Department of CSE(AI) and CSE(AIML)

Introduction to AI

REPORT

"IRIS FLOWER CLASSIFICATION"

NAME : DIVYANSH VERMA

PROGRAM : BTECH

BRANCH: CSE(AIML)

SECTION: B

SEMESTER: 2nd

SUBMITTED TO : MR. ABHISHEK SHUKLA SIR

INTRODUCTION

The Iris Flower Classification is the machine learning that The Iris Flower Classification is a machine learning classification problem where we classify three species of iris flowers based on their petal and sepal dimensions. It is a popular dataset in machine learning, widely used for beginner-level projects to understand classification techniques.

Here's the complete Python code for classifying Iris flower species using the famous Iris dataset. The code follows a structured flow:

1. Load Dataset
2. Explore Data
3. Preprocess Data
4. Train Model
5. Evaluate Model
6. Make Predictions

Explanation of Code Flow:

1. Import Libraries:

We import essential libraries:

NumPy & Pandas – Handle data efficiently.

Seaborn & Matplotlib – Create visualizations.

Scikit-learn – Load the dataset, preprocess data, train the model, and evaluate results.

sklearn is used for dataset loading, model training, and evaluation.

2. Load and Explore the Dataset:

The Iris dataset contains 150 samples of three flower species **(Setosa, Versicolor, Virginica)**.

Each sample has 4 features **(sepal length, sepal width, petal length, petal width)**.

We convert the dataset into a pandas DataFrame for better visualization.

3. Visualizing the Data:

We use **seaborn.pairplot()** to visualize feature distributions across different species.

4. Data Preprocessing:

The dataset is split into training (80%) and testing (20%) sets using **train_test_split().**

Features are standardized using **StandardScaler()** to improve model performance.

5. Model Training:

We use K-Nearest Neighbors (KNN) classifier with k=5.

The model is trained using **knn.fit().**

6. Making Predictions & Evaluating Model:

Predictions are made on the test set using **knn.predict()**.

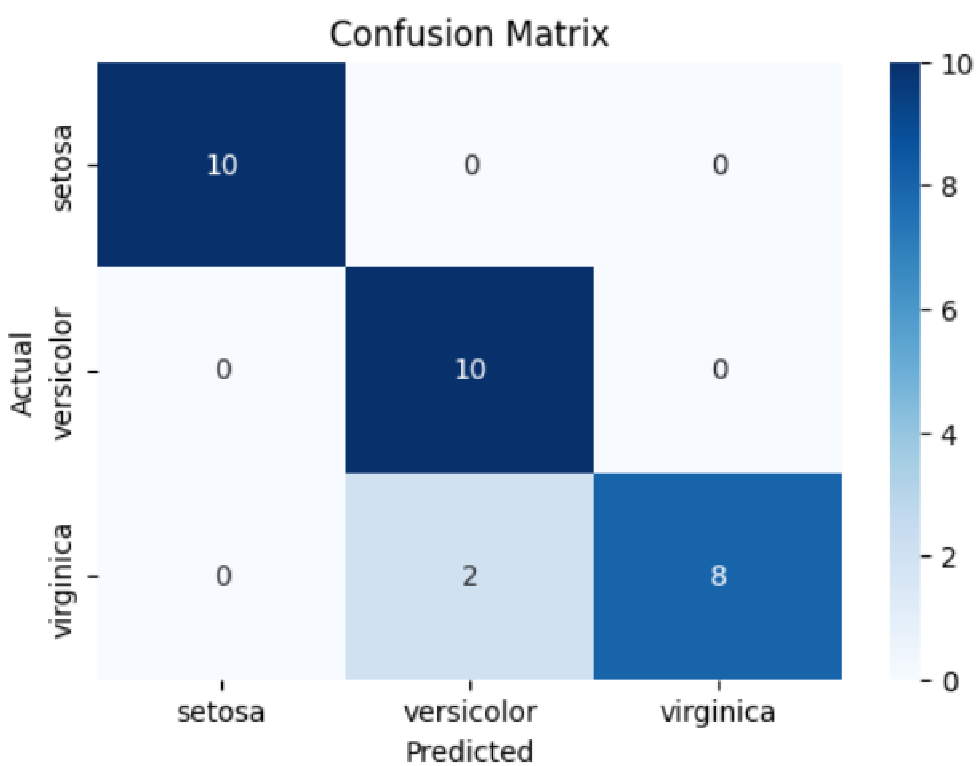
Accuracy score, classification report, and confusion matrix are generated for performance evaluation.

Explaination about Confusion Matrix in output:

It tells us how well the model predicts each class compared to the actual values.

Like this is the table of an order in which one of the species is compared with all the species and if the prediction matches to the actual species given it return the True Value and if the predict data missclassified with other species it return False Positive and if it not machted with the species it will give False Negative in the table form.

In this program we have given three species as to predict so the 3X3 table is formed as Confusion Matrix.



<u>Accuracy Score Calculation</u> :

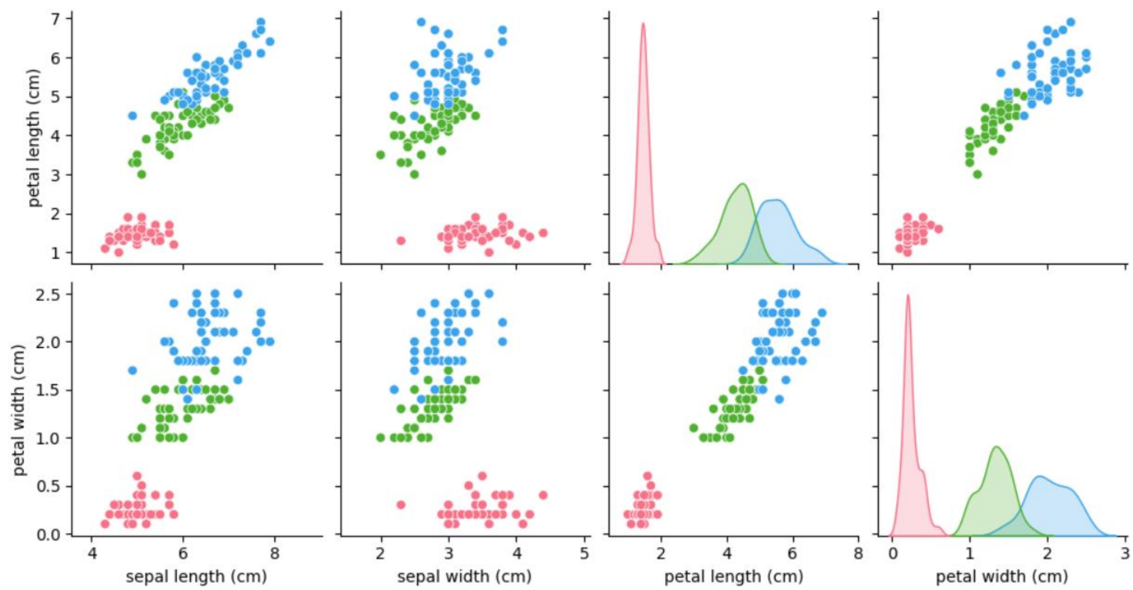
According to the Confusion Matrix, the accuracy is calculated as follows:

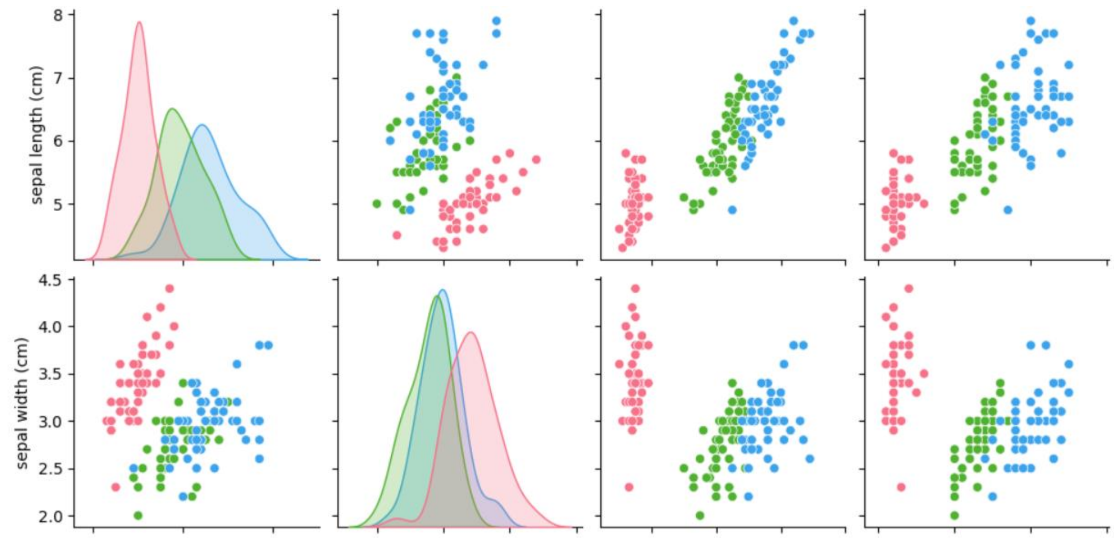
Accuracy = (Total correct predictions)/ (Total predictions given)

