overfitting Control	Randomness & averaging	Shrinkage, subsampling, regularization

8. Conclusion

Random Forest and XGBoost both remain essential tools. Random Forest is the go-to for quick, robust baselines and interpretability; XGBoost (and other modern GBDTs) are preferred when predictive power and scalability matter and when the practitioner can devote time to hyperparameter tuning.[15] The final choice should be guided by dataset size, sparsity, presence of categorical features, compute availability, and tolerance for tuning complexity. Future work should include standardized empirical benchmarks on domain-specific datasets and comparison of runtime / energy costs for large models. [6]

References

1. Chen, T., & Guestrin, C., "XGBoost: A Scalable Tree Boosting System," *Proceedings of the 22nd ACM SIGKDD*, 2016. DOI: **10.1145/2939672.2939785**.

2. Breiman, L., "Random Forests," *Machine Learning*, vol. 45, pp. 5–32, 2001. DOI: **10.1023/A:1010933404324**.

3. Ke, G., Meng, Q., Finley, T., Wang, T., Chen, W., Ma, W., Ye, Q., Liu, T.-Y., "LightGBM: A Highly Efficient Gradient Boosting Decision Tree," *NeurIPS*, 2017.

4. Prokhorenkova, L., Gusev, G., Vorobev, A., Dorogush, A. V., & Gulin, A., "CatBoost: unbiased boosting with categorical features," *NeurIPS*, 2018.

5. Bentéjac, C., Csörgő, A., & Martínez-Muñoz, G., "A comparative analysis of gradient boosting algorithms," *Artificial Intelligence Review*, 2021. DOI: **10.1007/s10462-020-09896-5**.

6. Florek, P., & Zagdanski, A., "Benchmarking state-of-the-art gradient boosting algorithms for classification," *arXiv:2305.17094*, 2023. DOI/ArXiv: **10.48550/arXiv.2305.17094**.

7. Shao, Z., Ahmad, M. N., & Javed, A., "Comparison of Random Forest and XGBoost Classifiers Using Integrated Optical and SAR Features for Mapping Urban Impervious Surface," *Remote Sensing*, vol. 16, 2024, Article 665. DOI: **10.3390/rs16040665**.

8. Boldini, D., et al., "Practical guidelines for the use of gradient boosting models in cheminformatics," *Scientific Reports / PMC*, 2023. (Open access guidance and best practices).

9. scikit-learn developers, "RandomForestClassifier — scikit-learn documentation" (latest), scikit-learn.org — implementation & API notes (2024).

10. Chen, T. Q., & Guestrin, C., "XGBoost (arXiv preprint)," arXiv:1603.02754, 2016.

11. Ke, G., Meng, Q., et al., *NeurIPS proceedings / LightGBM PDF*. .

12. Prokhorenkova, L., et al., *CatBoost NeurIPS paper (PDF / DOI page)*.

13. Bentéjac et al., downloadable author version (UAM repository) — DOI: **10.1007/s10462-020-09896-5**

14. Florek (2023) benchmarking preprint and related code repository for replicated experiments.

15. Additional applied comparisons (2021–2024) across domains that show XGBoost/LightGBM often outperform Random Forest when tuned — see the Remote Sensing 2024 paper and the Artificial Intelligence Review 2021 comparative study for benchmarks.

16. Turner, S. (2025). NeurIPS 2025 workshop on biosecurity safeguards for generative AI. https://doi.org/10.59350/ep7k2-ngq70