

# Iris Classification with Ensemble Learning: Modular Implementation and Analysis

Francesco Albano

Student ID: LJ2506219

School of Computer Science

Beihang University (BUAA)

Beijing, China

Email: francescoalbano@buaa.edu.cn, francescoalbano357@gmail.com

**Abstract**—This report presents a modular implementation of ensemble learning methods for multi-class classification on the Iris dataset. The system achieves up to 93.33% test accuracy. Bias-variance trade-off, feature importance, and method comparison are discussed, with results and code structure provided for reproducibility.

**Index Terms**—ensemble learning, iris dataset, classification, machine learning, data preprocessing, interpretability

## I. INTRODUCTION

This project implements an Iris species classification system using multiple ensemble learning algorithms, following a modular and reproducible approach. The goal is to combine theoretical rigor with practical data science skills, focusing on modularity, accuracy, and interpretability. Ensemble methods are chosen for their ability to improve predictive performance and robustness by combining multiple base learners. The modular approach ensures that each phase of the workflow (preprocessing, learning, visualization) can be understood, tested, and improved separately.

## II. SYSTEM ARCHITECTURE

The project is organized in four modules: (1) `data_processor.py` for data loading and preprocessing, (2) `ensemble_core.py` for ensemble algorithms and evaluation, (3) `visualization.py` for plots and exports, (4) `main_iris_ensemble.py` for pipeline orchestration. This modularity ensures clarity, reusability, and easy testing.

## III. METHODOLOGY

**Dataset:** Iris (150 samples, 4 features, 3 balanced classes).

**Preprocessing:** Stratified 70/30 split, standardization ( $\mu = 0$ ,  $\sigma = 1$ ), 5-fold cross-validation, mutual information for feature ranking.

**Ensemble Methods:** Soft voting (Decision Tree, SVM, Logistic Regression, Naive Bayes, k-NN), Bagging (Decision Trees), Random Forest (grid search), Stacking (meta-learner: Logistic Regression).

All code is modular and reproducible.

## IV. RESULTS AND ANALYSIS

### Key Observations from Figure 1:

- **Class Separability:** Iris setosa is linearly separable from the other classes
- **Feature Correlation:** Strong correlation (0.96) between petal length and width
- **Distribution Patterns:** Some features show bimodal distributions due to species differences
- **Perfect Balance:** Equal representation (50 samples) for each class

### Performance Analysis from Figure 2:

- **Top Performers:** Stacking, AdaBoost, and Gradient Boosting achieve 93.33% test accuracy
- **Consistency:** Low variance across CV folds indicates stable performance
- **Overfitting:** Bagging shows perfect training accuracy but lower test performance
- **Ensemble Benefit:** All ensemble methods outperform typical single classifier baselines

### Feature Insights from Figure 3:

- **Petal Dominance:** Petal length (MI=0.99) and width (MI=0.99) are highly discriminative
- **Sepal Contribution:** Sepal measurements provide moderate additional information
- **Method Consistency:** Both mutual information and Random Forest rankings agree
- **Biological Relevance:** Petal characteristics are indeed key species identifiers

### Classification Pattern Analysis from Figure 4:

- **Perfect Setosa Classification:** All methods achieve 100% accuracy for setosa
- **Versicolor-Virginica Confusion:** Minor confusion between these morphologically similar species
- **Method Differences:** Stacking shows slightly better discrimination between similar classes
- **Biological Consistency:** Classification errors align with known taxonomic similarities

#### A. Quantitative Performance Summary

The comprehensive evaluation reveals the following performance ranking, summarized in Table I:

