Overview of Tree-based Machine Learning Model

ECE225A Final Project Proposal

Enting Zhou, *PID: A16159365*

Abstract—**Decision trees ensembles are powerful predictive machine learning models across many different real-world applications. This projects presents a comprehensive overview of ensemble learning techniques on decision tree model, specifically bootstrap aggregation, adaptive boosting, and gradient boosting. Key papers are thoroughly reviewed and key characterises of each technique is summarized. Through empirical evaluation using seven benchmark datasets for binary classification tasks, the study identifies gradient boosting as the superior model, demonstrating exceptional performance and generalization capabilities. The findings also indicate that while AdaBoost shows resilience to data variability, it is slightly outperformed by gradient boosting. In contrast, random forests and decision trees tend to overfit and show more performance variability, respectively.**

I. Introduction

PREDICTIVE modeling is a vital aspect of data analysis, machine learning, and artificial intelligence. Among the most popular techniques for predictive modeling are tree-based algorithms: decision trees, random forests, adaptive boosting trees, and gradient boosting trees. These models have demonstrated great performance in various scenario, outperforming other classification methods like Logistic Regression, Support Vector Machine, and Deep Neural Networks. The significance of this project is to understand and give insights into why improved tree-based models works well, as well as their limitations.

The project reviewed the following listed papers in the reference sessions, covering decision trees, random forest, adaptive boosting trees, and gradient boosted trees. This project also conducted an empirical experiment on comparison on common decision trees implementations, as in those in scikit-learn[1] on 7 benchmark datasets for binary classification tasks.[2]. These benchmark datasets are suggested by Grinsztajn et al.[1], as the dataset have heterogeneous columns, I.I.D data, and non-deterministic data, attributes that similar to real-world applications. The link to the source code for the experiment is at: https://github.com/ETZET/MLTreeAnalysis.

Paper Reviewed:

- Freund, Yoav, Robert E. Schapire AT, and T. Labs. 1999. "A Short Introduction to Boosting." 1999. http://www.yorku.ca/gisweb/eats4400/boost.pdf.
- Geurts, Pierre, Damien Ernst, and Louis Wehenkel. 2006. "Extremely Randomized Trees." Machine Learning 63 (1): 3–42.
- Grinsztajn, Leo, Edouard Oyallon, and Gael Varoquaux. 2022. "Why Do Tree-Based Models Still Outperform Deep Learning on Typical Tabular Data?" In Advances in Neural Information Processing Systems, edited by S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh, 35:507–20. Curran Associates, Inc.
- Myles, Anthony J., Robert N. Feudale, Yang Liu, Nathaniel A. Woody, and Steven D. Brown. 2004. "An Introduction to Decision Tree Modeling." Journal of Chemometrics 18 (6): 275–85.
- Natekin, Alexey, and Alois Knoll. 2013. "Gradient Boosting Machines, a Tutorial." Frontiers in Neurorobotics 7 (December): 21.

[1]https://scikit-learn.org/stable/index.html

[2]https://www.openml.org/search?type=data&status=active&id=45024

II. Bagging and Random Forests

First, we will focus on the bootstrap aggregation (bagging) techniques on decision trees proposed by Breiman, Leo[**?**]. Bagging creates k sets of new training dataset, having the same size as the original training dataset, by sampling with replacement on the original training dataset. Using bootstraped dataset, we can creates decision trees models called random forests. We give the formal definition of random forest below.

A random forest is a classifier consisting of a collection of tree-structured classifiers h(x, k), k = 1,... where the k are independent identically distributed random vectors and each tree casts a unit vote for the most popular class at input x[2].

Random Forests have two sources of randomness: 1) bagging 2) Attribute Sampling. Bagging refers to training each subtree with a randomly sampled subset of the training dataset. Attribute Sampling refers to when growing the sub decision trees, a random subset of features are sampled to split the tree nodes. The two sources of randomness act as a powerful regularization for decision tree modeling, They ensure the relative independence between the decision trees. This independence corrects the overfitting of the individual decision trees. Consequently, the ensemble is not overfitted.

