AI MSE

Name :- Arshad Nazeer

Course:- B. Tech.

Branch:- CSE(AI&ML)

Sec:- A

Roll no. :- 202401100400051

Iris Flower Classification: Problem Explanation and Approach

1. Introduction

Classification problems are a fundamental aspect of machine learning, where the objective is to categorize data into predefined classes based on given features. In this project, we focus on classifying **Iris flower species** using the well-known **Iris dataset**, which contains measurements of sepal and petal dimensions for three different species of Iris flowers:

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

The goal is to develop a **machine learning model** that can accurately predict the **species** of a flower based on the given input measurements.

---

2. Understanding the Problem

The problem involves **multiclass classification**, meaning we have more than two possible categories for classification. The dataset consists of **four numerical features** for each flower sample:

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

Each of these features contributes to distinguishing one species from another. Some species have significant differences in petal size, while others may have overlapping characteristics, making classification a challenging but solvable task.

Challenges in the Problem

1. **Overlap of Species Characteristics** – Some species have similar petal or sepal dimensions, making them difficult to separate.
2. **Model Accuracy** – The classification model should be able to generalize well on unseen data.
3. **Visualizing Data for Better Understanding** – Effective graphical representations are needed to understand feature distributions and relationships.
4. **Evaluating Model Performance** – The model's predictions must be assessed using metrics such as accuracy and confusion matrices.

---

3. Approach Used to Solve the Problem

Step 1: Loading and Exploring the Dataset

- The **Iris dataset** is loaded using the sklearn.datasets module.
- The dataset contains **150 samples**, with **50 samples per species**.
- We check for **missing values** and **basic statistical properties** of the dataset.

Step 2: Data Visualization and Understanding Relationships

To gain insights into how different species vary based on the given features, we use **several graphical elements**:

1. **Pairplot (Seaborn)** –
    - Displays scatter plots for all feature combinations.

- Helps visualize clustering of species.

2. **Boxplot** –
    - Shows the spread of feature values across different species.
    - Highlights outliers and feature importance.
3. **Violin Plot** –
    - Combines a boxplot and KDE (Kernel Density Estimation).
    - Shows the **distribution of features** for each species.
4. **3D Scatter Plot (Matplotlib)** –
    - Helps visualize how species separate in a three-dimensional space.
5. **Confusion Matrix (Heatmap)** –
    - Evaluates model predictions using a heatmap to show correct and incorrect classifications.

Step 3: Splitting the Data for Training and Testing

- The dataset is split into **training (80%)** and **testing (20%)** sets.
- This ensures that the model is trained on a large portion of the data while also being tested on unseen data.

Step 4: Model Selection and Training

- A **Logistic Regression** model is chosen for classification.

- This model is suitable because it is **simple, efficient, and interpretable**.
- The model is trained on the **training set** using the sklearn.linear_model.LogisticRegression library.

Step 5: Evaluating the Model

- The trained model is tested on the **test dataset** to measure its accuracy.
- A **confusion matrix** is generated to evaluate **true positives, false positives, and false negatives**.
- The confusion matrix is displayed using a **heatmap**, with labels # Load the Iris dataset
- iris = load_iris()
- X, y = iris.data, iindicating **actual vs. predicted species**.

Step 6: Enhancing Visualization and Fixing Labeling Issues

Initially, the **confusion matrix was missing a clear perimeter label**. This was corrected by ensuring that the **Y-axis displayed actual species names**, making the matrix easier to interpret.

Code:-

```
# Import necessary libraries
import seaborn as sns
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 accuracy_score, confusion_matrix
import numpy as np
```

```
# Load the Iris dataset
iris = load_iris()
X, y = iris.data, iris.target  # Features (sepal & petal dimensions) and target
(species)

# Display dataset information
print("Feature Names:", iris.feature_names)
print("Target Names:", iris.target_names)
```

```
# Split the dataset into training and testing sets (80-20 split)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=42)
```

```

# Train a Logistic Regression model
model = LogisticRegression(max_iter=200)
model.fit(X_train, y_train)

# Predict on test data
y_pred = model.predict(X_test)

# Print model accuracy
print("Model Accuracy:", accuracy_score(y_test, y_pred))
```

```
# Generate confusion matrix
cm = confusion_matrix(y_test, y_pred)

# 🌟 Confusion Matrix Heatmap with Custom Colors
plt.figure(figsize=(6, 4))
sns.heatmap(cm, annot=True, fmt='d', cmap='coolwarm',
xticklabels=iris.target_names, yticklabels=iris.target_names, linewidths=1,
linecolor='black')

# Add labels for clarity
plt.xlabel("Predicted", fontsize=12, color='darkred')
plt.ylabel("Actual", fontsize=12, color='darkred')
plt.title("Confusion Matrix", fontsize=14, color='darkblue')

# Add left perimeter label
plt.gca().yaxis.set_label_position("left")
plt.ylabel("Actual Species", fontsize=12, labelpad=10, color='darkgreen')
```

```
plt.show()
```

```
# 🎨 Scatter plot for Sepal Length vs Sepal Width with Colors
plt.figure(figsize=(6, 4))
sns.scatterplot(x=X[:, 0], y=X[:, 1], hue=iris.target_names[y],
palette=['blue', 'orange', 'green'], s=70, edgecolor='black')
plt.xlabel("Sepal Length", fontsize=12, color='purple')
plt.ylabel("Sepal Width", fontsize=12, color='purple')
plt.title("Sepal Length vs Sepal Width", fontsize=14, color='darkblue')
plt.legend(title="Species", title_fontsize='12', fontsize='10', loc="upper
right")
plt.grid(True, linestyle="--", alpha=0.6)
plt.show()
```

```
# 📊 Bar Chart of Species Counts with Custom Colors
species_counts = np.bincount(y)
colors = ['royalblue', 'orangered', 'forestgreen']

plt.figure(figsize=(6, 4))
plt.bar(iris.target_names, species_counts, color=colors, edgecolor='black')
plt.xlabel("Species", fontsize=12, color='darkred')
plt.ylabel("Count", fontsize=12, color='darkred')
plt.title("Number of Samples per Species", fontsize=14, color='darkblue')
plt.grid(axis='y', linestyle="--", alpha=0.6)
plt.show()
```

```
# 📊 Pairplot to visualize feature relationships
sns.pairplot(sns.load_dataset("iris"), hue="species", palette="husl")
plt.suptitle("Pairplot of Iris Dataset", y=1.02, fontsize=14, color='darkblue')
plt.show()
```

