In [1]:
```
import numpy as np
import pandas as pd
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
```

In [2]:
```
from sklearn.datasets import load_iris

iris = load_iris()
X = iris.data
y = iris.target
```

In [3]:
```
df = pd.DataFrame(X, columns=iris.feature_names)
df['species'] = y
print(df.head())
```

```
   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        0
1        0
2        0
3        0
4        0
```

In [4]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)
```

In [7]:
```
knn = KNeighborsClassifier(n_neighbors=3)
knn.fit(X_train, y_train)
```

Out[7]:
```
KNeighborsClassifier(n_neighbors=3)
```

In [8]:
```
y_pred = knn.predict(X_test)
```

In [9]:
```
accuracy = accuracy_score(y_test, y_pred)
print(f"Accuracy: {accuracy * 100:.2f}%")
print(classification_report(y_test, y_pred))
print(confusion_matrix(y_test, y_pred))
```

```
Accuracy: 100.00%
              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

[[10  0  0]
 [ 0  9  0]
 [ 0  0 11]]
```

In [10]:
```
for i in range(len(y_test)):
    correct = y_test[i] == y_pred[i]
    print(f"Actual: {iris.target_names[y_test[i]]}, Predicted: {iris.target_names[y_pred[i]]} - {'Co
```

```
Actual: versicolor, Predicted: versicolor - Correct
Actual: setosa, Predicted: setosa - Correct
Actual: virginica, Predicted: virginica - Correct
Actual: versicolor, Predicted: versicolor - Correct
Actual: versicolor, Predicted: versicolor - Correct
Actual: setosa, Predicted: setosa - Correct
Actual: versicolor, Predicted: versicolor - Correct
Actual: virginica, Predicted: virginica - Correct
Actual: versicolor, Predicted: versicolor - Correct
Actual: versicolor, Predicted: versicolor - Correct
Actual: virginica, Predicted: virginica - Correct
Actual: setosa, Predicted: setosa - Correct
Actual: setosa, Predicted: setosa - Correct
Actual: setosa, Predicted: setosa - Correct
Actual: setosa, Predicted: setosa - Correct
Actual: versicolor, Predicted: versicolor - Correct
Actual: virginica, Predicted: virginica - Correct
Actual: versicolor, Predicted: versicolor - Correct
Actual: versicolor, Predicted: versicolor - Correct
Actual: virginica, Predicted: virginica - Correct
Actual: setosa, Predicted: setosa - Correct
Actual: virginica, Predicted: virginica - Correct
Actual: setosa, Predicted: setosa - Correct
Actual: virginica, Predicted: virginica - Correct
Actual: virginica, Predicted: virginica - Correct
Actual: virginica, Predicted: virginica - Correct
Actual: virginica, Predicted: virginica - Correct
Actual: virginica, Predicted: virginica - Correct
Actual: setosa, Predicted: setosa - Correct
Actual: setosa, Predicted: setosa - Correct
```

In [ ]: