**Assignment 5 (due by midnight MST the day prior to Live Session 10)**

Purpose: In this homework, you will perform appropriate clustering operations on several different datasets. And to provide a rationale for your model selection and evaluation decisions. Additional questions are included to evaluate your understanding of selected clustering concepts/methods.

The homework prompts require you to exercise ‘best professional judgement’ in relation to your decision-making and modeling process. And to share a rationale for your approach/interpretation in clear & concise terms.

Please note that there are a wide variety of clustering algorithms available, each with various strengths and weaknesses. This homework focuses on a limited subset of common clustering algorithms: Kmeans, Gaussian Mixture Model, DBSCAN, and Agglomerative Hierarchical.

Please show code and calculations.

**Learning Objectives**

You should be able to:

* Apply appropriate steps to clean and/or prepare data for cluster analyses.
* Undertake exploratory data analysis to support model selection.
* Select an appropriate clustering & evaluation method(s) for a given dataset.
* Validate model results following clustering when labelled data is available.
* Discuss the strengths/weaknesses of various clustering methods.
* Leverage clustering methods for outlier detection

**Problem 1a. Kmeans**

Please load the following dataset: ‘x1\_vals.npy’

Conduct an initial exploratory data analysis (EDA) to evaluate the data. Create at least two different types of visualizations to help you evaluate possible values for K (the number of clusters). Implement two different analytical methods to narrow your choice of K prior to modeling.

Use scikit-learn to fit a basic kmeans clustering model with random initialization and reproducible results. Then create a plot of your results that distinguishes each of the clusters by color. Extract values for your cluster centroids, the number of iterations to convergence, as well as a value that serves as a measure of cluster ‘coherence’.

Questions:

1. What method(s) did you use to identify an appropriate value for K? Why did you select this method? (5 pts)

I chose the elbow and the Silhouette methods. The Elbow method allows us to figure out the point in the data where a decrease in distortion happens, by showing a significant drop in the curve. The silhouette measures how close an object is to its own cluster compared to other clusters. The value ranges from -1 to 1, where a high value indicates that the object is well matched to its own cluster and poorly matched to its neighbored clusters, which is very useful in this problem!

1. What value did you select for K? Does your EDA support this choice? (2 pts)

I selected a k value of 4. This value seemed to balance the number of clusters, without overcomplicating the data. My EDA supported this choice, which is shown in the figures.

1. How many iterations were required before your model converged? (2 pts)

The model required 5 iterations to converge.

1. What were the values for each of your cluster centroids? (2 pts)

Cluster centroids: [[ 0.33713347 0.30818338]

[ 7.89148283 5.09540347]

[ 3.15664275 7.11149514]

[-0.54729704 1.80729252]].

1. What kmeans measure serves as a proxy for cluster coherence? What value did your model return? Discuss your interpretation of this value. (5 pts)

Inertia, or within-cluster sum of squares. Value: 1276.7913476012773. This value represents the total variance within the clusters; lower values indicate tighter, more coherent clusters, suggesting a good clustering fit to the data.

**Problem 1b. Silhouette Plot**

Using the data and Kmeans model from Problem 1a, create a set of silhouette plots (i.e., an appropriate range of k values) by adapting the following code for your use:

from sklearn.metrics import silhouette\_samples, silhouette\_score

# Range of values for k

range\_n\_clusters = [2, 3, 4, 5, 6]

plt.figure(figsize=(len(range\_n\_clusters) \* 6, 8))

for i, n\_clusters in enumerate(range\_n\_clusters):

# Initialize the clusterer

clusterer = KMeans(n\_clusters=n\_clusters, random\_state=42)

cluster\_labels = clusterer.fit\_predict(X)

# Compute the silhouette scores

silhouette\_avg = silhouette\_score(X, cluster\_labels)

sample\_silhouette\_values = silhouette\_samples(X, cluster\_labels)

plt.subplot(1, len(range\_n\_clusters), i + 1)

y\_lower = 10

for j in range(n\_clusters):

ith\_cluster\_silhouette\_values = sample\_silhouette\_values[cluster\_labels == j]

ith\_cluster\_silhouette\_values.sort()

size\_cluster\_j = ith\_cluster\_silhouette\_values.shape[0]

y\_upper = y\_lower + size\_cluster\_j

color = plt.cm.get\_cmap("Spectral")(float(j) / n\_clusters)

plt.fill\_betweenx(np.arange(y\_lower, y\_upper), 0, ith\_cluster\_silhouette\_values, facecolor=color, edgecolor=color, alpha=0.7)

plt.text(-0.05, y\_lower + 0.5 \* size\_cluster\_j, str(j))

y\_lower = y\_upper + 10

plt.title("Silhouette plot for {} clusters".format(n\_clusters))

plt.xlabel("Silhouette coefficient values")

plt.ylabel("Cluster label")

plt.axvline(x=silhouette\_avg, color="red", linestyle="--")

plt.yticks([])

plt.xticks(np.arange(-0.1, 1.1, 0.2))

plt.xlim([-0.1, 1.0])

plt.show()

Note that X refers to your dataset.

