# Data Analytics I - Clustering Project

**Team Members:**

Tushar Choudhary (2019111019)

Ayush Goyal (2019111026)

**Please go through the Jupyter notebooks as well as we have discussed our code and analyzed the results there as well.**

## 1. Data visualization

We imported the data from the csv file and printed it on the notebook. Then we extracted the information for the headers/ columns (like name, data type etc). Then we printed the shape i.e the dimension of the dataset. Then got the min value, max value, means etc for all the columns.

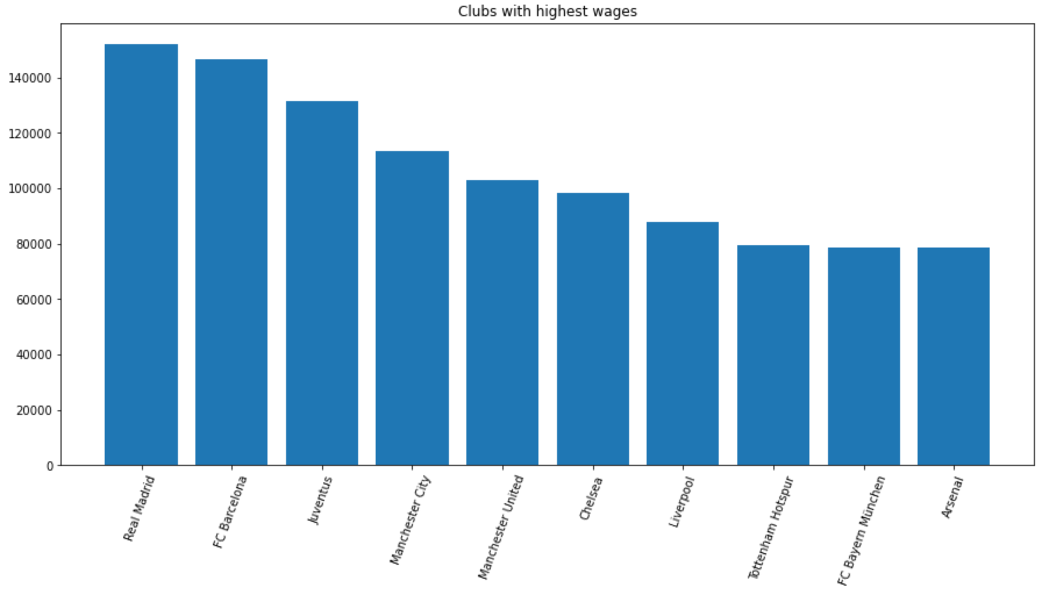
Getting the data from the CSV and printing it

Info for all dataset columns_name, dataType, null_count

Size of the dataset

Description of data such as min, max, mean, std values for all columns¶

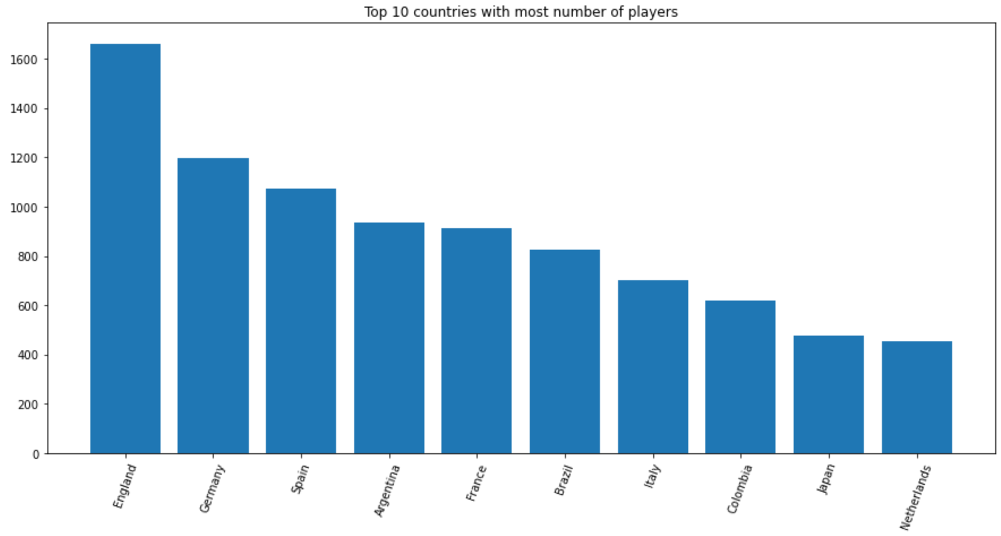
## Top 10 clubs which pay the highest average wages



Real Madrid came out to be the club that pays its players the highest as it pays the highest average wage.

## Top 10 countries with highest number of players

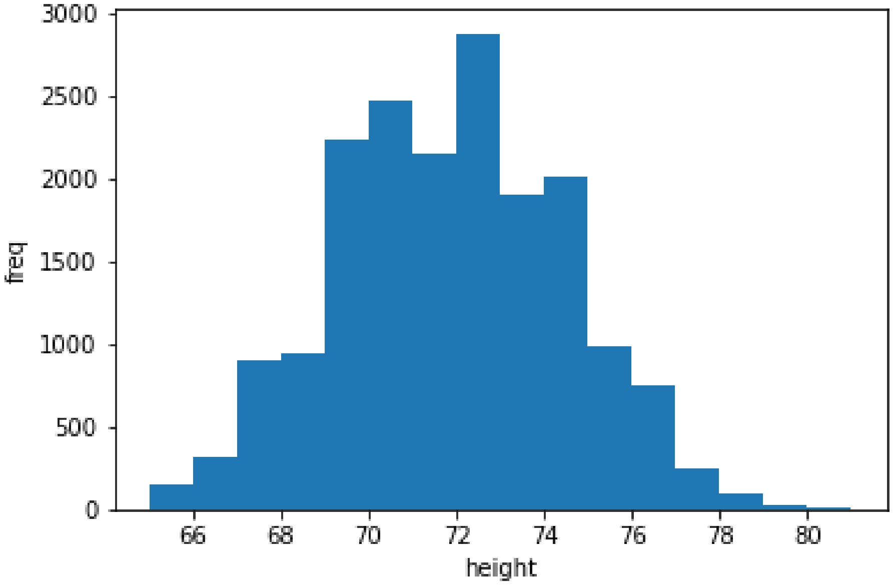
This will give us an idea about the countries where football is most popular



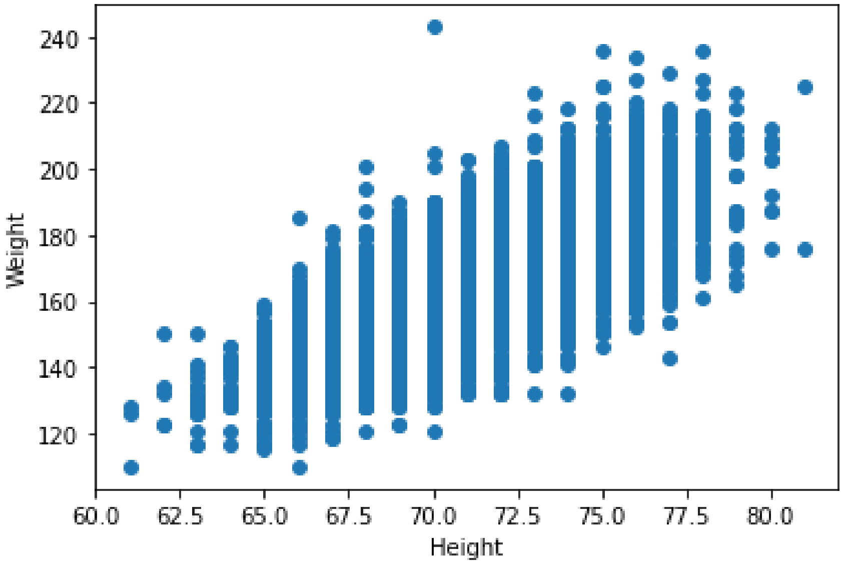
England is the country to which the maximum number of players

**Finding frequency of heights**

This graph shows that height overall has a Gaussian distribution with peak at 73.

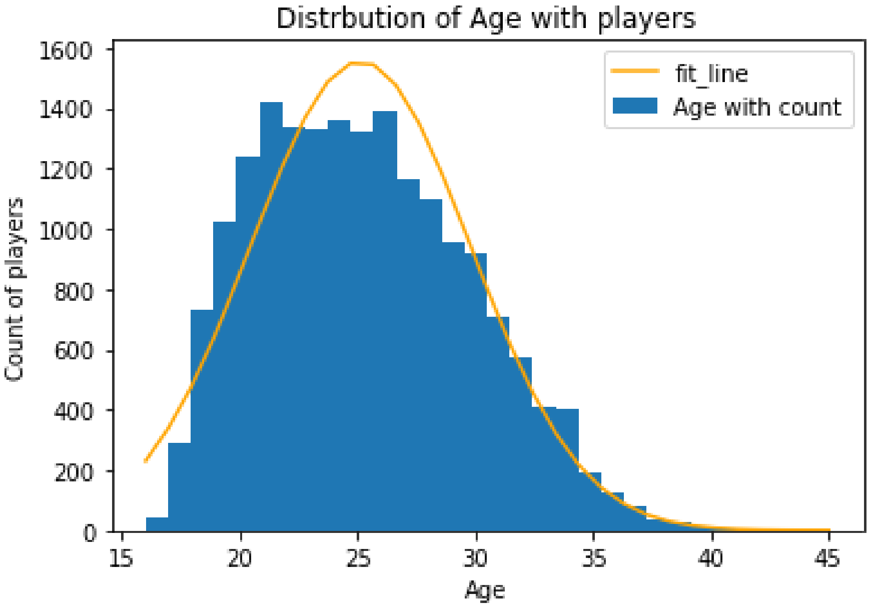


**Finding distribution of heights vs. weights**



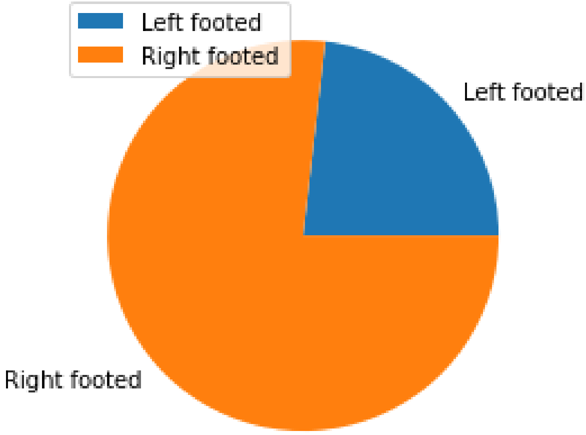
As weight increases average height increases which is expected.

**Finding frequency of different ages**



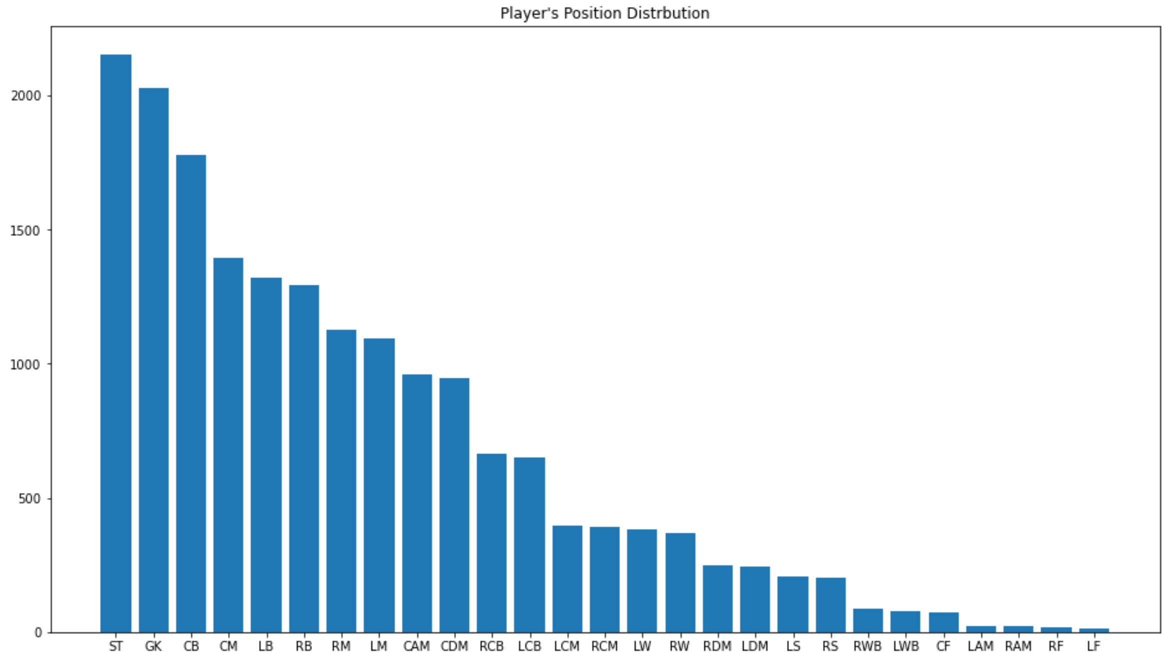
This shows that most of the players are between the ages of 21 to 27. Also younger people are more active in the sport than older people.

**Getting pie chart for count of players with right and left foot preferences**



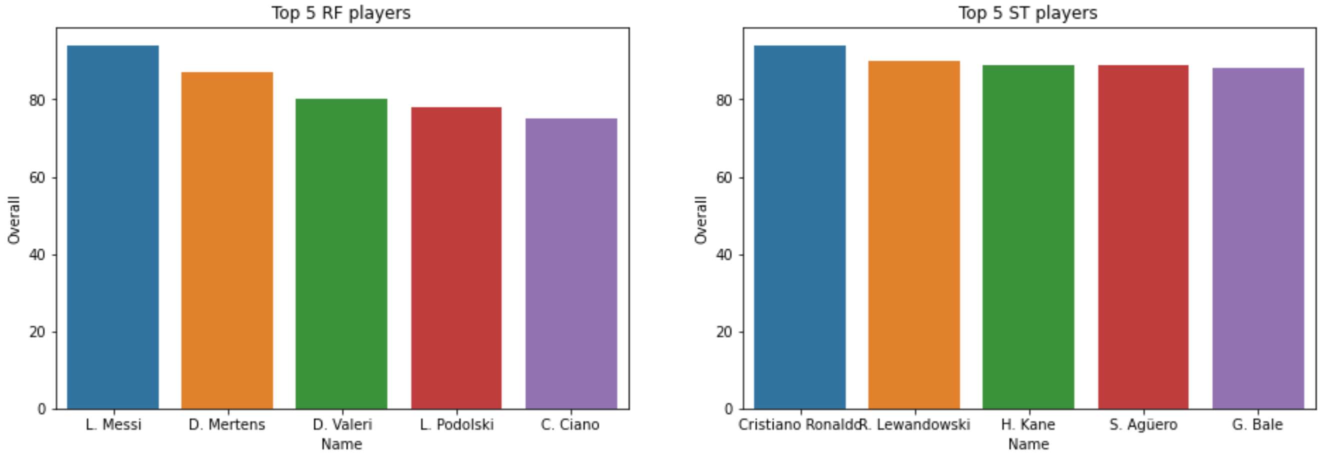
This majority of the players are right footed or prefer their right foot to kick the ball over the left.

### Getting the count for the number of players for each position



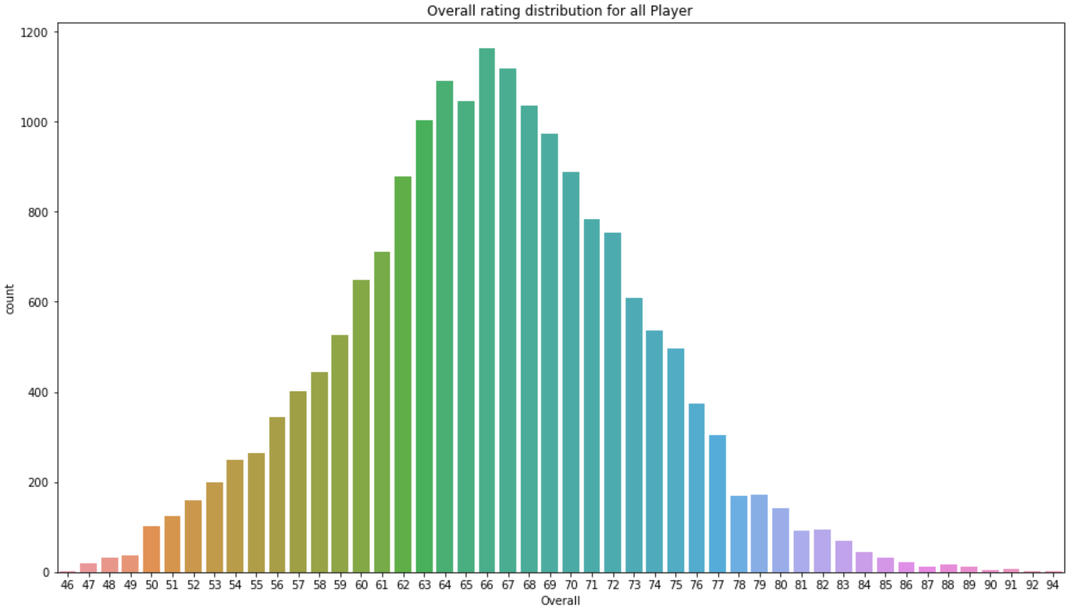
This shows that in this dataset most of the players play at ST (Striker) position followed by GK (Goalkeeper.)

### Getting top 10 strikers and right forwards



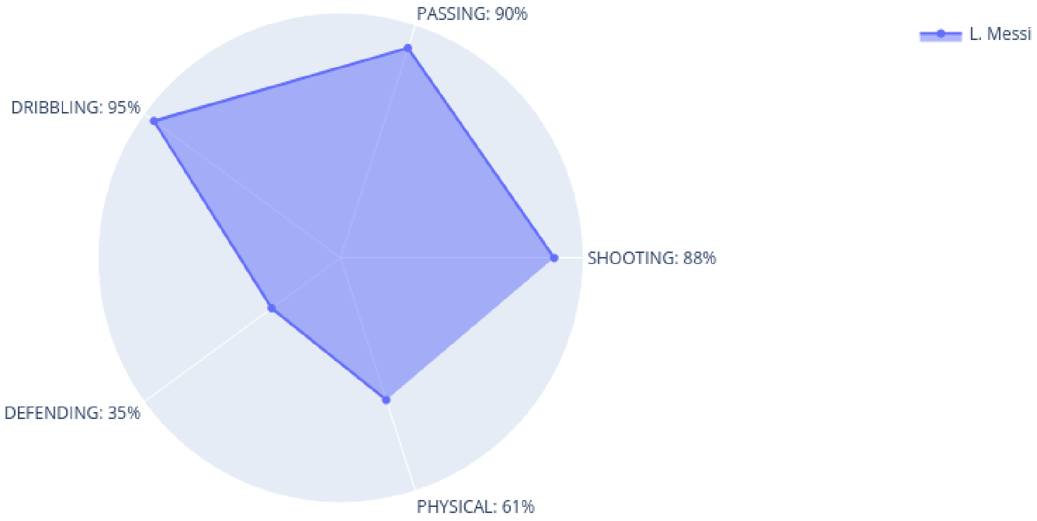
This shows that L. Messi is the best Right Forward and Cristiano Ronaldo is the best Striker (ST).

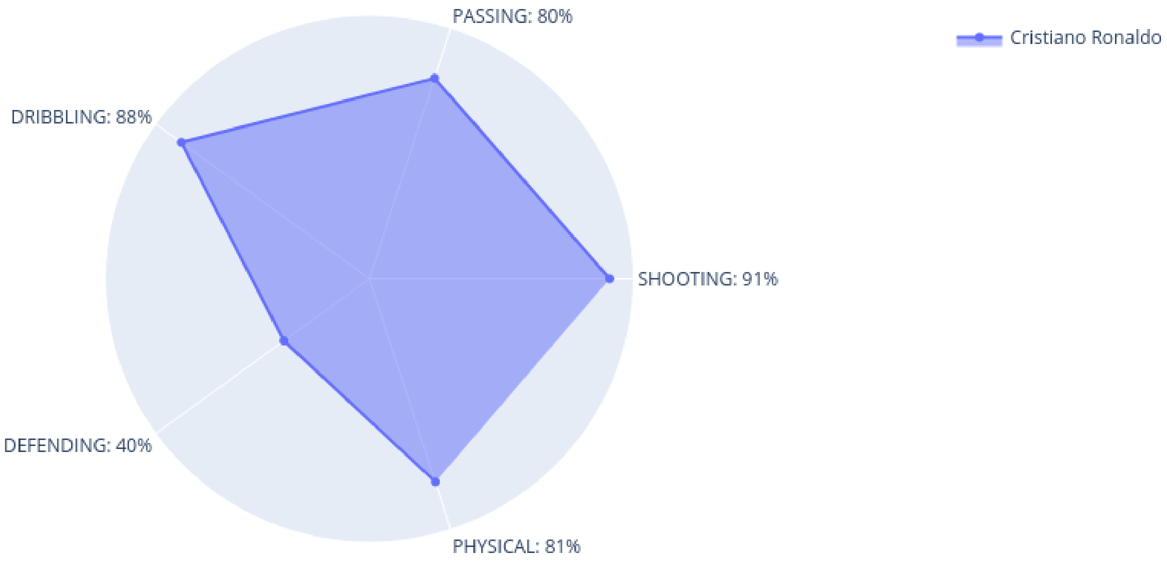
## Plotting histogram for the "overall" attribute

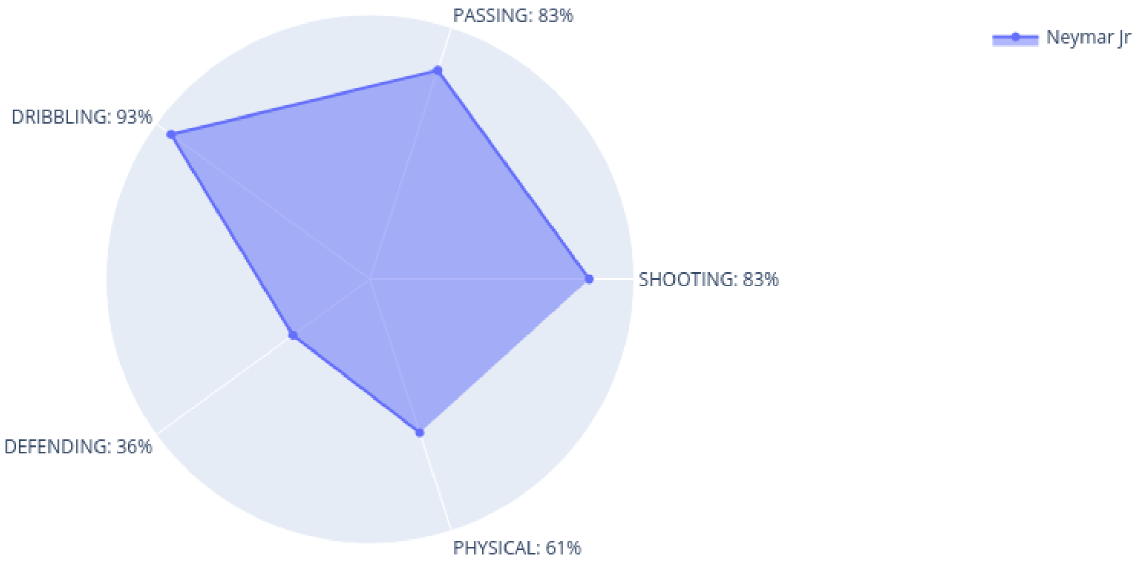


This graph shows that overall has a Gaussian distribution and players that play extremely well or extremely poorly are very few and 66 is the median for overall performance.

## Plotly graphs to analyze some of the players

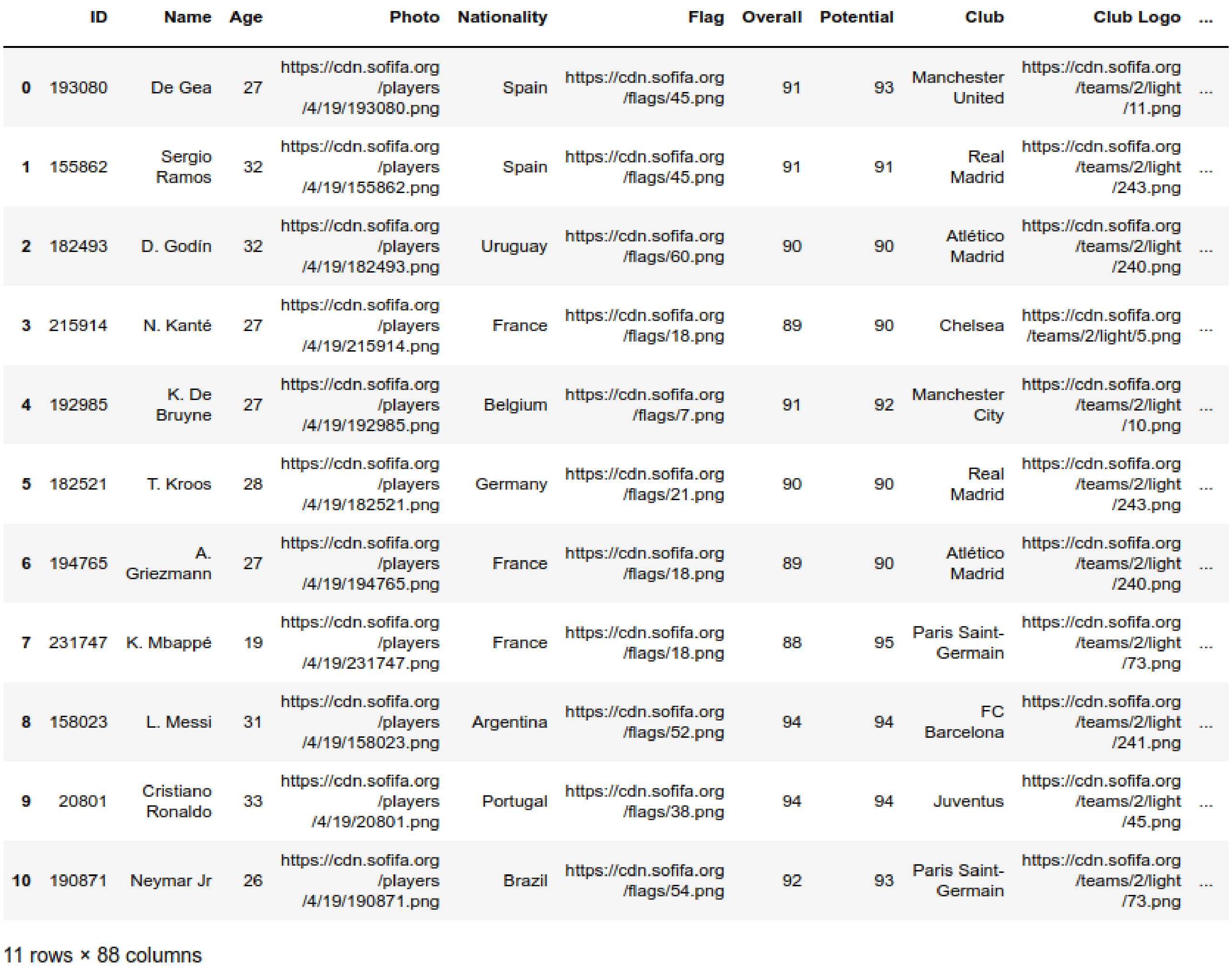



PASSING: 80%
DRIBBLING: 88%
SHOOTING: 91%
DEFENDING: 40%
PHYSICAL: 81%
Cristiano Ronaldo


