Text Mining Mini-Project - Part 2: Clustering

Andrew Branum, Nathaniel Cecil

INPUTS

For the Text Mining Mini-Project, we used the 'Twenty Newsgroups' dataset from the UC Irvine Machine Learning Repository. Within the dataset, there are 18,846 separate documents across twenty different newsgroup categories. These categories include broad topics like technology, sports, science, religion, and more. Each document is assigned a category label for a specific newsgroup category, making it suitable for supervised machine learning tasks. These labels however, will not be used for clustering other than for validation.

OBJECTIVES

1) Explore the Implementation of Clustering Techniques
    a) We used OPTION 1: Python and building on the notebook provided
    b) Two options for feature engineering, similarity, clustering, validation, etc.
2) Data Selection and Preprocessing
    a) Describe the steps for preprocessing including equations, algorithms, etc.
3) Clustering Analysis
    a) Perform clustering techniques on the dataset using our options from above
    b) Evaluate the performance with validation metrics such as entropy, purity, etc.
    c) Compare different methods strengths and weaknesses

DATA DESCRIPTION

Link to Dataset: https://archive.ics.uci.edu/dataset/113/twenty+newsgroups

18846 rows x 2 columns

The dataset is structured with the first column including the text. The second column has a category number 1-20. This category number is associated with one of the twenty newsgroup categories in the dataset. The categories are:

1) alt.atheism
2) comp.graphics
3) comp.os.ms-windows.misc
4) comp.sys.ibm.pc.hardware
5) comp.sys.mac.hardware
6) comp.windows.x
7) misc.forsale
8) rec.autos
9) rec.motorcycles
10) rec.sport.baseball
11) rec.sport.hockey
12) sci.crypt
13) sci.electronics
14) sci.med
15) sci.space
16) soc.religion.christian
17) talk.politics.guns
18) talk.politics.mideast
19) talk.politics.misc
20) talk.religion.misc

After Preprocessing

After Preprocessing, we decided that the best option was Bag of Words. The shape of the datasets was changed, keeping the same amount of rows, but changing the number of columns to reflect the amount of unique words contained within the message. The end result is a dataset shape of 18846 rows x 173762

PRE-PROCESSING

- TF-IDF:
    - initialized a TF-IDF vectorizer object with English stop words removed
    - applied the TF-IDF vectorization to the text data (newsgroups_data.data) using the initialized TF-IDF vectorizer. It converts the text data into a TF-IDF matrix (X_tfidf), where each row represents a document and each column represents a term weighted by its TF-IDF score
    - applied PCA to the TF-IDF matrix (X_tfidf) to reduce its dimensionality to 2 components while preserving the most important information
- Vectorization - Bag of Words
    - BASED ON CODE:
        - Tokenization: The CountVectorizer class automatically tokenizes the text data provided to it. Tokenization is the process of splitting the text into individual words or tokens.
        - Creating a Vocabulary: After tokenization, CountVectorizer creates a vocabulary from all the unique tokens (words) in the text data. Each word in the vocabulary is assigned an index.

        - Vectorization: The fit_transform method of CountVectorizer transforms the text data into a numerical format called the Bag of Words (BoW) representation. In this representation, each document is represented as a vector where each element corresponds to the count of a word from the vocabulary in that document.

- Normalization - MaxAbsScaler and StandardScaler
    - MaxAbsScaler: This scaler scales each feature by its maximum absolute value, which is useful for scaling sparse data. The formula for MaxAbsScaler is: X(scaled) = X / MAX(|X|)
    - Standard Scaler: This scaler standardizes features by removing the mean and scaling to unit variance. The formula for StandardScaler is: X(scaled) = (X-mean)/standard deviation

