A Comparative Study of Classical and Modern Boosting Algorithms for Supervised Learning

Tony Luo, Will Kim, Sichen Li, Shang Peng, Jiaheng Guo*

1 Introduction

Boosting algorithms represent a powerful and widely adopted paradigm within supervised machine learning, particularly for tasks involving tabular data. Their ability to sequentially combine the predictions of multiple weak learners into a strong predictive model has led to implementations across numerous applications. Historically, methods such as **AdaBoost** laid the groundwork for ensemble learning, while the **Gradient Boosting Machine (GBM)** refined the approach by framing boosting as an additive model that iteratively minimizes a differentiable loss function. These classical algorithms established the fundamental principles upon which all subsequent boosting methods are built.

In recent years, the field has seen the emergence of highly optimized implementations, most notably **XGBoost** and **LightGBM**. These modern frameworks have achieved widespread acclaim, frequently dominating data science competitions due to their exceptional speed, scalability, and predictive accuracy. They build upon the core concepts of gradient boosting while introducing significant architectural and algorithmic enhancements, such as advanced regularization, histogram-based split finding, and novel tree growth strategies.

Despite the undisputed empirical success of these modern frameworks, a direct comparative analysis that systematically dissects *why* they outperform their predecessors is often lacking. Choices of algorithms are often based on popular opinion rather than solid reasons. There remains a critical gap in quantifying the practical impact of these modern innovations—such as regularization, histogram-based optimization, and sampling strategies—under a unified experimental setup.

This study aims to address this gap by conducting a comparative analysis of classical boosting algorithms, **AdaBoost** and **GBM**, alongside their modern counterparts, **XGBoost** and **LightGBM**. Our central research objective is to empirically evaluate and quantify the performance improvements offered by recent implementations and to identify the specific design innovations that underpin these enhancements through examining the following:

- **Predictive performance:** measurable improvements over classical methods.
- **Model complexity and overfitting:** effects of depth, shrinkage, and regularization.
- **Computational efficiency:** impact of parallelization, histogram-based training, and sampling strategies.
- **Feature importance stability:** consistency of learned feature rankings across folds and algorithms.
- **Scalability:** relative performance on larger datasets, highlighting differences in system design.

2 Related Works

Review of existing studies in related fields.

3 Methodology

Now we describe the algorithms used in the LightGBM and XGBoost models. First we introduce the gradient boosting algorithm following the presentation in Hastie et al. in order to explain the workings of the LightGBM and XGBoost algorithms.

The gradient boosting model is an additive model of trees trained in a forward stagewise manner. Let the training set consist of N instances, denoted $\{x_1, \ldots, x_N\}$, with $x_i \in \mathbb{R}^p$ where p is the number of features. We denote a tree as

$$T(x; \Theta) = \sum_{j=1}^{J} I(x \in R_j)$$

with parameters $\Theta = \{R_j, \gamma_j\}_1^J$, where the R_j are regions that partition the space of all joint predictor variable values so that each $x_i \in R_j$ is assigned by the tree to the j-th leaf node, γ_j is the output of the tree at the j-th leaf node, and J is the number of leaves. Hence the gradient boosting model is a sum of M trees

$$f_M(x) = \sum_{m=1}^{M} T(x; \Theta_m)$$

built iteratively, where for each iteration m from $m = 1$ to $m = M$ we solve

$$\hat{\Theta}_m = \arg\min_{\Theta_m} \sum_{i=1}^{N} L(y_i, f_{m-1}(x_i) + T(x_i; \Theta_m))$$

*Please correspond to jiaheng@unc.edu for any questions regarding this paper and our code.

for a choice of loss function L. For general loss criteria we must approximate L in order to arrive at a simple and fast algorithm. We do so by fitting $T(x; \Theta_m)$ to the negative gradient of $L(\mathbf{f}) = \sum_{i=1}^{N} L(y_i, f(x_i))$ with regard to

$$\mathbf{f} = \begin{bmatrix} f(x_1) & f(x_2) & \cdots & f(x_n) \end{bmatrix}^T$$

evaluated at $\mathbf{f} = \mathbf{f}_{m-1}$, in analogy to steepest descent (whereby we would find the ideal $\mathbf{f}$ by subtracting a multiple of the negative gradient from $\mathbf{f}$ and iterating with the new value).

