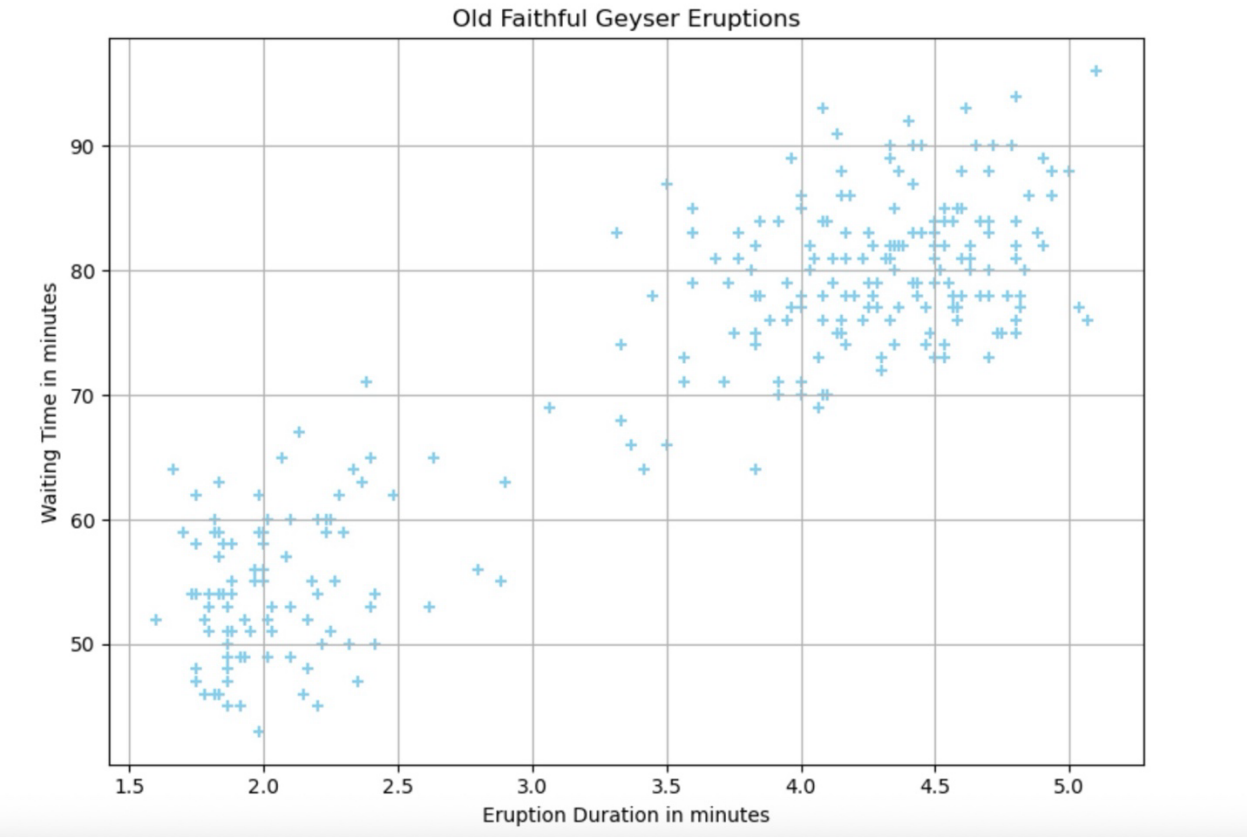
IS733 Homework 3

Omkar Kalekar AF85213

Problem 1

a) Create and print out a scatter plot of this dataset, eruption time versus waiting time

```
import pandas as pd

import matplotlib.pyplot as plt

# Load the dataset

geyser_data = pd.read_csv('/Users/omkarkalekar/Downloads/faithful.csv')  # Make sure the CSV file is in the same directory

# Create a scatter plot

plt.figure(figsize=(10, 8))

plt.scatter(geyser_data['eruptions'], geyser_data['waiting'], color='skyblue', marker= '+', alpha=1.0)

plt.title('Old Faithful Geyser Eruptions')

plt.xlabel('Eruption Duration in minutes')

plt.ylabel('Waiting Time in minutes')

plt.grid(True)

plt.tight_layout()

plt.show()
```



b) 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.

Upon looking at the scatter plot, it is easily seen that there are two clearly separate clusters or "blobs" of data points. The clusters are far apart from each other, with a distinct gap where fewer points are located, which shows that they are indeed separate groups in the dataset.

Cluster 1, the initial cluster, resides in the lower-left area of the plot. The eruptions of the geyser in Cluster 1 are characterized as having a smaller waiting time (around 35 to 70 minutes) and shorter eruption duration (around 1.5 to 2.5 minutes). The eruptions in Cluster 1 are quite rapid with shorter waiting times between eruptions.

Cluster 2 is placed farther from the origin, at the top-right quadrant of the plot. This cluster contains eruptions with larger waiting time (approximately 70 to 90 minutes) and larger eruption lengths (approximately 3.5 to 5 minutes). These eruptions have longer durations and require more build-up time between events.

