Iris Flower Classification Using Machine Learning

Note: this project is done in Ubuntu 24.04 LTS

Step 1: Install Required Software

1. Open a terminal and update your system:
   ‘’’
   sudo apt update && sudo apt upgrade -y
   ‘’’
2. Check if Python is installed:
   ‘’’
   python3 --version
   ‘’’
   If not, install it:
   ‘’’
   sudo apt install python3 python3-pip -y
   ‘’’
3. Install essential libraries for data science and machine learning:
   a. Create a virtual environment:
      ‘’’
      sudo apt update
          sudo apt install python3.12-venv

      python3 -m venv myenv
      ‘’’
   b. Activate the virtual environment:
      ‘’’
      source myenv/bin/activate
      ‘’’
   c. Install packages inside the virtual environment:
      ‘’’
      pip install numpy pandas scikit-learn matplotlib seaborn
      ‘’’

Step 2: Download the Iris Dataset

1. Download the Iris dataset:
   ‘’’
   wget https://archive.ics.uci.edu/ml/machine-learning-databases/iris/iris.data
   ‘’’
2. Rename the file for convenience:
   ‘’’

```
mv iris.data iris.csv
```

Step 3: Prepare the Environment

1. Create a project directory:
   ```
   mkdir iris_classification && cd iris_classification
   ```
2. Move the dataset to the project directory:
   ```
   mv ../iris.csv .
   ```

Step 4: Write the Python Code

1. Create a Python script for the project:
   ```
   nano iris_classification.py
   ```
2. Copy and paste the following code into the file:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix

# Load dataset
column_names = ['sepal_length', 'sepal_width', 'petal_length', 'petal_width', 'species']
data = pd.read_csv('iris.csv', header=None, names=column_names)

# Map species to numeric values
data['species'] = data['species'].astype('category').cat.codes

# Split data into features and target
X = data.iloc[:, :-1]
y = data.iloc[:, -1]

# Split into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

```
# Initialize k-NN classifier
knn = KNeighborsClassifier(n_neighbors=3)

# Train the model
knn.fit(X_train, y_train)

# Predict on test set
y_pred = knn.predict(X_test)

# Evaluate the model
accuracy = accuracy_score(y_test, y_pred)
print(f"Accuracy: {accuracy * 100:.2f}%\n")
print("Classification Report:")
print(classification_report(y_test, y_pred))

# Visualizations

# 1. Pairplot
sns.pairplot(data, hue='species', diag_kind='kde', markers=["o", "s", "D"])
plt.title("Pairplot of Features Grouped by Species")
plt.savefig('pairplot.png')

# 2. Confusion Matrix Heatmap
conf_matrix = confusion_matrix(y_test, y_pred)  # Define the confusion matrix
plt.figure(figsize=(8, 6))
sns.heatmap(conf_matrix, annot=True, cmap='Blues', fmt='d', xticklabels=np.unique(y),
yticklabels=np.unique(y))
plt.title("Confusion Matrix Heatmap")
plt.xlabel("Predicted")
plt.ylabel("Actual")
plt.savefig('confusion_matrix.png')  # Save the plot as an image

# 3. Decision Boundary Visualization (Optional)
# Select two features for visualization (e.g., sepal_length and sepal_width)
X = data[['sepal_length', 'sepal_width']]  # Update X to only two features
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
knn.fit(X_train, y_train)

h = 0.02  # Step size in the mesh
x_min, x_max = X.iloc[:, 0].min() - 1, X.iloc[:, 0].max() + 1
y_min, y_max = X.iloc[:, 1].min() - 1, X.iloc[:, 1].max() + 1
xx, yy = np.meshgrid(np.arange(x_min, x_max, h), np.arange(y_min, y_max, h))
```

```
Z = knn.predict(np.c_[xx.ravel(), yy.ravel()])
Z = Z.reshape(xx.shape)

plt.figure(figsize=(8, 6))
plt.contourf(xx, yy, Z, alpha=0.8, cmap=plt.cm.Paired)
plt.scatter(X.iloc[:, 0], X.iloc[:, 1], c=y, edgecolors='k', cmap=plt.cm.Paired)
plt.title("k-NN Decision Boundary")
plt.xlabel("Sepal Length")
plt.ylabel("Sepal Width")
plt.savefig('decision_boundary.png')  # Save the decision boundary plot
```

Note: this code contains the classification result and code for the data visualization.

