Exercise 3

Question 2

Instructions for running the code is given in the README.txt file in the Assignment3 folder.

You can also have look at the implementation of the code explained in pdf file named "Kmeans and iris dataset.pdf"

Code Explanation

For implementing Kmeans clustering first we define class Kmeans with constructor containing arguments.

- n_clusters : Number of clusters or lets say value of K
- max_iter: Maximum iteration that needs to be done for the data.
- random_state: denotes the state
- cluster_centers_: Coordinates of the centroids of clusters
- Inertia_: sum of squared distances from centroid and sample point.
- labels_: labels for each point in the dataset

Note that the name of the variables was taken from the scikit learn api.
https://scikit-learn.org/stable/modules/generated/sklearn.cluster.KMeans.html

Algorithm:
The class contains the **fit** method.
Fit method implementation contains the whole algorithm
Step1: **Random initialization of centroids**
The fit method first takes the data and select random data points as centroid and assigns it to cluster_centers_ attribute.
To randomly select the centroids, we define **random_centroid_initialization** method.
In this method we first generate a random centroids by generating permutation of random index and then choosing centroids from dataset here X equal to the n_clusters.

After step 1, we loop through max_iter and do the below steps for each iteration
Step2: Calculating distance of each point from cluster
Distance between each point and centroid is calculated using **calculate_distance_each_point** method.
This method calculated the euclidean distance between the data point and centroid and appends it in the distances list.

Step3: Find the closest cluster to data point and label it
Finding the closest cluster to data point is done using **closest_cluster** method which calculates minimum distance.
Then we label the data point into the cluster of minimum distance chosen.

Step4: After labelling is done, recalculate the new centroids because of addition of a new data point to the cluster

Since data point is added to the cluster we calculate the new centroids by taking the centroid or average of all the points in a cluster using **calculate_cluster_centers_** Method.
If the centroids converge then break out of the loop.
Calculate inertia_ using the **calculate_inertia_** method. This method calculates the sum of squared distances of sample data points to their closest centroid.

Also implemented the predict method
predict method will return the closest cluster

This is how we defined the class Kmeans_custom.

Now implementing it on the iris dataset. For this I first load the dataset from sklearn.datasets using load_iris method

Now we create a dataframe of data i.e. iris.data. Now for visualization of the iris dataset I ran some codes to get insights into the data.
Some of the functions that were used info(), head(), describe(). Later I plotted histogram for each feature.
Then I plotted heatmap for correlation values between the four features using seaborn library.

Answer a will continue after answer b
Answer b)
Before implementing kmeans on the dataset, I implemented Elbow method to find the best value of K.
For implementing this method I created a class ElbowMethod with inertias as an empty list.

Later to calculate inertias for different values of k I used **calculate_inertias method** to loop through different values of k here 1 to 10 and implement the Kmeans algorithm with iris dataset and append the inertia calculated for each iteration into the self.inertias attribute.

For plotting the curve of inertias over 1 to 10 values of k I used matplotlib pytplot function to plot the inertias value where x = range(1, 10) and y=self.inertias.
For plotting the elbow method I defined the plot_elbow_curve method.

After plotting, it can be easily inferred that 3 was the best value of k.

3 is the best value because from k=3 to k=4 there is not much drop in weights of the inertias but from k=2 to k=3 there is a large drop. Hence k=3 was chosen.

Back to answer a)
Now after the value of k is calculated using Elbow method, I first create an instance of our local Kmeans_custom method with n_clusters=3 and max_iter=100 and random_state=42
Now I trained the dataset on the above instance with instance.fit(dataset) method.
After the model is trained, I plot the scatter plot for each label 0, 1 and 2 with filter instance.labels_
So for getting the points for label 0
x = X[km.labels_== 0, 0]
Y = X[km.labels_== 0, 1]
Similarly we did filter for other labels 1 and 2 by changing km.labels_ == 1 and km.labels_ == 2
This way we plot the data points according to the labels.

For plotting the centroids, I took use of the km.cluster_centers_ values.
I just plotted the cluster_centers_ values for x and y i.e 0 and 1 axis. I.e for 2 dimensions.

After you run the kmeans_iris_dataset.py file, figures will be plotted and shown each one by one.

Fig 1. Histogram for each feature.

