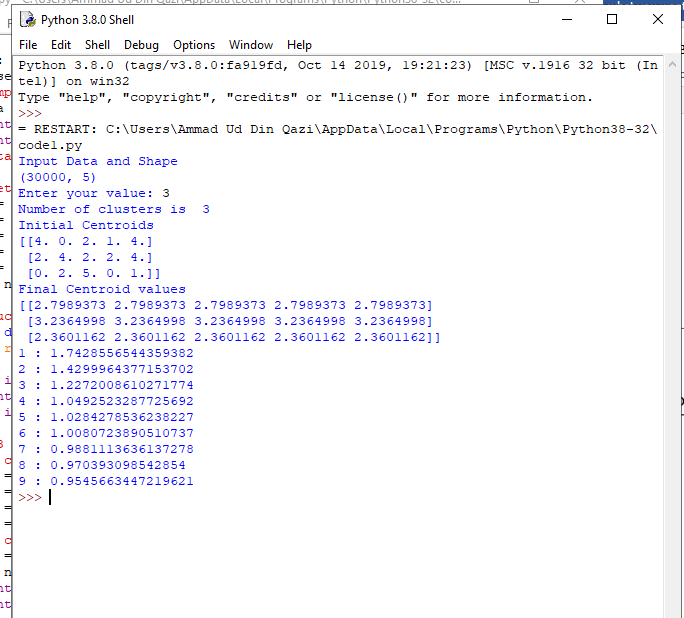
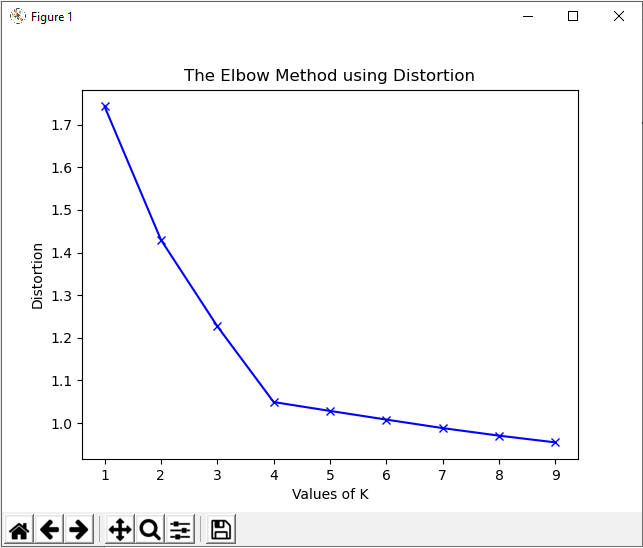
**K -MEANS CLUSTERING REPORT**

**PRACTICAL PART**

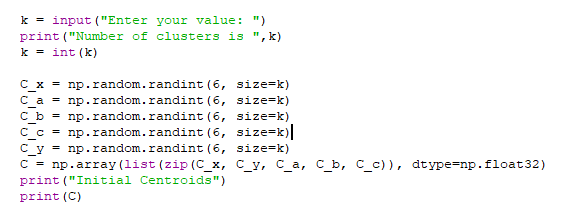
**OUTPUT OF CODE**

****

****

**EXPLANATION**

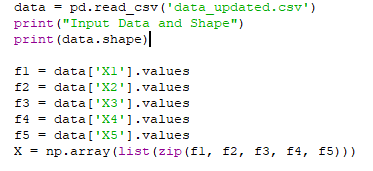
**STEP 1**

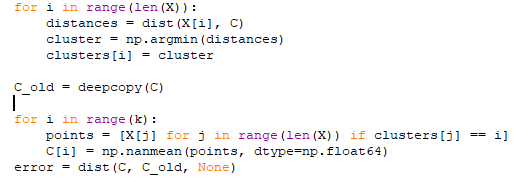
****

As specified in the assignment I have taken input from the user about the number of clusters he wants to choose for k-means clustering. I have converted the input to int by int() to prevent errors as it was being read as string.

Np.random.randint is used to generate random value in the range from 0 to 5 and since number of centroids is k each dimension should have k random values.

I then combine all these arrays into a single array and have printed the initial centroids generated.  
  
**STEP 2**

****

****

I am reading data from csv using panda. I have assigned each column a title to make data reading more easy. I have then combined all columns to form a single array.

Now for each data point I calculate the Euclidean distance by passing the data point and the centroids with which Euclidean distance will be calculated. Clusters represents the cluster labels. I am then assigning the data points with minimium distance to a specific cluster to their respective clusters by using np.argmin(distances) and putting it in clusters. The old centroids are saved in c\_old.

I then take average of all points placed in a cluster label to find new centroids. I do this by first placing all the data points in points and then calculating mean by using np.mean**. HERE I USED np.nanmean** **BECAUSE THERE WAS AN ERROR WITH np.mean which was only resolved by using np.nanmean. I found this solution from stack overflow.**

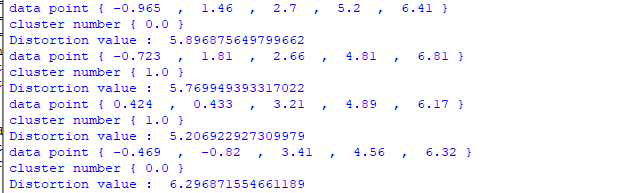
**STEP 3**

while error != 0:  
error = dist(C, C\_old, None)

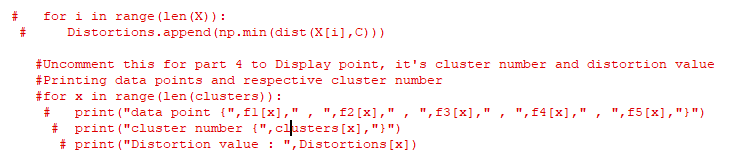
After mean I calculate euclidean distance of centroids with old centroids. If old centroids are same as new centroids this means that there is no change in cluster labels and no data point has changed its cluster which means we have reached the solution. Therefore the condition for my loop will be till error !=0. Since when distance is equal old and new centroids are same.

