Iris Flower Classification Report

Introduction

In this project, I explored the Iris dataset, a well-known dataset in the realm of machine learning. It is often used for testing and benchmarking classification algorithms due to its simplicity and the clear distinction between the classes. The dataset comprises data on three species of Iris flowers:

- *Setosa*
- *Versicolor*
- *Virginica*

Each sample in the dataset includes measurements of four features:

- *Sepal Length (in cm)*
- *Sepal Width (in cm)*
- *Petal Length (in cm)*
- *Petal Width (in cm)*

The dataset contains a total of 150 samples, with 50 samples for each species. I set out to evaluate and compare various classification models to see how well they perform on this dataset.

Descriptive Statistics

To understand the dataset better, I calculated the descriptive statistics for each feature:

Sepal Length:

- Mean: 5.843 cm
- Standard Deviation: 0.828 cm
- Range: 4.3 cm to 7.9 cm

Sepal Width:

- Mean: 3.057 cm
- Standard Deviation: 0.436 cm
- Range: 2.0 cm to 4.4 cm

Petal Length:

- Mean: 3.758 cm
- Standard Deviation: 1.765 cm
- Range: 1.0 cm to 6.9 cm

Petal Width:

- Mean: 1.199 cm
- Standard Deviation: 0.762 cm
- Range: 0.1 cm to 2.5 cm

These statistics gave me insight into the distribution of measurements across different Iris species.

Model Performance

1. Random Forest Classifier

When I applied the Random Forest Classifier, I was pleased to find that it achieved ***100%*** accuracy. Here's a breakdown of its performance:

- Setosa: Precision: 1.00, Recall: 1.00, F1-Score: 1.00
- Versicolor: Precision: 1.00, Recall: 1.00, F1-Score: 1.00
- Virginica: Precision: 1.00, Recall: 1.00, F1-Score: 1.00

Analysis:

The Random Forest model delivered perfect precision and recall for all species, indicating that it classified every instance correctly. With an F1-score of 1.00 for each species, it demonstrated a perfect balance between precision and recall. This robustness is due to the ensemble method, which combines multiple decision trees.

Insights:

The Random Forest model's performance was exceptional, handling the dataset without any misclassifications. It proved to be highly effective for this classification task.

2. K-Nearest Neighbors (KNN)

Similarly, the K-Nearest Neighbors (KNN) algorithm also achieved ***100%*** accuracy. The classification report was:

- Setosa: Precision: 1.00, Recall: 1.00, F1-Score: 1.00
- Versicolor: Precision: 1.00, Recall: 1.00, F1-Score: 1.00
- Virginica: Precision: 1.00, Recall: 1.00, F1-Score: 1.00

Analysis:

KNN's performance was flawless, with perfect precision and recall across all species. The F1-scores matched those of the Random Forest model, indicating that KNN also effectively handled the classification task.

Insights:

KNN performed excellently, likely due to the clear separation between species in the feature space. The algorithm's simplicity and effectiveness were evident.

3. Support Vector Machine (SVM)

The Support Vector Machine (SVM) model achieved *96.67%* accuracy. Here's how it performed:

- Setosa: Precision: 1.00, Recall: 1.00, F1-Score: 1.00
- Versicolor: Precision: 1.00, Recall: 0.89, F1-Score: 0.94
- Virginica: Precision: 0.92, Recall: 1.00, F1-Score: 0.96

Analysis:

SVM performed well, though not as perfectly as the Random Forest and KNN models. It had high precision but slightly lower recall for Versicolor and Virginica, leading to slightly lower F1-scores.

Insights:

While SVM showed strong performance, there was a minor trade-off in precision and recall, especially for Versicolor. This suggests that tuning hyperparameters or using different kernels could improve performance.

4. Tuned Random Forest

I also experimented with a tuned version of the Random Forest model, which maintained the *100%* accuracy. The classification metrics were:

- Setosa: Precision: 1.00, Recall: 1.00, F1-Score: 1.00
- Versicolor: Precision: 1.00, Recall: 1.00, F1-Score: 1.00
- Virginica: Precision: 1.00, Recall: 1.00, F1-Score: 1.00

Analysis:

The tuned Random Forest model performed identically to the untuned version, indicating that the initial settings were already optimal.

Insights:

The tuned Random Forest model confirmed the effectiveness of the initial configuration, showcasing its reliability and robustness.

Neural Network Training

For the Neural Network, I set up a training process with the following details:

- Epochs: 100
- Initial Accuracy: ***24.17%***
- Mid Training Accuracy (Epoch 50): ***97.50%***
- Final Accuracy (Epoch 100): ***97.50%***

Loss Details:

- Initial Loss: 1.1405
- Loss at Epoch 50: 0.0803
- Final Loss (Epoch 100): 0.0770

Analysis:

The neural network started with low accuracy but improved significantly over the epochs. By epoch 50, the accuracy stabilized at 97.50%, and the loss decreased steadily, indicating effective learning and convergence.

Insights:

The neural network demonstrated strong learning capabilities and high accuracy. However, it required more epochs compared to simpler models like Random Forest and KNN, highlighting the complexity involved in training neural networks.

Conclusion

- Random Forest and KNN: Both models achieved perfect accuracy and performance, with flawless precision, recall, and F1-scores. They were highly effective for classifying the Iris dataset.

- SVM: Although it performed slightly less well with 96.67% accuracy, it still provided robust results, with some trade-offs in precision and recall.

- Neural Network: The neural network showed impressive improvement over epochs, achieving high accuracy of 97.50%. However, it involved more complexity and training time compared to the other models.

In summary, the Iris dataset proved to be an excellent benchmark for evaluating classification algorithms. Random Forest and KNN excelled in performance, while SVM and Neural Networks offered valuable alternatives with their own strengths.