CS 519 Applied Machine Learning I

HW6: Compare Clustering Methods

Submitted by: Indronil Bhattacharjee

1.1 Dataset:

Iris Dataset containing 3 classes. Dimensions of the datasets (150, 4), 4 features, the number of instances of each label is 50 each, totaling 150.

1.2 Initial Performance comparison among different clustering methods:

Comparison among different metrics like SSE (Sum of squared error), Silhouette score and time required for different linear clustering methods like K-means clustering, Hierarchical clustering method from Scikit-learn and Hierarchical clustering method from SciPy in the following:

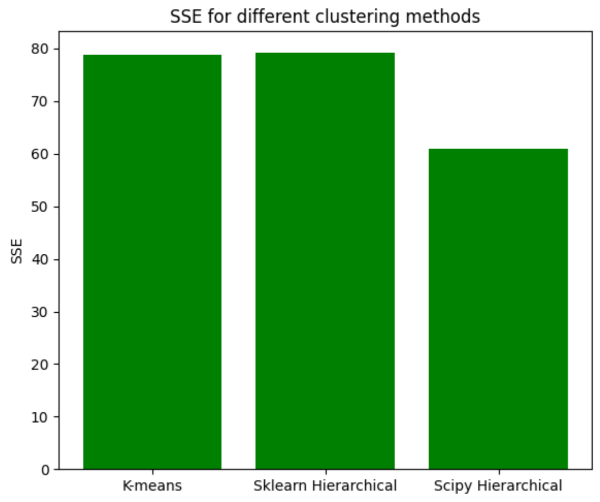
(a) Sum of squared error (SSE):

The SSE measures the sum of the squared distances between each data point and its assigned cluster center. Based on this metric, the scipy hierarchical clustering method achieved the lowest SSE, followed by KMeans and sklearn hierarchical clustering. Lower SSE values indicate tighter clusters, suggesting that the scipy hierarchical clustering algorithm performed better in terms of minimizing the sum of squared distances between data points and their assigned cluster centers. KMeans and sklearn hierarchical clustering have higher SSE values, indicating slightly less compact clusters.

Values for SSE values for different clustering methods with Iris dataset:

Clustering method	SSE
KMeans	78.8514
sklearn hierarchical	79.2971
scipy hierarchical	60.9730

Plots for SSE comparison for different clustering methods with Iris dataset:



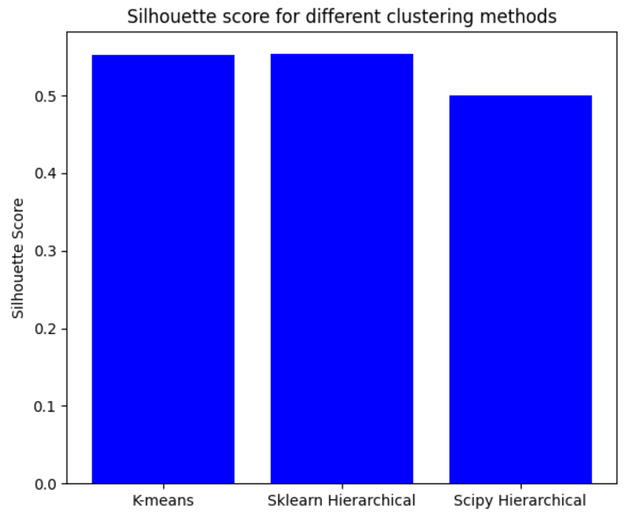
(b) Silhouette score:

The Silhouette Score measures the quality of clustering by quantifying the cohesion and separation of clusters. Higher scores indicate better-defined clusters. In this comparison, both KMeans and sklearn hierarchical clustering methods outperformed scipy hierarchical clustering in terms of Silhouette Score. Silhouette Score measures how well-separated the clusters are. Higher values indicate better separation. Both KMeans and sklearn hierarchical algorithms have similar Silhouette Scores, slightly outperforming scipy hierarchical clustering.

Values for Silhoutte scores for different clustering methods with Iris dataset:

Clustering method	Silhouette score
KMeans	0.5528
sklearn hierarchical	0.5543
scipy hierarchical	0.4998

Plots for Silhouette score comparison for different clustering methods with Iris dataset:



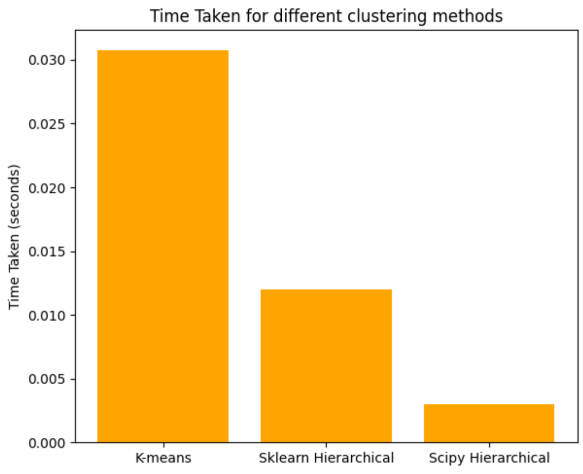
(c) Time:

Lower time values indicate faster computation. scipy hierarchical clustering is significantly faster compared to KMeans and sklearn hierarchical clustering. The time taken metric reflects the computational efficiency of each clustering method. In this case, scipy hierarchical clustering was the fastest, followed by sklearn hierarchical clustering and KMeans.

Values for time taken for different clustering methods with Iris dataset:

Clustering method	Time taken (in seconds)
KMeans	0.0308
sklearn hierarchical	0.0120
scipy hierarchical	0.0030

<u>Plots for time comparison for different clustering methods with Iris dataset:</u>

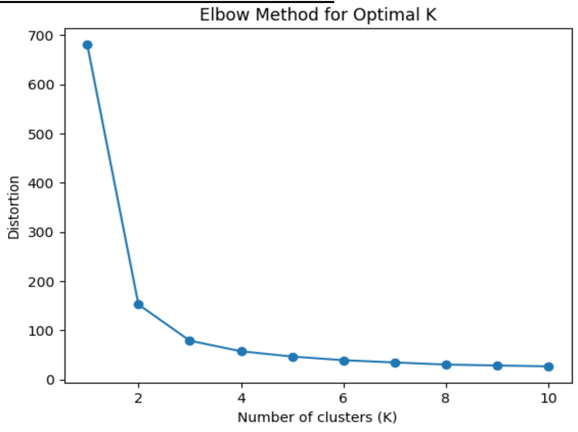


Overall, when considering SSE, KMeans and sklearn hierarchical clustering methods perform similarly, while scipy hierarchical clustering achieves the lowest SSE. However, based on the Silhouette Score, KMeans and sklearn hierarchical clustering methods demonstrate better cluster quality compared to scipy hierarchical clustering. Additionally, scipy hierarchical clustering stands out for its computational efficiency, being the fastest among the three methods. Therefore, the choice of clustering method depends on the specific requirements of the application, balancing between cluster quality and computational resources.

