DEPARTMENT OF ELECTRICAL AND COMPUTER ENGINEERING

Homework Assignment No. 08:

HW No. 08: K-Nearest Neighbors and K-Means Clustering

submitted to:

Professor Joseph Picone
ECE 8527: Introduction to Pattern Recognition and Machine Learning
Temple University
College of Engineering
1947 North 12th Street
Philadelphia, Pennsylvania 19122

March 18th, 2023

prepared by:

Gavin Koma
Email: gavintkoma@temple.edu

A. TASK 1

The goal of this assignment is split into two tasks. The first task is to utilize k-Nearest Neighbors to assess datasets 8, 9, and 10. K-Nearest Neighbors (kNN) is an incredibly important and fundaments algorithm that is utilized in machine learning; the algorithm functions by a data point based on how its neighbor is classified. It is important to note that we will vary our values of k (which represents how many neighbors we will assess) to classify new data points.

Our homework assignment is to perform kNN as well as k-Means Clustering (kMN) on the provided datasets. The only difference for this assignment and homework 4 is that the table in homework 4 requests to perform the analyses after we pool /train and /dev together. Homework 8, states to only optimize performance using /dev and to use this to classify /train, /dev, and /eval.

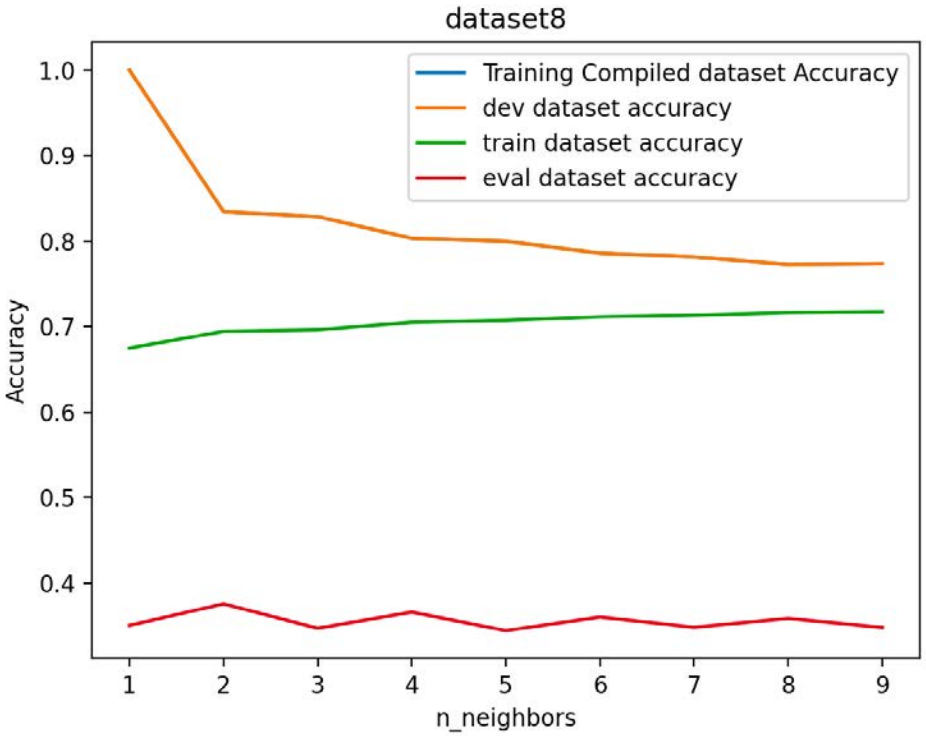
To begin, the data was first imported, and datasets were labeled accordingly.

