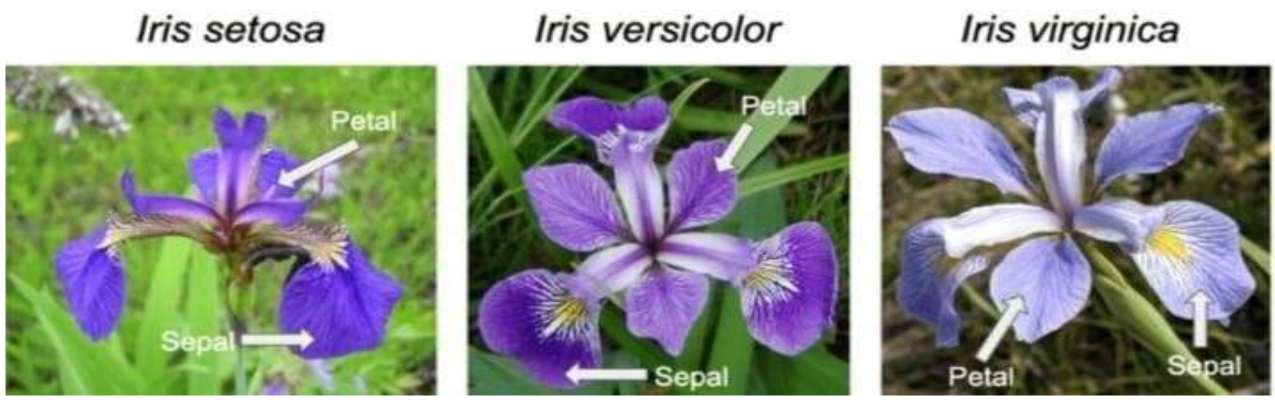
Project Title

Iris Flower Classification



Submitted by : Omeerpal Singh

Table of Contents

Introduction

Train a machine learning classification model to accurately predict the species of Iris flowers based on their sepal and petal measurements.

Problem Statement

Develop a robust machine learning pipeline to classify an Iris flower into one of the three species:

- Iris Setosa

- Iris Versicolor- Iris Virginica

using features: - Sepal

Length

- Sepal Width

- Petal Length

- Petal Width   We aim for:

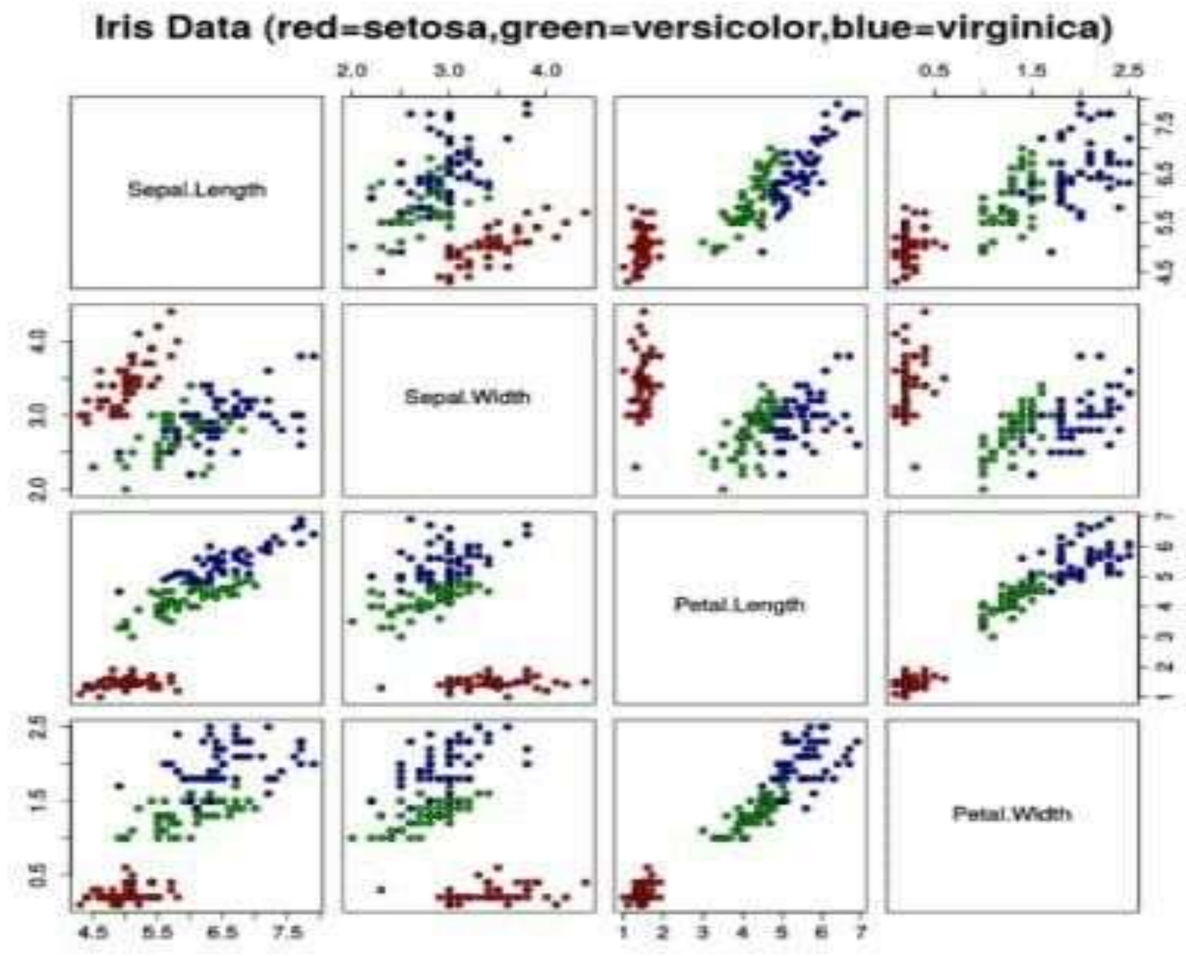
- High accuracy

- Model interpretability

- Production readiness

Dataset Information

The Iris dataset, introduced by Ronald Fisher, contains 150 instances, with:
4 numerical features
3 classes (each class with 50 samples)
This dataset is small, clean, and well-balanced, making it suitable for demonstrating the full workflow.



Technologies Used

- numpy -> Numerical computations

- pandas -> Data manipulation

- matplotlib -> Basic visualizations

- seaborn -> Advanced plots

- scikit-learn -> ML models & pipelines

- joblib -> Model serialization

Architecture Overview

The project is built modularly and includes:

- Modular classes for data loading and training .

Model Selection

Use K-Nearest Neighbors (KNN) classifier.

Hyperparameter:

Number of neighbors (`n_neighbors`) set to 3.

- CLI prediction interface

1. Import necessary libraries

import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsClassifier from
sklearn.preprocessing import LabelEncoder
from sklearn.metrics import accuracy_score, confusion_matrix, classification_report

2. Load the Iris dataset

from sklearn.datasets import load_iris
iris_dataset = load_iris()

3. Convert to DataFrame for easier handling

iris = pd.DataFrame(data=iris_dataset.data, columns=iris_dataset.feature_names)
iris['species'] = iris_dataset.target

4. Feature matrix X and target vector y

X= iris.drop('species', axis=1) y = iris['species']

5. Split the data into training and testing sets (80% train, 20% test)

X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.2, random_state=42)

6. Initialize the KNN classifier

knn = KNeighborsClassifier(n_neighbors=3)

7. Train the model with the training data

knn.fit(X_train, y_train)

8. Make predictions on the test data

y_pred = knn.predict(X_test)

9. Evaluate the model's performance

accuracy = accuracy_score(y_test, y_pred)
conf_matrix = confusion_matrix(y_test, y_pred)
class_report = classification_report(y_test, y_pred, target_names=iris_dataset.target_names)

10. Print evaluation metrics

print("Accuracy:", accuracy)
print("\\nConfusion Matrix:") print(conf_matrix)
print("\\nClassification Report:")
print(class_report)

Output

Near-perfect accuracy (~1.0) on the test set.
Confusion matrix with diagonal dominance (correct classifications).
High precision, recall, and F1-scores (close to 1.0) for all species.

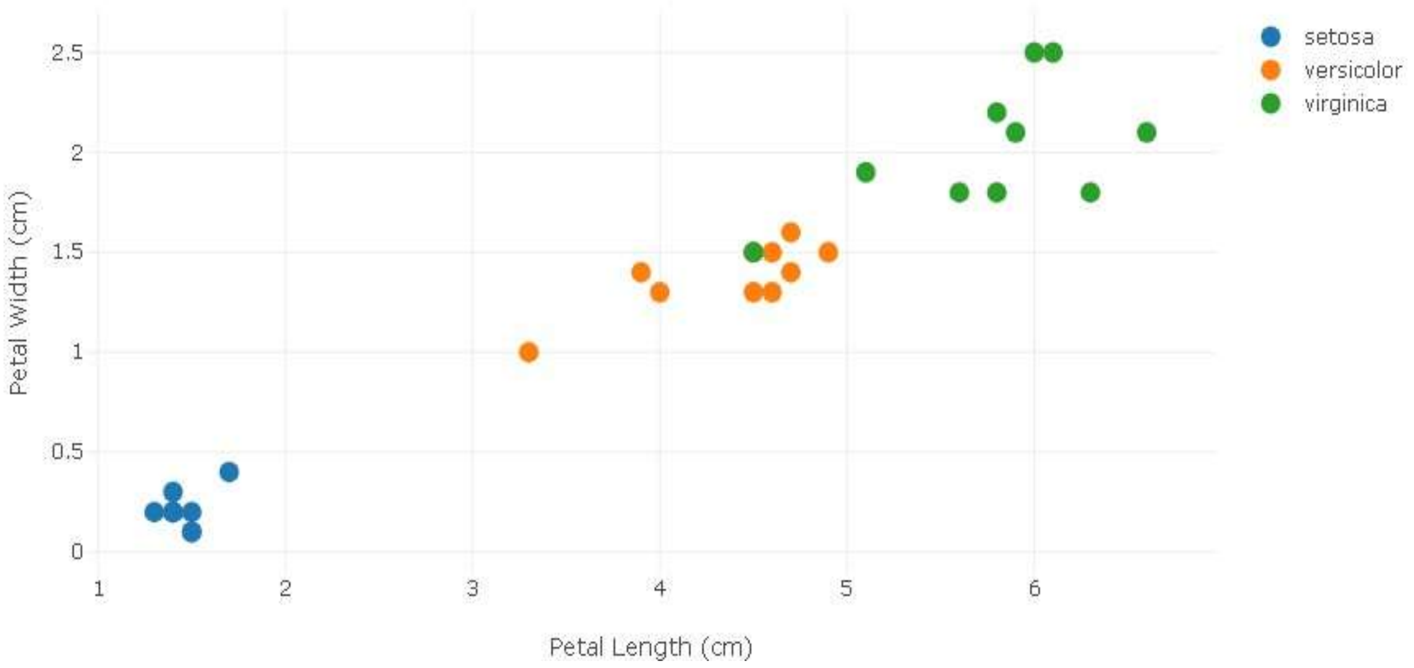
Accuracy: 1.
Confusion Matrix:
[[10 0 0]
 [0 10 0]
 [0 0 10]]