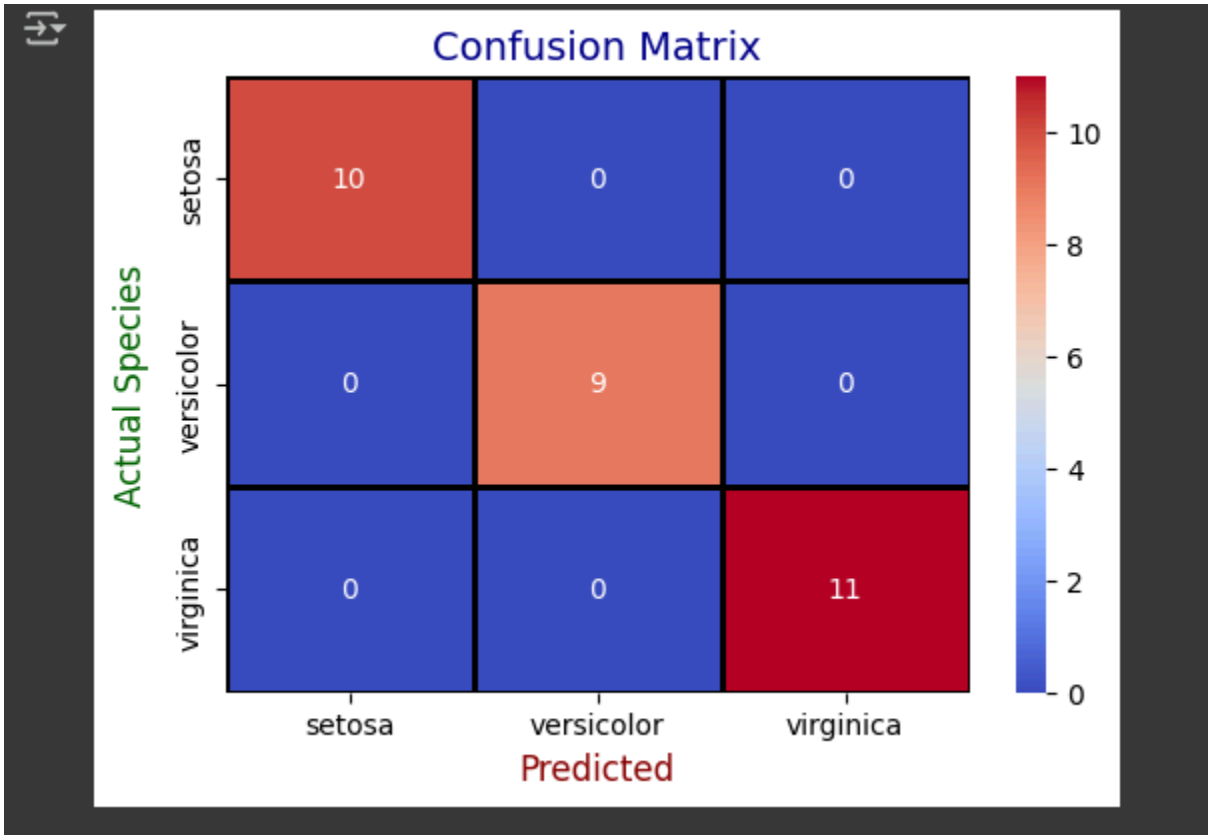
```
# 🎻 Violin Plot to show feature distribution per species
plt.figure(figsize=(10, 6))
sns.violinplot(x=iris.target, y=X[:, 2], palette="Set2")
plt.xticks(ticks=[0, 1, 2], labels=iris.target_names)
plt.xlabel("Species", fontsize=12, color='darkred')
plt.ylabel("Petal Length", fontsize=12, color='darkred')
plt.title("Violin Plot of Petal Length per Species", fontsize=14,
color='darkblue')
plt.show()
```