PASSING: 83%
DRIBBLING: 93%
SHOOTING: 83%
DEFENDING: 36%
PHYSICAL: 61%
Neymar Jr

## Finding the best possible team - The Dream 11

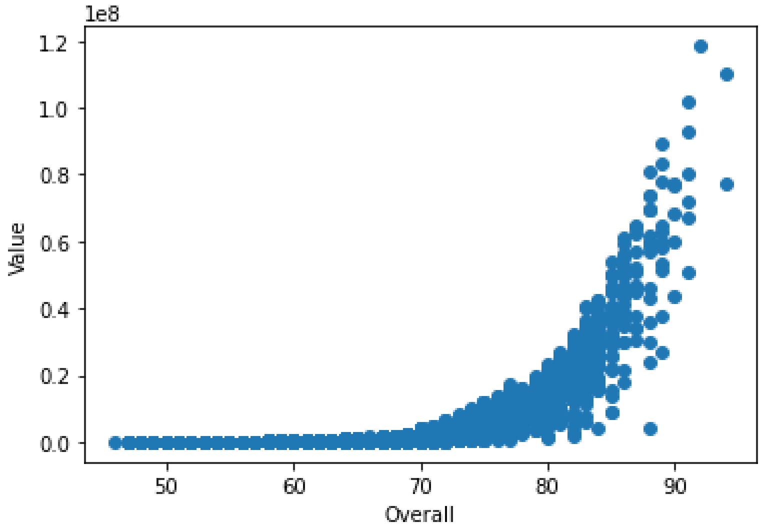
This is the team we would create if we were given an option to do so. To form this team, a 3-4-3 formation was chosen and players with best “overall” attributes were picked.



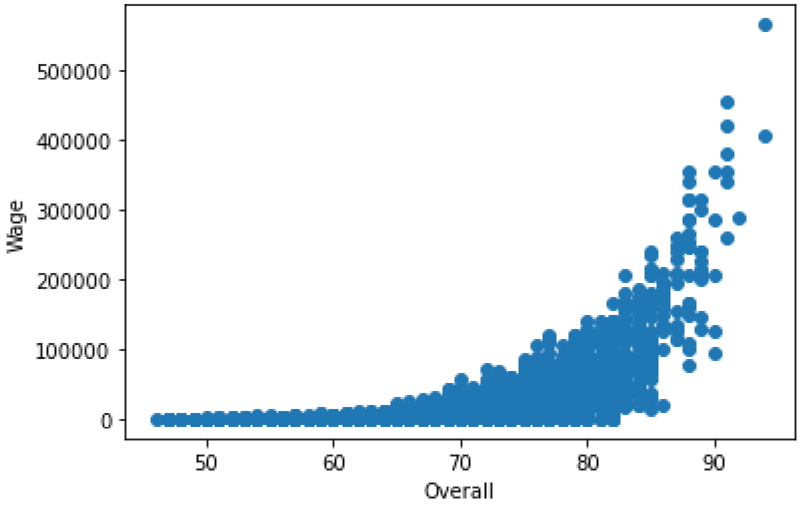
| | ID | Name | Age | Photo | Nationality | Flag | Overall | Potential | Club | Club Logo | ... |
|---|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 193080 | De Gea | 27 | https://cdn.sofifa.org/players/4/19/193080.png | Spain | https://cdn.sofifa.org/flags/45.png | 91 | 93 | Manchester United | https://cdn.sofifa.org/teams/2/light/11.png | ... |
| **1** | 155862 | Sergio Ramos | 32 | https://cdn.sofifa.org/players/4/19/155862.png | Spain | https://cdn.sofifa.org/flags/45.png | 91 | 91 | Real Madrid | https://cdn.sofifa.org/teams/2/light/243.png | ... |
| **2** | 182493 | D. Godín | 32 | https://cdn.sofifa.org/players/4/19/182493.png | Uruguay | https://cdn.sofifa.org/flags/60.png | 90 | 90 | Atlético Madrid | https://cdn.sofifa.org/teams/2/light/240.png | ... |
| **3** | 215914 | N. Kanté | 27 | https://cdn.sofifa.org/players/4/19/215914.png | France | https://cdn.sofifa.org/flags/18.png | 89 | 90 | Chelsea | https://cdn.sofifa.org/teams/2/light/5.png | ... |
| **4** | 192985 | K. De Bruyne | 27 | https://cdn.sofifa.org/players/4/19/192985.png | Belgium | https://cdn.sofifa.org/flags/7.png | 91 | 92 | Manchester City | https://cdn.sofifa.org/teams/2/light/10.png | ... |
| **5** | 182521 | T. Kroos | 28 | https://cdn.sofifa.org/players/4/19/182521.png | Germany | https://cdn.sofifa.org/flags/21.png | 90 | 90 | Real Madrid | https://cdn.sofifa.org/teams/2/light/243.png | ... |
| **6** | 194765 | A. Griezmann | 27 | https://cdn.sofifa.org/players/4/19/194765.png | France | https://cdn.sofifa.org/flags/18.png | 89 | 90 | Atlético Madrid | https://cdn.sofifa.org/teams/2/light/240.png | ... |
| **7** | 231747 | K. Mbappé | 19 | https://cdn.sofifa.org/players/4/19/231747.png | France | https://cdn.sofifa.org/flags/18.png | 88 | 95 | Paris Saint-Germain | https://cdn.sofifa.org/teams/2/light/73.png | ... |
| **8** | 158023 | L. Messi | 31 | https://cdn.sofifa.org/players/4/19/158023.png | Argentina | https://cdn.sofifa.org/flags/52.png | 94 | 94 | FC Barcelona | https://cdn.sofifa.org/teams/2/light/241.png | ... |
| **9** | 20801 | Cristiano Ronaldo | 33 | https://cdn.sofifa.org/players/4/19/20801.png | Portugal | https://cdn.sofifa.org/flags/38.png | 94 | 94 | Juventus | https://cdn.sofifa.org/teams/2/light/45.png | ... |
| **10** | 190871 | Neymar Jr | 26 | https://cdn.sofifa.org/players/4/19/190871.png | Brazil | https://cdn.sofifa.org/flags/54.png | 92 | 93 | Paris Saint-Germain | https://cdn.sofifa.org/teams/2/light/73.png | ... |

11 rows × 88 columns

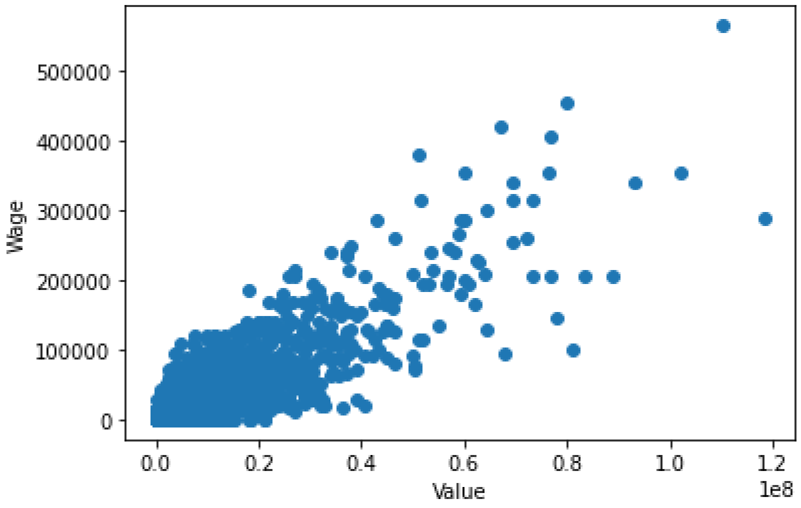
**Getting scatter plots to analyze relations between value, overall and wage**



As the overall performance of the player increases the value of that increases exponentially. This shows how significant very important football stars are.



As the overall performance of the player increases the wage of that increases



As the value of the player increases the wage of that increases. But there are very few players with extremely high wages.

## 2. K-means

Details about the implementation :

**Preprocessing** the data before clustering: while preprocessing the data, for **feature selection** we picked attributes with numerical values and converted all objects values to numeric float values so they can be used for K Means. After that we **imputed** all missing values by the mean of the same attribute, and **standardized** the entries attribute wise to prevent large vectors from dominating.

**K-means class**: The class implements the K-means algorithm. Initially it randomly chooses |n_clusters| and starts forming clusters based on distances obtained using 75 attributes. For each data point, the cluster assigned is the group whose mean has the smallest distance to the data point. The algorithm runs until no data point changes cluster or the maximum number of iterations have been run.

**Errors**: We will also find the sum of squared distance after getting our final centroids. We will be running k means number of times and the reason for that is to make sure that our clustering is not influenced by some bias present in the data. Different runs of k means might give different values of the sum of squared distances and we will choose the lowest.

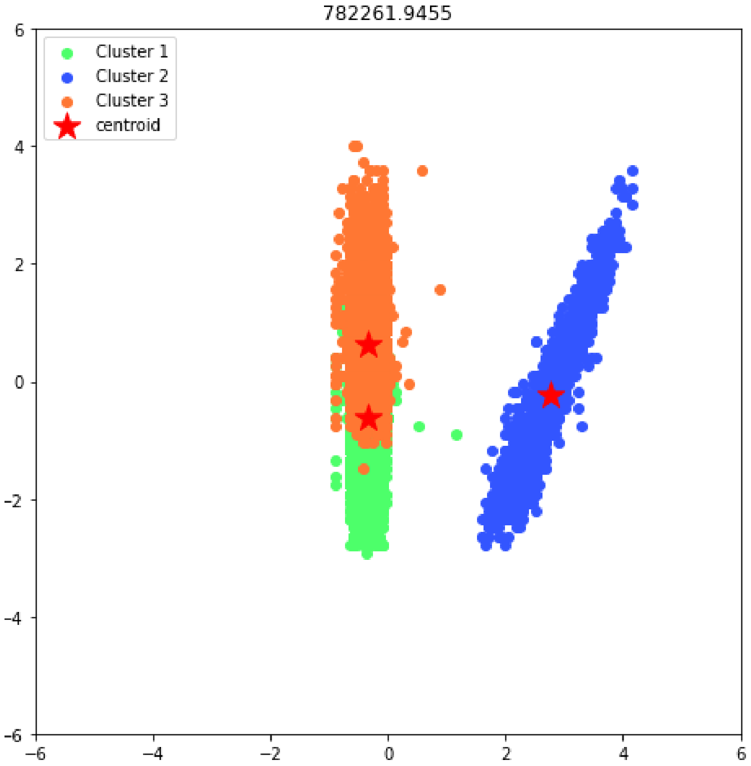
**Running K-Means and plotting the graphs:** Now we have run K-Means with values of K as 3, 5, and 7 respectively. For each K, K-means has been run multiple times with different initial centroids to make sure that the algorithm does not get trapped in some bias present in the data. It can be seen in the plots that result turn out to be roughly similar.

