Project: Human Activity Clustering

By Ahmad Baig

Introduction

The goal of this project is to explore how we can use data from smartphone sensors to figure out what kind of activity a person is doing. We're using a dataset that includes information from 30 people who did six different activities: walking, walking upstairs, walking downstairs, sitting, standing, and laying down. They carried a Samsung Galaxy S2 smartphone on their waist, which collected data through its accelerometer and gyroscope.

Before we dive into analyzing the data, we'll clean it up and organize it properly, which involves filtering out noise and separating different types of signals. After that, we're going to use two methods, KMeans and DBSCAN, to group the data into clusters. These methods help us see patterns in the data by grouping similar activities together without needing us to tell the computer what to look for ahead of time.

Once we have our clusters, we'll also try reducing the number of dimensions in our data. This process can make the data simpler and sometimes even highlight patterns better. After reducing the dimensions, we'll cluster the data again to see if or how our results change.

The report will go through each of these steps in detail, explaining how we preprocessed the data, how we applied the clustering methods, and what we found out from our analysis. The aim is to understand the dataset better and see what insights we can gather from the clustering and dimensionality reduction processes.

Data Processing

For our data processing, we have performed following techniques:

Missing Values

Missing values can affect our clustering, therefore, we must either impute or remove such rows. Luckily, we found **no missing values** (such as N/A) in the dataset.

Standardization

We standardized our dataset as most clustering techniques give better results when using it. Before standardization, our dataset's mean was at **0.578** while the standard deviation (SD) was at **0.2792**. This indicated the need for standardization as the mean should be closer to 0 while SD should be near to 1.

After applying our technique, the mean was reduced to $\mathbf{2.2^{-16}}$ while the SD was increased to **1.0000485519384645**.

Outlier Removal

For detecting and removing the outliers, we chose **IQR technique** as it is easy to interpret and implement. During investigation, we found that the dataset requires that we implement a **lenient** range than the strict 25% and 75% quartile range. Therefore, we applied a **5% and 95% range** to remove the outliers. After removing outliers, the number of rows were reduced from **10299** to **7663**.

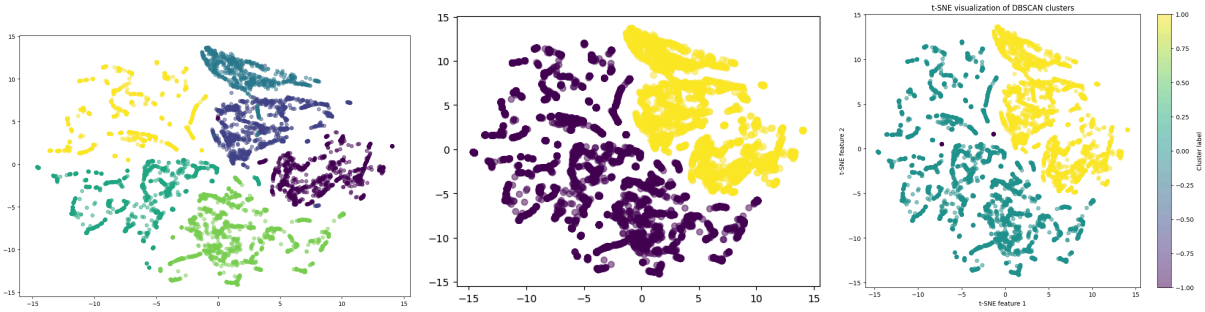
Feature Selection

We first removed all the **highly correlated features** (>0.85) as they overpower the effect of other meaningful features. We also removed **minorly correlated features** (<0.1) since they don't contribute enough towards meaning behind data. We were left with 366 features after this procedure.

We then moved towards calculating **mutual info** and **feature importance** by random forest. Among these features, only the top 20 from each technique made it into the final features - reducing the amount of features to **30** only.

Modeling

For our clustering, we selected KMeans and DBScan as our two choices.



The first figure shows the clustering on the original labels of the dataset.
The second figure shows clustering on K-means labels.
The third figure shows clustering on DBSCAN labels.

How do you choose the number of clusters in K-Means, is it the same number of clusters for DBSCAN?

The number of clusters of K-Means was found by the **Elbow Method**. This method uses the total *within-cluster sum of squares* as its technique. We found out that **2 to 3** seemed to be the optimal number of clusters.

For DBScan, the best number of clusters having least noise was calculated by **Silhouette Score.** We found that 2 was the optimal number for clusters, same as KMeans.

How do you find the optimal parameters' values?

In order to find the optimal parameters' values for KMeans, we used **GridSearchCV**. We found the following configuration for KMeans is optimal:

Init: k-means++
Clusters: 2
Tol: 0.0001
Max_Iterations: 300

For DBScan, we looked for parameters which gave us least noise and better **silhouette score**. The best parameters were:

eps: 2.82
min_samples: 60 (rule of thumb = 2 * features)

After dimensionality reduction, for DBScan, following were the best parameters:

eps: 0.77
min_samples: 4

What data processing steps do you apply?

As discussed earlier, the data used for clustering underwent many techniques involving missing values handling, standardization, outlier removal and feature selection. Later on, dimensionality reduction was also introduced.

What is the dimensionality reduction technique that you choose, and why?

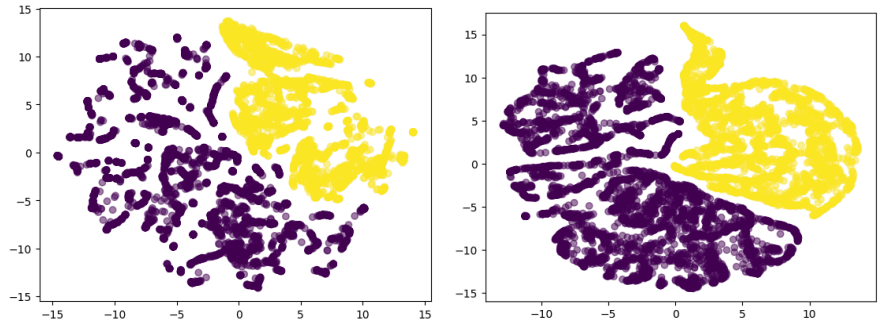
After getting initial results of KMeans clustering, we implemented **Principal Component Analysis** (PCA) on our dataset. This technique combines the effect of multiple features into two, reducing the model's computational and interpretability complexity.

Does it have any effect on your code efficiency, both in terms of computational efficiency and clustering output?

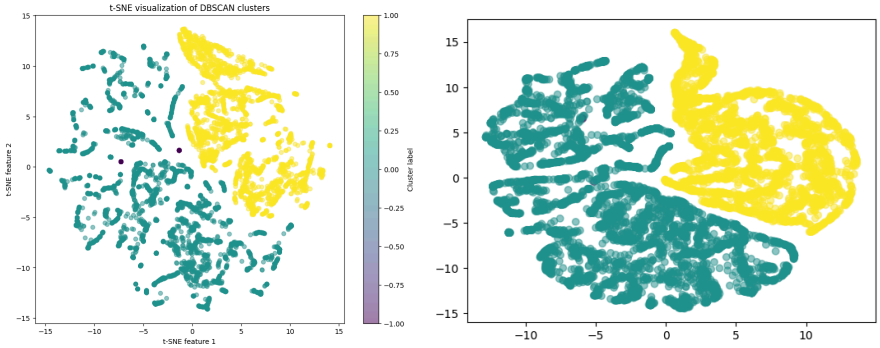
In terms of **computational efficiency**, clustering speed was increased. The probable reason can be that the original dataset contained **30 features** while the reduced dimensions dataset contained only two.

In terms of **clustering output**, the dimensionality-reduced data clusters were *more distinguishable* as compared to the original clusters.

K-Means:



DBScan:



The graph on the left shows the original clustering, while right shows **Dimensionality-Reduced** Clustering.

How do you compare the outcome of this model with the model where the dimensionality reduction technique was not applied to the dataset?

We used four metrics in order to measure the effects of dimensionality reduction.

Metric	**KMeans (Before)**	**KMeans (After)**	**DBScan (Before)**	**DBScan (After)**
Silhouette Score	0.485	0.599	0.471	0.381
Adjusted Rand Index (ARI)	0.321	0.321	0.320	0.321

Normalized Mutual Information	0.547	0.547	0.544	0.546
V-Measure	0.547	0.547	0.544	0.546

Summary:

Dimensionality reduction seems to have positively affected the KMeans clustering, enhancing the cluster separation. In contrast, it appears to have a negative impact on the DBScan clustering, making the clusters less distinct.

However, the stability of the clusters (as measured by ARI) and the correspondence with the true labels (as measured by NMI and V-measure) remained mostly unchanged for both clustering methods. This explains that while the overall cluster structures were preserved, the distinctness and possibly the quality of the clusters were influenced by dimensionality reduction differently for KMeans and DBScan.

For visualization, have you applied any dimensionality reduction techniques?

In terms of visualization, we used **T-SNE** in order to display our high dimensional data within two dimensions making it easier to understand.

Conclusion

How do you interpret the identified clusters? What do they represent?

To interpret our clustering for the given problem, we cross-tabulated these cluster labels with the original activity labels to see how they were grouped together.

K-means Clustering

Label 0 from k-means has been assigned to all instances of SITTING (4), STANDING (5), and LAYING (6) which are the **static activities**.

Label 1 from k-means has been assigned to all instances of WALKING (1), WALKING_UPSTAIRS (2), and WALKING_DOWNSTAIRS (3), which are the **dynamic activities**.

DBSCAN Clustering

Label 0 from DBSCAN has been assigned to almost all instances of SITTING (4) and STANDING (5), and most of LAYING (6), with 65 instances of LAYING labeled as noise (indicated by -1).

Label 1 from DBSCAN has been assigned to all instances of WALKING (1), WALKING_UPSTAIRS (2), and WALKING_DOWNSTAIRS (3), matching the k-means dynamic activity group.

In summary, both clustering methods have effectively distinguished between static activities and dynamic activities. DBSCAN has additionally identified some instances of LAYING as outliers.

What were the “scientific” bottlenecks?

The dataset was **very high dimensional** making it difficult to find the best optimal parameters for DBScan.

How did you overcome them?

After removing highly correlated features, we were able to find optimal parameters having low noise and more silhouette score.