DATA ANALYTICS

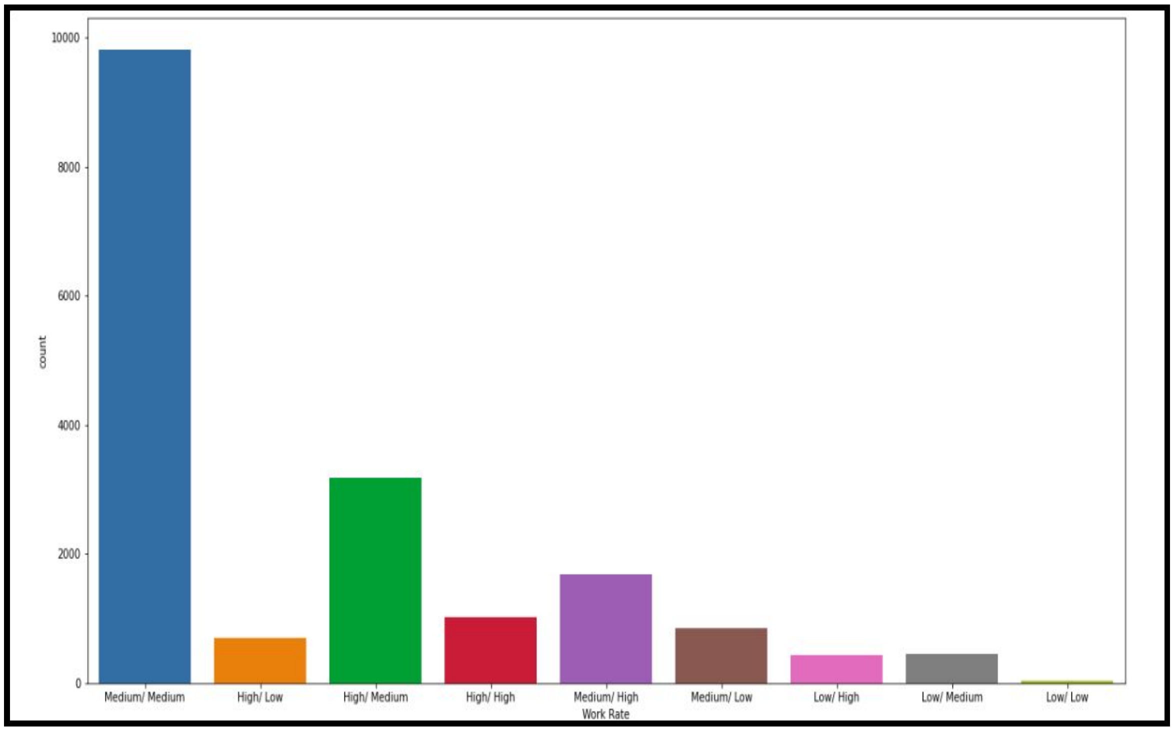
CLUSTERING PROJECT

Anishka Sachdeva (2018101112)
Satyam Viksit Pansari (2018101088)

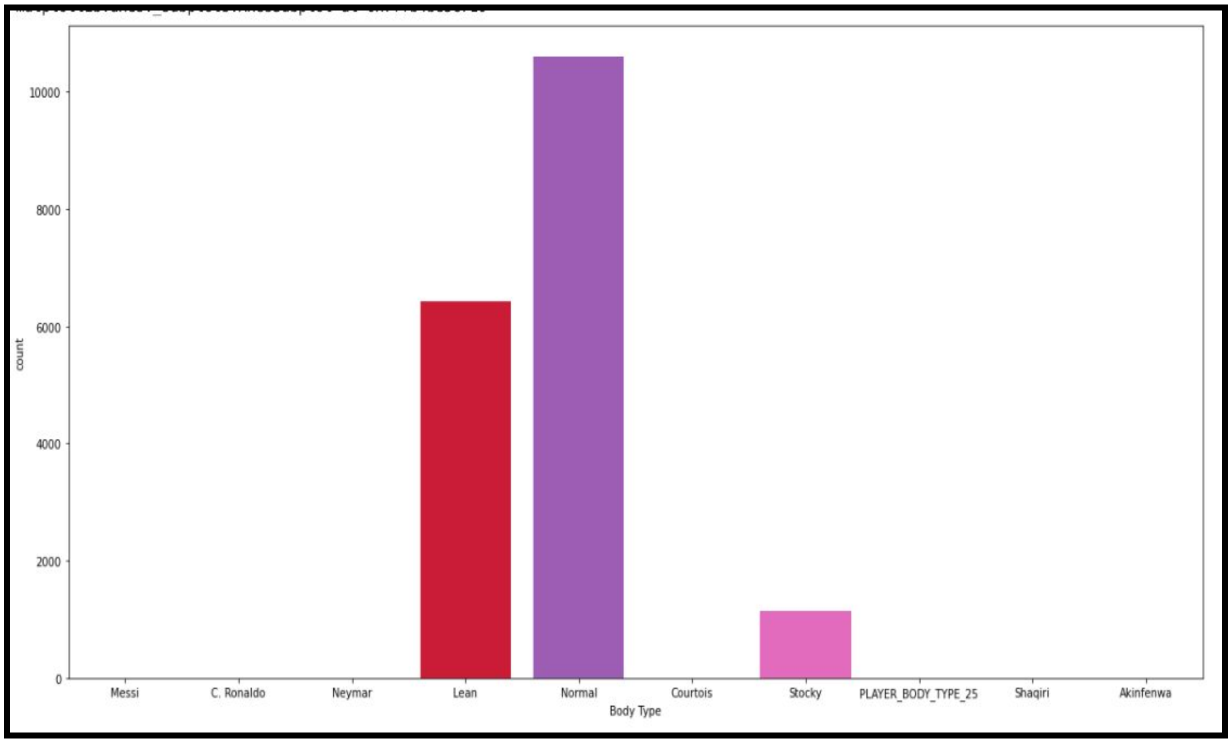
Note : The code for all parts is present in 2018101112_2018101088_clustering.ipynb file. References taken from GeeksForGeeks and other websites too.

PRE-TASK Data Preprocessing and Cleaning

1. **Features/Columns Dropped**
    - **Unnamed, ID, Name, Jersey Number, Photo, Nationality, Flag,Club, Club Logo, Value, Wage, Special, Joined, Loaned From, Contract Valid Until, Release Clause, Real Face -** Did not have any relevance with the skills required for football since our clustering methods were used to cluster players based on their skills.
    - **Work Rate** - This feature was related to skill for football but did not seem to be an important feature because most of the players had Medium/Medium work rate (as shown below) and hence would not account much in clustering.



- **Body Type** - This feature was related to skill for football but did not seem to be an important feature because most of the players had Normal body type (as shown in the figure below) and other body types accounted for a very low and negligible percentage and hence would not account much in clustering.



2. **Features/Columns Modified**
    - **Height** - The height was converted to inches so that it could be used as a numerical feature.
    - **Weight** - The weight was ripped off with its unit lbs as we needed to convert this feature into a numerical one to be used for clustering.
    - **Columns representing Positions (LS, ST etc.)** - The columns representing various positions like LS, ST, LB, RB etc. seemed to be one of the most important features regarding the skill set of footballers. Thus, we considered those features in the following manner :

        Each player had a default rating for each of the field positions (highest rating for the position he excelled in which they had skill) along with a bonus/additional rating (of about +2 or +3) which he gets if he plays well in that position. Thus, we averaged the bonus rating and added to the default one. Eg : 82 + 2 was taken as 83 i.e. (82 + 2/2).
    - **Preferred Foot** - The foot seemed to be an important feature and was a categorical feature. Thus we mapped Left and Right to 0 and 1 respectively.

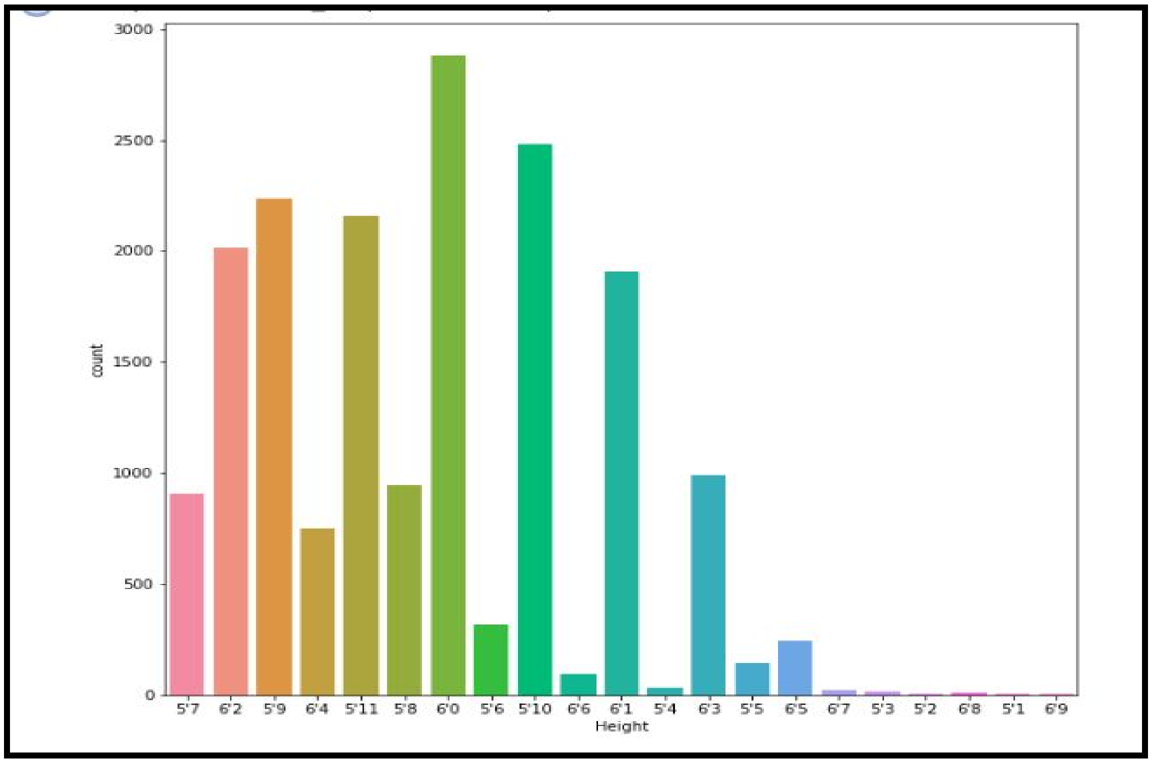
3. **Rows Dropped**

    We dropped all the rows which had some NULL values for some features since the number of such rows was only around 100 which would not impact much on the data loss.

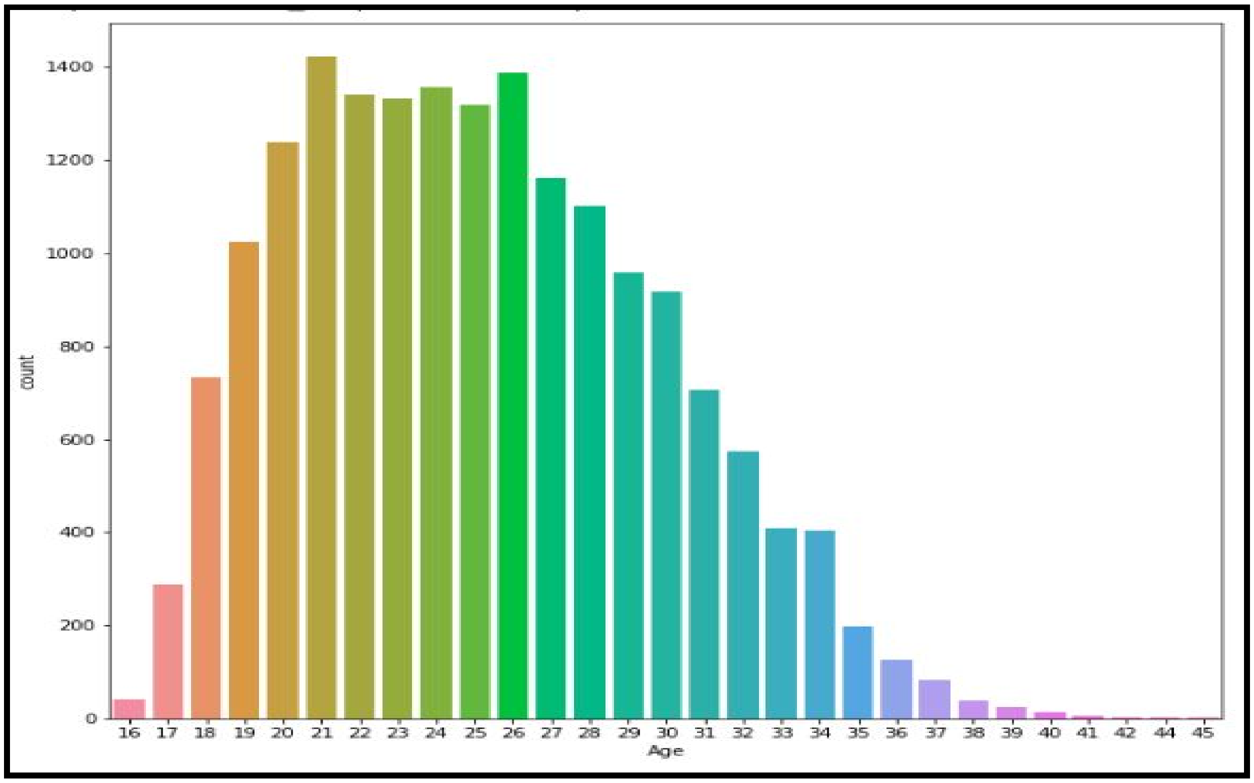
TASK 1 Data Visualization

Given data has the information about 18K football players and their different features, mainly their abilities and skills in the game including other attributes like their club, nationality, height etc.. Different features can be selected and the distribution can be visualised. Some such visualisations are as follows. Codes for the same are submitted in the notebook.

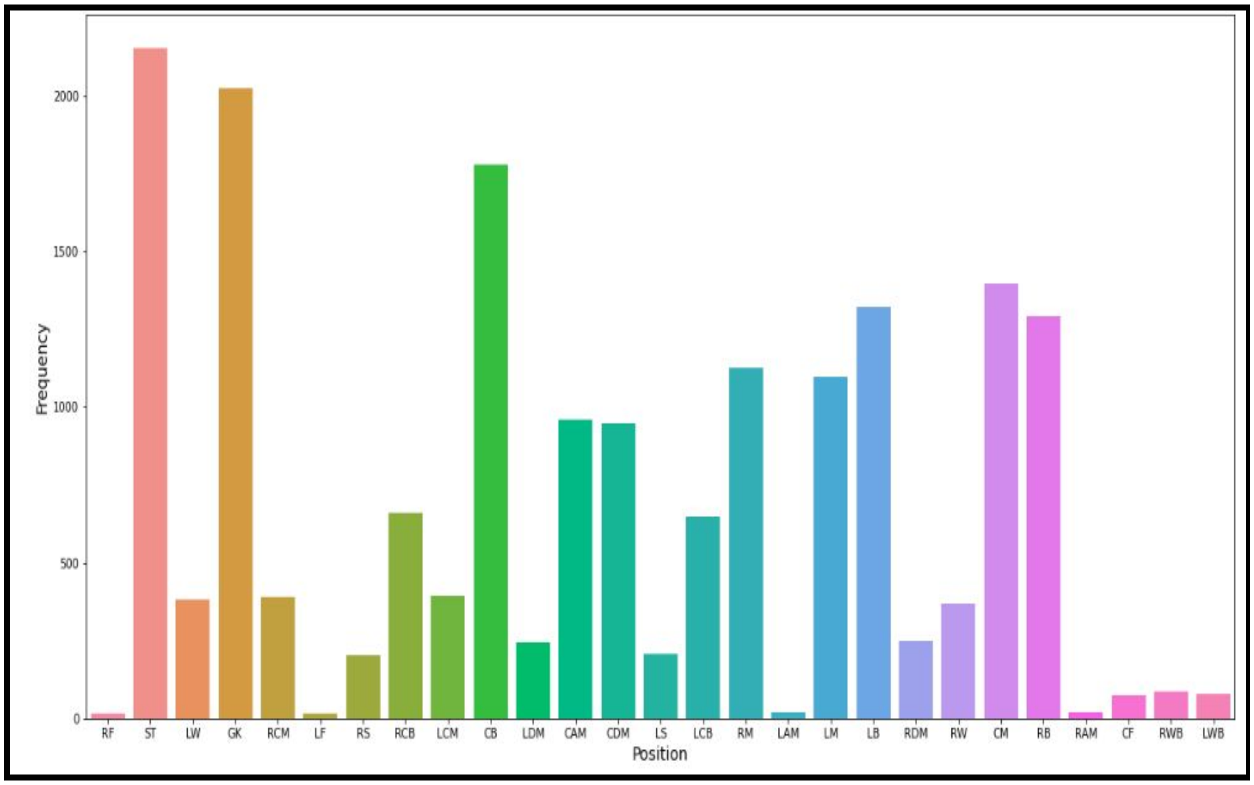
Note : Some more graphs in the Notebook Submitted.



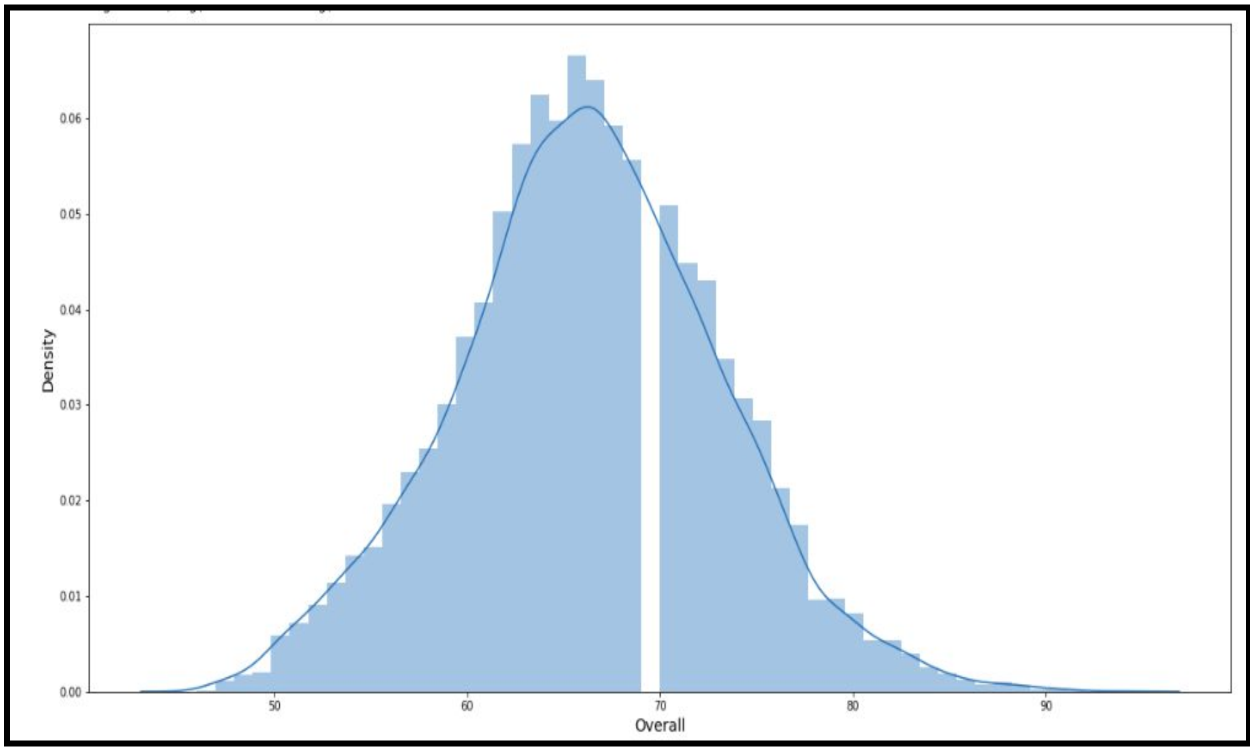
We plotted a "**HISTOGRAM"** for Height vs Number of Players



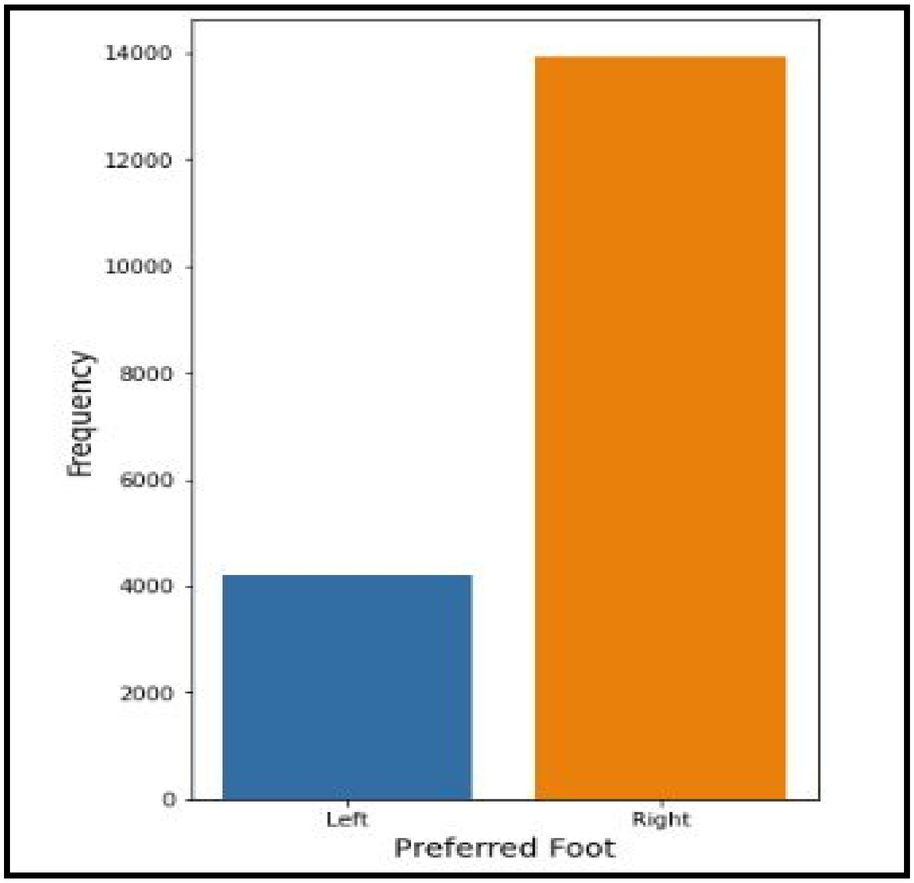
We plotted a **"HISTOGRAM"** for Age vs Number of Players