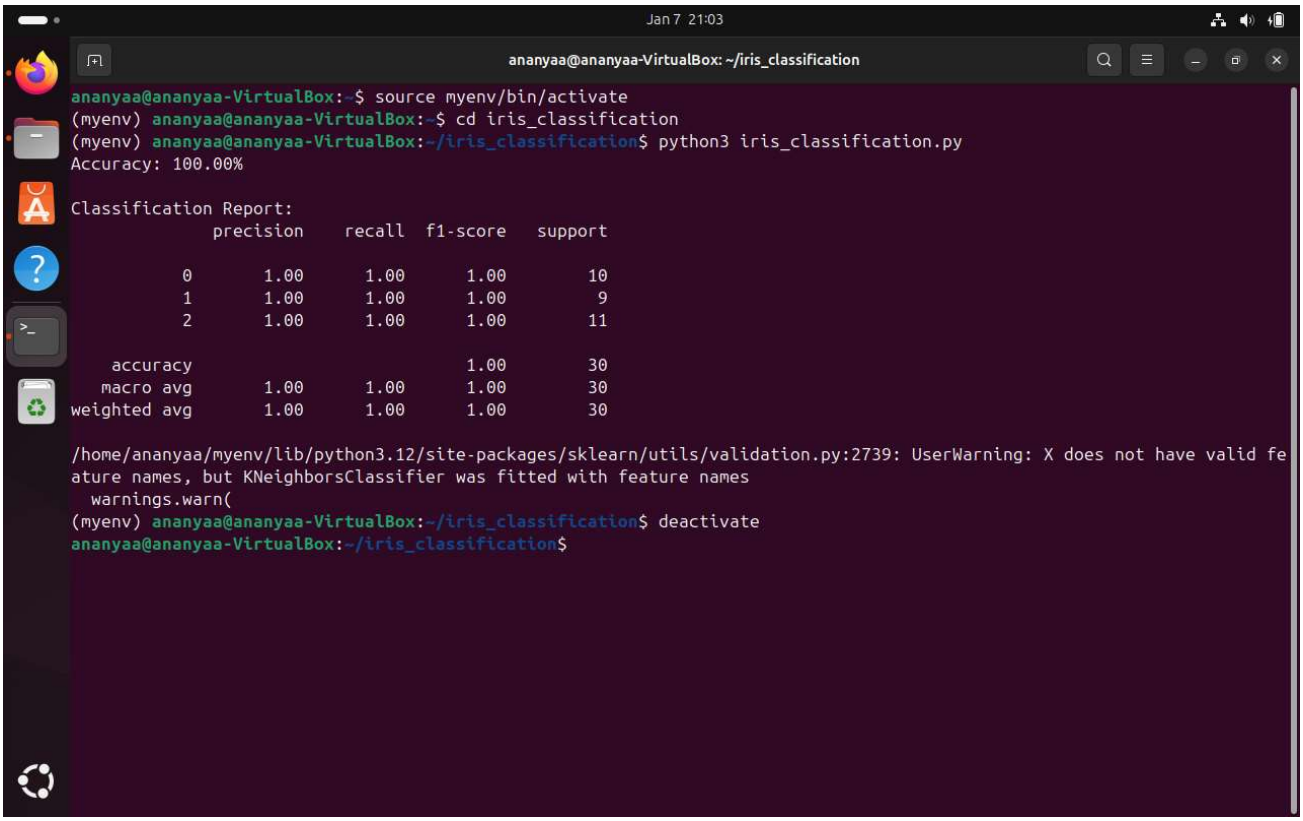
3. Save the file and exit by pressing `Ctrl + O`, `Enter`, and `Ctrl + X`.

