Mini Project 3: Human Activity Recognition Using Smartphones

Introduction

Understanding and classifying human movements is critical in many disciplines, including healthcare and sports analytics. With the introduction of wearable technology, such as cellphones equipped with accelerometers and gyroscopes, the capacity to record and analyze human movement patterns has become more accessible and insightful.

The dataset under examination is the result of thorough tests performed on a cohort of 30 volunteers ranging in age from 19 to 48 years. Each participant was tasked with doing six separate tasks while wearing a Samsung Galaxy S II smartphone around their waist. These exercises included daily movements such as walking, upstairs, downstairs, sitting, standing, and laying. Using the smartphone's inbuilt accelerometer and gyroscope, the studies recorded 3-axial linear acceleration and 3-axial angular velocity at a constant sampling rate of 50 Hz. To ensure data integrity and correctness, the experiments were videotaped, allowing for rigorous hand labeling of the collected data. The sensor signals (accelerometer and gyroscope) were pre-processed by applying noise filters and then sampled in fixed width sliding windows of 2.56 sec and 50% overlap (128 readings/window). The sensor acceleration signal, which has gravitational and body motion components, was separated using a Butterworth low-pass filter into body acceleration and gravity. The gravitational force is assumed to have only low-frequency components; therefore, a filter with a 0.3 Hz cutoff frequency was used. From each window, a vector of features was obtained by calculating variables from the time and frequency domains.

My main task is to use K-Means and DBSCAN to cluster on the given dataset and use a dimensionality reduction technique for K-Means and DBSCAN on the dataset. And analyze the result I get from 4 different models.

Data Overview

The dataset I used is from the link provided by the professor. I then converted the available sets of data into csv files and saved it as train.csv and test.csv file and loaded it in the drive.

There was in total (7352, 563) for training data and (2947, 563) for testing data.

```
! ls 'drive/MyDrive/EDISS/UCI_HAR/'
```

```
activity_labels.txt  features.txt                README.txt  test.csv  train.csv
features_info.txt    preprocessed_train_data.csv  test        train
```

Figure: Dataset in drive

```
Train Data:
   tBodyAcc-mean()-X  tBodyAcc-mean()-Y  tBodyAcc-mean()-Z  tBodyAcc-std()-X  \
0           0.288585          -0.020294          -0.132905         -0.995279
1           0.278419          -0.016411          -0.123520         -0.998245
2           0.279653          -0.019467          -0.113462         -0.995380
3           0.279174          -0.026201          -0.123283         -0.996091
4           0.276629          -0.016570          -0.115362         -0.998139

   tBodyAcc-std()-Y  tBodyAcc-std()-Z  tBodyAcc-mad()-X  tBodyAcc-mad()-Y  \
0         -0.983111         -0.913526         -0.995112         -0.983185
1         -0.975300         -0.960322         -0.998807         -0.974914
2         -0.967187         -0.978944         -0.996520         -0.963668
3         -0.983403         -0.990675         -0.997099         -0.982750
4         -0.980817         -0.990482         -0.998321         -0.979672

   tBodyAcc-mad()-Z  tBodyAcc-max()-X  ...  fBodyBodyGyroJerkMag-kurtosis()  \
0         -0.923527         -0.934724  ...                          -0.710304
1         -0.957686         -0.943068  ...                          -0.861499
2         -0.977469         -0.938692  ...                          -0.760104
3         -0.989302         -0.938692  ...                          -0.482845
4         -0.990441         -0.942469  ...                          -0.699205

   angle(tBodyAccMean,gravity)  angle(tBodyAccJerkMean),gravityMean)  \
0                    -0.112754                              0.030400
1                     0.053477                             -0.007435
2                    -0.118559                              0.177899
3                    -0.036788                             -0.012892
4                     0.123320                              0.122542
```

Figure: Training dataset

```
Test Data:
   tBodyAcc-mean()-X  tBodyAcc-mean()-Y  tBodyAcc-mean()-Z  tBodyAcc-std()-X  \
0           0.257178          -0.023285          -0.014654         -0.938404
1           0.286027          -0.013163          -0.119083         -0.975415
2           0.275485          -0.026050          -0.118152         -0.993819
3           0.270298          -0.032614          -0.117520         -0.994743
4           0.274833          -0.027848          -0.129527         -0.993852

   tBodyAcc-std()-Y  tBodyAcc-std()-Z  tBodyAcc-mad()-X  tBodyAcc-mad()-Y  \
0         -0.920091         -0.667683         -0.952501         -0.925249
1         -0.967458         -0.944958         -0.986799         -0.968401
2         -0.969926         -0.962748         -0.994403         -0.970735
3         -0.973268         -0.967091         -0.995274         -0.974471
4         -0.967445         -0.978295         -0.994111         -0.965953

   tBodyAcc-mad()-Z  tBodyAcc-max()-X  ...  fBodyBodyGyroJerkMag-kurtosis()  \
0         -0.674302         -0.894088  ...                          -0.705974
1         -0.945823         -0.894088  ...                          -0.594944
2         -0.963483         -0.939260  ...                          -0.640736
3         -0.968897         -0.938610  ...                          -0.736124
4         -0.977346         -0.938610  ...                          -0.846595

   angle(tBodyAccMean,gravity)  angle(tBodyAccJerkMean),gravityMean)  \
0                     0.006462                              0.162920
1                    -0.083495                              0.017500
2                    -0.034956                              0.202302
3                    -0.017067                              0.154438
4                    -0.002223                             -0.040046
```

Figure: Testing dataset

	Activity
0	STANDING
1	STANDING
2	STANDING
3	STANDING
4	STANDING

Figure: Activity

	tBodyAcc-mean()-X	tBodyAcc-mean()-Y	tBodyAcc-mean()-Z	\
count	7352.000000	7352.000000	7352.000000	
mean	0.274488	-0.017695	-0.109141	
std	0.070261	0.040811	0.056635	
min	-1.000000	-1.000000	-1.000000	
25%	0.262975	-0.024863	-0.120993	
50%	0.277193	-0.017219	-0.108676	
75%	0.288461	-0.010783	-0.097794	
max	1.000000	1.000000	1.000000	

	tBodyAcc-std()-X	tBodyAcc-std()-Y	tBodyAcc-std()-Z	tBodyAcc-mad()-X	\
count	7352.000000	7352.000000	7352.000000	7352.000000	
mean	-0.605438	-0.510938	-0.604754	-0.630512	
std	0.448734	0.502645	0.418687	0.424073	
min	-1.000000	-0.999873	-1.000000	-1.000000	
25%	-0.992754	-0.978129	-0.980233	-0.993591	
50%	-0.946196	-0.851897	-0.859365	-0.950709	
75%	-0.242813	-0.034231	-0.262415	-0.292680	
max	1.000000	0.916238	1.000000	1.000000	

Figure: Dataset Summary

Data Preprocessing

Data preprocessing is a critical phase in the data analysis pipeline, aimed at enhancing data quality, resolving inconsistencies, and preparing the data for subsequent analysis tasks. In this report, I have outlined the preprocessing steps undertaken on the ‘train_data’ and ‘test_data’ datasets, which are essential for ensuring integrity and reliability.

- **Handling Missing Values:**

  Missing values in a dataset can pose significant challenges to analysis and modeling tasks. Therefore, the first step in data preprocessing involves identifying and handling missing values appropriately.

  These findings indicate that both the training and test datasets are complete, with no missing data points, thereby eliminating the need for imputation or removal of missing values.

```
tBodyAcc-mean()-X           0
fBodyAccJerk-kurtosis()-Y   0
fBodyAccJerk-meanFreq()-X   0
fBodyAccJerk-meanFreq()-Y   0
fBodyAccJerk-meanFreq()-Z   0
                           ..
tBodyGyroJerk-iqr()-Z       0
tBodyGyroJerk-iqr()-Y       0
tBodyGyroJerk-iqr()-X       0
tBodyGyroJerk-energy()-Z    0
Activity                    0
Length: 563, dtype: int64
```

Figure: Checking Null Data

- **Handling Duplicate Data:**

  Duplicate data instances can skew statistical analyses and model performance. Therefore, identifying and removing duplicate entries is crucial to ensure the integrity of the dataset.

```
# Checking duplicate data
train_data.duplicated().sum()
```

```
0
```

```
# Checking duplicate data
test_data.duplicated().sum()
```

```
0
```

Figure: Checking Duplicate Data

The absence of duplicate entries in both the training and test datasets signifies the robustness of the data collection process and mitigates the risk of bias.

Exploratory Data Analysis (EDA)

Exploratory Data Analysis (EDA) is a crucial initial step in the data analysis process, aimed at gaining insights, detecting patterns, and understanding the structure of the dataset. In this report, I present the findings from EDA conducted on the 'train_data' dataset, focusing on various aspects such as data distribution, and feature characteristics.

Visualization of Activities Distribution

Upon examining the distribution of activities within the train_data dataset, I found that it comprises six distinct activities, each corresponding to various movements recorded by the smartphone's embedded sensors. The breakdown of activity counts is as follows:

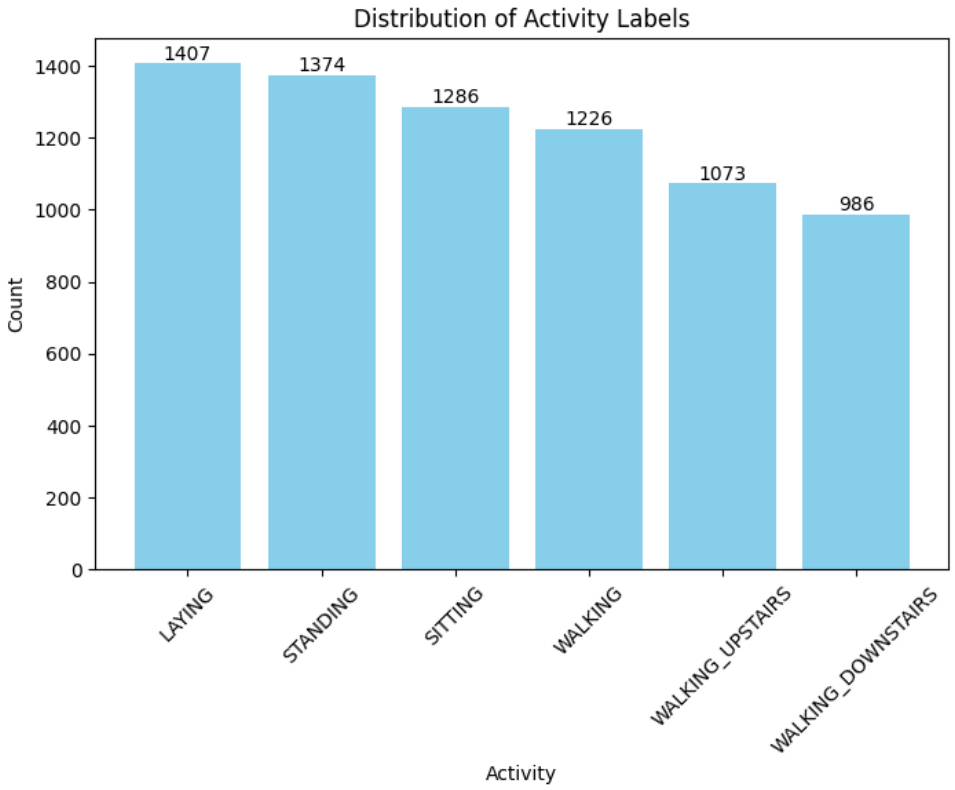

Figure: Distribution of Activity Labels

This distribution provides insights into the frequency of each activity recorded in the dataset, indicating the prevalence of certain activities over others.

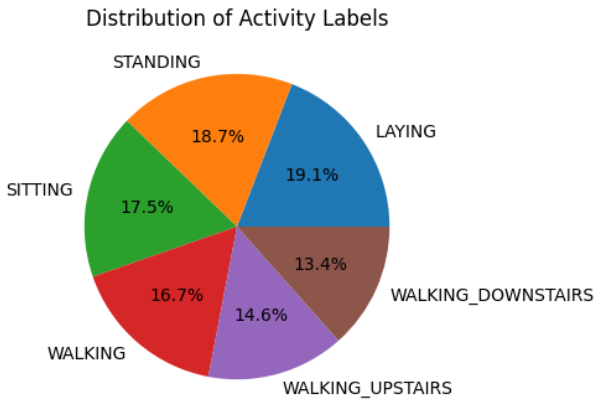

Figure: Distribution of Activity Labels by Percentage

The analysis of activity distribution provides valuable insights into the composition of the dataset and the prevalence of different activities. From the visualizations, it is evident that the dataset is relatively balanced, with no significant imbalance observed among the activity labels. This balanced distribution is advantageous for training robust machine learning models that can effectively generalize across all activities.

Visualization of Sensor Data

In this section, I tried to visualize the sensor data captured from the accelerometer and gyroscope sensors embedded in the smartphone worn by individuals during various activities. The visualization provides insights into the patterns and fluctuations of linear acceleration (accelerometer data) and angular velocity (gyroscope data) over time.

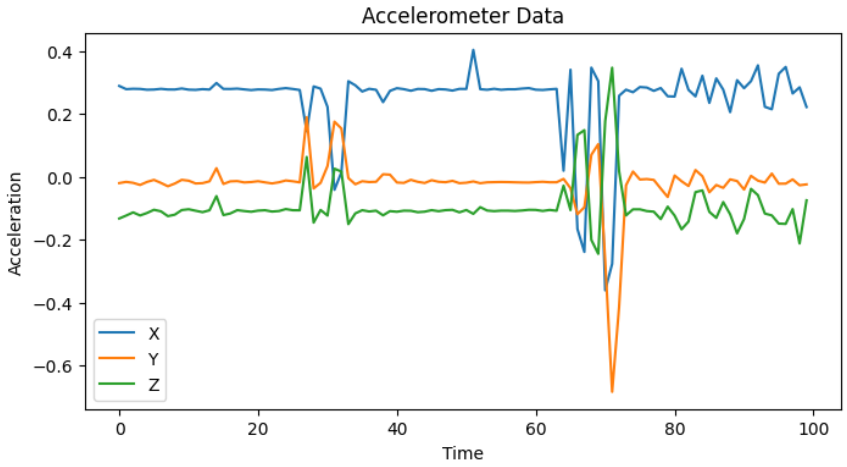

Figure: Visualization of Accelerometer Data

The accelerometer data is visualized using a line plot, with time on the x-axis and acceleration on the y-axis. Each axis of acceleration (X, Y, Z) is represented by a different color in the plot. The plot illustrates the variation of linear acceleration along each axis over the sampled time-period.

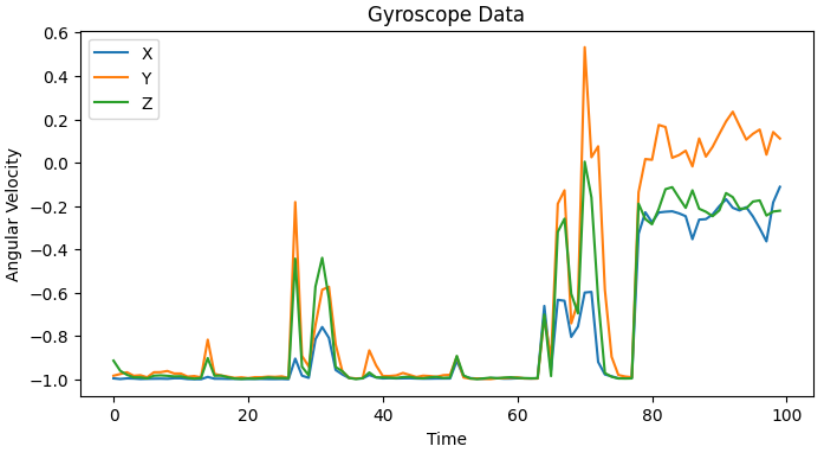

Figure: Visualization of Gyroscope Data

Similarly, the gyroscope data is visualized using a line plot, with time on the x-axis and angular velocity on the y-axis. Each axis of angular velocity (X, Y, Z) is distinguished by a different color in the plot. The plot depicts the changes in angular velocity along each axis throughout the sampled time interval.

Peaks or fluctuations in the accelerometer data may correspond to different activities such as walking, running, or stationary positions (sitting or standing). Changes in angular velocity captured by the gyroscope can indicate rotational movements or changes in orientation during activities such as turning, climbing stairs, or descending stairs.

By visually inspecting the accelerometer and gyroscope data, we can discern meaningful patterns and trends, laying the groundwork for further analysis and interpretation in the domain of human activity recognition.

Visualization of Distribution of Usage of Accelerometer and Gyroscope Sensor Signal

The analysis aims to explore the distribution of usage of accelerometer and gyroscope sensor signals within the dataset. These sensors play a crucial role in capturing human movement patterns, providing valuable insights into activities such as walking, running, and other physical motions.

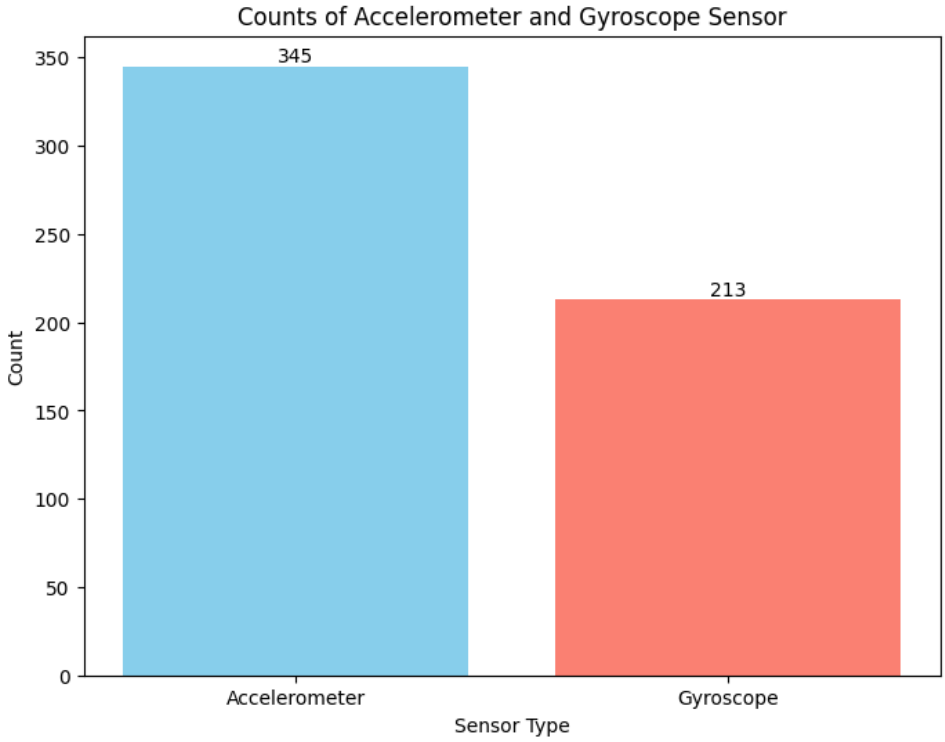

Figure: Distribution of Usage of Accelerometer and Gyroscope Sensor Signal

To assess the distribution of accelerometer and gyroscope sensor signals, the dataset was examined to count the occurrences of columns containing accelerometer gyroscope in their column names.

The analysis reveals that the dataset contains a higher number of accelerometer signals compared to gyroscope signals. Accelerometer sensors are utilized more frequently with 345 counts, indicating their importance in capturing a wide range of human activities such as walking, standing, sitting, and lying down. Gyroscope sensors with 213 counts, although fewer in count, are still significant in capturing angular velocity data, which is valuable for activities involving rotational movements and changes in orientation.

Modelling

The ‘train_data’ and ‘test_data’ datasets are concatenated into a single dataframe data using the pd.concat() function. Concatenating the datasets combines the training and testing data into a unified dataset for further preprocessing and modeling. Once the datasets are concatenated, the next step involves splitting the data into features (X) and the target variable (y).

The data is then standardized using the StandardScaler from the scikit-learn library. Standardization is a preprocessing technique commonly applied to numerical features in machine learning tasks. This preprocessing step enhances the robustness and interpretability of the models, ultimately leading to more accurate predictions and better insights into the underlying data patterns.

I used two models for this project as asked. K-means and DBSCAN clustering model, first without reducing dimensionality and after that using PCA dimensionality reduction.

K-Means

In the provided code snippet, the number of clusters for K-Means clustering is determined using the elbow method, and the results are visualized.

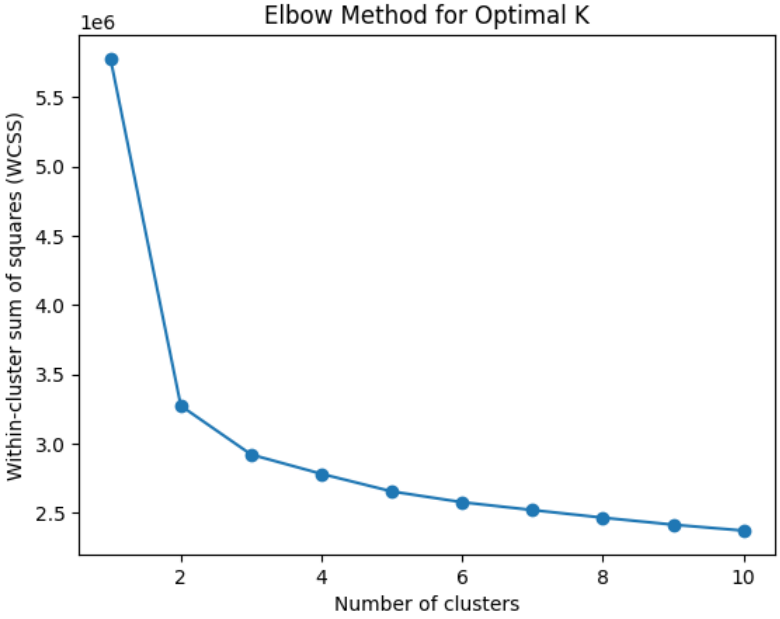

Figure: Elbow method for determining no of clusters

The inertia values are plotted against the number of clusters (K) to visualize the elbow curve. The elbow curve helps identify the optimal number of clusters by observing the point where the rate of decrease in inertia slows down, forming an "elbow" in the curve. Based on the elbow method analysis, the optimal number of clusters (optimal K) is selected. In this example, the optimal K is chosen as 2.

K-Means clustering is performed then with the optimal number of clusters determined using the elbow method. The K-Means clustering results are visualized using a scatter plot. Each data point is represented by its features tBodyAccJerkMag-entropy(), and tGravityAcc-energy()-X and the points are colored according to their assigned cluster labels (kmeans_labels).

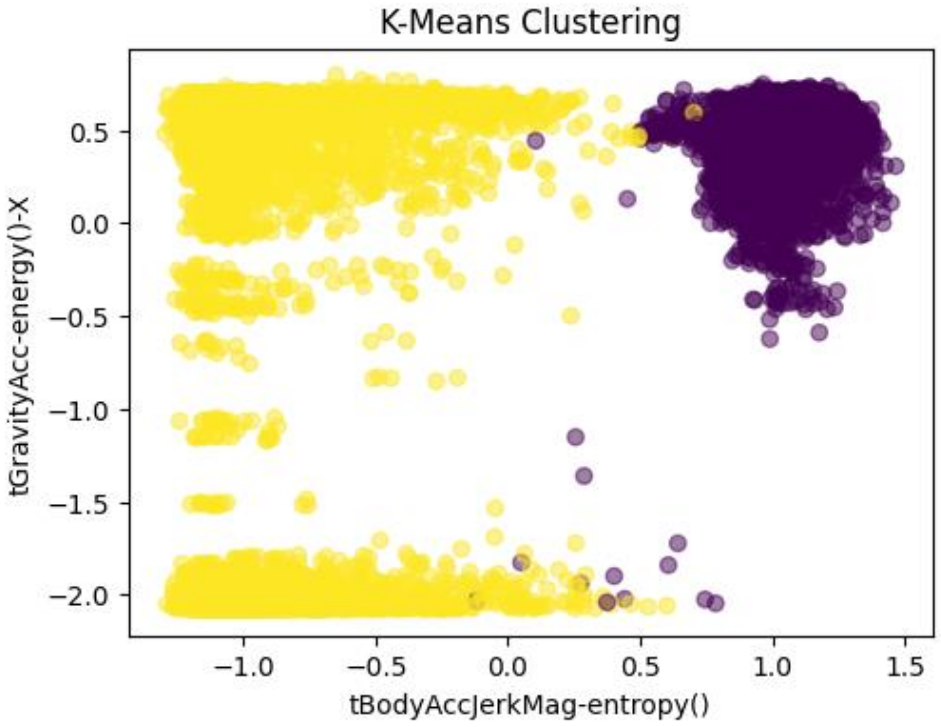

Figure: K-Means Clustering

This visualization provides insights into the clustering structure and separation of data points in the feature space. Based on underlying patterns and groupings within the data, which is essential for later analysis and interpretation in human activity recognition. Two clusters are identified for the specified features, where one cluster falls in between ranges 0.5 to 1.5 and –2.0 to 0.5 respectively.

DBSCAN

DBSCAN (Density-Based Spatial Clustering of Applications with Noise) is used to perform clustering on the same dataset. The variable minpts is set to the number of features in the dataset. In this case, it is 561, which is the total number of dimensions or features in the dataset. NearestNeighbors is used to calculate the distances between each data point and its minpts nearest neighbors.

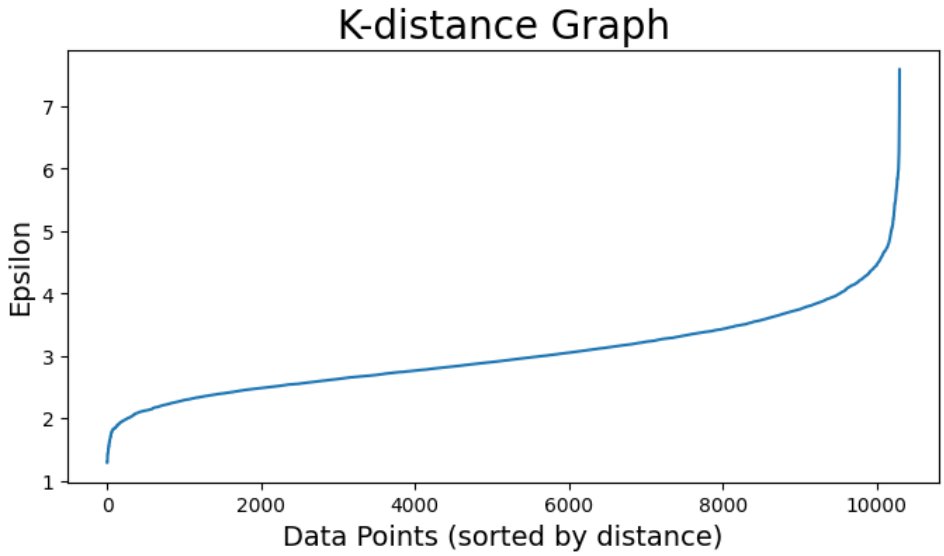

Figure: K-distance graph

A K-distance graph is plotted to visualize the distances to the minpts nearest neighbors for each data point. The value of epsilon can be determined from the point where the plot shows a significant increase, often referred to as the "knee" in the graph.

DBSCAN parameters, specifically epsilon, to find the optimal value that maximizes clustering performance. The number of clusters and silhouette scores are computed and visualized across different epsilon values.

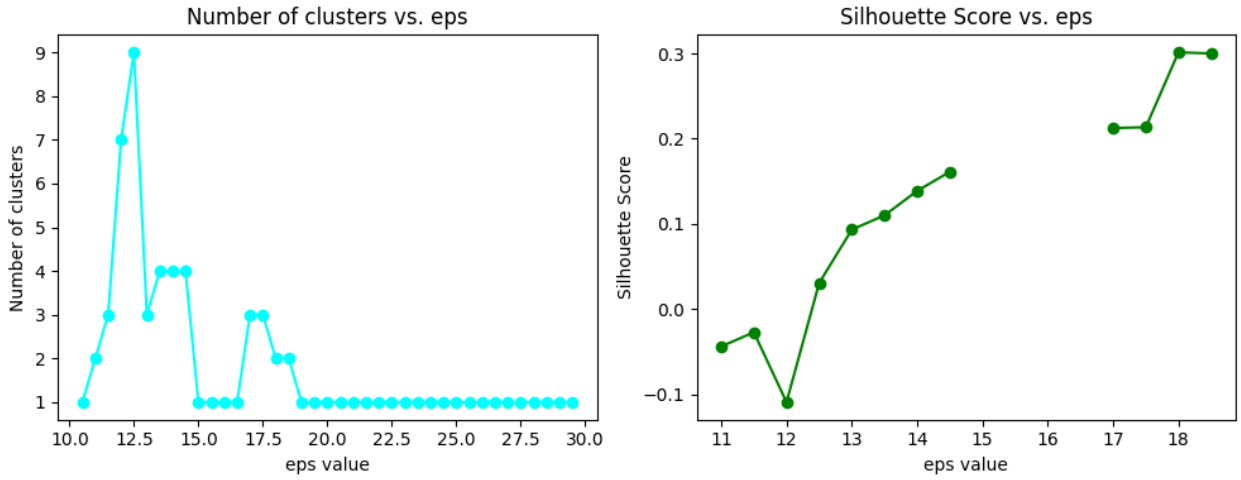

Figure: Parameter defining

The evaluation helps in selecting the epsilon value that yields the desired number of clusters with high silhouette scores, indicating dense and well-separated clusters. This process guides parameter

selection for DBSCAN, ultimately leading to more effective clustering results in human activity recognition. The identified EPS value is 17 as per above analysis.

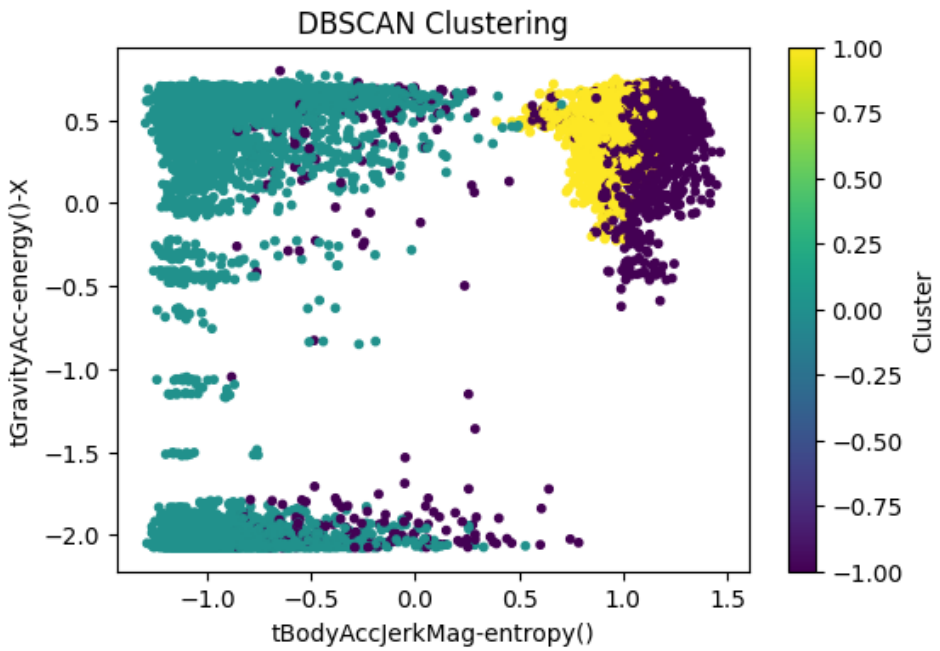

Figure: DBSCAN Clustering

DBSCAN clustering is applied with optimized parameters to identify clusters in the dataset. The chosen epsilon value of 17 and minPts value of 561 are used to define the neighborhood density criterion for forming clusters.

The number of clusters identified by DBSCAN, and the resulting clustering structure are visualized in the scatter plot. Each cluster is represented by a distinct color, allowing for easy interpretation of the clustering results. This visualization provides insights into the grouping of data points based on their feature characteristics, which can aid in understanding the underlying patterns in human activity recognition. The identified clusters are 3 and ranges in between –2.0 to 0.5, 0.5 to 1, and 1 to 1.5 respectively.

Computing PCA and Determining Optimal Number of Components

Principal Component Analysis (PCA) is applied to the dataset to reduce dimensionality while retaining as much variance as possible. A Scree plot is generated to visualize the explained variance ratio for each principal component. The individual explained variance (percentage of variance explained by each component) is plotted as bars, while the cumulative explained variance is shown as a step plot.

The elbow point in the Scree plot is identified, indicating the optimal number of principal components to retain. The second derivative of the cumulative explained variance is calculated to find the point where the change in the change of variance explained is greatest, which typically

corresponds to the elbow point. The optimal number of PCA components is then printed for further analysis and model training.

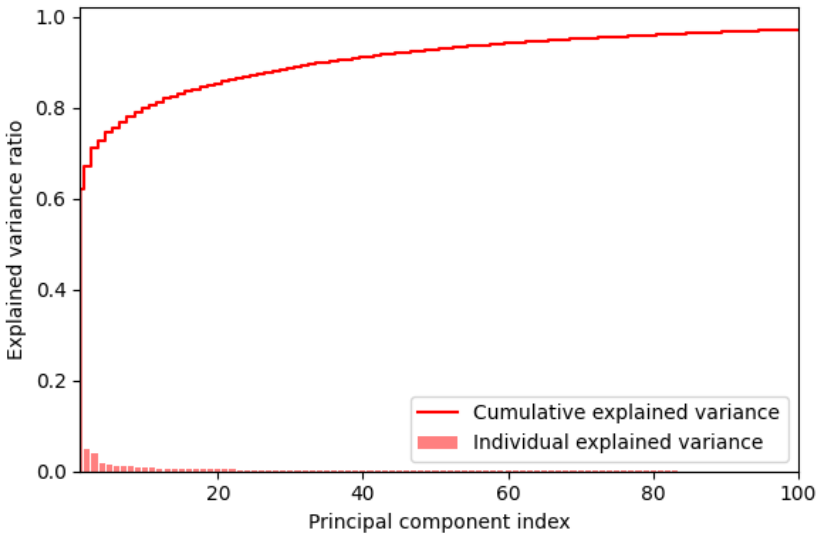

Figure: Identification of optimal number of PCA components

PCA is applied to reduce the dimensionality of the dataset to two dimensions, allowing for visualization of the clustering results in a more interpretable space. The scatter plot illustrates the distribution of data points in the reduced dimensional space, with each point colored according to its assigned cluster label.

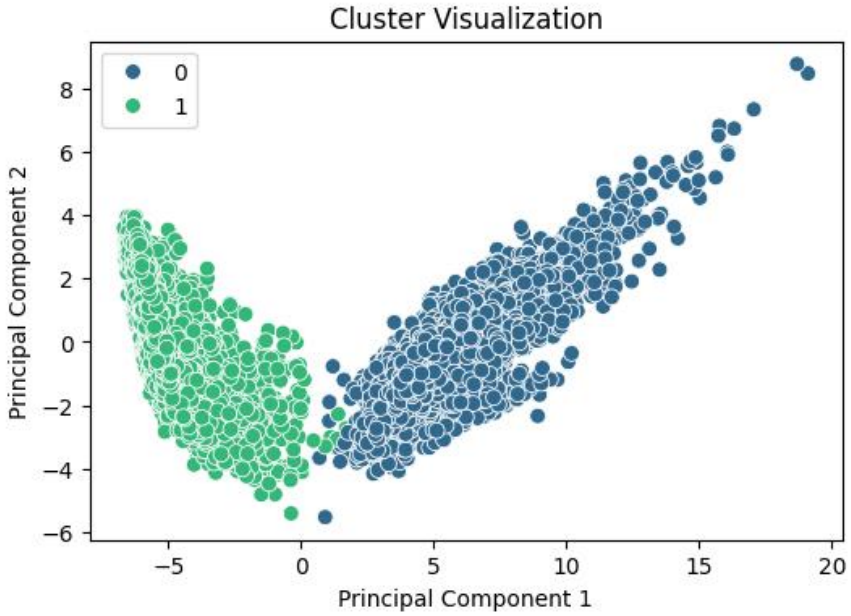

Figure: PCA Cluster Visualization

This visualization provides insights into the grouping of data points and the separation of clusters based on their principal component representations. It aids in understanding the clustering structure

and the effectiveness of the clustering algorithm in segregating different groups within the dataset. According to PCA, clusters are divided into two and more specifically.

K-Means with PCA for Dimensionality Reduction

In this section, K-Means clustering is performed on the dataset after reducing its dimensionality using Principal Component Analysis (PCA). This time specifying the optimal number of clusters determined earlier. By transforming the data into a lower-dimensional space, K-Means operates on a simplified feature space while preserving the essential structure of the data.

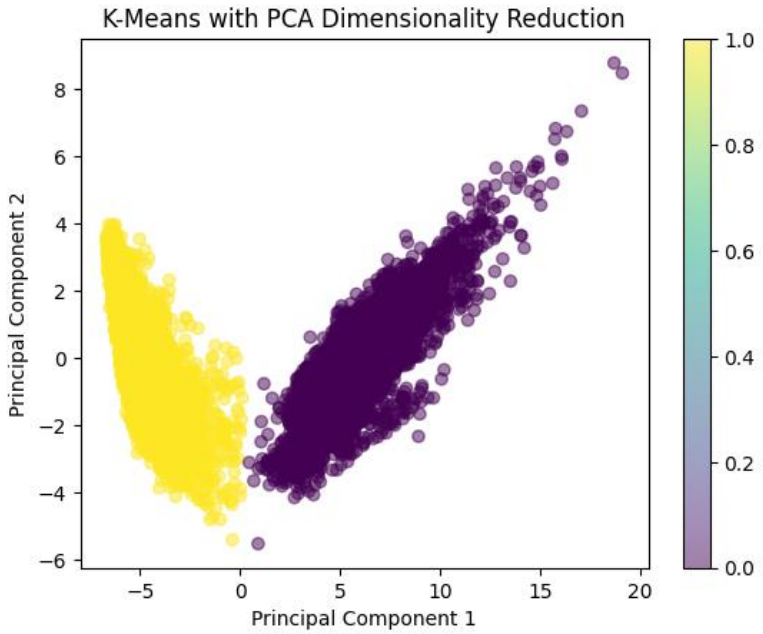

Figure: K-Means with PCA Dimensionality Reduction

The scatter plot illustrates the clustering results in the reduced two-dimensional space. It shows how K-Means assigns data points to clusters based on their principal component representations. This visualization helps in understanding the clustering structure and the distribution of clusters in the reduced dimensional space.

DBSCAN with PCA for Dimensionality Reduction

Here, DBSCAN clustering is performed on the dataset after reducing its dimensionality using Principal Component Analysis (PCA). The number of clusters formed by DBSCAN, and the silhouette score are calculated again and stored for evaluation. The number of clusters formed by DBSCAN, and the silhouette score are plotted against the ε values to analyze the clustering performance under different neighborhood distances.

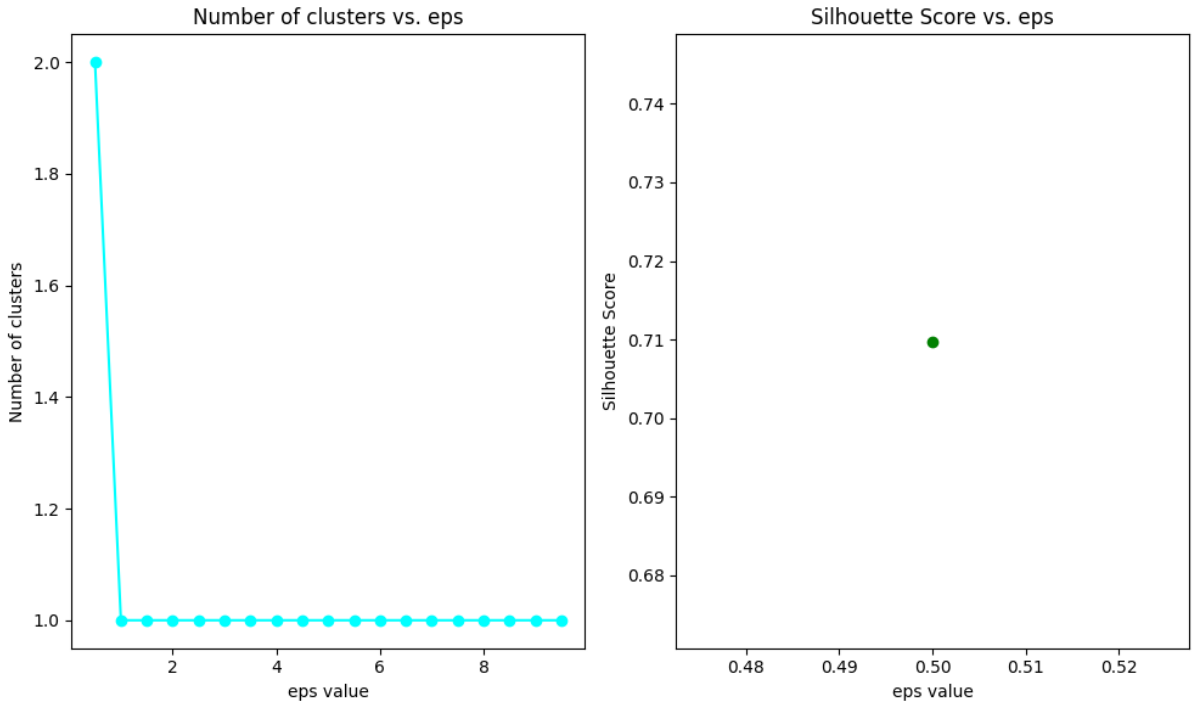

Figure: Determining EPS

These plots help in determining the optimal ε value for clustering the data after dimensionality reduction using PCA. According to the figure above, the EPS is determined to be 1.

DBSCAN is applied with the optimal ε value (1) and a minimum number of samples set to 561. The resulting cluster labels are visualized in the reduced two-dimensional space obtained from PCA.

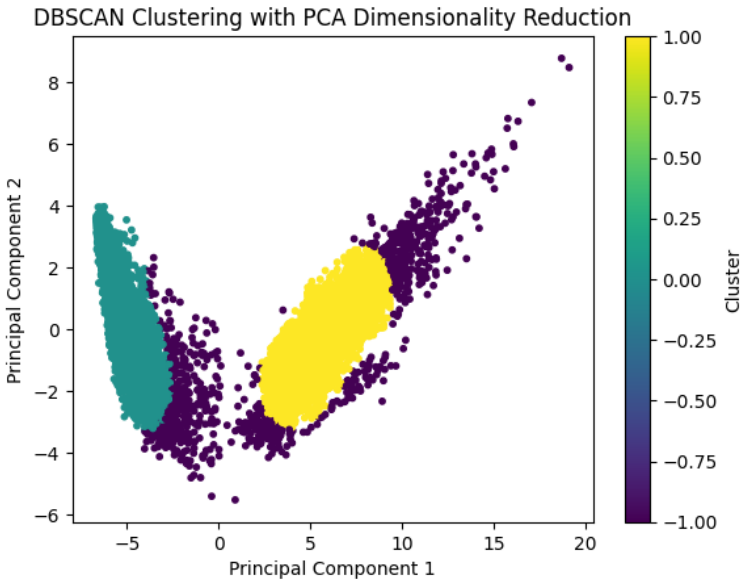

Figure: DBSCAN Clustering with PCA Dimensionality Reduction

DBSCAN clustering is applied to the dataset after reducing its dimensionality using PCA. By evaluating the number of clusters and silhouette scores for different ε values, the optimal neighborhood distance for clustering is determined. The scatter plot illustrates the clustering results in the reduced two-dimensional space, showing the distribution of clusters based on their principal component representations. This visualization helps in understanding the clustering structure and the separation of clusters after dimensionality reduction.

Comparision K-Means Models with PCA and without PCA

In this section, I tried to compare the execution times and clustering results of K-Means with and without dimensionality reduction using PCA. K-Means clustering is applied to the reduced-dimensional space (principalDf) obtained from PCA.

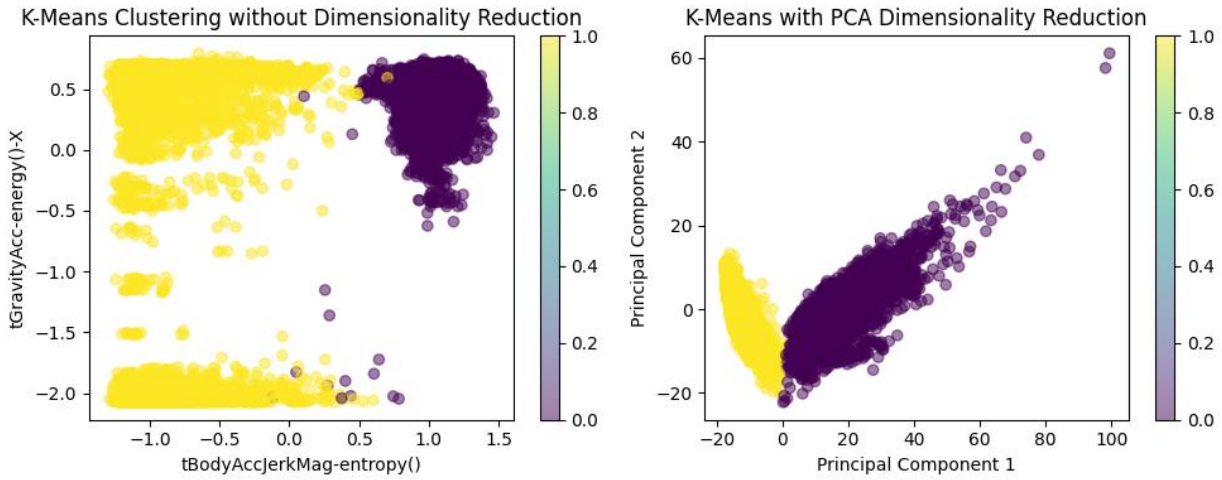

Figure: Comparision between K-Means with PCA and without PCA

Two subplots are created to visualize the clustering results of K-Means with and without dimensionality reduction side by side. The left subplot displays the clustering results in the original feature space. The right subplot displays the clustering results in the reduced two-dimensional space obtained from PCA. Clustering with PCA is clearer and reduced dimensional space.