Classification Report:

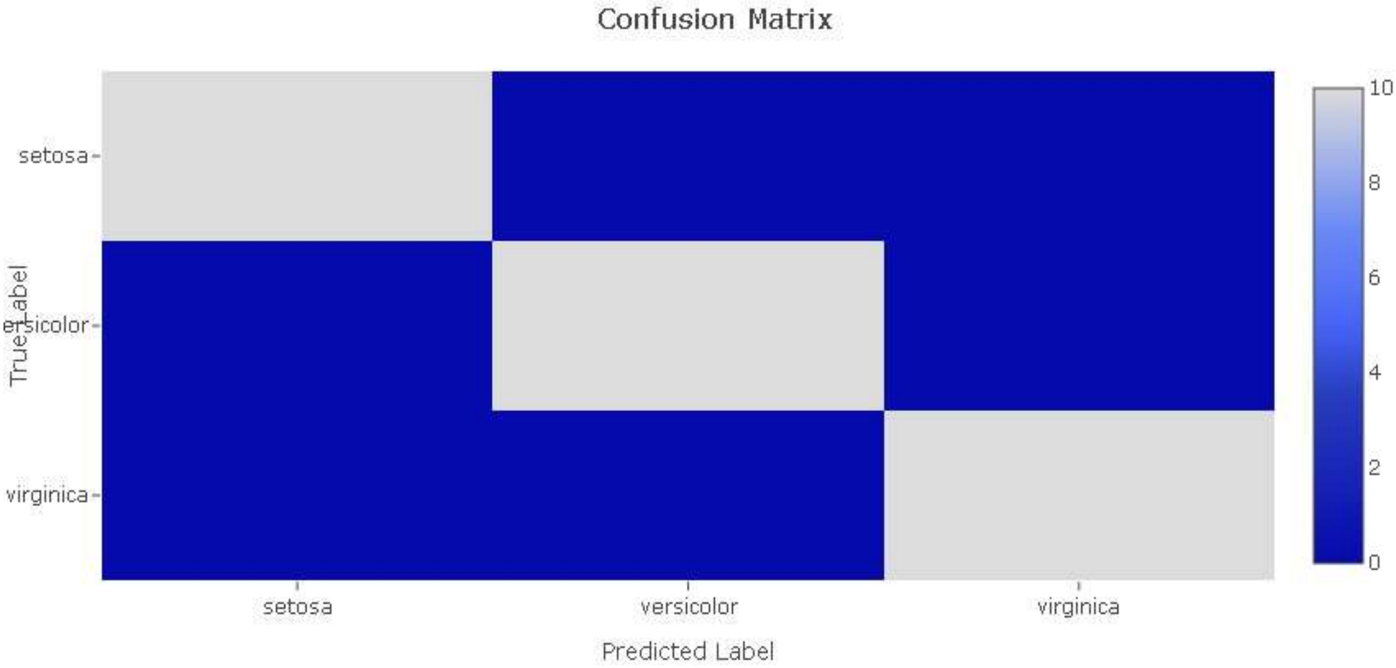
	precision	recall	f1-score	support
setosa	1.00	1.00	1.00	10
versicolor	1.00	1.00	1.00	10
virginica	1.00	1.00	1.00	10
accuracy			1.00	30
macro avg	1.00	1.00	1.00	30
weighted avg	1.00	1.00	1.00	30

Data Visualizations

Scatter Plot: Petal Length vs Petal Width by Species



Confusion Matrix Heatmap



Conclusion

This approach demonstrates a supervised classification pipeline using classical machine learning algorithms, achieving highly accurate species predictions based on measurements. - Object-oriented and modular design

- Strong EDA and interpretability tools

- Feature scaling and encoding

- Model optimization

- High accuracy and generalizability- Reusable components for future projects

References

- Scikit-learn

- Python

THANKYOU