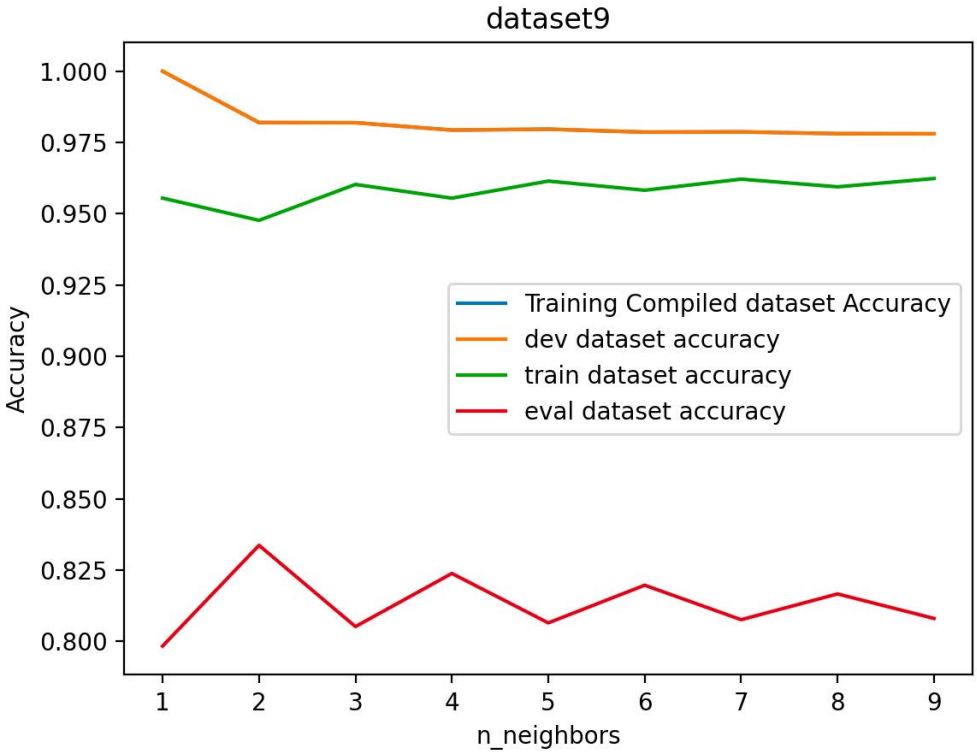
```
#%%hw pt1
#we need to test on /dev, /train, and /eval
labels_dev = dev.iloc[:,0] #label column
labels_train = train.iloc[:,0] #label column
labels_eval = evalu.iloc[:,0] #label column

xvalues_dev = pd.DataFrame([dev.iloc[:,1],dev.iloc[:,2]]).T
xvalues_train = pd.DataFrame([train.iloc[:,1],train.iloc[:,2]]).T
xvalues_eval = pd.DataFrame([evalu.iloc[:,1],evalu.iloc[:,2]]).T

training_data = xvalues_dev
label_data = labels_dev
```

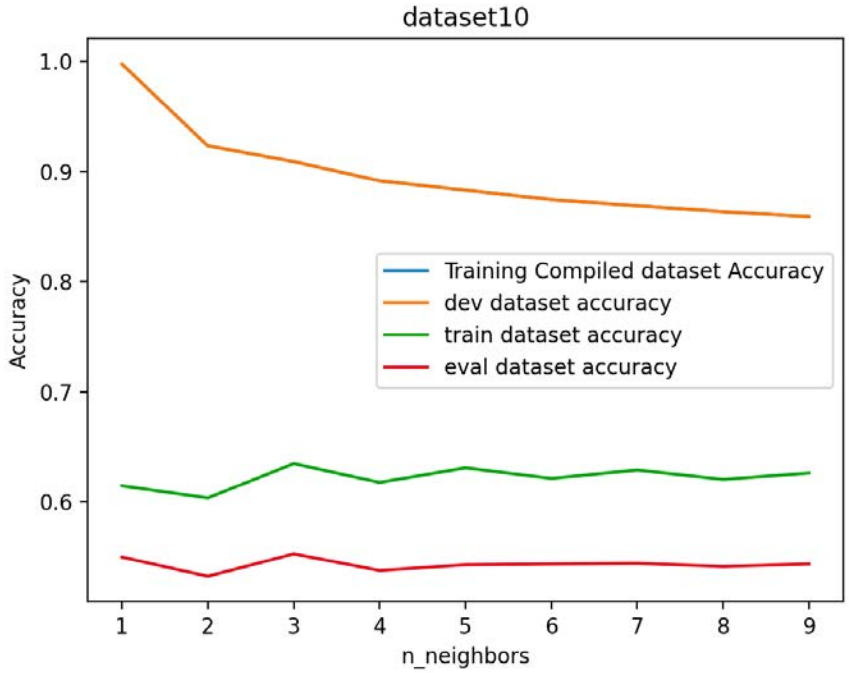
I then created a list of numbers from 1 to 10 and initiated empty vectors to store accuracy values for plots later. I then iterated through all 10 values and supplied them through a loop that performs a kNN algorithm for each varying number of neighbors.

```
for i, k in enumerate(neighbors_val):
    knn = KNeighborsClassifier(n_neighbors=k)
    knn.fit(training_data,label_data)

    #compute data using /dev + /train for /train /dev /eval
    train_accuracy[i] = knn.score(training_data,label_data)

    dev_test_accuracy[i] = knn.score(xvalues_dev, labels_dev)
    train_test_accuracy[i] = knn.score(xvalues_train, labels_train)
    eval_test_accuracy[i] = knn.score(xvalues_eval, labels_eval)
```