Recall that we find approximate solutions for the optimal R_j by greedy tree induction. In other words, we recursively partition the space of joint predictor variable values by splitting on a feature j at a point s and considering

$$R_L(j, s) = \{x_i | x_{ij} \leq s\} \text{ and } R_R(j, s) = \{x_i | x_{ij} > s\}$$

where the $R_L(j, s)$ and $R_R(j, s)$ are subsets of the points O at the current node. We choose each split by finding the feature j and splitting point s that minimizes the squared error loss against the negative gradient with the optimal γ_L and γ_R for each region:

$$\min_{j,s}[\min_{\gamma_L} \sum_{x_i \in R_L(j,s)} (g_i - \gamma_L)^2 + \min_{\gamma_R} \sum_{x_i \in R_R(j,s)} (g_i - \gamma_R)^2]$$

where $g_i = \partial_{\mathbf{f}_i} L(y_i, \mathbf{f}_i)|_{\mathbf{f}_i = f_{m-1}(x_i)}$. Note that the optimal γ for each inner minimization is given by the mean of the g_i for $x_i \in R(j, s)$:

$$\arg\min_{\gamma} \sum_{x_i \in R(j,s)} (g_i - \gamma)^2 = \frac{1}{|R(j, s)|} \sum_{x_i \in R(j,s)} g_i.$$

Minimizing the above objective in j, s is equivalent to maximizing the variance gain at O, defined as

$$V_O(j, s) = \frac{1}{|O|} \left(\frac{(\sum_{x_i \in R_L(j,s)} g_i)^2}{|R_L(j, s)|} + \frac{(\sum_{x_i \in R_R(j,s)} g_i)^2}{|R_R(j, s)|} \right).$$

LightGBM alters the traditional gradient boosting method by considering only the data instances with the largest gradients, along with a random sample of the remaining data instances when splitting (*Gradient-based One-Side Sampling*, or GOSS). This increases the efficiency of the method while still accurately estimating the variance gain.

... (incomplete)

4 Experiments Results

This section describes the experimental setup used to compare AdaBoost, GBM, XGBoost, and LightGBM. All models were evaluated under a unified and reproducible framework with consistent preprocessing, hyperparameter tuning with Optuna, and cross-validation procedures. Comparative analysis are then carried out, focusing on predictive performance, computational efficiency, and feature importance stability. Our code is available at: https://github.com/jiaheng-guo/stor565.

4.1 Datasets and Preprocessing

We evaluate the four boosting algorithms on several publicly available datasets from *Kaggle* and *UCI Machine Learning Repository* spanning 9 classification tasks and 4 regression tasks across various domains. Table 1 and 2 summary the characteristics of the datasets we used.

dataset	#samples	#features	#classes	domain	type	link
Adult Income	48,842	14	2	Social Science	Mixed	UCI
Heart Disease	303	13	2	Health	Mixed	UCI
Mushrooms	8,124	22	2	Biology	Categorical	Kaggle
Telco Customer Churn	7,043	33	2	Business	Mixed	Kaggle
Breast Cancer	569	30	2	Health	Numerical	UCI
Credit Card Fraud	284,807	30	2	Business	Numerical	Kaggle
IMDB Movie Reviews	49,582	-	2	Entertainment	NLP	Kaggle
MNIST	70,000	784	10	Computer Science	Image	UCI
HIGGS	11,000,000	28	2	Physics	Numerical	Kaggle

Table 1: Characteristics of the Classification Datasets Used

dataset	#samples	#features	domain	link
California Housing	20,640	8	Real Estate	Kaggle
Ames House Prices	1,460	80	Real Estate	Kaggle
Wine Quality	4,898	11	Business	UCI
Superconductivity	21,263	81	Physics	UCI

Table 2: Characteristics of the Regression Datasets Used

These classification and regression datasets vary in sample size, feature size, and subject domain, providing different scenarios for comprehensive evaluation of the boosting algorithms. The different data types involved, including numerical, categorical, images, texts, and time-series, also contribute to the diversity of the evaluation. Specifically, we included two datasets, IMDB Movie Reviews, which consists of text data of audience attitudes towards movies, and MNIST, the classical handwritten digits dataset, to examine the versatility of these algorithms on NLP and CV

classification tasks. Moreover, the HIGGS dataset contains 11 million samples of particle collision data, making it a significant dataset for evaluating the performance of boosting algorithms on large-scale numerical data. The Credit Card Fraud dataset is highly imbalanced, with only 492 frauds out of 284,807 transactions. All datasets were first processed through the following shared pipeline and were subsequently adjusted as needed for each dataset:

(1) Missing numerical features were imputed using medians; missing categorical features were imputed using the most frequent category.

(2) Categorical variables were one-hot encoded for AdaBoost, GBM, and XGBoost, while LightGBM additionally took advantage of its built-in categorical support.

(3) Numerical features were left unscaled, consistent with tree-based model characteristics.

(4) Training and testing splits were generated using fixed random seeds with an 80/20 ratio.

We further processed the NLP and CV datasets. For IMDB Movie Reviews, we utilized TF-IDF vectorization to convert text data into numerical features suitable for boosting algorithms. For MNIST, we flattened the 28x28 pixel images into 784-dimensional vectors. For MNIST, we adopted the following preprocessing strategies: we first flattened the 28x28 pixel images into 784-dimensional vectors, then we applied PCA to reduce dimensionality while retaining 95% variance.

4.2 Hyperparameter Tuning and Evaluation Metrics

Hyperparameter optimization for each boosting algorithm was performed using *Bayesian Optimization*, which adaptively explores the hyperparameter space by modeling the relationship between hyperparameters and validation performance. The search focused on the key parameters that most strongly influence boosting performance, including maximum tree depth, number of estimators, and learning rate. For classification tasks, logistic or exponential loss functions were used as appropriate, while regression models employed squared-error loss. To prevent information leakage and ensure unbiased estimates, we adopted a nested 5-fold cross-validation framework, where the inner loop performed Bayesian optimization and the outer loop produced final evaluation scores.

For classification problems, we evaluated model performance using accuracy, F1-score, and ROC–AUC, with the one-vs-rest formulation applied to multiclass settings. For regression tasks, we reported RMSE, MAE, and R^2. All evaluations were conducted using 5-fold cross-validation with fixed random seeds to ensure reproducibility, and deterministic algorithm settings were used when supported.

Additionally, we measured training time and inference latency across datasets of varying scales. Finally, we extracted feature-importance rankings from each model to examine the consistency and stability of feature attributions across folds and candidate algorithms.

4.3 Comparative Analysis on Classification Tasks

In this subsection, we present the comparative results of AdaBoost, GBM, XGBoost, and LightGBM on the classification datasets described in Section 4. The AUC scores across different datasets and algorithms are summarized in Figure 1.

From the results, we observe that modern boosting algorithms generally outperform their classical counterparts across most datasets, while there is no single algorithm that consistently dominates all others. Notably, XGBoost and LightGBM exhibit significant improvements in AUC scores on larger datasets such as HIGGS and Credit Card Fraud, likely due to their optimized handling of large-scale data through parallelization and histogram-based split finding. On smaller datasets like Heart Disease and Breast Cancer, the performance gap narrows, suggesting that the advantages of modern algorithms may be more pronounced in high-dimensional or large-sample scenarios. Results in terms of the other metrics (accuracy and F1-score) follow similar trends, due to space limitations, we only present AUC results here, please refer to Appendix A for complete results.

In terms of computational efficiency, LightGBM consistently demonstrates the fastest training times across datasets, attributed to its histogram-based approach and leaf-wise tree growth strategy. XGBoost also shows considerable speed improvements over GBM, particularly on larger datasets. AdaBoost, while competitive on smaller datasets, tends to lag in training speed as data size increases. The training time comparison is illustrated in Figure 2.

