Mini Project 3

Vinay Sanga

1 Introduction

The recognition and classification of human activities have garnered significant attention in recent years due to its wide range of applications in various fields such as healthcare, sports, security, and human-computer interaction. The ability to automatically identify and classify human activities using wearable sensors offers great potential for developing context-aware systems that can assist individuals in their daily lives.

In this project, I aim to perform clustering of human activity recognition using sensor data collected from a group of volunteers wearing smartphones equipped with accelerometers and gyroscopes.

2 Data Preprocessing

I started by loading the dataset into pandas. As the data was already normalised, I was initially planning to use it as it is. But I standardised it because clustering algorithms perform better on the standardised dataset. The normalised dataset and the standardised dataset both behave similarly in the formation of clusters. I did not perform any further data preprocessing.

3 Modelling

3.1 Task 1: Using K-Means and DBSCAN to do clustering on the given dataset.

- **How do you choose the number of clusters in K-Means, is it the same number of clusters for DBSCAN?**
  To choose the number of cluster in K-Means, I used the following three tests:

  1. **Elbow method:**
     - The elbow method is a heuristic used to find the optimal number of clusters in a dataset. It works by plotting the within-cluster sum of squares (WCSS) against the number of clusters.
     - WCSS measures the compactness of the clusters. It calculates the sum of squared distances between each data point and its assigned centroid within a cluster. Lower values of WCSS indicate that the data points are closer to the centroids, implying better clustering.
     - When you plot WCSS against the number of clusters, the plot typically forms an elbow shape. The "elbow point" is where the rate of decrease in WCSS slows down significantly. Beyond this point, adding more clusters doesn't significantly reduce WCSS.
     - The optimal number of clusters is often chosen as the number of clusters corresponding to the elbow point.
     - In my case, the optimal number of cluster comes out to be **2** (fig. 1).

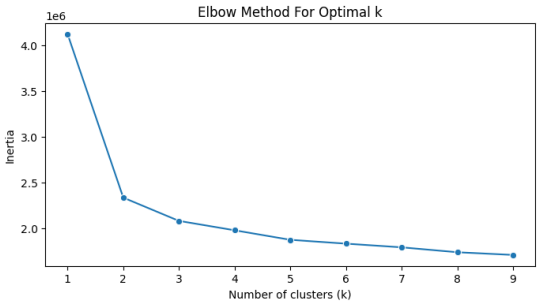

Figure 1: WCSS graph

  2. **Silhouette score:**
     - It is a measure of how similar an object is to its own cluster (cohesion) compared to other clusters (separation).
     - For each data point, the silhouette score ranges from -1 to 1. A high silhouette score indicates that the object is well matched to its own cluster and poorly matched to neighboring clusters.

- The silhouette score can be calculated for each sample and then averaged to get an overall score for the clustering.
- The optimal number of clusters is often chosen to maximize the silhouette score, as it indicates dense and well-separated clusters.

3. In my case, the highest silhouette score of about 0.4 comes at **k=2** (fig. 2).

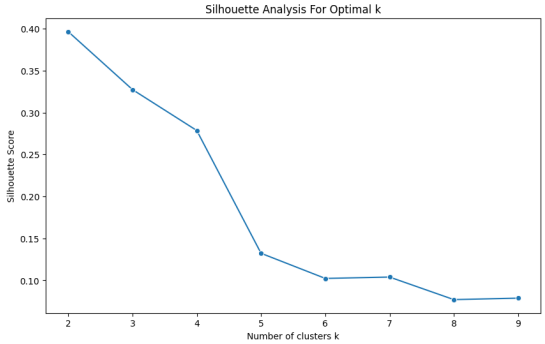

Figure 2: Silhouette score graph

- **The Davies-Bouldin index:** It is another metric used to evaluate clustering algorithms. It measures the average similarity between each cluster and its most similar cluster, where similarity is defined as the ratio of within-cluster distances to between-cluster distances.
- A lower Davies-Bouldin index indicates better clustering. It means that clusters are well-separated from each other and compact within themselves.
- The lowest index comes out to be 1 at **k=2** (fig. 3).

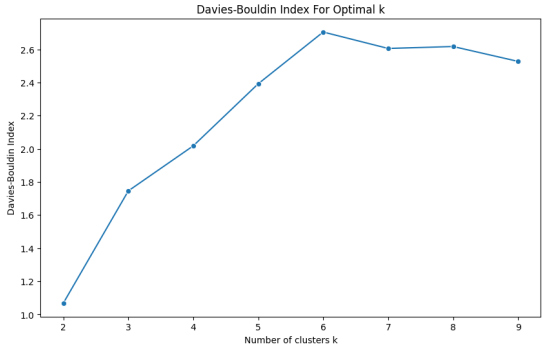

Figure 3: Davies-Bouldin index graph

Hence, I selected k=2 for the K-means clusters. It is same as the DBSCAN when using t-SNE for dimensionality reduction, else DBSCAN shows 1 cluster. There is no way to find the clusters for DBSCAN, as it has different hyperparameters.

4. **How do you find the optimal parameters' values?**

- For the K-means clustering algorithm, I have mentioned the steps for finding the parameters in the previous paragraphs.
- For DBSCAN, there are two hyperparameters - epsilon (eps) and minimum number of samples (min_samples):
  - **Epsilon:** Finding the epsilon value in DBSCAN involves selecting a distance threshold that determines the maximum distance between two points for them to be considered as part of the same neighborhood.
    1. **Plot the k-distance graph:**
       - Compute the distance from each point to its kth nearest neighbor, where k is typically chosen based on the number of dimensions in your dataset.
       - Sort the distances in ascending order.
       - Plot these distances on a graph.
    2. **Identify the knee or elbow point:**
       - Examine the plot of distances and look for a point where the rate of change in distances significantly changes. This point often resembles a knee or an elbow shape in the graph.
       - This point indicates a transition from lower distances (within the neighborhood) to higher distances (outside the neighborhood).
    3. **Choose epsilon:**
       - Select the distance corresponding to the knee or elbow point as your epsilon value.
       - I found the epsilon value to be **20** (fig. 4).

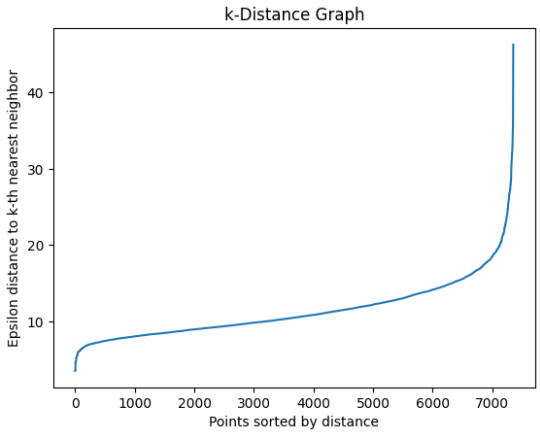

Figure 4: k-distance graph

* **Minimum no. of samples:** Finding the min_samples value is empirical process. But as a thumb-rule, it can be considered as total_number_of_features + 1. Hence, I chose its value as suggested.

3.2 Task 2: Using dimensionality reduction technique before using K-Means and DBSCAN

- **What is the dimensionality reduction technique that you choose, and why?**
  I used three dimensionality techniques - PCA, T-SNE and UMAP. The obtained results are discussed as follows:

  1. **Principal Component Analysis (PCA):** PCA is a dimensionality reduction technique commonly used for visualizing high-dimensional data in lower-dimensional space. It works by transforming the original data into a new coordinate system where the axes represent the principal components, which are orthogonal directions that capture the maximum variance in the data.
  2. **t-distributed Stochastic Neighbor Embedding (t-SNE):** t-SNE is a nonlinear dimensionality reduction technique commonly used for visualizing high-dimensional data in lower-dimensional space, particularly for preserving the local structure of the data. It works by modeling the pairwise similarities between data points in high-dimensional space and embedding them in a lower-dimensional space.
  3. **Uniform Manifold Approximation and Projection (UMAP):** UMAP is a nonlinear dimensionality reduction technique that is known for its scalability and ability to preserve both local and global structure in the data. It works by constructing a high-dimensional fuzzy topological representation of the data and then approximating it with a low-dimensional embedding.

