

Thomas Munoz Vasquez

Big Data Programming

Due April 03, 2024

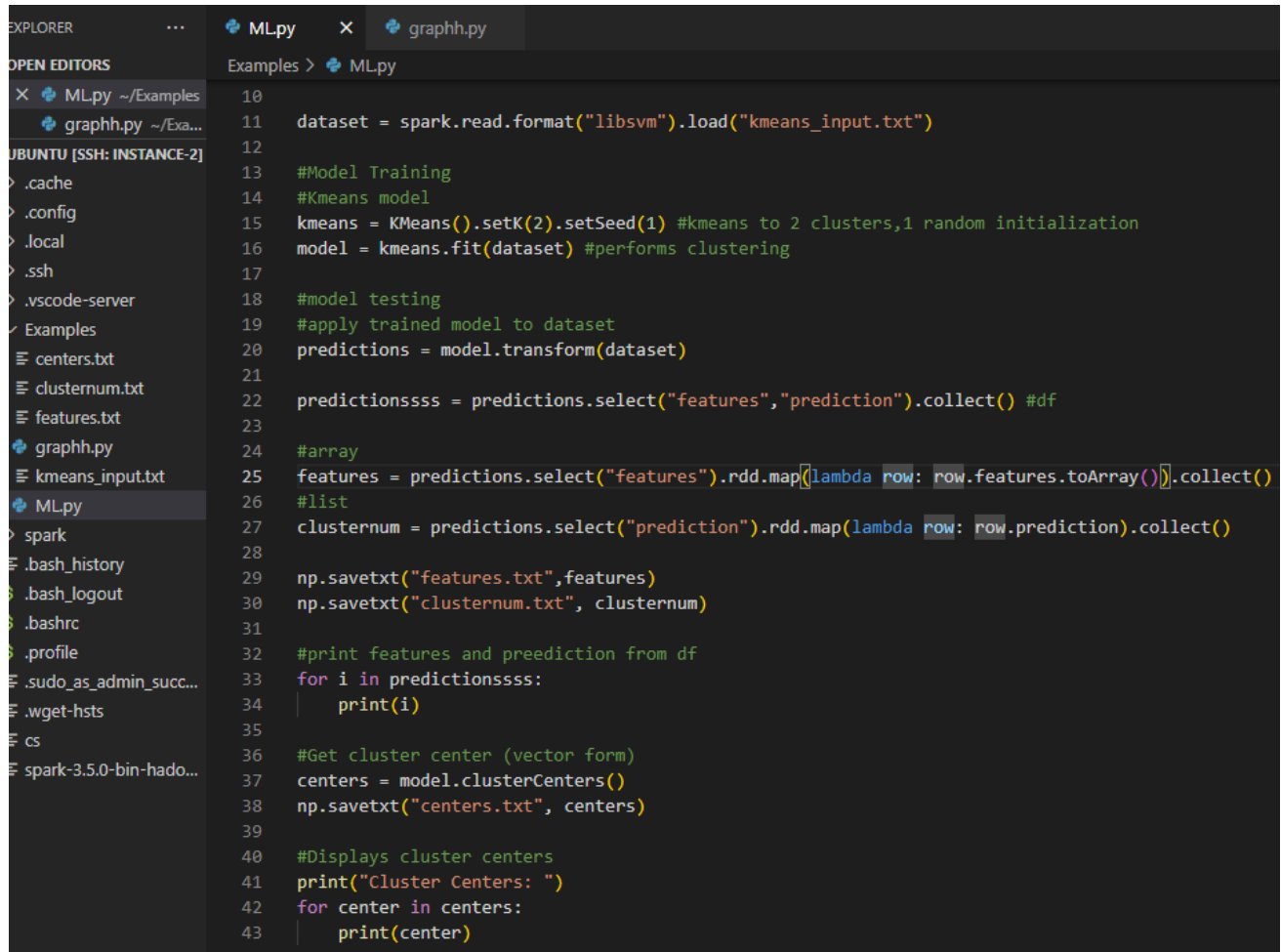
Assignment 4

In our analysis, we employed the k-algorithm to partition raw data into two different clusters. After processing the data, we obtained the cluster labels for each data point, indicating which group it belongs to, along with the coordinates of the centroids. The data provided was two dimensional, making it suitable for visualization in a scatter graph.

