TASK 3 : IRIS FLOWER CLASSIFICATION

In [ ]:
```
# Import Required Libraries
```

In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import classification_report, confusion_matrix
```

In [2]:
```
# Load Iris dataset from sklearn
iris = load_iris()
```

Explore and Prepare the Data

In [4]:
```
# Create DataFrame from the dataset
df = pd.DataFrame(data=iris.data, columns=iris.feature_names)
df['target'] = iris.target
```

In [5]:
```
# Map target names to target labels
target_names = iris.target_names
df['species'] = df['target'].map(lambda x: target_names[x])
```

In [6]:
```
# Display first few rows of the DataFrame
df.head()
```

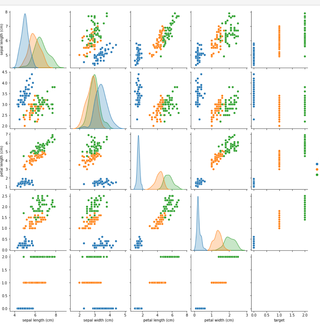
Out[6]:

	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)	target	species
0	5.1	3.5	1.4	0.2	0	setosa
1	4.9	3.0	1.4	0.2	0	setosa
2	4.7	3.2	1.3	0.2	0	setosa
3	4.6	3.1	1.5	0.2	0	setosa
4	5.0	3.6	1.4	0.2	0	setosa

Visualize the Data

Explore the dataset using visualizations.

In [7]:
```
# Pairplot to visualize relationships between features
sns.pairplot(df, hue='species')
plt.show()
```



Split Data into Train and Test Sets

Split the dataset into training and testing sets.

In [8]:
```
# Split data into features (X) and target (y)
X = df.drop(['target', 'species'], axis=1)
y = df['target']
```

In [9]:
```
# Split data into train and test sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_st
```

Feature Scaling

Scale the features using StandardScaler.

In [10]:
```
# Standardize features
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)
```

Train a Classifier (e.g., K-Nearest Neighbors)

Train a machine learning classifier on the training data.

In [11]:
```
# Create K-Nearest Neighbors classifier
knn = KNeighborsClassifier(n_neighbors=3)
knn.fit(X_train_scaled, y_train)
```

Out[11]:
```
KNeighborsClassifier(n_neighbors=3)
```

Evaluate the Model

Evaluate the trained model on the test data.

In [12]:
```
# Predictions on the test set
y_pred = knn.predict(X_test_scaled)
```

In [13]:
```
# Classification report and confusion matrix
print("Classification Report:\n", classification_report(y_test, y_pred))
print("Confusion Matrix:\n", confusion_matrix(y_test, y_pred))
```

```
Classification Report:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        10
           1       1.00      1.00      1.00         9
           2       1.00      1.00      1.00        11

    accuracy                           1.00        30
   macro avg       1.00      1.00      1.00        30
weighted avg       1.00      1.00      1.00        30

Confusion Matrix:
 [[10  0  0]
 [ 0  9  0]
 [ 0  0 11]]
```

Here's how you can structure the entire code in a Jupyter Notebook:

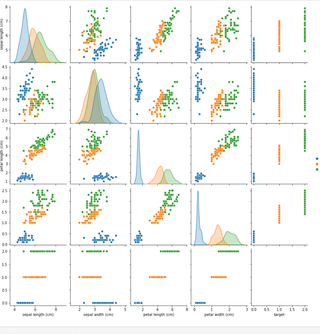
In [14]:
```
# Import libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import classification_report, confusion_matrix
```

In [15]:
```
# Load Iris dataset from sklearn
iris = load_iris()
```

In [16]:
```
# Create DataFrame from the dataset
df = pd.DataFrame(data=iris.data, columns=iris.feature_names)
df['target'] = iris.target
```

In [17]:
```
# Map target names to target labels
target_names = iris.target_names
df['species'] = df['target'].map(lambda x: target_names[x])
```

In [18]:
```
# Pairplot to visualize relationships between features
sns.pairplot(df, hue='species')
plt.show()
```



In [19]:
```
# Split data into features (X) and target (y)
X = df.drop(['target', 'species'], axis=1)
y = df['target']
```

In [20]:
```
# Split data into train and test sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_st
```

In [21]:
```
# Standardize features
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)
```

In [22]:
```
# Create K-Nearest Neighbors classifier
knn = KNeighborsClassifier(n_neighbors=3)
knn.fit(X_train_scaled, y_train)
```

Out[22]:
```
KNeighborsClassifier(n_neighbors=3)
```

In [23]:
```
# Predictions on the test set
y_pred = knn.predict(X_test_scaled)
```

In [24]:
```
# Classification report and confusion matrix
print("Classification Report:\n", classification_report(y_test, y_pred))
print("Confusion Matrix:\n", confusion_matrix(y_test, y_pred))
```

```
Classification Report:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        10
           1       1.00      1.00      1.00         9
           2       1.00      1.00      1.00        11

    accuracy                           1.00        30
   macro avg       1.00      1.00      1.00        30
weighted avg       1.00      1.00      1.00        30

Confusion Matrix:
 [[10  0  0]
 [ 0  9  0]
 [ 0  0 11]]
```

In [ ]: