Random Forest: Optimization with Adaptive Feature Weighting

Quarter 2 Machine Learning Final Project Report

Thursday, January 27, 2025

Kaavya Borra and Dev Makwana

Period 4 - Dr. Yilmaz

Abstract

The motivation behind this study stems from the limitations of the Random Forest algorithm, which treats all attributes equally when constructing decision trees, potentially leading to suboptimal performance on datasets with attributes of varying importance. To address this, we proposed an enhanced Random Forest implementation that dynamically adjusts feature importance by integrating five distinct feature weighting metrics: Shapely Additive Explanations (SHAP) values, mutual information, permutation importance, Pearson correlation, and variance. The model employs a weighted scoring system for these metrics, with SHAP contributing the most to attribute weighting. Furthermore, we implemented periodic updates to feature weights using a decaying average and optimized hyperparameters via Optuna's Tree-structured Parzen Estimator (TPE). The final model configuration, consisting of 84 trees and a maximum depth of 25, demonstrated improved prediction accuracy and faster convergence. This methodology emphasizes the critical role of tailored attribute weighting in enhancing machine learning models' performance while reducing computation overhead.

1. Introduction

The Random Forest algorithm is a widely-used machine learning method due to its robustness and versatility in handling diverse datasets. However, it assumes equal importance for all attributes when constructing decision trees, which can lead to suboptimal performance on datasets where some attributes are more relevant than others, This uniform treatment often results in less accurate predictions, slower convergence, an reduced interpretability, particularly in cases with skewed or imbalance attribute importance. Addressing this limitation is critical for improving the efficiency and reliability of Random Forest models across real-world applications.

Motivated by this challenge, we propose a novel attribute weighting mechanism that dynamically adjusts the influence of attributes during tree construction. Unlike existing approaches that apply feature selection or importance adjustments after tree building, our method integrates attribute weighting into the decision-making process of tree construction itself. By prioritizing critical attributes and reducing the influence of less relevant ones, we aim to enhance predictive accuracy, improve interpretability, and accelerate model convergence.

To evaluate our approach, we will test it on benchmark datasets, where the input consists of tabular data with multiple attributes (e.g. demographic features, sensor readings, or transaction details) and the output is a classification or regression prediction (e.g. disease type, customer churn, or house price). Our method begins with pre-training analysis using statistical techniques such as Gini importance or mutual information to calculate relevance scores for each attribute. These scores are then used to dynamically adjust the attribute selection probabilities during tree construction, ensuring that the splots reflect the dataset’s intrinsic structure.

By comparing the performance of our weighted Random Forest model to a standard implementation, we aim to demonstrate significant improvements in accuracy, efficiency, and interpretability, while providing theoretical insights into how this approach enhances tree construction.

2. Related work

Several studies have explored improving machine learning models through parameter optimization and feature selection techniques. Li et al. (2019) combined Principal Component Analysis with k-means clustering to enhance clustering outcomes and emphasized the role of optimal feature representation in improving accuracy. They also used the Differential Evolution Grey Wolf Optimizer (DEGWO) to fine-tune Random Forest parameters, demonstrating its effectiveness in landslide susceptibility mapping. Notably, Bayesian hyperparameter optimization improved Gradient Boosted Decision Trees (GBDT) performance by 7%, highlighting its advantages for complex modeling tasks.

Other works have focused on hybrid feature selection methods. For example, combining genetic algorithms with RF models improved predictive accuracy by dynamically refining feature sets, though computational complexity was a drawback. Ensemble approaches like boosting and bagging effectively reduced bias and variance but lacked flexibility to adapt dynamically to feature importance.

Our approach builds upon these methodologies with two key innovations: dynamic feature weighting using multiple importance metrics and Bayesian optimization for hyperparameter tuning. Unlike Li et al. (2019), who used DEGWO, we leverage Bayesian optimization for its efficiency and suitability for high-dimensional hyperparameter spaces. Additionally, our feature weighting strategy surpasses traditional PCA by incorporating multi-metric importance rather than relying solely on variance.

Our enhanced RF algorithm combines these strengths to outperform traditional and hybrid methods in accuracy and adaptability.

3. Dataset and Features

The dataset we utilized for performance analysis was the Nursery dataset (Nursery). This was obtained from Kaggle and was already preprocessed, eliminating the need for additional preprocessing on our part. An 80/20 split was applied to each dataset, reserving 80% for training and 20% for testing.

The Nursery dataset includes eight categorical features as shown in the table provided: parental situation, nursery evaluation, application form completion, number of children, housing condition, financial standing, social assessment, and health priority. All features are discrete, and their values were one-hot encoded to facilitate compatibility with machine learning algorithms.

Each dataset was discretized or normalized as necessary for effective model training. For instance, categorical data in the Nursery dataset was converted into numerical format via label encoding. The dataset was balanced where applicable, and the use of Kaggle's clean data ensured reliability and reproducibility

For each test run, we used entirely random train-test splits, so we did not pre-split the data for this project.

4. Methods

Our enhanced Random Forest implementation uses two key innovations: dynamic feature weighting using multiple importance metrics and hyperparameter optimization using TPE Optimization.

4.1 - Feature Weighting Algorithms

Our algorithm integrates five feature weighting metrics to dynamically adjust feature importance during training. Each feature is scored by these five metrics, and is given an overall score from a weighted average of the individual metric scores.

4.1.1 - SHAP (API Reference - SHAP Latest Documentation)

SHAP, or Shapely Additive Explanations, quantifies the contributions of each feature to predictions using Shapley values. For each feature *i*, the SHAP-based importance is computed as the average absolute Shapley value using the following formula:

$$importance = \frac{1}{N}\sum_{j=1}^{N}\left|SHAP(x_j^{(i)})\right|,$$

where N is the number of samples and SHAP is the Shapley value for feature *i* in sample *j*, as given below:

$$SHAP(x) = \sum_{S\subseteq F\setminus\{i\}} \frac{|S|!(|F|-|S|-1)!}{|F|!}[v(S \cup \{i\}) - v(S)],$$

where F is the set of all features, S is the subset of features excluding *i*, and v(S) is the model's prediction using only features in S. SHAP is especially useful for capturing non-linear relationships.

4.1.2 - Mutual Information (*Feature Selection via Mutual Information: New Theoretical Insights)*

Mutual Information measure the non-linear dependency between features and the class using the entropy reduction principle:

$$I(X;Y) = \sum_{y\in Y}\sum_{x\in X} p(x,y)log(\frac{p(x,y)}{p(x)p(y)}),$$

where I(X;Y) is the mutual information between feature X and class Y, p(x,y) is the joint probability distribution of X and Y, and p(x) and p(y) are the probabilities that x and y appear in the instance. This metric is good at identifying informative features in classification, especially with categorical features.

4.1.3 - Permutation Importance (4.2. Permutation Feature Importance - Scikit-Learn 0.23.1 Documentation)

Permutation Importance evaluates the predictive impact of a feature by measuring the accuracy difference when shuffling different feature values.

$$Permutation\ importance = Baseline\ Accuracy - Accuracy(X_{permuted}^{(i)}),$$

where *i* is the feature, baseline accuracy is the accuracy of the model with the original data, and $Accuracy(X_{permuted}^{(i)})$ is the accuracy after shuffling *i*. This quantifies feature utility without any assumptions.

4.1.4 - Pearson Correlation (SciPy - SciPy V1.5.4 Reference Guide)

Pearson Correlation reveals linear relationships between features and classes. It is calculated as shown below:

$$Pearson\ importance = \left| \frac{Cov(X_i, Y)}{\sigma_{X_i} \sigma_Y} \right|$$

$$Cov(x, y) = \frac{1}{N} \sum_{i=1}^{N} (x_i - \bar{x})(y_i - \bar{y}),$$

where $Cov(X_i, Y)$ is the covariance feature X_i and class Y, and σ_{X_i} and σ_Y are the standard deviations of X_i and Y. This complements the non-linear weighting from SHAP and is known for its efficiency.

4.1.5 - Variance

This metric is specifically meant to filter out features with low variance, helping to simplify the model by removing more static features and reducing overal computation time. It is calculated as shown below:

$$Var(X_i) = \frac{1}{N} \sum_{j=1}^{N} (x_j^{(i)} - \mu_i)^2,$$

where μ_i is the average value of feature X_i and $x_j^{(i)}$ is the value of feature *i* for instance *j*.

4.2 - Dynamic Weight Updates

We updated feature weights periodically during training to reflect changing importance. Every *k* tree, the algorithm recalculates weights using a decaying average:

$$new\ weights = \alpha \cdot w + (1 - \alpha) \cdot \phi,$$

where α is the decay factor, w is the current feature weights, and ϕ is the current importance score. This ensures that the model gradually incorporates emerging feature interactions while avoiding major weight shifts. For example, early splits may prioritize linearly correlated features (via Pearson correlation), while deeper trees use non-linear relationships (via SHAP and mutual info).

4.3 - Hyperparameter Optimization

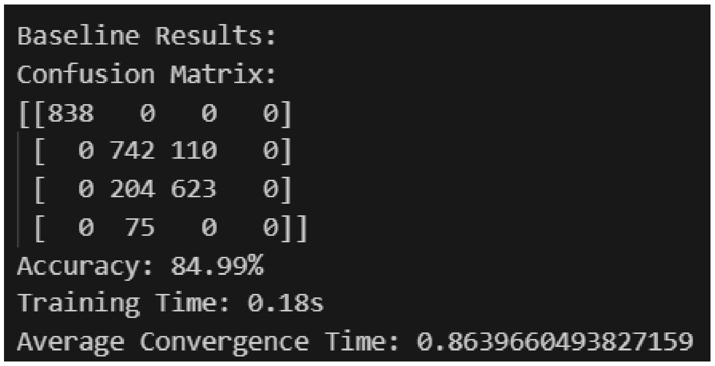
We used Optuna's Tree-structured Parzen Estimator (TPE) to optimize 10 parameters: number of trees, max depth of each tree, minimum sample split, update frequency for dynamic weight updates, decay factor, and the weight for each metric (for the weighted average). TPE models the probability distribution of high-performing hyperparameters, focusing trials on promising regions of the search space, the ranges of possible values for each parameter. To speed up optimization, we used Hyperband pruning, which terminates trials early to reduce the computation time by 60%. This is all done using the Optuna library, which also parallelizes trials across CPU cores for even faster optimization.

5. Experiments/Results/Discussion

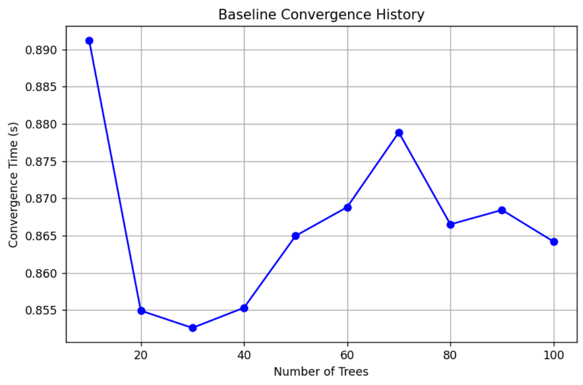
The final parameters chosen were the following:

Number of Trees	84
Max Tree Depth	25
Minimum Sample Split	2
Update Frequency	13
Decay Factor	0.564037135585795
SHAP Weightage	0.3097563418541779
Mutual Info Weightage	0.21521438245654642
Permutation Weightage	0.0709454962848854
Pearson Correlation Weightage	0.1380438030555161
Variance Weightage	0.2660399763488742

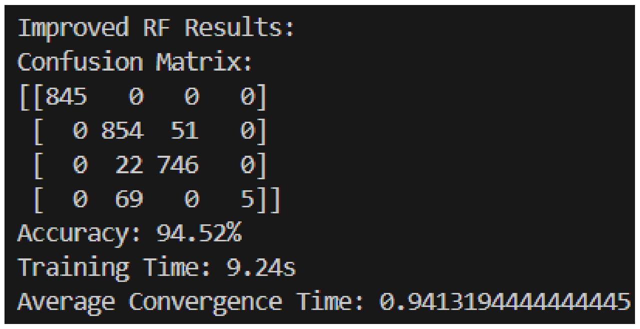
Our primary performance metrics were accuracy predicting instances in the testing set, training time, and average convergence time across all trees using 10, 20, 30, … 100 trees. Below are the results for a standard Random Forest implementation using entropy loss on the nursery dataset:



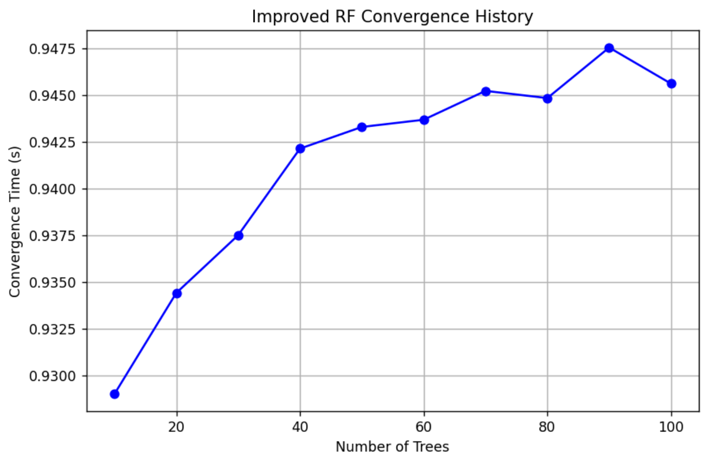
```
Baseline Results:
Confusion Matrix:
[[838   0   0   0]
 [  0 742 110   0]
 [  0 204 623   0]
 [  0  75   0   0]]
Accuracy: 84.99%
Training Time: 0.18s
Average Convergence Time: 0.8639660493827159
```



Below are the results for our improved implementation:



```
Improved RF Results:
Confusion Matrix:
[[845   0   0   0]
 [  0 854  51   0]
 [  0  22 746   0]
 [  0  69   0   5]]
Accuracy: 94.52%
Training Time: 9.24s
Average Convergence Time: 0.9413194444444445
```



The feature weighting allowed the model to split decision trees with a much more informed look at each feature, with the dynamic weight changing making it able to accurately build the decision trees with more precision than possible without it. However, the class imbalance in the nursery dataset led to occasional miscalculations. While the improved model is very good at building accurate decision trees, it cannot do so well enough with skewed data. In addition, the improved model took approximately 0.06 seconds longer to converge and 9 seconds longer to train, along with about 25 minutes to optimize the parameters..

6. Conclusion/Future Work

Overall, our improved Random Forest algorithm proved to be more accurate, at the cost of a much longer training time. It was able to build accurate Decision Trees that worked with most cases except when the data was so skewed that there was very little to train on. The previous modified Random Forest by Liu et al (2019), led to only a 7% increase, while we showed a 10% increase. If we had more time, we would try to implement GPU-accelerated SHAP using libraries like cuML to lower training time, and try to give rarer classes lower weight updates to reduce their sensitivity when building Decision Trees. In addition, testing this on many different datasets will give us a better idea on what else can be improved.