Questions:

1. What information does a silhouette analysis provide? (5 pts)

Silhouette analysis gives you a way to measure how good your clustering is. It calculates a score for each data point that reflects how well it fits into its assigned cluster compared to other clusters. This score ranges from -1 to 1:

A score close to +1 means the data point is well inside its own cluster and far from other clusters. A score around 0 means the data point is very close to a cluster boundary. A score close to -1 means the data point might be in the wrong cluster.

1. Describe how to interpret the individual silhouette plots in terms of cluster cohesion and separation. What are the key features of the graphs that aid your evaluation? (5 pts)

**Length of the Lines**: Longer lines mean higher silhouette scores

**Consistency in Lengths**: If most lines for a cluster are long and about the same length, it means the points are fitting well in that cluster.

**Width of the Clusters**: The width of the filled area represents the number of points in the cluster

**Above the Average Line**: Points above the dashed line are considered well clustered.

1. From the silhouette plots you created, what value of K affords the best cluster assignments? (3 pts)

A k value of 3 showed the best cluster assignments.

**Problem 2. External Validation**

When labels are available to distinguish groupings within a dataset, external cluster validation can be used to evaluate how well clustering results match the external criteria. In this problem, you will create a synthetic dataset with known cluster labels, implement a Kmeans model using the data, and then apply an appropriate technique for external validation.

Use sklearn’s ‘make\_blobs’ to create a synthetic dataset for clustering. Your data set should include 5 clusters with similar variance and number of observations, and 2 features. Ensure that your result is reproducible.

Use Kmeans to fit your model. And then save out predicted cluster assignments for each of your observations.

Select and employ an appropriate analytical method to assess the degree of agreement between your predicted vs. actual cluster assignments – i.e., external validation. You may need to do some research to identify the former. Create scatterplots for both predicted and actual cluster assignments using matplotlib’s subplot() method.

Questions:

1. what method did you select to assess cluster agreement and why? (5 pts)

The method I used is the ARI (Adjusted Rand Index). Because it measures the similarity between two clusterings. It could be used to directly compare the clustering result to the ground truth, which makes it useful to evaluate the performance of a clustering algorithm.

1. What do your results of this assessment suggest? (5 pts)

The results suggest a high degree of effectiveness in uncovering the underlying structure of the data compared to the ground truth.

**Problem 3a – Concepts: Interpreting SSE**

Total SSE is the sum of the SSE for each separate attribute in the Kmeans algorithm. What does it mean if the SSE for one variable is low for all clusters? Low for just one cluster? High for all clusters? High for just one cluster? How could you use the per variable SSE information to improve your clustering? (5 pts)

If the SSE for a particular variable is low across clusters, it means that the data points within each cluster are closely packed around the average for that variable.

If the SSE for a particular variable is low in just one cluster, it indicates that the data points in that cluster are very close to each other for that variable.

A high SSE for a variable across all clusters suggests that the data points are spread out over a wide range of values for that variable in each cluster. Which means that the variable is not doing a god job of separation between clusters.

If the SSE for a variable is high in just one cluster, it might indicate an outlier within that cluster, which could possibly mean that we could add an additional cluster, or not, depending on the number of outliers.

If a variable has a consistently high SSE across clusters, it might not be a good option for clustering. Combining variables or creating new ones could help. Variables with a larger range of values can dominate the SSE. Standardizing the data so that each variable contributes equally can help of the SSE is high across all clusters. Finally, analyzing one variable within a cluster might allow us to discover sub-groups or additional clusters formed from that one cluster.

**Problem 3b. Local and Global Objective Functions**

K-means. For the following sets of two-dimensional points,

(1) provide a sketch of how they would be split into clusters by K-means for the given number of clusters and

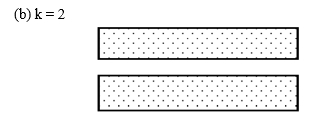
(2) indicate approximately where the resulting centroids would be. Assume that we are using the squared error objective function.

