**Experiment 6**

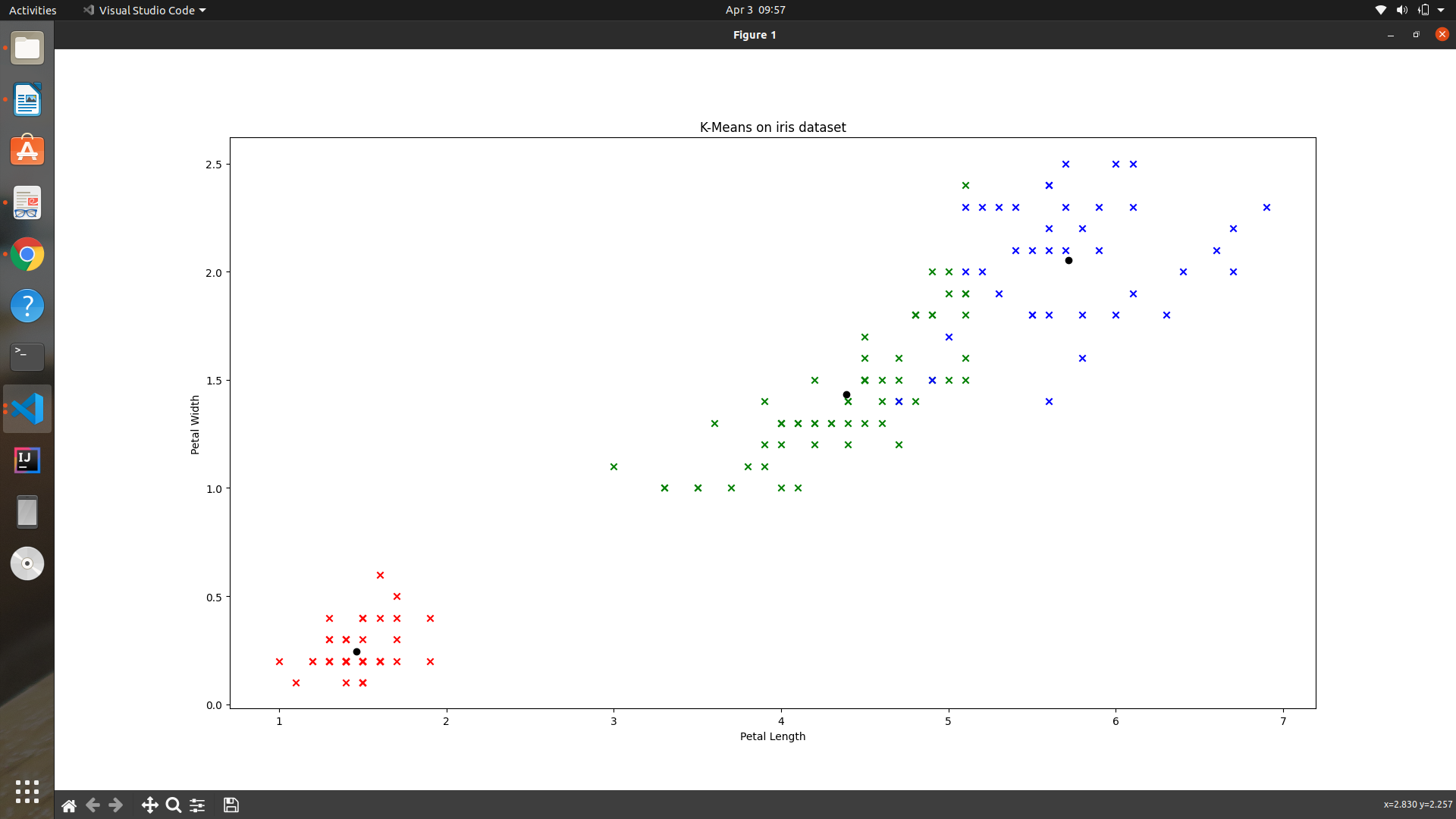
**Aim: To implement K-means and K-medoids algorithm on the Iris dataset.**