III. Boosting Algorithms

Other than bagging, boosting is another technique to create ensemble machine learning models. The difference between boosting and bagging is in the training procedure. Bagging involves training multiple models at once all together, whereas boosting trains new classifiers incrementally from the output of previous trained classifiers. Boosting improves machine models' predictive accuracy and performance by converting multiple weak learners into a single strong learning model. Machine learning models can be weak learners or strong learners: Weak learners have low accuracy resembling random guessing and vulnerability to overfitting, and may struggle with classifying data outside their original training set. Despite their limitation in predictive power, they are favorable in computation efficiency. In contrast, strong learners, achieving higher accuracy, are formed through boosting, a process that unifies weak learners into a robust system.

There are two main boosting mechanism: 1) Adaptive Boosting 2) Gradient Boosting. The rest of this section will delve deep into these two boosting methods.

AdaBoost: The first boosting algorithm we will review is Adaptive Boosting (AdaBoost)[3]. AdaBoost starts by assigning the same weights to all data samples. Then it repeatedly trains base classifiers to correctly classify all the samples, and at each iteration, it assigns more weights to one that are incorrectly classified by the previous base classifier. The final prediction result is a weighted majority vote of the base classifiers.

The intricate part of AdaBoost algorithm is how it adaptively corrects its own errors. Here we describe the procedure in detail. For each base learner, the error is quantified as

$$\epsilon_t = \Pr_{i\sim D_t}[h_t(x_i) \neq y_i] = \sum_{i:h_t(x_i)\neq y_i} D_t(i).$$

Where, $D_t(i)$ is the weight of each training sample. This error is intuitively the sum of weights of the incorrectly classified example. Next the algorithm chose α_t. α_t is the measure of the importance of the base learner in the final prediction. α_t is chose to be

$$\alpha_t = \frac{1}{2}\ln\left(\frac{1-\epsilon_t}{\epsilon_t}\right)$$

Next, the algorithm updates the weight for each training sample based on the classification result for the current base learner, up-scaling the weight for incorrectly classified ones and down-scaling the weight for correctly classified ones. Specifically,

$$\begin{aligned} D_{t+1}(i) &= \frac{D_t(i)}{Z_t} \times \begin{cases} e^{-\alpha_t} & \text{if } h_t(x_i) = y_i \\ e^{\alpha_t} & \text{if } h_t(x_i) \neq y_i \end{cases} \\ &= \frac{D_t(i)\exp(-\alpha_t y_i h_t(x_i))}{Z_t} \end{aligned}$$

Adaboost fused together multiple weak learners into a strong learner. If each weak learner is slightly better than the random guess, then the training error will drop exponentially fast. This is a property shared by many other boosting algorithms, but the good thing about Adaboost is the the generalization error can be bounded. Schapire et al.'s work have shown that theoretically, the generalization error for AdaBoost can be expressed as:

$$\hat{P}_r[\text{margin}(x,y) \leq 0] + \tilde{O}\left(\sqrt{\frac{d}{m\theta^2}}\right)$$

The amazing thing about this bound is that the upper bound is entirely independent of the number of weak learners we trained, showing that the generalization does not get worse if we train more iterations, thus tend not to overfit the training data. Another interesting observation for AdaBoost algorithms is its connection with the support vector machine.

One drawback for the AdaBoost algorithm is that it is very sensitive to outliers[3], as it iteratively assigns greater weights to misclassified points, and in terms of outlier in the data ends up with huge influence in the final prediction. But people have utilized this characteristic as an application of AdaBoost to identify outliers, after training the AdaBoost classifier, samples with the highest weights often turn out to be outliers.

Gradient Boosting: Similar to Adaboost, gradient boost, proposed by Friedman[4], builds the final prediction by successfully building a base learner to obtain the final ensemble model. However gradient boost does not scale each base learner to correct the base learner's error, instead it weights all base learners the same, the subsequent base learner does not aim to produce the full prediction result but predicting a residual term, based on previous prediction result to fit the training data correctly.

