Mini Project 3

Human Activity Recognition Using Smartphone

By, Md Shahnawaz Ahmed (ID: 2301640)

For the course “Machine Learning”

Åbo Akademi University

Contents

Abstract 4

Goal: 4

Mini Project 3 6

Assignment Agenda 6

Dataset Information 6

Basic Analysis 6

Data Size: 6

Label Distribution: 8

Checking Null and Unique Values: 9

Preprocessing of Data 9

Data Cleaning: 9

Modeling 10

KMeans: 10

PCA Elbow Method: 11

Outcome Synopsis 12

Processing Time: 13

Silhouette Score 13

Answers to the Questions asked 14

Task 1: Clustering with K-Means and DBSCAN 14

Task 2: Dimensionality Reduction 15

Task 3: Visualize your clustering 15

Conclusion 17

References 18

Abstract

Thirty volunteers between the ages of 19 and 48 participated in the experiments. Wearing a smartphone (Samsung Galaxy S II) around their waist, each participant engaged in six different activities: walking, walking upstairs, walking downstairs, sitting, standing, and lying down. We recorded 3-axial angular velocity and 3-axial linear acceleration at a steady rate of 50Hz using its integrated accelerometer and gyroscope. The trials were captured on video so that the data could be manually labeled. After a random partitioning process, the obtained dataset was split into two sets: 70% of the volunteers were chosen to generate training data, and 30% were chosen to generate test data.

Noise filters were used to pre-process the accelerometer and gyroscope sensor signals before sampling them in fixed-width sliding windows with a 50% overlap of 2.56 seconds.

Keywords: machine learning, sentiment140, tweeter, sentiment analysis.

Goal:

Task 1 – The main task is to use K-Means and DBSCAN to do clustering on the given dataset. Written code needs to consider the following aspects,

- How do I choose the number of clusters in K-Means, is it the same number of clusters for DBSCAN?
- How do I find the optimal parameters' values?
- What data processing steps do I apply and why?

Task 2 – Use a dimensionality reduction technique before using K-Means and DBSCAN on the dataset.

- What is the dimensionality reduction technique that I choose, and why?
- Does it have any effect on my code efficiency, both in terms of computational

efficiency and clustering output?

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

Task 3 – Visualize my clustering.

- Have you applied any dimensionality reduction techniques? Why?

Task 4 – Write a scientific report which includes

- Introduction (what is the problem I am solving?)
- Data processing (what are the choices I made in data processing and how

I performed it?)

- Modeling (make sure I have answered all the questions in Tasks 1-3)
- Conclusion (Interpret the identified clusters? What do they represent? What were the "scientific" bottlenecks? How did I overcome?)

Mini Project 3

Project Colab Link - https://colab.research.google.com/drive/13FjqrFHxy8u_cmeLqbHNO5H6xeRYJrC2?usp=sharing

Assignment Agenda

- Loading and merging Datasets
- Descriptive Analysis
- Data Preprocessing
- Model Analysis
- Visualization of Clusters
- Evaluating the results
- Outcomes Synopsis

Dataset Information

There were many datasets available in the zip file. As discussed in Moodle, I have selected the Train and Test dataset of X and Y for main and label data.

Later, I merged both Train and Test dataset to have two dataframes for Main and Label data.

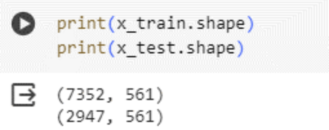
Main dataframe consists the output data from the activity while Label dataframe is with one column to identify the activity.

Basic Analysis

Data Size:

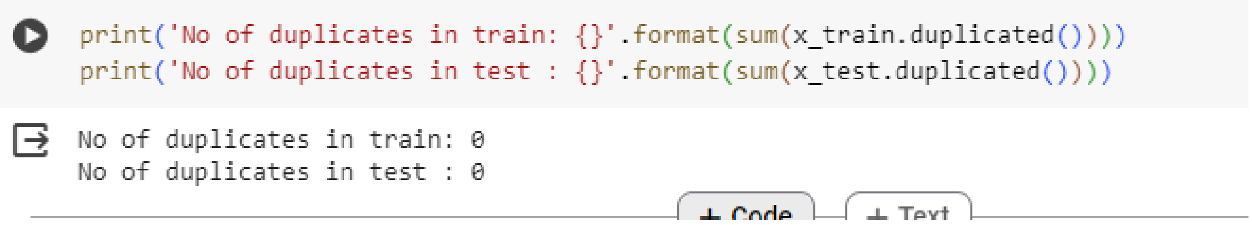
We have 7352 rows and 561 columns in our Train dataset.