  I used all the three because I wanted to compare the results obtained by all of them.

- **Does it have any effect on your code efficiency, both in terms of computational efficiency and clustering output?**
  Yes, there was an effect on both the terms. The run times are summarised in the table 1. We can see that without any transformation the run times are the highest for both the models. On the other hand, there is a significant performance gain while using any of the other data transformation techniques.
  There is also a difference in the clusters formed. They are discussed in the upcoming sections.

Model	Data Transformation	Total Time
KMeans	Standard	364 ms
	PCA	181 ms
	t-SNE	111 ms
	UMAP	6.85 ms
DBSCAN	Standard	4.59 s
	PCA	1.15 s
	t-SNE	140 ms
	UMAP	1.17 s

Table 1: Total time for different models with various data transformations

- **How do you compare the outcome of this model with the model where the dimensionality reduction technique was not applied to the dataset?**
  The comparison of different outputs is done in the upcoming sections. One comparison is done based on the algorithm run time, however we can not visualise them due to high dimensionality in unreduced dataset.

3.3 Task 3: Visualising the clusters

I applied the dimensionality reduction techniques as discussed in the previous section

to visualise the high dimensions in a two-dimensional space. Following are the results obtained:

- **KMeans Clustering with PCA: (fig. 5)**
  - Two distinct clusters are visible along the principal components.
  - PCA provides a clear linear separation between clusters.
  - There's a noticeable difference in the variance captured by each principal component.

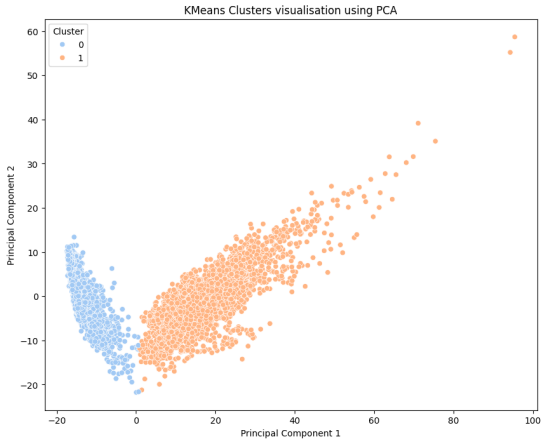

Figure 5: K-Means with PCA

- **KMeans Clustering with T-SNE: (fig. 6)**
  - Clusters are cleanly separated in space.
  - T-SNE provides a clear distinction between the clusters.

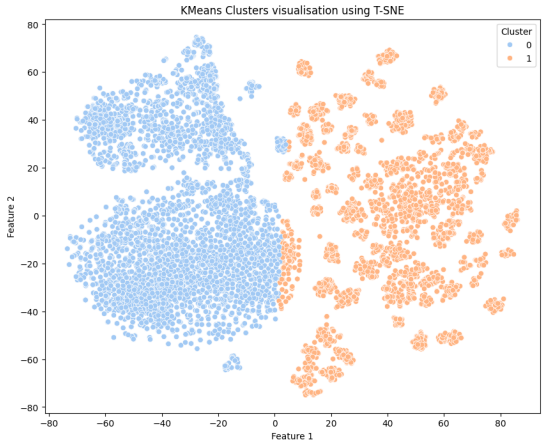

Figure 6: K-Means with T-SNE

- **KMeans Clustering with UMAP: (fig. 7)**
  - Data is partitioned into distinct clusters.
  - UMAP spreads out the clusters, improving cluster distinction.