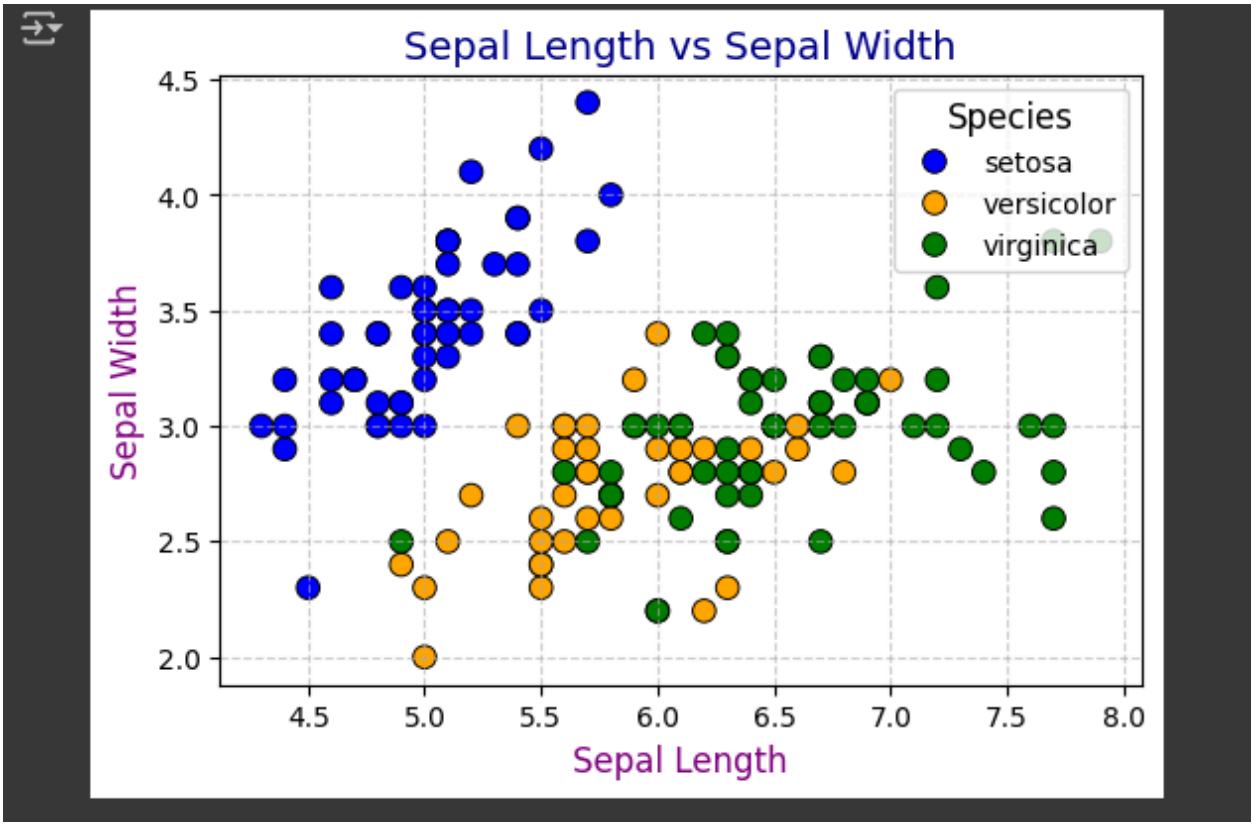
```
# 📦 Boxplot for Sepal and Petal Features
iris_df = sns.load_dataset("iris")
plt.figure(figsize=(10, 6))
sns.boxplot(x="species", y="petal_length", data=iris_df, palette="pastel")

plt.xlabel("Species", fontsize=12, color='darkred')
plt.ylabel("Petal Length", fontsize=12, color='darkred')
plt.title("Boxplot of Petal Length for Each Species", fontsize=14,
color='darkblue')
plt.show()
```