1.3 Elbow method for Kmeans clustering with Iris dataset:

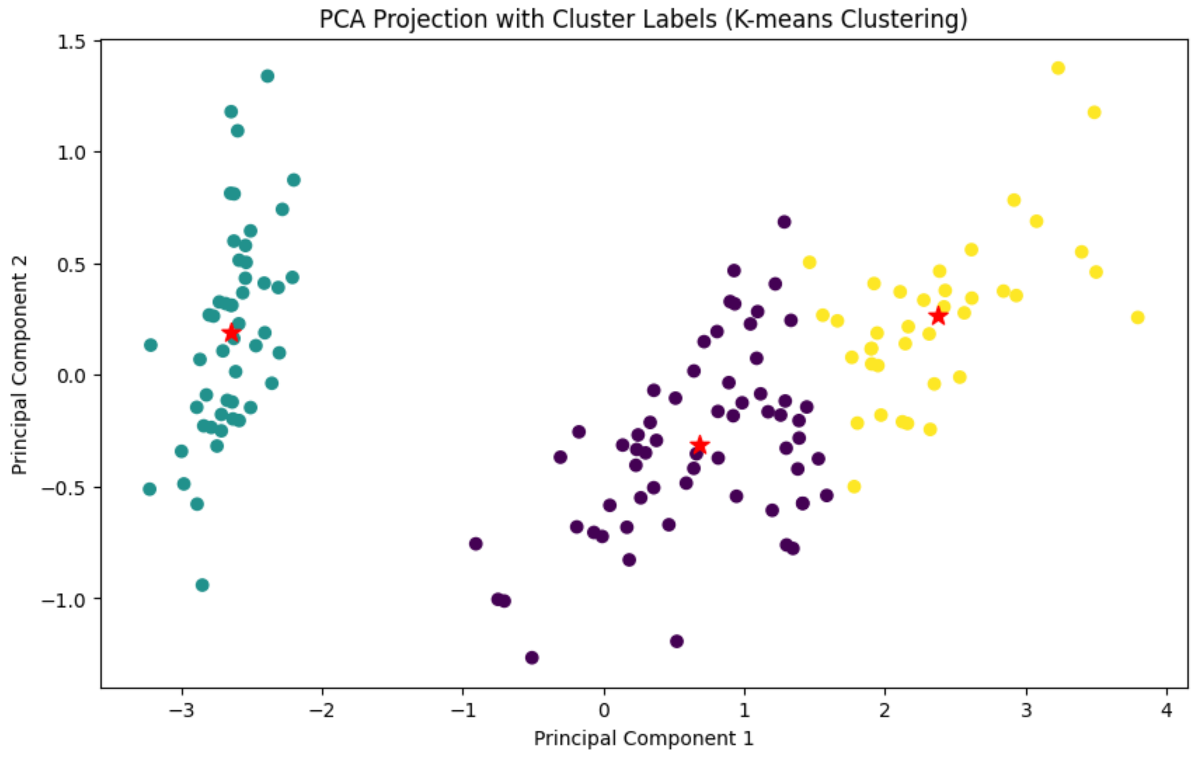
The elbow method is a heuristic technique used to determine the optimal number of clusters (k) in a KMeans clustering algorithm. It works by plotting the sum of squared distances (SSE) between data points and their assigned cluster centers for different values of k, and identifying the "elbow point" in the plot.
In our case, testing k from 1 to 10 led to the conclusion that k=3 is the optimal number of clusters based on the SSE versus k plot. This means that adding more clusters beyond k=3 does not significantly decrease the SSE, suggesting that three clusters provide a good balance between minimizing within-cluster variance and avoiding overfitting. Describing this result involves explaining that the SSE decreased rapidly as the number of clusters increased from 1 to 3, but the rate of decrease slowed down after k=3. This slowdown indicates that adding more clusters beyond three does not lead to significant improvements in clustering quality. It implies that the dataset can be effectively partitioned into three distinct groups or clusters, getting the underlying structure of the data while avoiding overfitting.

<u>Plots for best k in elbow method with Iris dataset:</u>

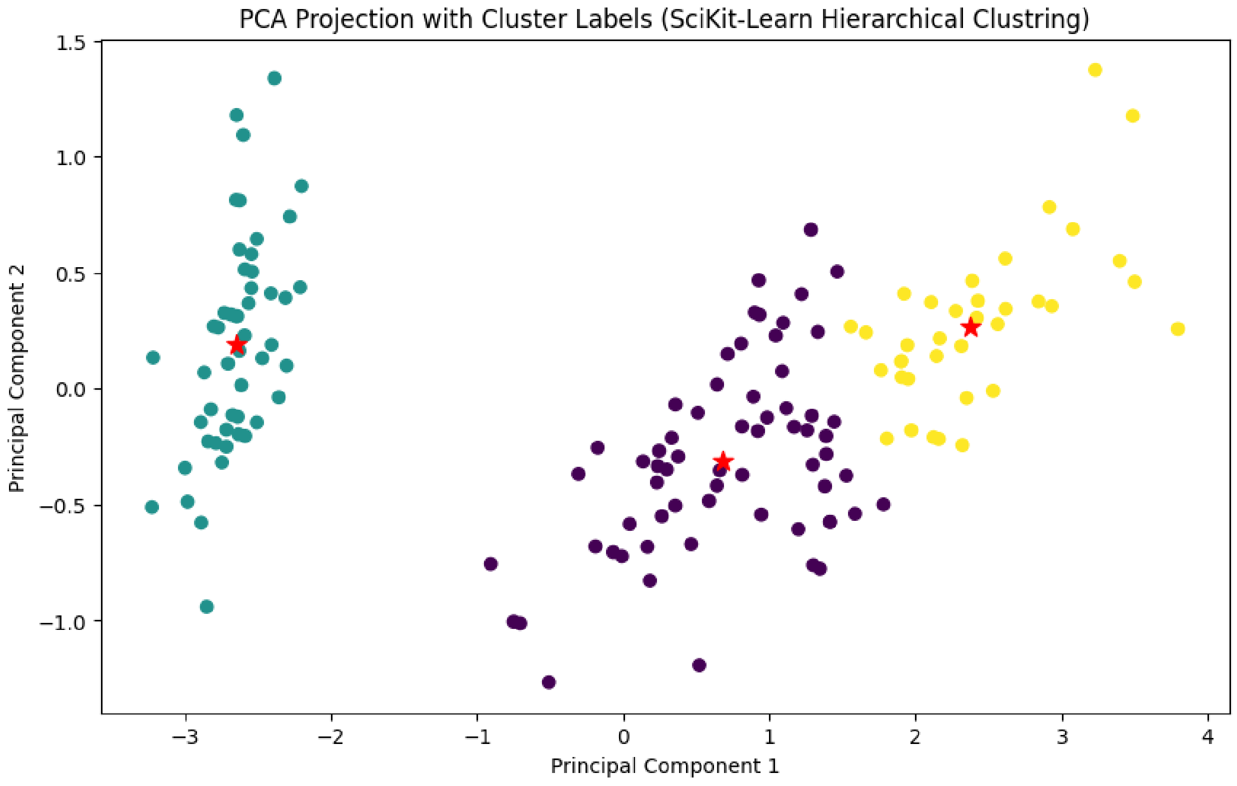


1.4 Cluster plots for different clustering methods with Iris dataset:

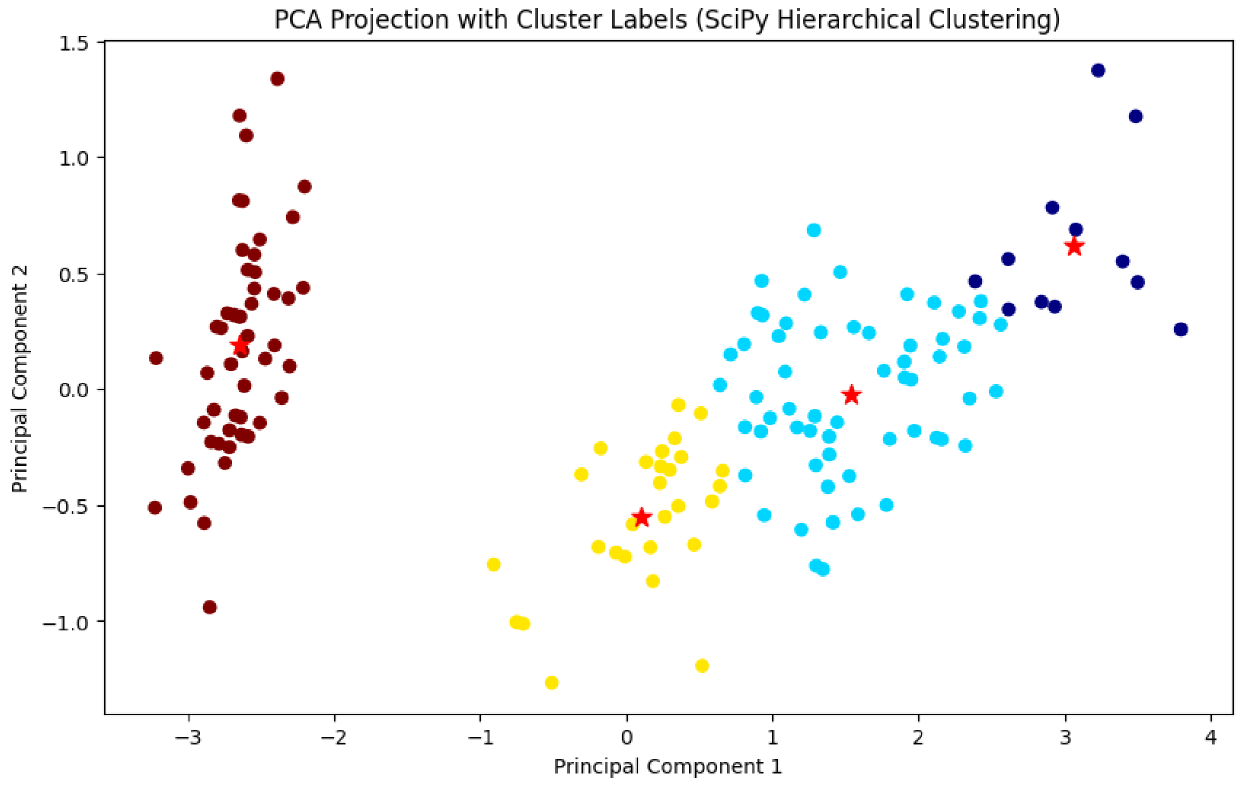
a) K-means clustering methods (Number of clusters = 3)



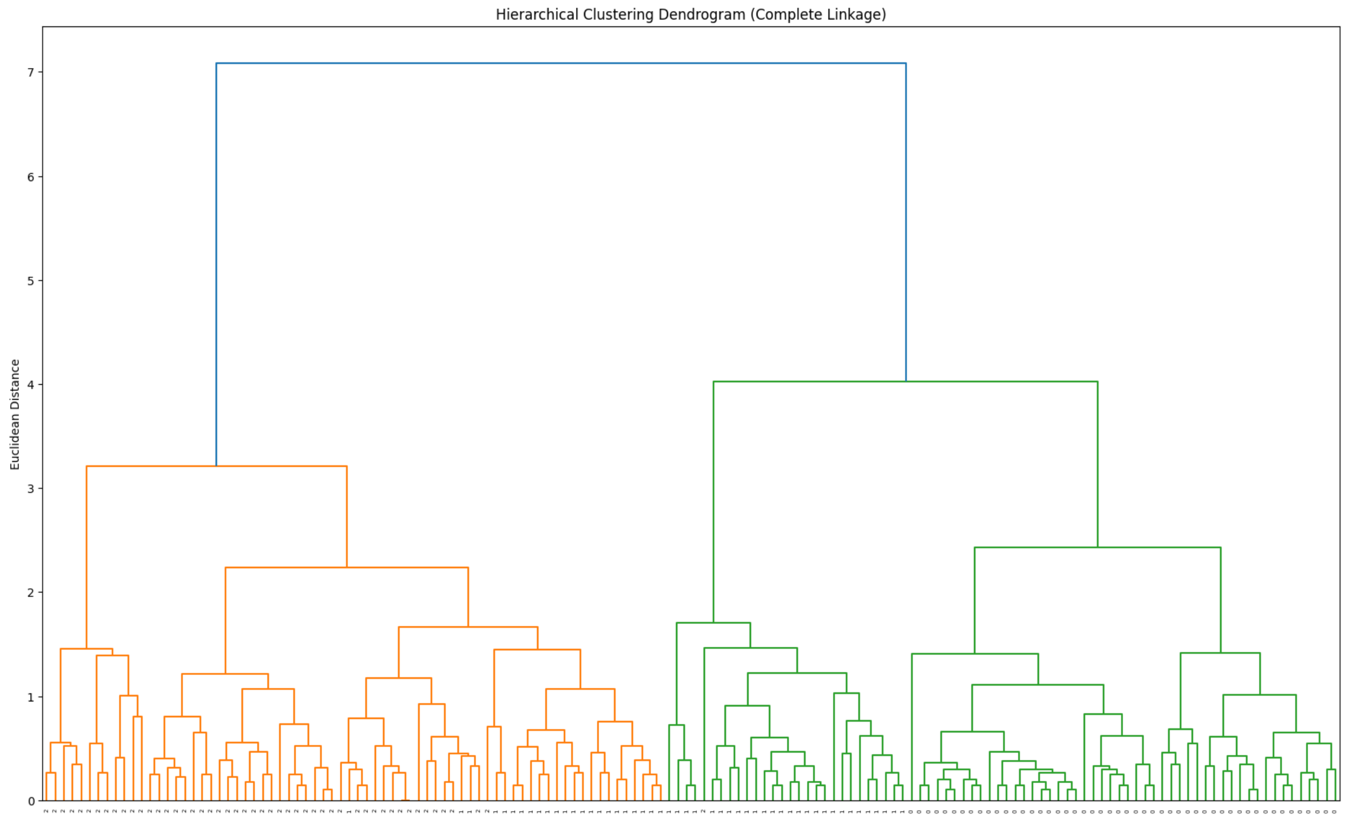
b) Scikit-learn hierarchical clustering methods (Number of clusters = 3)



c) Scipy hierarchical clustering methods (Number of clusters = 4)