Step 5: Run the Script

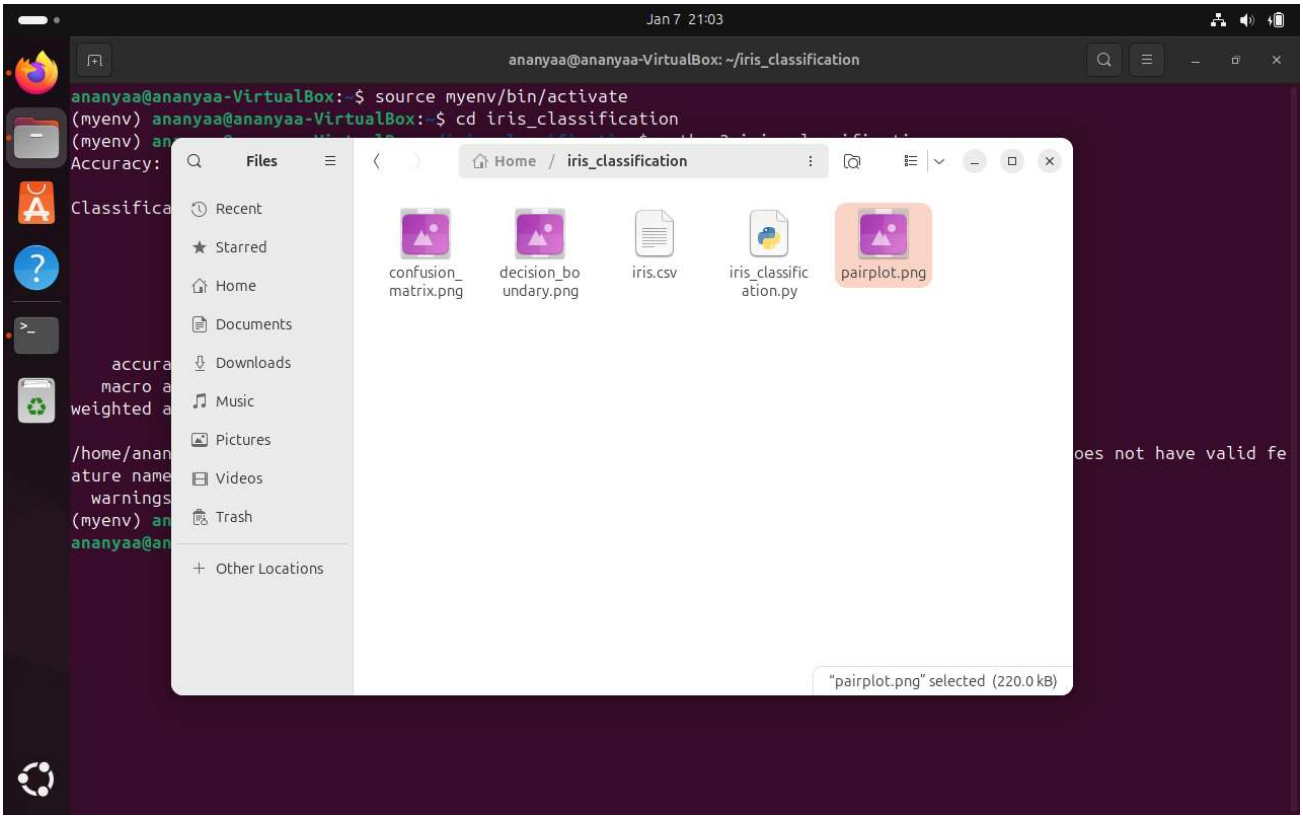
1. Run the script:
   ‘’’
   python3 iris_classification.py
   ‘’’
2. The output will display the model's accuracy and classification report.