We are able to obtain the following graph utilizing the above for-loop:


dataset8
Training Compiled dataset Accuracy
dev dataset accuracy
train dataset accuracy
eval dataset accuracy
Accuracy
n_neighbors
1.0
0.9
0.8
0.7
0.6
0.5
0.4
1
2
3
4
5
6
7
8
9

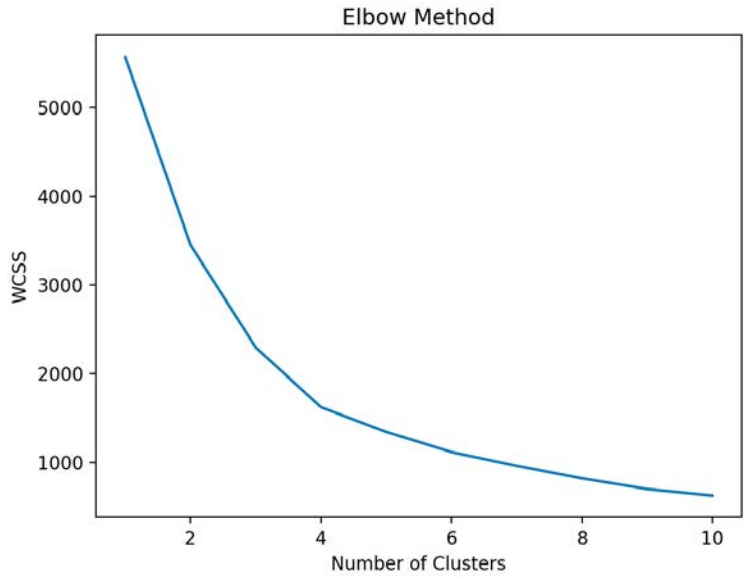

dataset9
Training Compiled dataset Accuracy
dev dataset accuracy
train dataset accuracy
eval dataset accuracy
Accuracy
n_neighbors
1.000
0.975
0.950
0.925
0.900
0.875
0.850
0.825
0.800
1
2
3
4
5
6
7
8
9



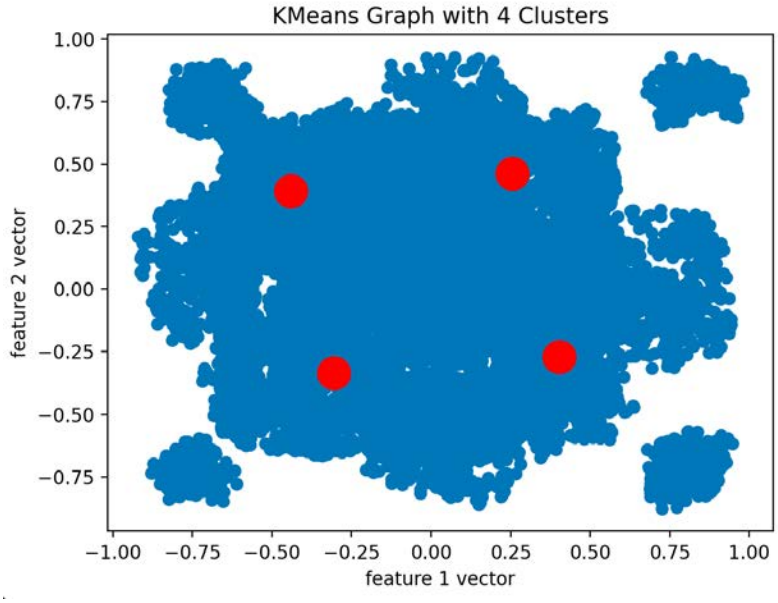
To save rewriting the same table of accuracy scores multiple times, I have elected to include the final table with accuracy scores for the summary portion of this assignment.