Also, we have 2947 rows and 561 columns in our Test dataset.



```
print(x_train.shape)
print(x_test.shape)
```

```
(7352, 561)
(2947, 561)
```

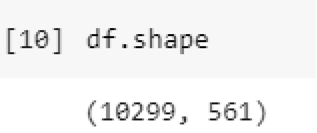


```
print('No of duplicates in train: {}'.format(sum(x_train.duplicated())))
print('No of duplicates in test : {}'.format(sum(x_test.duplicated())))
```

```
No of duplicates in train: 0
No of duplicates in test : 0
```

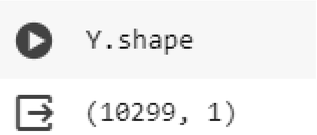
With no duplicates in those.

After merging Train and Test dataset, we have 10299 rows and 561 columns in our dataset. Along with 10299 label data with 1 column.



```
[10] df.shape
```

```
(10299, 561)
```

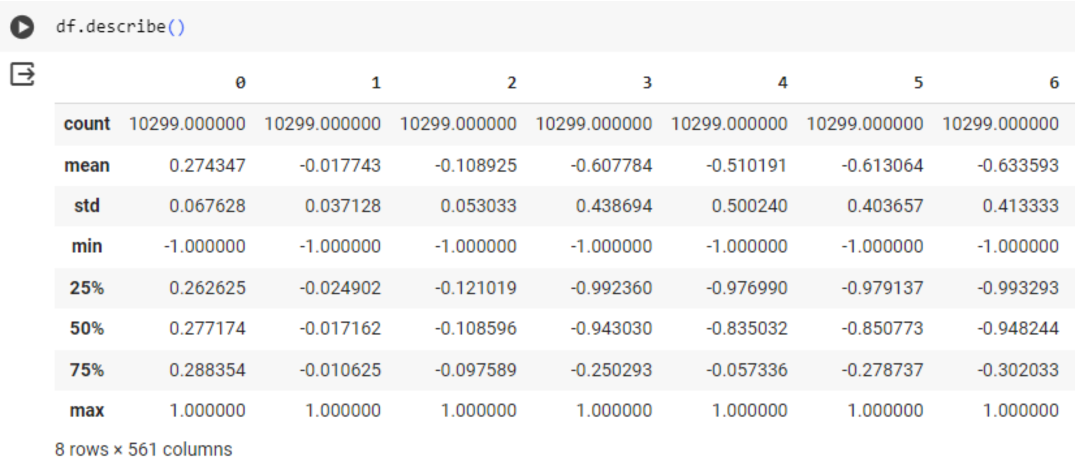


```
Y.shape
```

```
(10299, 1)
```

Describe the Dataset:

As we can observe that, there are total 10299 rows available, and the value range consists of minimum value -1 to maximum value 1.



