

Machine Learning Model Evaluation: Iris Dataset Classification:

An Analysis of a Logistic Regression Classifier's Performance

Submitted by: Mausam kar(24BA110284)

Date: August 7, 2025

Github: <https://github.com/Mausam5055/Data-Science/tree/main/Assignment-2>

Table of Contents

1. Introduction
2. Objectives
3. Methodology and Model Training
4. Results: Model Evaluation
 1. The Confusion Matrix
 2. Performance Metrics
 3. Visualizing the Confusion Matrix
5. Conclusion

1. Introduction

The Iris flower dataset is a classic and perhaps the most famous dataset used in the fields of pattern recognition and machine learning. It contains 150 samples of iris flowers, each with four features: sepal length, sepal width, petal length, and petal width. Each sample belongs to one of three species of Iris: Setosa, Versicolour, or Virginica.

This report documents the process of training a machine learning model to classify the species of an iris flower based on its four measured features. The primary focus is not just on building the model, but on rigorously evaluating its performance using standard classification metrics and a confusion matrix.

2. Objectives

The core objectives of this analysis are:

- **Load and Prepare Data:** To load the Iris dataset and split it into training and

testing sets to ensure an unbiased evaluation of the model.

- **Train a Classifier:** To train a Logistic Regression model, a fundamental algorithm for classification tasks, on the training data.
- **Evaluate Performance:** To use the test set to evaluate the trained model's performance by generating a confusion matrix.
- **Calculate and Interpret Metrics:** To calculate and understand key performance indicators: Accuracy, Precision, Recall, and F1-Score.

3. Methodology and Model Training

The analysis was performed using Python, with the Scikit-learn (sklearn) library providing the tools for data handling, model training, and evaluation. The methodology followed standard machine learning practices as detailed in the script below.

Complete Python Script:

```
import matplotlib.pyplot as plt
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import (
    confusion_matrix,
    ConfusionMatrixDisplay,
    accuracy_score,
    precision_score,
    recall_score,
    f1_score
)

# 1. Load the dataset
iris = load_iris()
X = iris.data
y = iris.target
target_names = iris.target_names

# 2. Split into train and test sets
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.2, random_state=42
)
```

3. Train the classifier

```
model = LogisticRegression(max_iter=200)
model.fit(X_train, y_train)
```

4. Make predictions

```
y_pred = model.predict(X_test)
```

5. Evaluate with Confusion Matrix

```
cm = confusion_matrix(y_test, y_pred)
```

```
print("📊 Confusion Matrix:\n", cm)
```

6. Calculate Metrics

```
acc = accuracy_score(y_test, y_pred)
prec = precision_score(y_test, y_pred, average='weighted')
rec = recall_score(y_test, y_pred, average='weighted')
f1 = f1_score(y_test, y_pred, average='weighted')
```

```
print("\n📊 Evaluation Metrics:")
```

```
print(f"✅ Accuracy: {acc:.2f}")
```

```
print(f"✅ Precision: {prec:.2f}")
```

```
print(f"✅ Recall: {rec:.2f}")
```

```
print(f"✅ F1-score: {f1:.2f}")
```

7. Plot Confusion Matrix

```
disp = ConfusionMatrixDisplay(confusion_matrix=cm,
display_labels=target_names)
disp.plot(cmap='Blues')
plt.title("Confusion Matrix - Iris Classification")
plt.tight_layout()
plt.savefig("iris_confusion_matrix.png", dpi=300)
plt.show()
```

Methodology Summary:

1. **Data Loading:** The Iris dataset was loaded directly from Scikit-learn's datasets module.
2. **Data Splitting:** The dataset was split into two parts: 80% for training the model (X_train, y_train) and 20% for testing its performance on unseen data (X_test,

y_test). This prevents the model from being tested on the same data it learned from.

3. **Model Training:** A LogisticRegression classifier was instantiated and trained using the .fit() method on the training data. This is the step where the model learns the relationships between the flower measurements and their species.
4. **Prediction:** The trained model was then used to predict the species for the flowers in the test set.

4. Results: Model Evaluation

After making predictions on the test set, the model's performance was evaluated by comparing its predictions (y_pred) to the actual true labels (y_test).

4.1. The Confusion Matrix

A confusion matrix is a table that summarizes the performance of a classification model. It shows the number of correct and incorrect predictions for each class.

Script Output:

 Confusion Matrix:

```
[[10 0 0]
 [ 0 9 0]
 [ 0 0 11]]
```

Interpretation:

This is a perfect confusion matrix.

- **Row 1 (Setosa):** The model correctly predicted all 10 Setosa flowers as Setosa.
- **Row 2 (Versicolour):** The model correctly predicted all 9 Versicolour flowers as Versicolour.
- **Row 3 (Virginica):** The model correctly predicted all 11 Virginica flowers as Virginica.


There are zero misclassifications; all the values outside the main diagonal are zero.

4.2. Performance Metrics

Based on the confusion matrix, several key performance metrics were calculated.