B. TASK 2

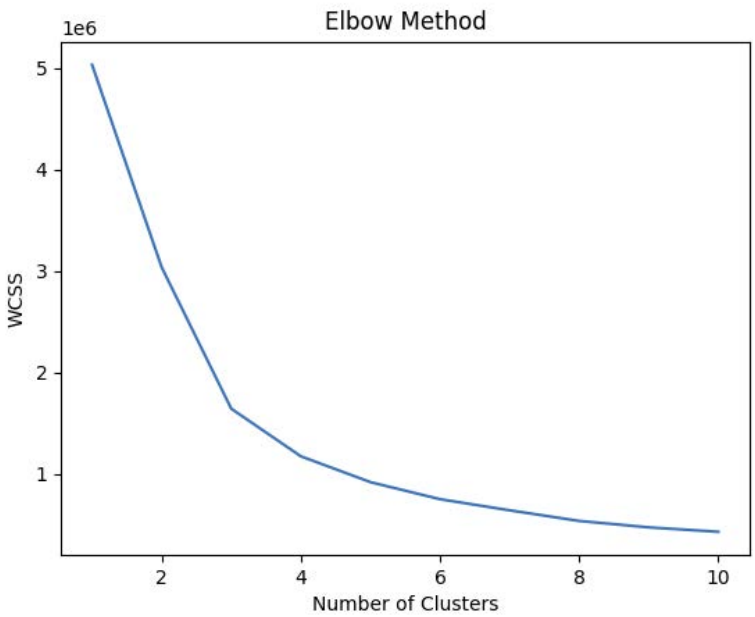
For part two of this assignment, we needed to implement k-Means Clustering. To do so, I first imported KMeans from the sklearn library. I then needed a method to assess what value of clusters to use for kMN; I utilized the 'Elbow Method' for this. The Elbow Method allows us to choose an ideal value of k based on the distance between a data point and their assigned clusters. We choose our k-value at the point of inflection on this graph as there is an acceptable trade off in accuracy/clusters. Here is one example of the outputted Elbow Graph:

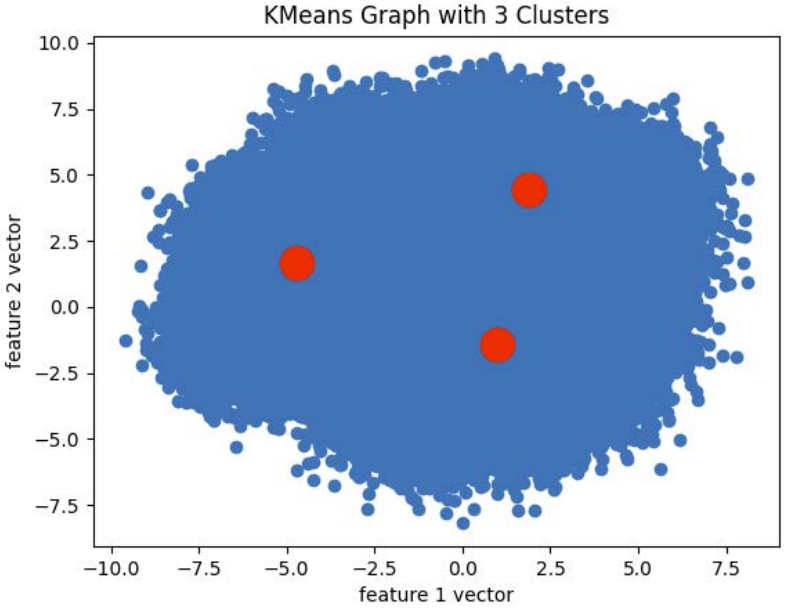


Based on the above graph, for dataset8, I will use 4 clusters. We essentially want to choose the point on the graph where we see the largest degree of inflection. There are ways to automate this process, but it seems easier to just pick. The same method was implemented for dataset 9 and dataset 10. The dataset, when graphed with four clusters, looks as following. The red dots represent the general location of where the clusters are located.

