```
import numpy as np
import pandas as pd
from sklearn import datasets
from sklearn.preprocessing import StandardScaler
from sklearn.cluster import KMeans
from sklearn.metrics import silhouette_score
import matplotlib.pyplot as plt
import seaborn as sns
```

```
iris = datasets.load_iris()
X = iris.data
y_true = iris.target
```

```
#create dataframe
df = pd.DataFrame(X, columns=iris.feature_names)
print("First five rows of dataset:\n")
print(df.head())
```

```
First five rows of 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
```

```
#feature scaling
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)
```

```
#k-means clustering from k=3
k = 3
kmeans = KMeans(n_clusters=k, random_state=42)
kmeans.fit(X_scaled)
```

```
KMeans(n_clusters=3, random_state=42)
```

```
#cluster assignments
labels = kmeans.labels_
print("\nCluster labels assigned to each data point:\n", labels[:20])
```

```
Cluster labels assigned to each data point:
 [1 2 2 2 1 1 1 1 2 2 1 1 2 2 1 1 1 1 1 1]
```

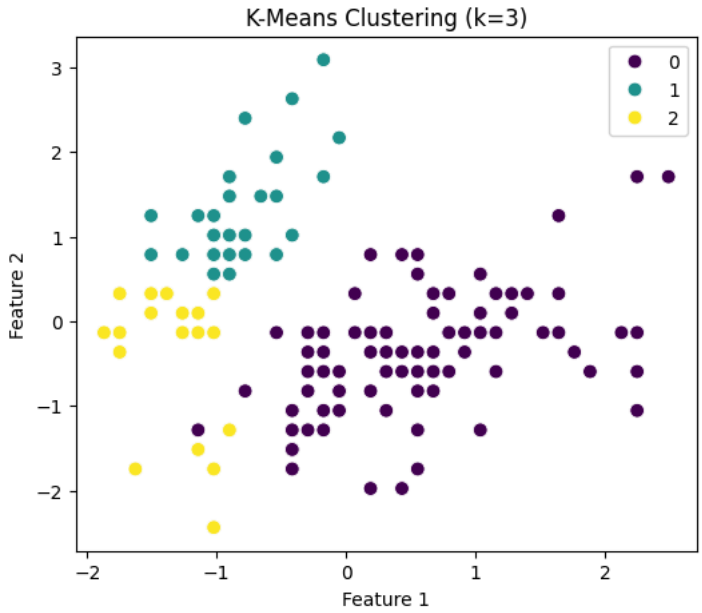
```
#analyze the result
df["Cluster"] = labels
print("\nClustered data sample:\n")
print(df.head(10))
```

```
Clustered data sample:

   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
5                5.4               3.9                1.7               0.4
6                4.6               3.4                1.4               0.3
7                5.0               3.4                1.5               0.2
8                4.4               2.9                1.4               0.2
9                4.9               3.1                1.5               0.1

   Cluster
0        1
1        2
2        2
3        2
4        1
5        1
6        1
7        1
8        2
9        2
```

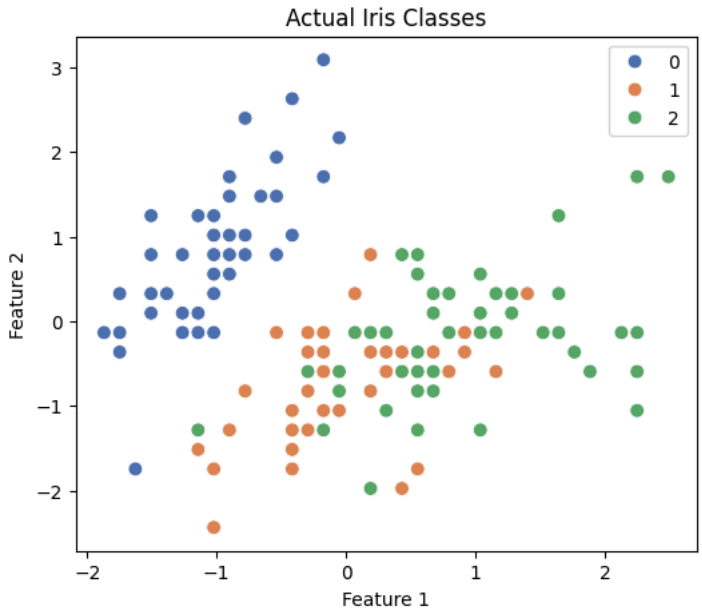
```
#evaluate clustering performance
print("\nInertia (sum of squared distances to closest cluster center):", kmeans.inertia_)
silhouette_avg = silhouette_score(X_scaled, labels)
print("Silhouette Score (cluster quality):", round(silhouette_avg, 3))
```

```
Inertia (sum of squared distances to closest cluster center): 191.02473685317958
Silhouette Score (cluster quality): 0.48
```

```
#Visualize the clusters (using first 2 features)
plt.figure(figsize=(6,5))
sns.scatterplot(x=X_scaled[:,0], y=X_scaled[:,1], hue=labels, palette="viridis", s=60)
plt.title("K-Means Clustering (k=3)")
plt.xlabel("Feature 1")
plt.ylabel("Feature 2")
plt.show()
```




```
#compare with true labels (just for reference)
plt.figure(figsize=(6,5))
sns.scatterplot(x=X_scaled[:,0], y=X_scaled[:,1], hue=y_true, palette="deep", s=60)
plt.title("Actual Iris Classes")
plt.xlabel("Feature 1")
plt.ylabel("Feature 2")
plt.show()
```




```
#find the best number of clusters (Elbow Method)
inertia_values = []
k_range = range(1, 11)
```

```
for k in k_range:
    km = KMeans(n_clusters=k, random_state=42)
    km.fit(X_scaled)
    inertia_values.append(km.inertia_)
```

```
plt.figure(figsize=(7,4))
plt.plot(k_range, inertia_values, marker='o')
plt.title("Elbow Method for Optimal k")
plt.xlabel("Number of clusters (k)")
plt.ylabel("Inertia (Within-Cluster SSE)")
plt.grid(True)
plt.show()
```