The k-means algorithm is used to cluster data points into distinct groups. In this scenario, we utilize k-means functions to perform data clustering. The functions trained the k-means model with $k = 2$, we obtain the cluster labels, data points and centroids for two clusters. The setseed is a random initialization for the centroids, it iterates every time until the centroids no longer change significantly. Lastly, fit performs clustering in the sample data with the specifications given for the model. The sample data serves as a starting point to train the model. It can perform clustering with any new data we provide, but in this case, we are using the same dataset.

Transform creates a new data frame with cluster labels and data points. They are categorized as features and predictions; we collect the data in the data frame to display the data points and cluster labels. To create visualizations, matplotlib provides the necessary APIs to perform plotting operations, but it may not work well in a remote environment because it typically opens a window to display the graph, which can be problematic for remote access. To resolve the issue, the program converts the features and predictions

columns into an array/list and saves it in a text file. This process ensures we can plot the features and predictions on the graph, we repeat this process to obtain the centroids as well.



```
10
11 dataset = spark.read.format("libsvm").load("kmeans_input.txt")
12
13 #Model Training
14 #Kmeans model
15 kmeans = KMeans().setK(2).setSeed(1) #kmeans to 2 clusters,1 random initialization
16 model = kmeans.fit(dataset) #performs clustering
17
18 #model testing
19 #apply trained model to dataset
20 predictions = model.transform(dataset)
21
22 predictionssss = predictions.select("features","prediction").collect() #df
23
24 #array
25 features = predictions.select("features").rdd.map(lambda row: row.features.toArray()).collect()
26 #list
27 clusternum = predictions.select("prediction").rdd.map(lambda row: row.prediction).collect()
28
29 np.savetxt("features.txt",features)
30 np.savetxt("clusternum.txt", clusternum)
31
32 #print features and prediction from df
33 for i in predictionssss:
34     print(i)
35
36 #Get cluster center (vector form)
37 centers = model.clusterCenters()
38 np.savetxt("centers.txt", centers)
39
40 #Displays cluster centers
41 print("Cluster Centers: ")
42 for center in centers:
43     print(center)
```