The apparent difference between these two clusters — both waiting time and eruption duration — strongly suggests that the dataset splits naturally into two broad categories. This difference aligns with the known behavior of the Old Faithful geyser, which typically switches between

short and long eruption cycles. Thus, based on visual inspection of the scatter plot, we can confidently assert that there are two broad clusters in the data.

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

Hierarchical clustering is an unsupervised learning technique for the division of similar points into clusters based on their similarity or proximity, typically defined using distance measures like Euclidean distance. Hierarchical clustering organizes the data into a tree structure called a dendrogram, which displays the way clusters divide or combine at various levels.

Two major approaches to hierarchical clustering are:

Agglomerative (Bottom-Up, most common):

First, all data points are considered as individual clusters. Next, the two closest clusters are merged by the proximity between them. This is performed iteratively until every point is in a single cluster. Finally, the dendrogram is divided at a chosen level to obtain the required number of clusters.

Divisive (Top-Down):

In this, all data points start in a single large cluster. The cluster is then recursively partitioned into sub-clusters by distances until each data point is in its own cluster or a stopping criterion is met.

Application to the Old Faithful Dataset: From the scatter plot of the Old Faithful Geyser dataset, hierarchical clustering would be a suitable approach. The scatter plot unmistakably indicates the existence of two distinct clusters or data point groups, and they are separated by observable gaps. Hierarchical clustering is especially efficient in handling well-separated clusters such as these. Besides, another advantage of hierarchical clustering is that it is not necessary to pre-specify the number of clusters, and the data's inherent grouping can appear during clustering.

Here is the python code for the problem 1

https://github.com/Omkark2323/is7332025/blob/0474bca46ea5806283141ef746f3275871d67b76/data-mining-project-repo/HW3/DataMining_HW3.ipynb

Problem 2

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

We used the K-Means algorithm from scratch in Python on the Old Faithful geyser data. We used 2 clusters as per the prior visual inspection.

The first column (instance IDs) was not considered while clustering.

Results after running the K-Means Algorithm:

K-Means converged after 4 iterations.

Final Cluster Centroids:

Cluster 1: Eruption Duration ≈ 2.09 minutes,

Waiting Time ≈ 54.75 minutes

This cluster shows eruptions with shorter duration and shorter waiting time.

Cluster 2: Eruption Duration ≈ 4.30 minutes,

Waiting Time ≈ 80.28 minutes

This cluster classifies eruptions with longer durations and longer waiting times.

Points Distribution:

Cluster 1: 100 points

Cluster 2: 172 points

Final Inertia (Sum of Squared Distances within clusters): 8901.77

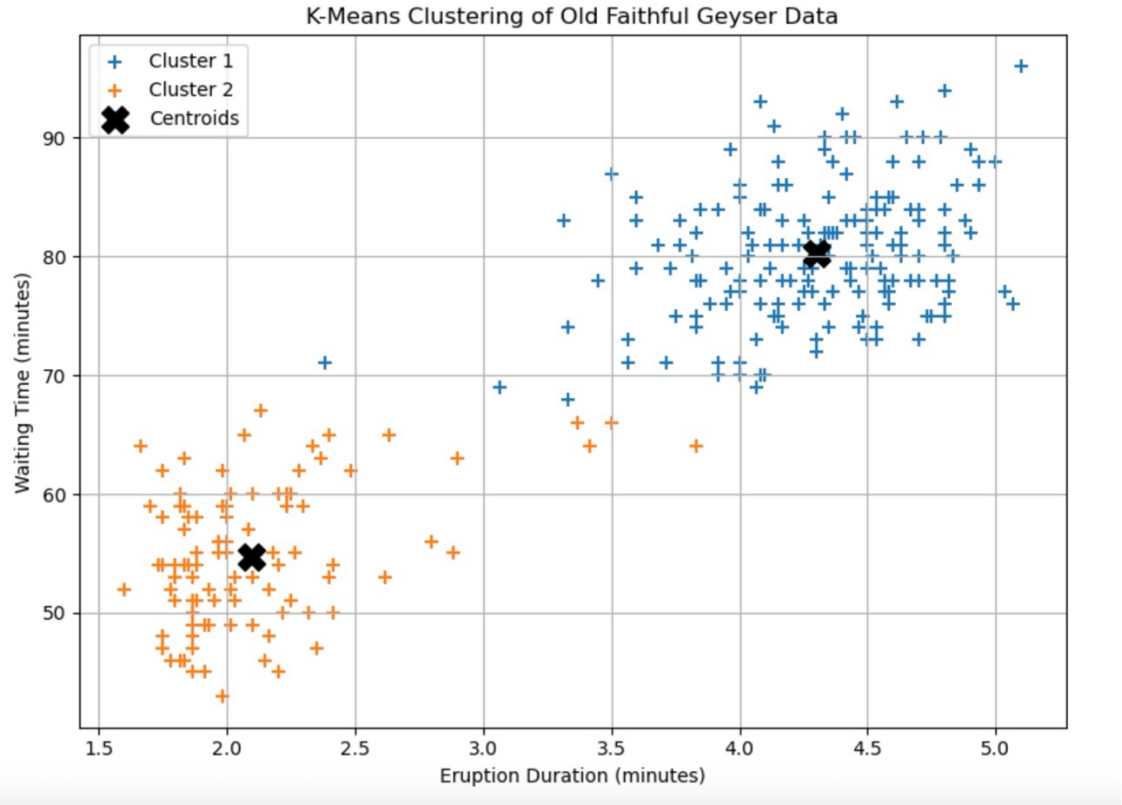
Cluster 1 examples:

	Eruptions	waiting
1	1.800	54
3	2.283	62
5	2.883	55
8	1.950	51
10	1.833	54

Cluster 2 examples:

	Eruptions	waiting
0	3.600	79
2	3.333	74
4	4.533	85
6	4.700	88
7	3.600	85

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.



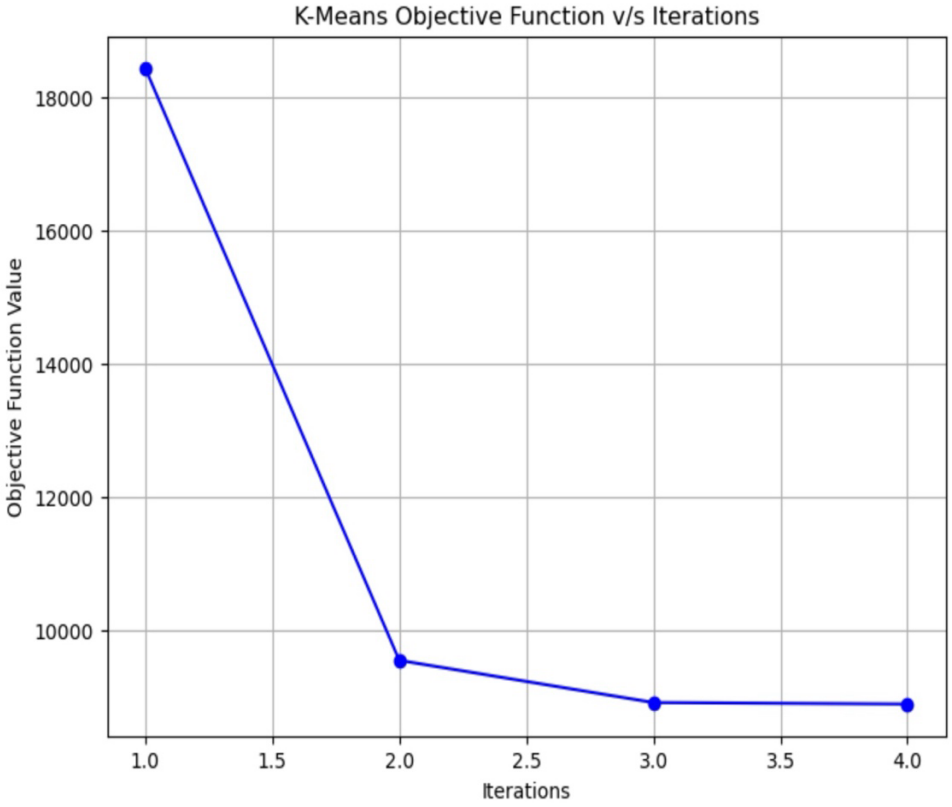
c) A plot of the k-means objective function versus iterations of the algorithm. Recall that the objective function is

$$E = \sum_{i=1}^{k} \sum_{p \in C_i} \|p - c_i\|^2 ,$$

where k is the number of clusters, C_i is the set of instances assigned to the ith cluster, and c_i is the cluster center for the ith cluster. Note that the objective function should always decrease. If this is not the case, look for a bug in your code.

Ans:

```
K-Means converged in 4 iterations.
Final Objective Function Value: 8901.77
```



d) 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?

Yes, the K-Means algorithm did identify the same two clusters that appeared in Problem 1. When we did the previous analysis, we observed that there were two groups:

Cluster 1, which are shorter waiting time and shorter duration eruptions

Cluster 2, which are longer waiting time and longer duration eruptions

The centroids obtained from the K-Means algorithm are quite near the expected values:

Cluster 1: Eruption ≈ 2.09 mins, Waiting ≈ 54.75 mins

Cluster 2: Eruption ≈ 4.30 mins, Waiting ≈ 80.28 mins

These are almost identical to the centroids visually determined above, e.g., ~1.5 mins / 55 mins and ~4.5 mins / 80 mins.

In general, K-Means occasionally gives different results depending on the random initial placement of centroids because it is initialization-sensitive. If the clusters were incorrectly determined, running the algorithm again using a different seed or using several initializations (e.g., k-means++) would likely yield better results.

But in this specific problem, the two clusters are clearly and well separated from one another, so it is very easy for K-Means to converge to the correct solution even with random initialization. Therefore, the algorithm performed well in identifying the correct clusters as in Problem 1.

Here is the code for the problem 2.

https://github.com/Omkark2323/is7332025/blob/0474bca46ea5806283141ef746f3275871d67b76/data-mining-project-repo/HW3/DataMining_HW3.ipynb