**Samples after running K-means**:

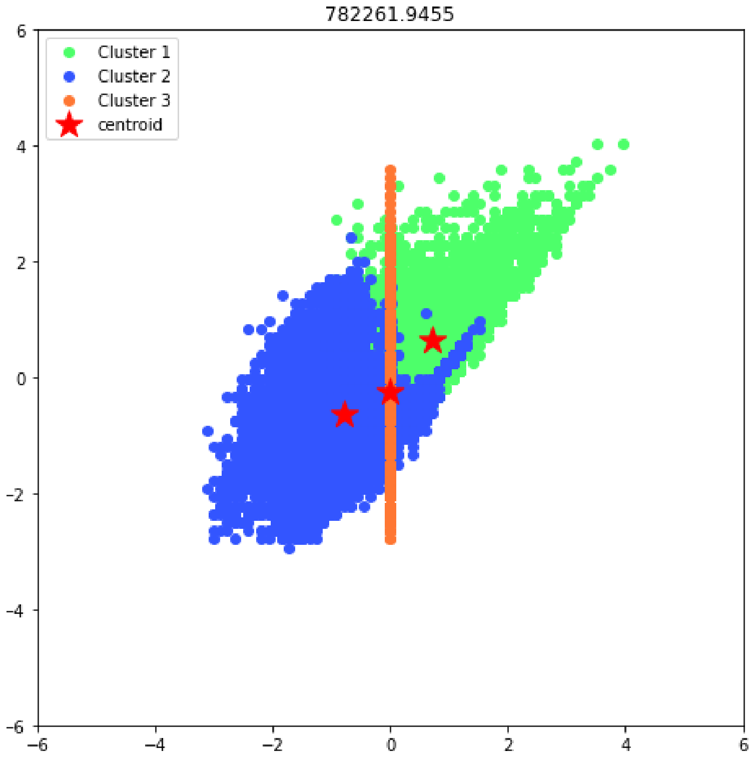
K-means has used 75 attributes to form the clusters. For visualization purposes, we have used 4 pairs of attributes and represented them on a 2D plane, with each cluster marked with a different colour. The pairs used are:

- Overall vs Gkdiving
- Overall vs ST
- Overall vs CM
- Overall vs CB

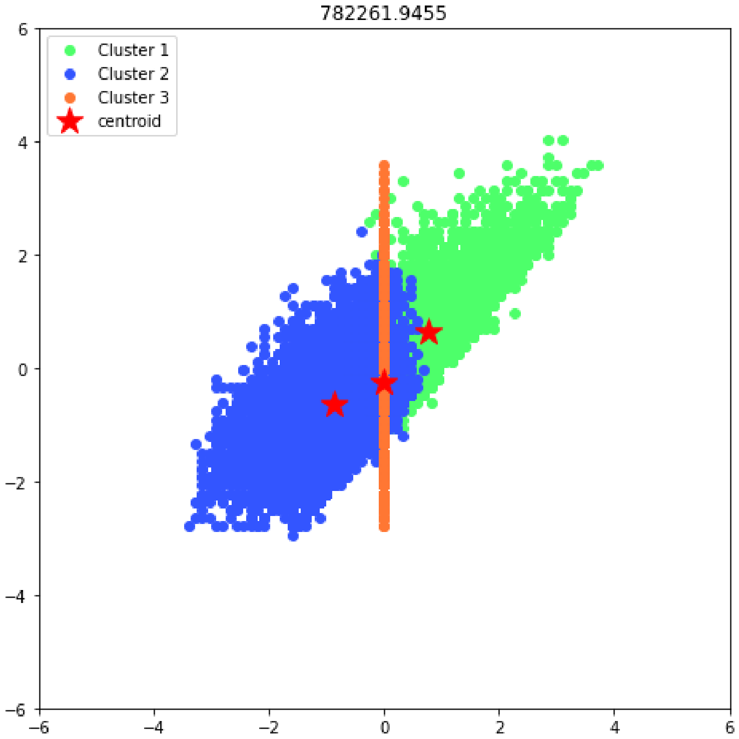
Clustering done using k=3 and graph plotted between Overall vs Gk diving-



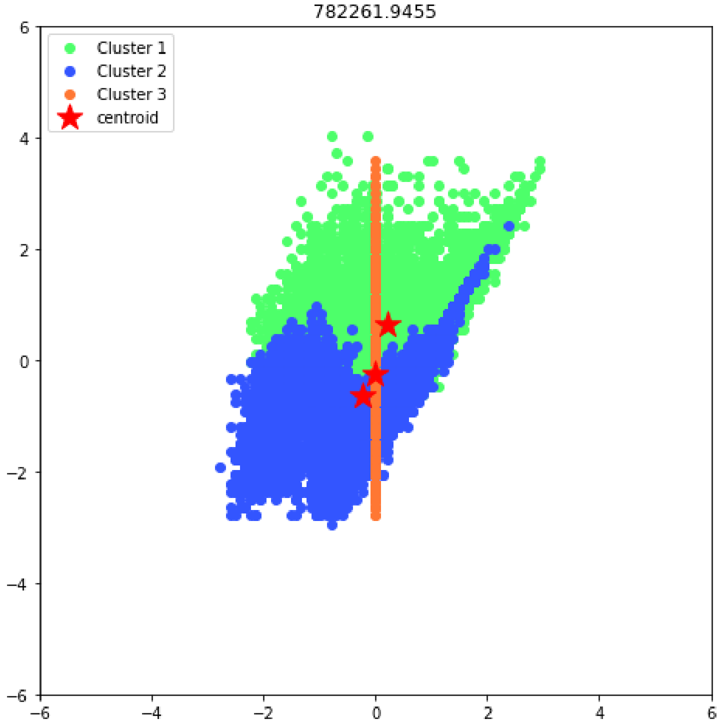
Clustering done using k=3 and graph plotted between Overall vs ST



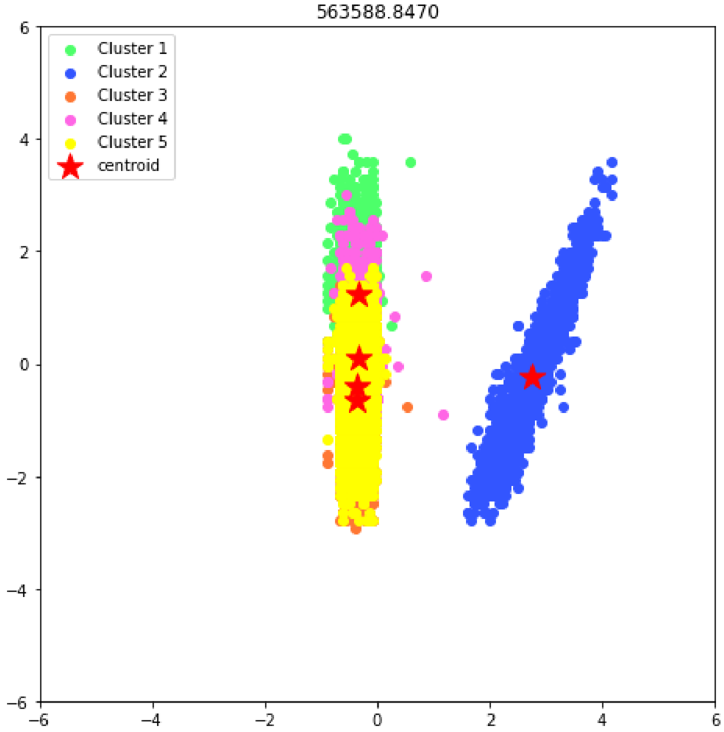
Clustering done using k=3 and graph plotted between Overall vs CM



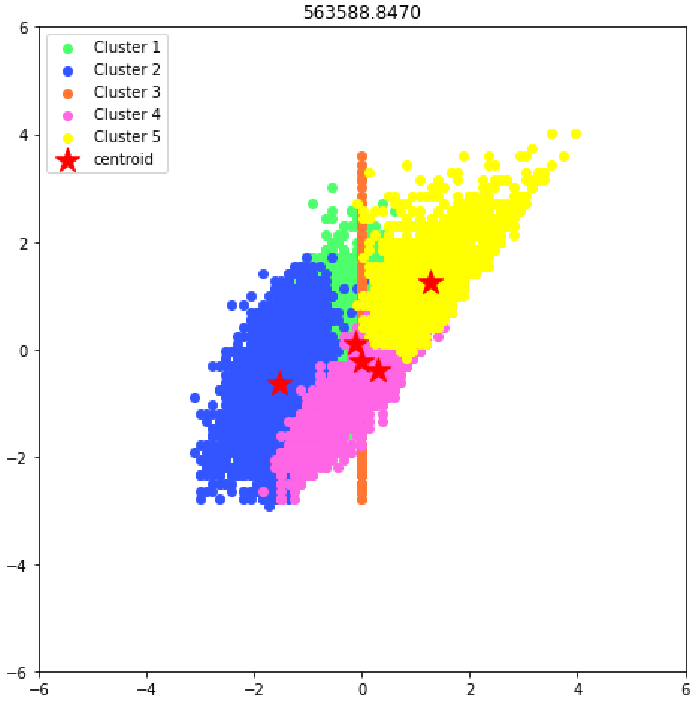
Clustering done using k=3 and graph plotted between Overall vs CB



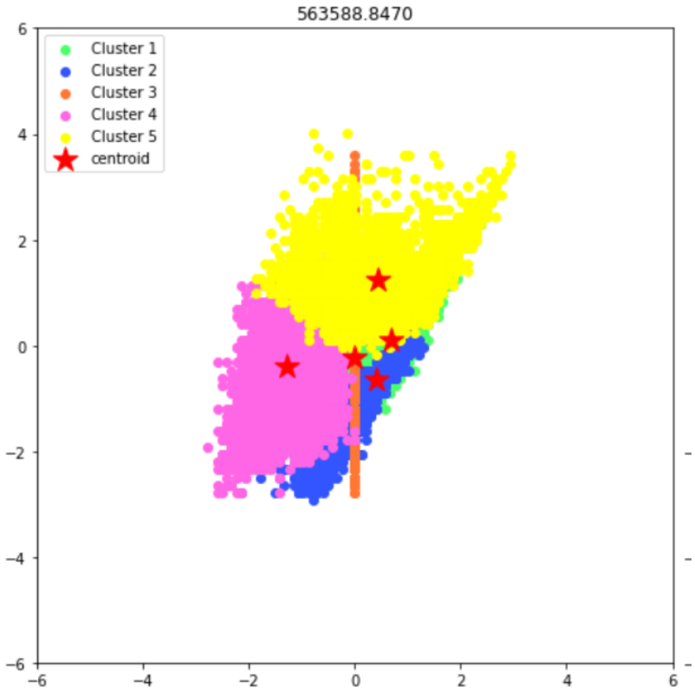
Clustering done using k=5 and graph plotted between Overall vs Gk diving



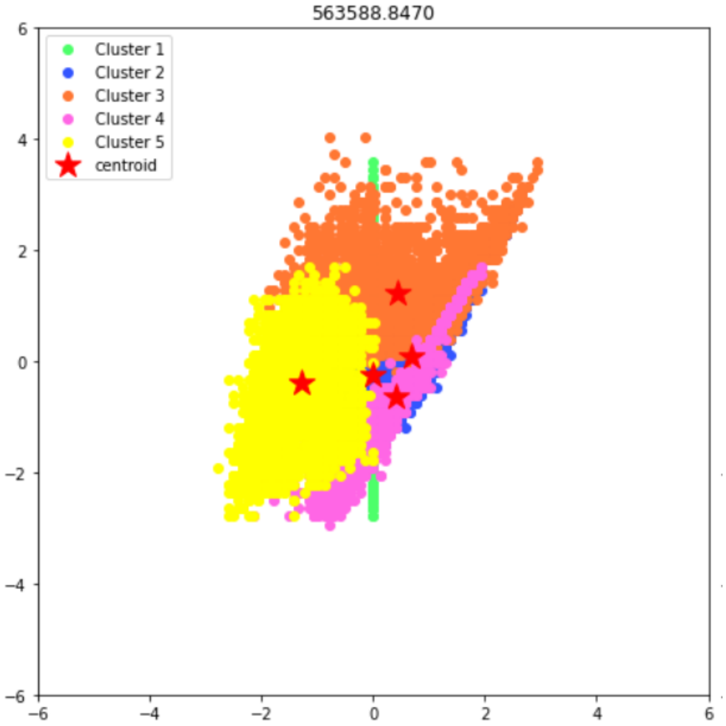
Clustering done using k=5 and graph plotted between Overall vs ST



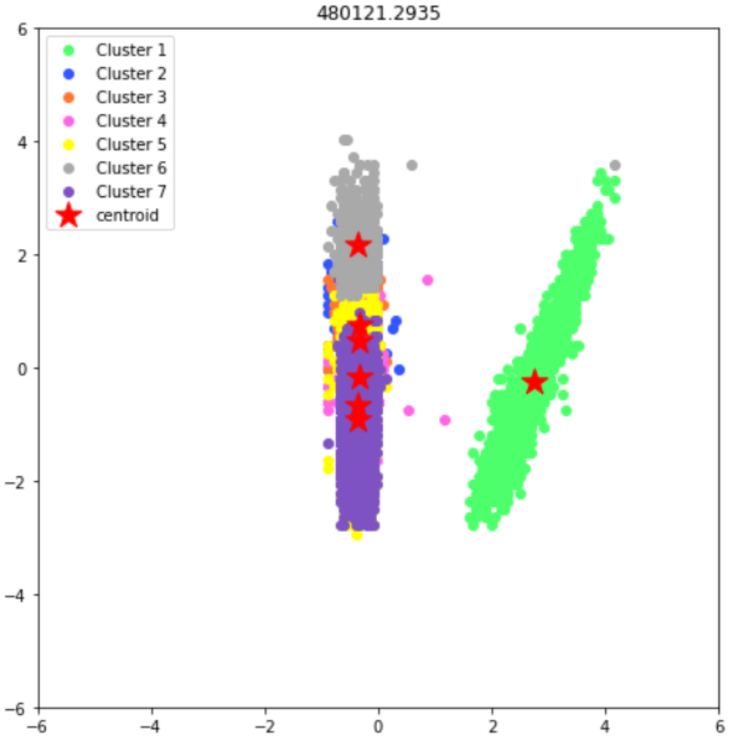
Clustering done using k=5 and graph plotted between Overall vs CM