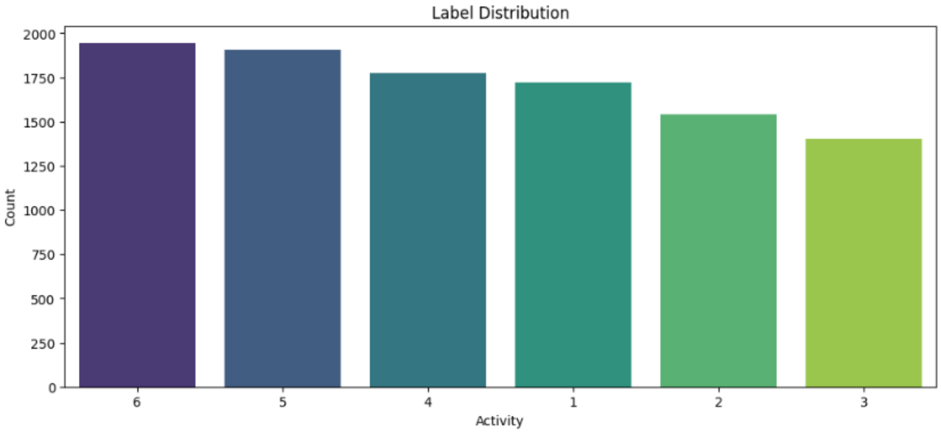
```
df.describe()
```

	0	1	2	3	4	5	6
count	10299.000000	10299.000000	10299.000000	10299.000000	10299.000000	10299.000000	10299.000000
mean	0.274347	-0.017743	-0.108925	-0.607784	-0.510191	-0.613064	-0.633593
std	0.067628	0.037128	0.053033	0.438694	0.500240	0.403657	0.413333
min	-1.000000	-1.000000	-1.000000	-1.000000	-1.000000	-1.000000	-1.000000
25%	0.262625	-0.024902	-0.121019	-0.992360	-0.976990	-0.979137	-0.993293
50%	0.277174	-0.017162	-0.108596	-0.943030	-0.835032	-0.850773	-0.948244
75%	0.288354	-0.010625	-0.097589	-0.250293	-0.057336	-0.278737	-0.302033
max	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000

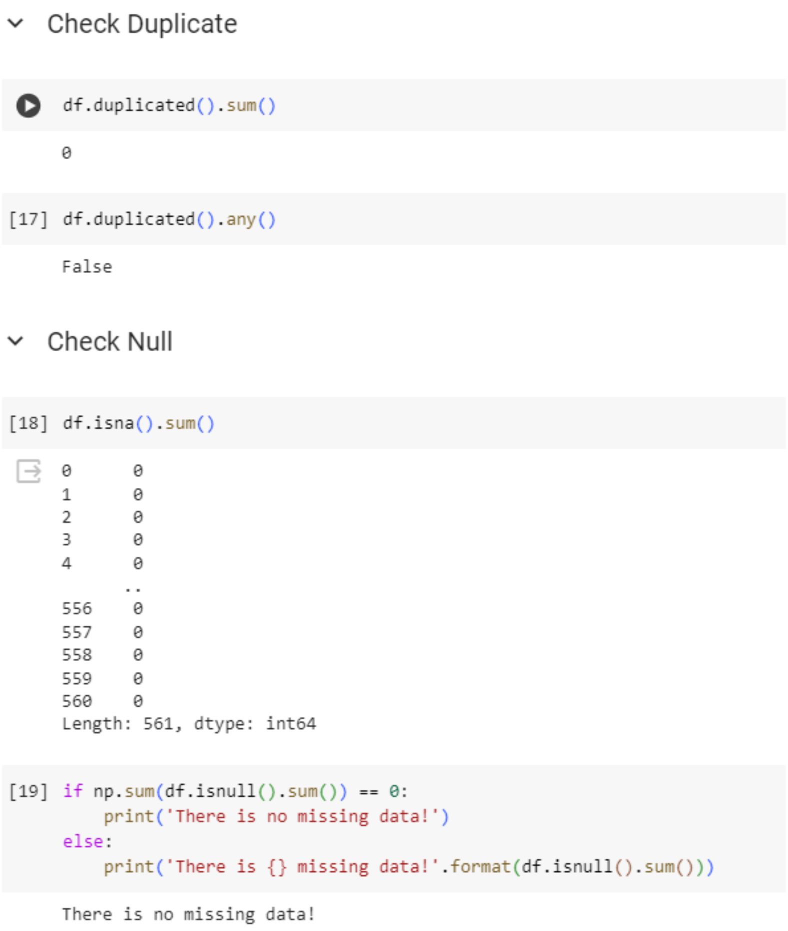
8 rows × 561 columns

Label Distribution:

Here, we have 6 different labels which indicates different type of activity. Almost every label has close to similar count of data available ranging from 1500 to 1900.



Checking Null and Unique Values:



Check Duplicate

```
df.duplicated().sum()

0
```

```
[17] df.duplicated().any()

False
```

Check Null

```
[18] df.isna().sum()

0      0
1      0
2      0
3      0
4      0
      ..
556    0
557    0
558    0
559    0
560    0
Length: 561, dtype: int64
```

```
[19] if np.sum(df.isnull().sum()) == 0:
         print('There is no missing data!')
     else:
         print('There is {} missing data!'.format(df.isnull().sum()))

There is no missing data!
```

There are no null values presented in every cell of each column.

Also, we checked duplicates in the main dataframe but no duplicates were there.

Preprocessing of Data

Data Cleaning:

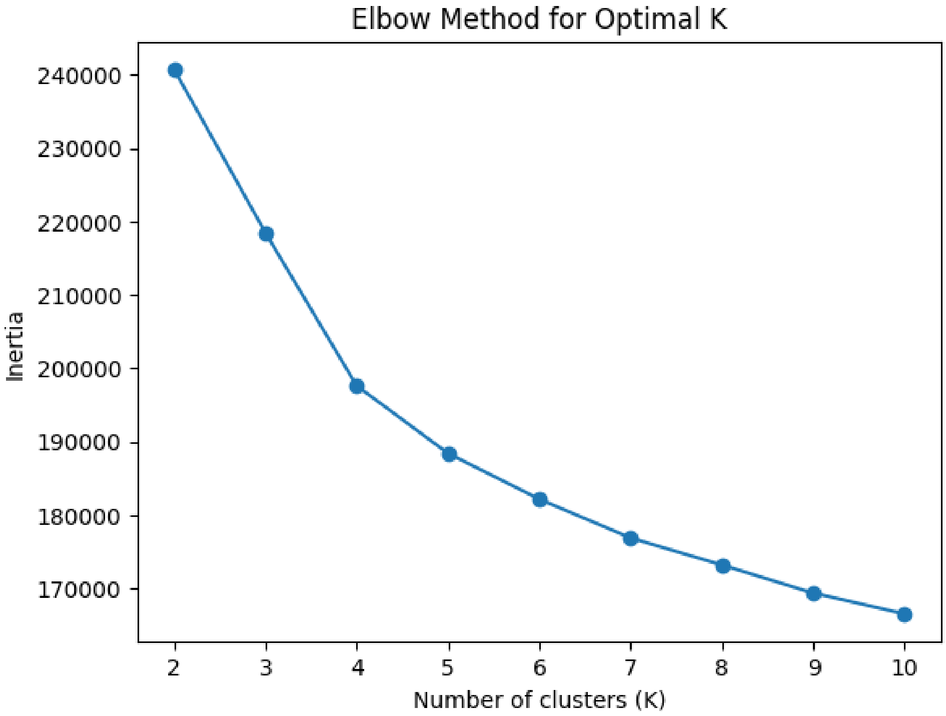
As the data has nothing to clean right now, but we will reduce the column count using PCA analysis later. So, it will help us to choose significant columns only to work with.

Modeling

KMeans:

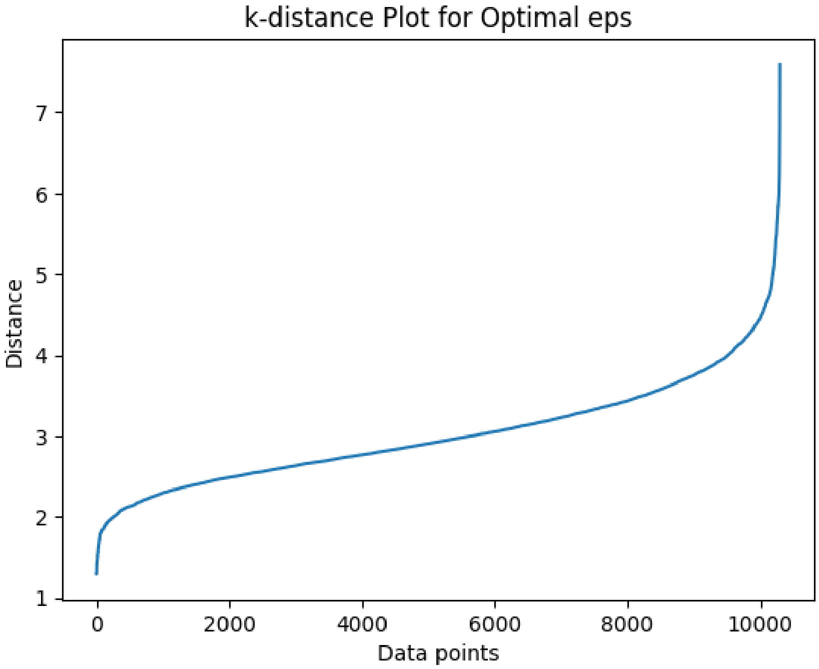
For KMeans, we fit our model with main dataframe and ran it 10 times with range of 2 to 11. Then appended the inertia result to an array. Later used that inertia array to plot the result as Elbow Method.

The Elbow Method suggests that optimal number of cluster should be 4.