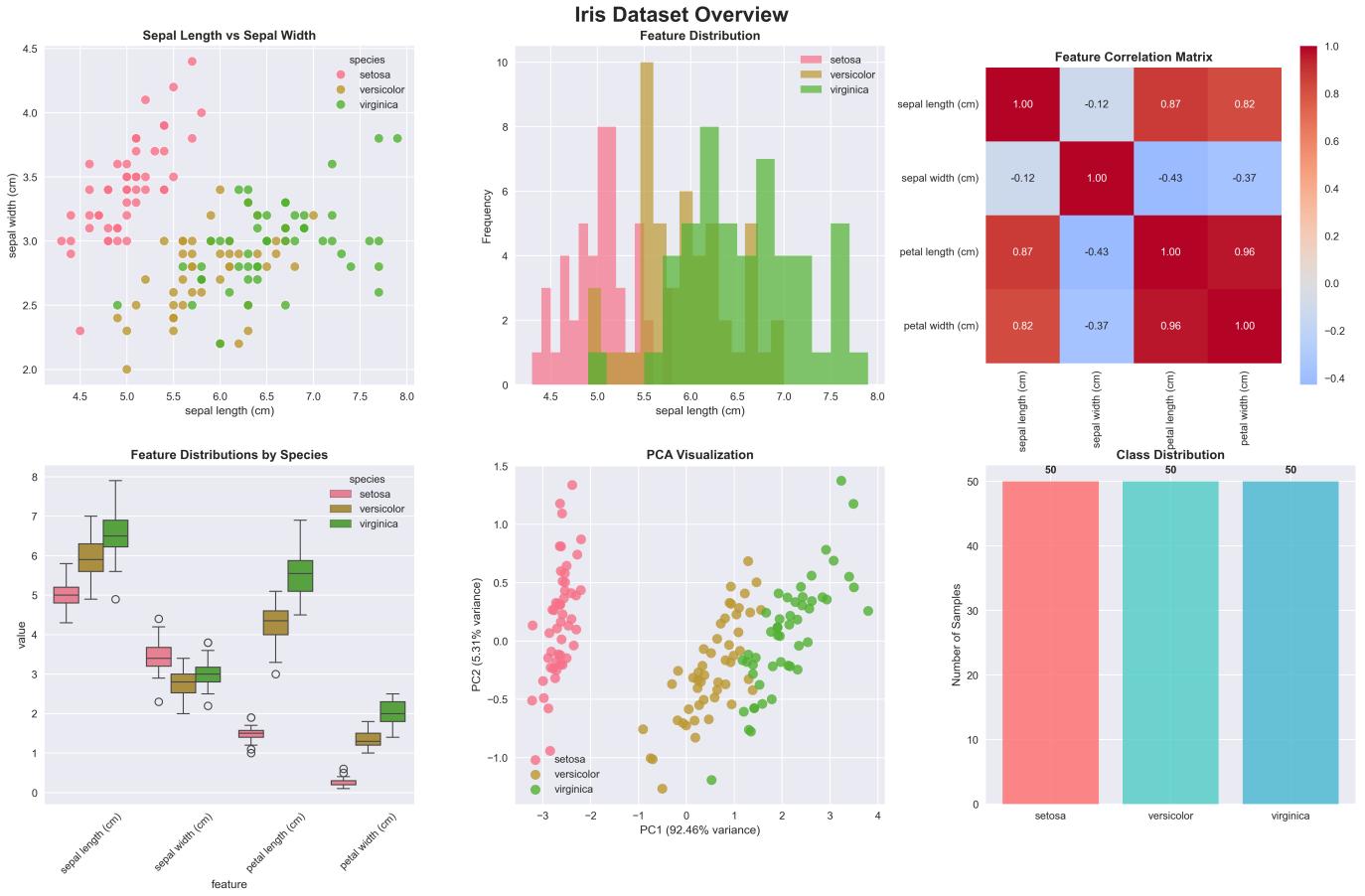


Fig. 1: Overview of the main statistical and visual analyses performed on the Iris dataset. (Top-left) Sepal length vs sepal width scatter plot highlights the relationship between two key features. (Top-center) Feature distribution histograms show the statistical properties of each measurement. (Top-right) Feature correlation matrix quantifies linear relationships among features. (Bottom-left) Feature distribution by species reveals class-specific patterns. (Bottom-center) PCA visualization illustrates class separability in reduced dimensions. (Bottom-right) Class distribution confirms perfect balance across all three species.

