TASK: K-Means Clustering Assignment.

Student’s Name

Institutional Name

Instructor’s Name

Due Date

**K-MEANS CLUSTERING ASSIGNMENT**

Cluster analysis is based on various kinds of object’s differences and used distance functions to make model classification. KMeans clustering is an unsupervised algorithm which is an interactive method that segments data into k clusters in which each observation belongs to the cluster with the nearest mean (cluster centroid). It is commonly used in data mining and pattern recognition.

Steps in applying KMeans Algorithm.

1. Import required libraries
2. Prepare data for plotting
3. Plot distribution of data points in the data
4. Scale the data
5. Find optimal number of clusters
6. Applying KMeans on the data
7. Plot cluster of label 0
8. Plot all KMeans clusters
9. Plot the cluster centroids
10. Import libraries
11. *# Import libraries*
12. from sklearn import metrics
13. from sklearn.cluster import KMeans
14. from sklearn.datasets import make\_blobs
15. from sklearn.preprocessing import StandardScaler, MinMaxScaler
16. from matplotlib import pyplot as plt
17. from scipy.spatial.distance import cdist
18. import numpy as np
19. import seaborn as sns
20. sns.set\_style()

2. Prepare data for plotting

*# initialize MinMax scaler : feature range = (0, 1)*

scaler = MinMaxScaler(feature\_range=(0, 1))

*# fit the scaler on X and transform*

X\_scaled = scaler.fit\_transform(X)

3. Plot distribution of data points in the data

*# visualize the dataset*

plt.plot()

plt.xlim([0, X\_scaled.shape[1]])

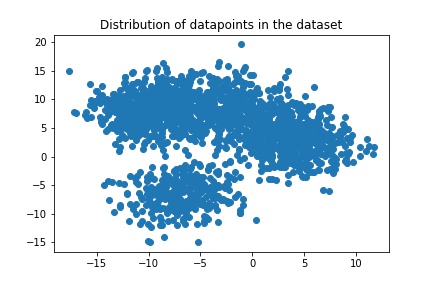
plt.ylim([0, X\_scaled.shape[1]])

plt.title('Dataset')

plt.scatter(X\_scaled[:, 0], X\_scaled[:, 1])

plt.savefig("dataset\_plot.jpg")

plt.show()



1. Scale the data
2. *# initialize MinMax scaler : feature range = (0, 1)*
3. scaler = MinMaxScaler(feature\_range=(0, 1))
4. *# fit the scaler on X and transform*
5. X\_scaled = scaler.fit\_transform(X)

4. Find optimal number of clusters

Elbow method

distortios = []

inertias = []

mapping1 = {}

mapping2 = {}

K = **range**(1, 10)

for k in K:

    kmeanModel = KMeans(n\_clusters=k).fit(X\_scaled)

    kmeanModel.fit(X\_scaled)

    distortios.append(**sum**(np.min(cdist(X\_scaled, kmeanModel.cluster\_centers\_, 'euclidean'), axis=1)) / X\_scaled.shape[0])

    inertias.append(kmeanModel.inertia\_)

    mapping1[k] = **sum**(np.min(cdist(X\_scaled, kmeanModel.cluster\_centers\_, 'euclidean'), axis=1)) / X\_scaled.shape[0]

    mapping2[k] = kmeanModel.inertia\_

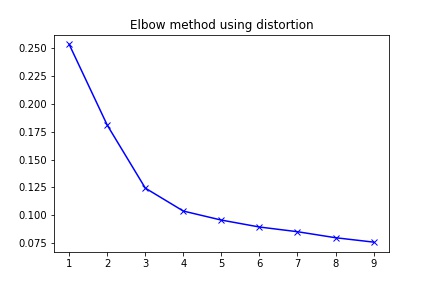
visualize distortion elbow

plt.plot(K, distortios, 'bx-')

plt.title("Elbow method using distortion")

plt.savefig("distortion-elbow\_plot.jpg")

plt.show()

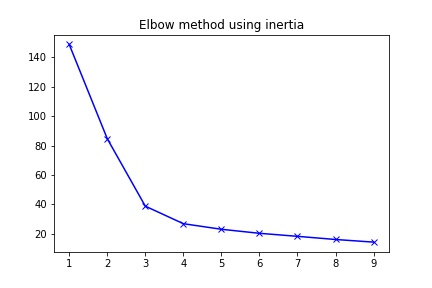


visualize inertia elbow

plt.plot(K, inertias, 'bx-')

plt.title("Elbow method using inertia")

plt.savefig("inertia-elbow\_plot.jpg")



To determine the optimal number of clusters, select the value of k at the elbow ie 3 as from the plots.

The elbow is the point where distortion or inertia start.

5. Applying KMeans on the data

*# Initialize the Kmeans object with k = 3*

kmeans = KMeans(n\_clusters=3)

*# fit the model and predict the classes*

label = kmeans.fit\_predict(X)

6. Plot cluster of label 0

*# visualize cluster with label 0*

label0 = X[label == 0]

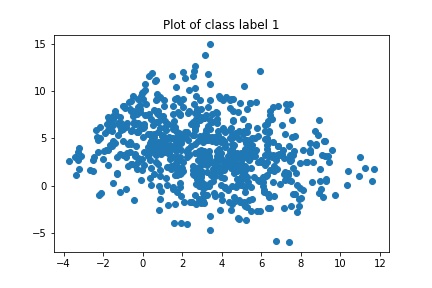
*# plotting results*

plt.scatter(label0[:, 0], label0[:, 1])

plt.title("Plot of class label 1")

plt.savefig("label0\_plot.jpg")

plt.show()



7. Plot all KMeans clusters

*# Plot all clusters*

u\_labels = np.unique(label)

*# plotting results*

for i in u\_labels:

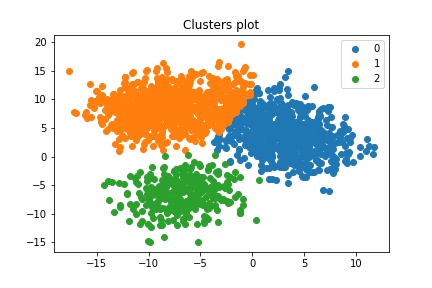
    plt.scatter(X[label == i, 0], X[label == i, 1], label=i)

plt.legend()

plt.title("Clusters plot")

plt.savefig("cluster\_plot.jpg")

plt.show()



8. Plot the cluster centroids

*# Plotting cluster centroids*

*# Get the centroids*

centroids = kmeans.cluster\_centers\_

ulabels = np.unique(label)

*# plotting results*

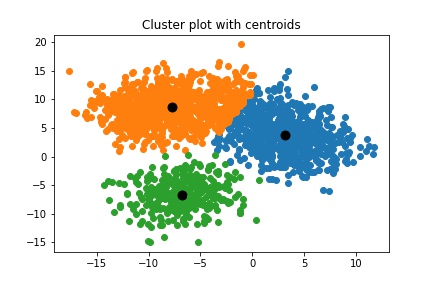
for i in u\_labels:

    plt.scatter(X[label == i, 0], X[label == i, 1], label=i)

plt.scatter(centroids[:, 0], centroids[:, 1], s=80, color='k')

plt.title("Cluster plot with centroids")

plt.savefig("centroid\_plot.jpg")



References

A clustering Method Based on K-Means Algorithm – Young Li, Haiwan Wu, 2012

Adaptive K-Means Clustering – Sanjiv K. Bhatia, 2004