Figure 1. Main program code.

```

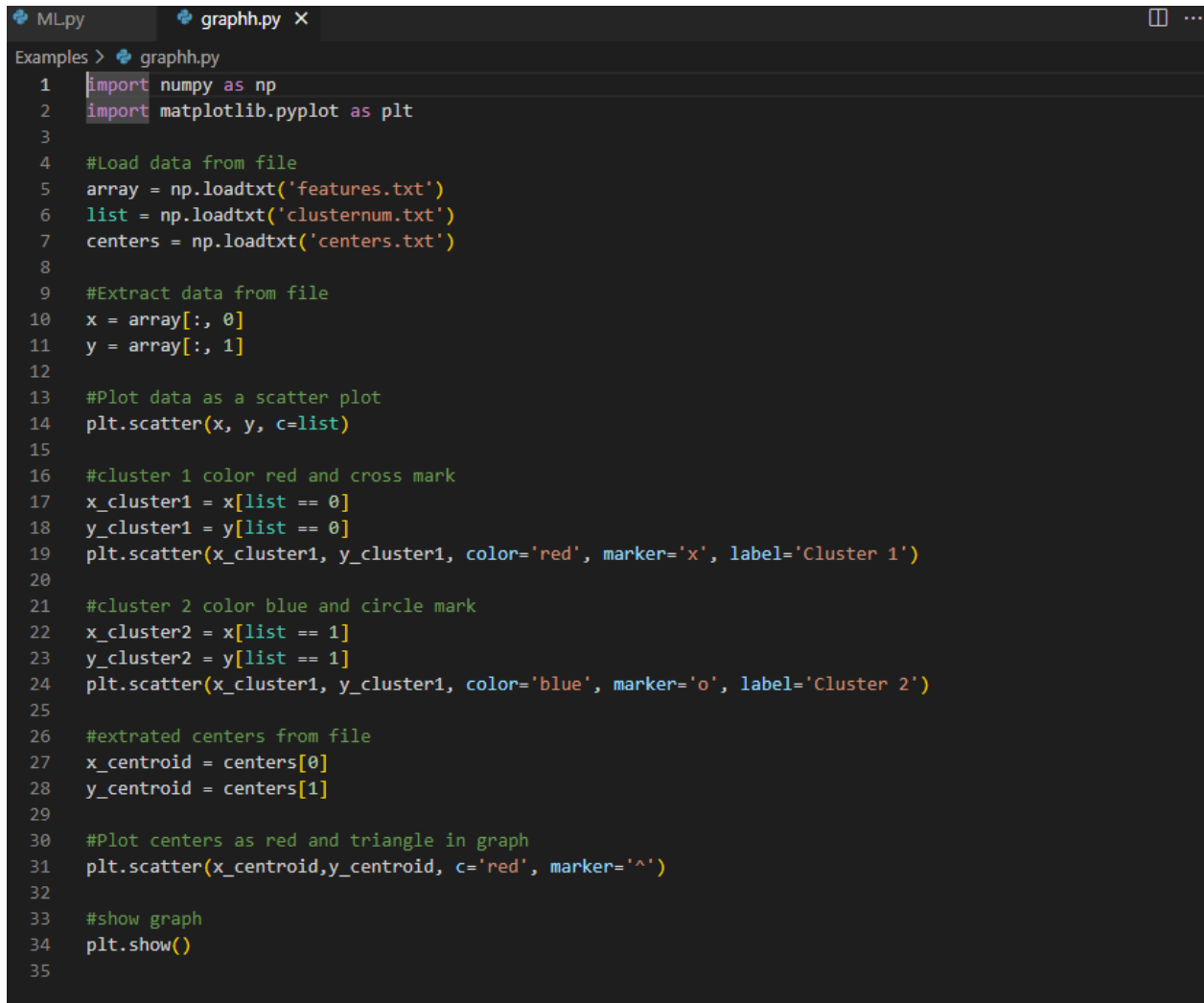
Row(features=SparseVector(2, {0: 1.9, 1: 1.22}), prediction=1)
Row(features=SparseVector(2, {0: 1.81, 1: 0.81}), prediction=1)
Row(features=SparseVector(2, {0: 1.97, 1: 1.05}), prediction=1)
Row(features=SparseVector(2, {0: 1.94, 1: 0.92}), prediction=1)
Row(features=SparseVector(2, {0: 2.34, 1: 0.85}), prediction=1)
Row(features=SparseVector(2, {0: 1.81, 1: 1.09}), prediction=1)
Row(features=SparseVector(2, {0: 2.16, 1: 0.7}), prediction=1)
Row(features=SparseVector(2, {0: 1.92, 1: 0.89}), prediction=1)
Row(features=SparseVector(2, {0: 2.32, 1: 1.2}), prediction=1)
Row(features=SparseVector(2, {0: 1.73, 1: 1.38}), prediction=1)
Row(features=SparseVector(2, {0: 2.13, 1: 1.06}), prediction=1)
Row(features=SparseVector(2, {0: 2.28, 1: 1.25}), prediction=1)
Row(features=SparseVector(2, {0: 2.05, 1: 1.07}), prediction=1)
Row(features=SparseVector(2, {0: 2.47, 1: 1.03}), prediction=1)
Row(features=SparseVector(2, {0: 1.77, 1: 0.91}), prediction=1)
Row(features=SparseVector(2, {0: 2.06, 1: 0.68}), prediction=1)
Row(features=SparseVector(2, {0: 1.43, 1: 1.21}), prediction=1)
Row(features=SparseVector(2, {0: 2.26, 1: 0.67}), prediction=1)
Row(features=SparseVector(2, {0: 1.82, 1: 0.63}), prediction=1)
Row(features=SparseVector(2, {0: 1.87, 1: 0.77}), prediction=1)
Row(features=SparseVector(2, {0: 1.91, 1: 1.19}), prediction=1)
Row(features=SparseVector(2, {0: 1.85, 1: 0.78}), prediction=1)
Row(features=SparseVector(2, {0: 1.88, 1: 1.3}), prediction=1)
Row(features=SparseVector(2, {0: 2.02, 1: 1.03}), prediction=1)
Row(features=SparseVector(2, {0: 1.94, 1: 0.75}), prediction=1)
Row(features=SparseVector(2, {0: 2.03, 1: 0.68}), prediction=1)
Row(features=SparseVector(2, {0: 2.24, 1: 1.09}), prediction=1)
Row(features=SparseVector(2, {0: 2.27, 1: 1.32}), prediction=1)
Row(features=SparseVector(2, {0: 2.03, 1: 0.97}), prediction=1)
Row(features=SparseVector(2, {0: 1.84, 1: 0.98}), prediction=1)
Row(features=SparseVector(2, {0: 1.87, 1: 1.08}), prediction=1)
Row(features=SparseVector(2, {0: 1.82, 1: 1.04}), prediction=1)
Row(features=SparseVector(2, {0: 1.99, 1: 1.0}), prediction=1)
Row(features=SparseVector(2, {0: 2.37, 1: 1.05}), prediction=1)
Row(features=SparseVector(2, {0: 2.36, 1: 1.31}), prediction=1)
Row(features=SparseVector(2, {0: 1.99, 1: 1.15}), prediction=1)
Row(features=SparseVector(2, {0: 2.2, 1: 1.33}), prediction=1)
Row(features=SparseVector(2, {0: 2.2, 1: 1.23}), prediction=1)
Row(features=SparseVector(2, {0: 2.09, 1: 1.19}), prediction=1)
Row(features=SparseVector(2, {0: 2.08, 1: 1.27}), prediction=1)
Row(features=SparseVector(2, {0: 1.78, 1: 1.1}), prediction=1)
Cluster Centers:
[1.0437 2.0085]
[1.9901 1.0093]