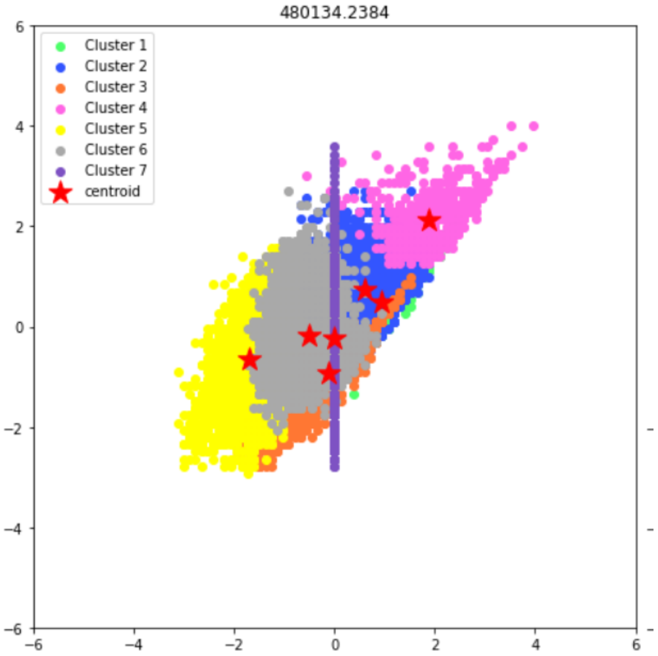
Clustering done using k=5 and graph plotted between Overall vs CB



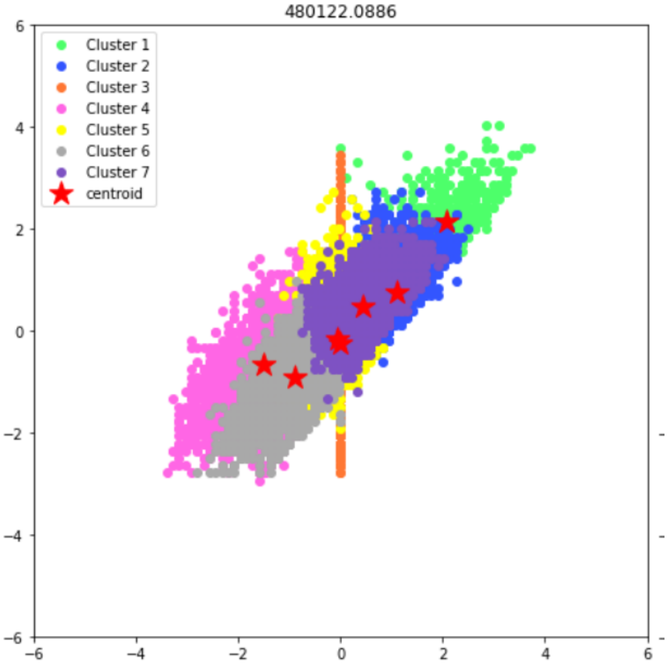
Clustering done using k=7 and graph plotted between Overall vs Gk diving



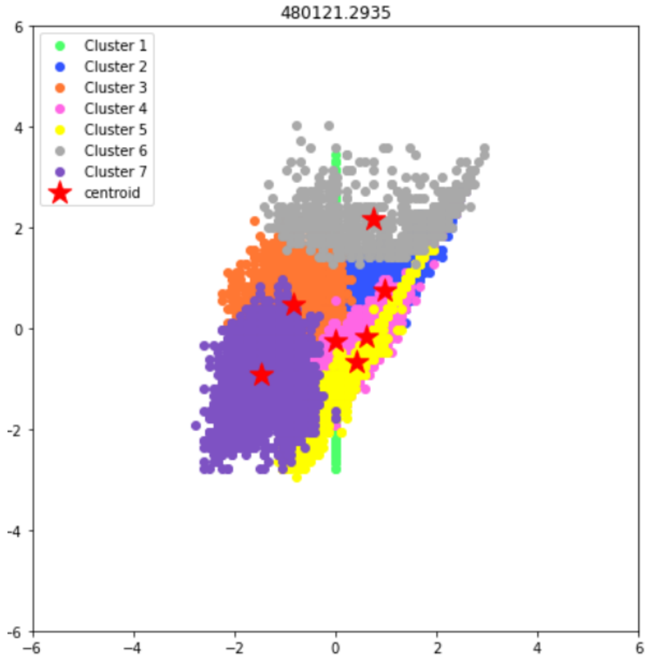
Clustering done using k=7 and graph plotted between Overall vs ST



Clustering done using k=7 and graph plotted between Overall vs CM



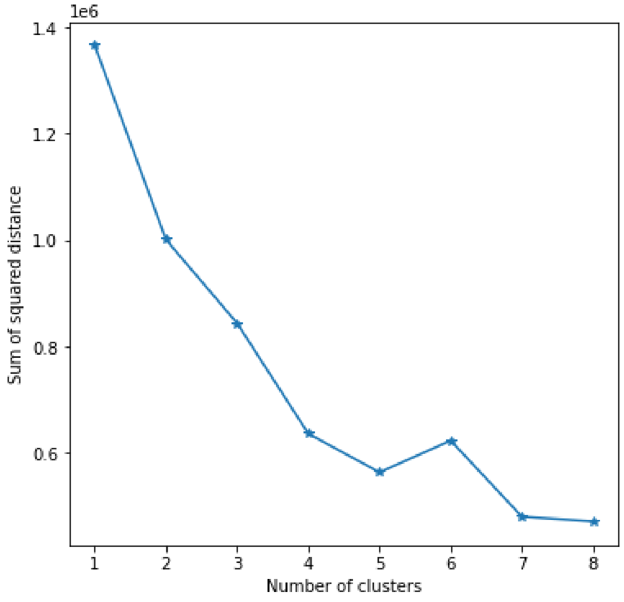
Clustering done using k=7 and graph plotted between Overall vs CB



## Elbow Method

1) In this method we will plot the sum of squared distance against the number of clusters.

2) We pick k at the spot where SSE starts to flatten out and form an elbow.

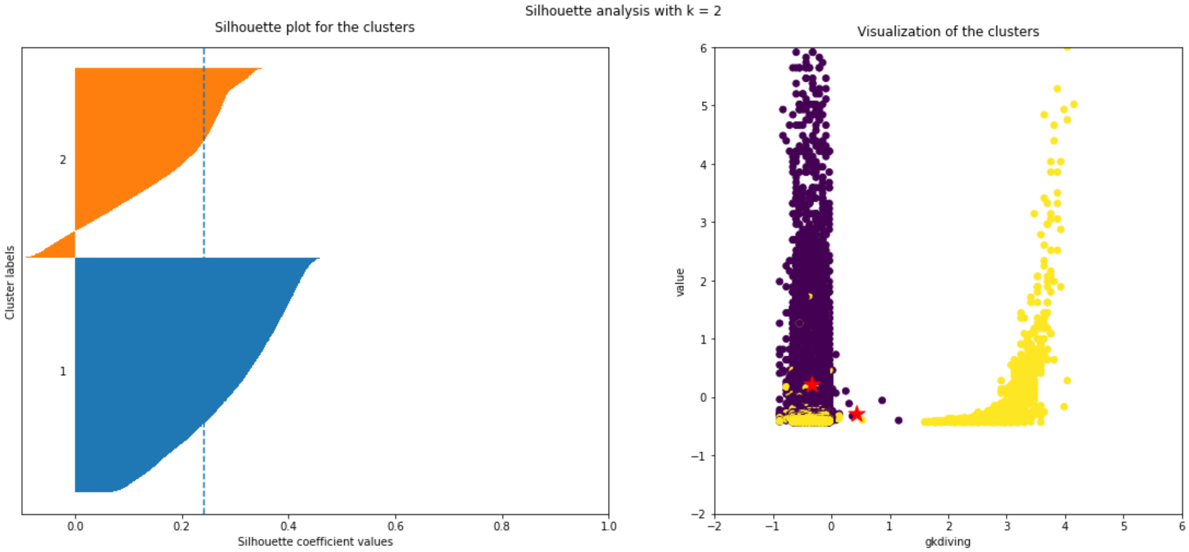


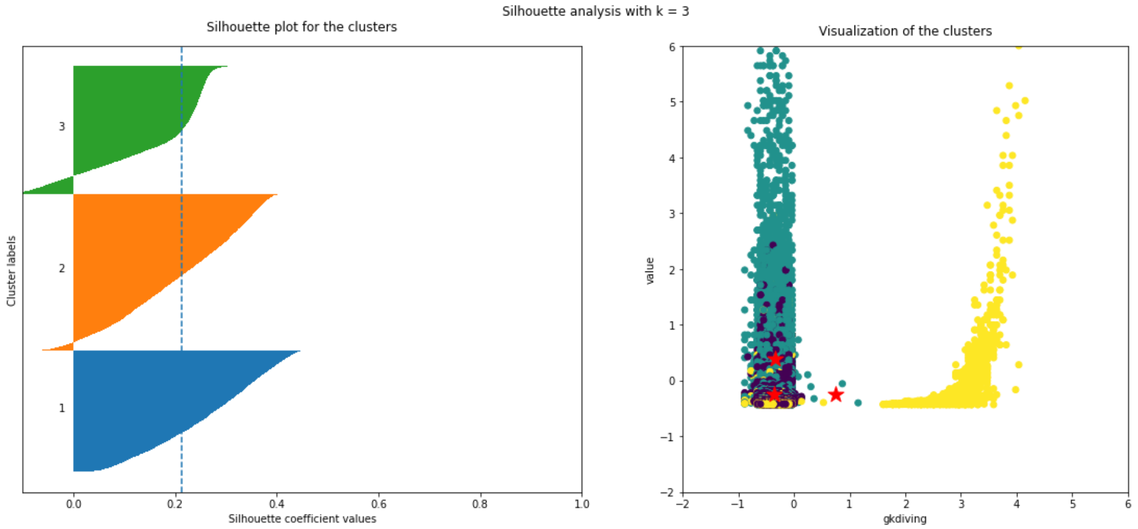
From the above graph we can see that the **best K is 4** because an elbow can be seen to be formed. The algorithm will give different graphs in different runs but K=4 **consistently** has an elbow in most of the plots.

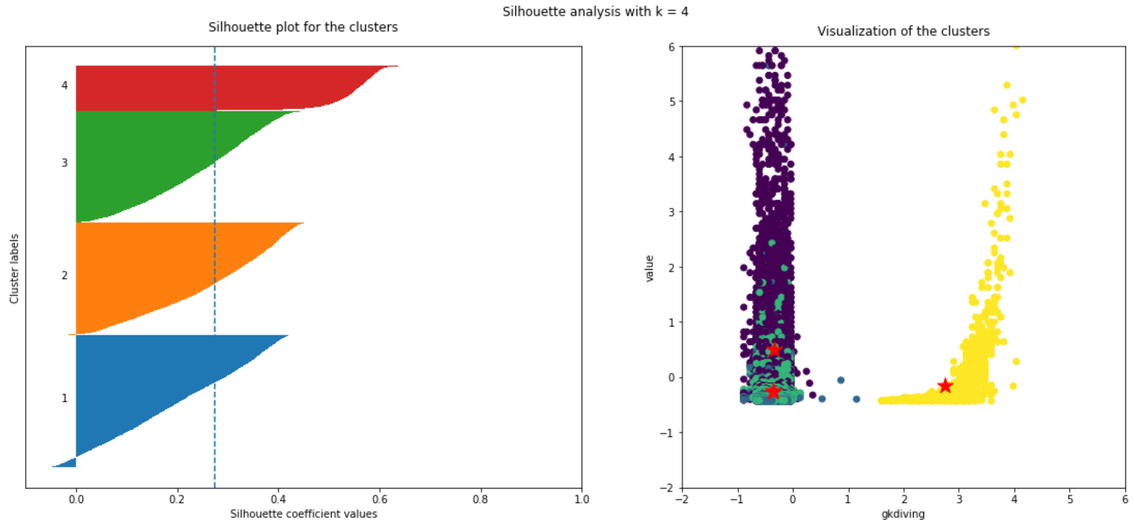
However, the elbow method is not the best method to select the number of clusters as the error function is mostly monotonically decreasing for all Ks.