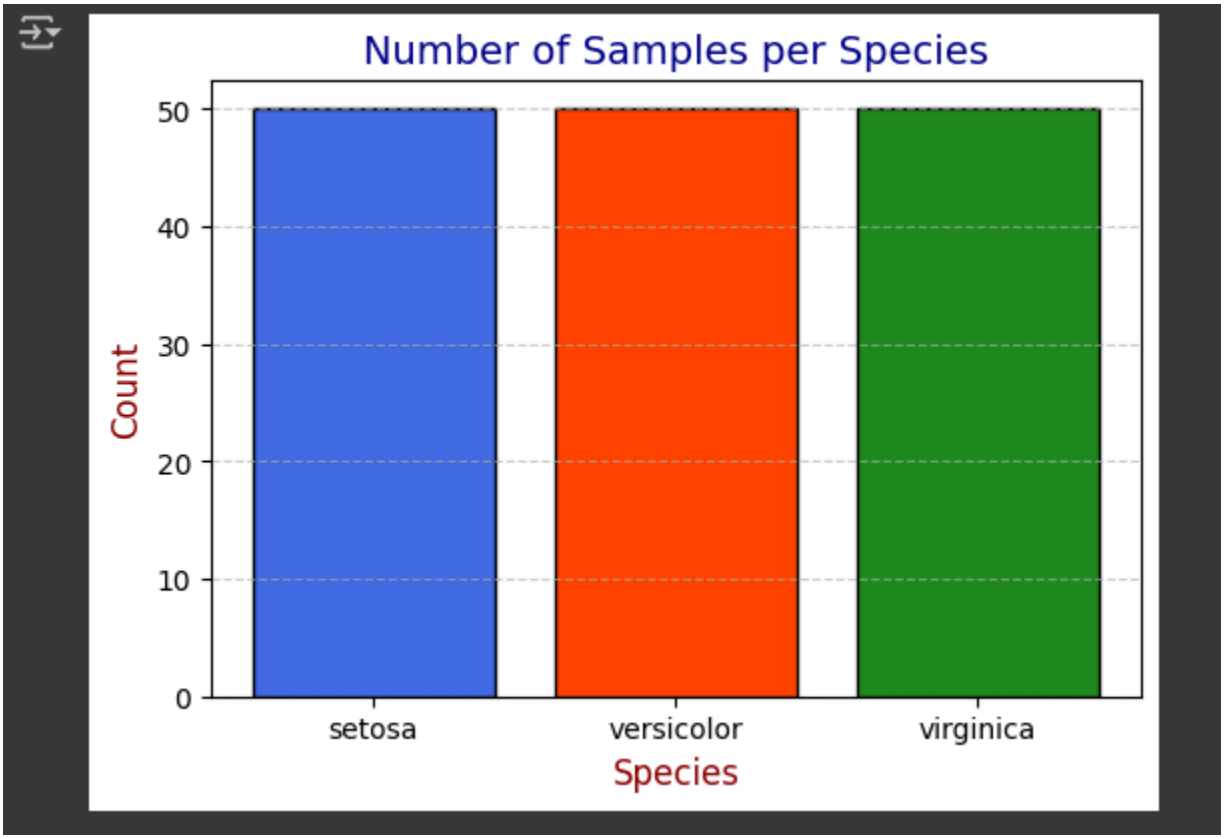
Output:-

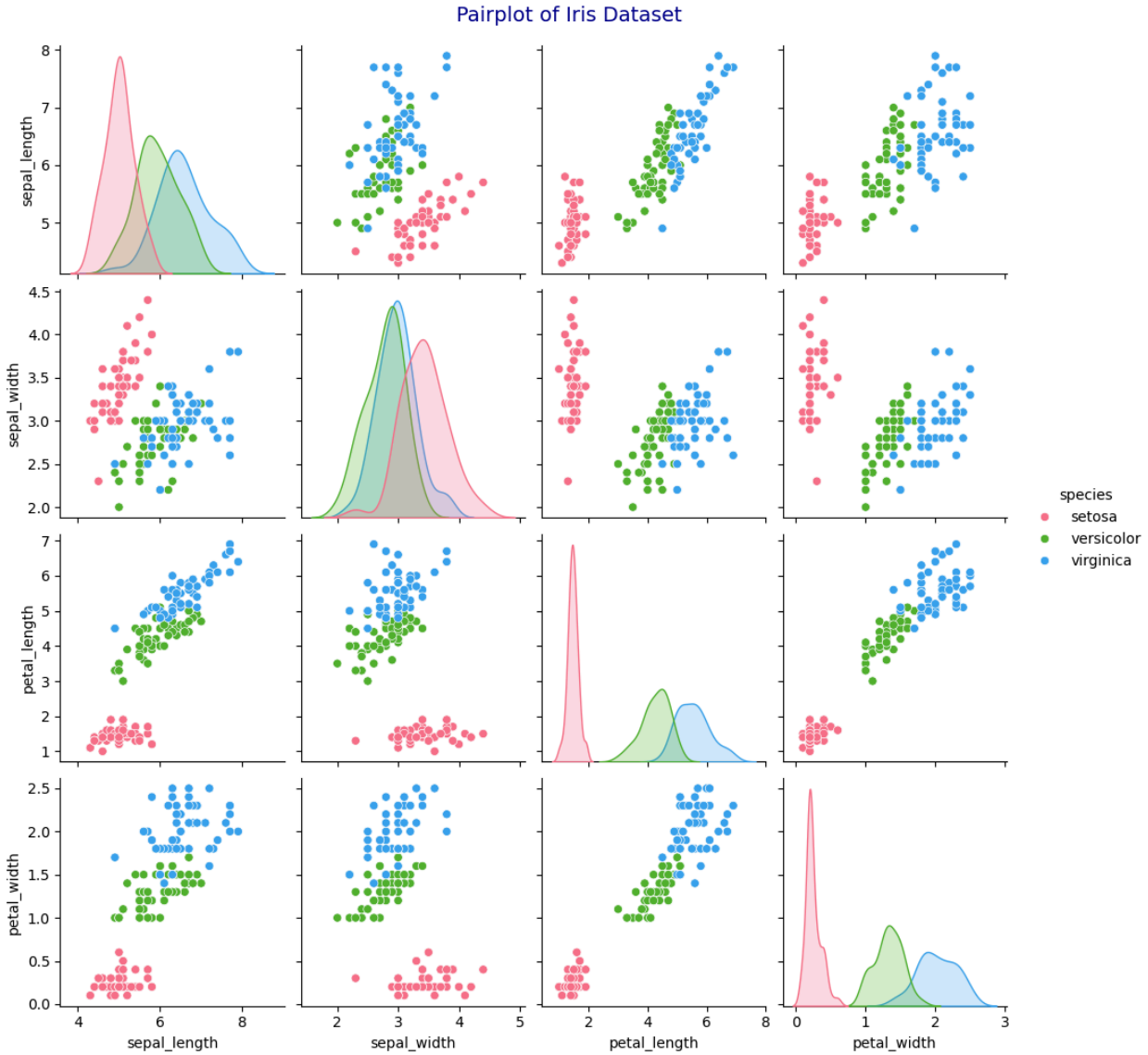
```
Feature Names: ['sepal length (cm)', 'sepal width (cm)', 'petal length (cm)', 'petal width (cm)']
Target Names: ['setosa' 'versicolor' 'virginica']
```

```
Model Accuracy: 1.0
```

