Άσκηση 2.2

Ερώτημα Α

```
PS G:\My Drive\Ece Ntua\7th Semester\Machine Learning\Series of Exercises\Exercise 2> python3 .\Exercise_2a.py

  === Iteration 0 ===
  Point: (2, 3), Distances: [1.0, 2.23606797749979], Assigned to Cluster: 1
  Point: (3, 2), Distances: [1.0, 2.23606797749979], Assigned to Cluster: 1
  Point: (1, 2), Distances: [2.23606797749979, 3.605551275463989], Assigned to Cluster: 1
  Point: (4, 5), Distances: [2.23606797749979, 1.0], Assigned to Cluster: 2
  Point: (5, 4), Distances: [2.23606797749979, 1.0], Assigned to Cluster: 2
  Point: (3, 4), Distances: [1.0, 1.0], Assigned to Cluster: 1
  Point: (6, 4), Distances: [3.1622776601683795, 2.0], Assigned to Cluster: 2
  Point: (6, 5), Distances: [3.605551275463989, 2.23606797749979], Assigned to Cluster: 2
  New Centroids: [[2.25 2.75]
   [5.25 4.5 ]]

  === Iteration 1 ===
  Point: (2, 3), Distances: [0.3535533905932738, 3.5794552658190883], Assigned to Cluster: 1
  Point: (3, 2), Distances: [1.0606601717798212, 3.3634060117684275], Assigned to Cluster: 1
  Point: (1, 2), Distances: [1.4577379737113252, 4.930770730829005], Assigned to Cluster: 1
  Point: (4, 5), Distances: [2.850438562747845, 1.346291201783626], Assigned to Cluster: 2
  Point: (5, 4), Distances: [3.020761493398643, 0.5590169943749475], Assigned to Cluster: 2
  Point: (3, 4), Distances: [1.4577379737113252, 2.3048861143232218], Assigned to Cluster: 1
  Point: (6, 4), Distances: [3.952847075210474, 0.9013878188659973], Assigned to Cluster: 2
  Point: (6, 5), Distances: [4.373213921133975, 0.9013878188659973], Assigned to Cluster: 2
  New Centroids: [[2.25 2.75]
   [5.25 4.5 ]]

  Convergence reached.

  === Final Output ===
  Final Centroids:
    Cluster 1: [2.25 2.75]
    Cluster 2: [5.25 4.5 ]

  Final Cluster Assignments:
    Point 1: Assigned to Cluster 1
    Point 2: Assigned to Cluster 1
    Point 3: Assigned to Cluster 1
    Point 4: Assigned to Cluster 2
    Point 5: Assigned to Cluster 2
    Point 6: Assigned to Cluster 1
    Point 7: Assigned to Cluster 2
    Point 8: Assigned to Cluster 2
PS G:\My Drive\Ece Ntua\7th Semester\Machine Learning\Series of Exercises\Exercise 2>
```

Βοηθητικός κώδικας που χρησιμοποιήθηκε:

```python
import numpy as np
# Data
points = np.array([[2, 3], [3, 2], [1, 2], [4, 5],
                   [5, 4], [3, 4], [6, 4], [6, 5]])

# Initialization of centroids
centroids = np.array([[3, 3], [4, 4]])

def euclidean_distance(point, centroid):
      """Calculation of the Euclidean distance between a point and a
centroid"""
    return np.sqrt(np.sum((point - centroid) ** 2))
```

```
def assign_points_to_clusters(points, centroids):
    """Assign each point to the nearest centroid"""
    clusters = []
    for point in points:
         distances = [euclidean_distance(point, centroid) for centroid in
centroids]
        cluster = np.argmin(distances)  # Returns the index of the nearest
centroid
        clusters.append(cluster)
    return np.array(clusters)

def update_centroids(points, clusters, k):
     """Calculation of new centroids as the mean of the points in each
cluster"""
    new_centroids = []
    for i in range(k):
        cluster_points = points[clusters == i]
        if len(cluster_points) > 0:  # If there are points in the cluster
            new_centroid = np.mean(cluster_points, axis=0)
        else:  # If there are no points, retain the old centroid
            new_centroid = centroids[i]
        new_centroids.append(new_centroid)
    return np.array(new_centroids)
# Execution of the algorithm
iteration = 0
k = len(centroids)
while True:
    print(f"\n=== Iteration {iteration} ===")
    # Calculation of distances and assignment
    clusters = assign_points_to_clusters(points, centroids)
    for i, point in enumerate(points):
         distances = [euclidean_distance(point, centroid) for centroid in
centroids]
         print(f"Point: ({point[0]}, {point[1]}), Distances: {distances},
Assigned to Cluster: {clusters[i] + 1}")
    # Calculation of new centroids
    new_centroids = update_centroids(points, clusters, k)
    print(f"New Centroids: {new_centroids}")
    # Convergence check
    if np.allclose(centroids, new_centroids):
        print("\nConvergence reached.")
        break
    centroids = new_centroids
    iteration += 1
# Final output
print("\n=== Final Output ===")
print("Final Centroids:")
for i, centroid in enumerate(centroids, 1):
    print(f"  Cluster {i}: {centroid}")
print("\nFinal Cluster Assignments:")
for i, cluster in enumerate(clusters, 1):
    print(f"  Point {i}: Assigned to Cluster {cluster + 1}")
```