If you believe that there is more than one possible solution, then please indicate whether each solution is a global or local minimum (draw pictures to represent your responses). Darker areas indicate higher density. Assume a uniform density within each shaded area. (6 pts)

A black and white circle with dots

Description automatically generated

For the above figure, k-means will place one centroied within each of the three circles because the points in one circle are closer to each other than to points in other circles. The centroid will be placed where the red circle is. This solution would be a global minimum as the distribution is uniform and there is clear separation between the concentric clusters.



For the above figure for k=2, each rectangle will have its own cluster and their centroids will be where the red circles are. This will also be a global minimum as the two groups are clearly separated based on the density of the points within each rectangle.

A black and white drawing of a circle

Description automatically generated

For the above figure for k=2, k-means will split the circle into two clusters, as shown by the red line. Any line that passes through the center of the circle will work! The centroids are represented also by the red circles shown in the figure, equidistant from the circle center. This would be a local minimum as there isn’t a single clear cut (as the line could really be drawn in any direction, as long as it is passing through the center of the circle.)

**Problem 3c. Density clustering**

Suppose we apply DBSCAN to cluster the following dataset using Euclidean distance.

A graph with red dots and numbers

Description automatically generated

A point is a core point if its density (num point within EPS) is ≥ MinPts. Given that MinPts =3 and EPS = , answer the following questions.

1. Label all point as ‘core points’, ‘boundary points’, and ‘noise’. (5 pts)

**Core Points** (≥ 2 other points within **EPS** distance):

* 1. C (has B, D, E within **EPS** = 1)
  2. E (has C, B, F within **EPS** = 1)
  3. J (has I, L within **EPS** = 1)
  4. L (has J, M within **EPS** = 1)

**Boundary Points** (< 2 other points within **EPS** distance but near a core point):

* 1. B (close to C, E)
  2. D (close to C)
  3. F (close to E)
  4. I (close to J)
  5. M (close to L)

**Noise Points** (isolated from core points):

* 1. A (too far from other points with **EPS** = 1)
  2. G (too far from other points with **EPS** = 1)
  3. H (too far from other points with **EPS** = 1)

1. What is the clustering result (i.e., how will the data cluster)? (5 pts)

One cluster consists of points B, C, D, E, F (since they are all connected to each other through core points). Another cluster consisting of points I, J, L, M (connected through core points).

**Problem 3d. Entropy vs. SSE**

Assume you are given a data set of objects, each of which is assigned to one of two classes, and suppose that C1 and C2 are two clusterings produced from this data set. If entropy judges C1 to be a more accurate clustering than C2, is it necessary that SSE will also judge C1 to be a more accurate clustering than C2? (5 pts)

It is not necessary that if one clustering has a lower entropy, it will also have a lower SSR. Each metric could lead to different interpretations of what constitutes the best clustering depending on whether the focus is on class homogeneity or cluster tightness.

A scenario could occur where C1 has a lower entropy (more homogeneous in terms of class labels) but a higher SSE because the clusters are more spread out. Conversely, C2 could have a higher entropy (less homogeneity in class labels) but a lower SSE because the clusters are tighter.

**Problem 4. Selecting an Appropriate Clustering Algorithm**

Please import the following dataset – x4\_vals.npy

Conduct an initial EDA to evaluate the data. Create at least two different types of visualizations to help you evaluate possible values for K (the number of clusters).

Parameterize and implement the following clustering algorithms using this dataset: Kmeans, Gaussian Mixture Model, DBSCAN, Agglomerative Hierarchical.

Create a plot of your results for each algorithm that distinguishes clusters by color.

Select the algorithm/parameters that you believe provides the ‘best’ clustering results. Note: you may find it helpful to iterate between clustering algorithms/parameterizations and related graphs to make this selection.

Questions:

1. What method(s) did you use to identify an appropriate value for K? Why did you select this method(s)? (5 pts)

I used the elbow method. This method is the most commonly use for determine the number of clusters in a dataset, providing a visual cue for the point beyond which adding more clusters does not significantly improve the fit of the model.

1. What value did you select for K? Does your EDA support this choice? (5 pts)

The elbow method suggests a k value of 4. The EDA supports this choice as there are 4 distinct groupings of data points.

1. Why is the algorithm that you selected for your final clustering model the best choice among those you evaluated? (5 pts)

KMeans, as it shows clearly the separated clusters which align with the circular distribution of data points in the scatter plot.