**Code:**

1. **K-means**

|  |
| --- |
| **import** **pandas** **as** **pd**  **import** **numpy** **as** **np**  **import** **matplotlib.pyplot** **as** **plt**  **import** **random**  iris\_data\_path = "./iris.data"  iris\_dataset = pd.read\_csv(iris\_data\_path, header=**None**)  iris\_dataset.pop(**4**)  K = **3**  # lets generate k random indices and then apply this algo  k\_random\_indices = [**0**, **51**, **101**]  centroids = [iris\_dataset.loc[i] **for** i **in** k\_random\_indices]  max\_iterations = **1000**  tol = **0.000001**  clusters = [[] **for** \_ **in** range(K)]  **for** i\_iter **in** range(max\_iterations):  clusters = [[] **for** \_ **in** range(K)]  **for** i **in** range(iris\_dataset.shape[**0**]):    # find dists from all the centroids  dists\_from\_centroid = [np.linalg.norm(iris\_dataset.loc[i] - centroids[j]) **for** j **in** range(**0**, K)]  # get the min dist from centroid  min\_dist\_idx = dists\_from\_centroid.index(min(dists\_from\_centroid))  clusters[min\_dist\_idx].append(iris\_dataset.loc[i])  prev\_centroids = centroids  centroids = [np.average(cluster, axis=**0**) **for** cluster **in** clusters]  # if optimized, then break else go for another round  is\_optimized = **True**  **for** i **in** range(K):  original\_centroid = prev\_centroids[i]  curr\_centroid = centroids[i]  diff = np.sum((curr\_centroid - original\_centroid) / original\_centroid \* **100.0**)  print(diff)  **if** diff > tol:  is\_optimized = **False**  **if** is\_optimized:  print(f'{i\_iter} iterations to find the optimal solutions')  **break**  colors = ["r", "g", "b"]  ######### mark the centroids  **for** i **in** range(len(centroids)):  plt.scatter(centroids[i][**2**], centroids[i][**3**], color= "k", marker="o")  i = **0**  **for** cluster **in** clusters:  color\_for\_this\_cluster = colors[i]  i += **1**  **for** point **in** cluster:  plt.scatter(point[**2**], point[**3**], color = color\_for\_this\_cluster, marker="x")  plt.title("K-Means on iris dataset")  plt.xlabel("Petal Width")  plt.xlabel("Petal Length")  plt.show() |

**Output:**

****

**2. K-medoids:**

|  |
| --- |
| **import** **pandas** **as** **pd**  **import** **numpy** **as** **np**  **import** **random**  **import** **matplotlib.pyplot** **as** **plt**  iris\_dataset = pd.read\_csv('iris.data', header=**None**)  iris\_dataset.pop(**4**)  # get initial KMediods  K = **3**  max\_iter = **1000**  initial\_medoids\_indices = {random.choice(range(**0**, **150**)) **for** \_ **in** range(K)}  initial\_medoids = [iris\_dataset.loc[idx] **for** idx **in** initial\_medoids\_indices]  non\_mediods\_data = []  **for** i **in** range(**150**):  **if** i **not** **in** initial\_medoids\_indices:  non\_mediods\_data.append(iris\_dataset.loc[i])  **def** **swap**(medoids, non\_medoids, med\_idx, non\_med\_idx):  temp = non\_medoids[non\_med\_idx]  non\_medoids[non\_med\_idx] = medoids[med\_idx]  medoids[med\_idx] = temp  **def** **compute\_cost**(medoids, non\_medoids):  clusters = [[] **for** \_ **in** range(len(medoids))]  cost = **0**  **for** non\_med **in** non\_medoids:  dists = [np.linalg.norm(non\_med - medoid) **for** medoid **in** medoids]  min\_dist\_idx = dists.index(min(dists))  cost += dists[min\_dist\_idx]  clusters[min\_dist\_idx].append(non\_med)  **return** cost, clusters  init\_cost, init\_clusters = compute\_cost(initial\_medoids, non\_mediods\_data)  i\_iters = **0**  **while** **True**:  i\_iters += **1**  # print(f'Non Medoids = {non\_mediods\_data}')  med\_copy = initial\_medoids.copy()  non\_med\_copy = non\_mediods\_data.copy()  med\_swap\_idx = random.randint(**0**, K-**1**)  non\_med\_swap\_idx = random.randint(**0**, len(non\_med\_copy) - **1**)  swap(initial\_medoids, non\_mediods\_data, med\_swap\_idx, non\_med\_swap\_idx)  new\_cost, new\_clusters = compute\_cost(initial\_medoids, non\_mediods\_data)  **if**(new\_cost > init\_cost):  initial\_medoids = med\_copy  non\_mediods\_data = non\_med\_copy  **break**  init\_clusters = new\_clusters  init\_cost = new\_cost  colors = ["r", "g", "b"]  ######### mark the centroids  **for** i **in** range(len(initial\_medoids)):  plt.scatter(initial\_medoids[i][**0**], initial\_medoids[i][**1**], color= "k", marker="o")  i = **0**  **for** cluster **in** init\_clusters:  color\_for\_this\_cluster = colors[i]  i += **1**  **for** point **in** cluster:  plt.scatter(point[**0**], point[**1**], color = color\_for\_this\_cluster, marker="x")  print(f'Number of iterations = {i\_iters}')  plt.title("K-Medoids on iris dataset")  plt.xlabel("Sepal Length")  plt.xlabel("Sepal Width")  plt.show() |