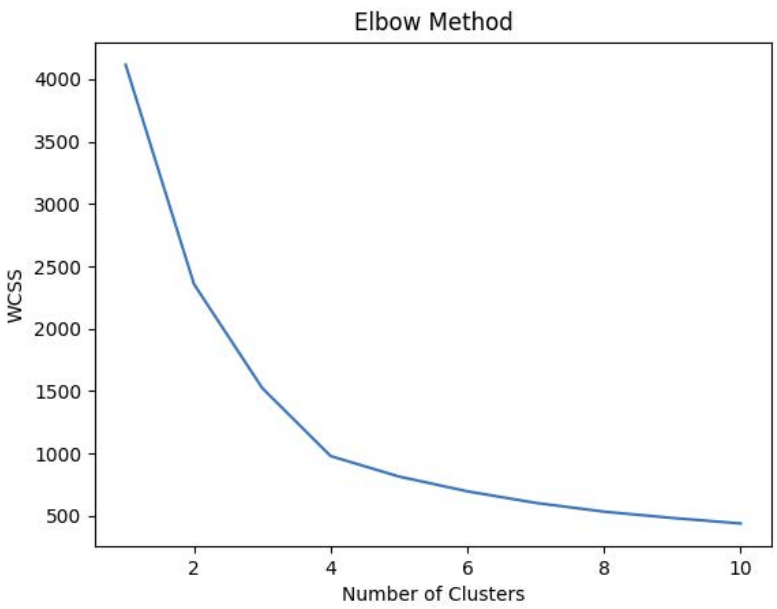


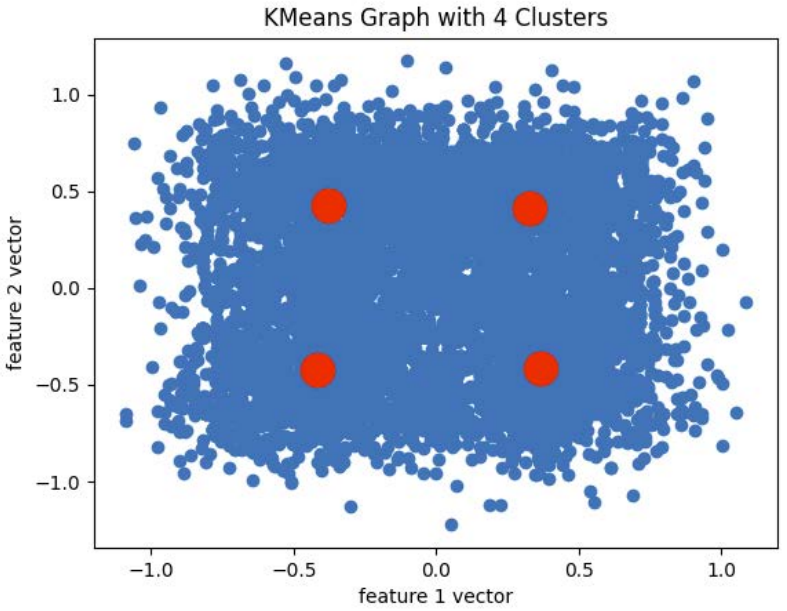
The following is the corresponding Elbow Graph and Cluster Graph for dataset 9:





The following is the corresponding Elbow Graph and Cluster Graph for dataset 10:





In order to assess the accuracy of our clusters, the following line of code was implemented to assess /dev, /train, and /eval:

```
scoredev = metrics.accuracy_score(labels_dev,kmeans.predict(xvalues_dev))
scoretrain = metrics.accuracy_score(labels_train,kmeans.predict(xvalues_train))
scoreeval = metrics.accuracy_score(labels_eval,kmeans.predict(xvalues_eval))

print("dev accuracy score: " + str(scoredev) +'\n\n')
print("train accuracy score: " + str(scoretrain) +'\n\n')
print("eval accuracy score: " + str(scoreeval) +'\n\n')
```

All values were appended to the table found in summary.

C. SUMMARY

Compiled error rates in table:

DS	System	Training Data	Train	Dev	Eval
#08	KNN	/dev	0.3210	0.1848	0.6090
	KMN	/dev	0.2772	0.2775	0.2261
#09	KNN	/dev	0.0460	0.0202	0.1774
	KMN	/dev	0.2926	0.3047	0.2369
#10	KNN	/dev	0.3727	0.1113	0.4640
	KMN	/dev	0.2204	0.2532	0.2406

This assignment was quite straightforward. There was a surplus of information online to read about the algorithms used. I think this was a great way to introduce the concepts of KNN and KMN and offered a simple way to practice with them. I had a vague understanding of KNN prior to this assignment but actually implementing it offered a better insight to how it works and how varying the neighbors affects the accuracy.

KMN was an algorithm that I had never previously worked with or heard of prior to lecture. The clustering methods was very interesting to read about but it wasn't until I had actually implemented it in code that I garnered some understanding of how it actually functions.