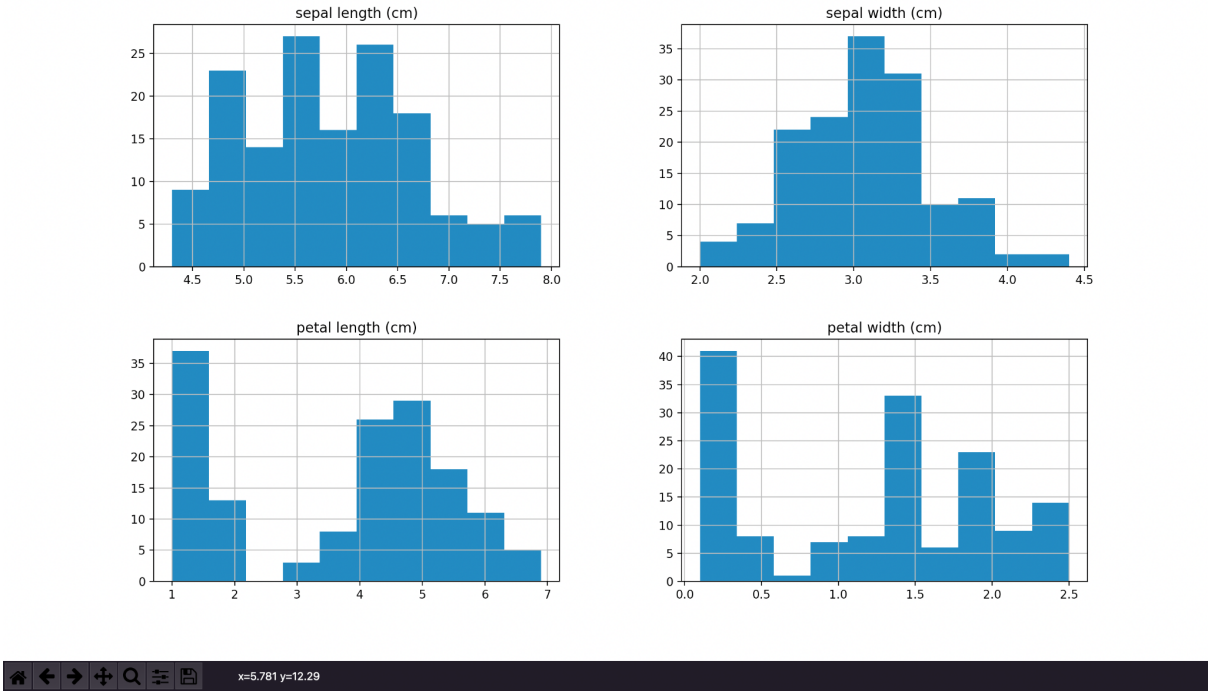

Fig 2. Heatmap of correlation between the features

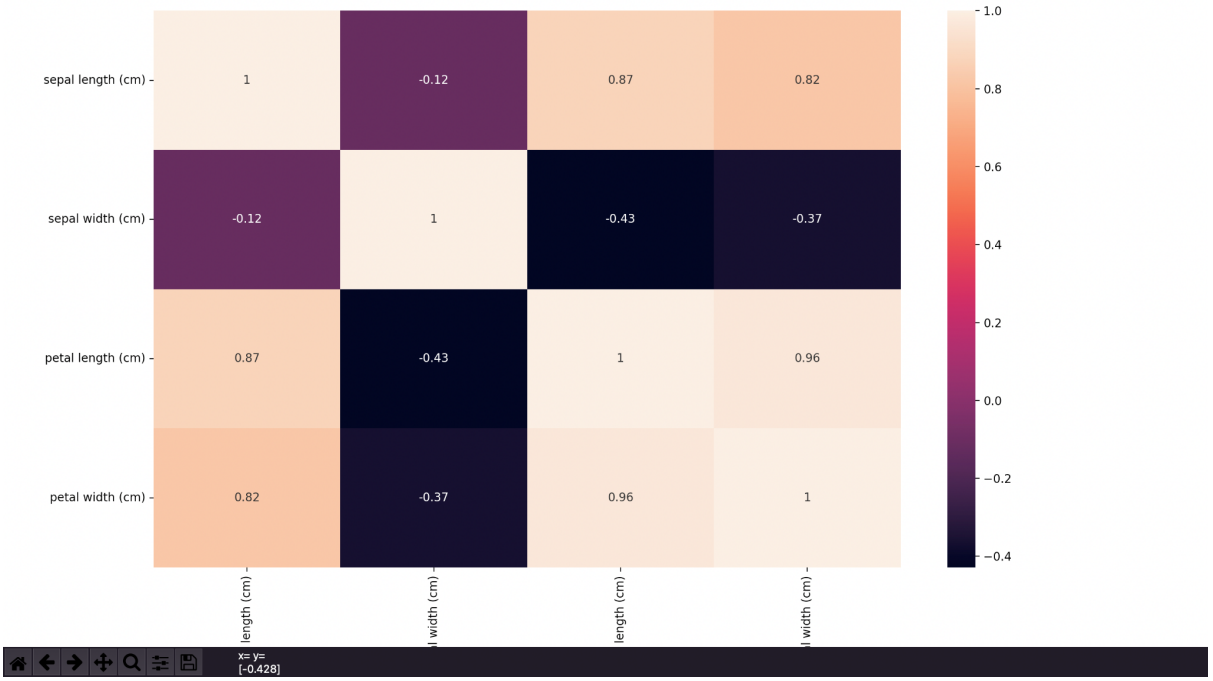