Here we delve more deeply into gradient boosting trees sequential modeling process. Specifically, for a given dataset of length m and d features, given a differentiable loss function L, gradient boosting tree uses k additive functions to predict the output:

$$\hat{y}_i = \phi(x_i) = \sum_{k=1}^{K} f_k(x_i), \quad f_k \in \mathcal{F} \tag{1}$$

where $\mathcal{F} = \{f(x) = w_{q(x)} \mid (q : R^m \to T, w \in R^T)\}$ is the space of regression trees. To learn the function to approximate the true output, the algorithm takes iterative approach to minimize the function:

$$\mathcal{L}(\phi) = \sum_i l(\hat{y}_i, y_i) + \sum_k \Omega(f_k)$$

where $\Omega(f) = \gamma T + \frac{1}{2}\lambda\|w\|^2$, posing a regularization of the learned function space, and l is a differentiable loss function.

The algorithm begins by initializing the model $F_0(x)$ to minimize the expected value of a loss function $L(y_i, \rho)$ over the training set. Then it iterates k steps to progressively improve the model. In each iteration, the algorithm calculates the pseudo-residuals $\tilde{y}_i$ for each observation in the training set, which are derived from the gradient of the loss function with respect to the predictions of the current model $F_{k-1}(x)$.

Next, the algorithm seeks an optimal parameter vector by fitting a base learner, typically a decision tree, to the negative gradients (pseudo-residuals). The base learner's output is then scaled by a factor ρ_m, a learning rate hyper parameter that determine how the scaled output is added to the current model's predictions. Note in practice, instead of using a tree stump as base learner like AdaBoost, gradient boost use a larger tree with fixed depth, ranging from 4 - 32 leaves per tree.

The final step of each iteration updates the model by adding the scaled output of the base learner to the existing model $F_{k-1}(x)$, thereby forming the new model $F_k(x)$. This additive model is built in a stage-wise fashion, where each new base learner is fitted on the current pseudo-residuals, thus reducing the overall model's residuals. The final output is a model that has been refined through successive iterations to provide a strong prediction by reducing the loss function's value over the training set.

The exact optimization process is contingent to the chosen loss function and base learner. The choice of loss function is usually determined by the task, and regularization posed on the learning objective. Sum of squared errors is usually used for regression task, while exponential likelihood error is usually used for classification task. Gradient boosting provides a versatile and powerful framework by supporting a wide variety of choice of loss function and base learner. In practice, gradient boosting is used together with fixed sized decision tree models as base learner.

IV. Empirical Evaluation

In this section, we conduct an empirical comparison of the tree-based machine learning model on binary classification tasks. We examine the predictive performance of Decision Tree, Random Forest, AdaBoost, and Gradient Boosting Decision Trees in real world datasets. Here we used 7 mid-size tabular dataset to bench marking the models' performance. The choice of dataset is suggested by suggested by Grinsztajn et al.[1], as these dataset have heterogeneous columns, I.I.D data, and non-deterministic data, attributes that similar to real-world applications. Sources and detailed description of each dataset can be found at the Hugging Face website.[3]

In our empirical study comparing the performance of various tree-based machine learning algorithms in a binary classification task, we employ key evaluation metrics to comprehensively assess model effectiveness. Accuracy, a foundational metric, quantifies the overall correctness of predictions. The F1 score, harmonizing precision and recall, provides a balanced measure considering both false positives and false negatives. Precision evaluates the proportion of true positives among positive predictions, emphasizing the model's ability to minimize false positives. Meanwhile, recall assesses the model's capability to identify all relevant instances, capturing the ratio of true positives to the sum of true positives and false negatives. These metrics collectively offer a detailed understanding of the classification models' efficacy in our experimental setting.

