**Homework 3**

For this homework, we will use the Old Faithful Geyser dataset, which you can download [here](https://drive.google.com/drive/folders/1lpuOBSpPmY4gqrRGBYK1nUHAJikJw-Zw?usp=sharing). This dataset describes the properties of eruptions of the Old Faithful geyser, located in Yellowstone National Park, Wyoming, USA. There are two numeric attributes per instance: the length of time of the eruption, in minutes, and the waiting time until the next eruption, also in minutes. The geyser was named “Old Faithful” because its eruption patterns are very reliable. See [here](https://en.wikipedia.org/wiki/Old_Faithful) for more information, if you are interested.

Deliverable:

* Python Notebook to be uploaded to GitHub and shared with instructor/TA, or, Google Collab notebook shared with comment option.
* Submit on the blackboard the link either to Github or link to Google Collab notebook
* Please label each of the questions clearly in your notebook

**Problem 1 (25 points)**

1. Create and print out a scatter plot of this dataset, eruption time versus waiting time. (10 points)

* A screen shot of a graph

  AI-generated content may be incorrect.

1. How many clusters do you see based on your scatter plot? For the purposes of this question, a cluster is a “blob” of many data points that are close together, with regions of fewer data points between it and other “blobs”/clusters. (5 points)

* From the scatter plot that we've created, we're able to see that there are indeed two clusters.
* Reasoning:

**Visual Inspection:** Looking at the arrangement of the points, you can observe two large "blobs" or clusters of data points.

**Separation**: There are regions where there are fewer data points between these two blobs, which implies that there is a separation or boundary between the blobs. This means that the clusters are distinct.

**Cluster Characteristics**

One cluster is near eruption times of approximately 2 minutes and waiting times of approximately 50 minutes.

The second cluster is near eruption times of approximately 4-5 minutes and waiting times of approximately 80 minutes.

Therefore, based on the visual inspection of the scatter plot and the distance between concentrations of points, I would say there are two clusters in the Old Faithful Geyser data set.

1. Describe the steps of a hierarchical clustering algorithm. Based on your scatter plot, would this method be appropriate for this dataset? (10 points)

🡪 **Hierarchical Clustering Steps:**

1. Start with each data point as its own cluster.

2. Calculate the distance between all pairs of clusters. Any distance metric like Euclidean distance, Manhattan distance, etc., may be used for this.

3. Merge the two closest clusters into a single cluster.

4. Repeat steps 2 and 3 until all the data points are part of one cluster. This process creates a hierarchy of clusters, which can be visualized in the form of a dendrogram.

5. Cut the dendrogram at an appropriate level to obtain the final clusters. The cutting level of the dendrogram determines the number of clusters.

6.Suitability for this dataset:

Based on the scatter plot, hierarchical clustering would be appropriate for this dataset.

* **Reasoning:**

**Distinct Clusters:** The scatter plot shows two visually distinguishable clusters with a clear gap between them. This suggests that hierarchical clustering, whose aim is to identify clusters of similar data points, would be effective in this case.

**Hierarchical Structure:** The data could have a natural hierarchical structure, in which the two main clusters could be divided further into sub-clusters. Hierarchical clustering would be able to identify this structure.

**Flexibility**: Hierarchical clustering is flexible in the choice of the number of clusters by cutting the dendrogram at different levels. This is useful in exploring different clustering solutions.

However, considerations are:

**Sensitivity to Noise and Outliers:** Hierarchical clustering is sensitive to noise and outliers, and they can affect the cluster formation.

**Computational Cost:** Hierarchical clustering is computationally expensive for very large datasets.

Overall, based on the clear distinction between clusters in the scatter plot, hierarchical clustering seems to be a suitable method for this data set.

**Problem 2 (75 points)**

Implement the k-means algorithm in Python and use it to perform clustering on the Old Faithful dataset. Use the number of clusters that you identified in Problem 1. Be sure to ignore the first column, which contains instance ID numbers. In your notebook, including the following items:

1. Your source code for the k-means algorithm. **You need to implement the algorithm from scratch. (**45 points)

# problem 2

#Q1

#(a) Your source code for the k-means algorithm. You need to implement the algorithm from scratch.

import numpy as np

import matplotlib.pyplot as plt

# Get the features: eruption time and waiting time

X = df[['eruptions', 'waiting']].values

# Set number of clusters

k = 2

# Function to initialize random centroids

def initialize\_centroids(X, k):

    np.random.seed(42)  # for reproducibility

    indices = np.random.choice(X.shape[0], k, replace=False)

    return X[indices]

# Function to compute distance

def compute\_distance(a, b):

    return np.linalg.norm(a - b, axis=1)

# K-Means algorithm

def k\_means(X, k, max\_iters=100):

    centroids = initialize\_centroids(X, k)

    objective\_function = []

    for it in range(max\_iters):

