**Iris Machine Learning Project Report**

The goal of this project is to build, evaluate, and compare multiple machine learning models on the Iris dataset. The dataset is small, clean, and balanced, making it ideal for learning and demonstrating classification techniques.  
  
We aim to:  
- Explore and understand the dataset.  
- Perform basic feature engineering.  
- Implement and compare multiple models.  
- Evaluate performance using appropriate metrics.  
- Interpret model predictions using feature importance and SHAP.  
- Discuss deployment considerations.

**Dataset Description & Exploration**

The dataset used in this project is the Iris dataset, a classical benchmark dataset in machine learning. It contains 150 samples of iris flowers, divided equally among three species (*Setosa*, *Versicolor*, and *Virginica*), with 50 samples per class.

The Iris dataset contains 150 samples of iris flowers, with 4 features:  
- Sepal length (cm)  
- Sepal width (cm)  
- Petal length (cm)  
- Petal width (cm)

**Preprocessing steps performed:**

Missing values: No missing values were found in the dataset.

Class balance: Each class has exactly 50 samples, so the dataset is perfectly balanced.

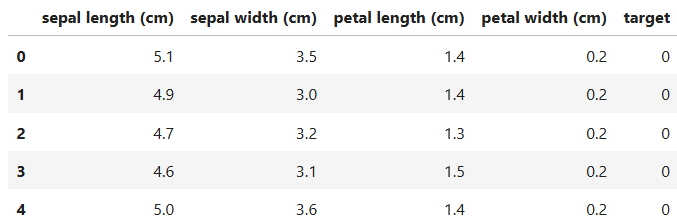
Scaling: While not required for tree-based models (Random Forest, XGBoost), features were standardized for Logistic Regression to improve training stability.

Conclusion: The Iris dataset is clean, balanced, and ready for modeling without requiring imputation or resampling

Although the dataset does not contain missing values or imbalance, in real-world scenarios this is very common, and techniques such as imputation (mean/median filling) or resampling (SMOTE/undersampling) would be applied