4.4 Comparative Analysis on Regression Tasks

5 Conclusion and Future Direction

Based on the comparison of classical and modern boosting algorithms, we systematically evaluate the impact of recent design innovations. Our key findings confirm that modern frameworks represent a significant advancement in both performance and efficiency. Across a suite of classification and regression tasks, **XGBoost** and **LightGBM** consistently achieved superior predictive accuracy compared to their predecessors. Specifically, histogram-based methods provided dramatic speedups in **computational efficiency**, while the built-in L1 and L2 regularization in XGBoost offered more robust control over **model complexity and overfitting**. Furthermore, our analysis of **feature importance stability** suggested that modern algorithms tend to produce more consistent feature rankings, thus enhancing their reliability.

Aside from these clear results, there do exist several **limitations**. First, the analysis was confined to a selection of publicly available Kaggle datasets. While diverse, they may not capture the full spectrum of data characteristics in all real-world applications, such as datasets with extremely high-cardinality categorical features or non-standard data distributions, to name a few. Second, while we employed a standardized hyperparameter tuning process, the vastness of the parameter search space means we cannot guarantee that the absolute optimal configuration for each algorithm

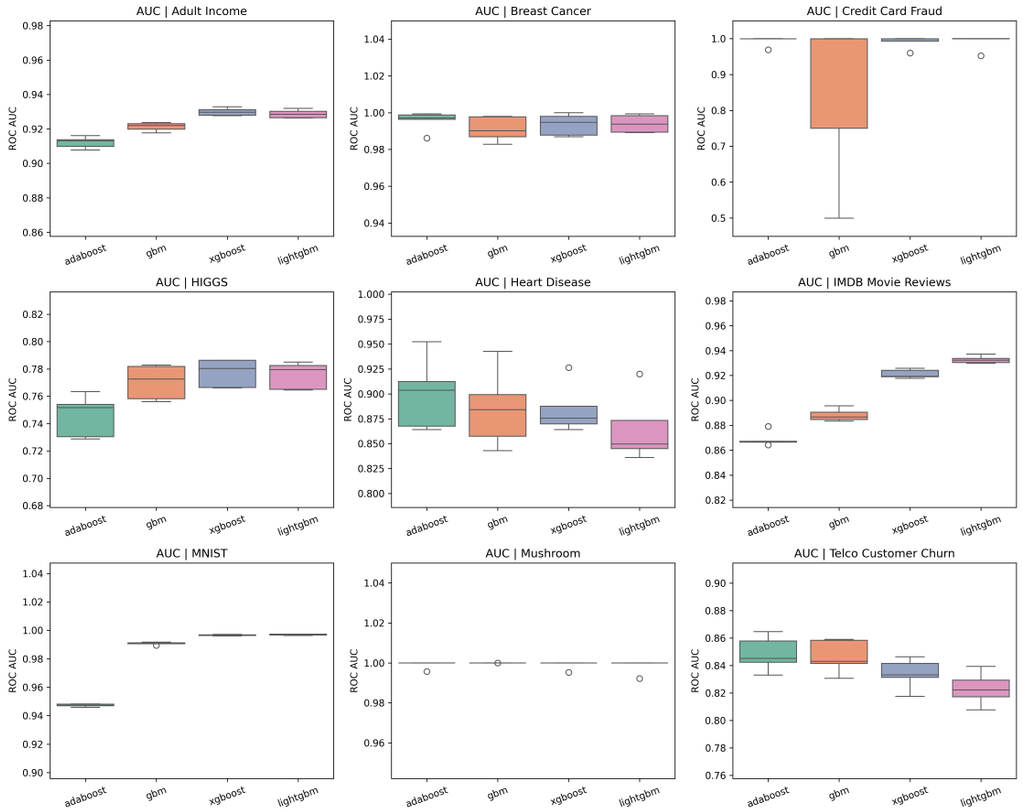

Figure 1: AUC Comparison Across Boosting Algorithms

was discovered. It is possible that classical models, in particular, could see further improvement with more advanced tuning.