PSEUDOCODE

```
- TF-IDF:
    - Step 1: Calculate Term Frequency (TF)
    - for each document in corpus:
        - initialize term frequency vector tf_vector for the document
    - for each term in document:
        - increment the count of term in tf_vector
    - Step 2: Calculate Inverse Document Frequency (IDF)
    - initialize document frequency df_vector for all terms in the corpus
    - for each document in corpus:
        - for each term in document:
```

```
        - if term appears in document:
            - increment df_vector[term]
    - calculate inverse document frequency idf_vector for all terms:
        - for each term in corpus:
            - idf_vector[term] = log(total_documents / (1 + df_vector[term]))
    - Step 3: Calculate TF-IDF
    - initialize tfidf_matrix for all documents in corpus
    - for each document in corpus:
        - for each term in document:
            - tfidf_matrix[document][term] = tf_vector[term] * idf_vector[term]
- Bag of Words Algorithm:
    - Create initial dictionary which maps each words to a count
    - For each word in the message, increase the count of the word
    - Initialize a vector a list of zeroes
    - For every word, set its index equal to its count
- MaxAbs Scaler
    - Initialize arrays to store maximum abs values for features
    - For each feature i in X, find the max abs value across all samples
    - For each sample j in X, divide the value of feature i in sample j by max abs value
- K-Means Algorithm:
    - Initialize cluster centroids
    - Assign data points to nearest centroids
    - Update centroids based on assigned data points
```

        - Repeat until convergence or maximum iterations
    - Hierarchical Clustering Algorithm
        - Initialize a distance matrix to store distances of pairs
        - Initialize a cluster list where each point is a cluster
        - Initialize a cluster distance matrix where each row and column is a cluster
        - Find the two closest clusters, merge them, update the distance matrix, and merge to two clusters

VALIDATION

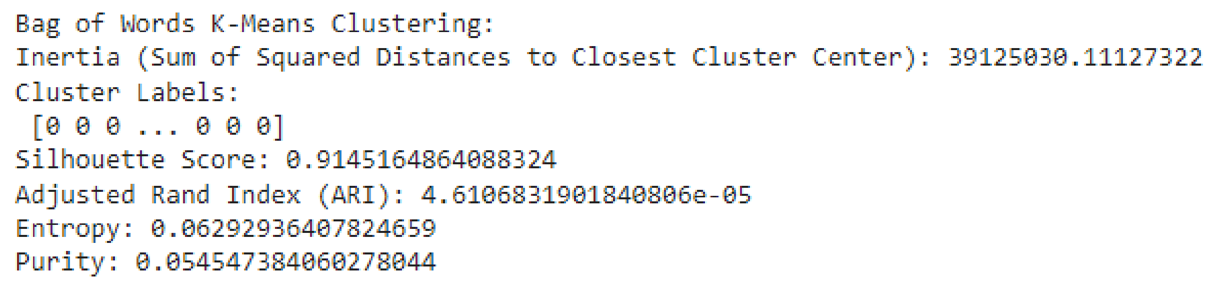
- Silhouette Score: The Silhouette Score is a measure of how similar an object is to its own cluster (cohesion) compared to other clusters (separation). It ranges from -1 to 1, where:
    - 1 indicates that the object is well-matched to its own cluster and clearly separated from neighboring clusters.
    - 0 indicates that the object is on or very close to the decision boundary between clusters.
    - -1 indicates that the object may have been assigned to the wrong cluster.
- The Silhouette Score is useful for evaluating the quality of clusters produced by clustering algorithms such as K-means.
- Adjusted Rand Index (ARI) Score: The Adjusted Rand Index measures the similarity between two clusterings, taking into account chance. It compares how pairs of data points are assigned in the true clustering versus the predicted clustering. ARI ranges from -1 to 1, where:
    - 1 indicates perfect similarity between the true and predicted clusterings.
    - 0 indicates random labeling (no agreement).