**Problem 5. Comparing Algorithms**

Compare the relative strengths and weaknesses of the clustering methods listed in Problem 3. Include your observations in a table form. (10 pts)

| **Clustering Method** | **Strengths** | **Weaknesses** |
| --- | --- | --- |
| KMeans | -Clear and distinct clusters. | - Assumes spherical clusters and doesn't work well with complex geometries. |
| Gaussian Mixture Model (GMM) | - Flexible in terms of cluster covariance and can accommodate elongated clusters | - More computationally intensive than KMeans |
| DBSCAN | - Does not require the number of clusters to be predefined. | - Difficult to choose appropriate parameters. |
| Agglomerative Hierarchical | - Does not assume any particular number of clusters. | - Computationally expensive for large datasets. Fixed once created, no reassignment of points like in KMeans. |

**Problem 6. Dimensionality Reduction (10 pts)**

Please import the following dataset – network.csv

Dimensional reduction is a standard preprocessing step before clustering for high dimensional datasets. This owes to the fact that distance measures do not work well in high dimensional spaces. It’s worth noting, however, that the ability to cluster data owes more to the feature signal-to-noise ratio than to the number of dimensions.

For this problem we are going to read a dataset that contains performance information from LTE cell sites operated by a rural provider. The dataset is comma delimited and contains 33 columns. The ‘day’ column is a string in the 2020-12-16 format (year-month-day). You will need to convert this to pandas datetime. The BTS\_ID column is a unique identifier for the cell site and radio. The first number is the cell site and the number after the dash indicates the radio. Do not open and save the dataset in excel. Excel will convert all the BTS\_ID’s that is thinks look vaguely like a date into a date (10-1 will be converted to October 1st). The remaining columns are different performance measurements for how well the mobile phones are doing while on that site. Some measures are best when they are very small values and some are best when they are large. It would require a deep dive into the LTE cell phone standard to explain all the measurements. Some of the key values are FDD\_Cell\_DL\_MAC\_Data\_Volume\_MB which shows how much data was downloaded in MegaBytes. RRC\_Drop\_Pct is the percentage of data sessions that were dropped due to bad coverage.

It is very time consuming for the engineers in charge of this network to look at all 31 measurements for each cell site to see if there are any problems. We need to simplify the problem by doing dimensionality reduction and finding sites that are not “normal”. The goal of this analysis is to find cell sites that are performing differently than the normal sites by seeing how close their performance is to all the other sites. We are going to reduce the 31 dimensions down to two dimensions and then find outliers from the main cluster.

Part 1: filter the dataframe down to just the last day, 2021-06-13. Drop the day column and save the BTS\_ID column to another variable. Standard scale the data and run PCA with two principal components. Join the BTS\_ID data back into the PCA dataframe so it can be used with the plot. Plot the data using plotly scatter plot with the hover data showing the BTS\_ID. Labeling all points in scatter plot is going to be impossible to read, so hover data allows you to read the points you want and not overlap with the others.

The plot should look like this when you hover over BTS\_ID 143-3:A graph showing a number of data

Description automatically generated

Part 2: Run DBSCAN on the PCA data from above to find outliers, list BTS\_ID's that are outliers (the points DBSCAN calls noise). Use eps=1 and min\_samples=2 to get a reasonable number of outliers. Make the labels into a column in the PCA dataframe. Next, change the labels in the dataframe to be -1 if it is an outlier and 0 if it is not. Now do a plotly plot with the color being the labels column of the dataframe so we can clearly see the outliers.

The plot should look like this when you hover over BTS\_ID 143-3: A screen shot of a graph

Description automatically generated

Part 3: Perform PCA on all the dates in the dataset except "2021-06-13" just like you did above with a single date. Compute the mean Euclidian distance from the center of the plot (coordinate 0,0) for each BTS\_ID. Remember each BTS\_ID has measurements for each day so get the mean distance across all the days. Compute the distance from the center of the plot for each BTS\_ID using the "2021-06-13" data.

Print a list of all the outlier sites are farther from the "normal", meaning center of the plot, than their mean over the last 6 months. Print a list of all the sites that are getting better (meaning heading toward the other sites performance in the center of the cluster).

The sites that are outliers and getting worse should be reported to the engineers in charge of system performance. The outliers that are getting better show the engineers are making some progress.

Are the engineers making progress with the system performance?

Yes. Please see the code for more details!