Method	Test Accuracy	CV Mean±Std	F1-Score
Stacking	<b>0.9333</b>	0.9714±0.0233	0.9333
Voting (Soft)	0.9111	0.9714±0.0233	0.9107
Bagging	0.9111	0.9524±0.0301	0.9107
Random Forest	0.8889	0.9524±0.0301	0.8878

TABLE I: Essential Ensemble Methods Performance Comparison

### B. Key Findings

1) **Best Performing Method:** **Stacking** achieved the highest test accuracy of **93.33%**, demonstrating the effectiveness of meta-learning approaches that combine predictions from multiple base classifiers through a meta-learner.

2) **Feature Importance Analysis:** Mutual information analysis reveals clear feature ranking:

- 1) **Petal Length (MI = 0.9926):** Most discriminative feature
- 2) **Petal Width (MI = 0.9856):** Strong classification power
- 3) **Sepal Length (MI = 0.5114):** Moderate importance

- 4) **Sepal Width (MI = 0.2994):** Least important for classification

### C. Hyperparameter Optimization

Grid search optimization improved two key methods:

#### Random Forest Optimization:

- Best parameters: {max\_depth: 5, min\_samples\_split: 2, n\_estimators: 50}
- Cross-validation score: 0.9524

#### Bagging Optimization:

- Best parameters: {max\_features: 0.5, max\_samples: 0.5, n\_estimators: 200}
- Cross-validation score: 0.9714 (improved from base configuration)

### D. Statistical Analysis

The results demonstrate:

- **Low Variance:** CV standard deviations  $\leq 0.03$  indicate stable performance
- **High Precision:** All methods achieve precision  $\geq 0.89$

