Computational intelligence

Kiana Kermani , Ali Hamidzadeh

April 12, 2024

Abstract

1 Introduction

In this project, we intend to cluster images of people using different clustering algorithms.

Firstly, we import the libraries and set the location of the files. The models we used for training our data are VGG16 and VGG19 and Resnet50. These models are pre-trained and are used for feature extraction that we got from here

2 Phase 1

2.1 Feature extraction

This script is designed to extract features from images using pre-trained models like VGG16, VGG19, or ResNet50. The extracted features are then saved into a pickle file for further use.The extract-feature function takes an image file path and the pre-trained model as input and returns the extracted features as a flattened NumPy array.When we flatten these multi-dimensional arrays, we concatenate all the elements into a single one-dimensional array, effectively "flattening out" the structure. This allows us to represent the extracted features in a format that can be easily used as input for subsequent analysis.

2.2 Feature vector

Two dictionaries dataresnet and datavgg19 are initialized to store image filenames and their corresponding feature vectors extracted by ResNet50 and VGG19 models, respectively. the dictionaries containing filenames and feature vectors can be saved into separate pickle files. Then we print the feature vectors for each image in the dataset.

3 Phase 2

3.1 Dimension reduction

A dictionary named subject-color-map is defined to map subjects to colors for visualization purposes. In this case, all subjects are assigned the same color since specific subject information is not provided.

We use PCA (Principal Component Analysis) to reduce the dimensionality of the feature vectors to 2 dimensions and then Fit and transform the feature vectors, Then we do thesame for t-SNE (t-distributed Stochastic Neighbor Embedding). After that we Create pandas DataFrames and plot the datas using seaborn.scatterplot()

Due to the plot, we think t-SNE plotting is better than PCA. and the cluster number is around 17. cause we have 17 nodes of data. We can use K-Means via K = 17 and due to the result, change the K.

4 Phase 3

In this phase we implemented K-Means and DB-scan and agglomarative.

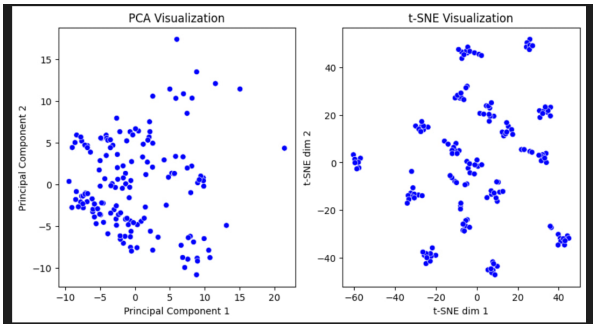


Figure 1: .

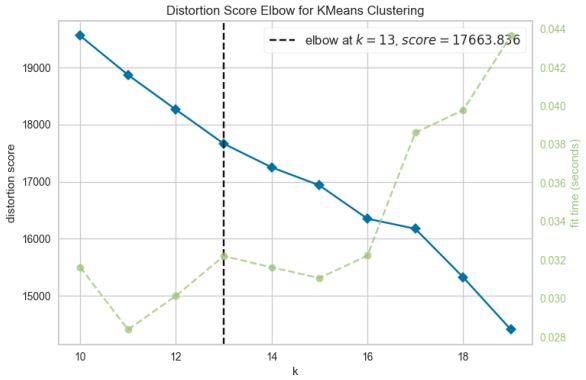


Figure 2: .

4.1 implementing k-means

this script provides a convenient way to visually determine the optimal number of clusters for KMeans clustering using the elbow method.

Using the number we got from the elbow method we perform KMeans clustering on the extracted features and organizes the images into clusters based on their predicted labels. It then saves the images into separate folders representing each cluster. Finally, it applies PCA and t-SNE for dimensionality reduction on the feature vectors and visualizes the clusters in a 2D space.

4.2 DBscan

After feature extracting and making the feature vector, we implement DBscan. The note we should add, firstly we implemented it with VGG16 because we didn't get the accuracy we wanted, we changed the model to VGG19 , still didn't get the accuracy we changed the model to RESNET50, finally we got

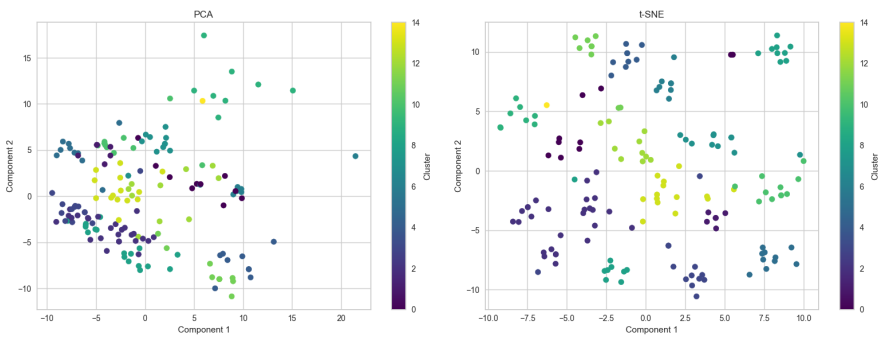


Figure 3: .

