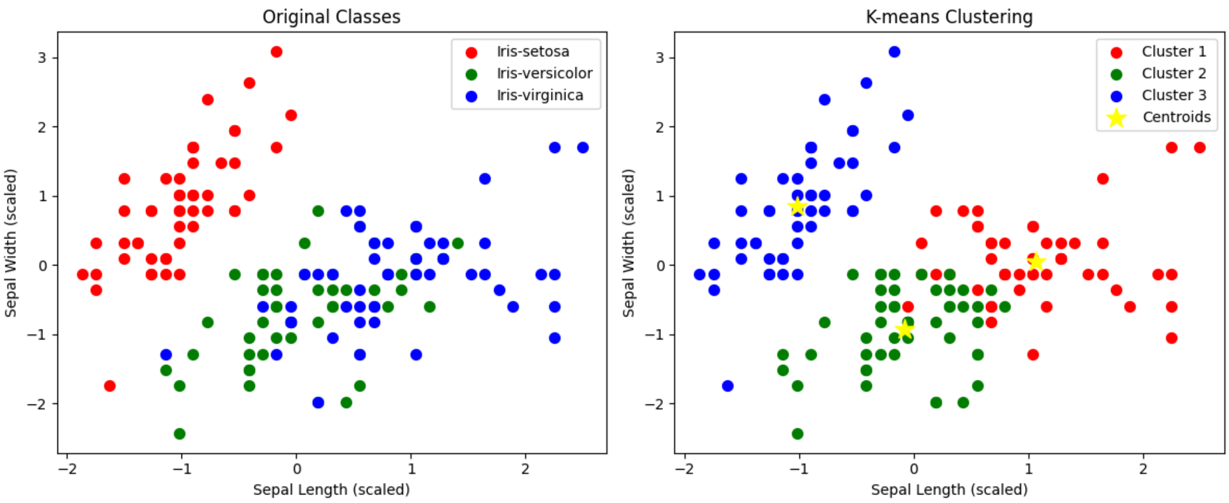
```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
from sklearn.datasets import load_iris
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import silhouette_score, adjusted_rand_score

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

scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)

class KMeans:
def __init__(self, n_clusters=3, max_iters=100, random_state=None):
self.n_clusters = n_clusters
self.max_iters = max_iters
self.random_state = random_state
self.centroids = None
self.labels = None

def initialize_centroids(self, X):
np.random.seed(self.random_state)
centroids = []
centroids.append(X[np.random.randint(X.shape[0])])

for _ in range(1, self.n_clusters):
distances = np.array([min([np.linalg.norm(x-c)**2 for c in centroids])
for x in X])
probs = distances / distances.sum()
cumulative_probs = probs.cumsum()
r = np.random.rand()
for j, p in enumerate(cumulative_probs):
if r < p:
centroids.append(X[j])
break

return np.array(centroids)

def assign_clusters(self, X):
distances = np.sqrt(((X - self.centroids[:,
np.newaxis])**2).sum(axis=2))
return np.argmin(distances, axis=0)

def update_centroids(self, X, labels):
centroids = np.zeros((self.n_clusters, X.shape[1]))
for k in range(self.n_clusters):
```

```
if np.sum(labels == k) > 0:
centroids[k] = np.mean(X[labels == k], axis=0)
return centroids

def fit(self, X):
self.centroids = self.initialize_centroids(X)

for _ in range(self.max_iters):
old_labels = self.assign_clusters(X)

self.centroids = self.update_centroids(X, old_labels)

self.labels = self.assign_clusters(X)
if np.all(old_labels == self.labels):
break

return self

def predict(self, X):
return self.assign_clusters(X)

def inertia(self, X):
return np.sum(np.min(((X - self.centroids[:,
np.newaxis])**2).sum(axis=2), axis=0))

kmeans = KMeans(n_clusters=3, random_state=42)
kmeans.fit(X_scaled)
predicted_labels = kmeans.labels

inertia = kmeans.inertia(X_scaled)
silhouette = silhouette_score(X_scaled, predicted_labels)
rand_index = adjusted_rand_score(y, predicted_labels)

print(f"Inertia: {inertia:.4f}")
print(f"Silhouette Score: {silhouette:.4f}")
print(f"Adjusted Rand Index: {rand_index:.4f}")

plt.figure(figsize=(12, 5))

plt.subplot(1, 2, 1)
colors = ['red', 'green', 'blue']
for i in range(3):
plt.scatter(X_scaled[y == i, 0], X_scaled[y == i, 1], s=50,
c=colors[i], label=f"Iris-{iris.target_names[i]}")
plt.title('Original Classes')
plt.xlabel('Sepal Length (scaled)')
plt.ylabel('Sepal Width (scaled)')
plt.legend()
```

```
plt.subplot(1, 2, 2)
for i in range(3):
plt.scatter(X_scaled[predicted_labels == i, 0],
X_scaled[predicted_labels == i, 1], s=50, c=colors[i], label=f"Cluster
{i+1}")

plt.scatter(kmeans.centroids[:, 0], kmeans.centroids[:, 1], s=200,
c='yellow', marker='*', label='Centroids')
plt.title('K-means Clustering')
plt.xlabel('Sepal Length (scaled)')
plt.ylabel('Sepal Width (scaled)')
plt.legend()

plt.tight_layout()
plt.savefig('kmeans_iris_clustering.png')
plt.show()

inertias = []
silhouette_scores = []
k_values = range(1, 11)

for k in k_values:
if k == 1:
inertias.append(0)
silhouette_scores.append(0)
else:
kmeans = KMeans(n_clusters=k, random_state=42)
kmeans.fit(X_scaled)
inertias.append(kmeans.inertia(X_scaled))
if k > 1:
silhouette_scores.append(silhouette_score(X_scaled, kmeans.labels))
else:
silhouette_scores.append(0)
plt.figure(figsize=(12, 5))

plt.subplot(1, 2, 1)
plt.plot(k_values, inertias, 'bo-')
plt.xlabel('Number of clusters (k)')
plt.ylabel('Inertia')
plt.title('Elbow Method for Optimal k')

plt.subplot(1, 2, 2)
plt.plot(k_values[1:], silhouette_scores[1:], 'ro-')
plt.xlabel('Number of clusters (k)')
plt.ylabel('Silhouette Score')
plt.title('Silhouette Method for Optimal k')

plt.tight_layout()
plt.savefig('kmeans_elbow_method.png')
plt.show()
```



kmeans_iris_clustering.png

Screenshot From 2025-04-13 12-54-02.png