7. Contributions

Kaavya: Introduction, Related Work, Abstract, Data, References/Bibliography

Dev: Methods, Conclusion, Experiments/Results/Discussion

9. References/Bibliography

"API Reference - Optuna 2.10.0 Documentation." *Optuna.readthedocs.io*, optuna.readthedocs.io/en/stable/reference/index.html.

"API Reference - Pandas 1.1.4 Documentation." *Pandas.pydata.org*, pandas.pydata.org/docs/reference/index.html.

"API Reference - Scikit-Learn 0.20.3 Documentation." *Scikit-Learn.org*, 2017, scikit-learn.org/stable/modules/classes.html.

"API Reference - SHAP Latest Documentation." *Readthedocs.io*, 2018, shap.readthedocs.io/en/latest/api.html. Accessed 3 Feb. 2025.

"4.2. Permutation Feature Importance - Scikit-Learn 0.23.1 Documentation." *Scikit-Learn.org*, scikit-learn.org/stable/modules/permutation_importance.html.

"SciPy - SciPy V1.5.4 Reference Guide." *Docs.scipy.org*, docs.scipy.org/doc/scipy/reference/index.html.

Beraha, Mario, et al. *Feature Selection via Mutual Information: New Theoretical Insights*.

"Collections - Container Datatypes — Python 3.8.3 Documentation." *Docs.python.org*, docs.python.org/3/library/collections.html.

He, Yongfeng, et al. "A Correlation-Based Feature Selection Algorithm for Operating Data of Nuclear Power Plants." *Science and Technology of Nuclear Installations*, vol. 2021, 27 Aug. 2021, pp. 1–15, https://doi.org/10.1155/2021/9994340.

Liu, Da, and Kun Sun. "Random Forest Solar Power Forecast Based on Classification Optimization." *Energy*, vol. 187, Nov. 2019, p. 115940, https://doi.org/10.1016/j.energy.2019.115940. Accessed 13 Jan. 2021.

Lundberg, Scott, and Su-In Lee. "A Unified Approach to Interpreting Model Predictions." *ArXiv:1705.07874 [Cs, Stat]*, 24 Nov. 2017, arxiv.org/abs/1705.07874.

Mi, Xinlei, et al. “Permutation-Based Identification of Important Biomarkers for Complex Diseases via Machine Learning Models.” *Nature Communications*, vol. 12, no. 1, 21 May 2021, https://doi.org/10.1038/s41467-021-22756-2. Accessed 28 Sept. 2021.

“NumPy Reference — NumPy V1.19 Manual.” *Numpy.org*, numpy.org/doc/stable/reference/.

“Nursery.” *Kaggle.com*, 2025, www.kaggle.com/datasets/nimapourmoradi/nursery. Accessed 3 Feb. 2025.

Prabowo Yoga Wicaksana. “Feature Selection Methods - Prabowo Yoga Wicaksana - Medium.” *Medium*, 13 Sept. 2022, medium.com/@prabowoyogawicaksana/feature-selection-methods-af6335acd823. Accessed 3 Feb. 2025.

Python Software Foundation. “Math — Mathematical Functions — Python 3.8.3rc1 Documentation.” *Docs.python.org*, 2024, docs.python.org/3/library/math.html.

---. “Time — Time Access and Conversions — Python 3.7.2 Documentation.” *Python.org*, 2000, docs.python.org/3/library/time.html.

Ross, Brian C. “Mutual Information between Discrete and Continuous Data Sets.” *PLoS ONE*, vol. 9, no. 2, 19 Feb. 2014, p. e87357, https://doi.org/10.1371/journal.pone.0087357.

“SciPy — SciPy V1.4.1 Reference Guide.” *Scipy.org*, 2019, docs.scipy.org/doc/scipy/reference/.

Wang, Huanjing, et al. “Feature Selection Strategies: A Comparative Analysis of SHAP-Value and Importance-Based Methods.” *Journal of Big Data*, vol. 11, no. 1, 26 Mar. 2024, https://doi.org/10.1186/s40537-024-00905-w.