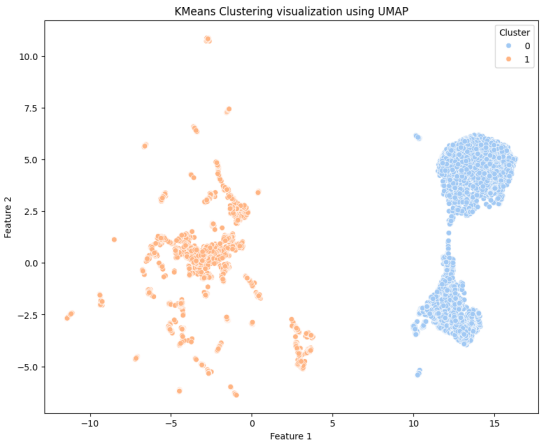

Figure 7: K-Means with UMAP

- **DBSCAN Clustering with PCA: (fig. 8)**
  - There is only 1 cluster (excl. noise). This did not work well with the given parameters.

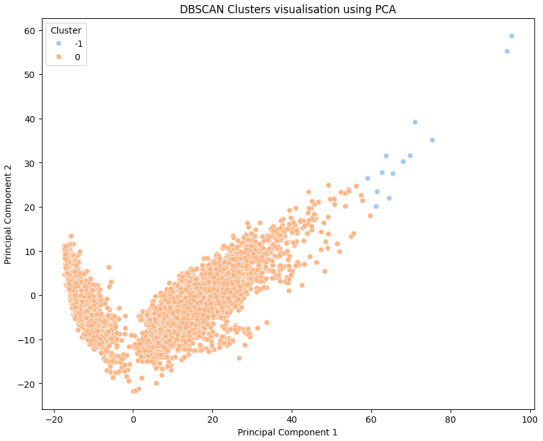

Figure 8: DBSCAN with PCA

- **DBSCAN Clustering with T-SNE: (fig. 9)**
  - Clusters are well-separated in the reduced dimensional space.
  - T-SNE effectively visualizes clusters and noise points distinctly.
- **DBSCAN Clustering with UMAP: (fig. 10)**
  - There is only 1 cluster (excl. noise). This did not work well with the given parameters.

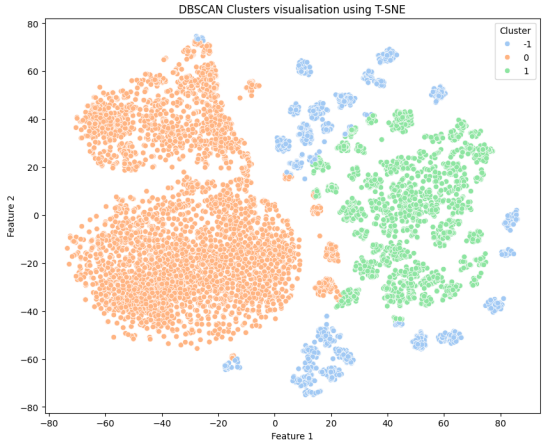

Figure 9: DBSCAN with T-SNE

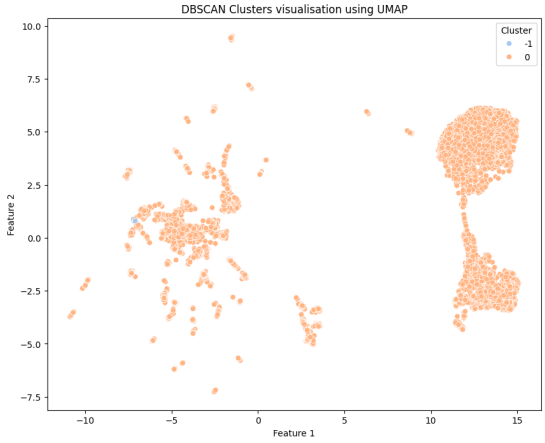

Figure 10: DBSCAN with UMAP

4 Conclusion

There is clear separation in two clusters, which signify the distinction between the WALKING, and NON-WALKING (SITTING, STANDING, LAYING) activities. Other than this, I could not make any significant interpretation of the formed clusters. There were a few scientific bottlenecks, such as choosing the right dimensionality reduction technique. I choose the three main techniques and then compared the results across all.