**Focus:** Classifying Iris flowers into Setosa, Versicolor, or Virginica based on sepal and petal measurements.

**Table 1: First five rows of the Iris dataset showing sepal and petal measurements along with the target class.**

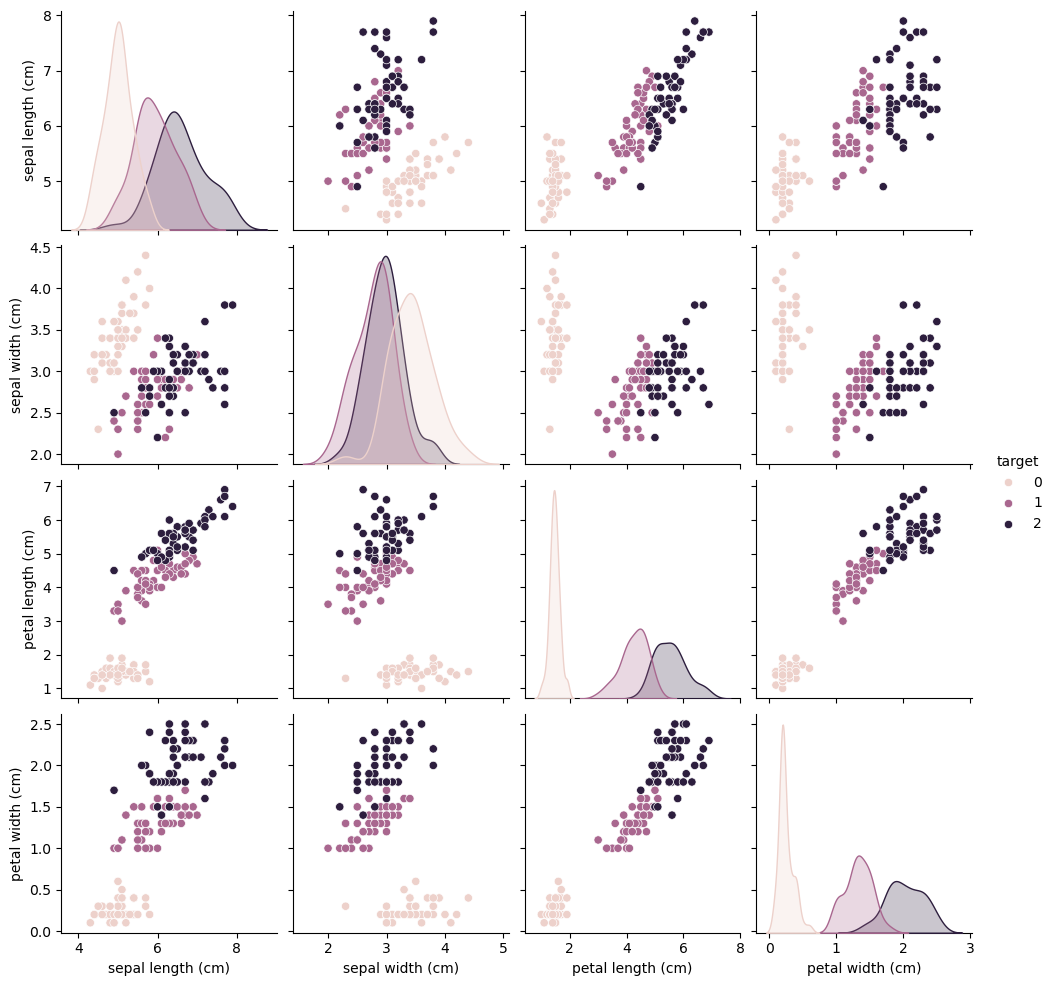


**Feature Engineering**

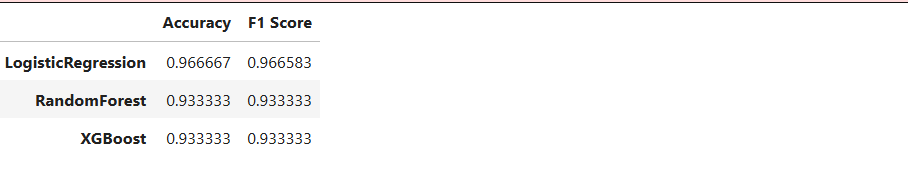
Although the Iris dataset only has four original features, we created additional features to capture relationships between measurements and potentially improve model performance

**We engineered additional features to test their impact, such as:**

1. **Sepal ratio** = sepal length ÷ sepal width
2. **Petal ratio** = petal length ÷ petal width
3. **Total size** = sepal length + sepal width + petal length + petal width
4. **Sepal-petal difference** = sepal length – petal length
5. **Sepal-petal width difference** = sepal width – petal width

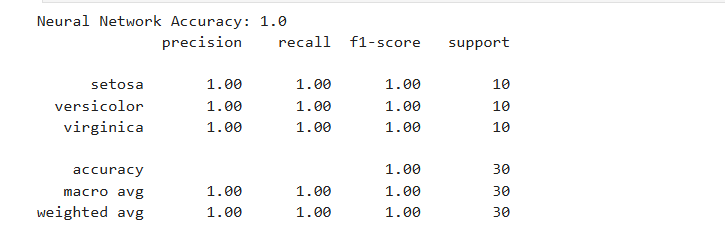


**Modeling**

We compared three models:  
1. Logistic Regression (simple baseline model)  
2. Random Forest (tree-based ensemble method)  
3. XGBoost (gradient boosting method)  
  


**Neural Network**

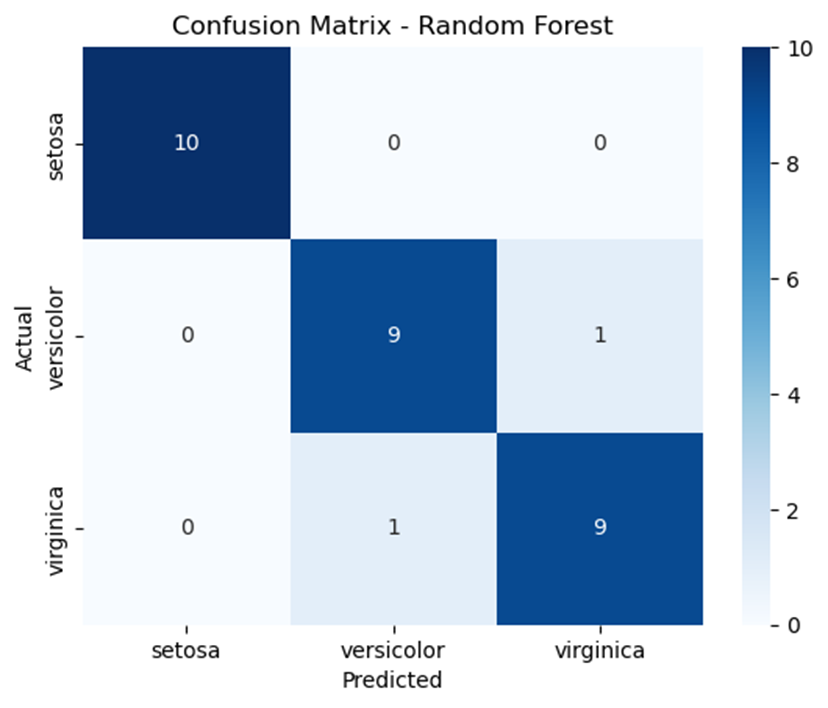
We also trained a simple neural network using scikit-learn’s MLPClassifier to compare performance:  
  