Figure 1 bar plots compares the averaged accuracy, F1 score, precision, and recall between training and testing sets for the same four models. Here, a clear trend of overfitting is visible with the Decision Tree and Random Forest models, where the training scores are significantly higher than the testing scores, indicating that these models may not generalize well to unseen data. The AdaBoost and Gradient Boosting models exhibit better generalization with closer training and testing scores. Notably, Gradient Boosting maintains high levels of both training and testing metrics, suggesting it is the most effective model among those compared for balancing bias and variance, thus yielding a model that performs well on both seen and unseen data.

Figure 2 depicts the distribution of testing accuracy, F1 score, precision, and recall for four different machine learning models: Decision Tree, Random Forest, AdaBoost, and Gradient Boosting. It is evident that Gradient Boosting tends to have a higher median testing accuracy and F1 score, as well as less variance in its accuracy distribution, indicating more consistent performance across different runs. The AdaBoost model also shows robustness, with a narrower interquartile range in precision and recall, though with a few outliers indicating some variations in performance. The Decision Tree model appears to be the least stable, with the widest distribution in accuracy and F1 score, suggesting its performance is more sensitive to the variations in the testing data.

Based on the visualizations, we can draw several conclusions about the performance of the four machine learning models. The Gradient Boosting model stands out as the most proficient, demonstrating high and consistent scores across all metrics, with a strong balance between training and testing results which indicates good generalization without significant overfitting. The AdaBoost model follows as a close competitor, with relatively stable performance metrics and fewer outliers, suggesting resilience to data variability. However, it doesn't reach the same peak performance as Gradient Boosting. The Random Forest model, while showing promising training results, reveals a tendency to overfit, as seen by the drop in testing metrics. The Decision Tree model, although simple and interpretable, shows the most variability and the lowest scores in testing metrics, suggesting it may be too simplistic to capture the complexities in the data as compared to the ensemble methods. These insights would favor Gradient Boosting as the preferred model for this particular dataset, given its superior and more reliable predictive performance.

It is worth mentioning that the performance of each model can be much improved through extensive feature engineering and hyperparameter tuning. These processes involve selecting the most informative attributes, transforming data into a better representation, and meticulously adjusting the model parameters for optimal performance. However, such enhancements are beyond the scope of this project. Our objective was to compare the fundamental performance of different algorithms under a consistent set of conditions, allowing for a more direct comparison of their inherent capabilities and behaviors. This foundational analysis provides valuable insights into which models show the most promise before further investment in fine-tuning and optimization is made.

V. Conclusion

In summary, our exploration encompassed key papers in the field, each contributing unique insights into the design and functioning of tree-based models. The empirical evaluation, conducted on seven benchmark datasets for binary classification tasks, provided a thorough examination of decision tree, random forest, AdaBoost, and gradient boosting models. Leveraging key evaluation metrics such as accuracy, F1 score, precision, and recall, we gained a nuanced understanding of their respective performances.

The findings revealed that gradient boosting emerged as a standout performer, demonstrating superior and consistent predictive capabilities across various metrics. Its ability to