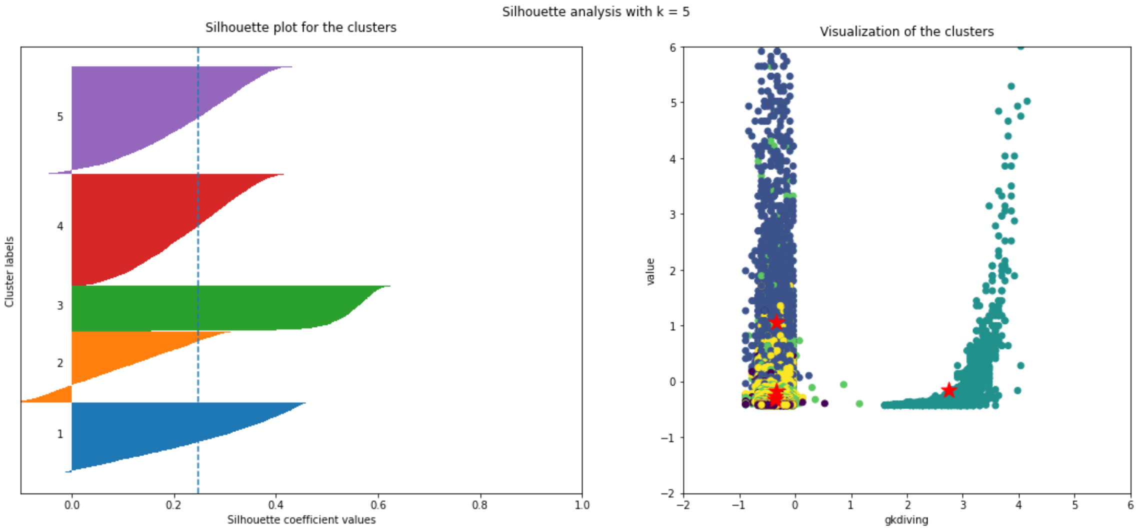
## Silhouette Analysis

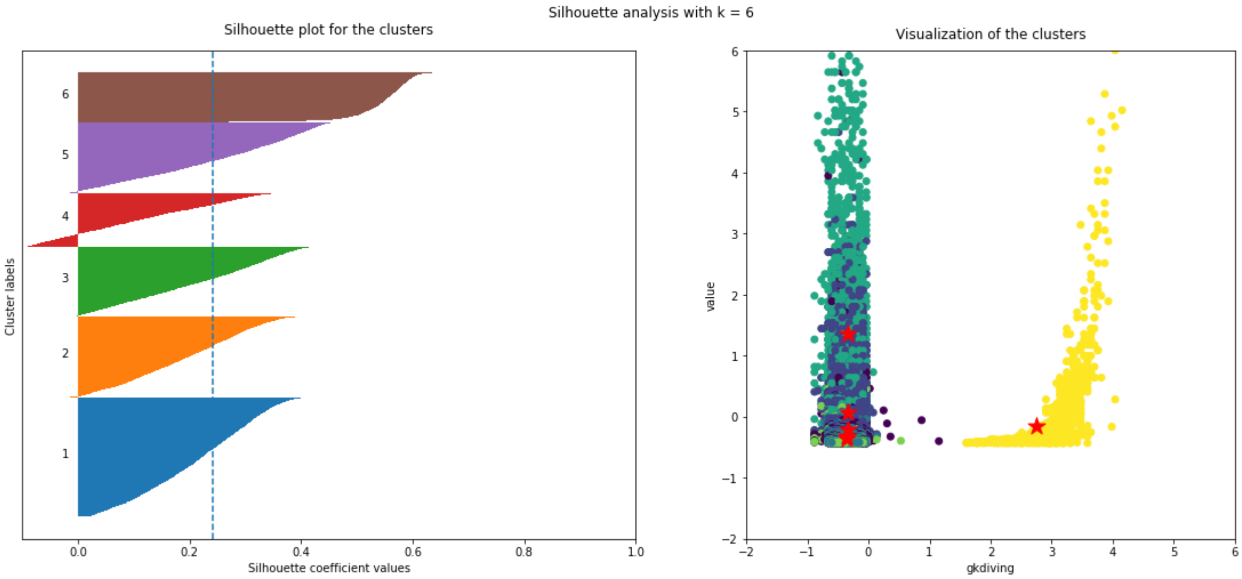
Silhouette analysis can be used to determine the degree of separation between clusters. For each sample we can find its average distance in the same and closest cluster and apply the required formula. The plots obtained for different k are:

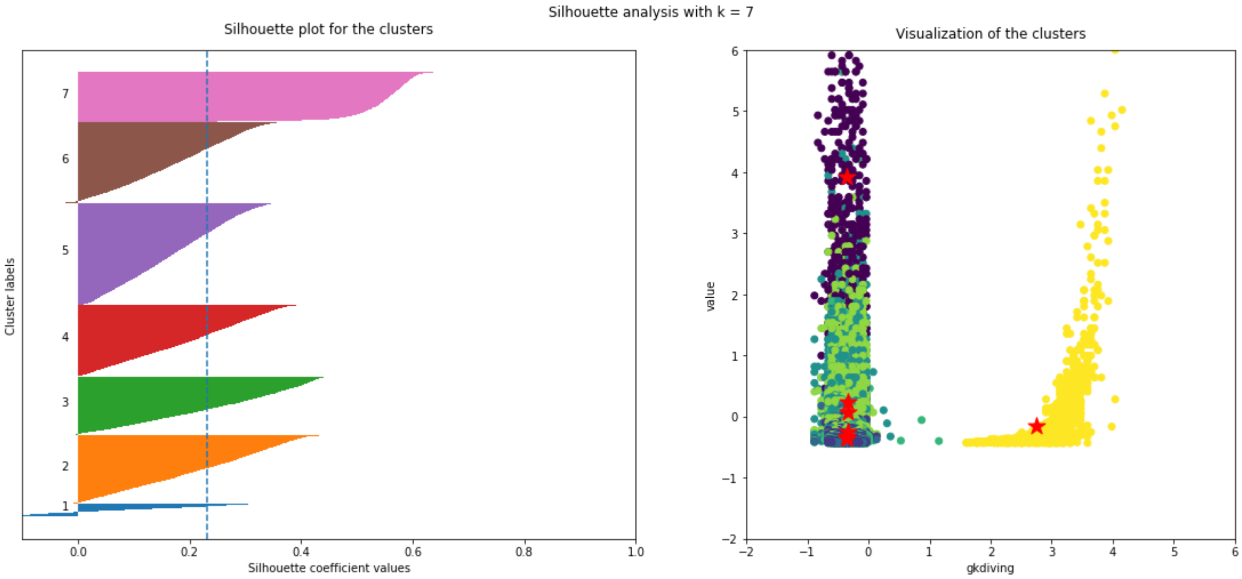


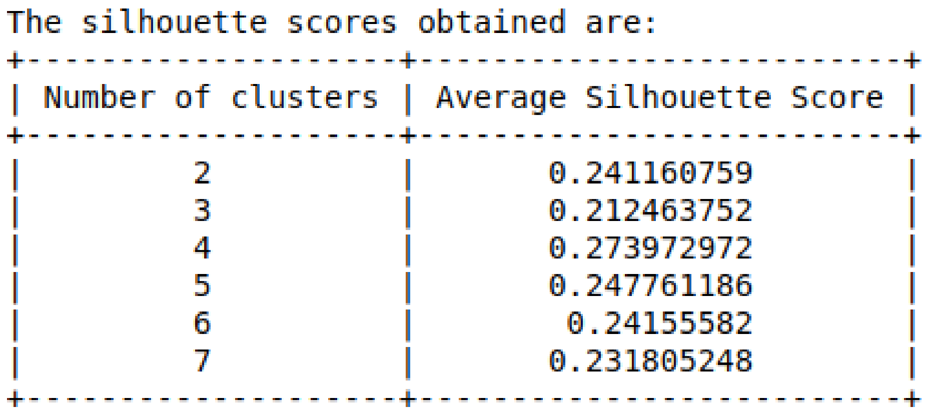

Silhouette analysis with k = 3
Silhouette plot for the clusters
Cluster labels
3
2
1
0.0
0.2
0.4
0.6
0.8
1.0
Silhouette coefficient values
Visualization of the clusters
value
6
5
4
3
2
1
0
−1
−2
−2
−1
0
1
2
3
4
5
6
gkdiving


Silhouette analysis with k = 4
Silhouette plot for the clusters
Cluster labels
4
3
2
1
0.0
0.2
0.4
0.6
0.8
1.0
Silhouette coefficient values
Visualization of the clusters
value
6
5
4
3
2
1
0
−1
−2
−2
−1
0
1
2
3
4
5
6
gkdiving


Silhouette analysis with k = 5
Silhouette plot for the clusters
Cluster labels
5
4
3
2
1
0.0
0.2
0.4
0.6
0.8
1.0
Silhouette coefficient values
Visualization of the clusters
value
6
5
4
3
2
1
0
−1
−2
−2
−1
0
1
2
3
4
5
6
gkdiving







```
The silhouette scores obtained are:
+--------------------+-------------------------+
| Number of clusters | Average Silhouette Score |
+--------------------+-------------------------+
|         2          |       0.241160759       |
|         3          |       0.212463752       |
|         4          |       0.273972972       |
|         5          |       0.247761186       |
|         6          |        0.24155582       |
|         7          |       0.231805248       |
+--------------------+-------------------------+
```

From the above runs, it can be seen that the silhouette average score **for 4 is the best** so k = 4 is the optimal number of clusters. Between 3 and 5 **sometimes 3 performs better and sometimes 5 performs better**. The reason for this might be some inherent bias in the data due to which the random initial centroids give different results.

Important points

- Kmeans assumes spherical shapes of clusters and doesn't work well when clusters are in different shapes such as ellipsoids.
- In case of overlapping between clusters, k means doesn't have any intrinsic measure to determine for which cluster to assign to the data point.

# 3. Hierarchical Clustering

## Agglomerative Hierarchical Clustering

Details about the implementation :

**Preprocessing**: Same data cleaning was done as last time. Adding to that, Principal Component Analysis was used later on to lower the number of dimensions of the data. We ran the clustering algorithm for n_components in range(5,20). After applying PCA we plotted the dendrogram for each and manually figured out the correct number of clusters in each case. The method for selecting the correct number of clusters is that the distance between two consecutive horizontal lines is high. Finally we apply agglomerative clustering using the current value of the number of clusters and use inter-cluster and intra-cluster similarity to choose between the best among all the models. (Intra cluster distance should be low and Inter cluster distance should be high)

**Results**:

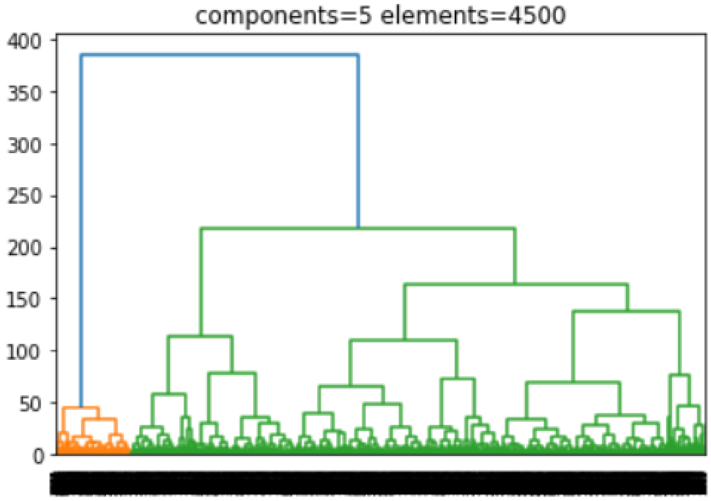
When the number of clusters were 3 , the best results were with the PCA component as 5.

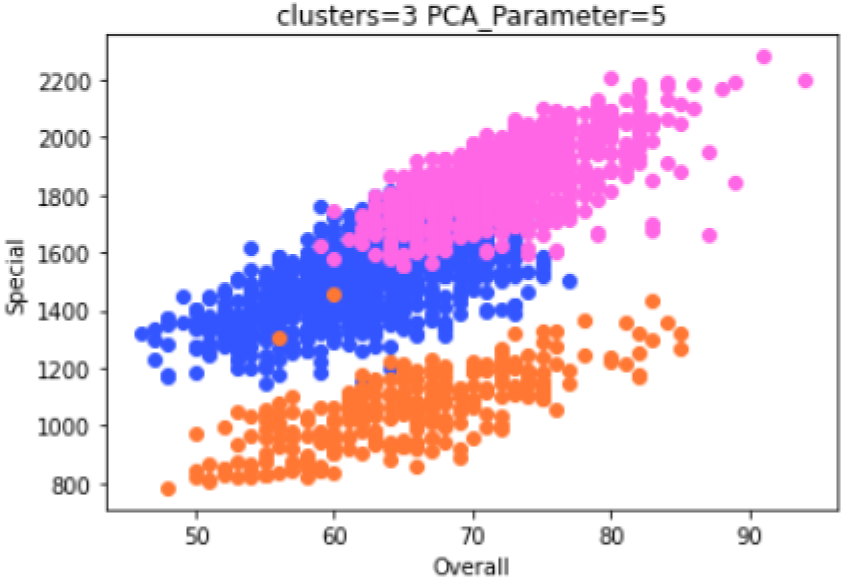
When the number of clusters were 4 , the best results were with the PCA component as 10.

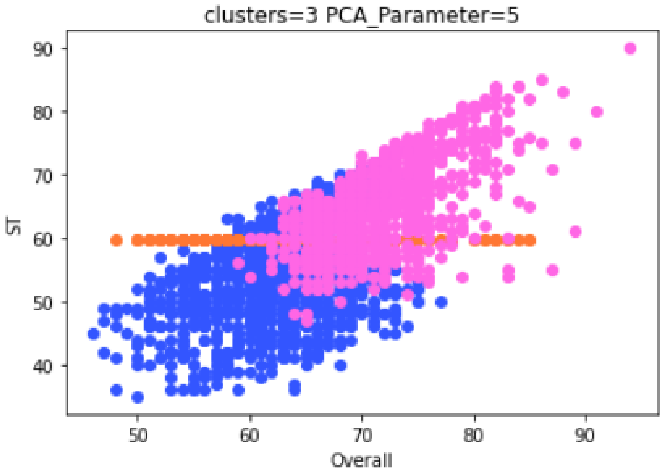
**Sample plots** for 3 clusters and 5 PCA:

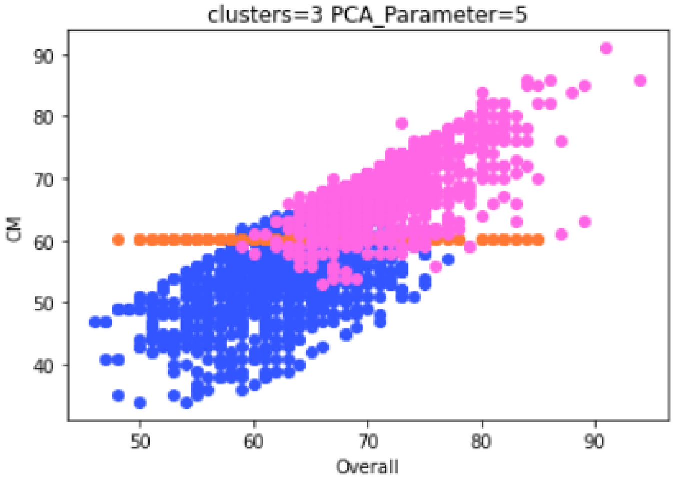
Intra-cluster distance = 6.444101

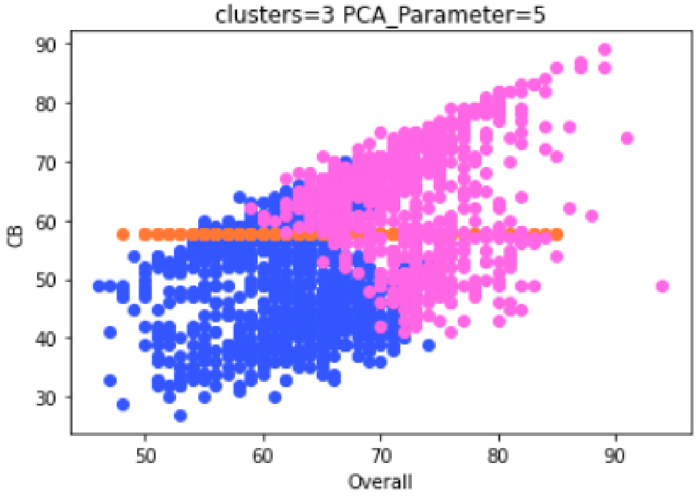
Inter-cluster distance = 12.57303

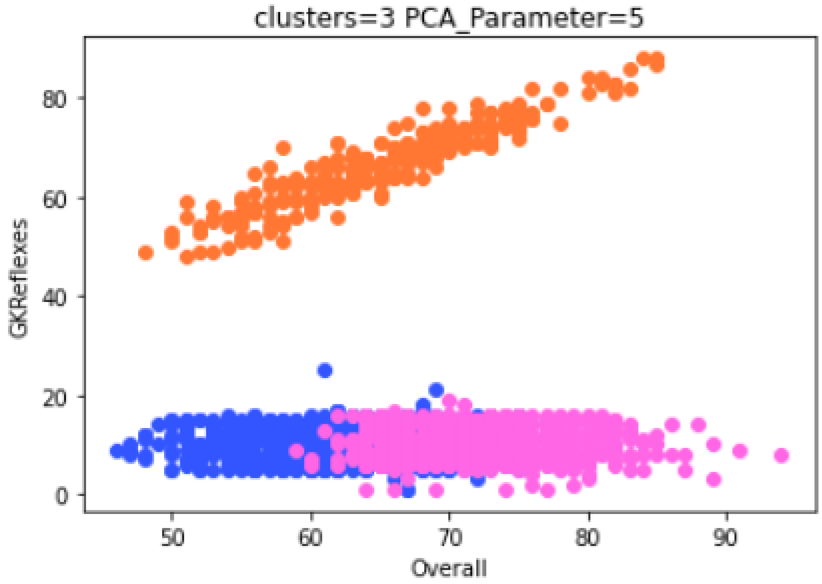





clusters=3 PCA_Parameter=5


clusters=3 PCA_Parameter=5


clusters=3 PCA_Parameter=5



```
6.444101270980227  12.573034102967535
```

**Observations**: In the plots generated, separate clusters were formed for goalkeepers. Another cluster formed for players with overall less than 70 and poor defense. Other players with low overall were in another cluster. Another cluster was formed for the remaining best players.

## Divisive Hierarchical Clustering

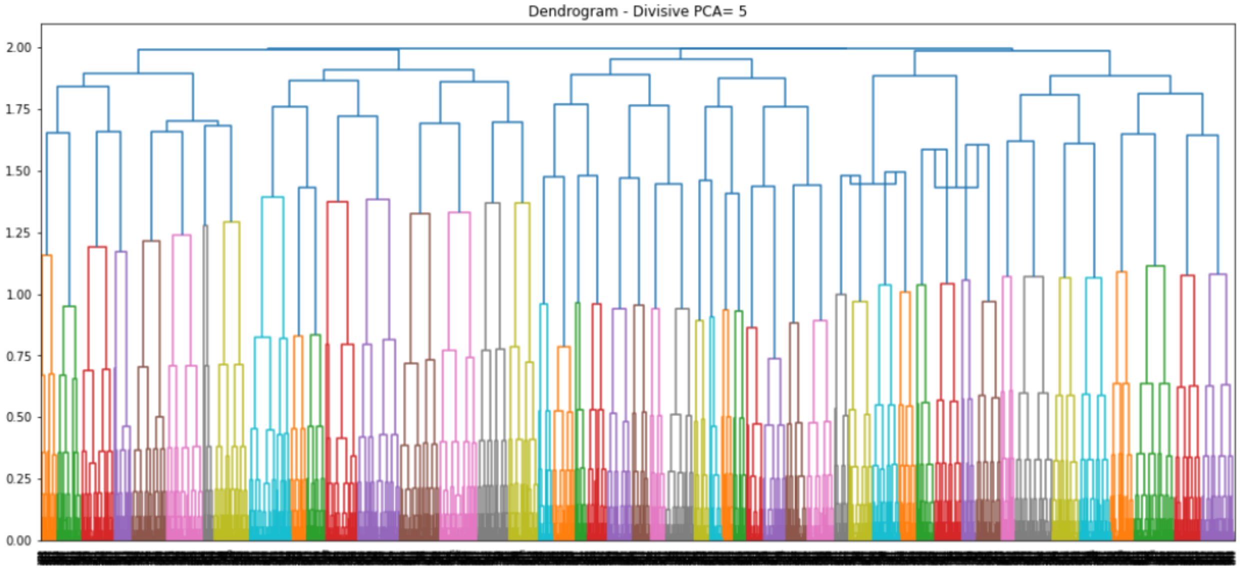
**Preprocessing**: Similar data cleaning was done as before. This tim after applying PCA we plotted the dendrogram for each and manually figured out the correct number of clusters in each case. The metric for plotting the dendrogram were euclidean, cosine, cityblock, l1, l2, manhattan. After looking at the clusters, the clusters made by cosine distance as the metric make the most sensible and produce the best results. Finally we applied divisive clustering and we take number of clusters to be 2, 3, 4 use inter-cluster and intra-cluster similarity to choose between the best among all the models (intra cluster distance should be low,  inter cluster distance should be high).

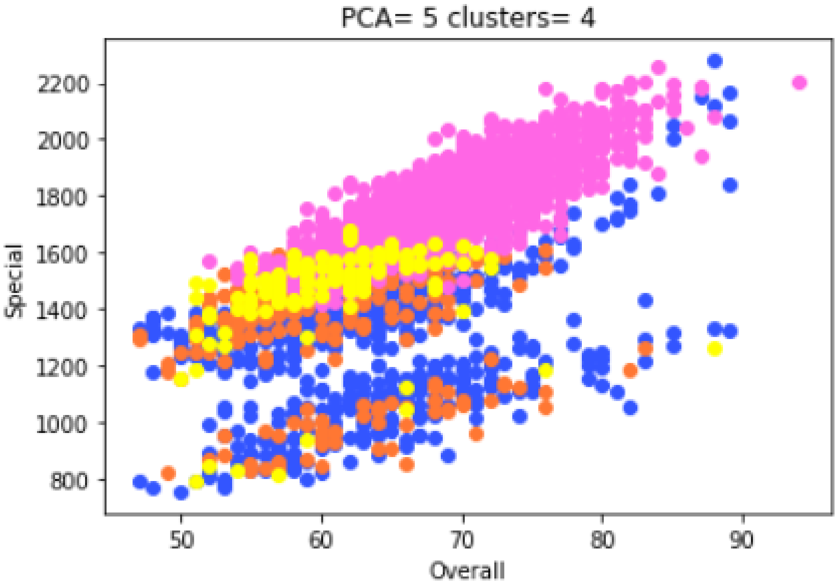
**Results**:

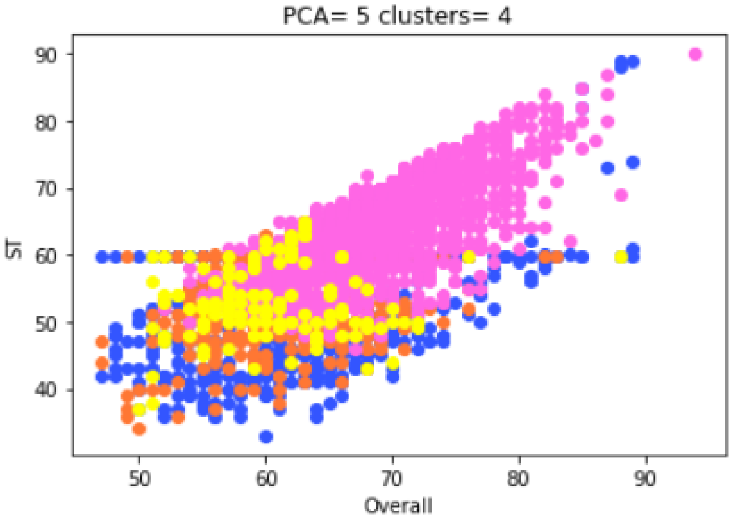
We tried with different numbers of clusters and in each case best PCA values differed. For 2 clusters we got the best results with PCA as 15, for 3 clusters we got best results with PCA as 20, and for 4 clusters we got the best results with PCA as 5.