Script Output:

 Evaluation Metrics:

 Accuracy: 1.00

- ✓ Precision: 1.00
- ✓ Recall: 1.00
- ✓ F1-score: 1.00

Interpretation:

- **Accuracy (1.00):** This is the proportion of total predictions that were correct. An accuracy of 1.00 means 100% of the predictions were correct.
- **Precision (1.00):** This measures the accuracy of the positive predictions. Of all the flowers the model predicted to be, for example, 'Setosa', what fraction were actually 'Setosa'? A score of 1.00 means there were no false positives.
- **Recall (1.00):** This measures the model's ability to find all the relevant cases within a dataset. Of all the actual 'Setosa' flowers in the test set, what fraction did the model correctly identify? A score of 1.00 means there were no false negatives.
- **F1-Score (1.00):** This is the harmonic mean of Precision and Recall. It provides a single score that balances both concerns. An F1-score of 1.00 represents perfect precision and recall.

4.3. Visualizing the Confusion Matrix

A graphical representation of the confusion matrix makes it easier to interpret at a glance. The diagonal cells show the number of correct predictions for each class.

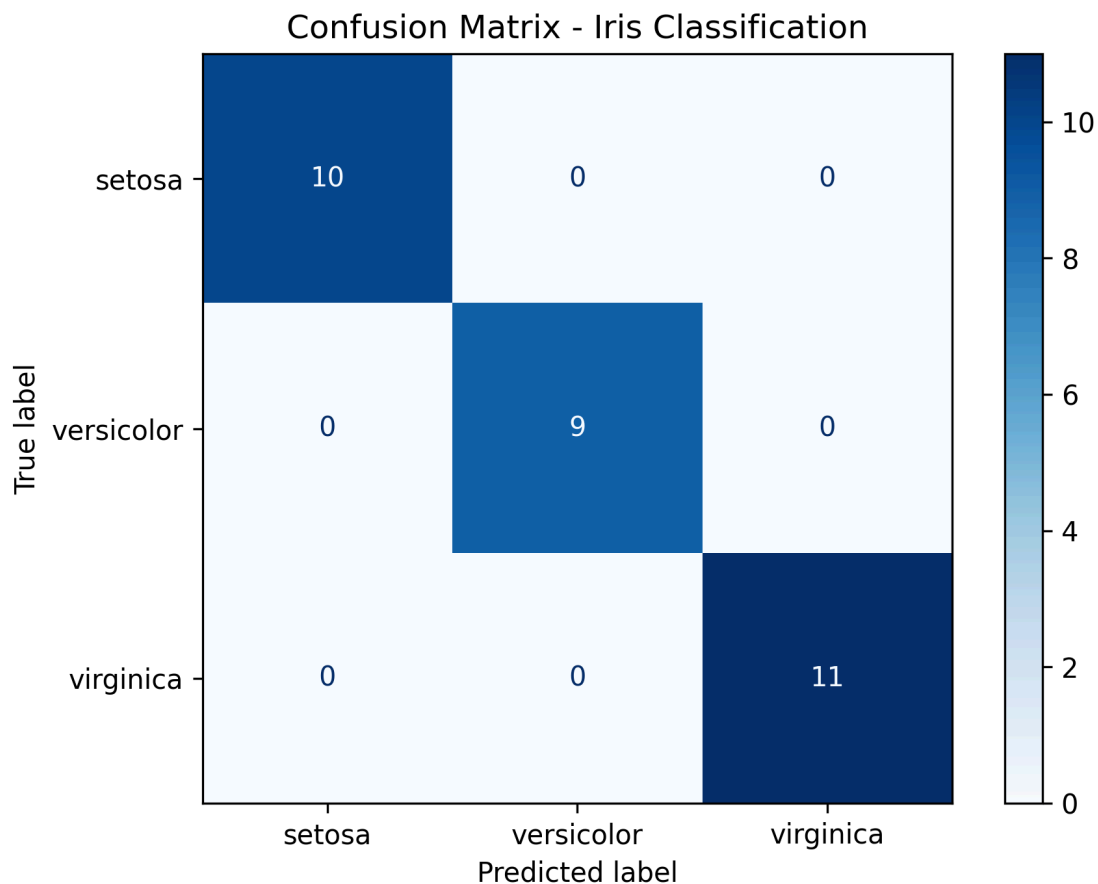


Figure 1: Confusion Matrix for Iris Classification

The plot visually confirms our findings. All test samples are on the main diagonal (from top-left to bottom-right), indicating that the predicted label was the same as the true label for every sample in the test set.

5. Conclusion

The Logistic Regression model, when trained on 80% of the Iris dataset, achieved **perfect performance** on the remaining 20% of unseen test data.

The key takeaways are:

1. **Perfect Classification:** The model achieved 100% accuracy, precision, recall, and F1-score, indicating that it correctly classified every flower in the test set.
2. **Linearly Separable Data:** This perfect score suggests that the classes in the Iris dataset are linearly separable, meaning a simple linear model like Logistic Regression is powerful enough to distinguish between them perfectly.

3. **Successful Evaluation:** The methodology of splitting the data into training and testing sets allowed for a robust and unbiased evaluation, confirming the model's excellent predictive power on new data.

While such perfect scores are rare in more complex, real-world problems, this analysis serves as a clear and effective demonstration of the machine learning evaluation process.