The execution times for K-Means clustering with and without dimensionality reduction are printed for comparison.

```
K-Means without Dimensionality Reduction:
Execution Time: 0.3461637496948242

K-Means with PCA Dimensionality Reduction:
Execution Time: 0.06310200691223145
```

Figure: Execution Time

It compares the execution times and clustering results of K-Means with and without dimensionality reduction using PCA. It illustrates how PCA can reduce computation time while preserving clustering quality by transforming the data into a lower-dimensional space.

Comparision DBSCAN Models with PCA and without PCA

I compare the execution times of DBSCAN clustering with and without dimensionality reduction using PCA. DBSCAN clustering is applied to the reduced-dimensional space (principalDf) obtained from PCA. Again, the optimal parameters with dimensionality reduction are calculated for DBSCAN model.

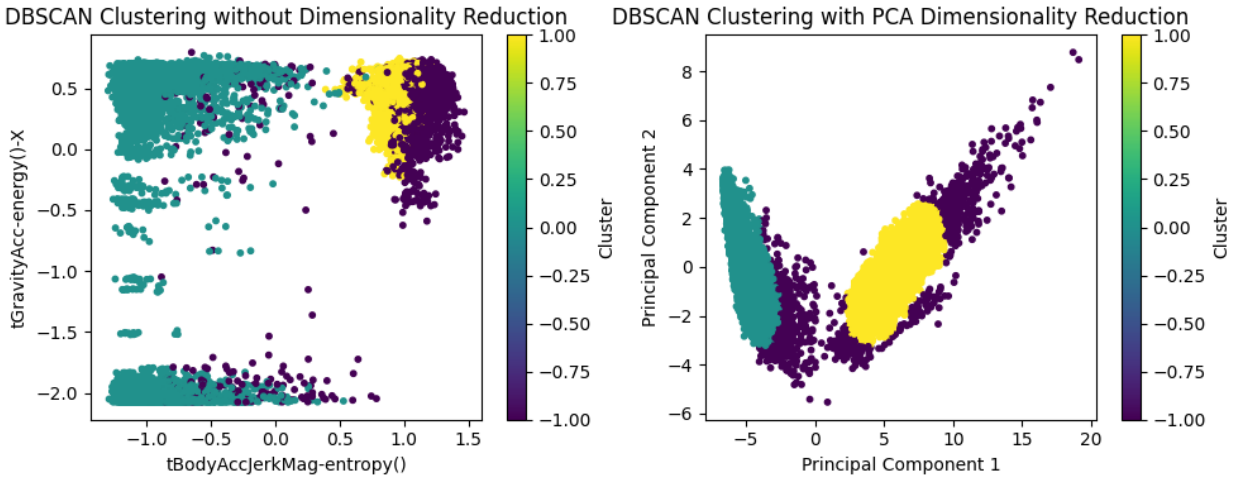

Figure: DBSCAN Clustering with and without PCA

Two subplots are created to visualize the clustering results of DBSCAN with and without dimensionality reduction side by side. The left subplot displays the clustering results in the original feature space. The right subplot displays the clustering results in the reduced two-dimensional space obtained from PCA.

```
DBSCAN without Dimensionality Reduction:
Execution Time: 3.9310734272003174

DBSCAN with PCA Dimensionality Reduction:
Execution Time: 0.25217294692993164
```

Figure: Execution Time

The measured execution times highlight the efficiency gains achieved by employing PCA for dimensionality reduction in the context of DBSCAN clustering. The significant reduction in execution time observed when applying PCA for dimensionality reduction highlights the efficiency gains achieved. This improvement in computational efficiency makes PCA a valuable preprocessing technique for clustering large datasets or applications requiring real-time processing.

Conclusion

In conclusion, I applied K-Means and DBSCAN clustering algorithms to the Human Activity Recognition dataset obtained from smartphone sensors. I also explored the effects of dimensionality reduction using PCA on the clustering results. Through clustering analysis, I

identified distinct groups of data points representing different activities performed by individuals, such as walking, walking upstairs, walking downstairs, sitting, standing, and laying. Each cluster represents a specific activity, or a combination of activities characterized by patterns in the sensor data captured by the smartphone accelerometer and gyroscope. The clusters identified through clustering algorithms provide insights into the patterns and variations in human activities based on sensor data. Understanding these clusters helps in analyzing and recognizing human activities accurately, which is essential for various applications such as health monitoring, fitness tracking, and behavior analysis.

One of the main scientific bottlenecks encountered in this project was the high dimensionality of the sensor data, which could lead to computational inefficiency and reduced clustering performance. Another bottleneck was the challenge of selecting optimal parameters for clustering algorithms, such as the number of clusters for K-Means and the epsilon parameter for DBSCAN.

To address the high dimensionality issue, I employed dimensionality reduction techniques such as PCA, which effectively reduced the feature space while preserving relevant information. I utilized data preprocessing techniques to standardize the data and applied noise filters to enhance data quality before clustering. Additionally, I employed heuristic methods such as the elbow method and silhouette score to select optimal parameters for the clustering algorithms. By implementing these strategies, I overcame the scientific bottlenecks associated with high-dimensional data and parameter selection, leading to improved clustering performance and computational efficiency.

Overall, this project demonstrates the effectiveness of clustering algorithms in identifying meaningful patterns in sensor data and recognizing human activities. Through proper data preprocessing, dimensionality reduction, and parameter tuning, I was able to overcome scientific bottlenecks and obtain reliable clustering results. These findings contribute to the advancement of research in human activity recognition and have practical implications in various domains, including healthcare, sports, and human-computer interaction. Besides it was a great task enhancing my knowledge in machine learning algorithms and dealing with these sets of datasets and understanding the concept and opportunities with these datasets.