Ερώτημα Β, Γ

Ακολουθεί τα αποτελέσματα του αλγορίθμου k-means με όλα τα χαρακτηριστικά και με μόνο δύο, καθώς και ο αντίστοιχος βοηθητικός κώδικας που χρησιμοποιήθηκε.

```
PS G:\My Drive\Ece Ntua\7th Semester\Machine Learning\Series of Exercises\Exercise 2> python .\Exercise_2b_c_d.py

Convergence reached!

=== (b) Full Feature Set ===
Confusion Matrix:
[[50  0  0]
 [ 0 48  2]
 [ 0 14 36]]

Accuracy (Full Features): 89.33%

Convergence reached!

=== (c) Two Features ===
Confusion Matrix:
[[50  0  0]
 [ 0 48  2]
 [ 0  6 44]]
Accuracy: 94.67%

=== (d) Comparison ===
Accuracy with Full Features: 89.33%
Accuracy with Two Features: 94.67%
```

```python
import pandas as pd
import numpy as np
from sklearn.metrics import confusion_matrix, accuracy_score
import matplotlib.pyplot as plt
from sklearn.preprocessing import LabelEncoder
from collections import Counter

# Load the dataset
data = pd.read_csv('iris.csv')
data = data.drop(columns=['Id'])

# Encode Species to numerical values for comparison
label = LabelEncoder()
data['SpeciesEncoded'] = label.fit_transform(data['Species'])

# K-means algorithm implementation
def k_means(X, k, epsilon=1e-5, max_iterations=100):
    # Random initialization of centroids
    np.random.seed(42)
    centroids = X[np.random.choice(X.shape[0], k, replace=False)]

    for iteration in range(max_iterations):
        # Assign points to the nearest centroid
        distances = np.linalg.norm(X[:, np.newaxis] - centroids, axis=2)
```

```
        clusters = np.argmin(distances, axis=1)

        # Update centroids
        new_centroids = np.array(
            [X[clusters == i].mean(axis=0) for i in range(k)])

        # Convergence criterion
        if np.linalg.norm(new_centroids - centroids) < epsilon:
            print("\nConvergence reached!")
            break

        centroids = new_centroids

    return clusters, centroids

# Function to remap clusters to match true labels
def remap_clusters(true_labels, clusters):
    # Create a mapping from clusters to true labels based on majority vote
    mapping = {}
    for cluster_id in np.unique(clusters):
        # Find true labels corresponding to points in this cluster
        true_labels_in_cluster = true_labels[clusters == cluster_id]

        # Get the most common true label for this cluster
        most_common_label = Counter(
            true_labels_in_cluster).most_common(1)[0][0]
        mapping[cluster_id] = most_common_label

    # Remap the clusters
    remapped_clusters = np.array([mapping[cluster] for cluster in clusters])
    return remapped_clusters

# (b) Apply k-means with all features
X_full = data[['SepalLengthCm', 'SepalWidthCm',
               'PetalLengthCm', 'PetalWidthCm']].values
clusters_full, centroids_full = k_means(X_full, k=3)

# Map cluster labels to the true labels for comparison
true_labels = data['SpeciesEncoded'].values
clusters_full = remap_clusters(true_labels, clusters_full)
conf_matrix_full = confusion_matrix(true_labels, clusters_full)
accuracy_full = accuracy_score(true_labels, clusters_full)
```