    - -1 indicates perfect dissimilarity between the true and predicted clusterings.
- ARI is often used when the ground truth labels (true clustering) are known and used for evaluating clustering algorithms.
- Entropy: Used to evaluate the quality of clustering results by assessing the degree of disorder or uncertainty within the clusters. Lower entropy values indicate better clustering performance, suggesting that the clusters are more homogeneous and well-separated.
    - Ranges from 0 to log2(k), where k is the number of clusters
    - A 0 would indicate perfect clustering
- Purity: Used to evaluate the quality of clustering results by seeing which clusters contain data points from a single class. It calculates the proportion of the majority class within each cluster, with higher purity values indicating better clustering performance and greater homogeneity within clusters.
    - Ranges from 0 to 1
    - A 1 would indicate perfect clustering

OUTPUTS

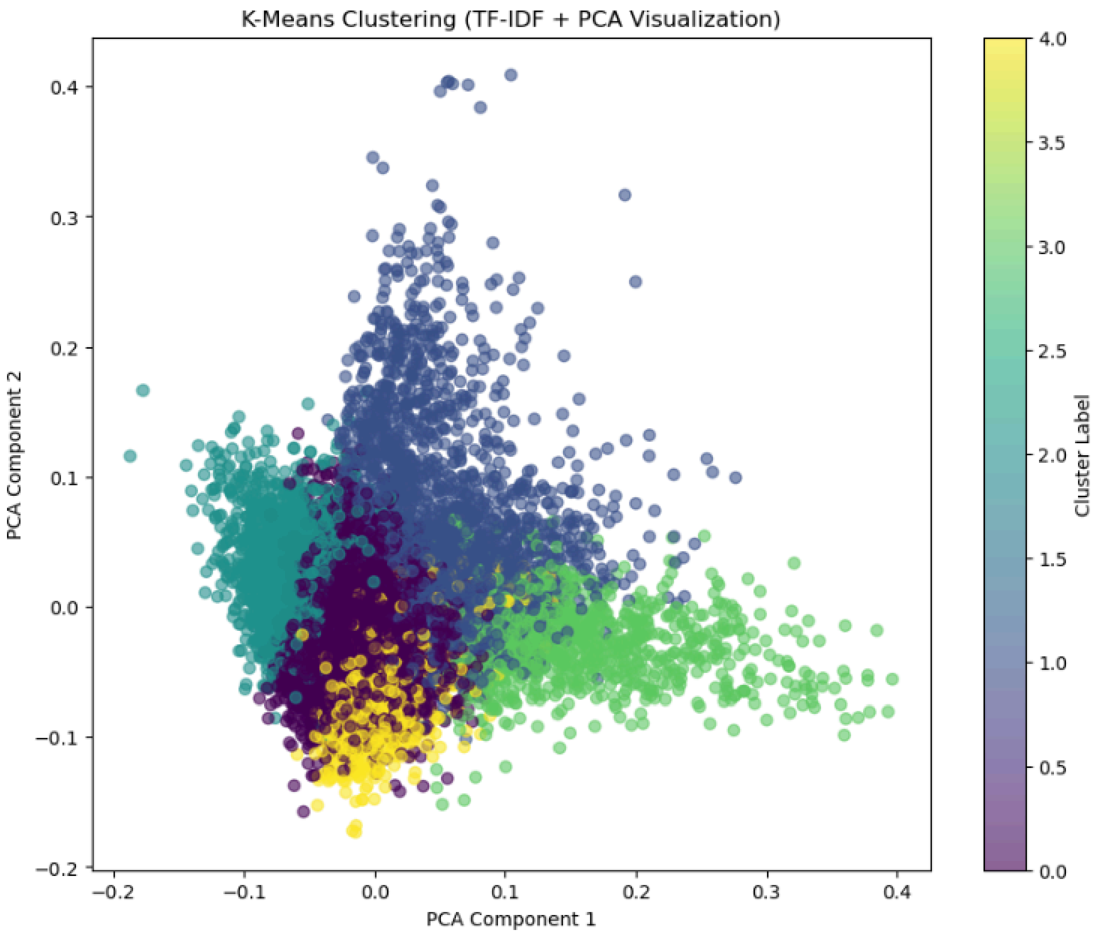
After going through the process of finding the data, pre-processing the data, running the clustering algorithms, and finding ways to validate its effectiveness, this is the final results for the two clustering algorithms we chose:

K-Means Clustering



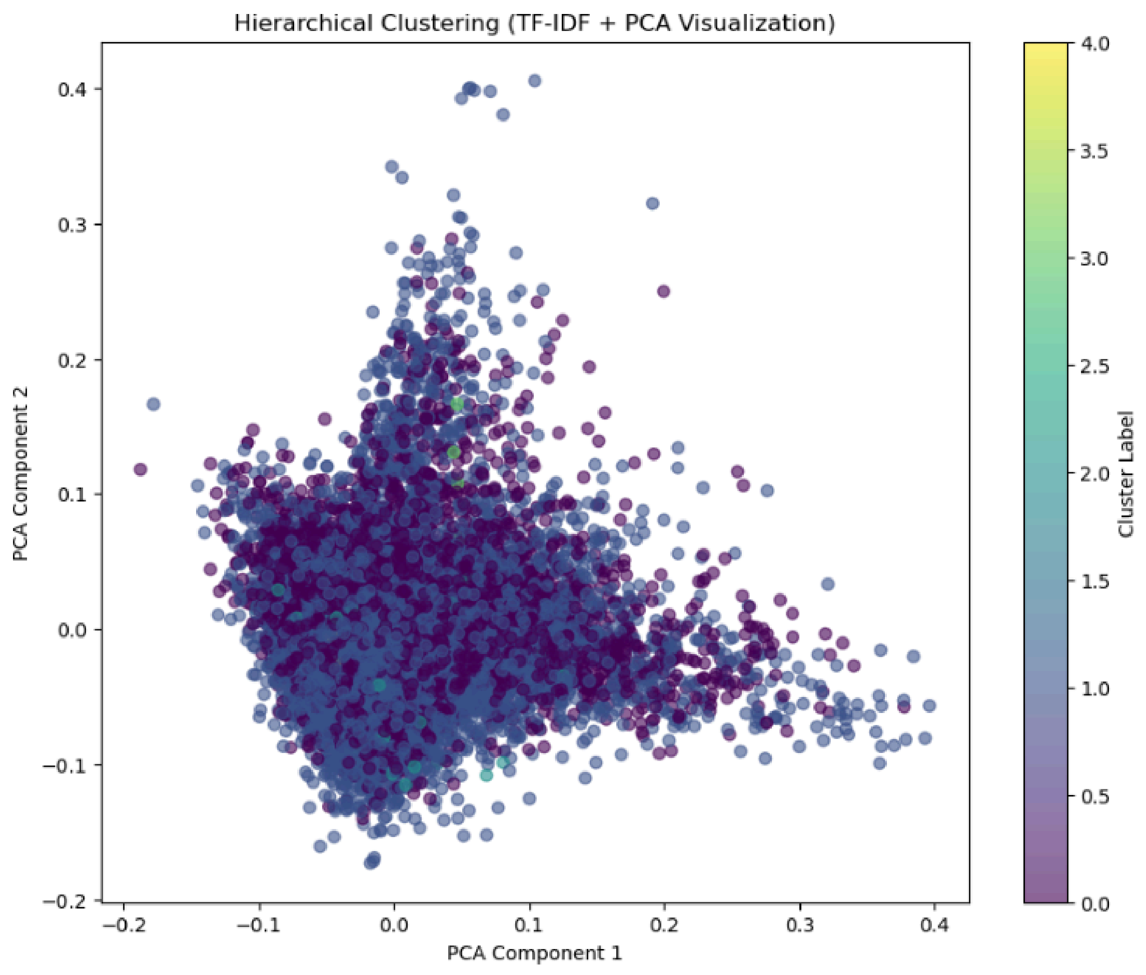
```
Bag of Words K-Means Clustering:
Inertia (Sum of Squared Distances to Closest Cluster Center): 39125030.11127322
Cluster Labels:
 [0 0 0 ... 0 0 0]
Silhouette Score: 0.9145164864088324
Adjusted Rand Index (ARI): 4.6106831901840806e-05
Entropy: 0.06292936407824659
Purity: 0.054547384060278044
```