```

Figure 2. Main program output.

The program graph.py plots the data points, cluster labels, and centroids utilizing the files created in the main program. First, it retrieves the data points and stores them in x and y since they are two dimensional objects. We plot them in a scatter graph, it then retrieves the cluster labels from the file and matches it with each cluster, 0 for the first cluster and 1

for the second cluster. The color and mark is set for each cluster when we plot them. Lastly, we repeat the process for the centroids.



```
1 import numpy as np
2 import matplotlib.pyplot as plt
3
4 #Load data from file
5 array = np.loadtxt('features.txt')
6 list = np.loadtxt('clusternum.txt')
7 centers = np.loadtxt('centers.txt')
8
9 #Extract data from file
10 x = array[:, 0]
11 y = array[:, 1]
12
13 #Plot data as a scatter plot
14 plt.scatter(x, y, c=list)
15
16 #cluster 1 color red and cross mark
17 x_cluster1 = x[list == 0]
18 y_cluster1 = y[list == 0]
19 plt.scatter(x_cluster1, y_cluster1, color='red', marker='x', label='Cluster 1')
20
21 #cluster 2 color blue and circle mark
22 x_cluster2 = x[list == 1]
23 y_cluster2 = y[list == 1]
24 plt.scatter(x_cluster1, y_cluster1, color='blue', marker='o', label='Cluster 2')
25
26 #extrated centers from file
27 x_centroid = centers[0]
28 y_centroid = centers[1]
29
30 #Plot centers as red and triangle in graph
31 plt.scatter(x_centroid, y_centroid, c='red', marker='^')
32
33 #show graph
34 plt.show()
35
```

Figure 3. Graph program.