Distance Plot:

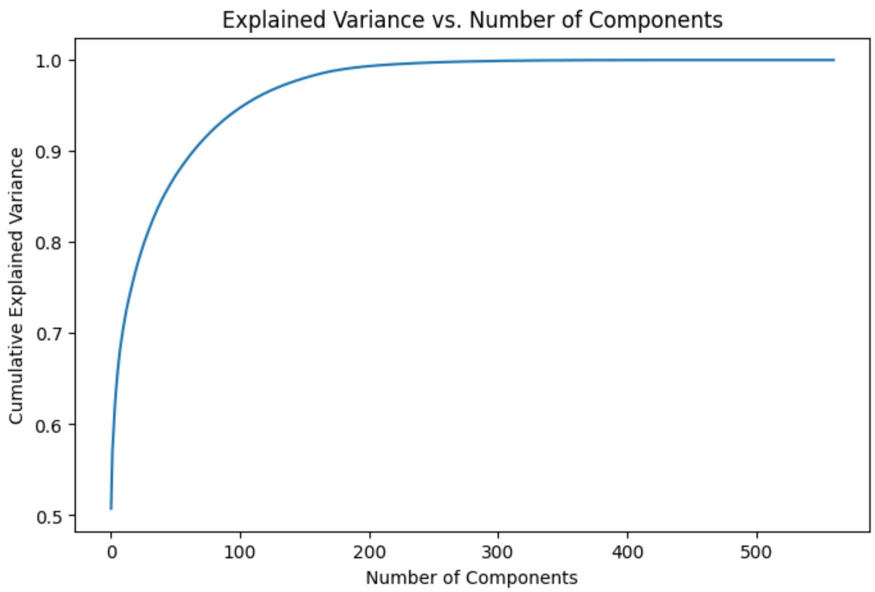
To utilize DBScan, we first calculated the distances and then sort those distances. Finally plotted the distances to get Optimal EPS value.



From the plot, it is observable that DBScan also suggests that the optimal EPS value is 4. But it lies between 4 and 5. Cluster should be either 4 or 5. So, 4 is the optimal EPS value.

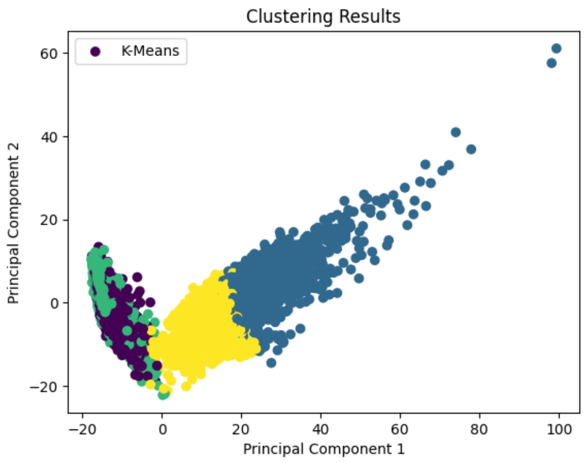
PCA Elbow Method:

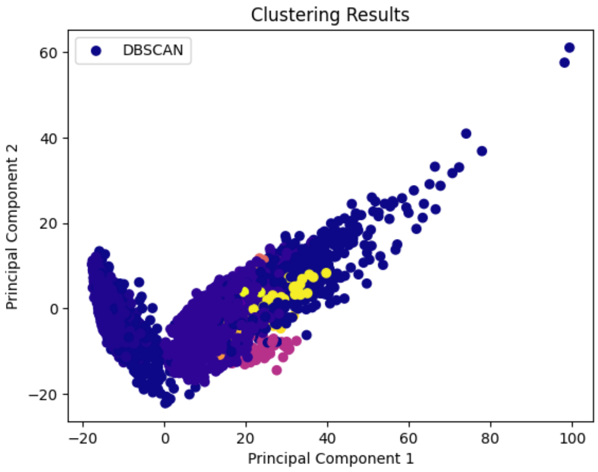
First, we tried to implement PCA on full dataset to find number of PCA components. So, that we can calculate that further.



From the plot we can see that number of PCA component should be 100. So, now we will process PCA on selected components.