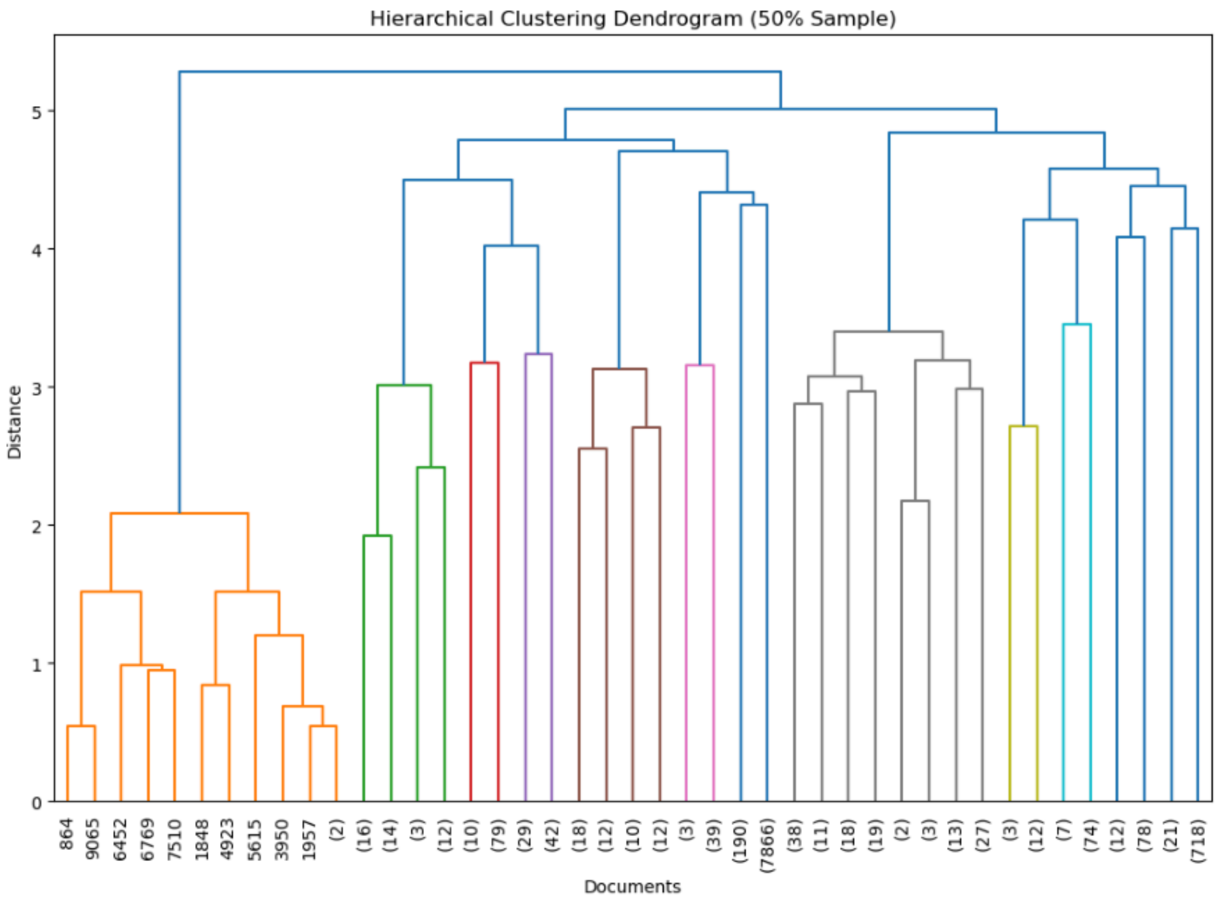
```
MaxAbs Scaling K-Means Clustering:
Inertia (Sum of Squared Distances to Closest Cluster Center): 413537.1412599124
Cluster Labels:
 [2 2 2 ... 2 2 2]
Silhouette Score: -0.06375217547433625
Adjusted Rand Index (ARI): 2.442950856846669e-06
Entropy: 0.004768471640512126
Purity: 0.05332696593441579
```

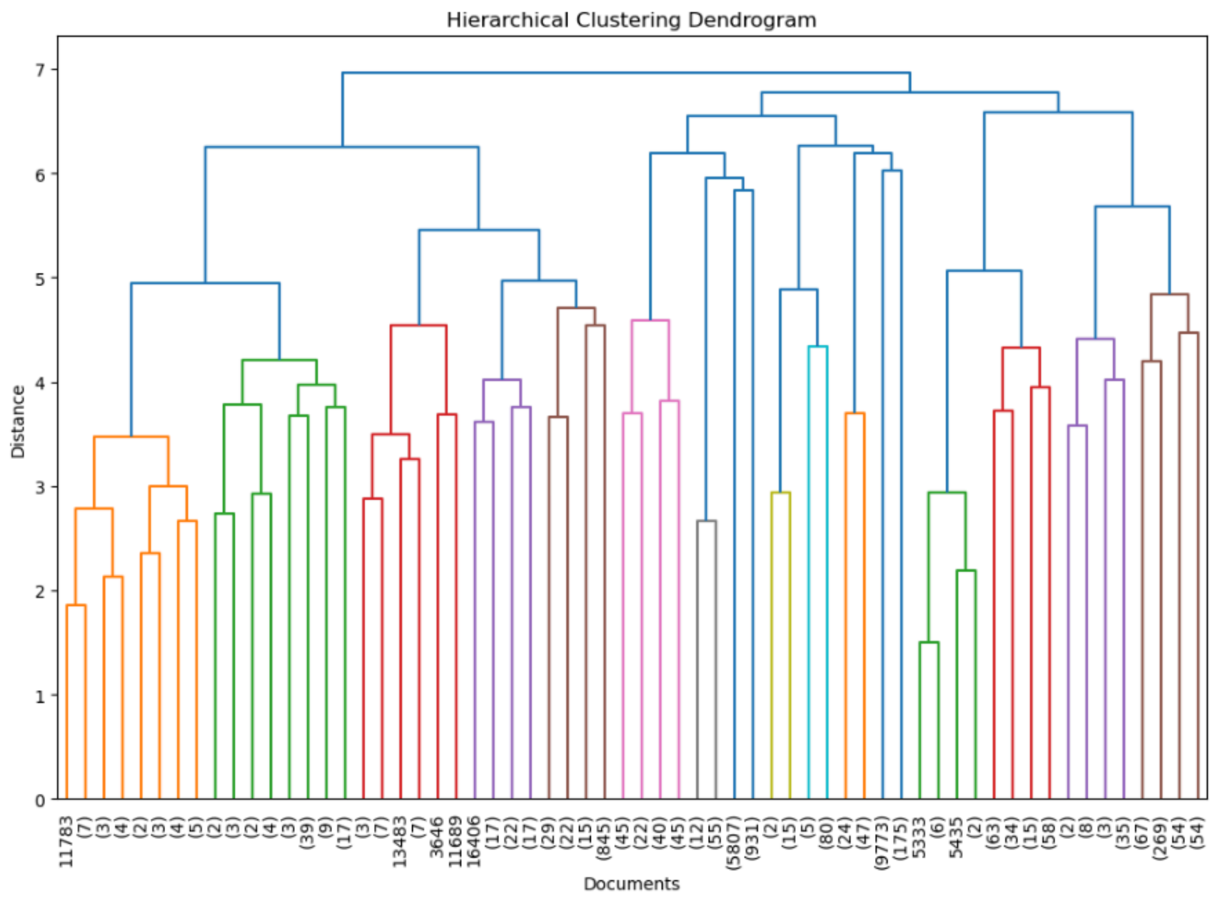


Hierarchical Clustering

```
Hierarchical Clustering:
Cluster Labels:
 [2 0 0 ... 1 1 0]
Silhouette Score: -0.035577021357558074
Adjusted Rand Index (ARI): 0.0016327972311052964
```




Hierarchical Clustering Dendrogram (50% Sample)
Distance
Documents
0
1
2
3
4
5
864
9065
6452
6769
7510
1848
4923
5615
3950
1957
(2)
(16)
(14)
(3)
(12)
(10)
(79)
(29)
(42)
(18)
(12)
(10)
(12)
(3)
(39)
(190)
(7866)
(38)
(11)
(18)
(19)
(2)
(3)
(13)
(27)
(3)
(12)
(7)
(74)
(12)
(78)
(21)
(718)



ANALYSIS AND CONCLUSION

Pre-processing:

We decided to primarily go with the vectorization method for pre-processing of Bag of Words (BoW). Not only was this necessary for clustering the dataset we were given, it is extremely helpful in the context of text processing. It is very efficient due to its nature of using matrices which are memory-efficient and easy to use for large datasets, it both captures and keeps the frequency of words within documents, and it is easily scalable which helps as we need to keep track of large amounts of words.

We chose MaxAbsScaler for our second preprocessing even though Bag of Words can function on its own. MaxAbsScaler can handle matrices efficiently so it was an easy choice as we were already using Bag of Words. It helps by preserving the sparseness of the dataset while making it less sensitive to outliers that would negatively affect the training. It also computationally efficient over large dimensionality which ours certainly has after applying Bag of Words.

We also utilized TF-IDF and removed all stop words for preprocessing.

Clustering:

KMeans

Using Bag of Words, the clustering algorithm returned these validation scores:

```
Silhouette Score: 0.9145164864088324

Adjusted Rand Index (ARI): 4.6106831901840806e-05

Entropy: 0.06292936407824659

Purity: 0.054547384060278044
```

- The silhouette score close to 1 indicates that the clusters themselves are tight, with data points within that cluster close to each other and far from others.
- The ARI score suggests that the clusterings are essentially random and have little similarity
- The Entropy score is close to 0, indicating the individual clustering performances have low uncertainty within them
- The purity score is also close to 0, indicating that there could be a mix of classes within some clusters. This could be a result of the labels being similar to each other.

Using MaxAbsScaler, the clustering algorithm returned these validation scores:

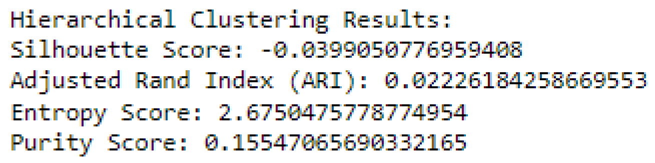
```
Silhouette Score: -0.06375217547433625

Adjusted Rand Index (ARI): 2.442950856846669e-06

Entropy: 0.004768471640512126

Purity: 0.0533269659344157
```

- The silhouette score in the negative indicates that after scaling, the clustering is poor. This is due to the scaling nature of the processing and moving some clustering decisions closer to one another after scaling.
- The ARI score suggests that the clusterings are essentially random and have little similarity
- The entropy score is extremely low, indicating that there is still low uncertainty within the actual clustering assignments. This is much lower than before scaling and largely as a result of it as the homogeneity of the clusters increase.
- The purity is still somewhat similar as before which is to be expected, if there is a mix of classes within some classes that is not expected to change when scaling the data.

KMeans works well with this dataset after we have changed the original dataset into one with a large amount of dimensions. The defined labels provided also help extremely well as it allows the algorithm to cluster data points correctly and with confidence as shown in the silhouette and entropy scores above.

Hierarchical Clustering

Using Hierarchical Clusting the algorithm returns:



```
Hierarchical Clustering Results:
Silhouette Score: -0.0399050776959408
Adjusted Rand Index (ARI): 0.02226184258669553
Entropy Score: 2.6750475778774954
Purity Score: 0.15547065690332165
```

- The silhouette score is slightly above 0 suggesting that there is little agreement between the clusters and true labels.

- The ARI score suggests that the clusterings are essentially random and have little similarity
- The entropy score is relatively high indicating that there is high disorder and uncertainty or overlap in the clustering results.
- The purity is low meaning that clusters are not pure and further suggests poor clustering.

Based on the provided metrics, the hierarchical clustering for your data did not perform well in capturing the inherent patterns or groupings present in the dataset.