OUTPUTS:

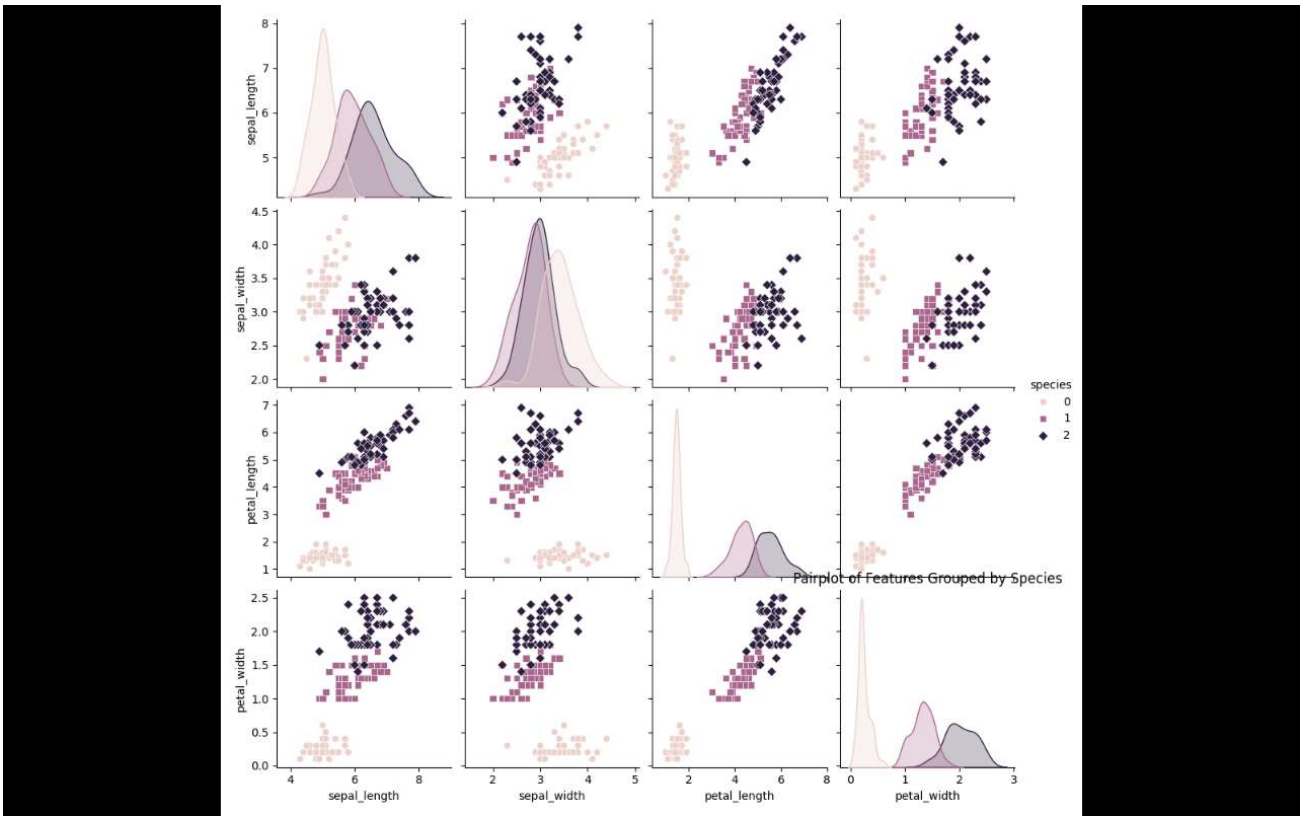
1. Model's accuracy and classification report in the ubuntu terminal:



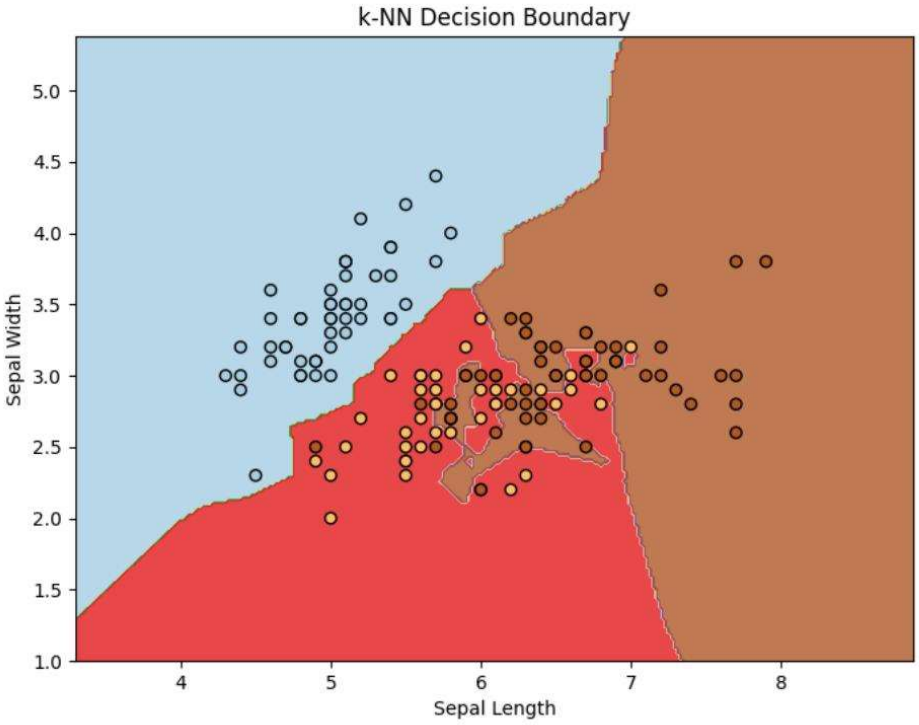
2. The contents of the iris_classification project folder after execution:



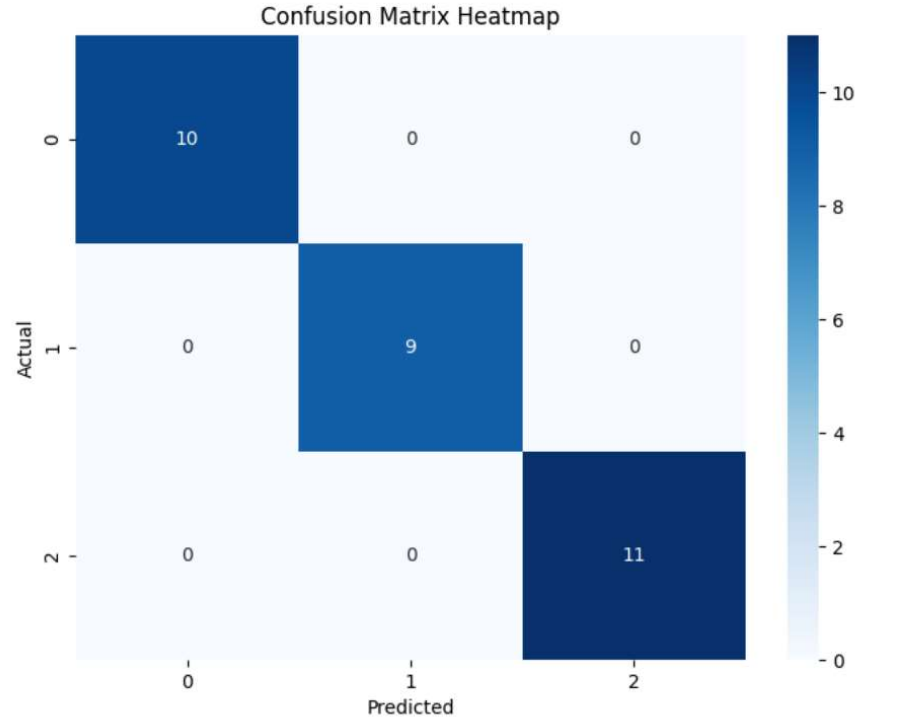
3. Pairplot.png:



4. Decision_boundary.png:



5. Confusion_matrix.png:



Note: in the terminal type: *deactivate* (this is a must step after everything to stop environment.)