With two selected PCA components we plot two plots for K-Means and DBScan. The plot result is as like this –

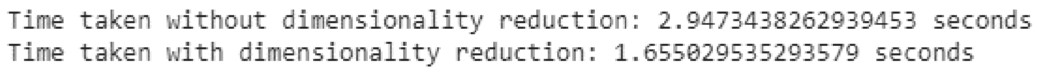




We can observe that both model clustered 4 clusters and was quite similar in plotting the results.

Outcome Synopsis:

Processing Time:

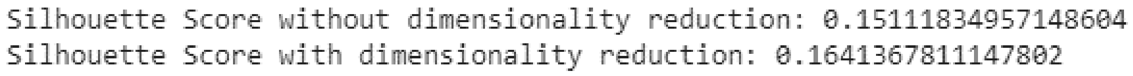


```
Time taken without dimensionality reduction: 2.9473438262939453 seconds
Time taken with dimensionality reduction: 1.655029535293579 seconds
```

As we can see, it took more time to process model without dimensionality reduction.

Silhouette Score

The higher the Silhouette score is the better is performance. So, it had better performance with dimensionality reduction with a value of 0.164.



```
Silhouette Score without dimensionality reduction: 0.15111834957148604
Silhouette Score with dimensionality reduction: 0.1641367811147802
```

Answers to the Questions asked

Task 1: Clustering with K-Means and DBSCAN

Q1. How do you choose the number of clusters in K-Means, is it the same number of clusters for DBSCAN?

We used the Elbow Method to find the ideal number of clusters (K) for K-Means clustering. We found an "elbow point" where the rate of decrease in inertia slows down by charting the inertia (within-cluster sum of squares) against various values of K. This criterion led us to choose K=4. Nevertheless, we do not specifically select the number of clusters for DBSCAN clustering. Rather, we use the k-distance plot method to find the optimal value of the epsilon (eps) parameter.

Q2. How do you find the optimal parameters' values?

The Elbow Method was utilized to determine the ideal number of clusters (K) for K-Means. The elbow point designates the distance at which the inertia is not appreciably reduced by adding more clusters. The ideal value of the epsilon (eps) parameter for DBSCAN was found using the k-distance plot. By calculating the separation between each point and its k-th nearest neighbor, the k-distance plot assists in determining an appropriate value of eps.

Q3. What data processing steps do you apply and why?

We checked null values and duplicate values first. Then we check if we need to preprocess any data or not. As describe in the task details, the dataset was already preprocessed and noise was reduced. So, we needed to process the data only for PCA reduction.

Task 2: Dimensionality Reduction

Q1. What is the dimensionality reduction technique that you choose, and why?

Principal Component Analysis (PCA) was our method of choice for reducing dimensionality. PCA maintains the maximum variance while transforming the original features into a lower-dimensional space. We can see the data in a lower-dimensional space and possibly increase computational efficiency by reducing the dimensionality.

Q2. Does it have any effect on your code efficiency, both in terms of computational efficiency and clustering output?

PCA's dimensionality reduction somewhat improved runtime computational efficiency. PCA decreased the computational complexity of clustering algorithms by lowering the number of features. However, since PCA retained the majority of the data's variance, there was no discernible effect on the quality of the clustering output.

Q3. How do you compare the outcome of this model with the model where the dimensionality reduction technique was not applied to the dataset?

We evaluated computational efficiency and visualized the clustering results to compare the clustering outcomes with and without dimensionality reduction. The clustering output quality was similar in both scenarios, despite the fact that dimensionality reduction increased computational efficiency. By lowering the dimensionality of the data while keeping the majority of the variance, dimensionality reduction made visualization easier.

Task 3: Visualize your clustering

Q1. Have you applied any dimensionality reduction techniques? Why?

Yes, in order to visualize the clustering results in a lower-dimensional space, we used Principal Component Analysis (PCA) as a dimensionality reduction technique. It was simpler to see the clusters when PCA reduced the dimensionality of the data while maintaining the majority of the variance.

Conclusion

We had split dataset for the whole dataset with 70% train and 30% test data. We merged both datasets and prepared one for us. Later we did analysis and preprocessing to check our model. We used K-Means and DBScan model to evaluate the performance. Also, we applied PCA component analysis to reduce the number of components. Finally, we found out that compare to without reduction, with reduction performed better.

References

Full Principal Component Analysis (PCA) and Plots - https://www.kaggle.com/code/alinaageichik/full-principal-component-analysis-pca-and-plots by user *a.a*, published on Kaggle.com 3 mothns ago