Fig 3. Elbow method

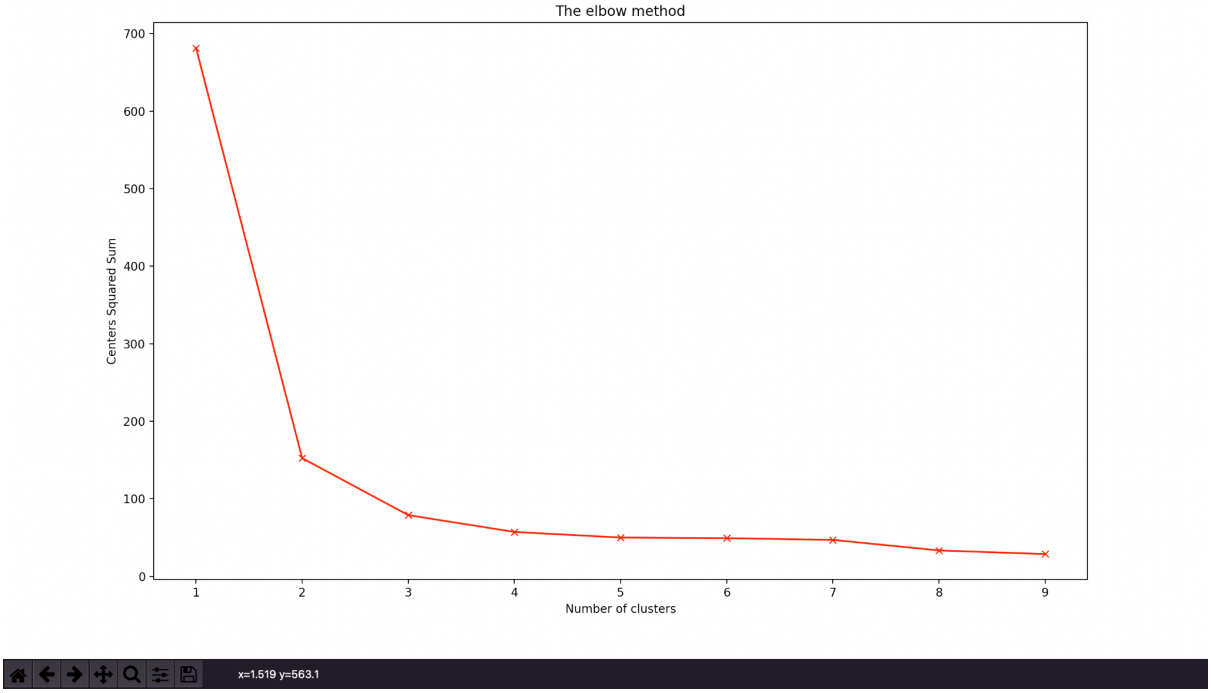

Fig 4. Scatter plot for data points and centroids