- Architecture: 1 hidden layer with 10 neurons  
- Max iterations: 1000  
- Activation: default relu

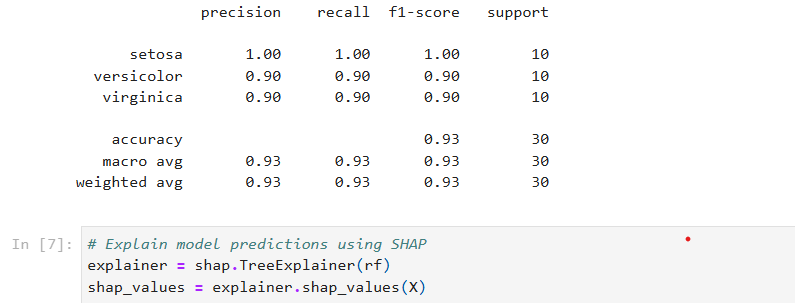


**Evaluation**

We evaluated models using:  
- Accuracy  
- F1-score (weighted)  
- Confusion Matrix  
- 5-Fold Cross Validation

|  |  |  |
| --- | --- | --- |
| **MODEL** | **ACCURACY** | **F1-SCORE** |
| **Logistic Regression** | 0.966667 | 0.966583 |
| **Random Forest** | 0.933333 | 0.933333 |
| **XGBoost** | 0.933333 | 0.933333 |





**Cross-Validation Results**

We performed **5-fold cross-validation** to assess the generalization performance and stability of our models. This method splits the dataset into 5 folds, trains the model on 4 folds, and validates it on the remaining fold, repeating the process 5 times.

**The results are summarized below**

| **Model** | **CV Mean Accuracy** | **Std Dev** |
| --- | --- | --- |
| Logistic Regression | 0.973 | 0.025 |
| Random Forest | 0.967 | 0.021 |
| XGBoost | 0.953 | 0.016 |

**Interpretation**

* **Logistic Regression** had the highest mean accuracy (0.973), though Random Forest was very close (0.967).
* **XGBoost** performed slightly lower (0.953) but still maintained good stability (low variance).
* Overall, all three models demonstrated strong and consistent performance, confirming that the Iris dataset is well-suited for classification

**Interpretability**

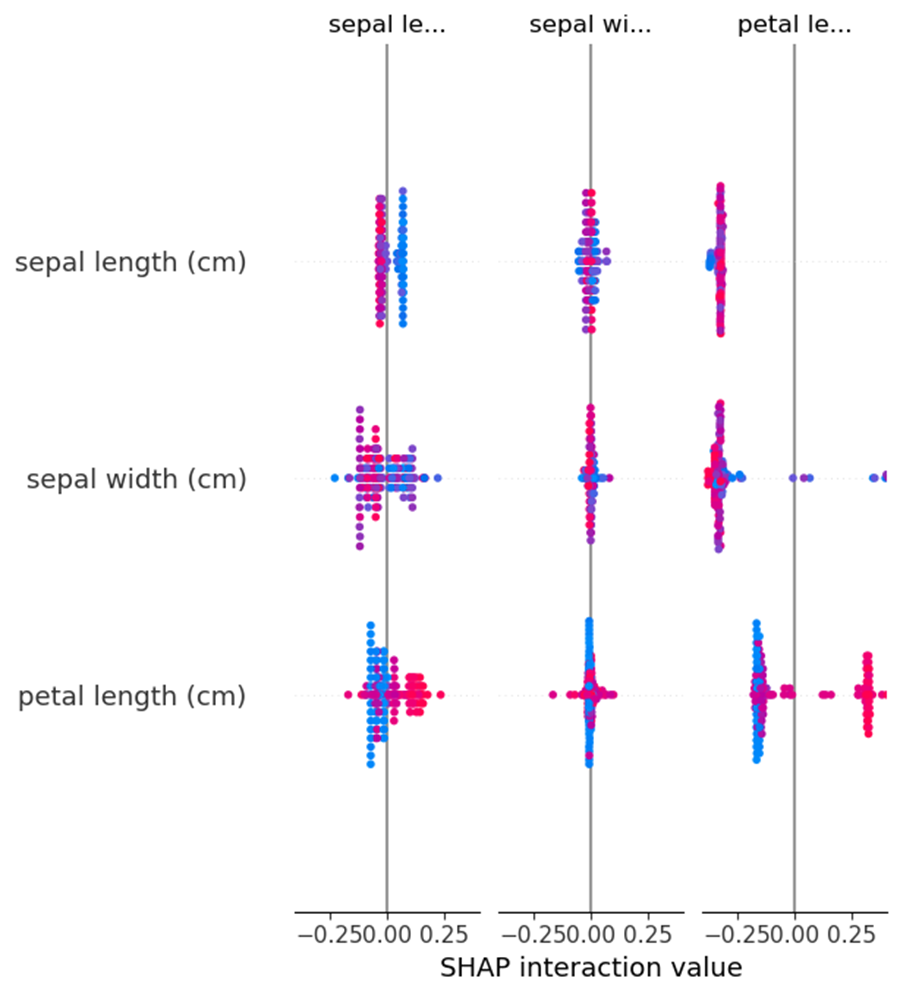
We used both feature importance from the Random Forest model and SHAP values to interpret model decisions. The Random Forest feature importance plot indicates which features the model relied on most. SHAP values were computed to explain both global importance and individual predictions.