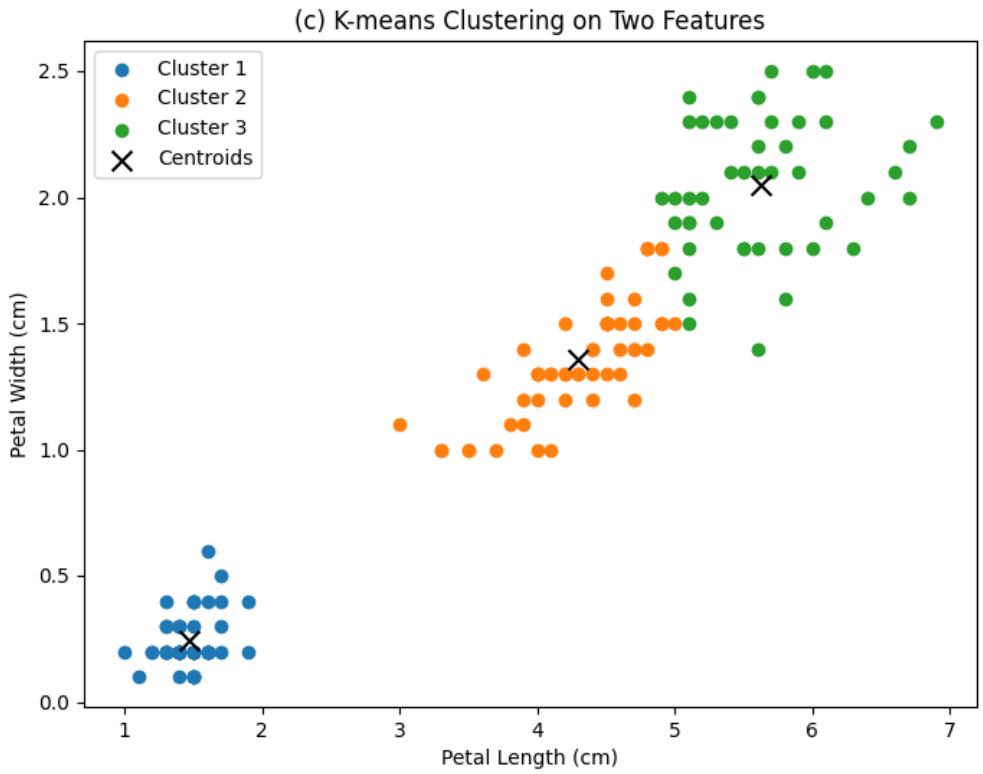
```
print("\n=== (b) Full Feature Set ===")
print("Confusion Matrix:")
print(conf_matrix_full)
print(f"\nAccuracy (Full Features): {accuracy_full * 100:.2f}%")

# (c) Apply k-means with only two features
X_two = data[['PetalLengthCm', 'PetalWidthCm']].values
clusters_two, centroids_two = k_means(X_two, k=3)
clusters_two = remap_clusters(true_labels, clusters_two)
conf_matrix_two = confusion_matrix(true_labels, clusters_two)
accuracy_two = accuracy_score(true_labels, clusters_two)

print("\n=== (c) Two Features ===")
print("Confusion Matrix:")
print(conf_matrix_two)
print(f"Accuracy: {accuracy_two * 100:.2f}%")
# (d) Compare results
print("\n=== (d) Comparison ===")
print(f"Accuracy with Full Features: {accuracy_full * 100:.2f}%")
print(f"Accuracy with Two Features: {accuracy_two * 100:.2f}%")

# Plotting for (c)
plt.figure(figsize=(8, 6))
for i in range(3):
    cluster_points = X_two[clusters_two == i]
    plt.scatter(cluster_points[:, 0],
                cluster_points[:, 1], label=f'Cluster {i+1}')
plt.scatter(centroids_two[:, 0], centroids_two[:, 1],
            color='black', marker='x', s=100, label='Centroids')
plt.xlabel('Petal Length (cm)')
plt.ylabel('Petal Width (cm)')
plt.title('(c) K-means Clustering on Two Features')
plt.legend()
plt.show()
```

Επίσης, ακολουθεί γραφική απεικόνιση του ζητούμενου.



Ερώτημα Δ

Χρησιμοποιώντας και τα 4 χαρακτηριστικά, το success rate ήταν 89.33%, ενώ με μόνο το μήκος και το πλάτος των πετάλων αυξήθηκε στο 94.67%. Αυτό δείχνει ότι τα χαρακτηριστικά των πετάλων είναι πιο διακριτικά για τις κλάσεις, ενώ η προσθήκη των χαρακτηριστικών των σεπάλων εισάγει θόρυβο και μειώνει την ακρίβεια.

Συνεπώς, η χρήση όλων των χαρακτηριστικών δεν οδηγεί πάντα σε καλύτερα αποτελέσματα, καθώς μπορεί να περιλαμβάνει περιττή ή παραπλανητική πληροφορία που επηρεάζει αρνητικά την ταξινόμηση.