[3]https://huggingface.co/datasets/inria-soda/tabular-benchmark

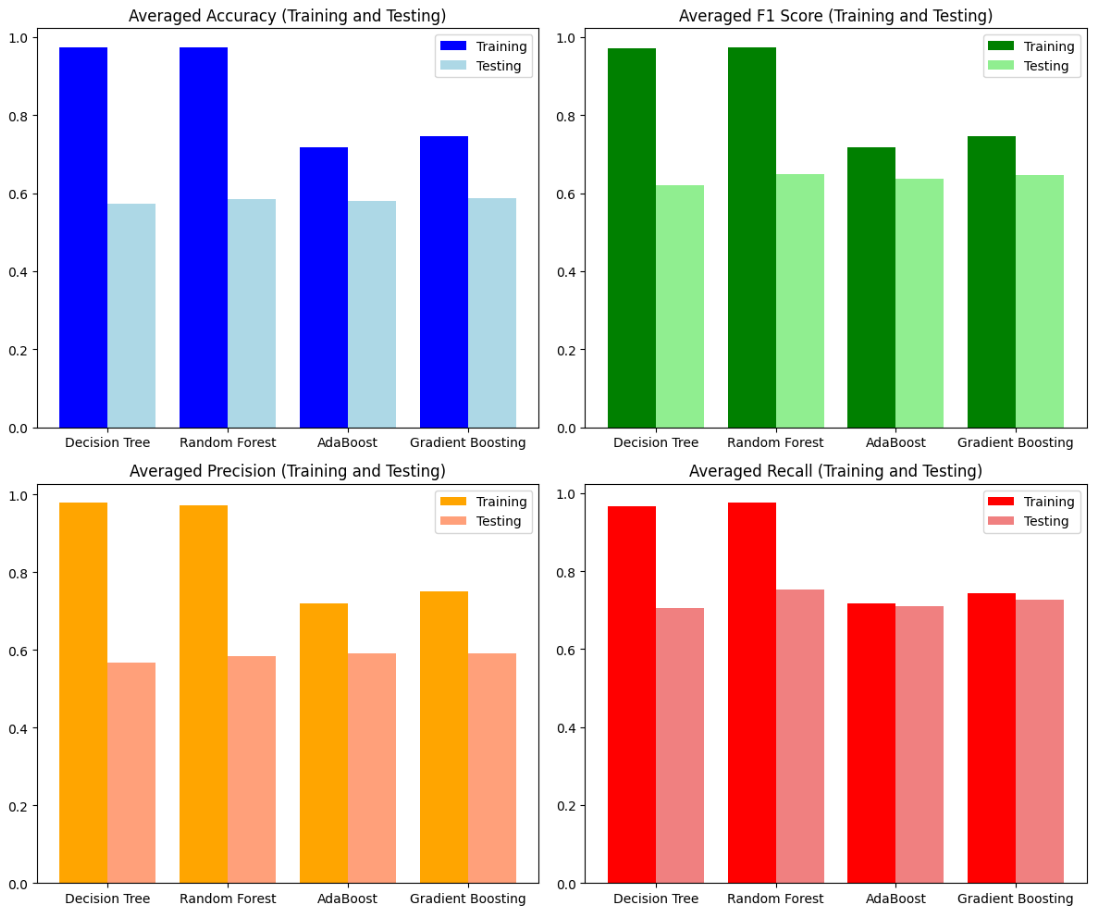

Fig. 1. Training vs. Testing Performance Metrics for Decision Tree, Random Forest, AdaBoost, and Gradient Boosting Models.

balance bias and variance, showcasing high generalization without overfitting, positions it as a compelling choice for the explored dataset. AdaBoost exhibited resilience to data variability but fell slightly short of the peak performance achieved by gradient boosting. Random forests, while promising in training, displayed a tendency to overfit, and basic decision trees exhibited the most variability and comparatively lower testing metrics.

For future work, it would be beneficial to extend this research to explore multi-class classification problems and to include additional sophisticated ensemble techniques, such as XGBoost[5] and LightGBM[6]. Investigating the impact of feature selection methods and dimensionality reduction techniques on model performance could also yield significant insights. Furthermore, the application of advanced hyperparameter optimization strategies, like Bayesian optimization or genetic algorithms, could refine model performances even further. Conducting a similar empirical evaluation on datasets with a higher dimensionality or on real-world data with noise and outliers would provide a comprehensive understanding of the robustness of these models.

References

[1] L. Grinsztajn, E. Oyallon, and G. Varoquaux, “Why do tree-based models still outperform deep learning on typical tabular data?” in *Advances in Neural Information Processing Systems*, S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh, Eds., vol. 35. Curran Associates, Inc., 2022, pp. 507–520.

[2] L. Breiman, “Random forests,” *Mach. Learn.*, vol. 45, no. 1, pp. 5–32, Oct. 2001.

[3] Y. Freund, R. E. Schapire AT, and T. Labs, “A short introduction to boosting,” http://www.yorku.ca/gisweb/eats4400/boost.pdf, 1999, accessed: 2023-12-6.

[4] J. H. Friedman, “Greedy function approximation: A gradient boosting machine,” *Ann. Stat.*, vol. 29, no. 5, pp. 1189–1232, 2001.