This is the actual output according to the predictions :

```
✅ Model Accuracy: 93.33%

📌 Classification Report:
              precision    recall  f1-score   support

      setosa       1.00      1.00      1.00        10
  versicolor       0.83      1.00      0.91        10
   virginica       1.00      0.80      0.89        10

    accuracy                           0.93        30
   macro avg       0.94      0.93      0.93        30
weighted avg       0.94      0.93      0.93        30
```

These are the following graphs that are predicted as output :





Here is the attached code for the problem statement given :

Import necessary libraries

import numpy as np

import pandas as pd

import seaborn as sns

import matplotlib.pyplot as plt

```
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix
from sklearn import datasets

# Load the Iris dataset
iris = datasets.load_iris() #loading required dataset for the output and plotting

# Convert dataset into a DataFrame
df = pd.DataFrame(iris.data, columns=iris.feature_names)
df['species'] = iris.target

# Mapping species labels to actual names
df['species'] = df['species'].map({0: 'Setosa', 1: 'Versicolor', 2: 'Virginica'})

# Display the first few rows of the dataset
print("\n📌 First 5 rows of the dataset:")
print(df.head())

# Summary of dataset
print("\n📌 Dataset Summary:")
print(df.describe())

# Visualizing the dataset
plt.figure(figsize=(10, 6))
```

```
sns.pairplot(df, hue="species", palette="husl")

plt.show()


# Splitting data into features (X) and target labels (y)

X = iris.data    # Features (sepal & petal dimensions)

y = iris.target    # Target variable (species)


# Splitting into training (80%) and testing (20%) sets

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


# Standardizing the data (important for KNN)

scaler = StandardScaler()

X_train = scaler.fit_transform(X_train)

X_test = scaler.transform(X_test)


# Train a K-Nearest Neighbors (KNN) classifier

knn = KNeighborsClassifier(n_neighbors=5)

knn.fit(X_train, y_train)


# Make predictions

y_pred = knn.predict(X_test)


# Model Evaluation

accuracy = accuracy_score(y_test, y_pred)

print(f"\n✅ Model Accuracy: {accuracy:.2%}")    # Display as percentage
```

Display classification report

print("\n📌 Classification Report:")

print(classification_report(y_test, y_pred, target_names=iris.target_names))

Plotting confusion matrix

conf_matrix = confusion_matrix(y_test, y_pred)

plt.figure(figsize=(6, 4))

sns.heatmap(conf_matrix, annot=True, fmt='d', cmap='Blues', xticklabels=iris.target_names, yticklabels=iris.target_names)

plt.xlabel('Predicted')

plt.ylabel('Actual')

plt.title('Confusion Matrix')

plt.show()

Outputs:

```
📌 First 5 rows of the dataset:
   sepal length (cm)  sepal width (cm)  petal length (cm)  petal width (cm)  \
0                5.1               3.5                1.4               0.2
1                4.9               3.0                1.4               0.2
2                4.7               3.2                1.3               0.2
3                4.6               3.1                1.5               0.2
4                5.0               3.6                1.4               0.2

  species
0  Setosa
1  Setosa
2  Setosa
3  Setosa
4  Setosa
```

📌 Dataset Summary:

```
       sepal length (cm)  sepal width (cm)  petal length (cm)  \
count         150.000000        150.000000         150.000000
mean            5.843333          3.057333           3.758000
std             0.828066          0.435866           1.765298
min             4.300000          2.000000           1.000000
25%             5.100000          2.800000           1.600000
50%             5.800000          3.000000           4.350000
75%             6.400000          3.300000           5.100000
max             7.900000          4.400000           6.900000

       petal width (cm)
count        150.000000
mean           1.199333
std            0.762238
min            0.100000
25%            0.300000
50%            1.300000
```

```
       petal width (cm)
count        150.000000
mean           1.199333
std            0.762238
min            0.100000
25%            0.300000
50%            1.300000
75%            1.800000
max            2.500000
```