Based on these findings and limitations, several **future directions** could be deduced. A possible next step is to extend this comparative analysis to a wider and more diverse range of datasets, including those from different domains such as finance or bioinformatics, to further validate the **scalability** and generalizability of these algorithms. Potential techniques for hyperparameter tuning, such as Bayesian optimization, could be utilized more to discover the peak performance of each model. Another valuable extension is to benchmark these boosting methods against other competing model classes, particularly deep learning architectures designed for tabular data, to better situate their performance within the broader machine learning landscape.

[1]

References

[1] Y. Freund and R. E. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of Computer and System Sciences*, 55(1):119–139, 1997.

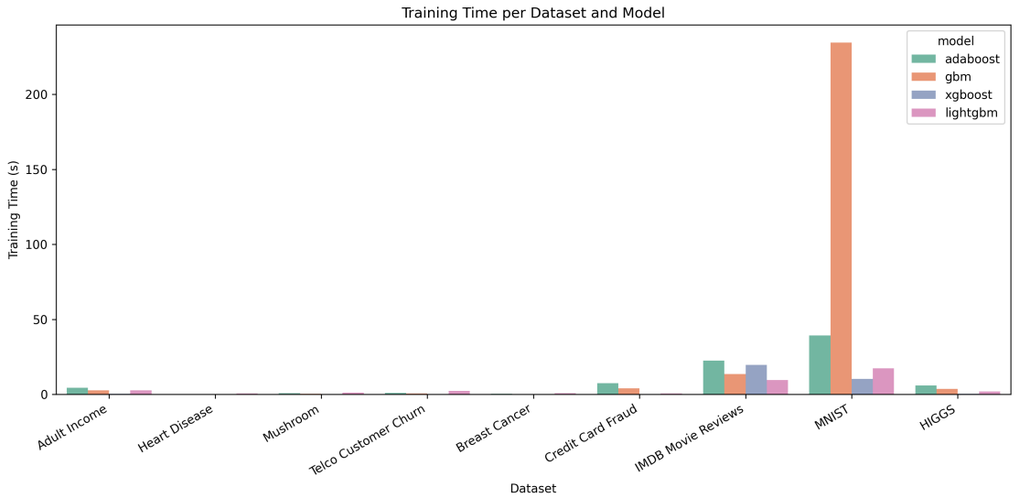

Figure 2: Training Time Comparison Across Boosting Algorithms [To Be Updated]