[5] T. Chen and C. Guestrin, “XGBoost: A scalable tree boosting system,” in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ser. KDD ’16. New York, NY, USA: Association for Computing Machinery, Aug. 2016, pp. 785–794.

[6] “Lightgbm: A highly efficient gradient boosting decision tree,” https://proceedings.neurips.cc/paper/2017/hash/6449f44a102fde848669bdd9eb6b76fa-Abstract.html, accessed: 2023-11-18.

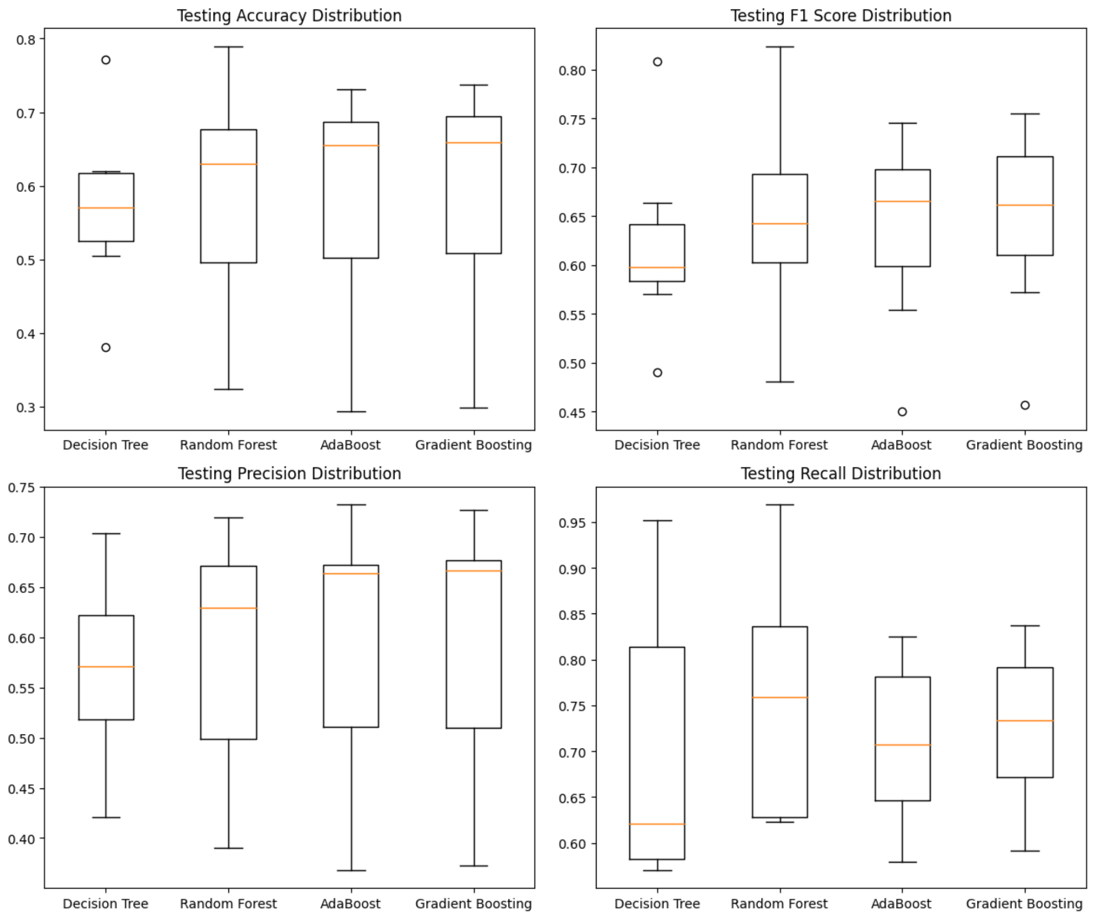

Fig. 2. Comparative Distribution of Testing Metrics Across Decision Tree, Random Forest, AdaBoost, and Gradient Boosting Models.