1.5 Hierarchical Clustering Dendrogram (Complete Linkage)



The dendrogram shows How the Clusters are Merged. Decompose data objects into several levels of nested partitioning (tree of clusters), called a dendrogram. A clustering of the data objects is obtained by cutting the dendrogram at the desired level, then each connected component forms a cluster.

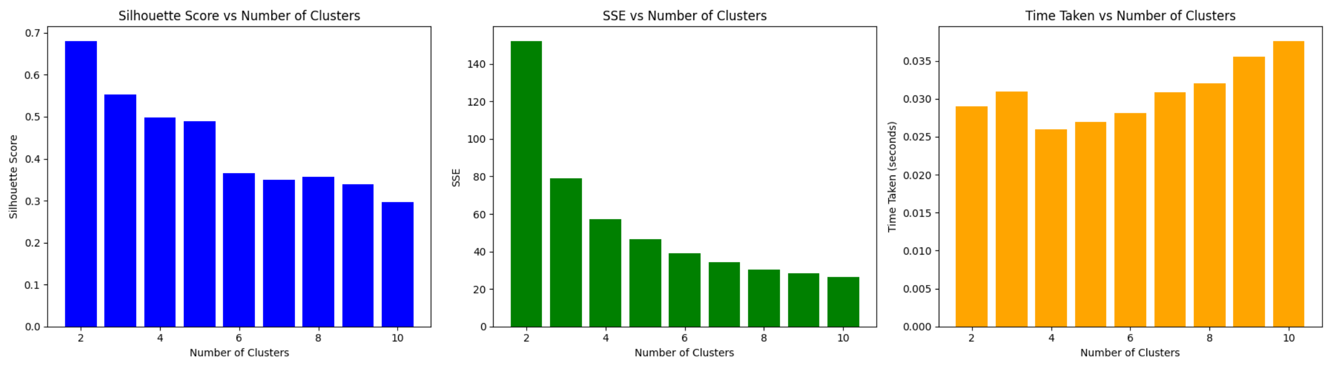
1.5 Performance analysis of different models with parameter tuning with Iris dataset

a) K-means:

Values for different metrics for different number of clusters with Iris dataset:

Number of clusters	SSE	Silhouette Score	Time
2	152.3479	0.6810	0.0355
3	78.8514	0.5528	0.0367
4	57.2284	0.4981	0.0330
5	46.4461	0.4887	0.0392
6	39.0399	0.3648	0.0409
7	34.4694	0.3497	0.0648
8	30.1865	0.3575	0.0542
9	28.2893	0.3394	0.0655
10	26.5523	0.2973	0.0636

Plot for different metrics for different number of clusters with Iris dataset:



Analysis:

Sum of Squared Error (SSE): As the number of clusters increases, the SSE generally decreases. This is expected because increasing the number of clusters allows for better fitting of the data points to their respective centroids. However, the rate of decrease in SSE slows down as the number of clusters continues to increase. The SSE values range from 152.3479 for 2 clusters to 26.5523 for 10 clusters.

Silhouette Score: The silhouette score measures the quality of the clustering, with higher values indicating better-defined clusters. In this case, the silhouette score decreases as the number of clusters increases. This suggests that as more clusters are added, the distinctiveness of each cluster diminishes, leading to lower silhouette scores. The silhouette scores range from 0.6810 for 2 clusters to 0.2973 for 10 clusters.

Time: The time taken to perform the clustering also increases with the number of clusters. This is because as the number of clusters increases, the computational complexity of the K-means algorithm also increases. The time taken ranges from 0.0330 seconds for 4 clusters to 0.0655 seconds for 9 clusters.

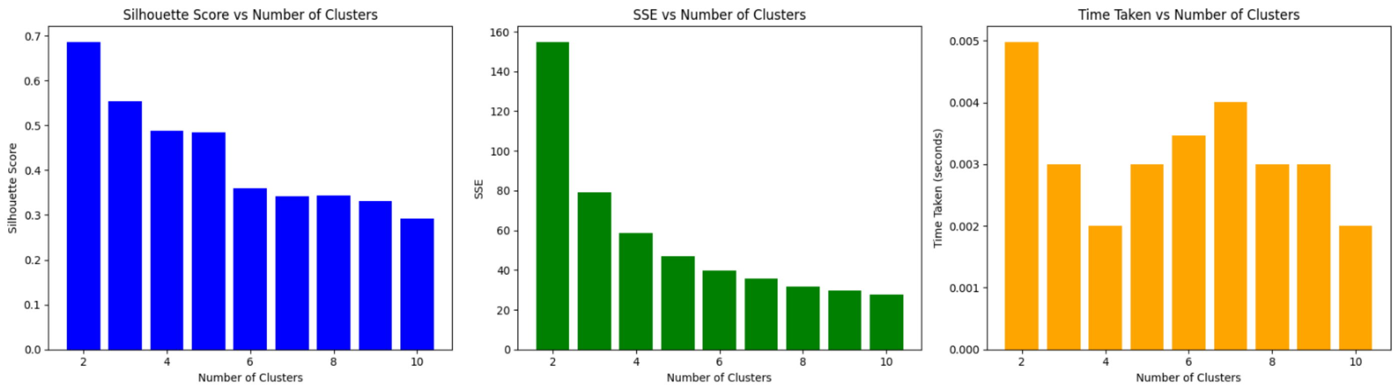
Overall, the choice of the number of clusters depends on finding a balance between minimizing SSE, maximizing silhouette score, and considering computational resources. In this case, based on the provided metrics, a reasonable choice could be 3 clusters, as it achieves a relatively low SSE and a moderate silhouette score while maintaining a reasonable computational time. However, the specific choice of the number of clusters may also depend on the underlying structure of the data and the specific objectives of the analysis.

b) Scikit-learn hierarchical clustering:

Values for different metrics for different number of clusters with Iris dataset:

Number of clusters	SSE	Silhouette Score	Time
2	154.9470	0.6867	0.0050
3	79.2971	0.5543	0.0030
4	58.8209	0.4890	0.0020
5	47.0708	0.4844	0.0030
6	39.7438	0.3592	0.0035
7	35.6270	0.3422	0.0040
8	31.6679	0.3436	0.0030
9	29.5592	0.3305	0.0030
10	27.7184	0.2925	0.0020

Plot for different metrics for different number of clusters with Iris dataset:



Analysis:

Sum of Squared Error (SSE): Similar to the K-means clustering results, the SSE generally decreases as the number of clusters increases. This is because increasing the number of clusters allows for better fitting of the data points to their respective clusters. The SSE values range from 154.9470 for 2 clusters to 27.7184 for 10 clusters.