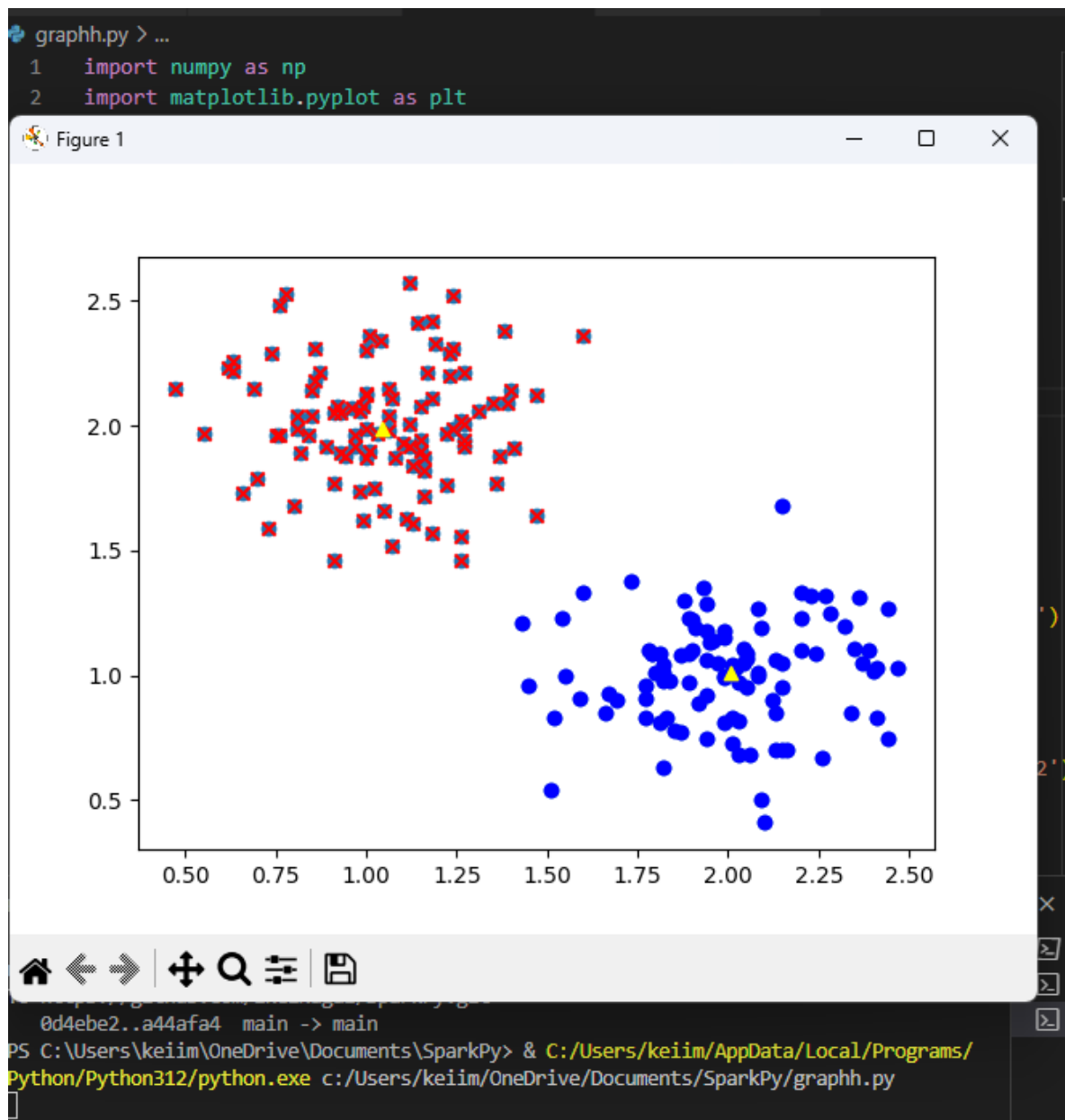


Figure 4. Clusters Visualizations