Research on the Effectiveness of the Random Forest Algorithm in Multiclass Classification Problems: Theoretical Analysis and Experimental Comparison

This work examines the machine learning algorithm "Random Forest." A detailed theoretical analysis of its operating principles is conducted, including the methods of bagging and random subspaces. The experimental part compares the performance of Random Forest with a single Decision Tree and a Gaussian Naive Bayes classifier on a synthetic dataset. The results show that the ensemble method demonstrates higher generalization ability and accuracy (Accuracy = 0.8833) compared to single models.

---

1. Introduction

In modern data science, classification problems hold a central place. Despite the rising popularity of deep learning, classical machine learning algorithms remain the standard for working with tabular data due to their interpretability and efficiency with limited computational resources. A major issue with single decision trees is their tendency toward **overfitting**, where they build overly complex decision boundaries that describe the training sample perfectly but perform poorly on new data (high variance). The **Random Forest** algorithm, proposed by Leo Breiman, solves this problem by combining many weak classifiers into one strong ensemble. The goal of this work is to demonstrate the superiority of the ensemble approach over single models using a complex synthetic dataset with overlapping classes as an example.

---

2. Algorithm Theory

Random Forest belongs to the class of ensemble learning methods, utilizing the **Bagging (Bootstrap Aggregating)** strategy. The core idea is that combining multiple models, each having high variance and low bias, helps reduce the overall variance of the ensemble without significantly increasing the bias.

2.1. Operating Principle

The algorithm builds an ensemble of N decision trees. Training involves three steps:

- **Bootstrap**: For each tree, a separate training sample is created by selecting M objects from the initial dataset *with replacement*. Some objects may be selected multiple times, while others may not be included at all (out-of-bag).
- **Random Subspaces Method**: When building each node, the optimal feature for splitting is chosen from a *random subset* of features, typically $k = \sqrt{D}$ where D is the total number of features, rather than from all available features. This reduces the correlation between the trees.
- **Aggregation**: For classification, the final prediction is determined by **majority voting**, where the class voted for by the largest number of trees becomes the ensemble's answer.

2.2. Splitting Criteria

The Gini Impurity criterion is used for selecting the split point. For a node t with class distribution p_i, the Gini index is calculated as:

$$I_G(t) = 1 - \sum_{i=1}^{C} p_i^2$$

The goal is to minimize the weighted sum of Gini indices for the child nodes after the split, where C is the number of classes.

2.3. Advantages and Disadvantages

- **Advantages**:
  - **Resistance to Overfitting** due to averaging results.
  - **Work with Outliers** as it is less sensitive to noise than a single tree.
  - **No Need for Scaling** since trees do not require feature normalization (unlike neural networks or KNN).
- **Disadvantages**:
  - **Complexity of Interpretation** because the forest is a "black box" unlike a single tree.

- **Prediction Speed** is slower, requiring more time for inference as data must pass through N trees.

---

3. Data and Experimental Methodology

A synthetic dataset was generated using sklearn.datasets.make_classification to test the hypothesis of Random Forest's superiority, simulating a complex classification task.

3.1. Dataset Characteristics

The data had the following parameters:

- **Sample Size (N)**: 1000 objects.
- **Feature Space**: 2 informative features (for visualization).
- **Number of Classes**: 3.
- **Noise**: flip_y=0.05 (5% of class labels were randomly inverted).
- **Separability**: class_sep=1.2 (classes had partial overlap).

3.2. Preprocessing

The data was split into Training (X_{train}: 700 samples) and Testing (X_{test}: 300 samples) samples in a 70/30 ratio. A fixed random seed (random_seed=42) ensured reproducibility.

---

4. Implementation

The experiment was conducted in Python using scikit-learn, numpy, and matplotlib. The three models compared were:

- **Decision Tree**: Gini criterion.
- **Random Forest**: Gini criterion, ensemble of trees.
- **Gaussian Naive Bayes**: Used as a baseline.

Neural networks were not included, focusing on tree-based algorithms.

---

5. Experimental Results

Quality assessment was performed on the test sample (300 objects), using **Accuracy** as the primary metric, along with **Precision**, **Recall**, and **F1-score**.

Detailed Metrics for Random Forest

The algorithm achieved the highest accuracy:

- **Overall Accuracy**: 0.8833

Class	Precision	Recall	F1-score	Support (count)
0	0.91	0.80	0.86	107
1	0.87	0.95	0.91	96
2	0.87	0.91	0.89	97

Analysis: The model performed best for Class 1 (Recall = 0.95), missing very few objects of this class. For Class 0, the precision was high (0.91), meaning it rarely made false predictions for this class, but the recall was lower (0.80), indicating it missed 20% of the actual Class 0 objects.

---

6. Comparison with Other Models

A comparison was made with the single Decision Tree and Gaussian Naive Bayes classifier.

Summary Table of Results (Test Set):

Model	Accuracy	Macro Avg F1	Note
Random Forest	0.8833	0.88	Best result
Decision Tree	0.8567	0.86	Tendency towards overfitting
Naive Bayes	0.8567	0.86	Limited by linearity

Discussion

- **Random Forest vs. Decision Tree**: The Forest outperformed the single tree by nearly 2.7% (0.8833 vs. 0.8567). This supports the theory that ensembling reduces error variance; the Forest's voting mechanism smoothed the decision boundary where a single tree might have overfit noise.
- **Random Forest vs. Naive Bayes**: Naive Bayes achieved the same result as the Decision Tree (0.8567) but was inferior to the Forest. This result is expected because Naive Bayes assumes feature independence and normal data distribution, conditions not always perfectly met by the synthetic

dataset, whereas the Random Forest can model complex nonlinear dependencies.

---

7. Conclusions

The study analyzed the Random Forest algorithm and compared it with classical classification methods. The experiment on synthetic data (1000 samples, 3 classes) concluded that:

- **Random Forest is the most effective model** in this experiment, reaching an accuracy of **88.33%**.
- The ensemble improved F1-scores for all classes compared to the single decision tree.
- The theory that bagging and feature randomization enhance model stability was confirmed by the practical results.

Therefore, for medium-sized tabular data tasks where high accuracy is crucial without extensive hyperparameter tuning, Random Forest remains a preferable choice over simple linear models or single trees.

References

- E. Usupova and A. Khan, "Optimizing ML Training with Perturbed Equations," 2025 6th International Conference on Problems of Cybernetics and Informatics (PCI), Baku, Azerbaijan, 2025, pp. 1-6, doi: 10.1109/PCI66488.2025.11219819.
- S. Rakimbekuulu, K. Shambetaliev, G. Esenalieva and A. Khan, "Code Generation for Ablation Technique," 2024 IEEE East-West Design & Test Symposium (EWDTS), Yerevan, Armenia, 2024, pp. 1-7, doi: 10.1109/EWDTS63723.2024.10873640.