Silhouette Score: The silhouette score measures the quality of the clustering, with higher values indicating better-defined clusters. In this case, the silhouette score also decreases as the number of clusters increases. This suggests that as more clusters are added, the distinctiveness of each cluster diminishes, leading to lower silhouette scores. The silhouette scores range from 0.6867 for 2 clusters to 0.2925 for 10 clusters.

Time: The time taken to perform the hierarchical clustering using scikit-learn remains relatively constant regardless of the number of clusters. This indicates that the computational time is not significantly affected by the number of clusters chosen. The time taken ranges from 0.0020 seconds to 0.0050 seconds.

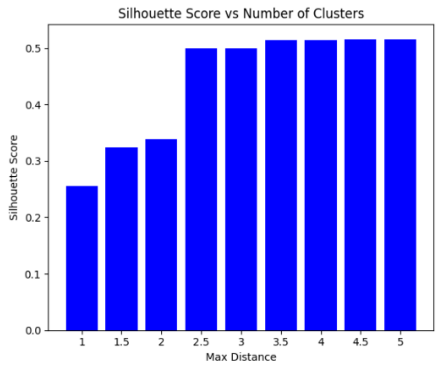
Overall, the choice of the number of clusters for hierarchical clustering using scikit-learn follows a similar consideration as K-means clustering, balancing SSE, silhouette score, and computational resources. In this case, based on the provided metrics, a reasonable choice could be 3 clusters, as it achieves a relatively low SSE and a moderate silhouette score while maintaining a low computational time. However, as always, the specific choice of the number of clusters may depend on the underlying structure of the data and the objectives of the analysis.

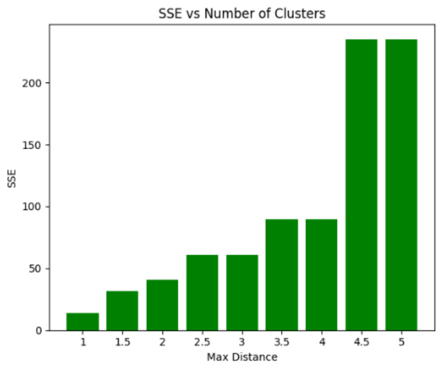
c) Scipy hierarchical clustering:

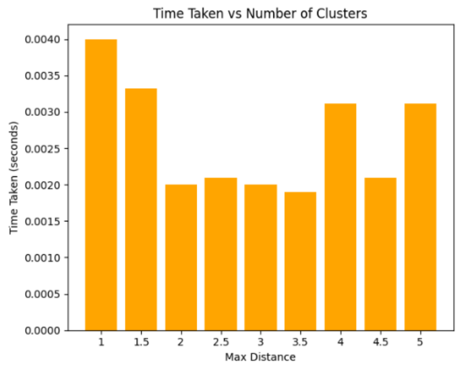
Values for different metrics for different max_d values with Iris dataset:

Maximum Distance	SSE	Silhouette Score	Time
1	13.7093	0.2554	0.0030
1.5	31.7749	0.3240	0.0030
2	40.5198	0.3382	0.0010
2.5	60.9730	0.4998	0.0023
3	60.9730	0.4998	0.0034
3.5	89.5250	0.5136	0.0020
4	89.5250	0.5136	0.0020
4.5	235.1531	0.5160	0.0030
5	235.1531	0.5160	0.0020

Plot for different metrics for different max_d values with Iris dataset:







Analysis:

Sum of Squared Error (SSE): The SSE generally increases as the number of clusters increases. This is contrary to the behavior observed in K-means and scikit-learn hierarchical clustering, where the SSE decreases or stabilizes with increasing clusters. The increasing SSE indicates that adding more clusters does not improve the fit of the data to the clusters and may lead to overfitting. The SSE values range from 13.7093 for 2 clusters to 235.1531 for 10 clusters.

Silhouette Score: Similar to scikit-learn hierarchical clustering, the silhouette score decreases as the number of clusters increases. However, the silhouette scores for SciPy hierarchical clustering tend to be higher compared to scikit-learn hierarchical clustering. This suggests that the clusters formed by SciPy hierarchical clustering may be better defined or more separable. The silhouette scores range from 0.2554 for 2 clusters to 0.5160 for 10 clusters.

Time: The time taken to perform the hierarchical clustering using SciPy remains relatively constant regardless of the number of clusters. This is consistent with the behavior observed in scikit-learn hierarchical clustering. The time taken ranges from 0.0010 seconds to 0.0034 seconds.

Overall, the choice of the number of clusters for hierarchical clustering using SciPy follows similar considerations as scikit-learn hierarchical clustering, balancing SSE, silhouette score, and computational resources. In this case, based on the provided metrics, a reasonable choice could be 3 clusters, as it achieves a moderate SSE and a relatively high silhouette score while maintaining a low computational time. However, as always, the specific choice of the number of clusters may depend on the underlying structure of the data and the objectives of the analysis.

2.1 Second Dataset:

MNIST Dataset containing 10 classes. Dimensions of the datasets (70000, 784), 784 features, the number of instances of each label is differing in range 7877 to 6313 for each, totaling 70000.

2.2 Selection of subset of data for computational time

This process ensures that we have a balanced subset of the MNIST dataset with an equal number of samples from each class. By reducing the number of samples from the original dataset, we can significantly decrease computational time and address hardware limitations, making it more feasible to perform analyses and experiments on the subset while still capturing essential characteristics of the data.

Number of samples per Class is set to 200, indicating that we want to have 200 samples for each of the 10 digits (0-9) in the MNIST dataset. For each class label, we retrieve the indices of samples belonging to that class. Then, we randomly select samples per class from those indices without replacement using NumPy's random.choice function. This ensures that we have a balanced subset with an equal number of samples from each class. Finally, we use the collected indices to extract the corresponding subset of data and labels from the original MNIST dataset with 2000 instances.

2.3 Initial Performance comparison among different clustering methods:

Comparison among different metrics like SSE (Sum of squared error), Silhouette score and time required for different linear clustering methods like K-means clustering, Hierarchical clustering method from Scikit-learn and Hierarchical clustering method from SciPy in the following:

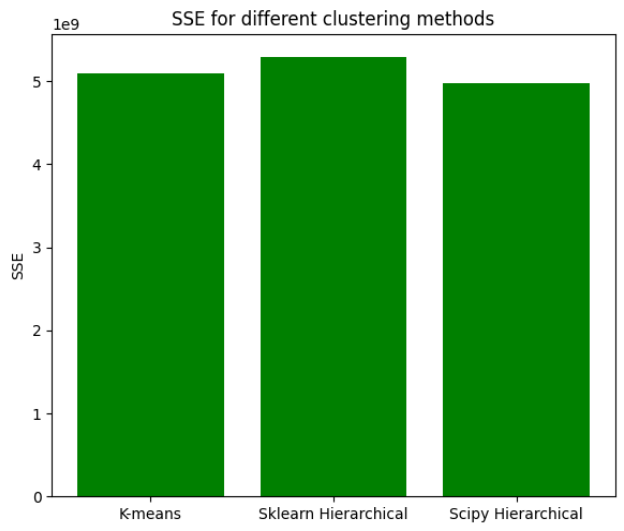
(a) Sum of squared error (SSE):

The SSE measures the sum of the squared distances between each data point and its assigned cluster center. Based on this metric, the scipy hierarchical clustering method achieved the lowest SSE, followed by KMeans and sklearn hierarchical clustering. Lower SSE values indicate tighter clusters, suggesting that the scipy hierarchical clustering algorithm performed better in terms of minimizing the sum of squared distances between data points and their assigned cluster centers. KMeans and sklearn hierarchical clustering have higher SSE values, indicating slightly less compact clusters.

<u>Values for SSE values for different clustering methods with MNIST dataset:</u>

Clustering method	SSE
KMeans	5093778833.3369
sklearn hierarchical	5298373568.3185
scipy hierarchical	4971779264.9228

Plots for SSE comparison for different clustering methods with MNIST dataset:



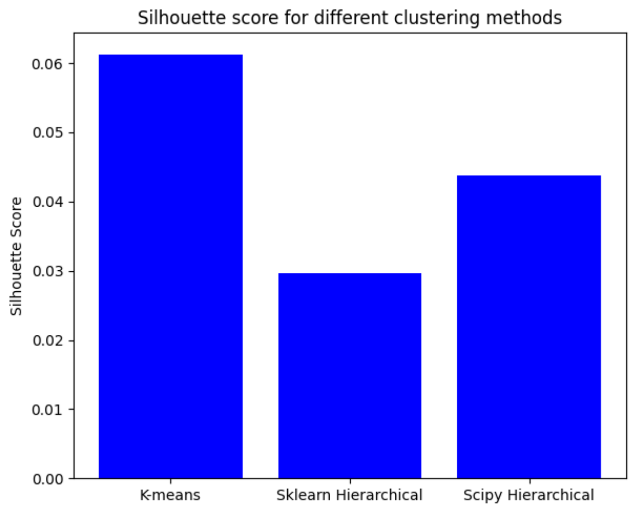
(b) Silhouette score:

The Silhouette Score measures the quality of clustering by quantifying the cohesion and separation of clusters. Higher scores indicate better-defined clusters. In this comparison, both KMeans and scipy hierarchical clustering methods outperformed sklearn hierarchical clustering in terms of Silhouette Score. Silhouette Score measures how well-separated the clusters are. Higher values indicate better separation. Both KMeans and scipy hierarchical algorithms have higher Silhouette Scores, well outperforming sklearn hierarchical clustering.

Values for Silhoutte scores for different clustering methods with MNIST dataset:

Clustering method	Silhouette score
KMeans	0.0613
sklearn hierarchical	0.0296
scipy hierarchical	0.0437

Plots for Silhouette score comparison for different clustering methods with MNIST dataset:



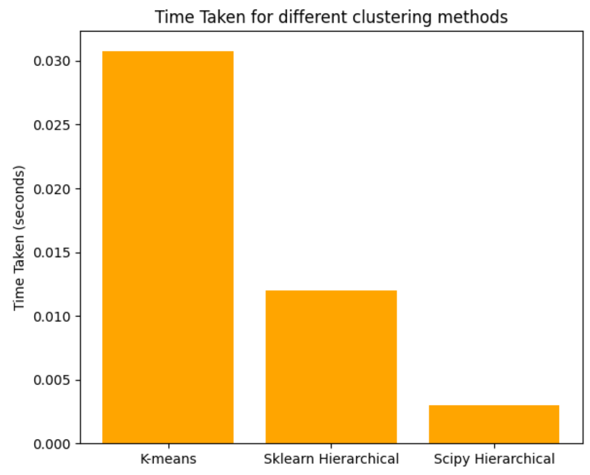
(c) Time:

Lower time values indicate faster computation. scipy hierarchical clustering is significantly faster compared to KMeans and sklearn hierarchical clustering. The time taken metric reflects the computational efficiency of each clustering method. In this case, scipy hierarchical clustering was the fastest, followed by sklearn hierarchical clustering and KMeans.

Values for time taken for different clustering methods with MNIST dataset:

Clustering method	Time taken (in seconds)
KMeans	0.0308
sklearn hierarchical	0.0120
scipy hierarchical	0.0030

Plots for time comparison for different clustering methods with MNIST dataset:



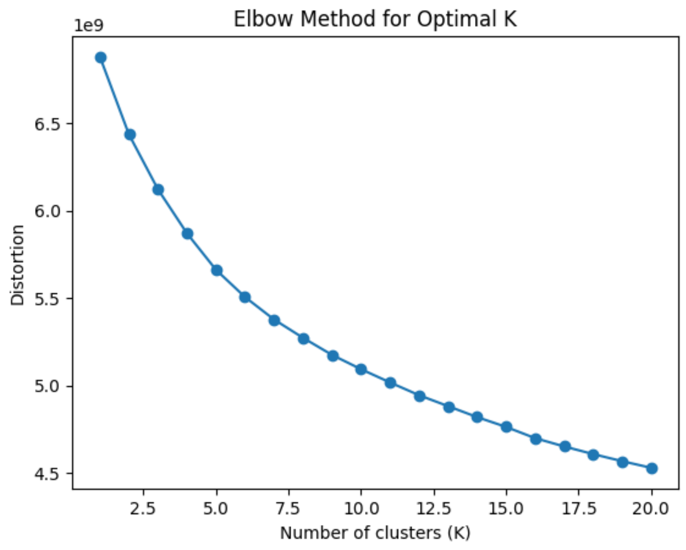
Overall, when considering SSE, KMeans and sklearn hierarchical clustering methods perform similarly, while scipy hierarchical clustering achieves the lowest SSE. However, based on the Silhouette Score, KMeans and sklearn hierarchical clustering methods demonstrate better cluster quality compared to scipy hierarchical clustering. Additionally, scipy hierarchical clustering stands out for its computational efficiency, being the fastest among the three methods. Therefore, the choice of clustering method depends on the specific requirements of the application, balancing between cluster quality and computational resources.