**STEP 4**

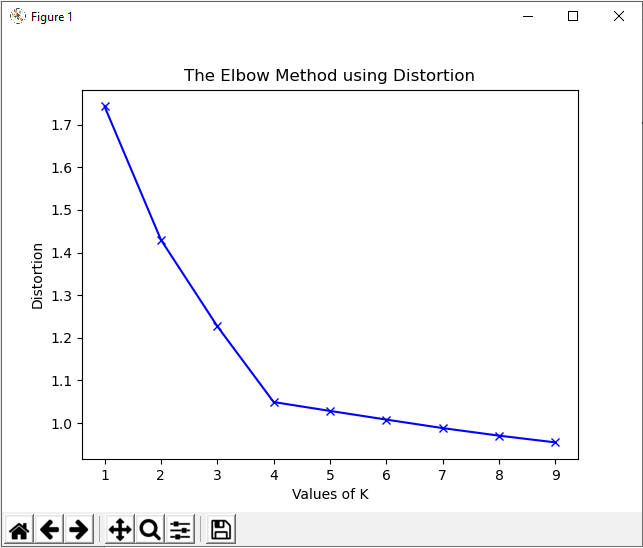
The commented code part in my code is used to print each point with its assigned cluster number and distortion value. Since dataset has 30,000 values this code takes a lot of time to run so I have commented it. You can uncomment to verify result. The below image is the output if you uncomment it.



Also uncomment Distortions declaration at start before running.



**STEP 5**

****

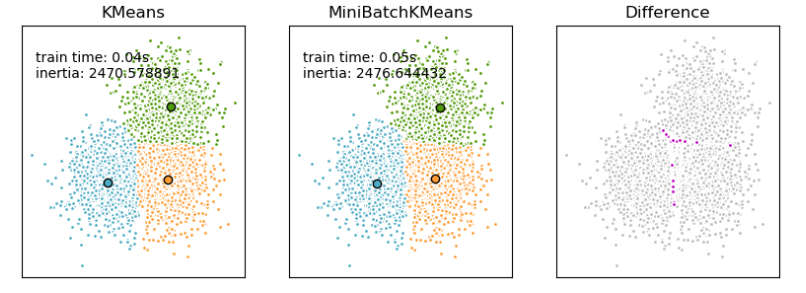
From my elbow plot the optimal number of clusters is 4 since after k value of 4 the decrease in distortion is linear. I used a range of k values till 10. The elbow plot is called so because at the point at which elbow forms that is optimal number of clusters. The elbow method basically looks at the percentage of variance explained as a function of number of clusters. The point at which variance is linear indicates that adding another cluster won’t give better modelling of data.

**THEORATICAL PART**

**Mini Batch K-means Vs K-means**

Mini Batch K-means is similar to K-means except the fact that the most computationally costly step is conducted on only a random sample of observations as opposed to all observations. This cause mini batch to find result in less time in k-means but however there is a compromise in quality.

Mini Batch basically has the **batch\_size** parameter which controls the number of randomly selected observations in each batch. Its main idea is to use small random batches of examples of a fixed size so they can be stored in memory. Each iteration a new random sample from the dataset is obtained and used to update the clusters and this is repeated until convergence. It applies a learning rate that decreases with the number of iterations. As the number of iterations increases, the effect of new examples is reduced, so convergence can be detected when no changes in the clusters occur in several consecutive iterations which is a problem of mini batch. Increasing the number of clusters, decreases the similarity of the mini batch K-means solution to the K-means solution. Despite that the agreement between the partitions decreases as the number of clusters increases, the objective function does not degrade at the same rate. It means that the final partitions are different, but closer in quality.



Mini Batch should be used when data set is very large as the saving in computational time is more noticeable only when the number of clusters is very large. When we have a large dataset and quality is not the utmost priority mini batch is preferred as we will obtain result more quickly. In cases where quality is the top most priority k-means will be used.

**Limitations of using K-means algorithm**

1. Choosing K manually

The user has to specify k (the number of clusters) in the beginning. K value can be found out by elbow plot however we do not have a fixed data set and therefore we don’t know if k is a static number. Like the data, it may change over time and we have to check for the optimal k periodically. Previous domain knowledge about the data is required as well making it difficult to predict k value.

1. Initial seeds have a strong impact on the final results

For a low k, you can mitigate this dependence by running k-means several times with different initial values and picking the best result. As k increases, you need advanced versions of k-means to pick better values of the initial centroids (called **k-means seeding**).

1. Doesn’t work with global cluster and clusters of different size and different density

K-means has trouble clustering data where clusters are of varying sizes and density. To cluster such data, you need to generalize k-means. Rescaling your datasets (normalization or standardization) will completely change results.

1. Clustering outliers

Centroids can be dragged by outliers, or outliers might get their own cluster instead of being ignored. Consider removing or clipping outliers before clustering.

5. Scaling with number of dimensions.

As the number of dimensions increases, a distance-based similarity measure converges to a constant value between any given examples. Reduce dimensionality either by using PCA on the feature data, or by using “spectral clustering” to modify the clustering algorithm as explained below.

1. The order of the data has an impact on the final results