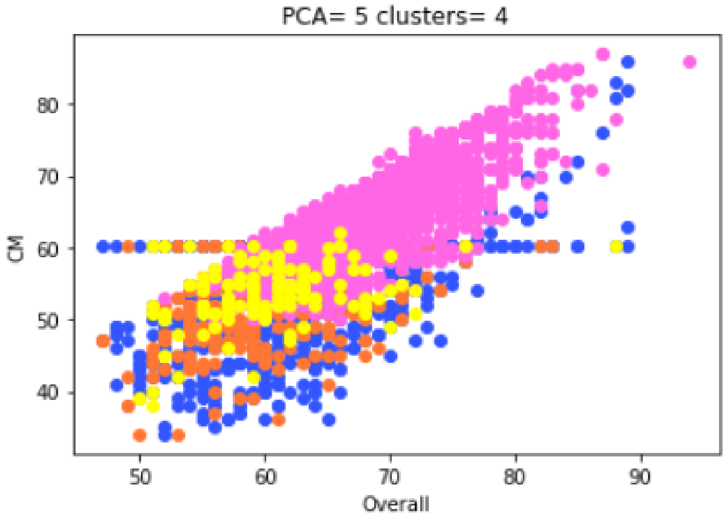
**Sample for plots** for clusters = 4 and PCA = 5 are below:

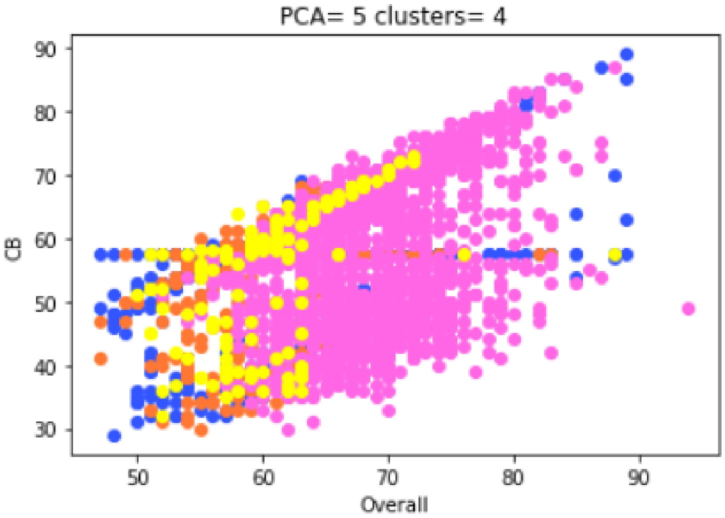
For this case, Intra-cluster distance = 0.7209505 and Inter-cluster distance = 0.9892289

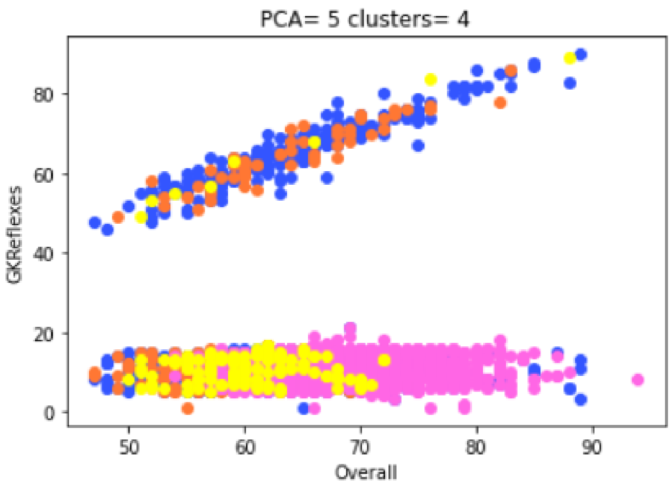





PCA= 5 clusters= 4
ST
90
80
70
60
50
40
50
60
70
80
90
Overall


PCA= 5 clusters= 4
CM
80
70
60
50
40
50
60
70
80
90
Overall


PCA= 5 clusters= 4
CB
90
80
70
60
50
40
30
50
60
70
80
90
Overall



```
0.7209505733266407    0.9892289457362677
```

**Observations** :

One of the clusters included the weakest players which are not goalkeepers, with the only good players being centerbacks. Another cluster has the best players which are not goalkeepers.

This time some of the clusters were vague with not so clear explanations for them. In many runs, the algorithms gave results which had similar clusters and there is not much to differentiate between them.

# Comparison of divisive and agglomerative strategy

● Agglomerative uses the bottom up strategy whereas divisive uses the top down strategy.

● The clusters in **agglomerative were almost perfect for a human to make sense** out of, whereas we did get some interesting insights from the divisive method but the results were not as good..

● Euclidean distance **did not work well for the divisive strategy** whereas it **worked very well for the agglomerative strategy**.

● **Agglomerative was able to differentiate among players based on forward, defense, goalkeeping** but **Divisive differentiated between clusters on the basis of skill and overall**.

# 4. DBSCAN

Few Details about the implementation :

**Preprocessing**: While preprocessing the data, we have converted all objects values to numerical float values so they can be used for kmeans, imputed all missing values by the mean of the same attribute, and standardized the entries attribute wise to prevent large vectors from dominating. This time attributes with very low distance from each other were also removed (eg ST and CF). Principal Component Analysis was used later on to lower the number of dimensions of the data. We ran the clustering algorithm for n_components in range(10,20).

**Experiments done to arrive at the final eps and minPts**

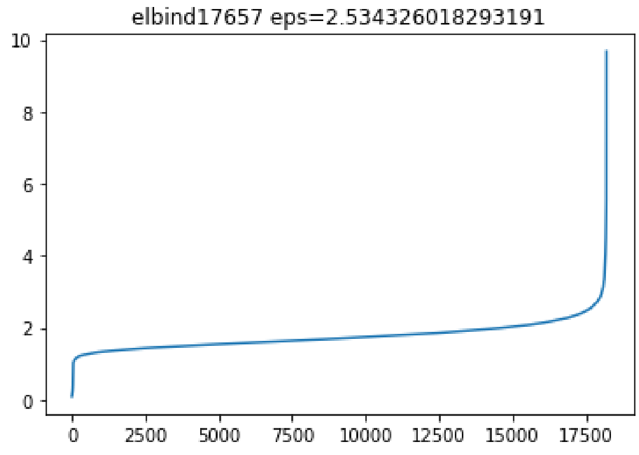
For finding epsilon-

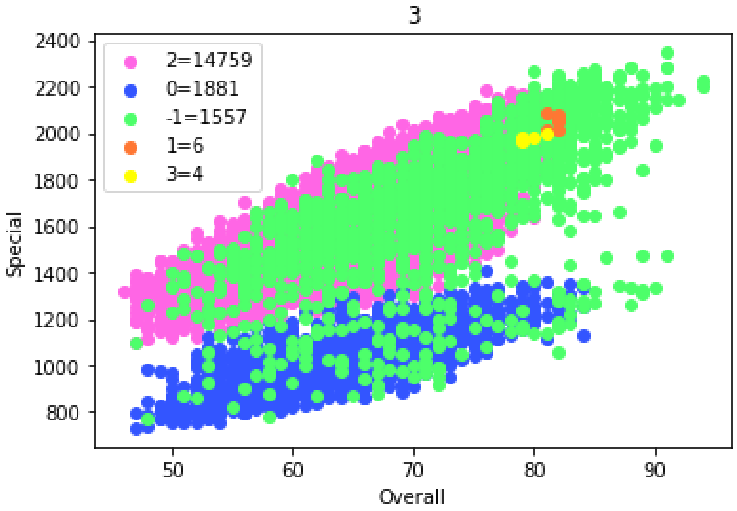
- We run the algorithm of dbscan for minPts ranging from (5,15) and for each value of minPts we plot the nearest neighbour distances graph.
- In that grah we calculate the elbow and use that distance as epsilon.

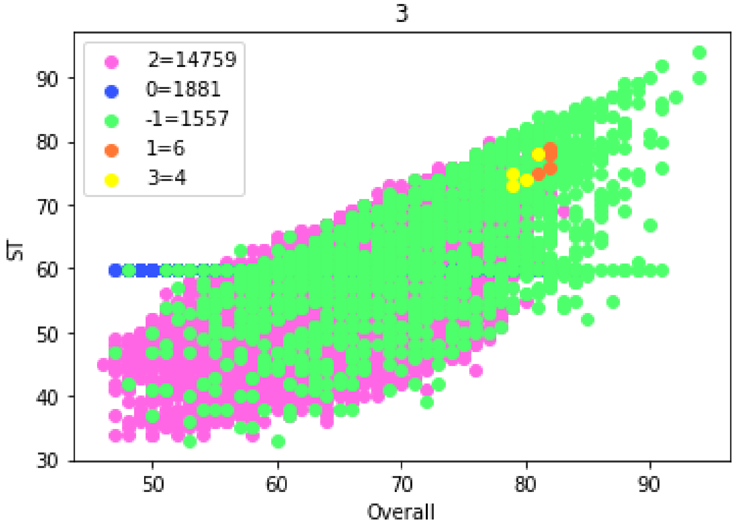
Finally we apply the DBSCAN current value of the minPts , epsilon and use inter-cluster and intra-cluster similarity to choose between the best among all the models.

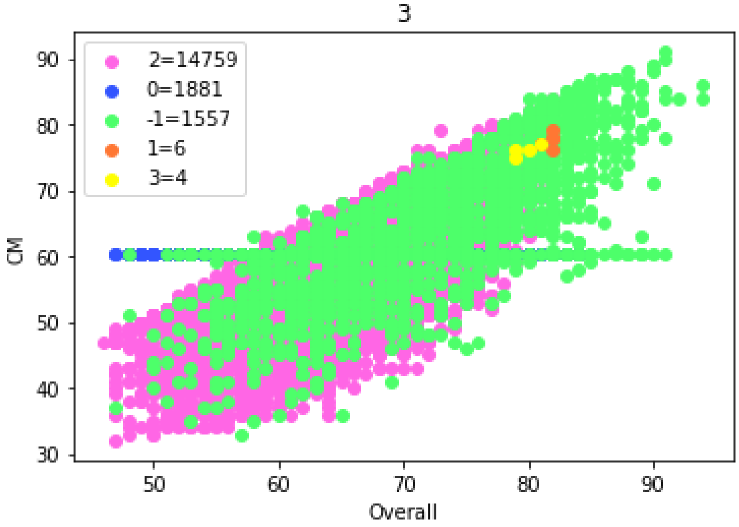
**Results**:

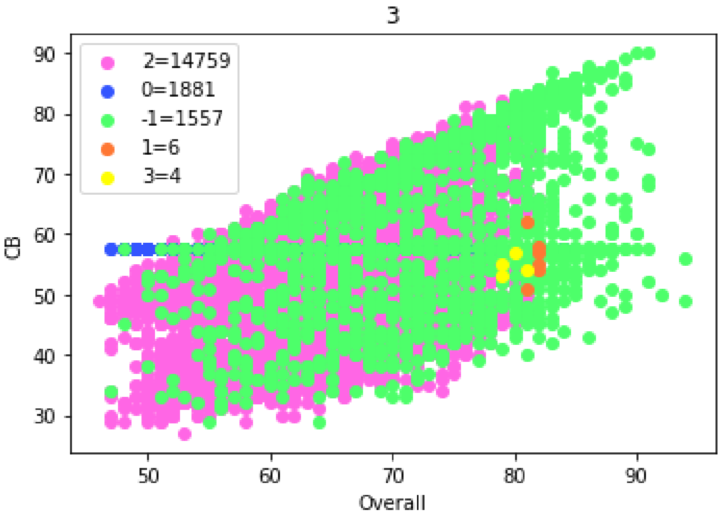
The best results were when minPts = 5 and epsilon = 2.534326 and the number of dimensions were 20. First we used the elbow to get the correct epsilon for minPts=5. Then using these minPts and epsilon we used DBSCAN.

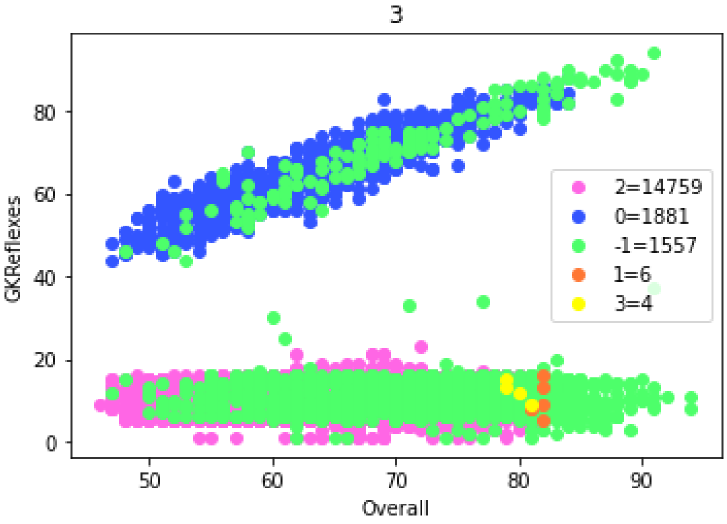



3
2=14759
0=1881
-1=1557
1=6
3=4
Special
2400
2200
2000
1800
1600
1400
1200
1000
800
50
60
70
80
90
Overall


3
2=14759
0=1881
-1=1557
1=6
3=4
ST
90
80
70
60
50
40
30
50
60
70
80
90
Overall


3
2=14759
0=1881
-1=1557
1=6
3=4
CM
90
80
70
60
50
40
30
50
60
70
80
90
Overall





For the above cases,

inter cluster distance =12.898034611195163 and intra cluster distance = 4.370497074443179

**Observations :**

**These observations are based from the above plots and the category to cluster mapping was consistent throughout numerous runs of the algorithm.**

- Blue cluster is for the goalkeeper.
- The **green ones were the outliers**.
- The red ones include some of the best strikers.
- The yellow ones include good defenders.
- The **players in pink do not have any specific characteristics**.

DBSCAN is density based and hence it is not very good for the dataset since this dataset is very dense and does not have a very clear separation anywhere. That is why most of the players got into only one cluster that is the pink one.

**Conclusion**

- The **best method is agglomerative**. The clusters formed by agglomerative are the **cleanest** with clear separation and also hold a lot of meaning for us humans. We can clearly see clusters being split into goalkeeper and forward with some split also between midfielders and defenders.
- DBScan and divisive tend to create one large cluster which has the bulk of data and other smaller clusters. It also leaves a lot of outliers which is not a good thing
- In K Means the cluster separation is not the best.
- In case of overlapping between clusters, k means doesn’t have any intrinsic measure to determine for which cluster to assign to the data point.