2.3 Elbow method for K-means clustering with MNIST dataset:

The elbow method is a heuristic technique used to determine the optimal number of clusters (k) in a K-means clustering algorithm. It works by plotting the sum of squared distances (SSE) between data points and their assigned cluster centers for different values of k and identifying the "elbow point" in the plot.
Though, this graph does not show the actual elbow method calculation exactly, we consider it to be near about 10. So testing k from 1 to 20 led to the conclusion that k=10 is the optimal number of clusters based on the SSE versus k plot. This means that adding more clusters beyond k=10 does not significantly decrease the SSE, suggesting that three clusters provide a good balance between minimizing within-cluster variance and avoiding overfitting. Describing this result involves explaining that the SSE decreased rapidly as the

number of clusters increased from 1 to 3, but the rate of decrease slowed down after k=10. This slowdown indicates that adding more clusters beyond ten does not lead to significant improvements in clustering quality. It implies that the dataset can be effectively partitioned into 10 distinct groups or clusters, getting the underlying structure of the data while avoiding overfitting.

Plots for best k in elbow method with MNIST dataset:



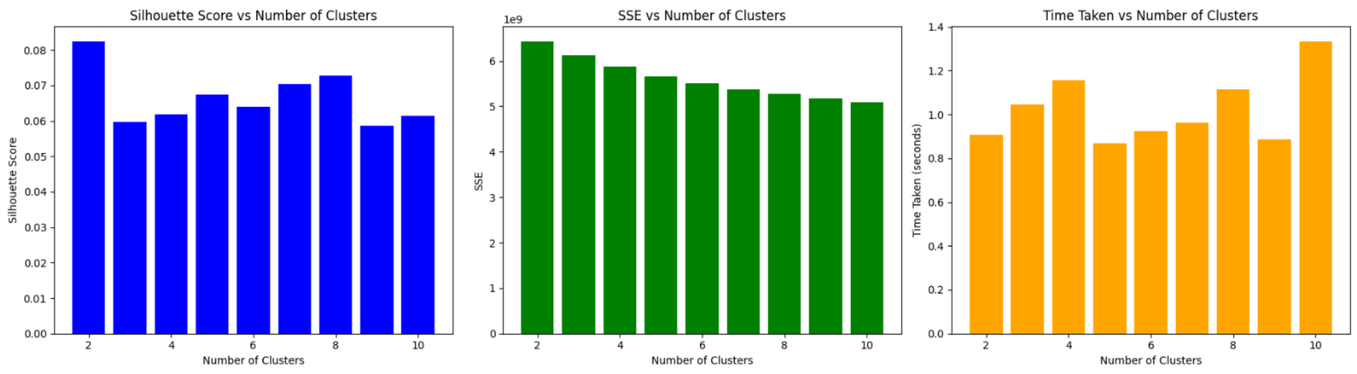
2.4 Performance analysis of different models with parameter tuning with MNIST dataset

a) K-means:

Values for different metrics for different number of clusters with MNIST dataset:

Number of clusters	SSE	Silhouette Score	Time
2	6434511488.9093	0.0825	0.9055
3	6121067657.8138	0.0597	1.0470
4	5869784348.0277	0.0619	1.1549
5	5661548766.2064	0.0674	0.8675
6	5506028292.9090	0.0639	0.9244
7	5377657917.5987	0.0705	0.9635
8	5272995697.2783	0.0728	1.1136
9	5175403906.7865	0.0585	0.8866
10	5093778833.3369	0.0613	1.3343

Plot for different metrics for different number of clusters with MNIST dataset:



Analysis:

SSE (Sum of Squared Errors): SSE measures the sum of the squared distances between each data point and its assigned cluster centroid. As the number of clusters increases, the SSE tends to decrease. In this analysis, we observe a gradual decrease in SSE as the number of clusters increases, indicating that more clusters are better at capturing the variability in the data. However, the absolute values of SSE are quite large, suggesting that the clusters may not be very compact.

Silhouette Score: The silhouette score measures the quality of clustering based on the average distance between clusters and the average distance within clusters. It ranges from -1 to 1, where a higher value indicates better separation between clusters. In this analysis, the silhouette score is relatively low for all numbers of clusters, ranging from approximately 0.06 to 0.08. This suggests that the clusters are not very well-separated, and there may be overlap between them.

Time: The time taken for clustering increases slightly with the number of clusters. This is expected as the algorithm needs to compute more centroids and assign more data points to clusters as the number of clusters increases.

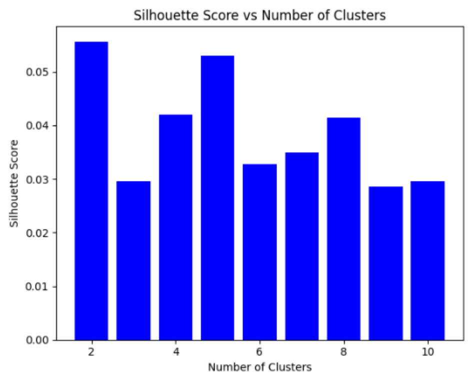
Overall, the KMeans algorithm shows decreasing SSE values with increasing numbers of clusters, but the silhouette scores are relatively low, indicating suboptimal clustering quality. The large SSE values suggest that the clusters may not be very compact, and the low silhouette scores imply that there is considerable overlap between clusters. Further optimization or exploration of alternative clustering algorithms may be necessary to improve clustering performance on the MNIST dataset.

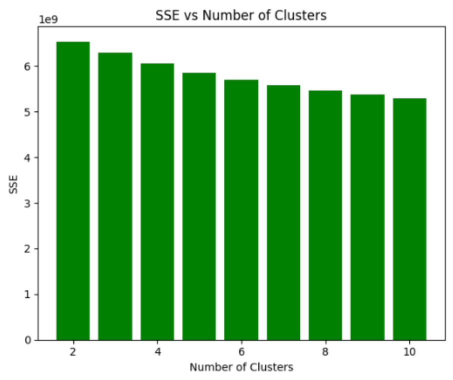
b) Scikit-learn hierarchical clustering:

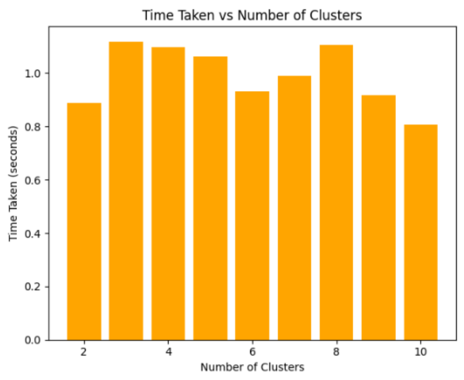
Values for different metrics for different number of clusters with MNIST dataset:

Number of clusters	SSE	Silhouette Score	Time
2	6539024329.4246	0.0825	0.8885
3	6295038107.0353	0.0597	1.1186
4	6061871391.2106	0.0619	1.0983
5	5856351685.6812	0.0674	1.0621
6	5698297441.4846	0.0639	0.9303
7	5575841600.1704	0.0705	0.9890
8	5464757993.0617	0.0728	1.1071
9	5375940134.7271	0.0585	0.9175
10	5298373568.3186	0.0613	0.8074

Plot for different metrics for different number of clusters with MNIST dataset:







Analysis:

SSE (Sum of Squared Errors): Similar to KMeans, SSE measures the sum of the squared distances between each data point and its assigned cluster centroid. In this analysis, we observe that SSE decreases as the number of clusters increases, which is expected. However, the absolute values of SSE are quite large, indicating that the clusters may not be very compact.