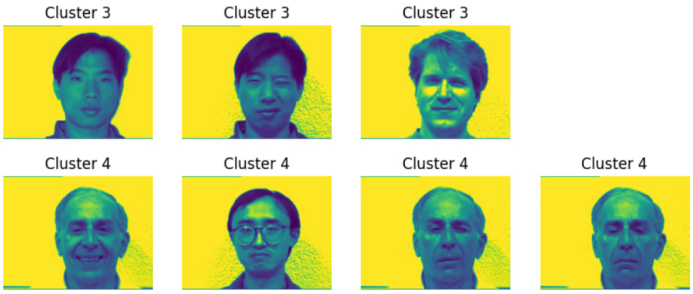


Figure 4: .

the accuracy that was acceptable. The accuracy that we did not get , first we changed the epsilon and min several times but we could not see any change, so we understood the problem was from models,

By computing the Silhouette Score once on different eps and once on diffent min samples, we got the results for Best Silhouette Score.Then we ploted the data.

After several times of running the DBscan and changing the eps and min samples and changing the model, The result we got the best was with model Resnet and eps=10.5 and min-samples=3.

After implementing the DBcan we Move images to cluster-specific folders and plot the data in two different ways, one is to plot the images in the folder, second is to plot based on the features.

4.3 agglomerate

We implement the agglomerative in different ways. Once we assume we do know the number of clusters and do the agglomerative clustering, the accuracy is very high.

If we do not know the number of clusters, we do perform hierarchical clustering on a set of features extracted from images and visualize the resulting dendrogram. Additionally, it provides guidance on determining the optimal number of clusters and performing clustering based on this determination. Then it would report the Optimal number of clusters and saves in the cluster output folder.Although it may not be the best way and may cause over-clustering.

To get better results we use the help of Silhouette Score to get an optimal score. It results in better clustering and better results.

Compute the silhouette score for different numbers of clusters ranging from 2 to 20. Identify the number of clusters with the highest silhouette score as the optimal number of clusters.

Perform clustering using the Agglomerative Clustering algorithm with the optimal number of clusters determined, Create Directories for Cluster Outputs, and Move Images to Cluster-Specific Folders.

5 Phase 4

5.1 Rand Index

The rand index takes two parameters,the true cluster labels and predicted cluster labels, and preprocesses the filenames to extract labels from them. It initializes true positives (tp), false positives (fp), true negatives (tn), and false negatives (fn) to 0. It iterates through each pair of data points and compares their true and predicted labels to determine tp, tn, fp, and fn. It calculates the Rand Index using the formula:

$$(tp + tn)/(tp + tn + fp + fn)$$

Finally, it returns the calculated Rand Index. Now that we have the implementation of the rand index, we perform it on the outputs we got and saved from the phase 3 algorithms.

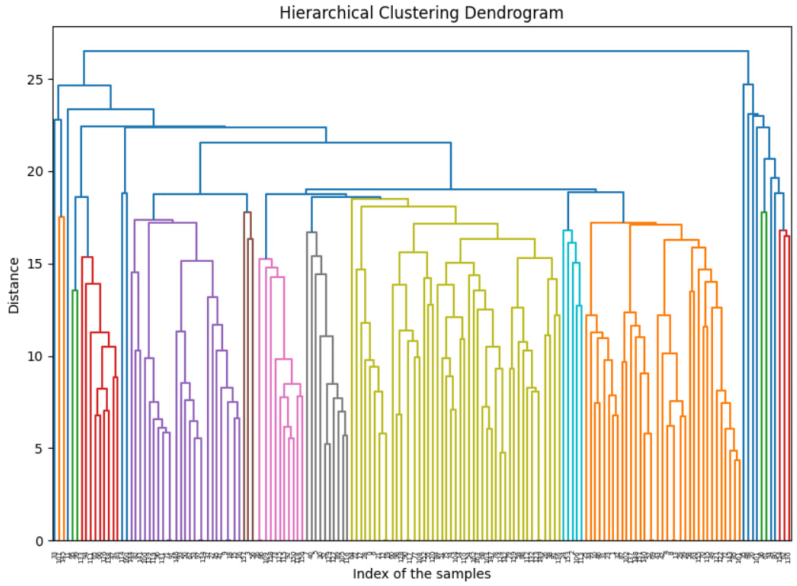
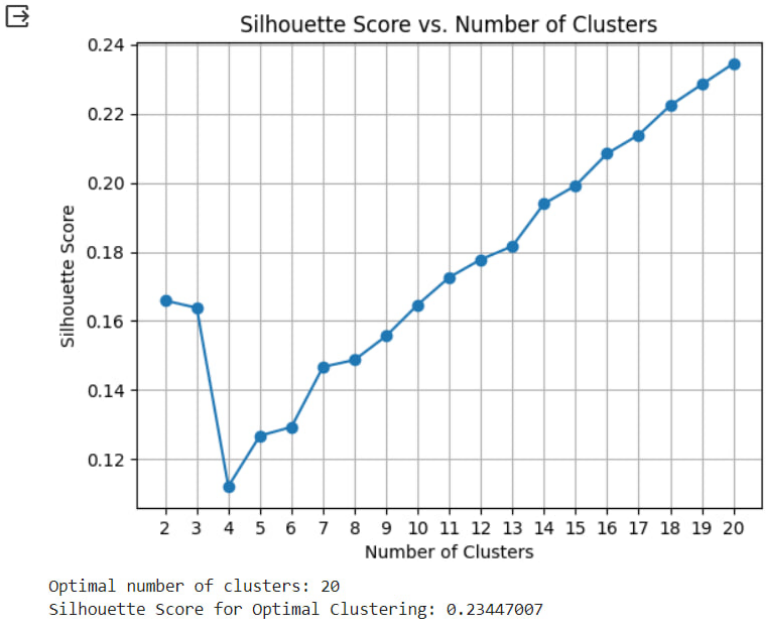


Figure 5: .




```
Optimal number of clusters: 20
Silhouette Score for Optimal Clustering: 0.23447007
```

Figure 6: .

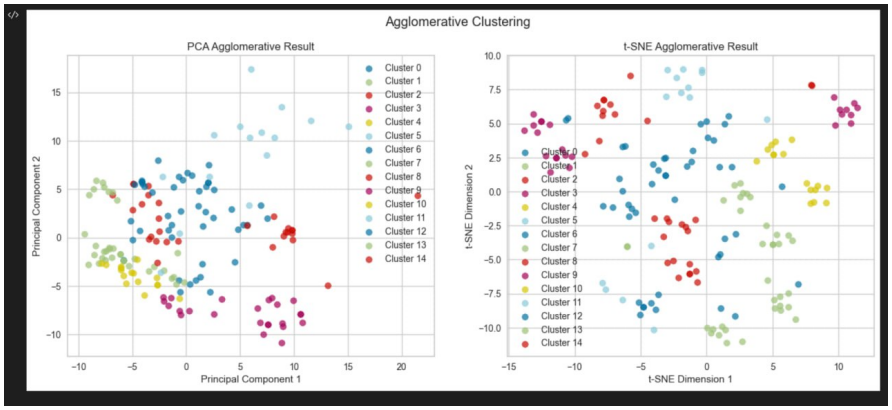


Figure 7: .

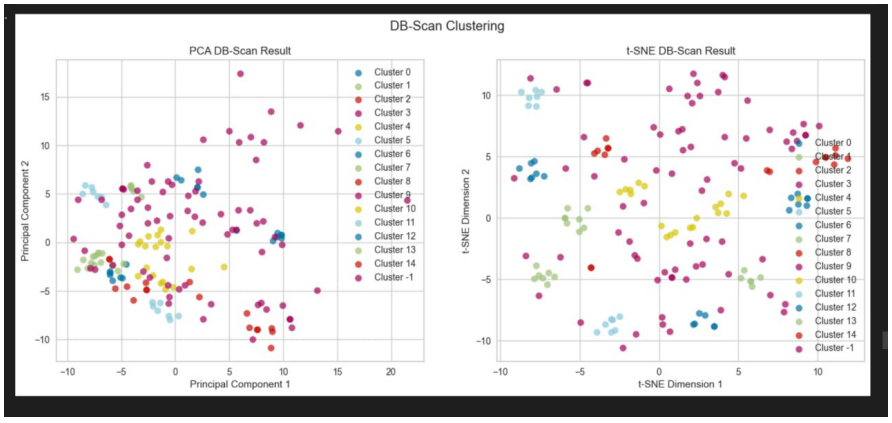


Figure 8: .

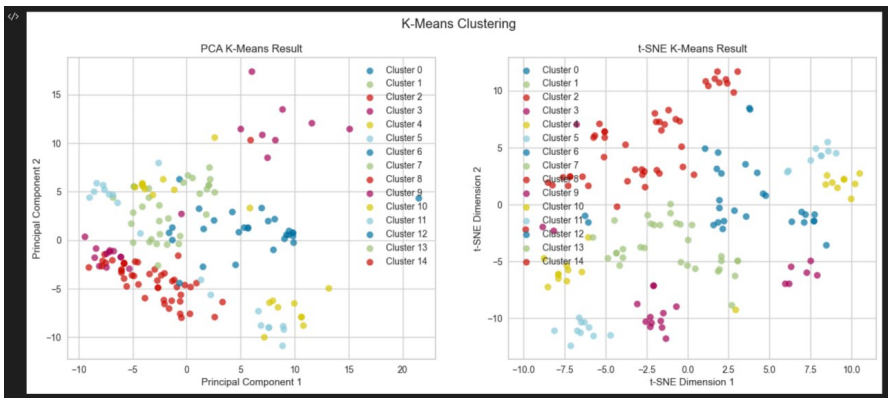


Figure 9: .

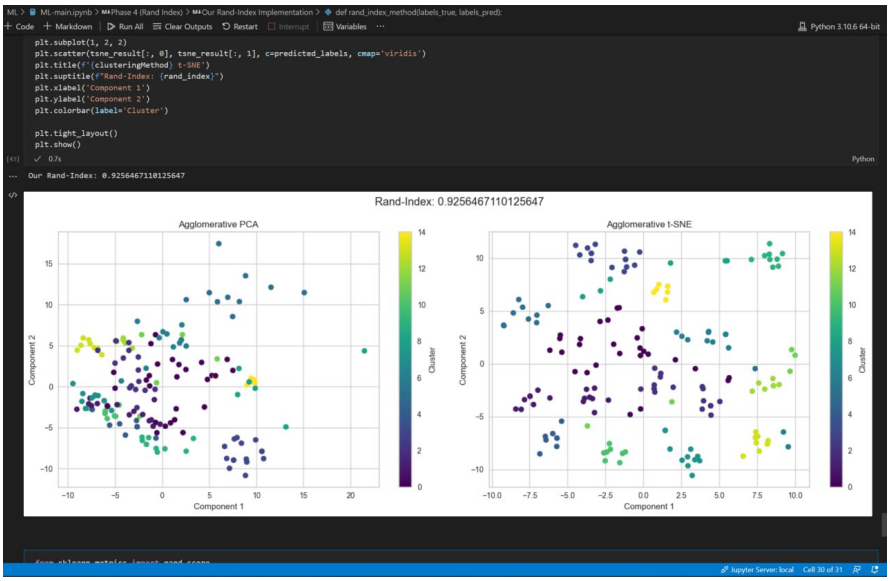


Figure 10: .

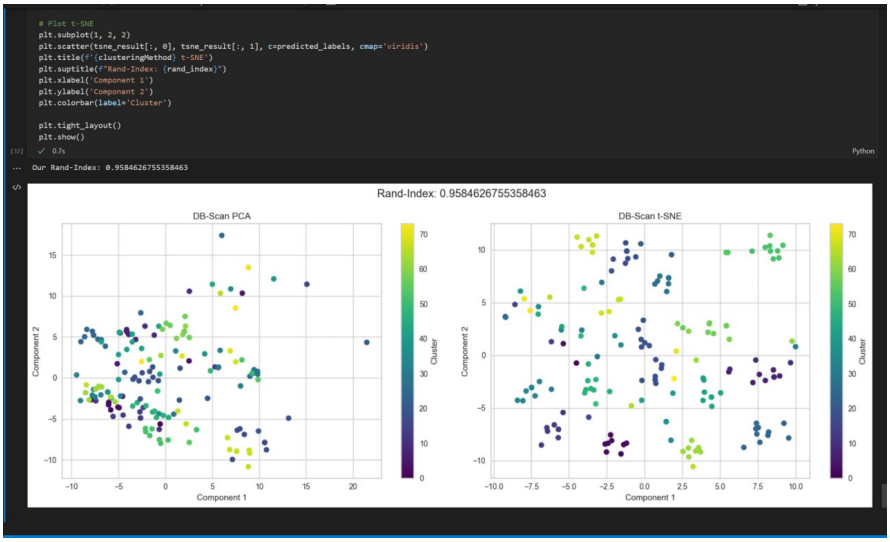


Figure 11: .


ML > ML-main.ipynb > Phase 4 (Rand Index) > Our Rand-Index Implementation > def rand_index_method(labels_true, labels_pred):
Code
Markdown
Run All
Clear Outputs
Restart
Interrupt
Variables
Python 3.10.6 64-bit
# Plot t-SNE
plt.subplot(1, 2, 2)
plt.scatter(tsne_result[:, 0], tsne_result[:, 1], c=predicted_labels, cmap='viridis')
plt.title(f'{clusteringMethod} t-SNE')
plt.suptitle(f"Rand-Index: {rand_index}")
plt.xlabel('Component 1')
plt.ylabel('Component 2')
plt.colorbar(label='Cluster')
plt.tight_layout()
plt.show()
[14]
0.7s
Python
Our Rand-Index: 0.9373244641537325
Rand-Index: 0.9373244641537325
K-Means PCA
K-Means t-SNE
Component 1
Component 2
Cluster

Figure 12: .