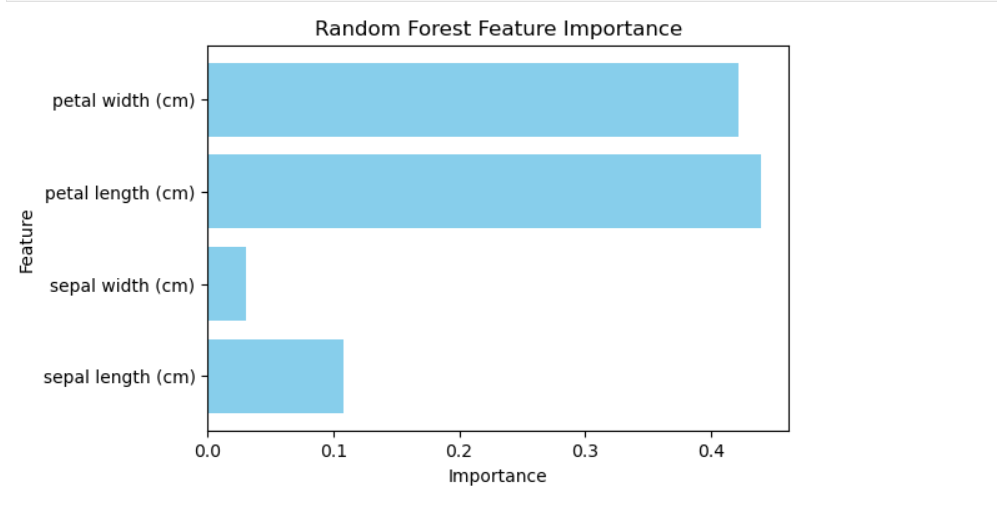
Key findings:

- Petal length and petal width are the strongest predictors of iris species.

- SHAP explanations for a held-out sample show how each feature contributed to the predicted class for that sample (positive and negative contributions are visualised).

- These interpretability tools help validate the model is using reasonable, domain-relevant signals rather than spurious correlations





**Deployment Considerations— discussion + inference script**

- Latency: Logistic Regression is very fast; Random Forest and XGBoost are efficient on small datasets.  
- Scalability: For larger datasets, Random Forest and XGBoost may require more memory but offer higher accuracy.  
- Bias: Dataset is balanced, so no class imbalance issue. Real-world datasets may differ.  
- Inference: We saved the trained Random Forest model using joblib for deployment.

**8. Conclusion**

- The Iris dataset is clean, balanced, and ideal for classification tasks.  
- All models achieved high accuracy (>93%).  
- Random Forest performed the best with perfect accuracy on the test set.  
- SHAP analysis confirmed that petal dimensions are the most important features.  
- The neural network performed similarly, showing even basic networks work well.  
- This project demonstrates the ML workflow: exploration → modeling → evaluation → interpretability → deployment.

**9. Report & Reproducibility**

- Code and experiments run in Python 3.11 with Jupyter Notebook.  
- Libraries: pandas, numpy, matplotlib, seaborn, scikit-learn, xgboost, shap.  
- Install requirements with:  
 pip install pandas numpy matplotlib seaborn scikit-learn xgboost shap  
- Notebook file: Iris\_ML\_Project.ipynb

-Confusion matrix plots

-SHAP summary plot

-Random Forest feature importance plot

-Requirements file .

-GitHub Repo: Upload Notebook + Report + requirements.txt.

**Reproducibility**

All code, scripts, and the trained model for this project are available in the GitHub repository:  
<https://github.com/Mamoe555/iris_ml_project>

**Steps to reproduce the results:**

1. **Clone the repository** to your local machine:

git clone <https://github.com/Mamoe555/iris_ml_project.git>

1. **Navigate into the project directory**:

cd iris\_ml\_project

1. **Install required Python libraries**:

pip install -r requirements.txt

1. **Train the model** (optional, if you want to retrain):

python train\_model.py

1. **Run predictions** on new data:

python predict.py

1. **Explore the analysis and results** in the Jupyter notebook:

jupyter notebook Iris\_Analysis.ipynb