Dataset : [Mobile-phone-classification (kaggle.com)](https://www.kaggle.com/datasets/ojoolasehindeitunu/mobile-phone-classification?select=train.csv)

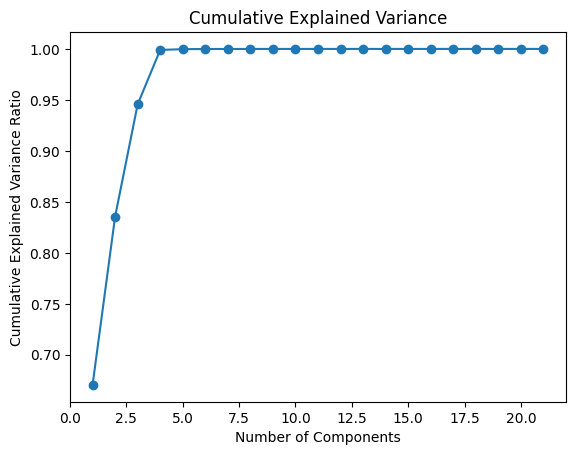
We used a mobile phone classification dataset, and we used specific features to make the data unlabeled.

**Phase 1:**

**Process**:

In our process in using **PCA**:

1. First we get the mean of the dataset
2. Then we shifted the mean
3. After that we compute the covariance matrix of our data
4. Then we got the eigenvectors and eigen values from the covariance matrix
5. Then we Normalized the Eigenvectors and sorted the eigen values
6. After that we computed the number of components needed for to have a 99% variance from the data, the 95% is a threshold we choose as it was preferred to choose a number between 95% and 99% according to many research topics about the PCA in this part of the process.
7. After that we got the perfect number of components Is 7 (which will give us and represent 99% of the whole data)



* Here is a plot for the relationship between cumulative variance and number of components as we can see their relationship is directly proportional as variance increase with the increase of number of components.

A graph with blue squares

Description automatically generated

* This plot represent the amount of variance of the data each component has , as we can see the first component has the highest rate (variance of data) , and this plot is just according to the first 7 components.

-Trying different Q-Matrix:

1) in our first case we tried number of components = 7

We got the shape of our data in the reduction become (2000,7)

2) Second Case we tried Num components = 10

The shape of the Matrix = (21,10)

Shape of reduction was (2000,10)

-In our opinion, the best Q-Matrix depends on the number of components, especially the ones that represent a high percentage of the data like the second plot we have shown earlier we have seen that the first component represents the highest amount of data.

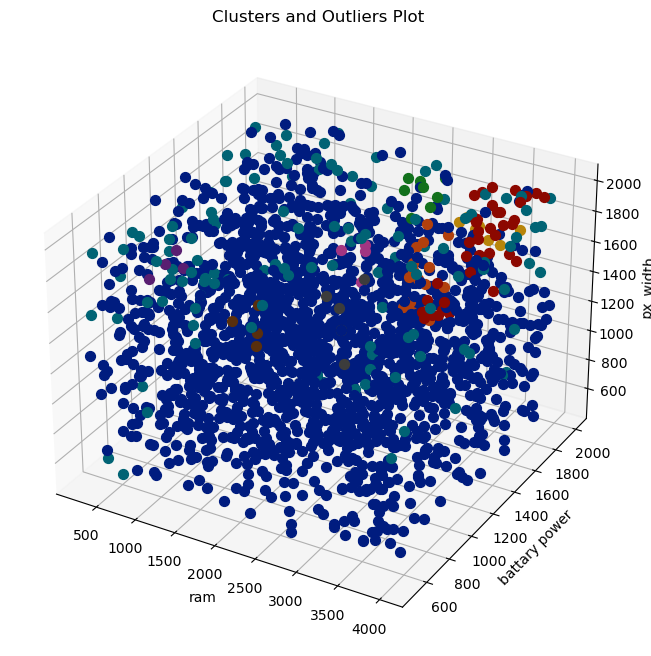
**Phase 2:**

We started by Creating DBSCAN clustering algorithm from scratch:

**First Case** (Applying DBSCAN in the original feature space)

Steps we walked through:

1. Calculated the Euclidean distance between data points in the feature space
2. Specified parameters we need for DBSCAN like: (epsilon = 280 , minimum points = 4)
3. We Identified the core points, border points and noise points
4. A graph with many colored dots

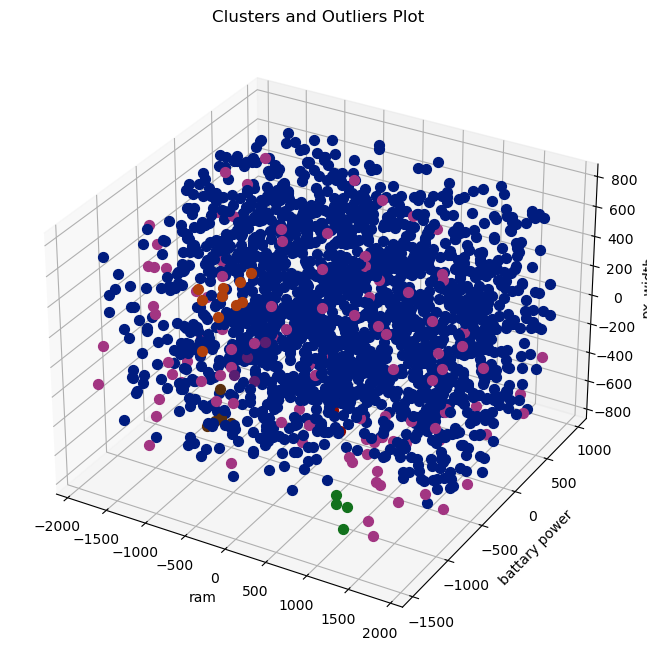
   Description automatically generatedAfter that clusters were created (in our case we had 9 clusters) and checked the size of each cluster.

**Second Case** (Applying DBSCAN in the PCA-reduced data)

Steps we walked through:

1. First, we used the PCA to reduce the dimensions of our data, we took the same steps we mentioned in Phase 1 as we already implemented PCA from scratch.
2. After applying PCA to the data we use the reduced data in DBSCAN
3. Calculated the Euclidean distances between data points in the reduced feature space.
4. With the same parameters we used in first case like epsilon and minimum points
5. We Identified the core points, border points and noise points.
6. Clusters were created (6 clusters)

A diagram of a graph

Description automatically generated

The Effect of using PCA:

* PCA can improve the interpretability of clusters by emphasizing the most important features.
* PCA could improve Clustering algorithms like DBSCAN, especially when dealing with a high-dimensional data.
* In our case PCA reduced the number of clusters created.