In [1]:

```python
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

# Load Iris dataset from sklearn
iris = load_iris()
```

In [2]:

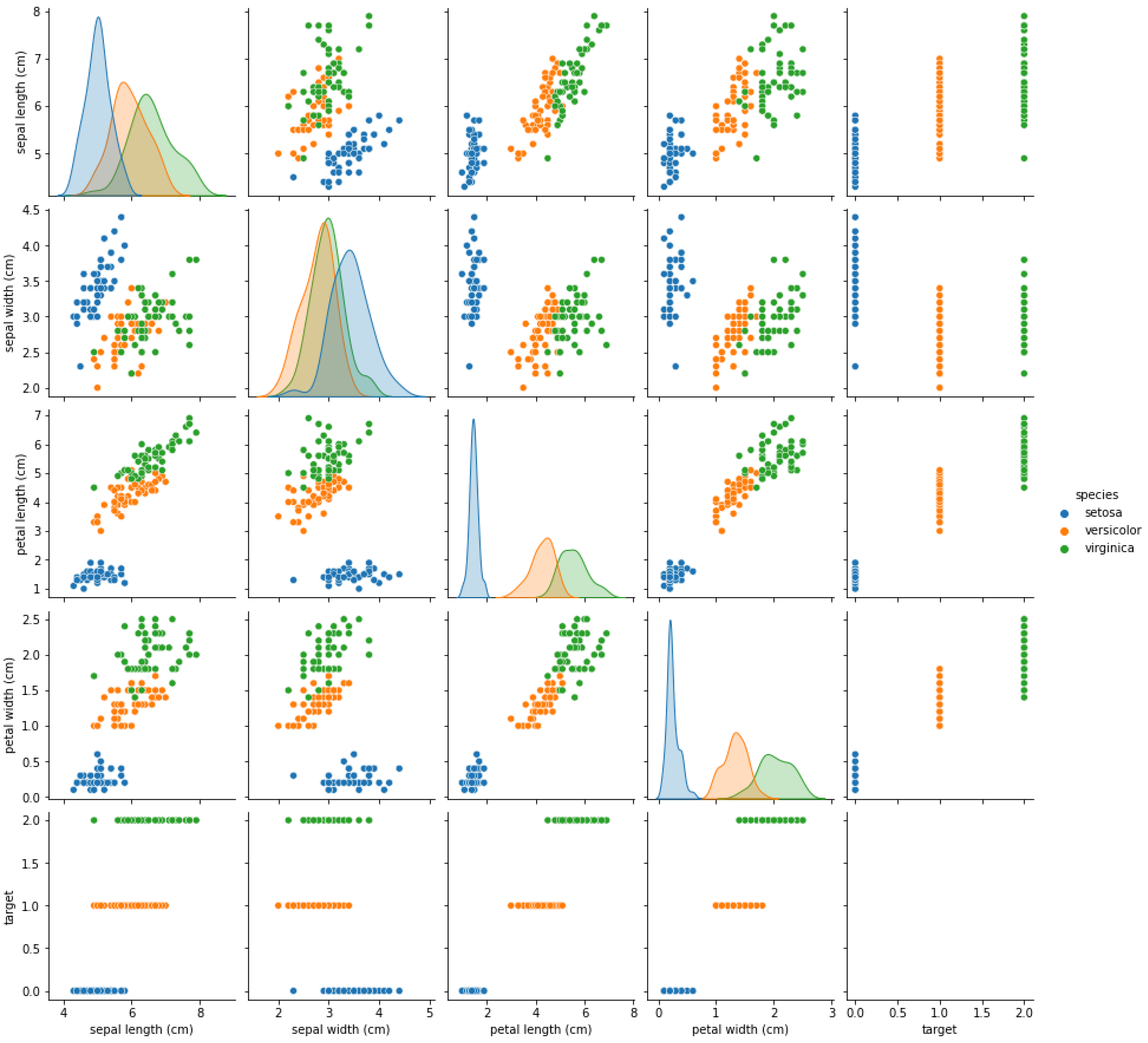
```python
# Create DataFrame from the dataset
df = pd.DataFrame(data=iris.data, columns=iris.feature_names)
df['target'] = iris.target

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

# Display first few rows of the DataFrame
df.head()
```

Out[2]:

	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

In [3]:

```python
# Pairplot to visualize relationships between features
sns.pairplot(df, hue='species')
plt.show()
```



In [4]:

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

# 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_state=42)
```

In [5]:

```python
# Standardize features
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)
```

In [6]:

```python
# Create K-Nearest Neighbors classifier
knn = KNeighborsClassifier(n_neighbors=3)
knn.fit(X_train_scaled, y_train)
```

Out[6]:

```
KNeighborsClassifier(n_neighbors=3)
```

In [10]:

```python
# Predictions on the test set
y_pred = knn.predict(X_test_scaled)

# 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))
```

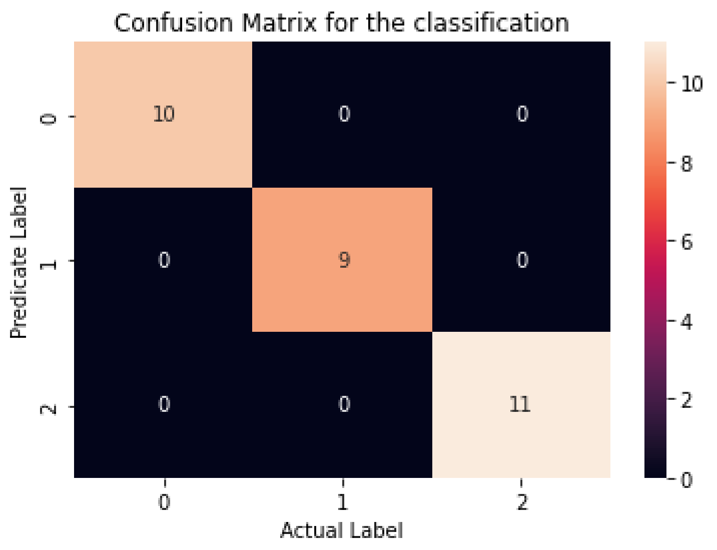
```
Classification Report:
               precision    recall  f1-score   support

           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 [12]:

```python
cm=confusion_matrix(y_test,y_pred)
sns.heatmap(cm,annot=True)
plt.xlabel('Actual Label')
plt.ylabel('Predicate Label')
plt.title('Confusion Matrix for the classification')
```

Out[12]:

```
Text(0.5, 1.0, 'Confusion Matrix for the classification')
```



In [ ]: