Anthony Gemignani

Hima Spandana Barla

Pramathesh Bhaskarbhai Shukla

Dsc 478 Final Report

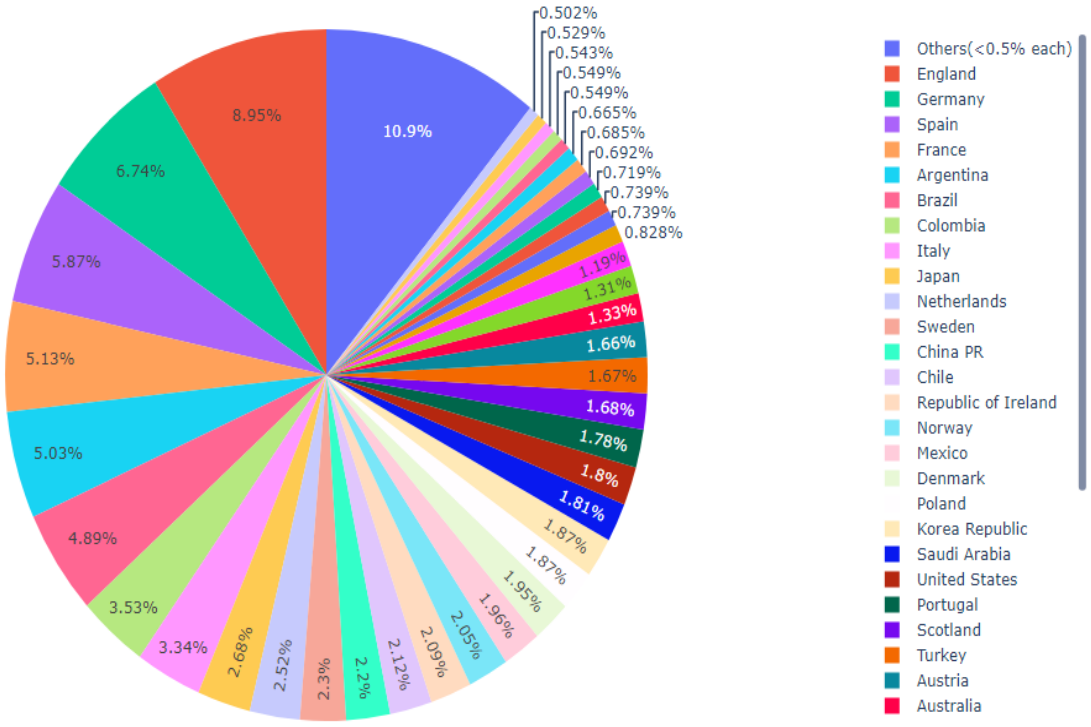
Data Preprocessing, Exploratory Analysis and Linear Regression

The FIFA World Cup is an international association football tournament contested by senior men's national teams from members of FIFA, the sport's global governing body. Since the first tournament in 1930, the championship has been awarded every four years, with the exception of 1942 and 1946, when it was not held due to World War II. Dataset is obtained from kaggle https://www.kaggle.com/karangadiya/fifa19 and consists of 18207 records with 89 variables with categorical and numerical values and descriptions for exemplary variables given below.

Exploratory Data Analysis

After removing a few inaccurate and unnecessary variables from the table such as Name, ID and a few other variables, I imported the libraries which were required to perform the data modeling and features. Started with Exploratory data analysis where a few pie charts are drawn to identify a few key points in the dataset.

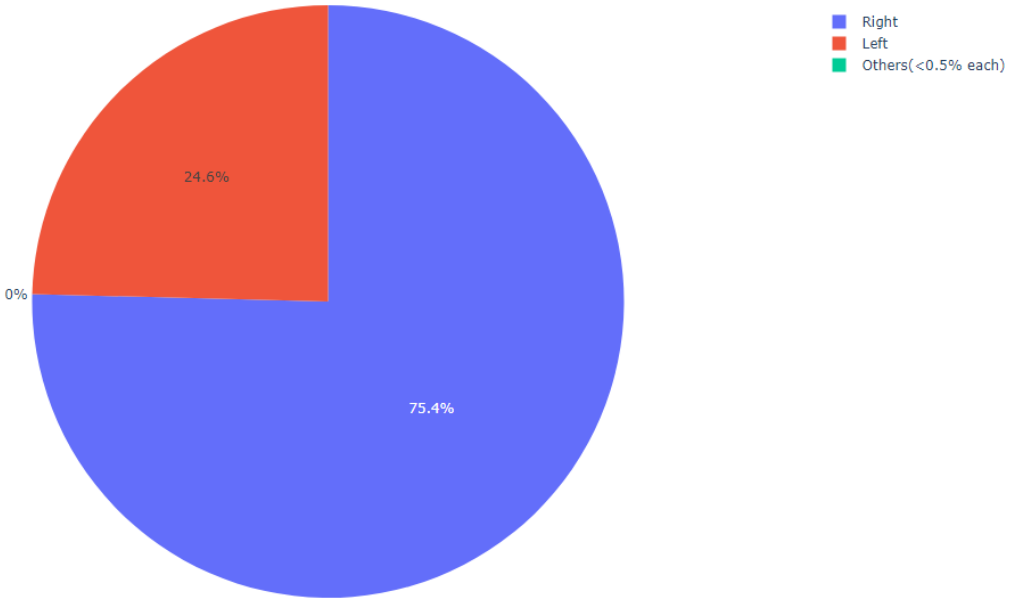
Number of players by Nationality



In the dataset most of the players are representing England with a percentage of 8.95% and the second biggest team after England is Germany.Almost a bunch of countries have less than 1% of players playing football.

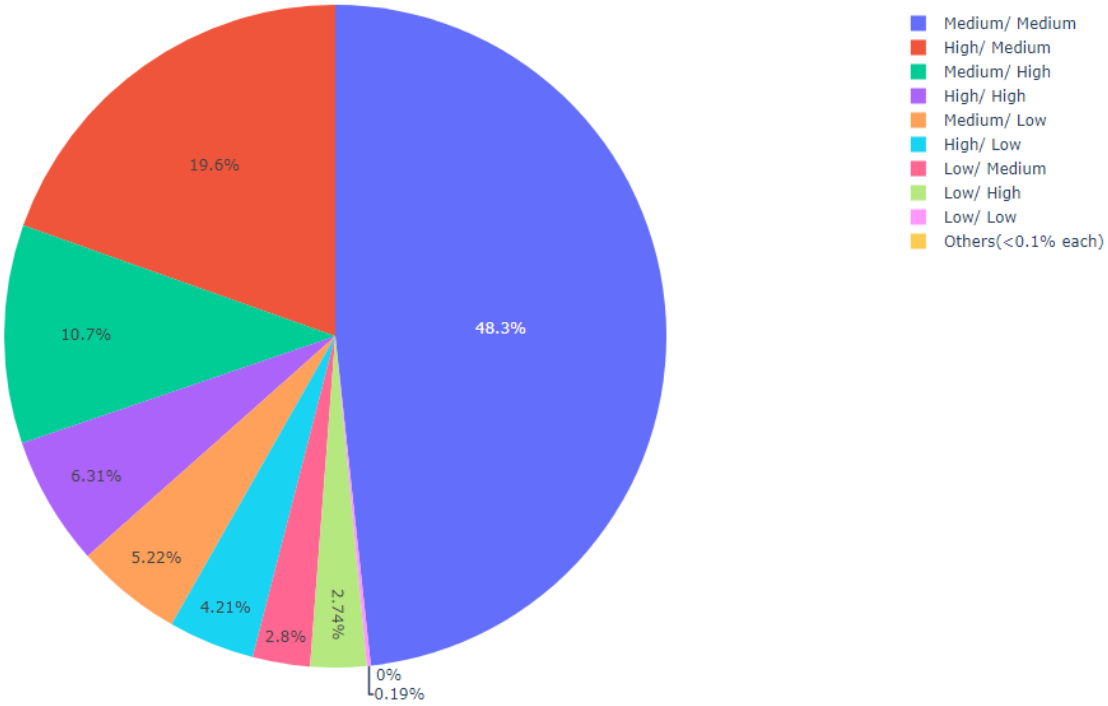
Foot Preference

Most of the players tend to use their right foot while playing and only 24.6% of the players prefer their left foot while kicking the ball.

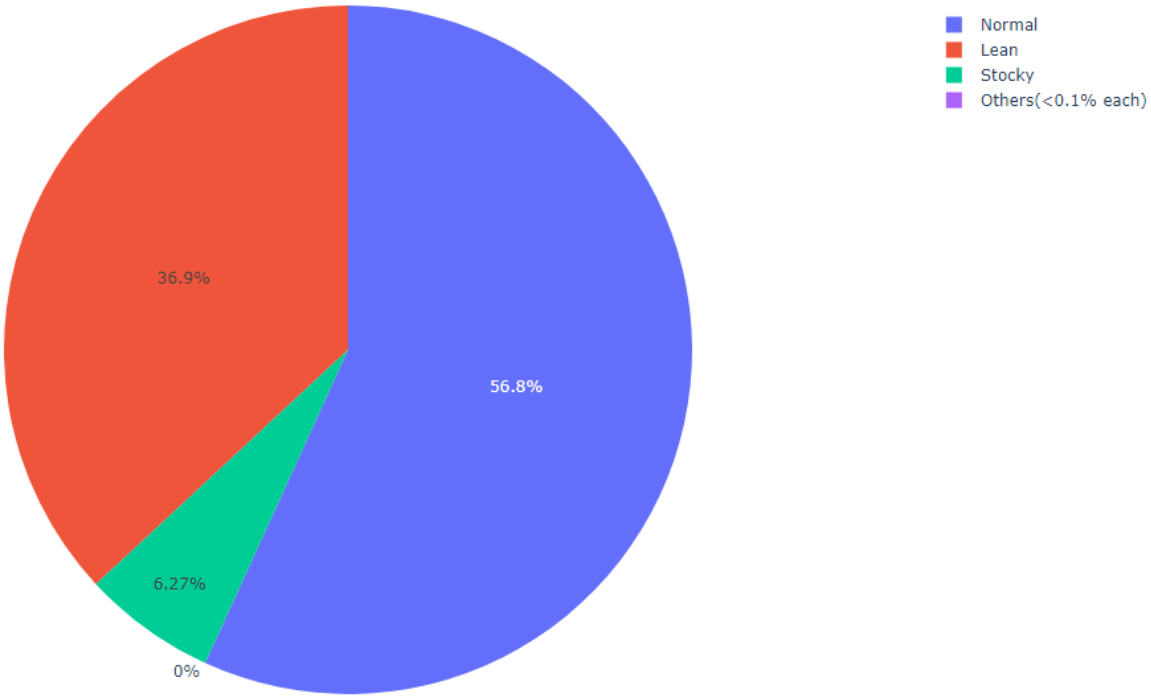


Number of players by work rate

Work rate refers to the extent to which a player contributes to running and chasing in a match while not in possession of the ball.Work rate is generally indicated by the distance covered by a player during a match. Most of the players are in the medium range.

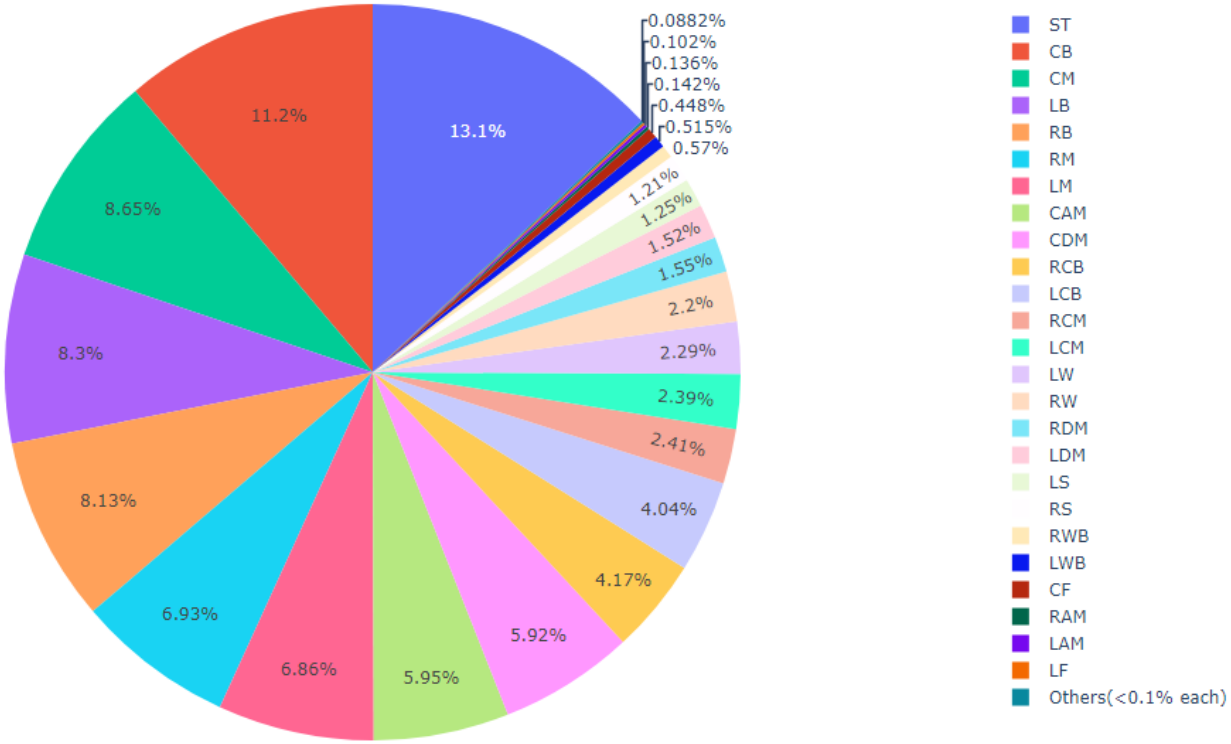


Number of players by body type

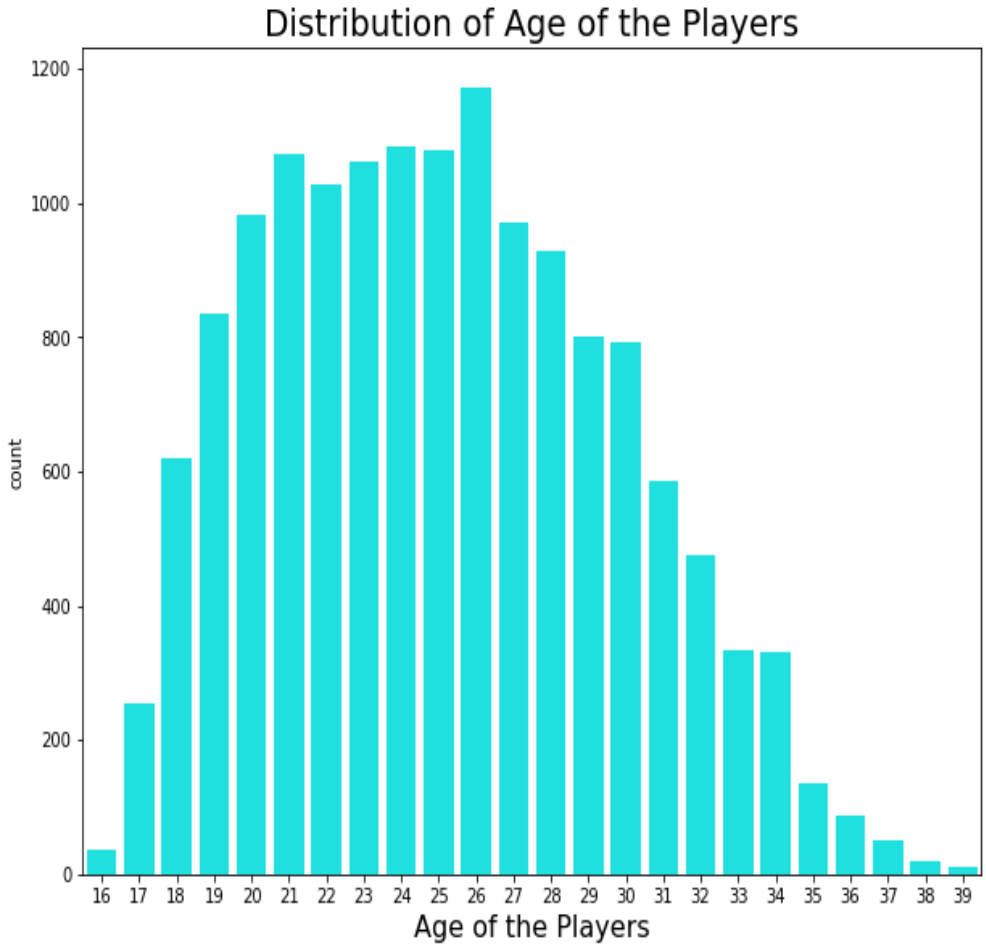


Most of the players in the dataset have normal body type which is around 56.8%. 36.9% players have a lean body and 6.27% players have a stocky type of body.

Number of players by position



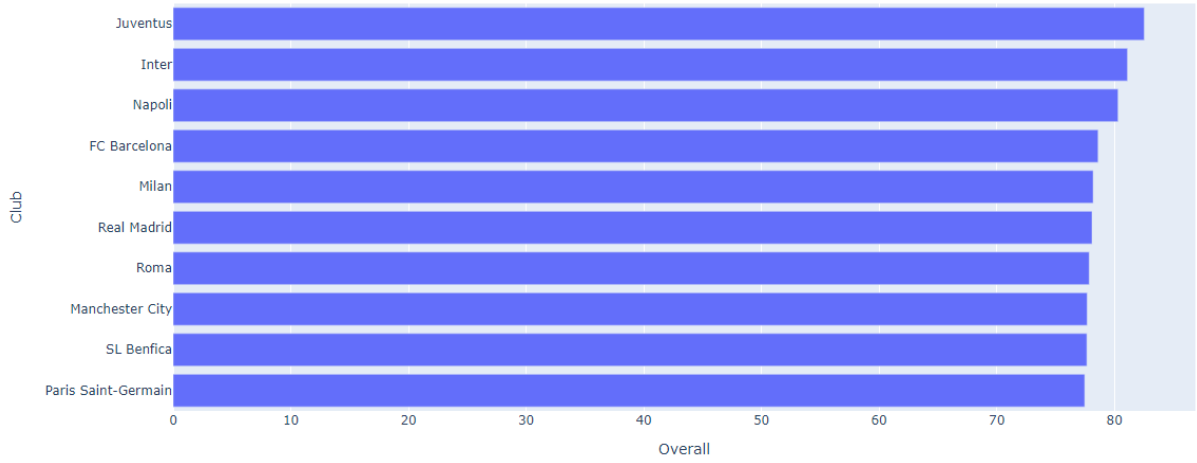
In order from left to right, they are: the Left Tackle (LT), Left Guard (LG), Center (C), Right Guard (RG) and Right Tackle (RT). It is their job to either pass block for the QB so he has time to throw or run block for the RB or FB. In gridiron football, a cornerback (CB) is a member of the defensive backfield or secondary. Most of the time, cornerbacks cover receivers, but they also blitz and defend against offensive running plays like sweeps and reverses. Hard tackles, interceptions, and deflected forward throws all result in turnovers.CB is almost 11.2%.



Most of the players who are playing professional football matches at the international level are in the range of 20-30 years of age. More than 1000 players represented by the teams are 26 years of age.

Top 10 teams with highest player's average Overall rating

Juventus has the best player overall rating in the complete year and remained consistent throughout the year and second on the chart is Inter who is competing with first position. Napoli and FC Barcelona hold third and fourth position.



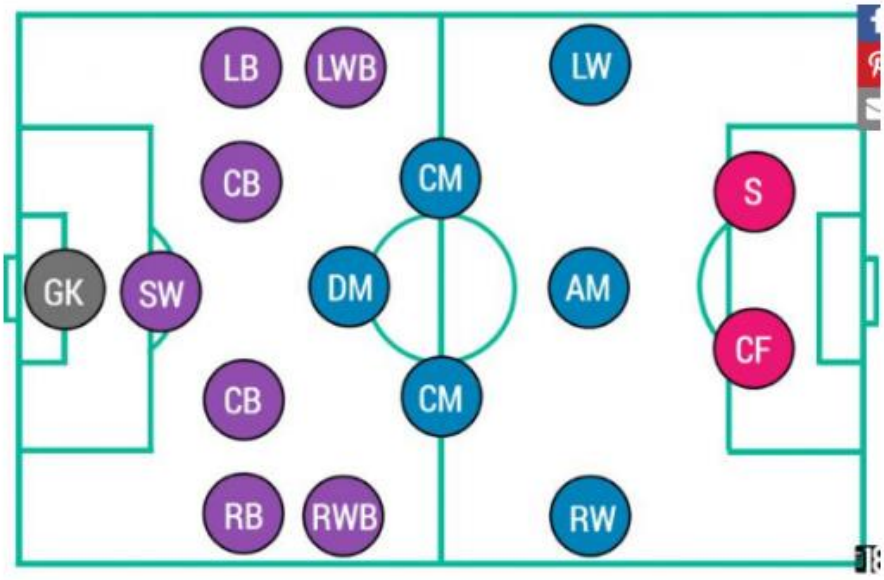
Positional Features

Each position in the ground should have the following positional features in order to play and win the match.

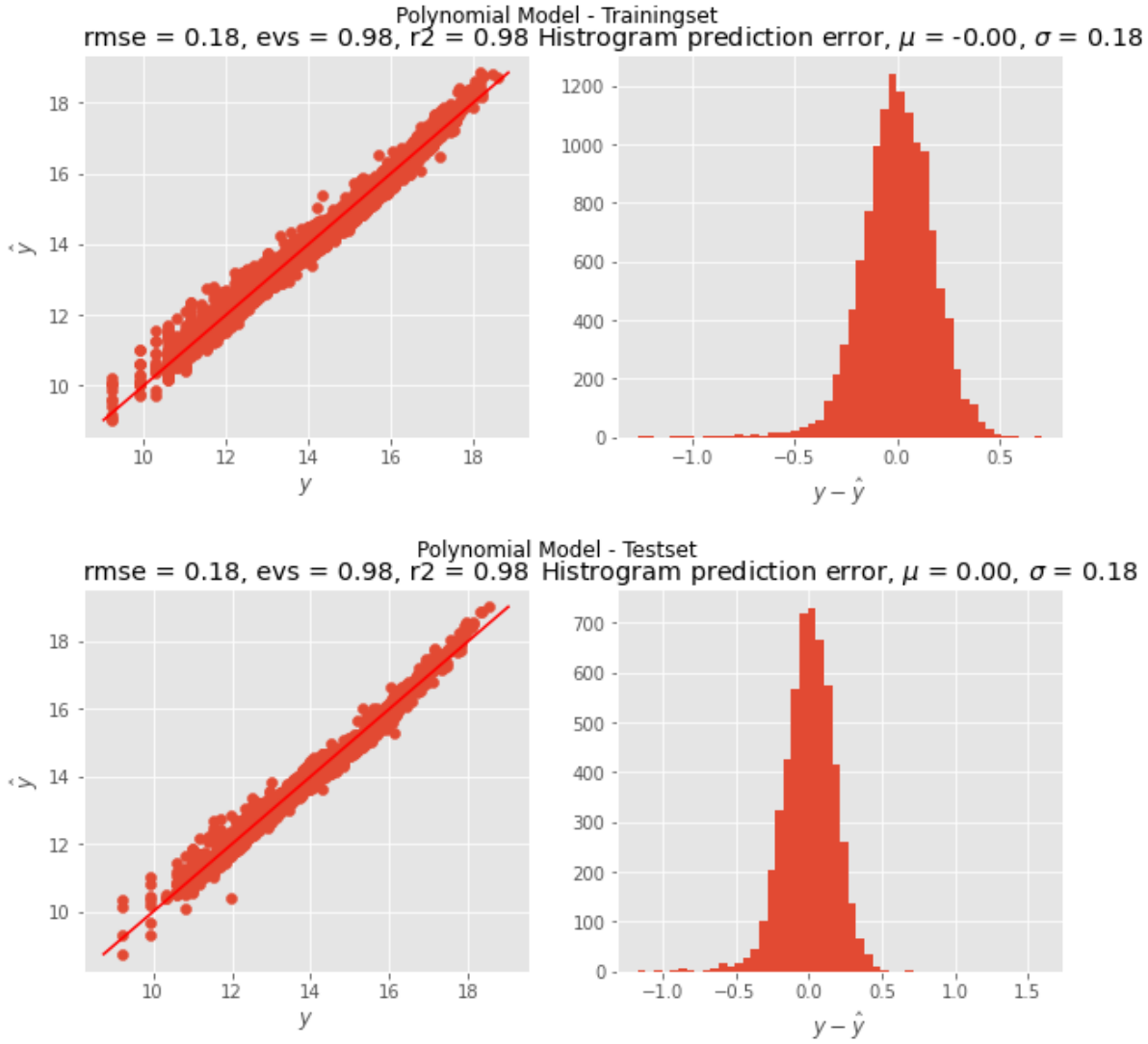
]:

	CAM	CB	CDM	CF	CM	LAM	LB	LCB	LCM
0	Balance	Strength	Stamina	Agility	Balance	Agility	SprintSpeed	Strength	Stamina
1	Agility	Jumping	Aggression	Balance	ShortPassing	Balance	Acceleration	Jumping	ShortPassing
2	Acceleration	StandingTackle	Strength	Acceleration	Agility	SprintSpeed	Stamina	StandingTackle	Balance
3	SprintSpeed	Aggression	ShortPassing	SprintSpeed	Stamina	Acceleration	Balance	Aggression	Agility
4	BallControl	HeadingAccuracy	Jumping	Dribbling	Acceleration	Dribbling	Agility	HeadingAccuracy	BallControl

5 rows × 26 columns



Linear Regression



After slicing the data into training and testset Linear regression is performed individually on each set. The RMSE value of the training and test set obtained is 0.18 and histogram prediction is (Mu = 0.00)Regression performs the task to predict a dependent variable value (Value) based on a given independent variable and the graph appears to be normal and uniform withalomsot no outliers.

Data Cleaning- Preprocessing, PCA and Rocchio Methods.

Data Cleaning and Preprocessing:

I started the initial step of importing all the libraries. I then read in the data and set the working directory. I started preprocessing of data by removing unwanted columns from the data frame. The unit of wage was in the thousands. In the cell I cleaned the wage cool, from the signs that indicated currency and amount. The unit of value column is in million. In this cell I cleaned wage cool from the signs that indicate currency and amount. Then I Saved the clean df into csv to use it in the slide deck file to not clean it again there. Took the most important columns to work with them instead of taking 40+ columns and not using them all. Then I checked for na values and checked if there was any left.

```
#the unit of Wage column is in thousand
#in this cell i'm cleaning the Wage cool from the signs that indicate currencey and amount
df_clean.Wage = df_clean.Wage.apply(lambda x: str(x).replace('€',''))
df_clean.Wage = df_clean.Wage.apply(lambda x: str(x).replace('K',''))
df_clean.Wage = df_clean.Wage.astype(float)
```

```
#the unit of Value column is in million
#in this cell i'm cleaning the Wage cool from the signs that indicate currencey and amount
df_clean.Value = df_clean.Value.apply(lambda x: str(x).replace('€',''))
df_clean.Value = df_clean.Value.apply(lambda x: str(x).replace('K','000'))
df_clean.Value = df_clean.Value.apply(lambda x: str(x).replace('M','00000') \
                                      if '.' in str(x) else str(x).replace('M','000000') )
df_clean.Value = df_clean.Value.apply(lambda x: str(x).replace('.',''))
df_clean.Value = df_clean.Value.astype(float)
df_clean.Value = df_clean.Value / 10**6
```

```
#taking the most important columns to work with them instead of taking 40+ cols and not use them all
df_sal = df_clean[['Name', 'Age', 'Overall', 'Potential','Club', 'Value', 'Wage',
                   'Preferred Foot', 'Position','Stamina','ShotPower','International Reputation']].copy()
df_sal.head()
```

	Name	Age	Overall	Potential	Club	Value	Wage	Preferred Foot	Position	Stamina	ShotPower	International Reputation
0	L. Messi	31	94	94	FC Barcelona	110.5	565.0	Left	RF	72.0	85.0	5.0
1	Cristiano Ronaldo	33	94	94	Juventus	77.0	405.0	Right	ST	88.0	95.0	5.0
2	Neymar Jr	26	92	93	Paris Saint-Germain	118.5	290.0	Right	LW	81.0	80.0	5.0
3	De Gea	27	91	93	Manchester United	72.0	260.0	Right	GK	43.0	31.0	4.0
4	K. De Bruyne	27	91	92	Manchester City	102.0	355.0	Right	RCM	90.0	91.0	4.0

```
df_sal = df_sal.drop(['Name'], axis=1)
df_sal.head()
```

	Age	Overall	Potential	Club	Value	Wage	Preferred Foot	Position	Stamina	ShotPower	International Reputation
0	31	94	94	FC Barcelona	110.5	565.0	Left	RF	72.0	85.0	5.0
1	33	94	94	Juventus	77.0	405.0	Right	ST	88.0	95.0	5.0
2	26	92	93	Paris Saint-Germain	118.5	290.0	Right	LW	81.0	80.0	5.0
3	27	91	93	Manchester United	72.0	260.0	Right	GK	43.0	31.0	4.0
4	27	91	92	Manchester City	102.0	355.0	Right	RCM	90.0	91.0	4.0

```
df_sal2 = pd.get_dummies(df_sal, prefix=['Club', 'Preferred Foot'])
df_sal2.head()
```