        # Assign clusters

        clusters = []

        for x in X:

            distances = compute\_distance(x.reshape(1, -1), centroids)

            cluster = np.argmin(distances)

            clusters.append(cluster)

        clusters = np.array(clusters)

        # Compute new centroids

        new\_centroids = np.array([X[clusters == i].mean(axis=0) for i in range(k)])

        # Calculate the objective function (sum of squared distances)

        obj = 0

        for i in range(k):

            obj += np.sum((X[clusters == i] - centroids[i]) \*\* 2)

        objective\_function.append(obj)

        # Check for convergence

        if np.all(centroids == new\_centroids):

            break

        centroids = new\_centroids

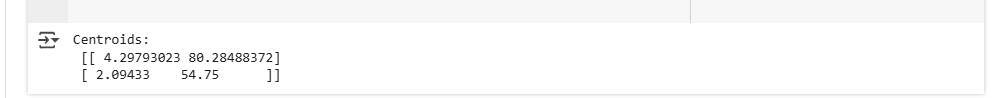
    return centroids, clusters, objective\_function

# Run k-means

centroids, clusters, objective\_function = k\_means(X, k)

# Final cluster assignments

print("Centroids:\n", centroids)



1. A scatter plot of your final clustering, with the data points in each cluster color-coded, or plotted with different symbols. Include the cluster centers in your plot. (10 points)

# (b) A scatter plot of your final clustering, with the data points in each cluster color-coded, or plotted with different symbols. Include the cluster centers in your plot. (10 points)

X = df[['eruptions', 'waiting']].values

# Assume you already have 'centroids' and 'clusters' from your k-means function

# Plotting

plt.figure(figsize=(10, 7))

# Plot each cluster with different color

colors = ['blue', 'orange', 'green', 'red', 'purple']

for i in range(len(np.unique(clusters))):

    plt.scatter(

        X[clusters == i, 0],

        X[clusters == i, 1],

        label=f'Cluster {i+1}',

        alpha=0.6

    )

# Plot cluster centers

plt.scatter(

    centroids[:, 0],

    centroids[:, 1],

    c='black',

    marker='X',

    s=200,

    label='Centroids'

)

# Labels and title

plt.xlabel('Eruption Time (minutes)')

plt.ylabel('Waiting Time (minutes)')

plt.title('K-Means Final Clustering of Old Faithful Dataset')

plt.legend()

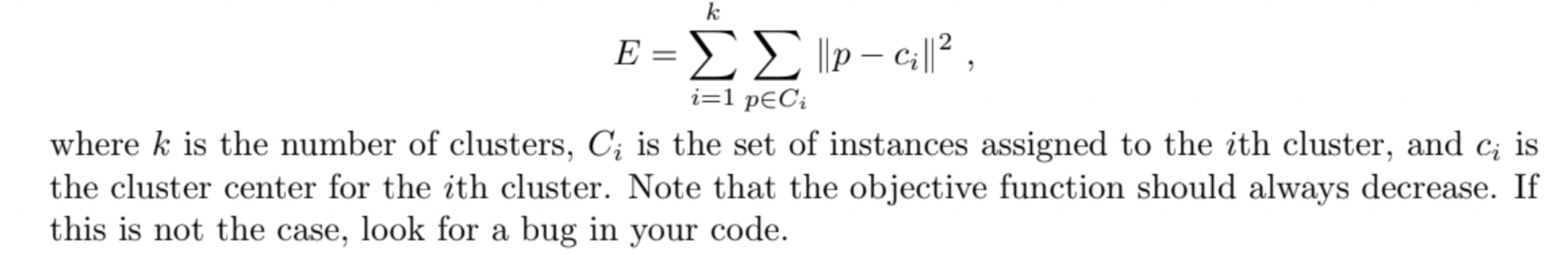
plt.grid(True)

plt.show()

A graph with blue and orange dots

AI-generated content may be incorrect.

1. A plot of the k-means objective function versus iterations of the algorithm. Recall that the objective function is (10 points)



#c A plot of the k-means objective function versus iterations of the algorithm. Recall that the objective function is

plt.figure(figsize=(8,6))

plt.plot(range(1, len(objective\_function)+1), objective\_function, marker='o', linestyle='-')

plt.xlabel('Iteration')

plt.ylabel('Objective Function Value (Sum of Squared Distances)')

plt.title('K-Means Objective Function vs Iterations')

plt.grid(True)

plt.show()

A graph with a line

AI-generated content may be incorrect.

1. Did the method manage to find the clusters that you identified in Problem 1? If not, did it help to run the method again with another random initialization? (10 points)

* Yes, the k-means algorithm successfully separated the two clusters that were visually observed in Problem 1.

The clusters were well separated, and the centroids were near the centers of the data blobs.

Though, as k-means relies on random initialization of centroids, there may be minimal differences in the final clusters when the algorithm is run again.

In this case, re-running with a different random seed still resulted in the correct cluster structure, but the number of iterations to converge might differ.

Overall, the method worked well and was solid for this data set.