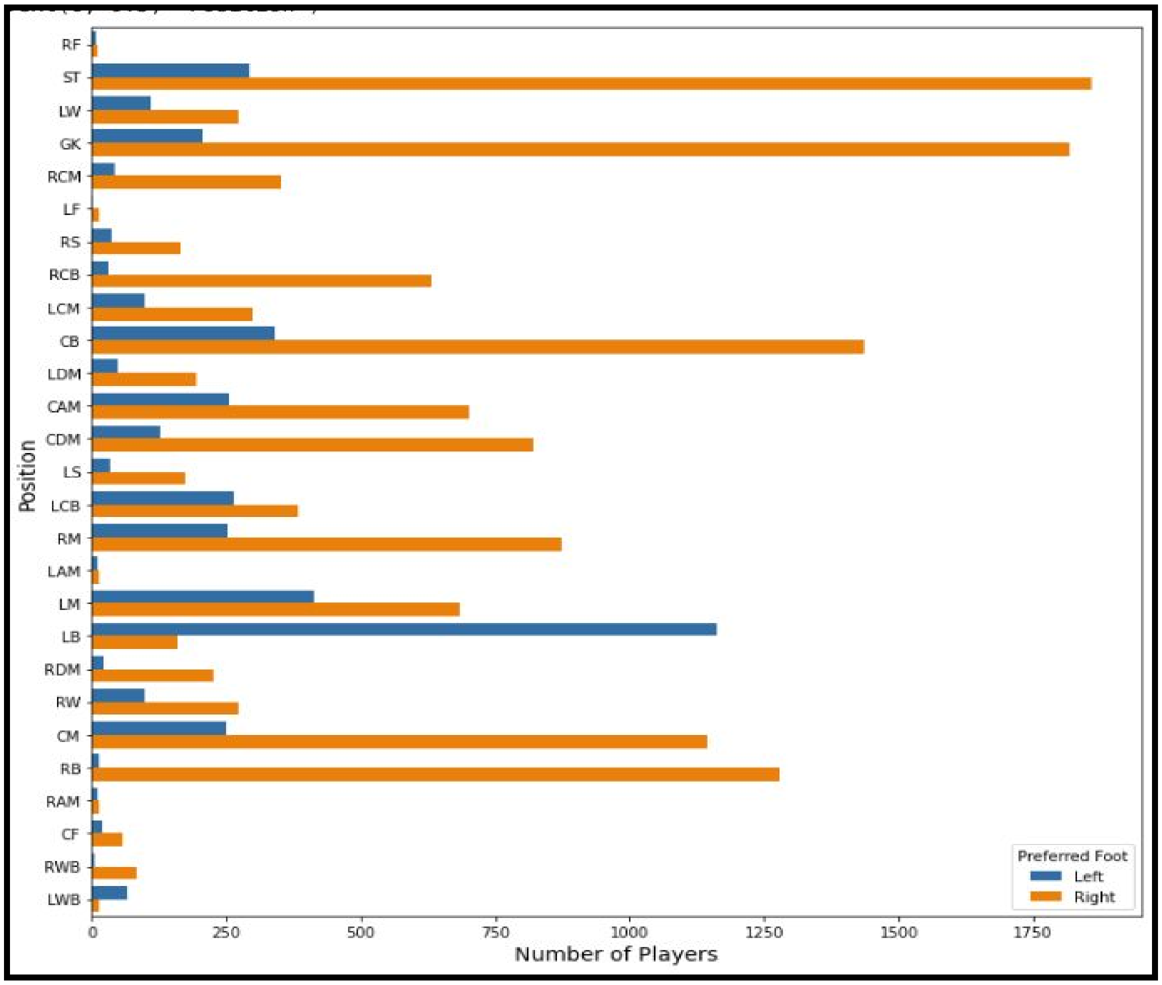
We plotted a **"HISTOGRAM"** for Field Positions.



“DISTPLOT” for Overall rating. We can see that it is a normal distribution curve.

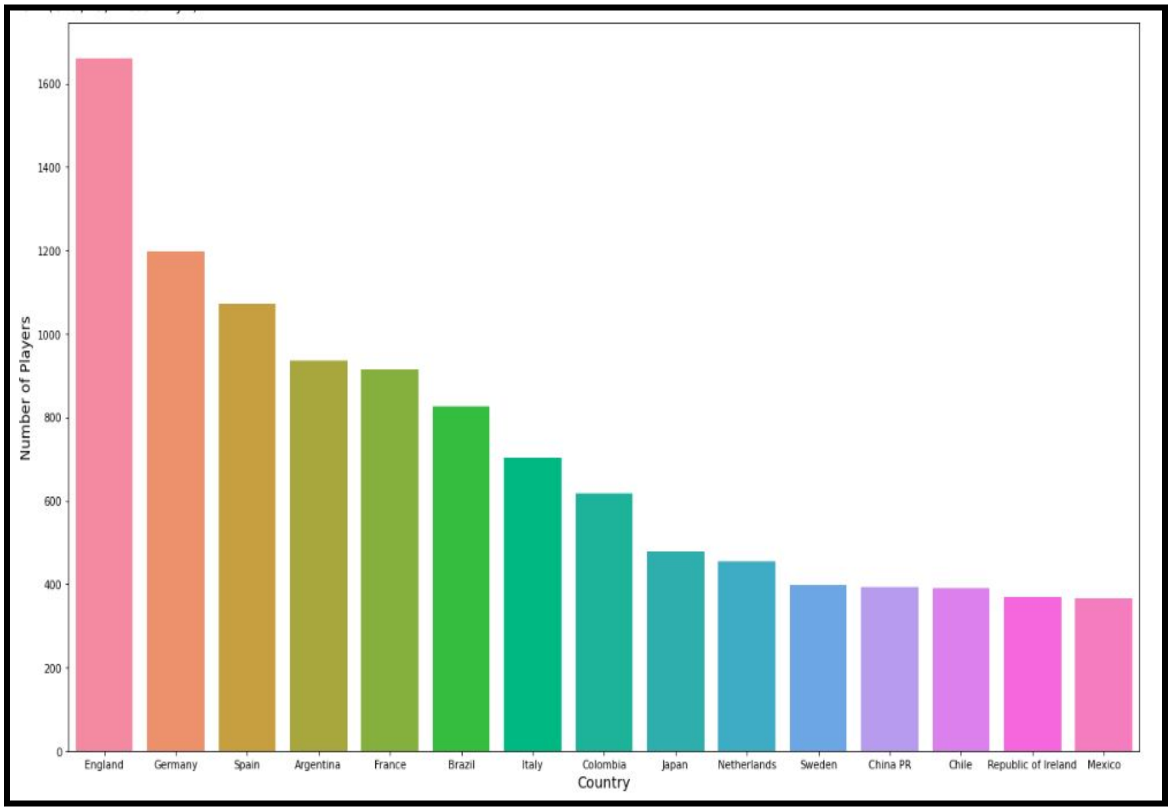


“COUNTPLOT” for Preferred Plot where we can observe that the majority of the players are Right-Footed.



We plotted the "**GROUPED COUNTPLOT**" to know the distribution of **"preferred foot"** for every position.

On looking at the graph, we got to know that even though preference of right foot is very high as compared to preference of left foot, there are some positions like LB, LF, LWB where the preference is higher for left foot and comparable for positions like LCB, LM, LW. This clearly states that in the left part of the field, left foot is preferred.



We plotted a **“BARPLOT”** for top 15 countries**.**

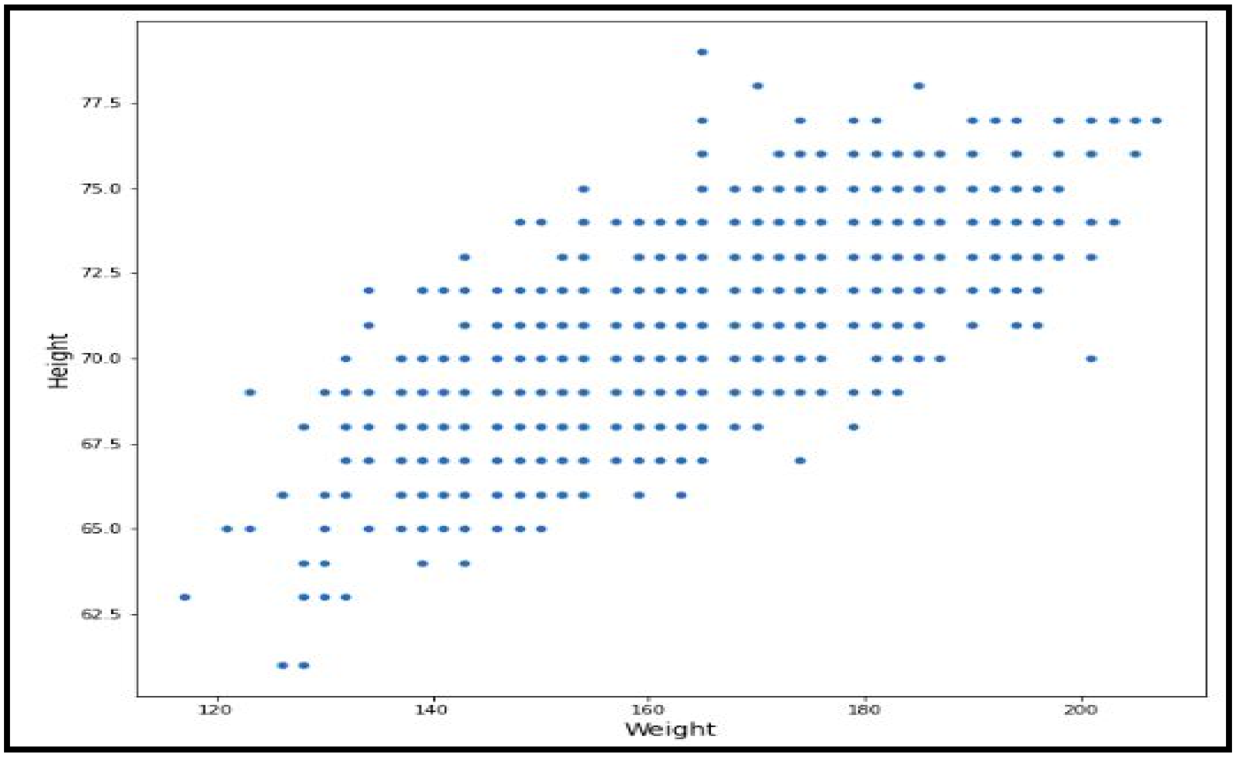
This graph shows us

that the country that

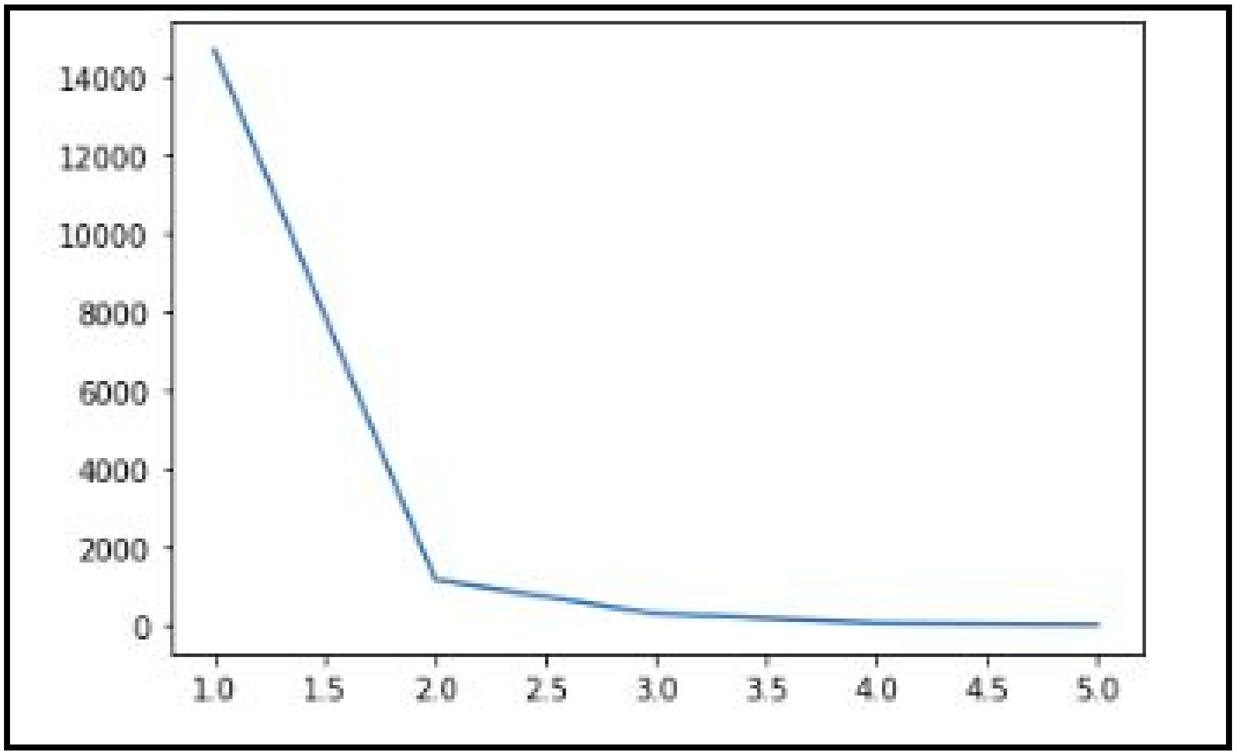
produces the most number of players is England, with over 1600

players. The next highest countries are Germany and Spain, which have players in the range (1000 - 1200) and no other country has over 1000

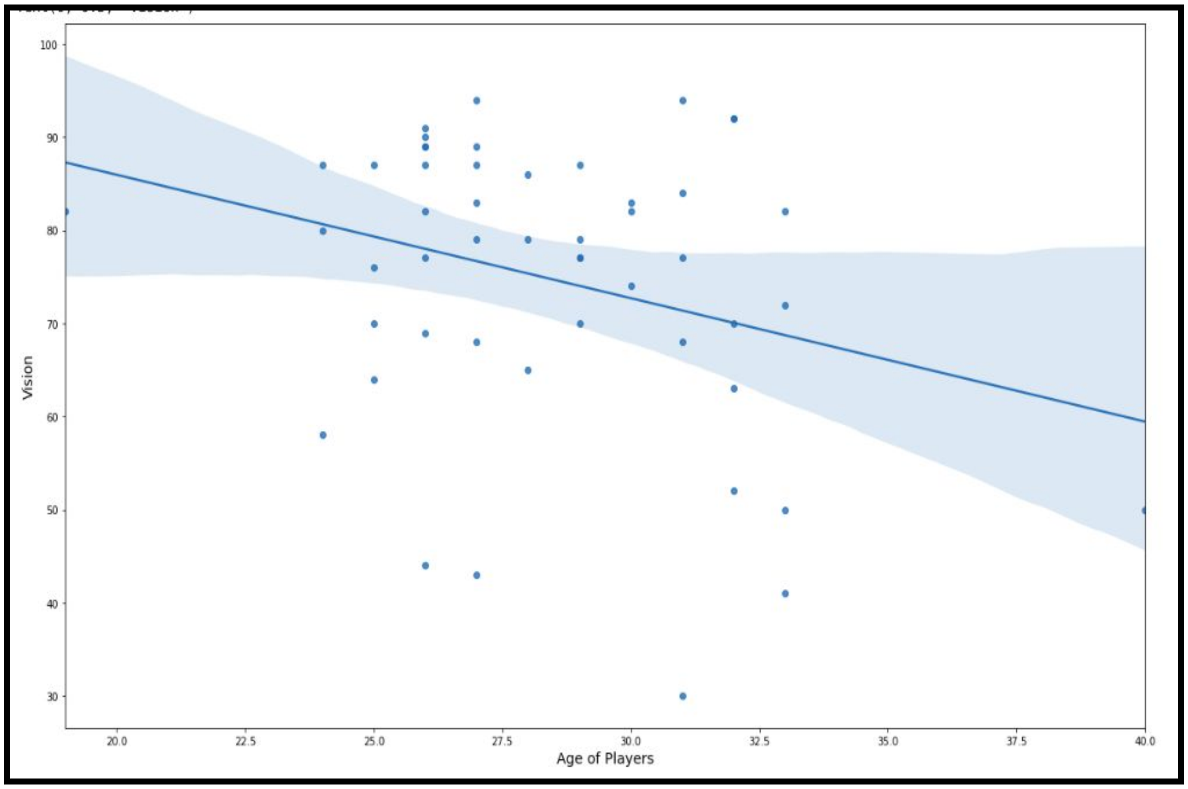
players.



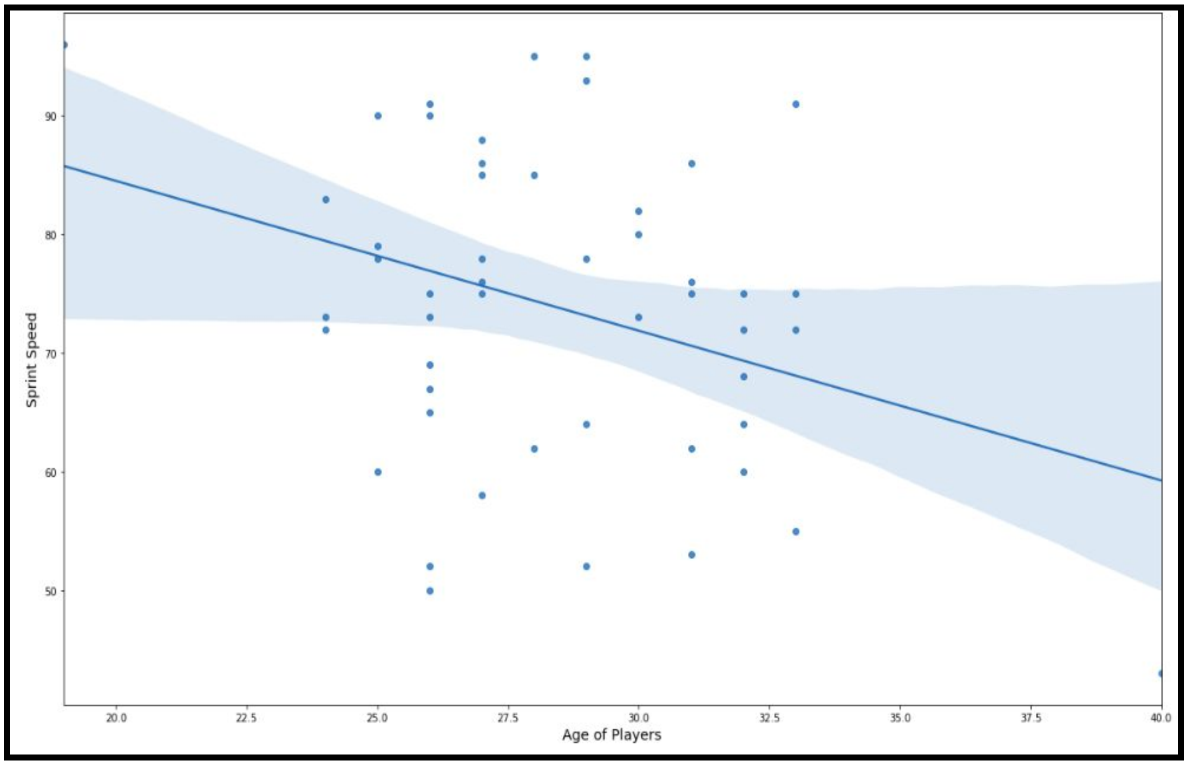
“SCATTER PLOT” for height and weight of the players of some 2000 players.