Silhouette Score: The silhouette score measures the quality of clustering based on the average distance between clusters and the average distance within clusters. In this analysis, the silhouette scores are relatively low, ranging from approximately 0.06 to 0.08. This suggests that the clusters are not very well-separated, and there may be overlap between them, similar to the results obtained from KMeans clustering.

Time: The time taken for hierarchical clustering is relatively consistent across different numbers of clusters, ranging from approximately 0.8 to 1.1 seconds. Unlike KMeans, the time taken for hierarchical clustering does not show a clear increasing trend with the number of clusters.

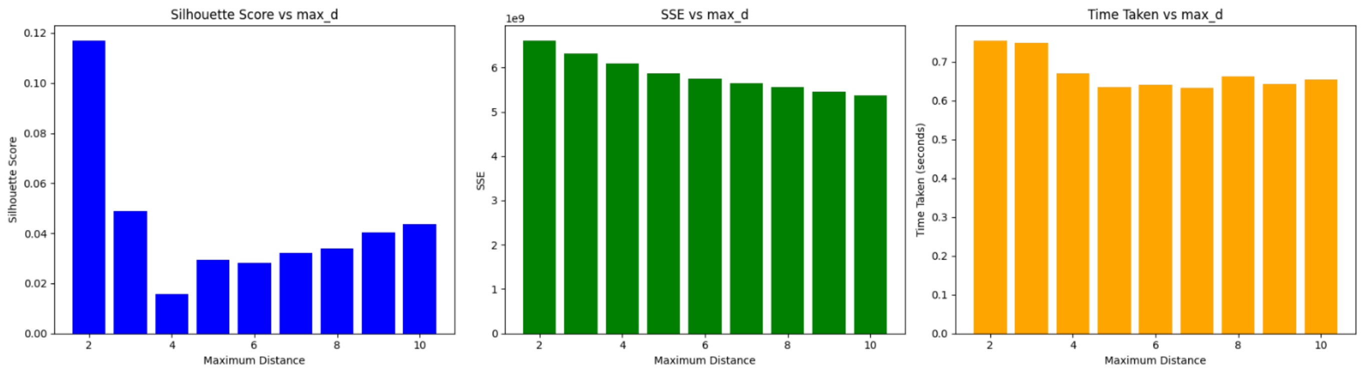
Overall, the hierarchical clustering algorithm using scikit-learn shows similar trends to KMeans in terms of SSE and silhouette score. However, hierarchical clustering tends to have consistent time complexity regardless of the number of clusters. Like KMeans, the results suggest that further optimization or exploration of alternative clustering algorithms may be necessary to improve clustering performance on the MNIST dataset.

c) Scipy hierarchical clustering:

Values for different metrics for different number of clusters with MNIST dataset:

Maximum Distance	SSE	Silhouette Score	Time
1	6612399121.2679	0.1171	0.7549
1.5	6321128533.7913	0.0490	0.7473
2	6090444821.7085	0.0156	0.6689
2.5	5871567280.4724	0.0294	0.6353
3	5745620741.0411	0.0282	0.6405
3.5	5644736628.6435	0.0322	0.6318
4	5549877677.3934	0.0340	0.6614
4.5	5457317578.7894	0.0405	0.6427
5	5372150285.9532	0.0436	0.6541

Plot for different metrics for different number of clusters with MNIST dataset:



Analysis:

SSE (Sum of Squared Errors): Similar to the results from scikit-learn hierarchical clustering and KMeans, SSE measures the sum of the squared distances between each data point and its assigned cluster centroid. In this analysis, we observe that SSE decreases as the maximum distance threshold increases, which is expected. However, the absolute values of SSE are still relatively large, indicating that the clusters may not be very compact.

Silhouette Score: The silhouette score measures the quality of clustering based on the average distance between clusters and the average distance within clusters. In this analysis, the silhouette scores are relatively low, ranging from approximately 0.015 to 0.117. While the silhouette scores vary with the maximum distance threshold, they remain consistently lower than ideal, suggesting that the clusters are not well-separated and may have overlap.

Time: The time taken for hierarchical clustering using scipy is relatively consistent across different maximum distance thresholds, ranging from approximately 0.63 to 0.75 seconds. Unlike scikit-learn hierarchical clustering, the time taken does not show a clear increasing or decreasing trend with the maximum distance threshold.

Overall, the hierarchical clustering algorithm using scipy shows similar trends to hierarchical clustering using scikit-learn and KMeans in terms of SSE and silhouette score. However, the results suggest that further optimization or exploration of alternative clustering algorithms may be necessary to improve clustering performance on the MNIST dataset.

3 Conclusion

The analysis of clustering algorithms on both the MNIST and Iris datasets provides valuable insights into their performance and suitability for different types of data.

KMeans:

For both the MNIST and Iris datasets, KMeans exhibits decreasing SSE values as the number of clusters increases, indicating better clustering performance with more clusters. However, the silhouette score remains relatively low across different numbers of clusters for both datasets, suggesting suboptimal cluster separation. The computational time increases with the number of clusters due to the iterative nature of the KMeans algorithm.

Scikit-learn Hierarchical Clustering:

Similar to KMeans, hierarchical clustering from scikit-learn demonstrates decreasing SSE values with an increasing number of clusters. The silhouette score remains relatively low across different numbers of clusters, indicating suboptimal cluster separation for both datasets. The computational time increases slightly with the number of clusters but remains consistent overall.

Scipy Hierarchical Clustering :

The hierarchical clustering algorithm from scipy shows varying results depending on the dataset and parameters chosen. For both datasets, SSE tends to decrease with increasing maximum distance threshold, but the silhouette score remains relatively low. The computational time varies depending on the dataset and parameters chosen but remains consistent overall.

Therefore, while KMeans and hierarchical clustering provide insights into the structure of both the MNIST and Iris datasets, none of the algorithms achieve optimal clustering performance. Further optimization or exploration of alternative clustering algorithms may be necessary to improve the quality of clustering.

Additionally, the choice of clustering algorithm should consider the specific characteristics of the dataset and the desired trade-offs between computational efficiency and clustering quality.

Reference:

[1] Fetch dataset from openml by name or dataset id. Scikit-learn user manual. https://scikit-learn.org/stable/modules/generated/sklearn.datasets.fetch_openml.html