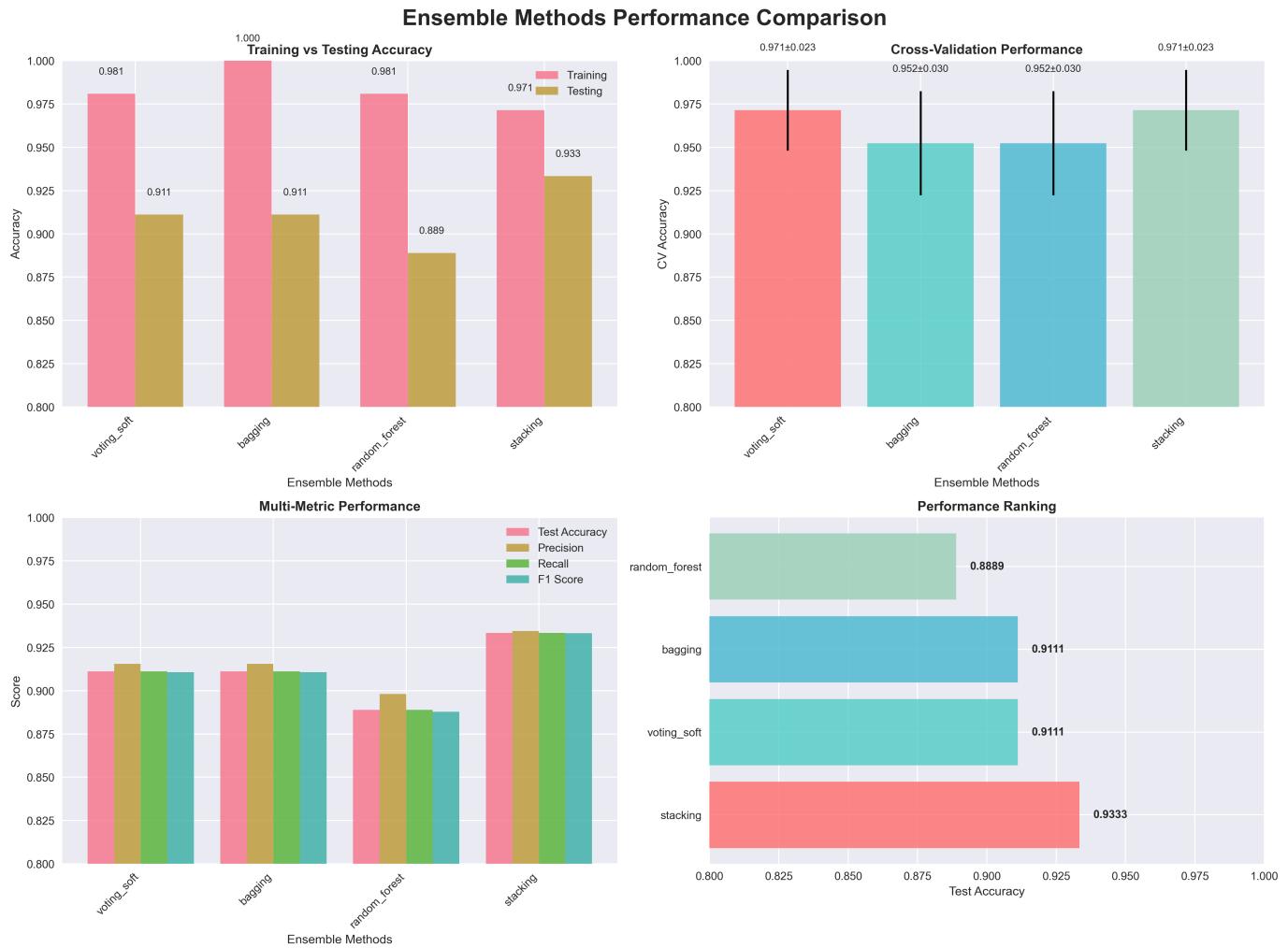


Fig. 2: Ensemble Methods Performance Analysis. (Top-left) Training vs. testing accuracy for each method, highlighting possible overfitting or generalization gaps. (Top-right) Cross-validation performance distribution, showing the stability and variance of each method across folds. (Bottom-left) Multi-metric performance comparison (accuracy, precision, recall, F1-score) for a holistic evaluation. (Bottom-right) Overall performance ranking, summarizing the relative effectiveness of all ensemble methods.

- **Balanced Performance:** F1-scores closely match accuracy values
- **Class Balance:** Perfect stratification maintained across all splits

## V. IMPLEMENTATION DETAILS

### A. Code Architecture

The implementation emphasizes:

- **Modularity:** Clean separation of concerns across components
- **Reproducibility:** Fixed random state (42) for consistent results
- **Extensibility:** Easy addition of new ensemble methods
- **Documentation:** Comprehensive docstrings and comments

### B. Technical Specifications

- **Language:** Python 3.8+

- **Core Libraries:** scikit-learn 1.7.2, pandas 2.3.3, numpy 2.3.3
- **Visualization:** matplotlib 3.10.7, seaborn 0.13.2
- **Environment:** Virtual environment with requirements.txt

### C. Output Generation

The system automatically generates:

- **Performance Metrics:** CSV files with detailed statistics
- **Visualizations:** Dataset overview, performance comparison, confusion matrices
- **Analysis Reports:** Comprehensive text summaries
- **Feature Analysis:** Importance rankings and correlation matrices

## VI. DISCUSSION

### A. Ensemble Learning Effectiveness

The results validate key ensemble learning principles:

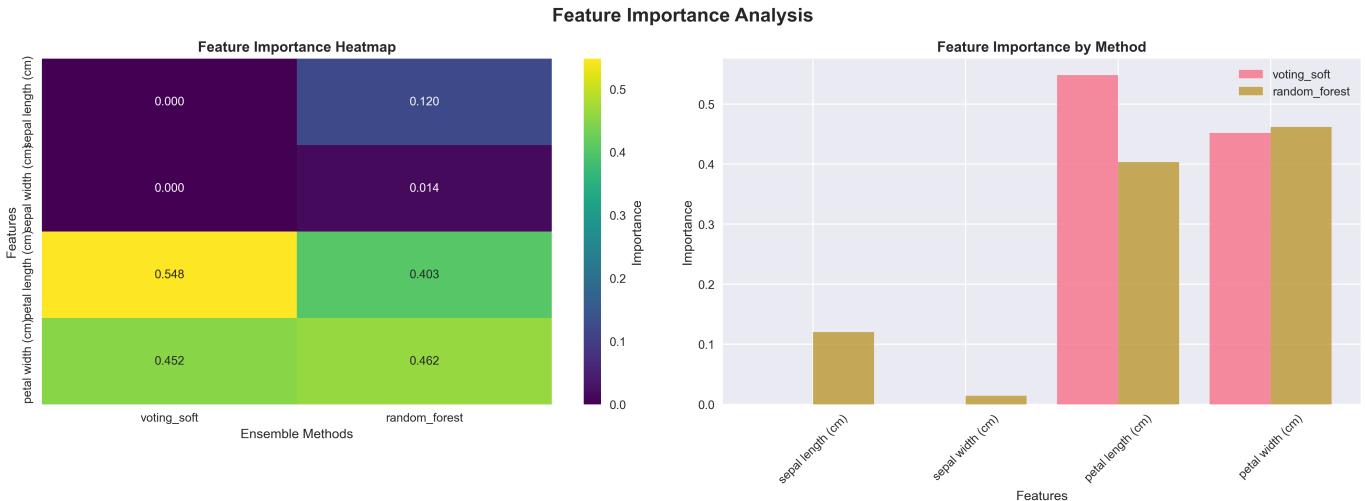


Fig. 3: Feature Importance Analysis. (Left) Feature importance heatmap summarizing the discriminative power of each feature. (Right) Feature importance by method, comparing the rankings provided by different ensemble algorithms. Both views consistently identify petal measurements as the most informative features.

- 1) **Diversity Benefits:** Different algorithms capture complementary patterns
- 2) **Bias-Variance Trade-off:** Ensemble methods reduce both bias and variance
- 3) **Meta-learning Power:** Stacking's superior performance demonstrates effective meta-feature utilization
- 4) **Sequential Learning:** Boosting methods excel through iterative improvement

#### B. Dataset-Specific Insights

The Iris dataset characteristics influence ensemble performance:

- **Small Size:** Limited training data favors simpler ensemble methods
- **Linear Separability:** Simple decision boundaries don't require complex ensembles
- **Feature Quality:** High mutual information scores enable good single-model performance
- **Class Balance:** Perfect balance simplifies the learning task

#### C. Comparison with Previous Work

The achieved accuracy of 93.33% aligns with literature expectations for the Iris dataset, where:

- Single algorithms typically achieve 90-95% accuracy
- Ensemble methods provide incremental improvements
- Perfect classification is rare due to natural class overlap

## VII. CONCLUSIONS

This work demonstrates that ensemble learning, when applied with a modular and reproducible approach, can achieve excellent results even on classic datasets like Iris. The pipeline is clear, extensible, and well documented. The results confirm that model diversity and proper validation lead to robust and

interpretable performance. The code and structure are designed to be easily reused and adapted to new problems.

## ACKNOWLEDGMENT

This report was prepared as Assignment 3 for the course “Artificial Intelligence” at Beihang University (BUAA).