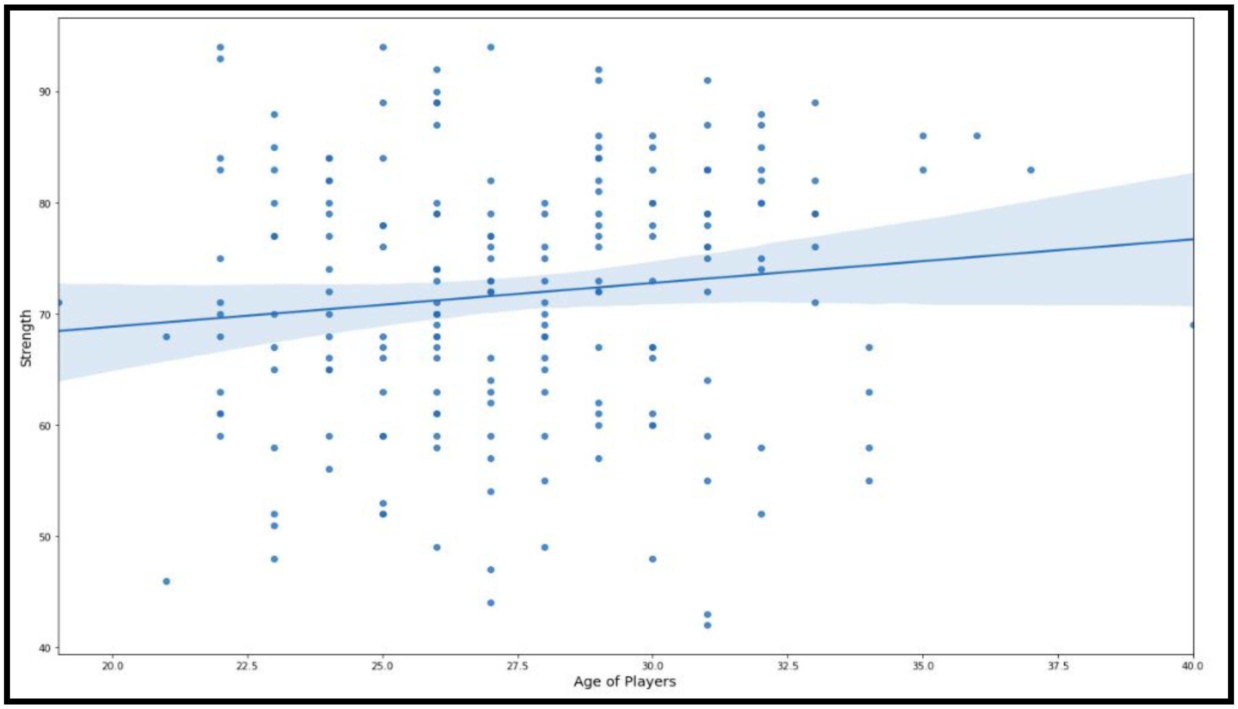
“LINE PLOT” to show how the majority of the players have a poor international reputation.



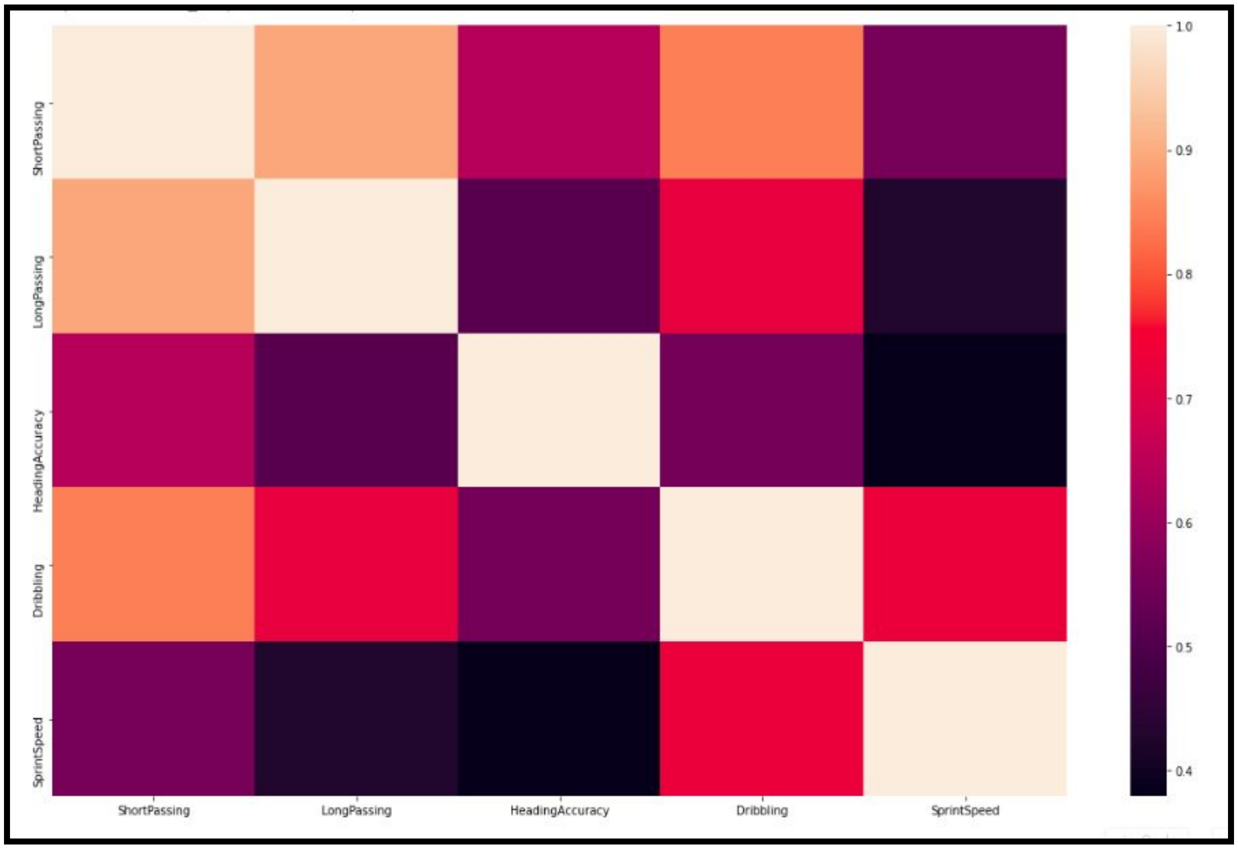
We plotted a **“REGPLOT”** for top 50 players which shows that the correlation between Vision and Age of players, with Vision generally **decreasing** as Age increases.



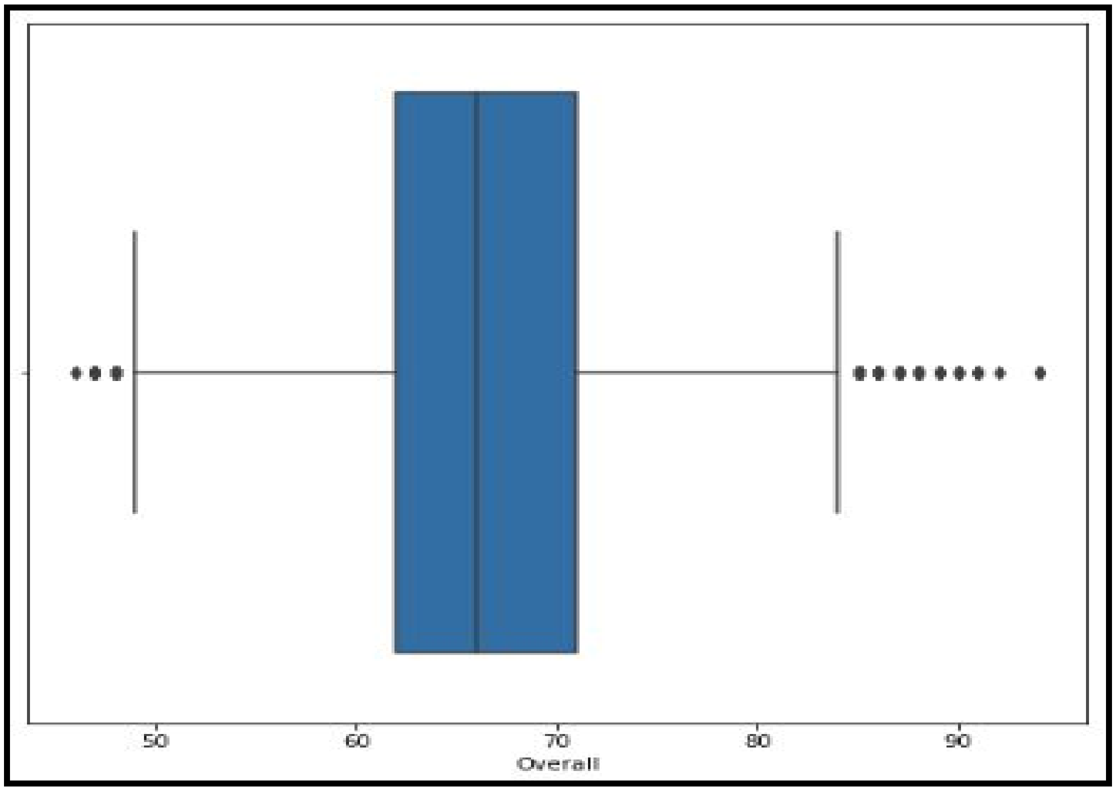
We plotted a **“REGPLOT”** for top 50 playerswhich shows that the correlation between SprintSpeed and Age of players, with SprintSpeed generally **decreasing** as Age increases.

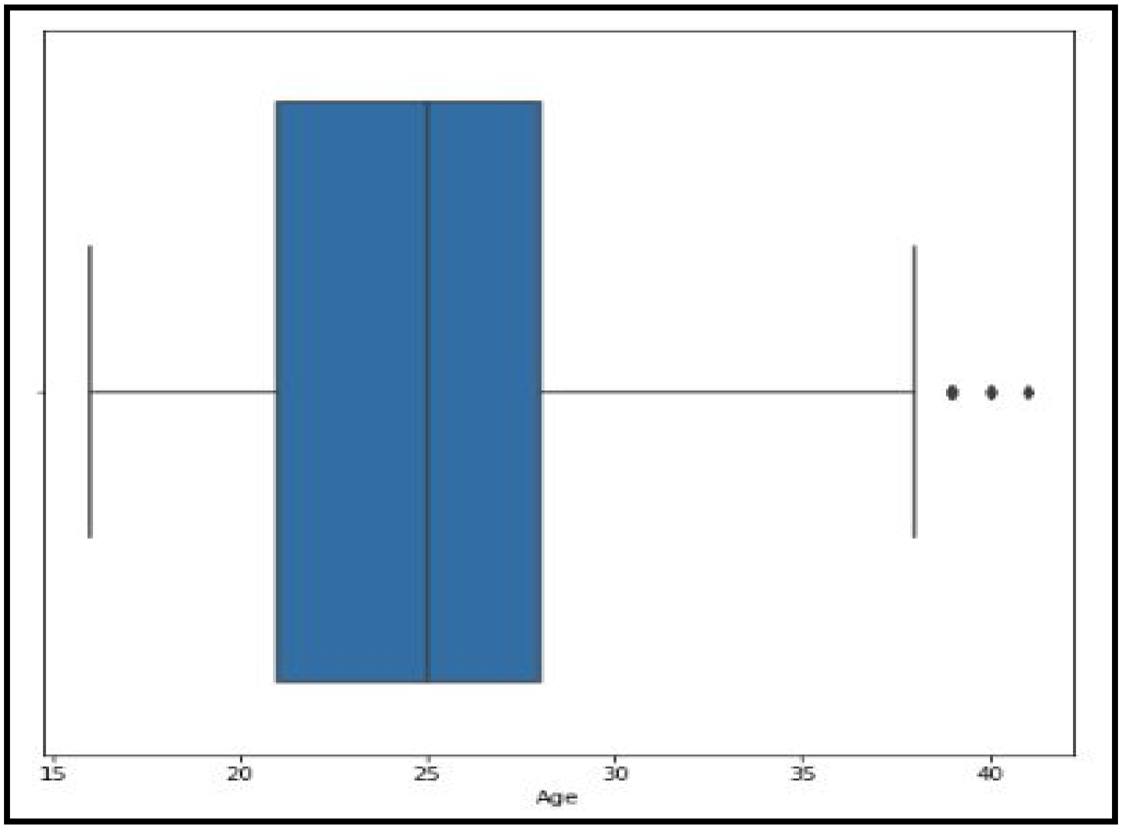


We plotted a **"REGPLOT"** for some 200 players which shows that the correlation between Strength and Age of players, with Strength generally **increasing** as Age increases.



"HEATMAP" is plotted to see the correlation between features like ShortPassing, LongPassing, HeadingAccuracy, Dribbling, SprintSpeed. Clearly, ShortPassing is very much related to LongPassing and Dribbling.





“BOXPLOT” of Age and Overall is plotted to find the mean and outlier players for both age and overall.

We see two graphs to find how players like Messi and Ronaldo are Outliers.

We notice that both overall rating and age of messi and ronaldo are in/near outlier class which tells us that even as their age is more, their overall rating is very high for their age.

TASK 2 K-Means Clustering Algorithm

About :

K-means algorithm is an iterative algorithm that tries to partition the dataset into *K* pre-defined distinct non-overlapping subgroups (clusters) where each data point belongs to only one group. It tries to make the intra-cluster data points as similar as possible while also keeping the clusters as different (far) as possible. It assigns data points to a cluster such that the sum of the squared distance between the data points and the cluster's centroid (arithmetic mean of all the data points that belong to that cluster) is at the minimum. The less variation we have within clusters, the more homogeneous (similar) the data points are within the same cluster.

The way k-means algorithm works is as follows:

1. Specify number of clusters *K*.
2. Initialize centroids by first shuffling the dataset and then randomly selecting *K* data points for the centroids without replacement.
3. Keep iterating until there is no change to the centroids. i.e assignment of data points to clusters isn't changing.
    - Compute the sum of the squared distance between data points and all centroids.
    - Assign each data point to the closest cluster (centroid).
    - Compute the centroids for the clusters by taking the average of the all data points that belong to each cluster.

Implementation :

The algorithm was implemented from scratch and the code is available in **2018101112_2018101088_clustering.ipynb** file.

Data used :

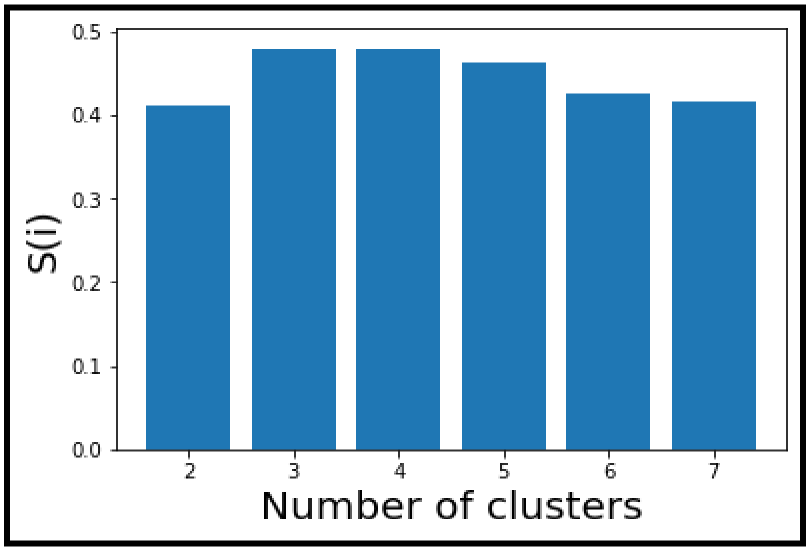
First the data was scaled using the **"StandardScaler"** function of the sklearn library. We scaled the data because K-Means is a distance based algorithm and is thus affected by the scale of the variables . So scaling is required to prevent the algorithm from being biased towards columns with higher magnitude data .

Then we further normalized the data using the "normalize" function of sklearn library . This was required as K-means clustering is "isotropic" in all directions of space and therefore tends to produce more or less round (rather than elongated) clusters. In this situation leaving variances unequal is equivalent to putting more weight on variables with smaller variance.

This normalised data was used as input data for the algorithm . We further did pca to reduce dimensionality of data to 2 but this was only done for the purpose of plotting the results in form of a scatter plot to help in analysis .Even to calculate inter and intra cluster similarity metrics and elbow and silhouette score the normalised data was used .

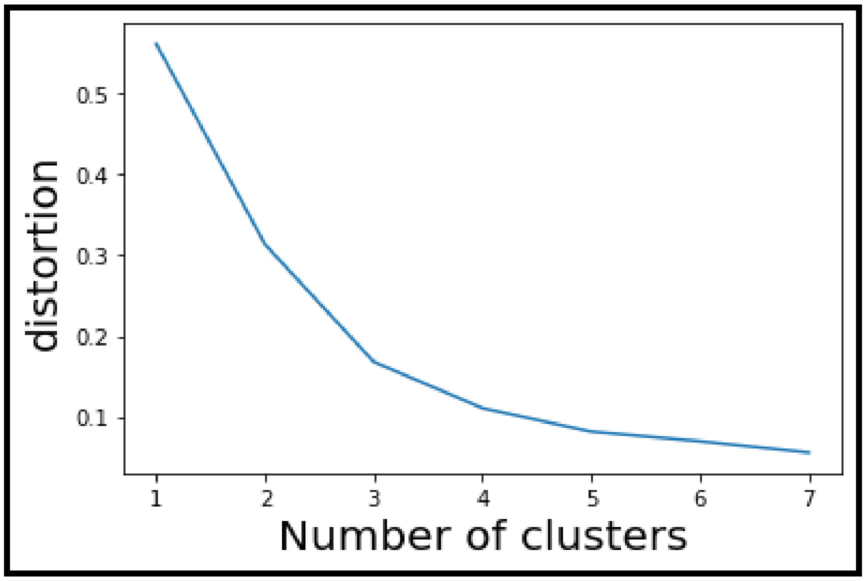
Results :

1. **Silhouette score for k = {2,3,4,5,6,7}**



   Average silhouette scores = [0.4117, 0.4792, 0.4792, 0.4614, 0.4252, 0.4163]

2. **Elbow method**

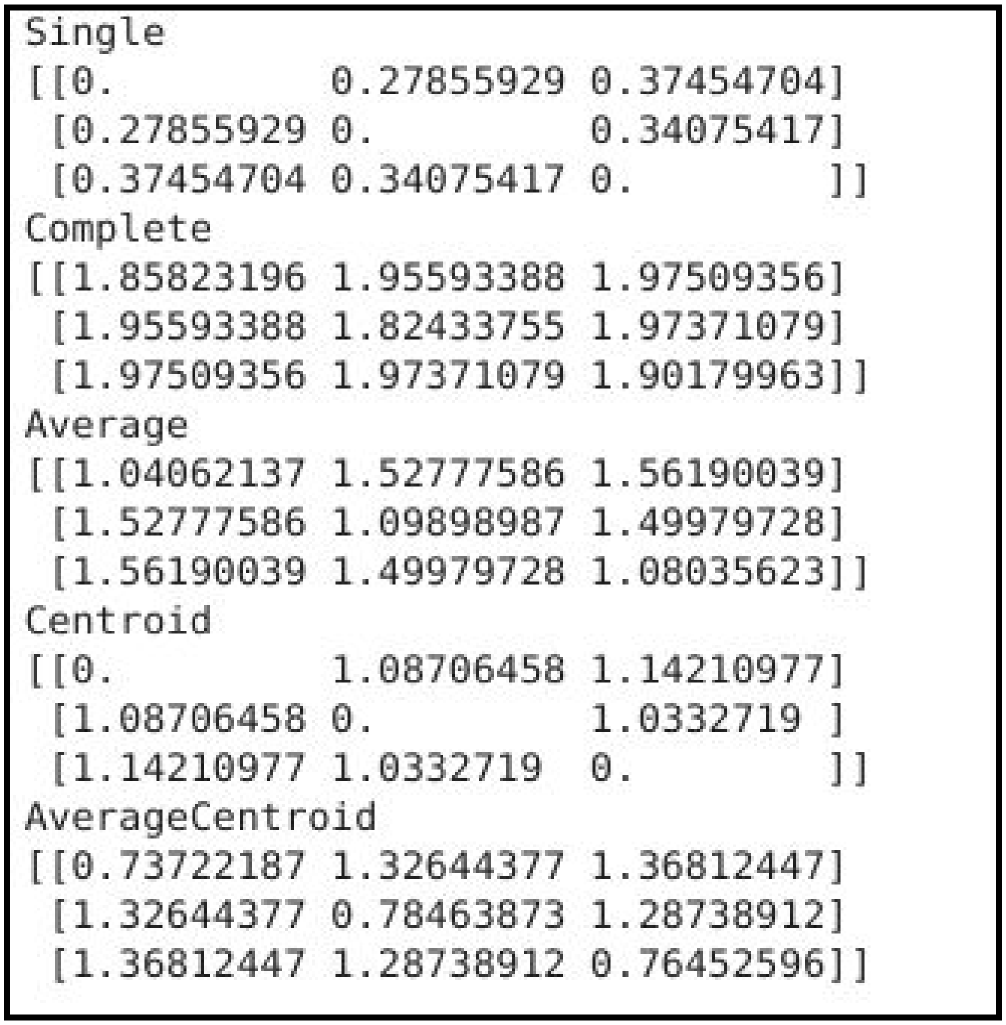


Distortion values = [0.5616, 0.3134, 0.1680, 0.1106, 0.0815, 0.0695, 0.0556]

From the elbow curve and average silhouette score it's clear that the best choice for **k is 3**. The reason being the fact that a higher silhouette score signifies that points are closer to their own cluster than to the neighbouring clusters. Also the drop in distortion values is most significant till k=3 after which its not too big hence elbow occurs at k = 3. Continuing the analysis further reveals the superiority of k = 3 over other values as mentioned below.

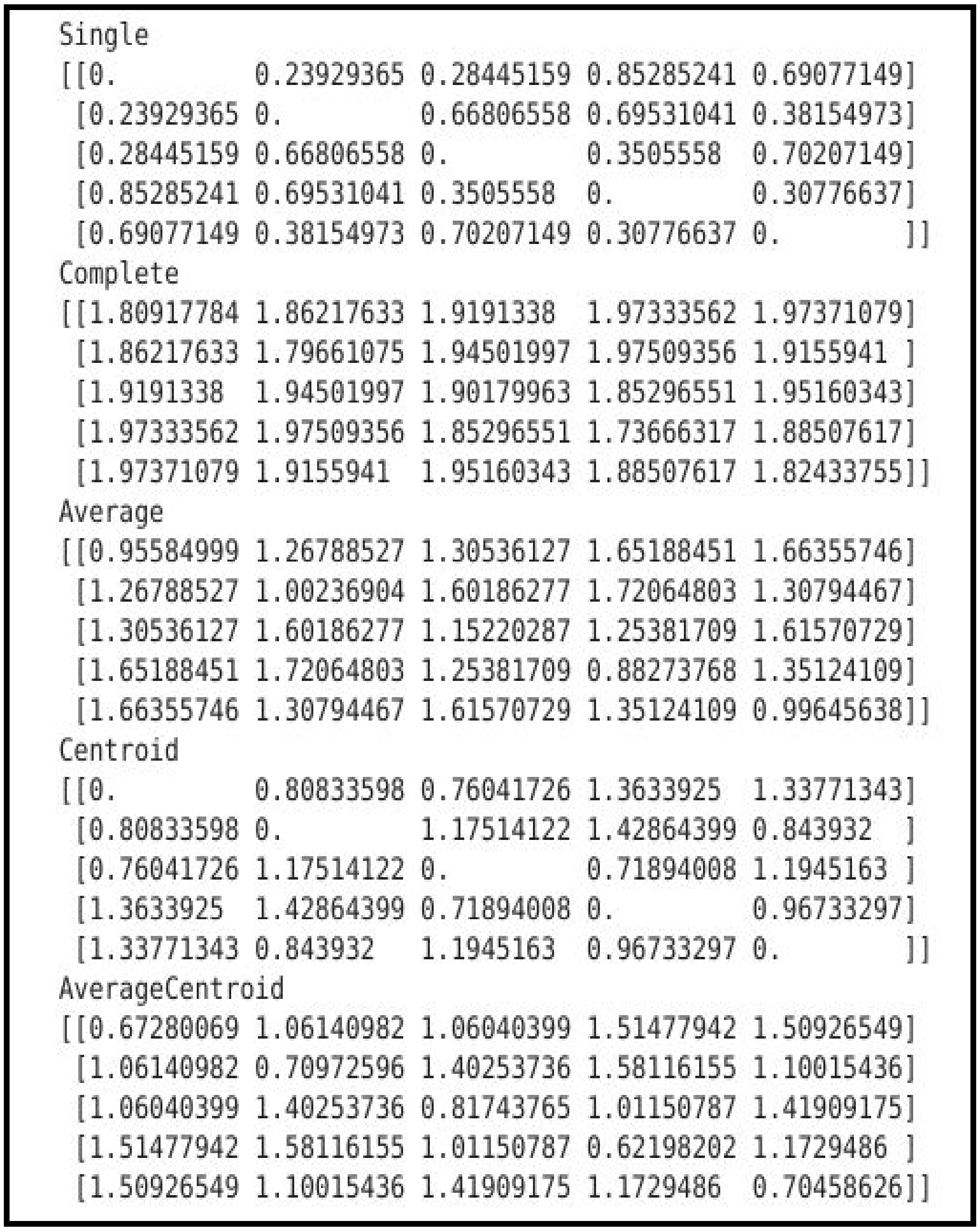
3. Following metrics were returned :

For k = 3



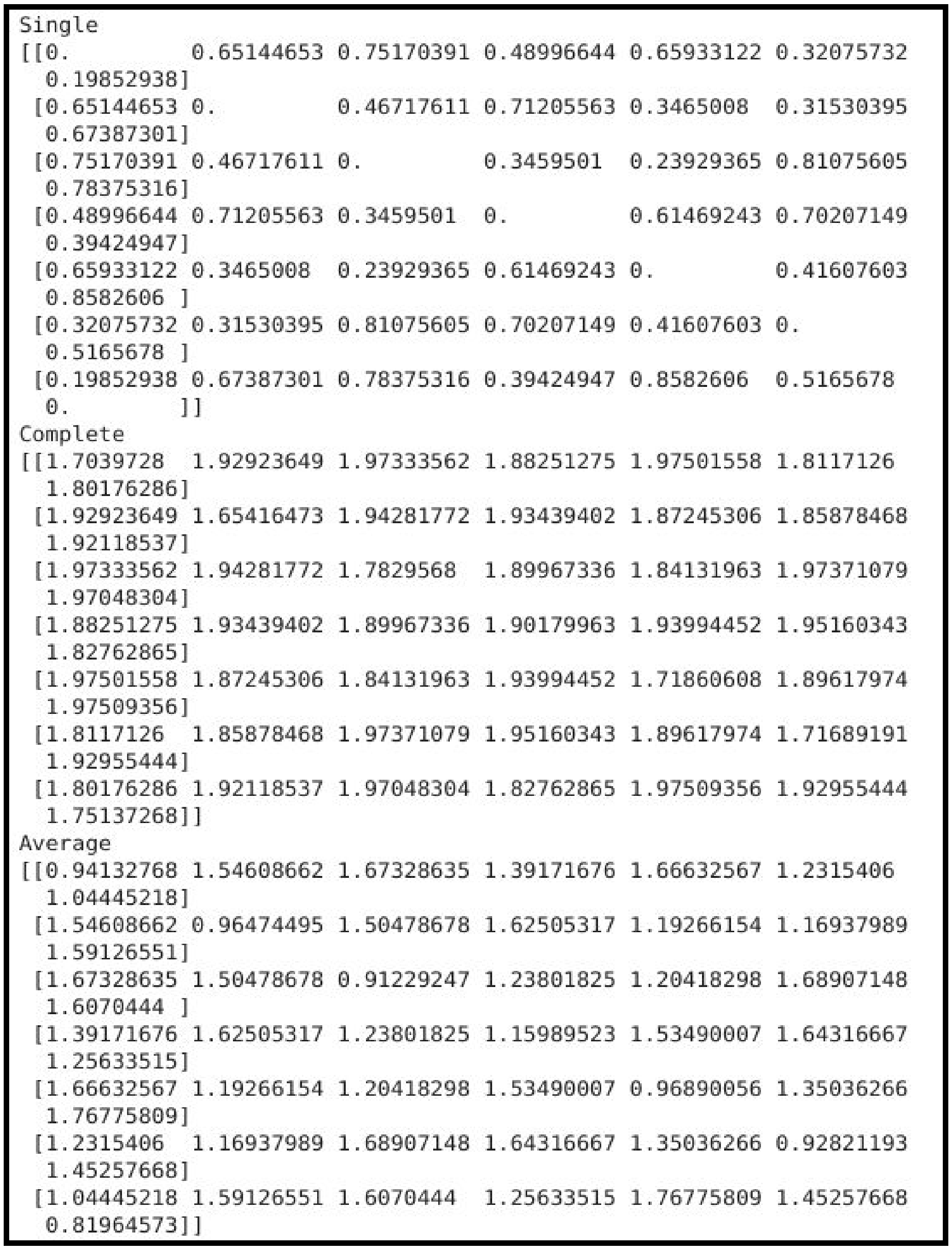
```
Single
[[0.         0.27855929 0.37454704]
 [0.27855929 0.         0.34075417]
 [0.37454704 0.34075417 0.        ]]
Complete
[[1.85823196 1.95593388 1.97509356]
 [1.95593388 1.82433755 1.97371079]
 [1.97509356 1.97371079 1.90179963]]
Average
[[1.04062137 1.52777586 1.56190039]
 [1.52777586 1.09898987 1.49979728]
 [1.56190039 1.49979728 1.08035623]]
Centroid
[[0.         1.08706458 1.14210977]
 [1.08706458 0.         1.0332719 ]
 [1.14210977 1.0332719  0.        ]]
AverageCentroid
[[0.73722187 1.32644377 1.36812447]
 [1.32644377 0.78463873 1.28738912]
 [1.36812447 1.28738912 0.76452596]]
```

For k = 5

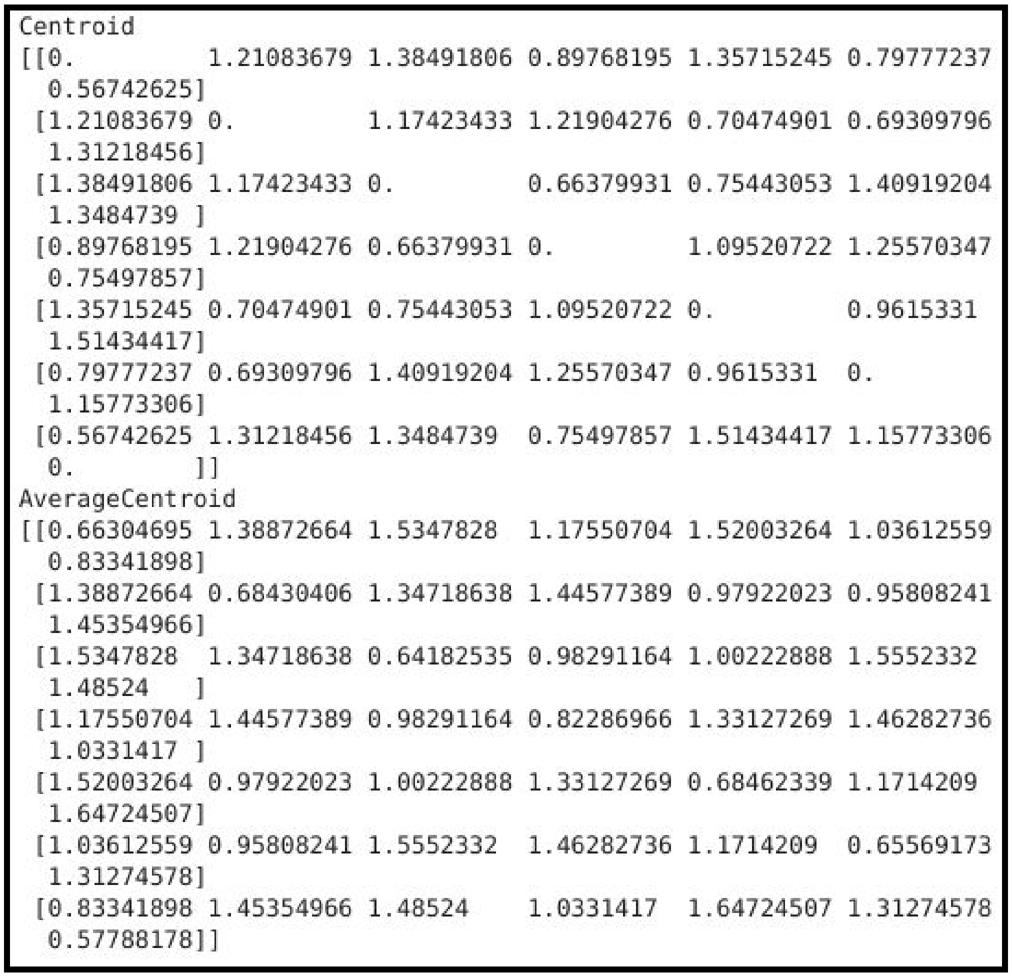


```
Single
[[0.         0.23929365 0.28445159 0.85285241 0.69077149]
 [0.23929365 0.         0.66806558 0.69531041 0.38154973]
 [0.28445159 0.66806558 0.         0.3505558  0.70207149]
 [0.85285241 0.69531041 0.3505558  0.         0.30776637]
 [0.69077149 0.38154973 0.70207149 0.30776637 0.        ]]
Complete
[[1.80917784 1.86217633 1.9191338  1.97333562 1.97371079]
 [1.86217633 1.79661075 1.94501997 1.97509356 1.9155941 ]
 [1.9191338  1.94501997 1.90179963 1.85296551 1.95160343]
 [1.97333562 1.97509356 1.85296551 1.73666317 1.88507617]
 [1.97371079 1.9155941  1.95160343 1.88507617 1.82433755]]
Average
[[0.95584999 1.26788527 1.30536127 1.65188451 1.66355746]
 [1.26788527 1.00236904 1.60186277 1.72064803 1.30794467]
 [1.30536127 1.60186277 1.15220287 1.25381709 1.61570729]
 [1.65188451 1.72064803 1.25381709 0.88273768 1.35124109]
 [1.66355746 1.30794467 1.61570729 1.35124109 0.99645638]]
Centroid
[[0.         0.80833598 0.76041726 1.3633925  1.33771343]
 [0.80833598 0.         1.17514122 1.42864399 0.843932  ]
 [0.76041726 1.17514122 0.         0.71894008 1.1945163 ]
 [1.3633925  1.42864399 0.71894008 0.         0.96733297]
 [1.33771343 0.843932   1.1945163  0.96733297 0.        ]]
AverageCentroid
[[0.67280069 1.06140982 1.06040399 1.51477942 1.50926549]
 [1.06140982 0.70972596 1.40253736 1.58116155 1.10015436]
 [1.06040399 1.40253736 0.81743765 1.01150787 1.41909175]
 [1.51477942 1.58116155 1.01150787 0.62198202 1.1729486 ]
 [1.50926549 1.10015436 1.41909175 1.1729486  0.70458626]]
```

For k = 7



```
Single
[[0.         0.65144653 0.75170391 0.48996644 0.65933122 0.32075732
  0.19852938]
 [0.65144653 0.         0.46717611 0.71205563 0.3465008  0.31530395
  0.67387301]
 [0.75170391 0.46717611 0.         0.3459501  0.23929365 0.81075605
  0.78375316]
 [0.48996644 0.71205563 0.3459501  0.         0.61469243 0.70207149
  0.39424947]
 [0.65933122 0.3465008  0.23929365 0.61469243 0.         0.41607603
  0.8582606 ]
 [0.32075732 0.31530395 0.81075605 0.70207149 0.41607603 0.
  0.5165678 ]
 [0.19852938 0.67387301 0.78375316 0.39424947 0.8582606  0.5165678
  0.        ]]
Complete
[[1.7039728  1.92923649 1.97333562 1.88251275 1.97501558 1.8117126
  1.80176286]
 [1.92923649 1.65416473 1.94281772 1.93439402 1.87245306 1.85878468
  1.92118537]
 [1.97333562 1.94281772 1.7829568  1.89967336 1.84131963 1.97371079
  1.97048304]
 [1.88251275 1.93439402 1.89967336 1.90179963 1.93994452 1.95160343
  1.82762865]
 [1.97501558 1.87245306 1.84131963 1.93994452 1.71860608 1.89617974
  1.97509356]
 [1.8117126  1.85878468 1.97371079 1.95160343 1.89617974 1.71689191
  1.92955444]
 [1.80176286 1.92118537 1.97048304 1.82762865 1.97509356 1.92955444
  1.75137268]]
Average
[[0.94132768 1.54608662 1.67328635 1.39171676 1.66632567 1.2315406
  1.04445218]
 [1.54608662 0.96474495 1.50478678 1.62505317 1.19266154 1.16937989
  1.59126551]
 [1.67328635 1.50478678 0.91229247 1.23801825 1.20418298 1.68907148
  1.6070444 ]
 [1.39171676 1.62505317 1.23801825 1.15989523 1.53490007 1.64316667
  1.25633515]
 [1.66632567 1.19266154 1.20418298 1.53490007 0.96890056 1.35036266
  1.76775809]
 [1.2315406  1.16937989 1.68907148 1.64316667 1.35036266 0.92821193
  1.45257668]
 [1.04445218 1.59126551 1.6070444  1.25633515 1.76775809 1.45257668
  0.81964573]]
```



```
Centroid
[[0.         1.21083679 1.38491806 0.89768195 1.35715245 0.79777237
  0.56742625]
 [1.21083679 0.         1.17423433 1.21904276 0.70474901 0.69309796
  1.31218456]
 [1.38491806 1.17423433 0.         0.66379931 0.75443053 1.40919204
  1.3484739 ]
 [0.89768195 1.21904276 0.66379931 0.         1.09520722 1.25570347
  0.75497857]
 [1.35715245 0.70474901 0.75443053 1.09520722 0.         0.9615331
  1.51434417]
 [0.79777237 0.69309796 1.40919204 1.25570347 0.9615331  0.
  1.15773306]
 [0.56742625 1.31218456 1.3484739  0.75497857 1.51434417 1.15773306
  0.        ]]
AverageCentroid
[[0.66304695 1.38872664 1.5347828  1.17550704 1.52003264 1.03612559
  0.83341898]
 [1.38872664 0.68430406 1.34718638 1.44577389 0.97922023 0.95808241
  1.45354966]
 [1.5347828  1.34718638 0.64182535 0.98291164 1.00222888 1.5552332
  1.48524   ]
 [1.17550704 1.44577389 0.98291164 0.82286966 1.33127269 1.46282736
  1.0331417 ]
 [1.52003264 0.97922023 1.00222888 1.33127269 0.68462339 1.1714209
  1.64724507]
 [1.03612559 0.95808241 1.5552332  1.46282736 1.1714209  0.65569173
  1.31274578]
 [0.83341898 1.45354966 1.48524    1.0331417  1.64724507 1.31274578
  0.57788178]]
```

- <u>**Inter Cluster Analysis (Non diagonal entries of the above matrices)**</u>
  Intercluster distance is the distance between two objects belonging to two different clusters. It is of 5 types –

  By Analysis we find that K = 3 is the ideal choice because of the following reasons :

  1. **Single Linkage Distance** : The single linkage distance is the closest distance between two objects belonging to two different clusters defined as –

     $$\delta_1(S, T) = \min \left\{ \begin{array}{c} d(x, y) \\ x \in S, y \in T \end{array} \right\}$$

     Thus according to the definition, the closest distance between two points in different clusters was maximum for **K = 3**. This implies that on increasing the value of K, there were points in different clusters which had very less distance between them.

2. **Complete Linkage Distance** : The complete linkage distance is the distance between two most remote objects belonging to two different clusters defined as –

$$\boxed{\delta_2(S,T) = \max\left\{ \underset{x \in S, y \in T}{d(x,y)} \right\}}$$

The minimum value of the maximum distance between any two points in different clusters was maximum for K = 3. On increasing the value of K, the maximum distance between any top points belonging to different clusters remained almost the same, however the minimum value of this same metric decreased as K increased which implies that certain clusters came closer to each other which is not ideal for clustering.

3. **Average Linkage Distance** : The average linkage distance is the average distance between all the objects belonging to two different clusters defined as –

$$\boxed{\delta_3(S,T) = \frac{1}{|S|\,|T|} \sum_{\substack{x \in S \\ y \in T}} d(x,y)}$$

There is a significant drop in the minimum Average Linkage Value as we increase the value of K from 3 to 5. Hence, it is not ideal for clustering because this drop implies that the clusters come very close to each other.

4. **Centroid Linkage Distance** : The centroid linkage distance is the distance between the centers vs and vt of two clusters S and T respectively, defined as –

$\boxed{\delta_4(S,T) = d(v_s, v_t)}$ where, $\boxed{v_s = \frac{1}{|S|} \sum_{x \in S} x, \; v_t = \frac{1}{|T|} \sum_{y \in T} y}$

On increasing the value of K, the maximum Centroid Linkage Distance also increases which is an ideal case for clustering, however this happens at the cost of the minimum value of this same metric, which implies that although there was a pair of cluster which were further away from each other, but at the same time there were clusters which came closer to each other which is not an ideal case.

5. **Average Centroid Linkage Distance** : The average centroid linkage distance is the distance between the center of a cluster and all the objects belonging to a different cluster, defined as –

$$\delta_5(S,T) = \frac{1}{|S|+|T|}\left\{ \sum_{x \in S} d(x, v_T) + \sum_{y \in T} d(y, v_S) \right\}$$

   This supports our explanation on the Centroid Linkage Distance. The minimum value of the Average Centroid Linkage Distance decreases more significantly as compared to what the expected increase in maximum value of this metric should be. This further justifies previous metric as even the Average Centroid linkage distance shows a similar trend.

- **<u>Intra Cluster (Entries on the diagonal of the above matrices)</u>**

  Intracluster distance is the distance between two objects belonging to the same cluster. It is of 3 types –

1. **Complete Diameter Distance :** The complete diameter distance is the distance between two most remote objects belonging to the same cluster defined as –

$$\Delta_1(S) = \max_{x,y \in S} \{d(x,y)\}$$

2. **Average Diameter Distance :** The average diameter distance is the average distance between all the objects belonging to the same cluster defined as –

$$\Delta_2(S) = \frac{1}{|S| \cdot (|S|-1)} \sum_{\substack{x,y \in S \\ x \neq y}} \{d(x,y)\}$$

3. **Centroid Diameter Distance :** The centroid diameter distance is double average distance between all of the objects and the cluster center of s defined as –

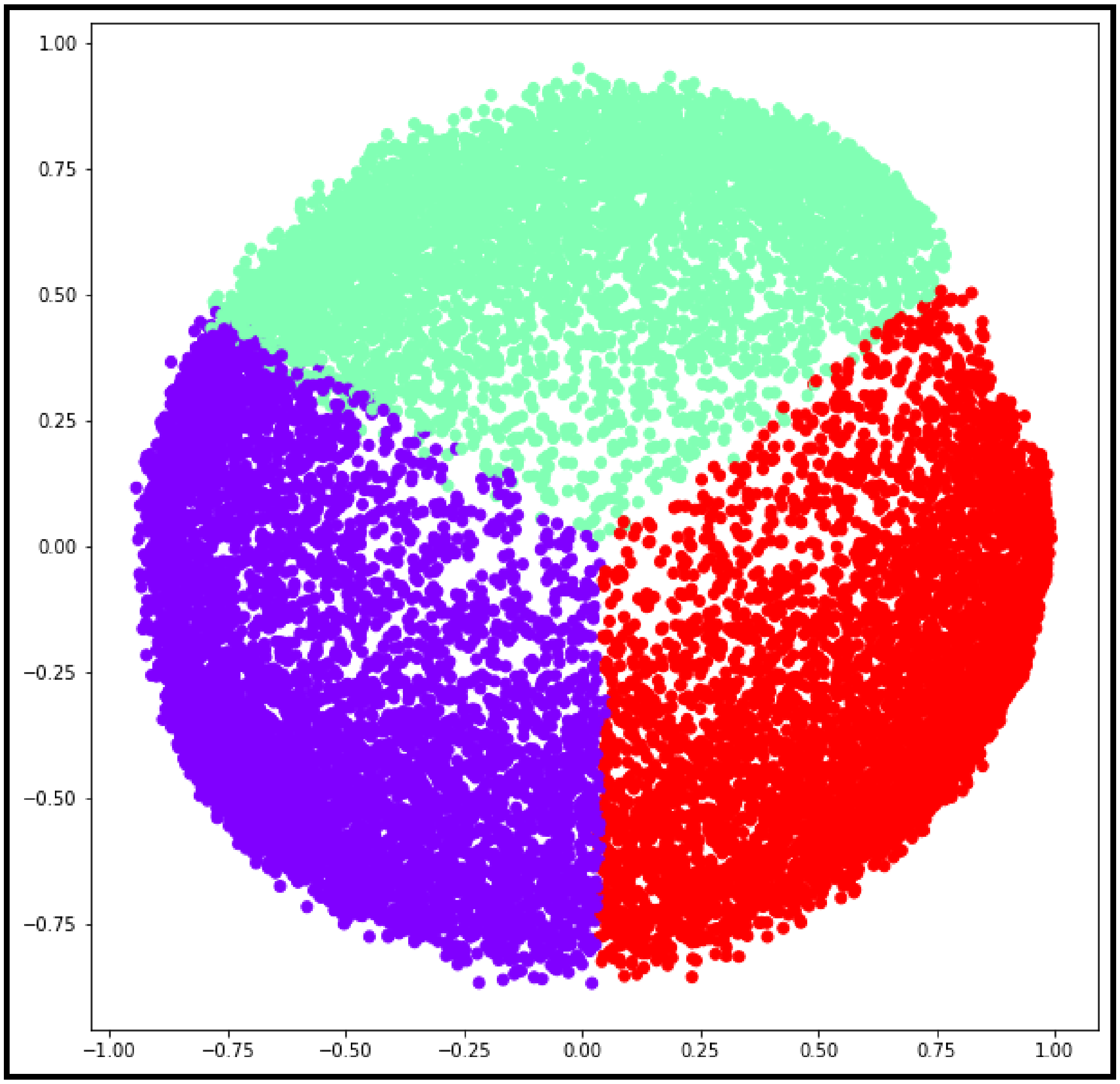
$$\Delta_3(S) = 2\left\{ \frac{\sum_{x \in S} d(x, \overline{v})}{|S|} \right\}$$ where, $$\overline{v} = \frac{1}{|S|} \sum_{x \in S} x$$

For all the three metrics explained above (**Complete Diameter distance, Average Diameter distance,Centroid Diameter distance**) the distance increases between the minimum and maximum value of each metric. A smaller difference implies similar size clusters, whereas with increasing k, the difference increases which implies uneven sized clusters.

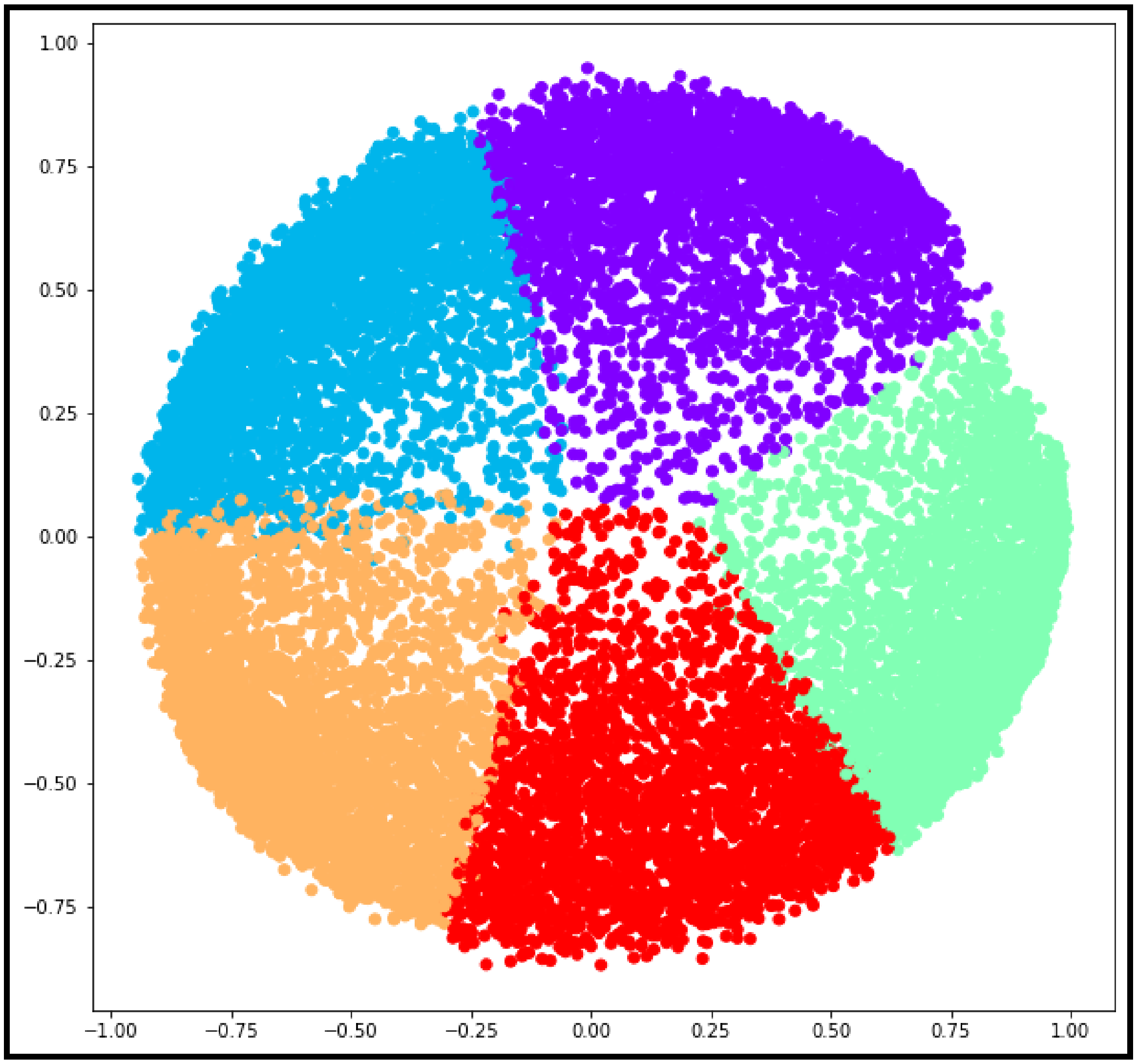
Thus from the analysis above we can further imply that K=3 is the ideal value for K means Clustering for this dataset.

4. The following scatter plots were obtained when we plotted the results of algorithm ie different colour represent different cluster and 71 dimensional data point converted to 2 dimensional (represented by the 2 axis of the graph)

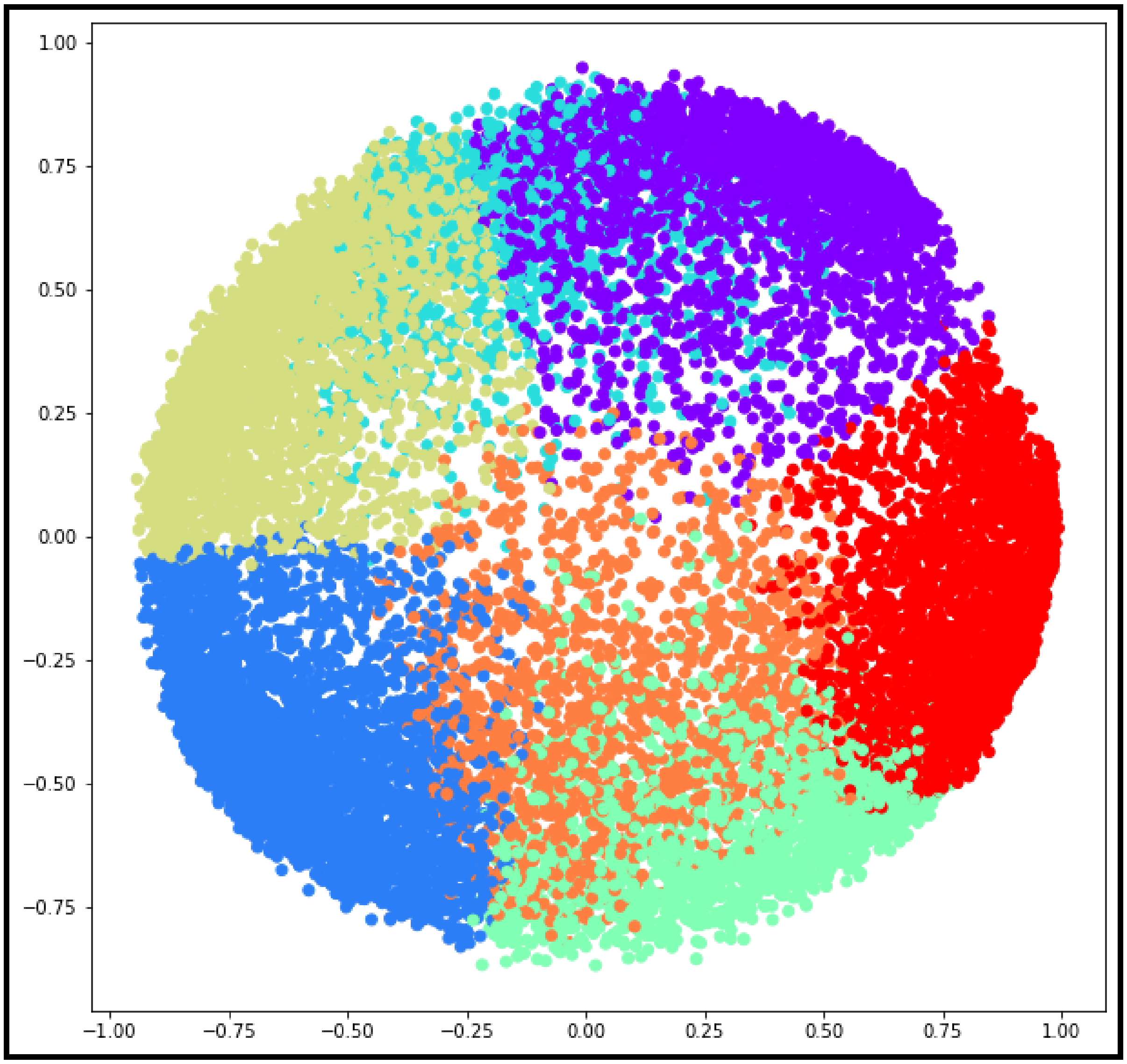
For K = 3



For K = 5


1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
−0.75
−0.50
−0.25
0.00
0.25
0.50
0.75
1.00

For K = 7



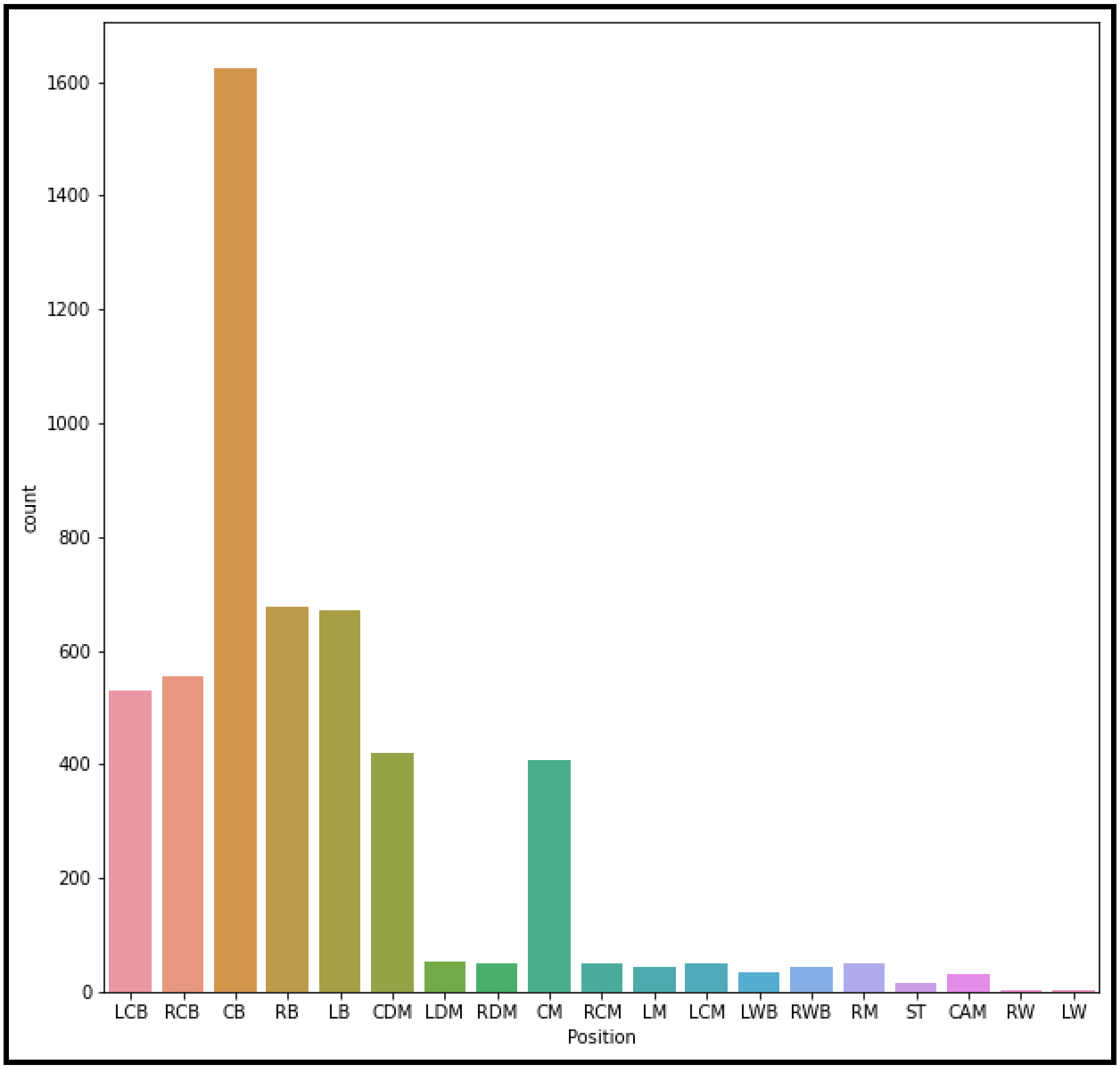
The above plots clearly show how the quality of clusters formed is better for k=3 as compared to k=5 and k=7 .

Also for k=3 the data points were divided almost equally ie(almost 5000 in each cluster)

We marked the cluster for this case in the following manner .

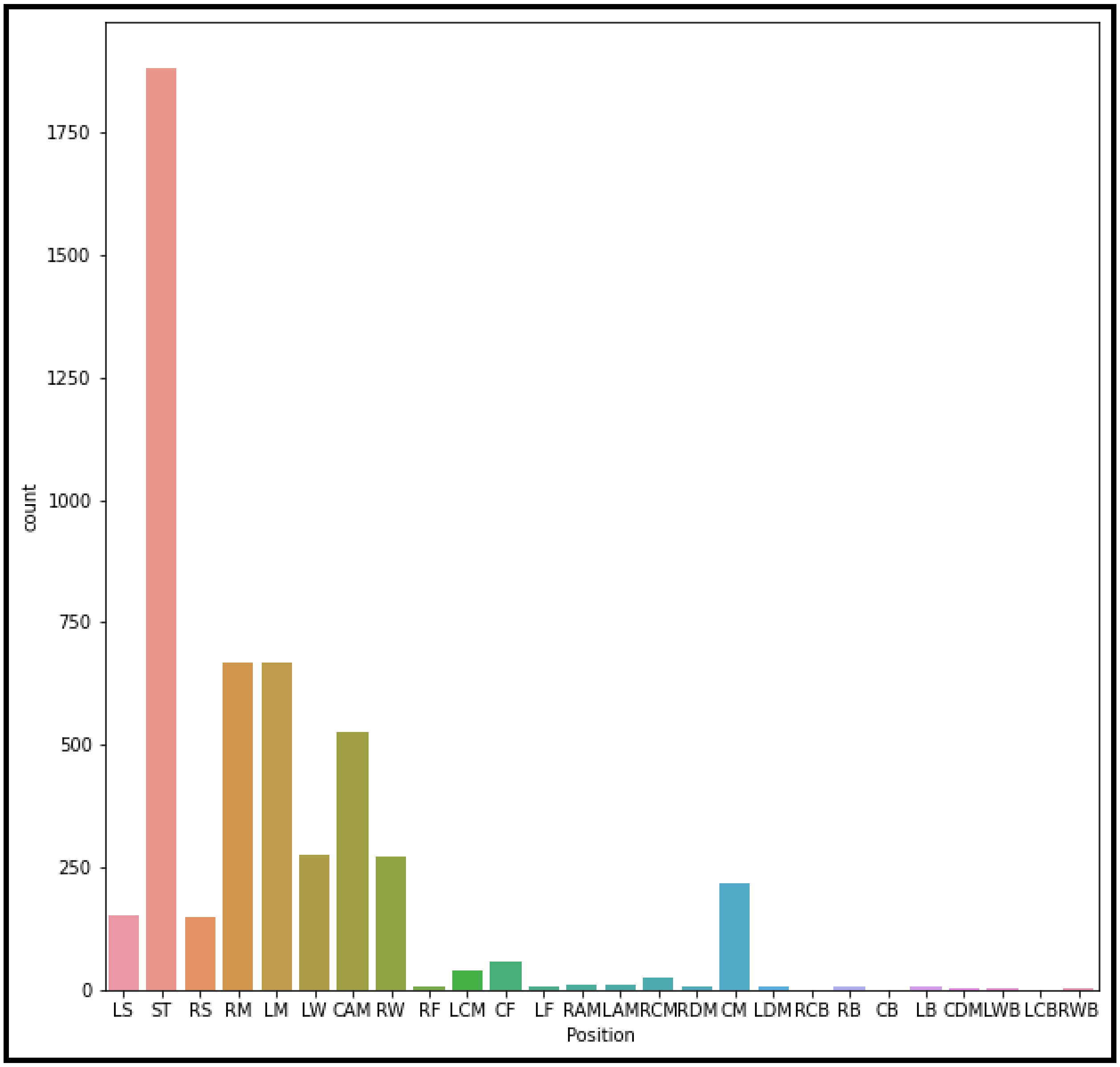
Cluster 1 :

Positions belonging to and near CB(centre back) have been captured.



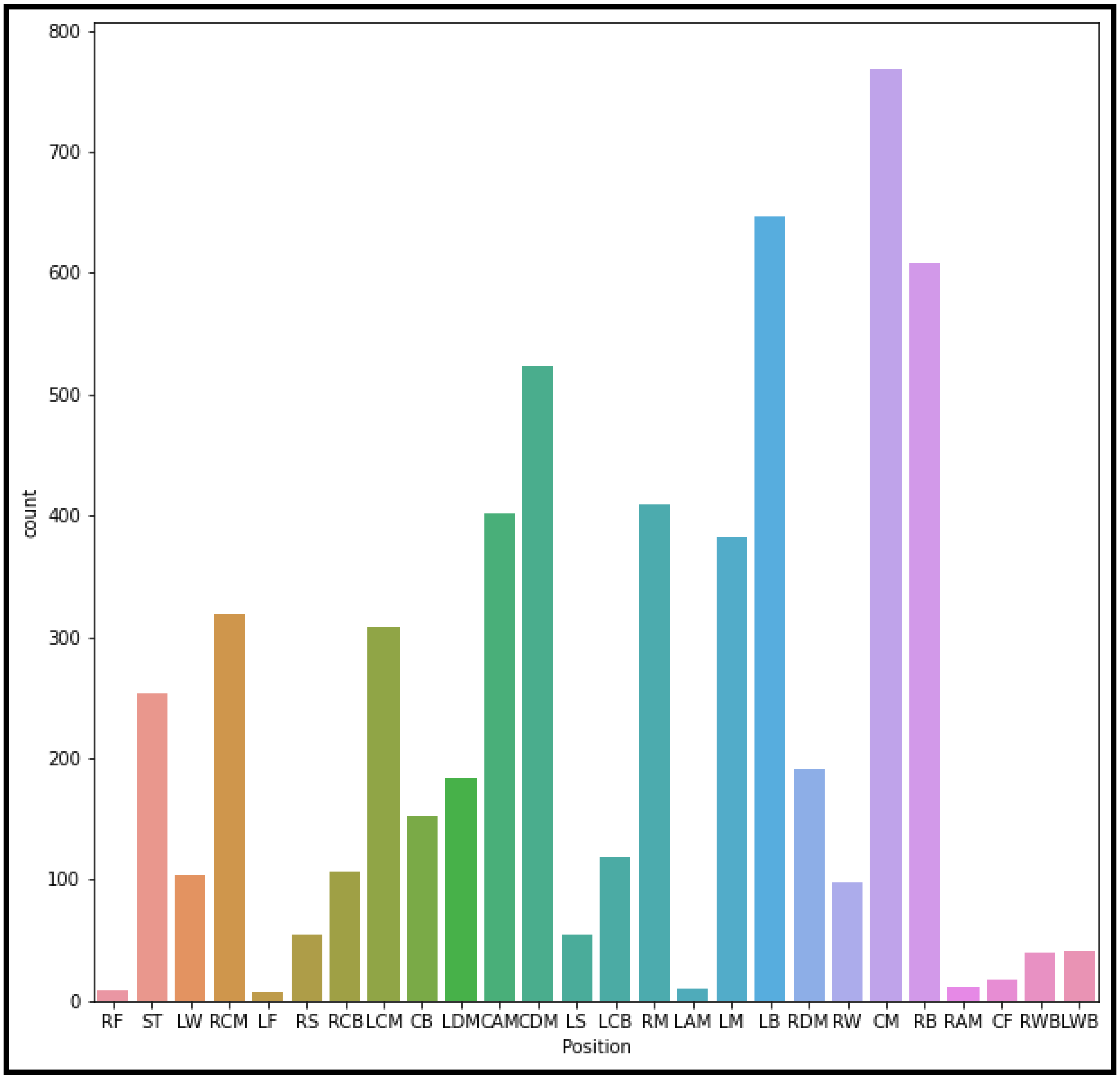
Cluster 2 :

Positions belonging to and near ST(Striker) have been captured.



Cluster 3 :

Positions belonging to and near CDM(central defensive midfielder) have been captured.



So Roughly 3 clusters have captured players belonging to 3 different regions (positions) of the field.

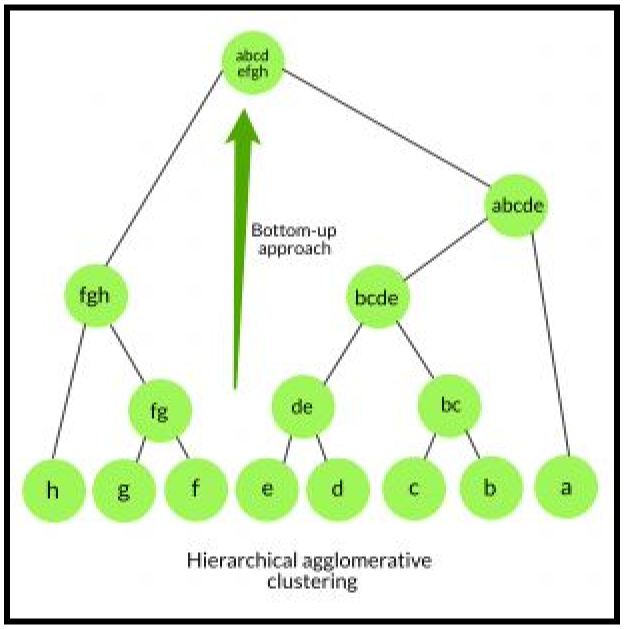
TASK 3 Hierarchical Clustering Algorithm

In data mining and statistics, hierarchical clustering analysis is a method of cluster analysis which seeks to build a hierarchy of clusters i.e. tree type structure based on the hierarchy.

1. Agglomerative Clustering (Bottom-Up Strategy)

About :

Also known as bottom-up approach or hierarchical agglomerative clustering (HAC). A structure that is more informative than the unstructured set of clusters returned by flat clustering. This clustering algorithm does not require us to pre-specify the number of clusters. Bottom-up algorithms treat each data as a singleton cluster at the outset and then successively agglomerates pairs of clusters until all clusters have been merged into a single cluster that contains all data.

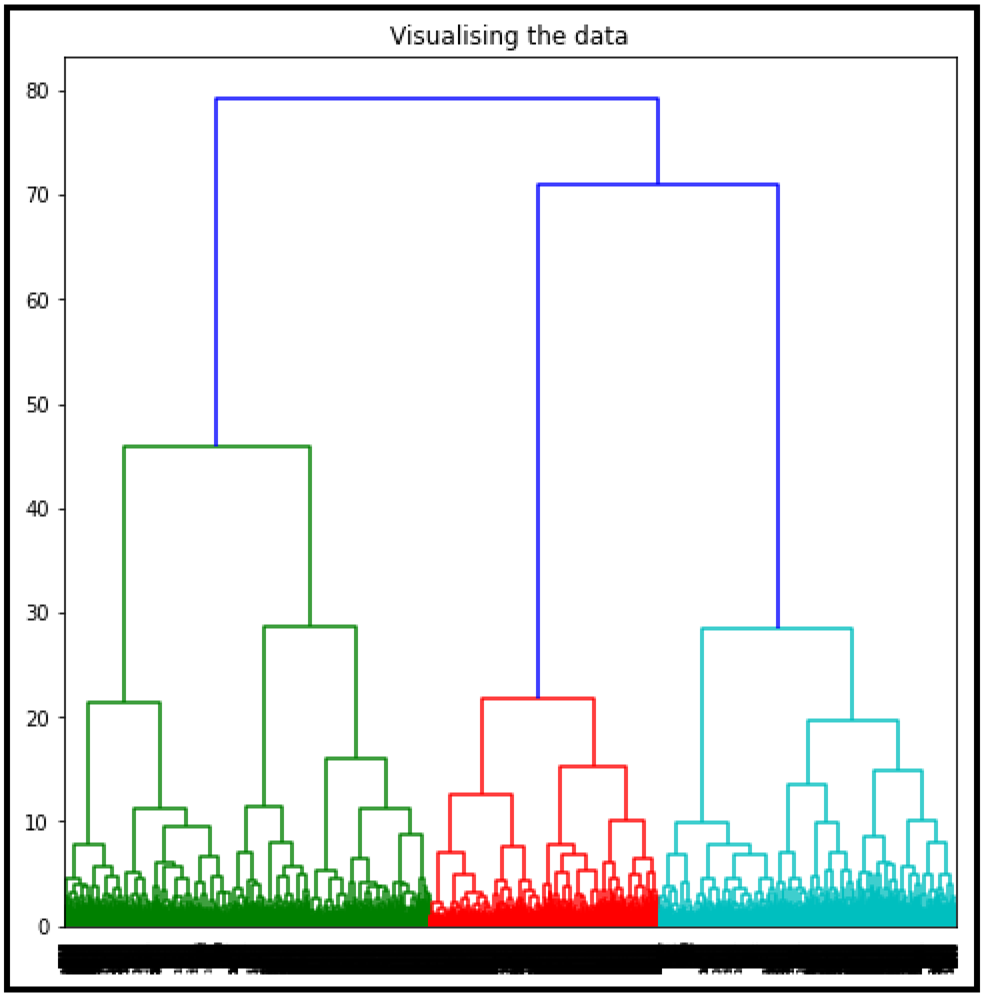


Implementation :

We have used inbuilt library functions (AgglomerativeClustering) to implement the algorithm. Data is first scaled and then normalized and used as input for clustering. PCA has only been used for dimensionality reduction to allow us to visualize points on a coordinate plane and analyze the clusters.

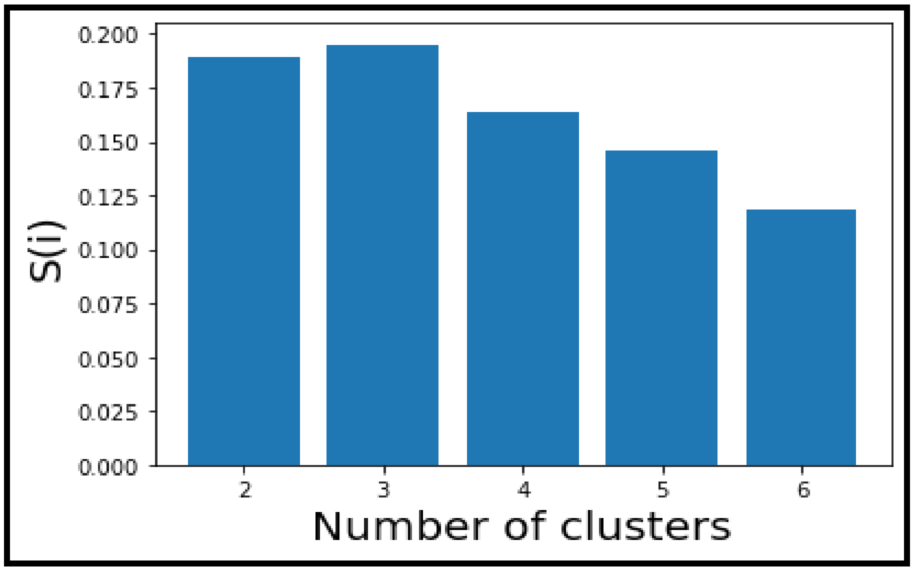
Scipy.cluster.hierarchy library is used to make the linkage matrix and to make the dendrogram. There are various linkage methods to build this matrix, the ones which are mostly used are ; complete, ward,average.

We have used ward as it gave the maximum Cophenetic Correlation Coefficient (The closer the value is to 1, the better the clustering preserves the original distances) out of the three and also minimizes the variance of clusters being merged . We obtained the dendrogram shown below :



We did agglomerative clustering with a number of clusters = {2,3,4,5,6} and plotted the scatter plots , distance metrics and silhouette scores as given below .

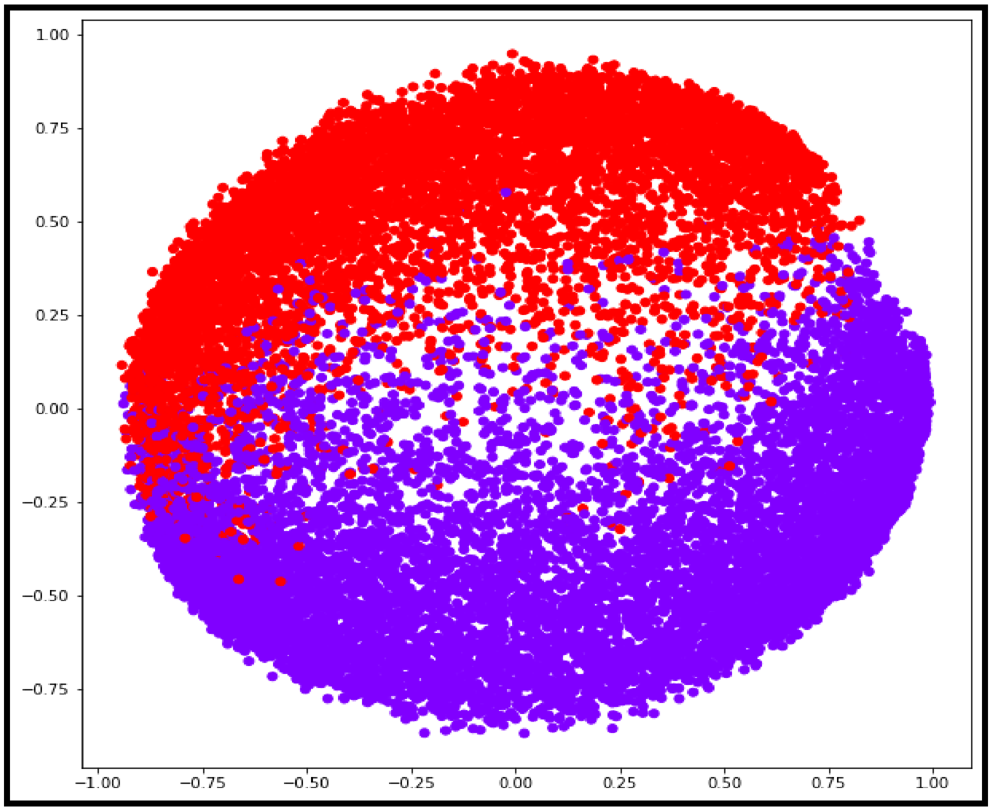
Silhouette scores:



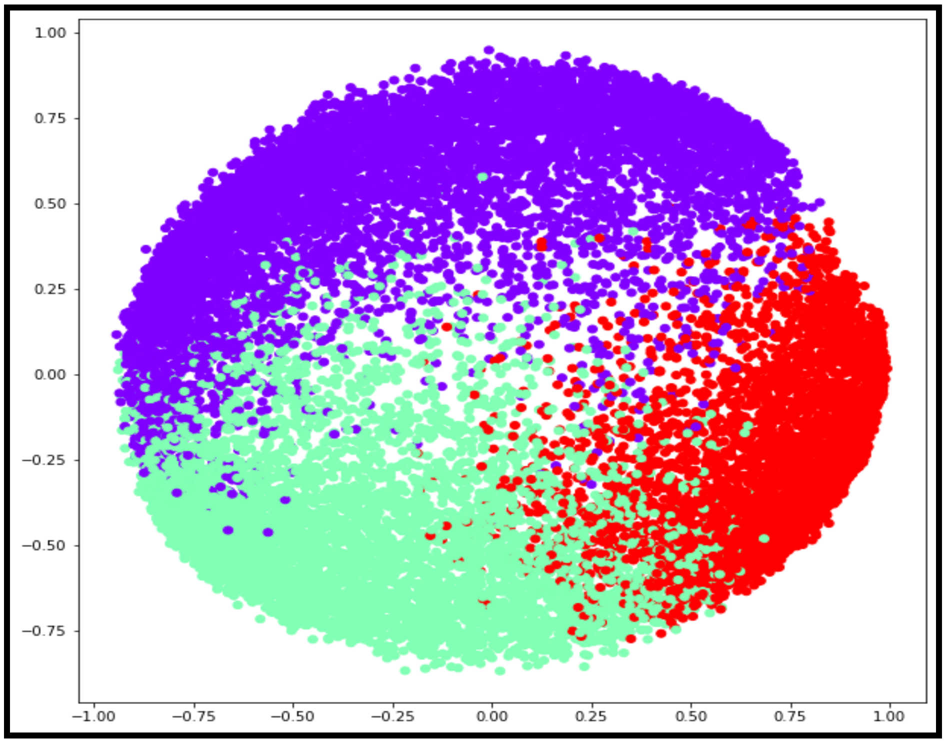
S(i) = [0.18872681807399358, 0.1950761194328005, 0.1632769909335091, 0.14539390172725675, 0.11892892669258613]

The scores suggest 3 to be the best choice for the number of clusters. This is also evident in the following scatter plots.