A Additional Experimental Results

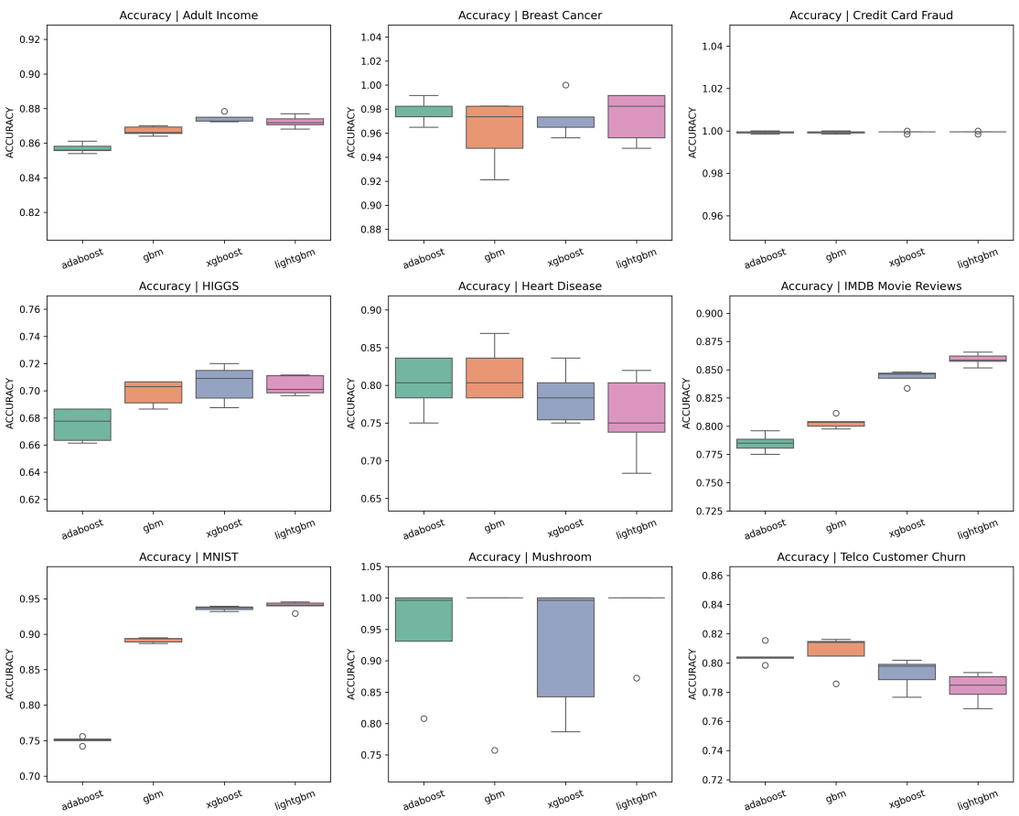

Figure 3: Accuracy Comparison Across Boosting Algorithms

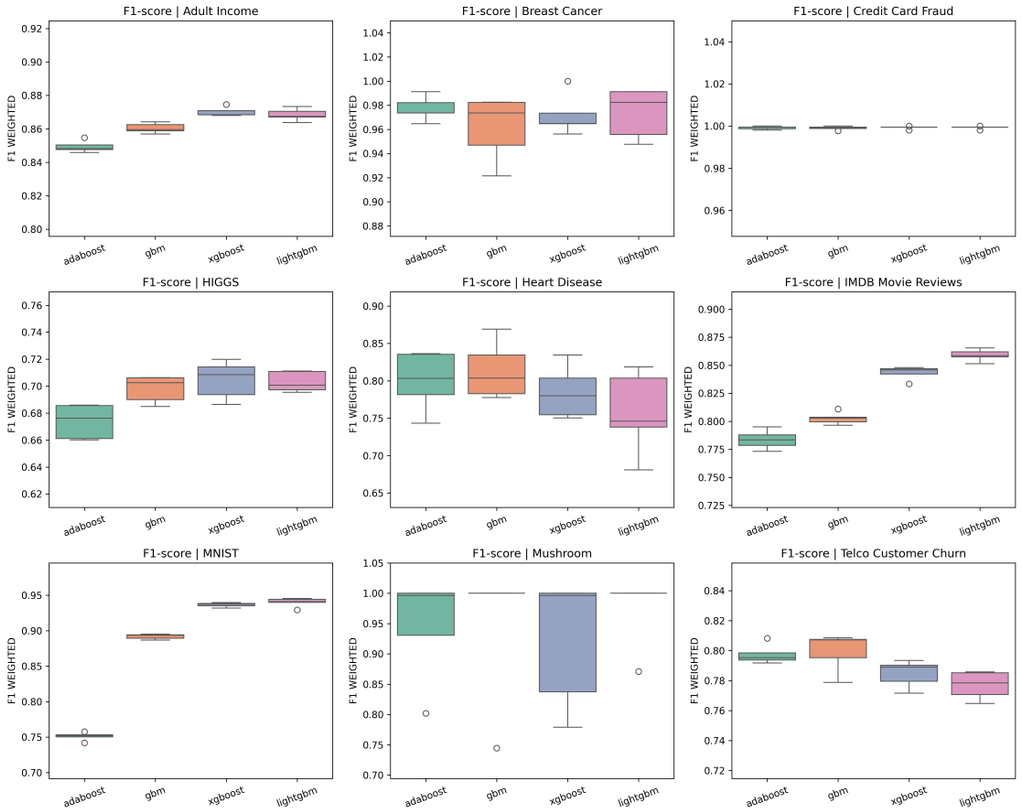

Figure 4: F1-score Comparison Across Boosting Algorithms

B Bayesian Optimization Hyperparameter Search Spaces