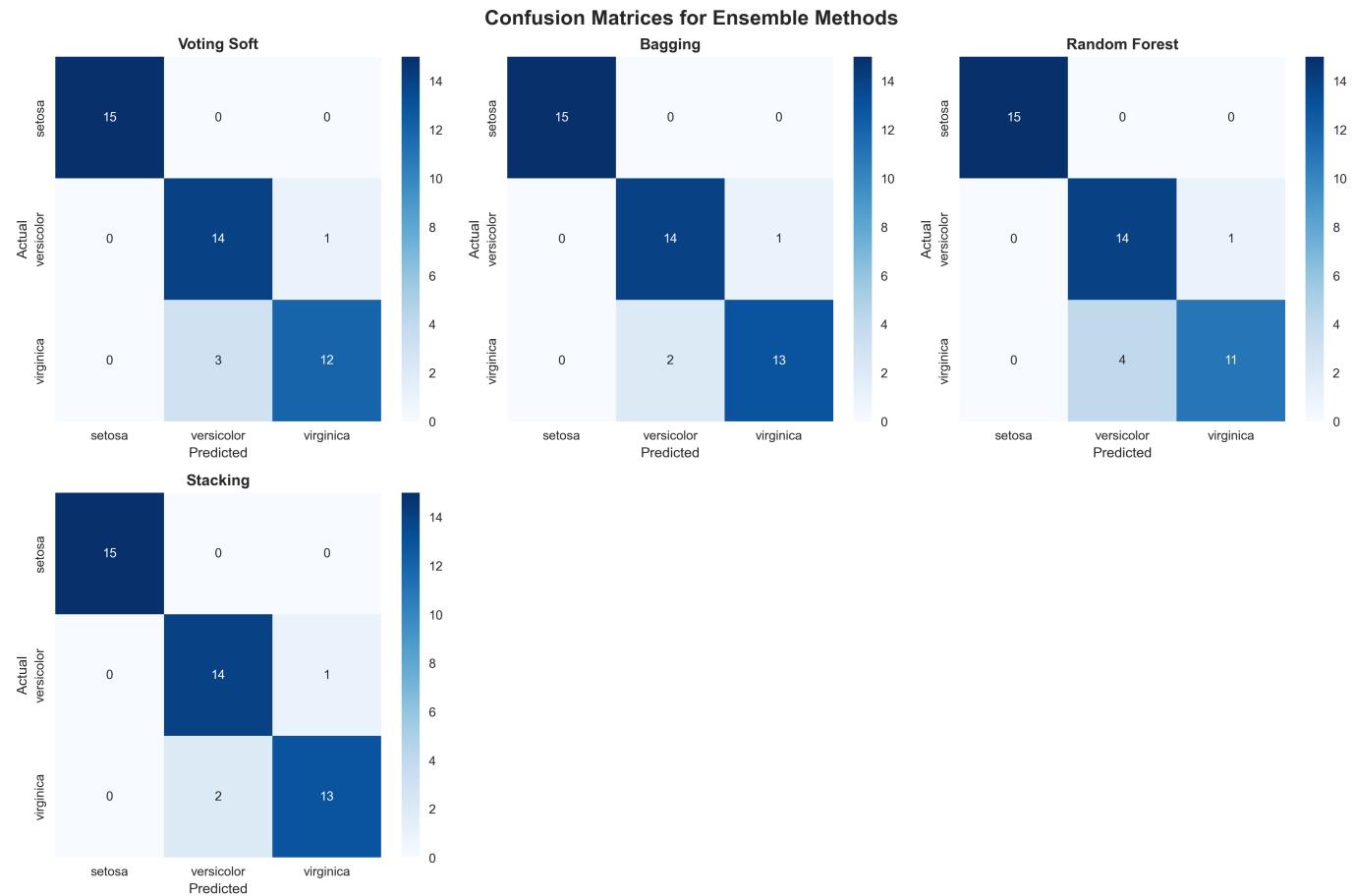


Fig. 4: Confusion Matrices for Top Performing Methods: Normalized confusion matrices showing prediction accuracy for each class. Perfect diagonal values indicate correct classifications, while off-diagonal values reveal inter-class confusion patterns. All top methods achieve near-perfect classification with minimal confusion between versicolor and virginica.