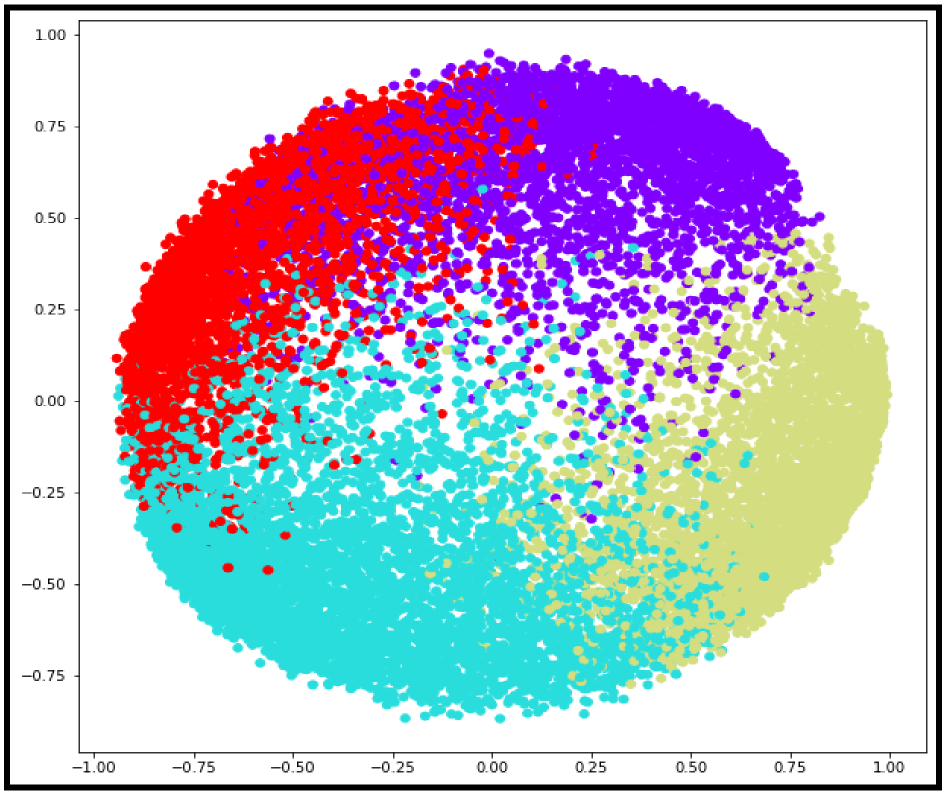
Number of clusters = 2



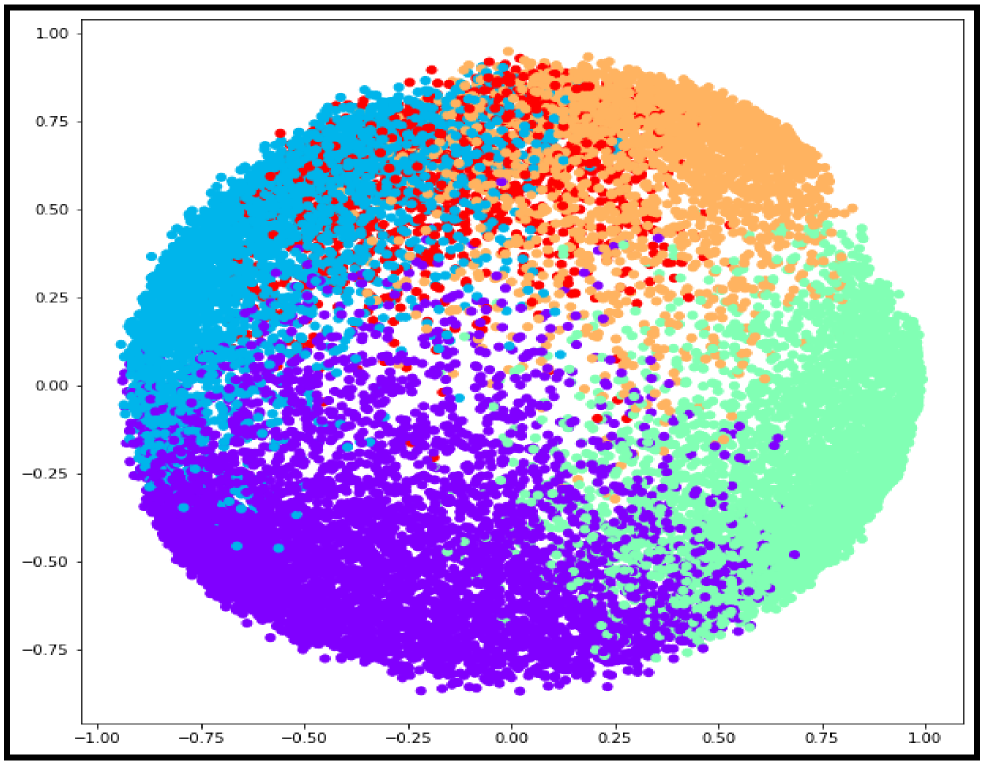
Number of clusters = 3



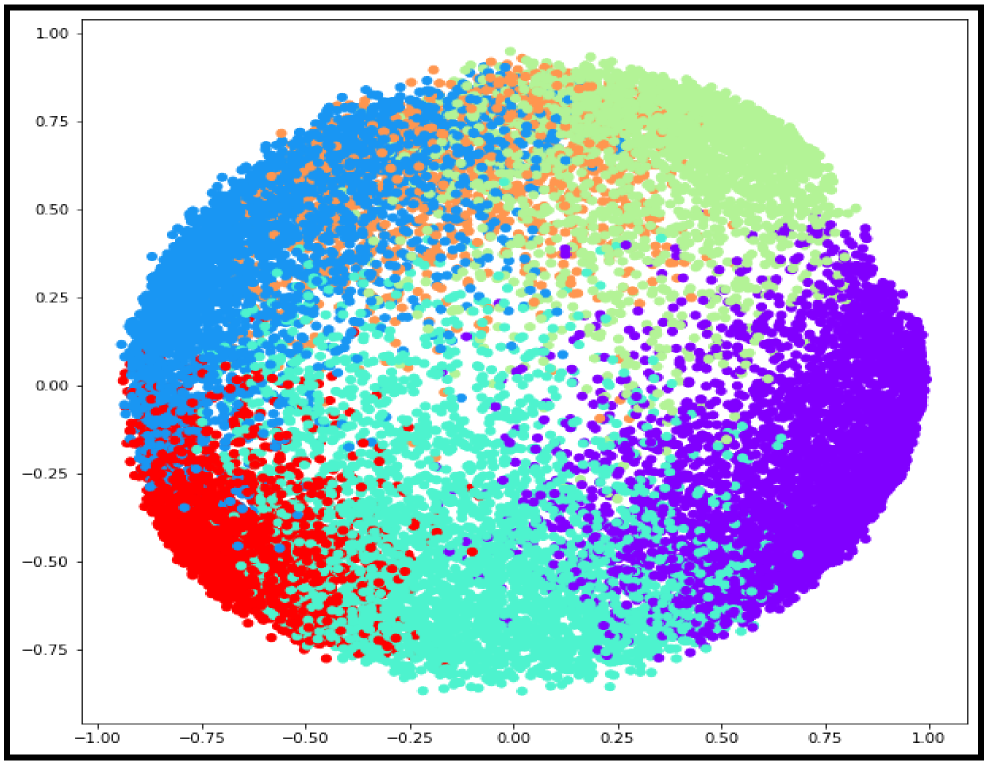
Number of clusters = 4



Number of clusters = 5



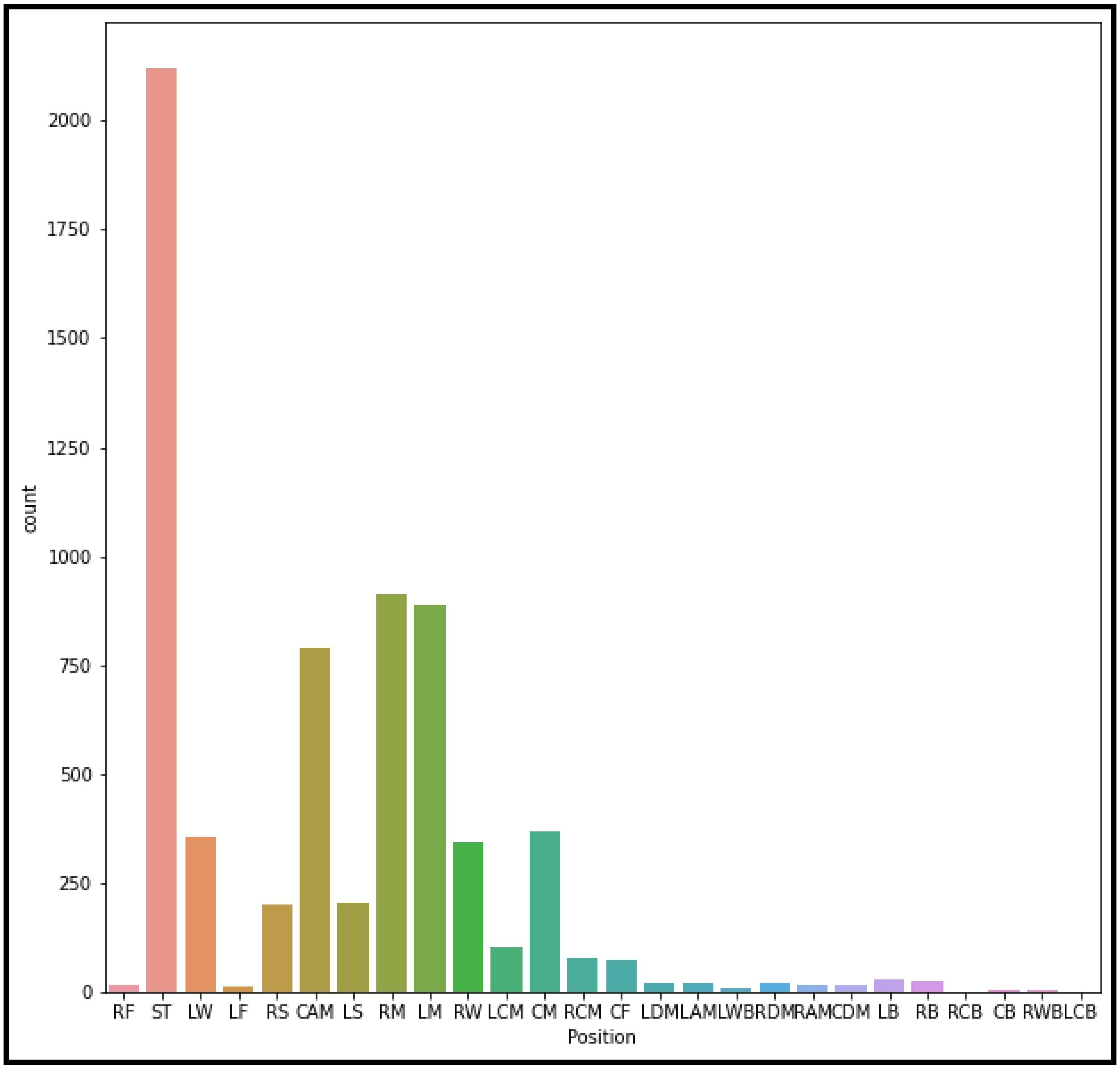
Number of clusters = 6



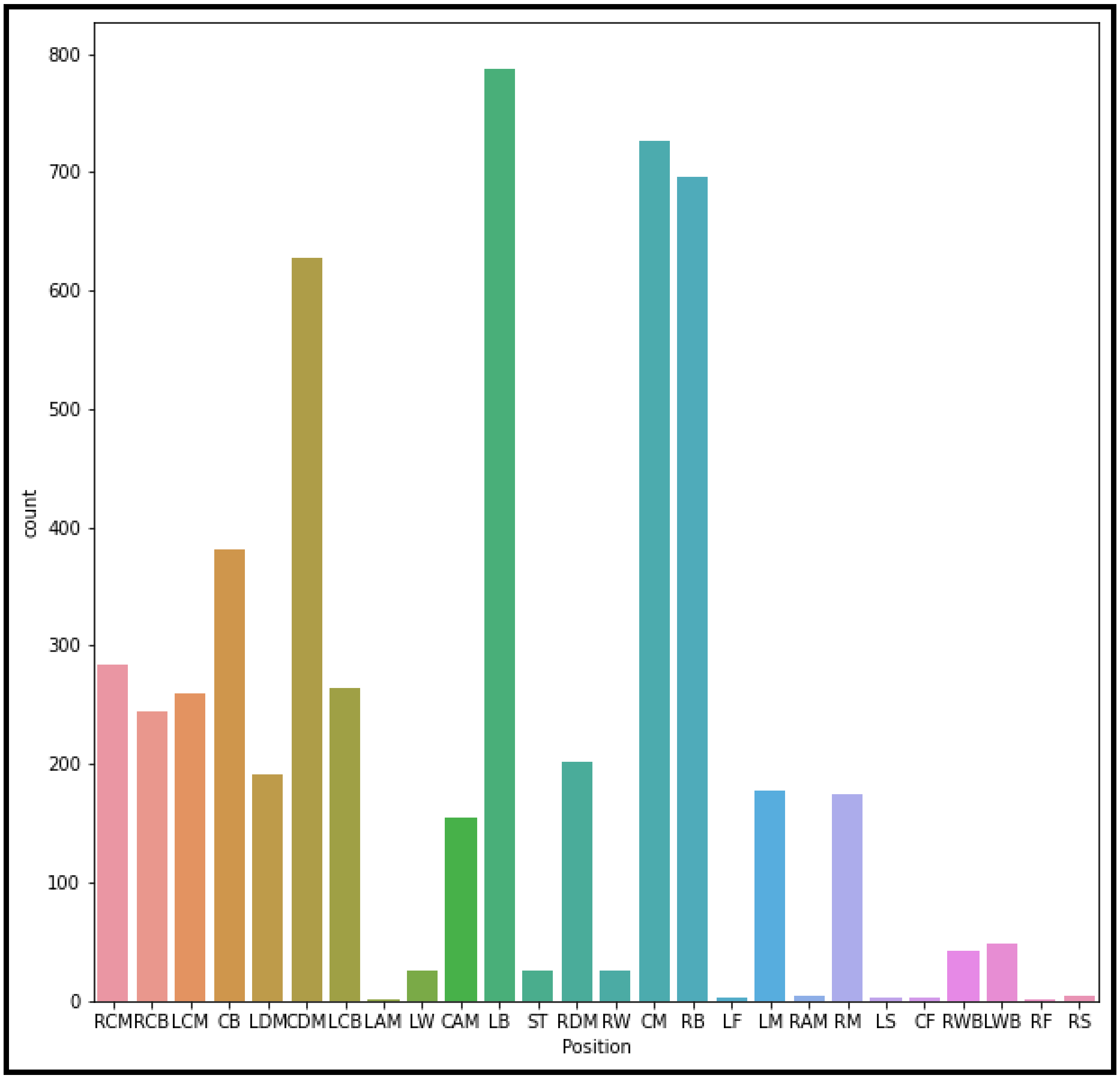
Out of all the above plots, plot for **number of clusters = 3** is the most suitable with most reasonable division of clusters and least amount of mixed regions among all 5 (number of clusters = 2,3,4,5,6) cases.

Histograms on the basis of position for number of clusters=3 of all the clusters :

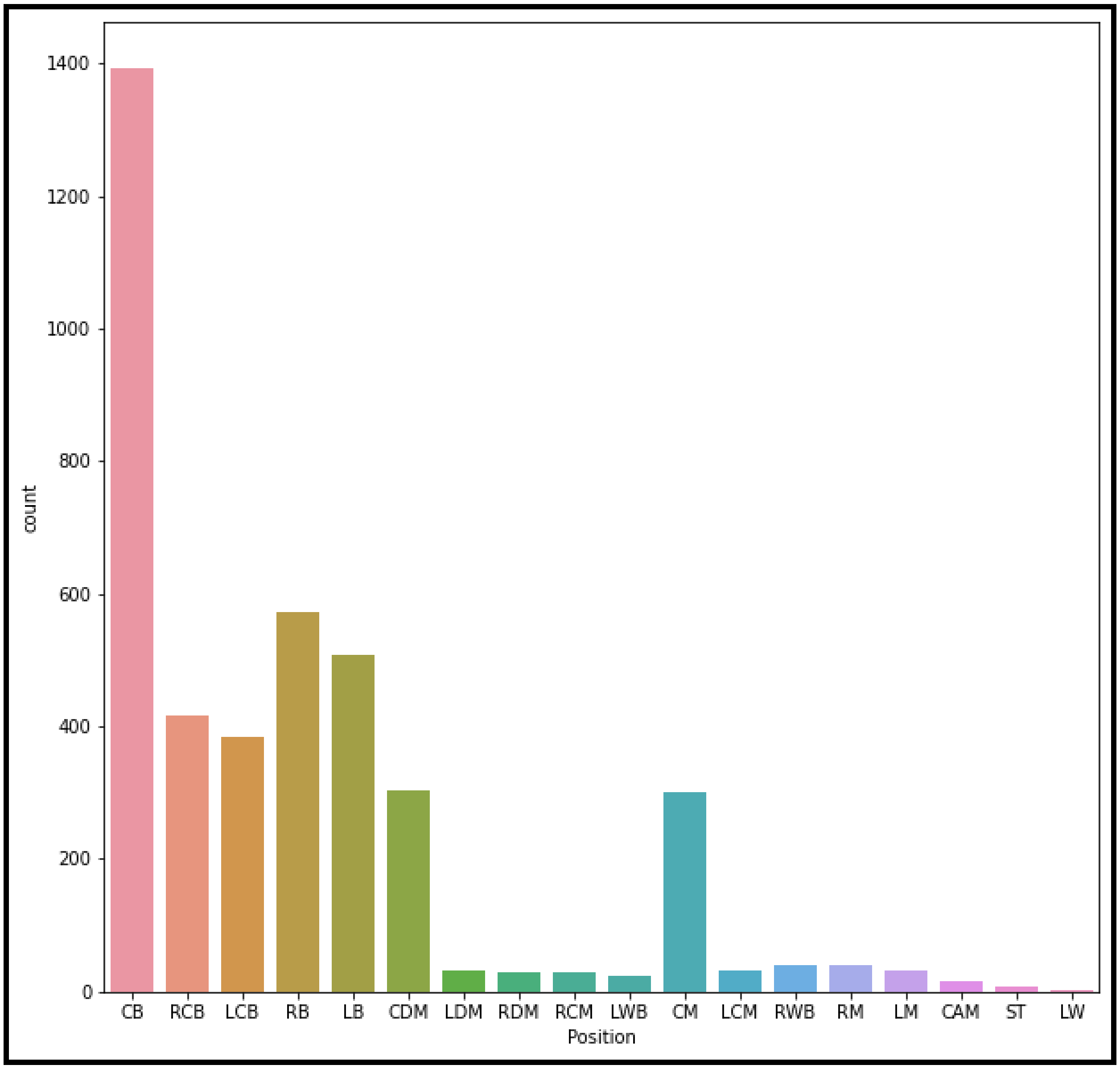
Cluster 0 :



Cluster 1 :


count
800
700
600
500
400
300
200
100
0
RCM
RCB
LCM
CB
LDM
CDM
LCB
LAM
LW
CAM
LB
ST
RDM
RW
CM
RB
LF
LM
RAM
RM
LS
CF
RWB
LWB
RF
RS
Position

Cluster 2

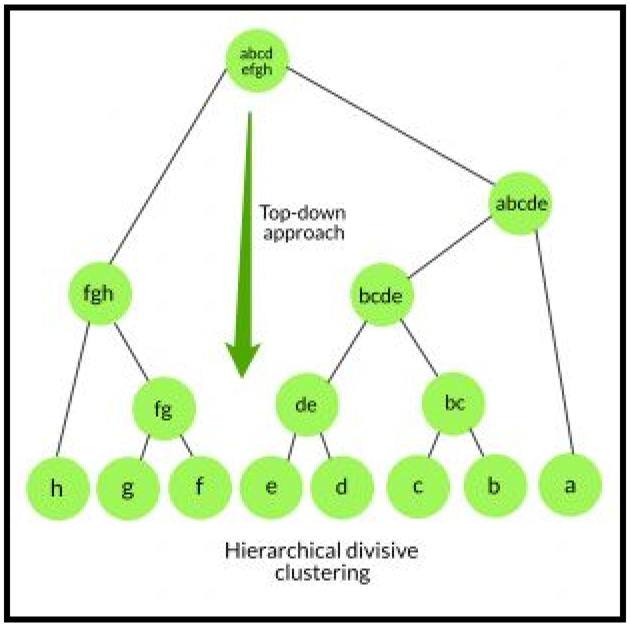


It's clearly visible that the algorithm has brought players with similar play styles together i.e. goalkeepers have been collected in one while forwards in another. Though the standard deviation of attributes in a cluster is relatively high for some values which might be the reason for lower silhouette score compared to Kmeans.

2. Divisive Clustering (Top-Down Strategy)

About :

Also known as top-down approach. This algorithm also does not require to prespecify the number of clusters. Top-down clustering requires a method for splitting a cluster that contains the whole data and proceeds by splitting clusters recursively until individual data have been splitted into singleton clusters.



Implementation :

The whole dataset is treated as a single cluster at the start and then divided into 2 clusters using K-Means. This division into 2 sub clusters is repeated till each data point is in its own cluster. Cluster with max error (This is done using a priority queue. The 2 clusters are then pushed in a priority queue i.e. the cluster with more error will have more priority) is split at every step and the error is calculated as the sum of squared error between the center of the cluster and all the points in the cluster. The cluster with more priority is taken out of the queue and K-Means is again applied to that cluster to get two more clusters which are then pushed in the priority queue according to their error. This method is carried out until we have all the elements as a cluster (single elements).

Agglomerative vs Divisive Clustering :

Divisive seems to be much better and more accurate than Agglomerative Clustering

Reason :

Agglomerative clustering makes decisions by considering the local patterns or neighbor points without initially taking into account the global distribution of data. These early decisions cannot be undone. whereas divisive clustering takes into consideration the global distribution of data when making top-level partitioning decisions.

TASK 4 DBSCAN Clustering Algorithm

About :

This is **Density-based spatial clustering of applications with noise** (DBSCAN) clustering method. Clusters are dense regions in the data space, separated by regions of the lower density of points. The ***DBSCAN algorithm*** is based on this intuitive notion of "clusters" and "noise". The key idea is that for each point of a cluster, the neighborhood of a given radius has to contain at least a minimum number of points.

DBSCAN algorithm requires two parameters –

1. **eps** : It defines the neighborhood around a data point i.e. if the distance between two points is lower or equal to 'eps' then they are considered as neighbors. If the eps value is chosen too small then a large part of the data will be considered as outliers. If it is chosen very large then the clusters will merge and majority of the data points will be in the same clusters. One way to find the eps value is based on the ***k-distance graph***.
2. **MinPts**: Minimum number of neighbors (data points) within eps radius. Larger the dataset, the larger value of MinPts must be chosen. As a general rule, the minimum MinPts can be derived from the number of dimensions D in the dataset as, MinPts >= D+1. The minimum value of MinPts must be chosen at least 3.

DBSCAN algorithm can be abstracted in the following steps –

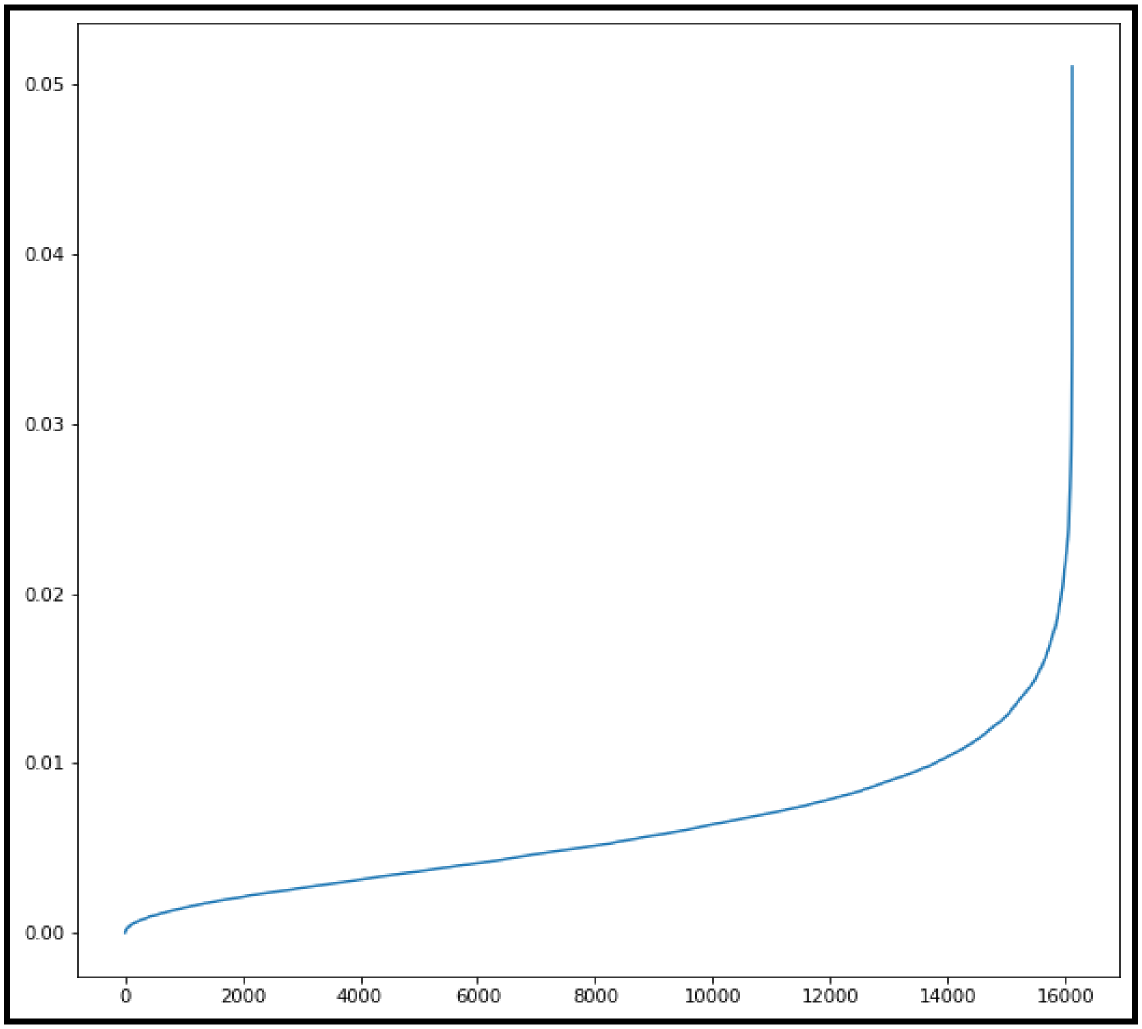
1. Find all the neighbor points within eps and identify the core points or visited with more than MinPts neighbors.
2. For each core point if it is not already assigned to a cluster, create a new cluster.
3. Find recursively all its density connected points and assign them to the same cluster as the core point.
   A point *a* and *b* are said to be density connected if there exist a point *c* which has a sufficient number of points in its neighbors and both the points *a* and *b* are within the *eps distance*. This is a chaining process. So, if *b* is neighbor of *c*, *c* is neighbor of *d*, *d* is neighbor of *e*, which in turn is neighbor of *a* implies that *b* is neighbor of *a*.

4. Iterate through the remaining unvisited points in the dataset. Those points that do not belong to any cluster are noise.

Implementation & tuning of parameters :

We first scaled the data then normalized it and then finally applied pca for dimensionality reduction and better analysis . We used the inbuilt library (DBSCAN)function to implement .

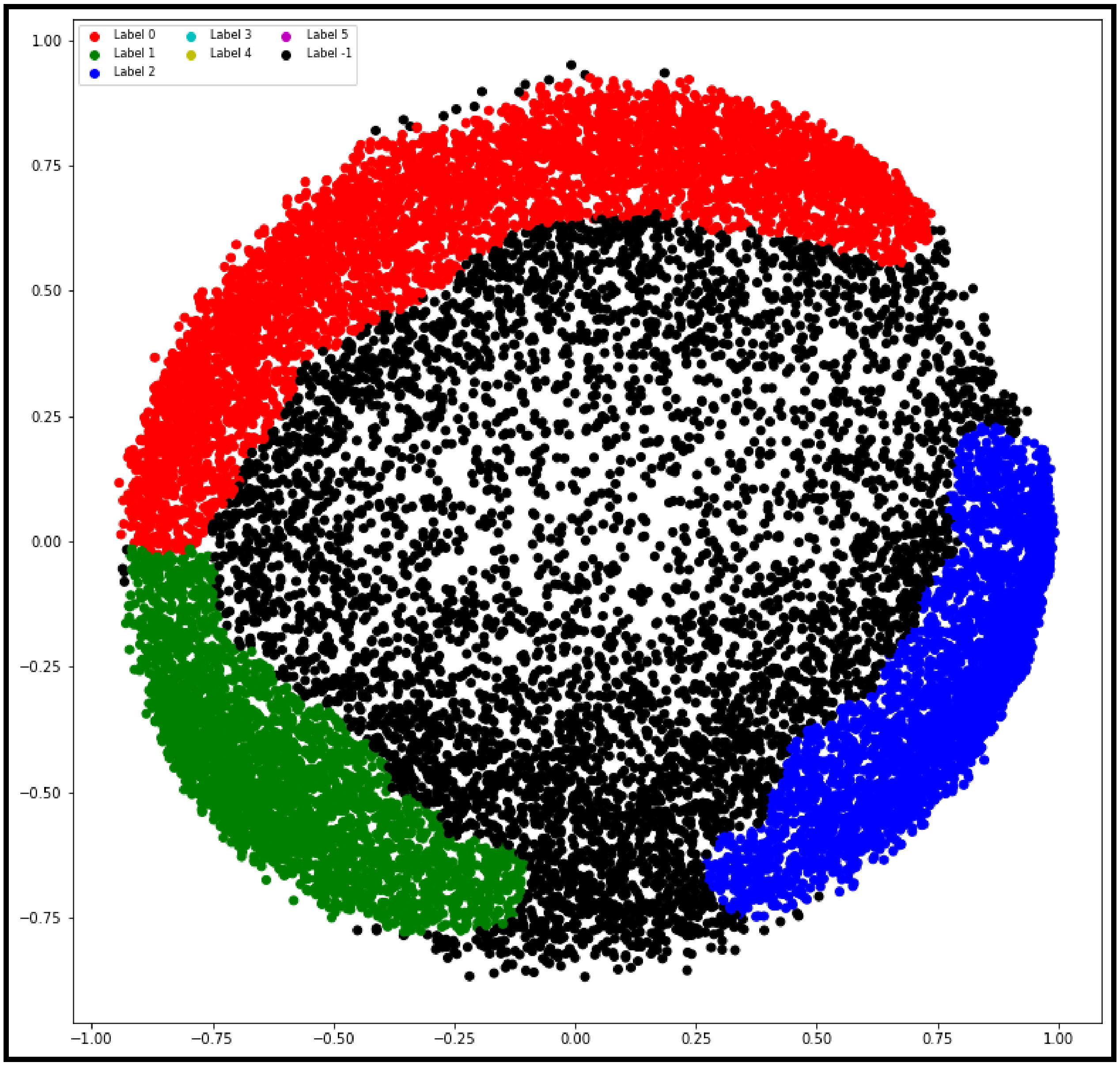
To select the optimum value of epsilon and min points we first did a hit and trial and scatter plot just to see how the clusters were being formed . Then we plotted the k-distance graph, plotting the distance to the k = minPts-1 nearest neighbor ordered from the largest to the smallest value. The values near the elbow give the optimal values of ε. For most of the values of minpoints, the elbow occurs between ε = 0.01 and 0.02 which is visible in the k-distance plot below .



We further checked for eps ={0.01,0.015,0.02…………,0.09} and for each epsilon we calculated silhouette score for minpts ranging from 75 to 150 and recorded the max silhouette score for each epsilon . The following results were observed :

Epsilon	**best minpts value**	**max silhouette score**
0.01	N/A	N/A
0.015	N/A	N/A
0.02	N/A	N/A
0.025	83	0.043320909653243256
0.03	97	0.07403260622021335
0.035	115	0.09913095019409887
0.04	141	0.10138754099738309
0.045	N/A	N/A
0.05	85	0.07126880262118462
0.055	99	0.1513105634875311
0.06	118	0.1610099807271151
0.065	113	0.24318257916120084
0.07	128	0.2556848911274334
0.075	143	0.16650384690966719
0.08	143	0.16098142175622634
0.085	N/A	N/A
0.09	N/A	N/A

The best 3 selections considering silhouette score, scatter plots ,inter and intra cluster similarity and distribution in each cluster are given below .



Outliers=5350

Cluster 0 - 4719

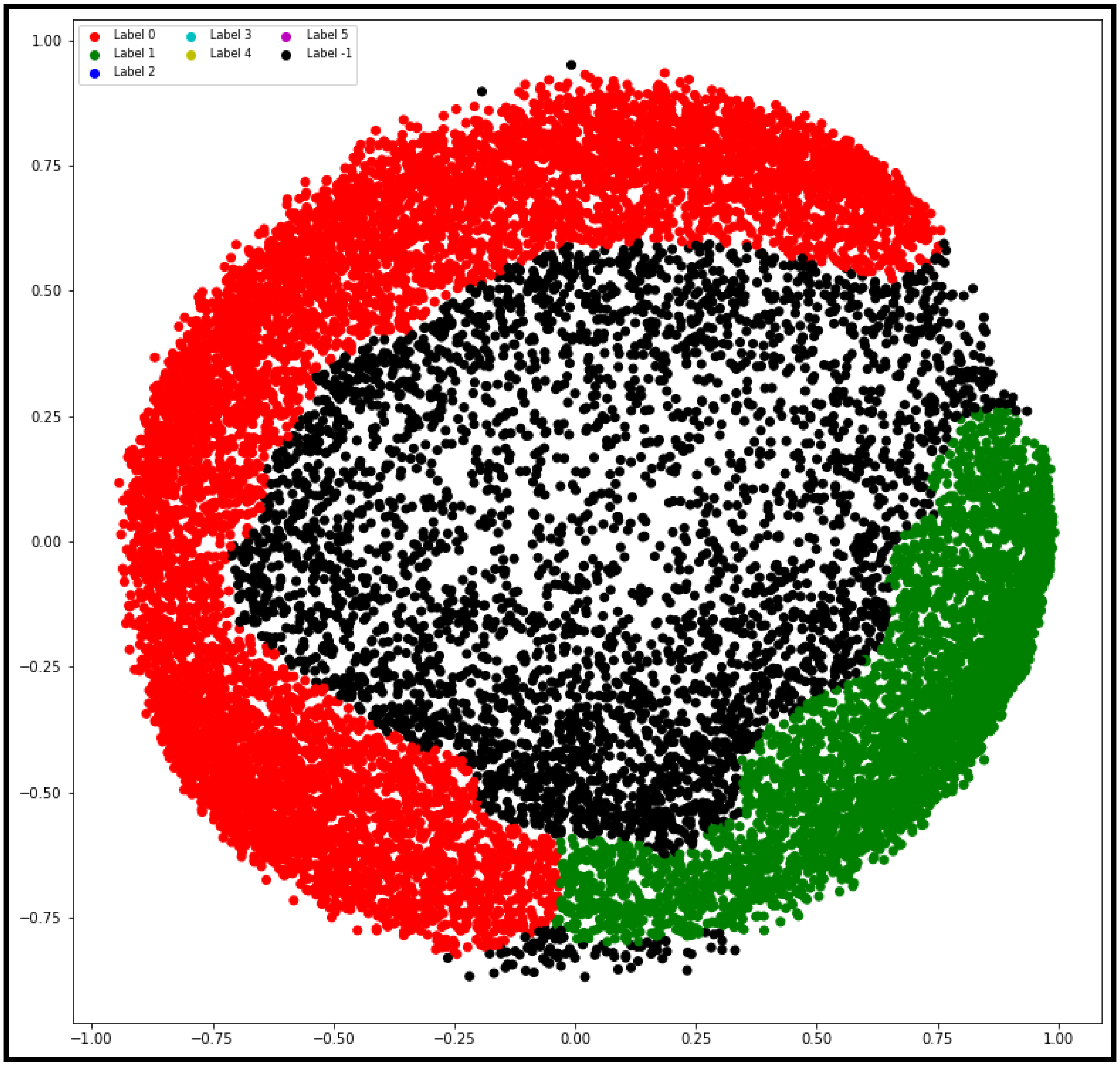
Cluster 1 - 2840

Cluster 2 - 3213

epsilon=0.07

minpts=128

Silhouette score=0.2556848911274334



Outliers=3813

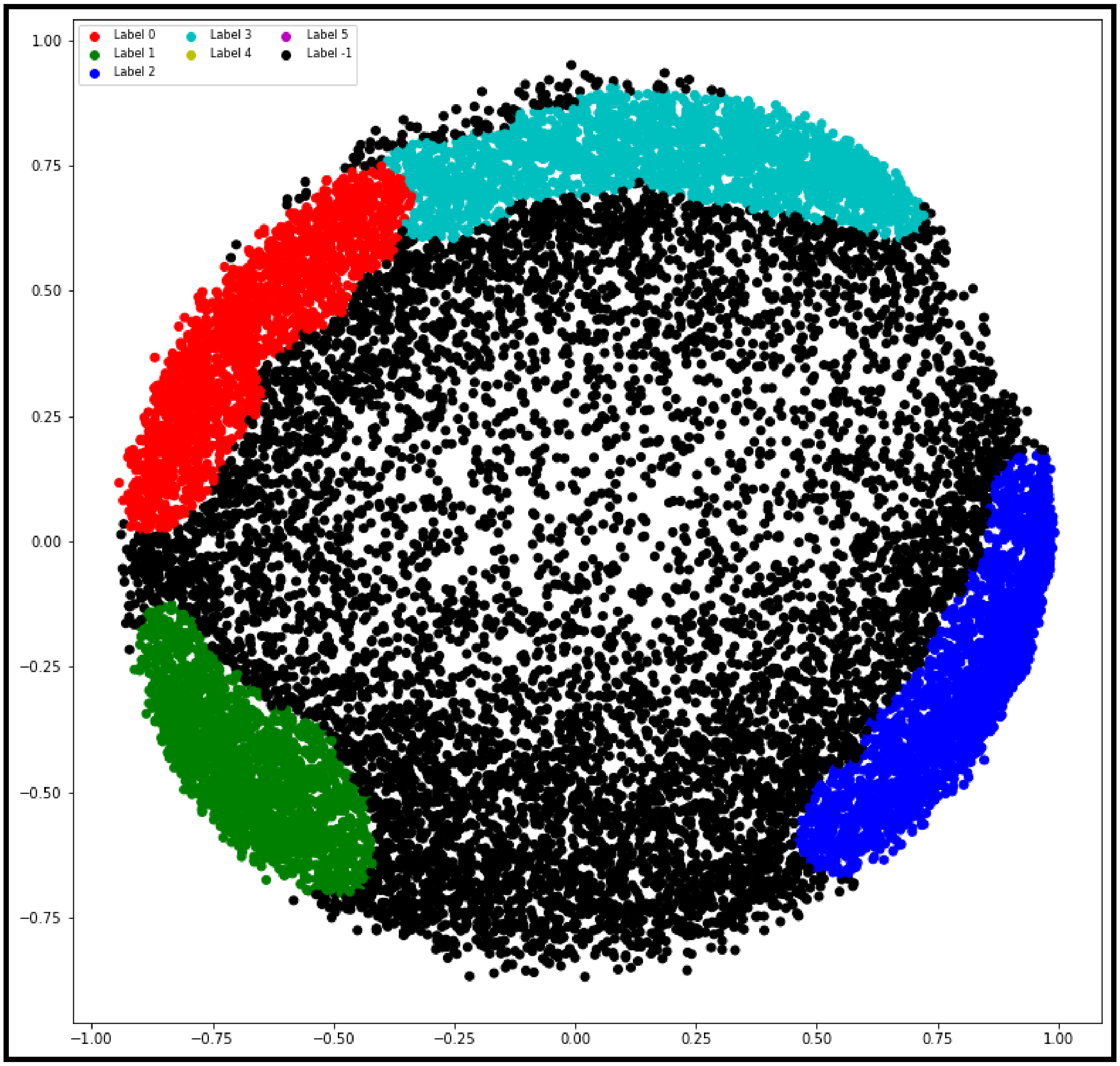
Cluster 0 - 8337

Cluster 1 - 3972

epsilon=0.08

minpts=143

Silhouette score=0.16098142175622634



Outliers=7822

Cluster 0 - 1812

Cluster 1 - 1942

Cluster 2 - 2499

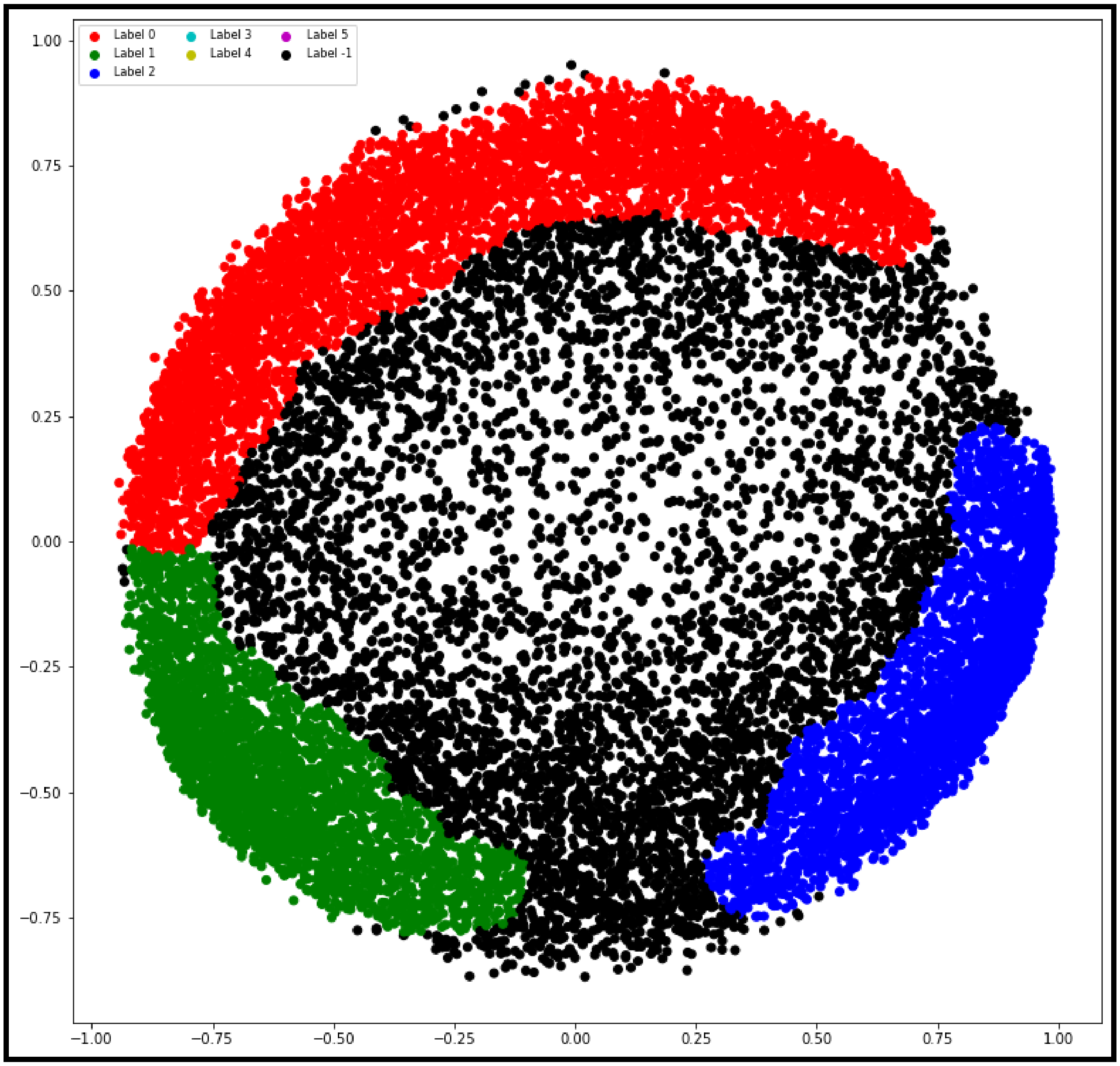
Cluster 3 - 2097

epsilon=0.055

minpts=99

Silhouette score=0.1513105634875311

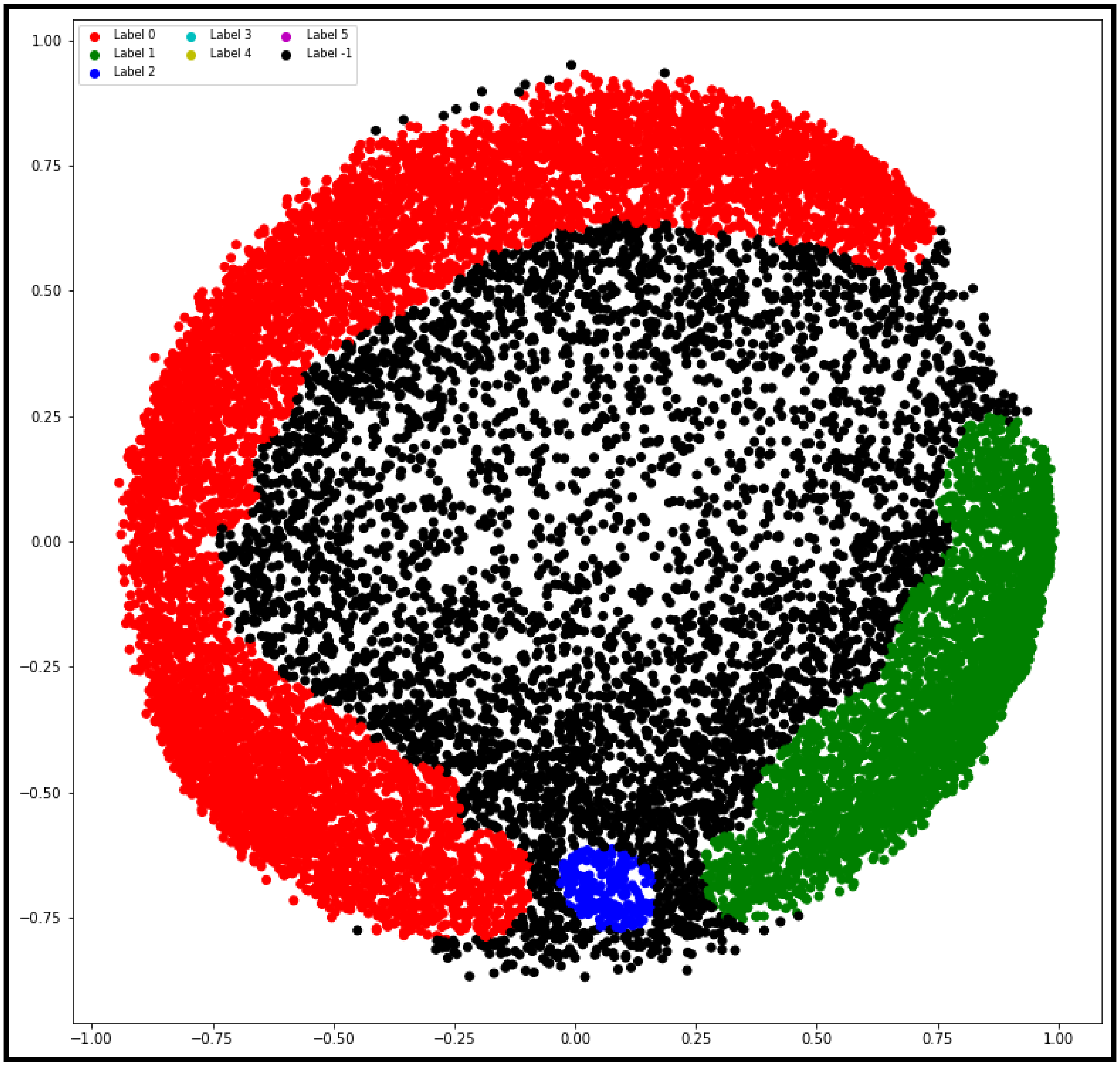
The best and final selection for dbscan :



Outliers=5350
Cluster 0 - 4719
Cluster 1 - 2840
Cluster 2 - 3213
epsilon=0.07
minpts=128
Silhouette score=0.2556848911274334

Analysis :

The above model has the maximum silhouette score and has best clusters in terms of inter and intra cluster distances . Inter cluster distances are high while intra cluster distances are like it should be if we increase the value of epsilon slightly (for instance say 0.072) silhouette score will reduce but clusters will expand a little which may be considered as improvement in terms of reducing the outliers as the clusters will be expanded a little but it led to merging of 2 clusters and hence making the increment undesirable as in figure below so it we stopped here.



CONCLUSION

We found that **K-Means** was the Best Clustering method. The major reason being : Among all the clustering methods that we did, we feel that K means gave us the best outcome as it was trained on X_Normalized and yet for K = 3 we got a very neat cluster and the Inter/Intra cluster matrices were analysed.Other reasons are first the silhouette score is maximum by a considerable margin secondly cleaner scatter plot (ie better in terms of inter and intra cluster similarity)and finally because the data was more suited for k-means .It's clear that the choice of best algorithm depends a lot on the data and for this data k-means is the best choice as it made the most meaningful clusters .

Major Reference to understand how to use libraries-Geeks for geeks

To brief up all the Tasks :

1. Best clustering method according to us **K-Means.** Inter and Intra class similarities have been explained in detail for K means, and a similar approach is followed for Hierarchical Clustering as well.
2. However, DBScan made the most meaningful clusters in terms of identifying arbitrary shaped clusters and the outliers . But we didn't consider it on par with k-means as it was trained on X_principal which keeps similar players together as the PCA algorithm would compress the similar ones to closer points in the 2D plane. We don't regard this as the best method as it is trained on incomplete data.
3. Since DBSCAN is unidirectional, Hierarchical has an advantage over DBSCAN. If a threshold error value is known below which the error is required, hierarchical is the best approach to follow because the error keeps on decreasing while traversing down the tree.