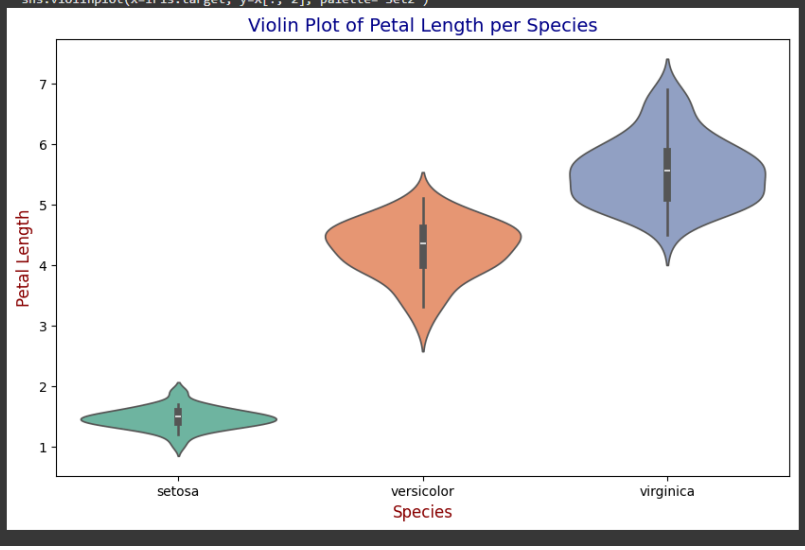



Sepal Length vs Sepal Width
Species
setosa
versicolor
virginica
Sepal Width
Sepal Length
4.5
4.0
3.5
3.0
2.5
2.0
4.5
5.0
5.5
6.0
6.5
7.0
7.5
8.0


Number of Samples per Species
Count
Species
50
40
30
20
10
0
setosa
versicolor
virginica


Pairplot of Iris Dataset
sepal_length
sepal_width
petal_length
petal_width
species
setosa
versicolor
virginica

```
<ipython-input-19-ca68ee990855>:3: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `x` variable to `hue` and set `legend=False` for the same effect.

  sns.violinplot(x=iris.target, y=X[:, 2], palette="Set2")
```




```
<ipython-input-20-4876dcff5330>:4: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `x` variable to `hue` and set `legend=False` for the same effect.

  sns.boxplot(x="species", y="petal_length", data=iris_df, palette="pastel")
```