	Age	Overall	Potential	Value	Wage	Stamina	ShotPower	International Reputation	Club_ SSV Jahn Regensburg	Club_1. FC Heidenheim 1846	...	Club_Yeovil Town	Club_Yokohama F. Marinos	Club_Zagłębie Lubin	Club_Za Sosr
0	31	94	94	110.5	565.0	72.0	85.0	5.0	0	0	...	0	0	0	
1	33	94	94	77.0	405.0	88.0	95.0	5.0	0	0	...	0	0	0	
2	26	92	93	118.5	290.0	81.0	80.0	5.0	0	0	...	0	0	0	
3	27	91	93	72.0	260.0	43.0	31.0	4.0	0	0	...	0	0	0	
4	27	91	92	102.0	355.0	90.0	91.0	4.0	0	0	...	0	0	0	

5 rows × 661 columns

```
df_sal2.shape
```

```
(17918, 661)
```

PCA (Principal Component Analysis):

When using principal component analysis, one of the main goals is to reduce the dimensionality of the data — in effect, reducing the complexity of the problem. The main idea behind PCA is to figure out patterns and correlations among various features in the data set. On finding a strong correlation between different variables, a final decision is made about reducing the dimensions of the data in such a way that the significant data is still retained.

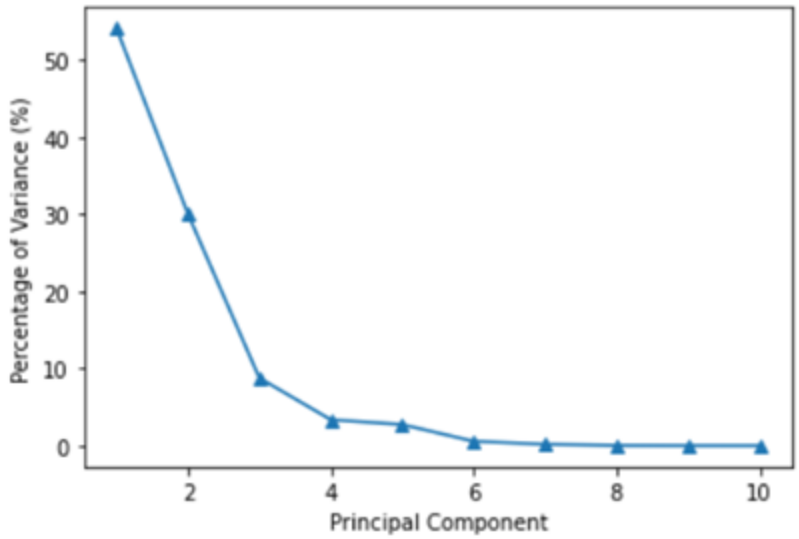
However, dimensionality reduction comes at a cost as fewer dimensions used implies less of the total variance in the data is being explained. Because of this, the cumulative explained variance ratio is extremely important for knowing how many dimensions are necessary for the problem. Additionally, if a significant amount of variance is explained by only two or three dimensions, the reduced data can be visualized afterwards. Since PCs describe variation and account for the varied influences of the original characteristics, we can plot the PCs to find out which feature produces the differences among clusters.

To do this we plot the loadings, or vectors representing each feature of the PC plot centered at (0, 0) with the direction and length of these vectors showing how much significance each feature has on the PCs. Also, the angle between these vectors let us know correlation between the features with a small angle denoting high correlation.

```
print(pca.explained_variance_ratio_)
```

```
[0.54 0.3  0.09 0.03 0.03 0.01 0.   0.   0.   0.  ]
```

```
fig = plt.figure()
ax = fig.add_subplot(111)
ax.plot(range(1, 11), pca.explained_variance_ratio_*100, marker='^')
plt.xlabel('Principal Component')
plt.ylabel('Percentage of Variance (%)')
plt.show()
```



Rocchio Method:

The Rocchio algorithm is based on a method of relevance feedback found in information retrieval systems which stemmed from the SMART Information Retrieval System. The algorithm is based on the assumption that most users have a general conception of which documents should be denoted as relevant or non-relevant.

```
pc = 0
total_var = 0
for i in pca.explained_variance_ratio_:
    pc += 1
    total_var += i*100
    print("Variance captured by PC " + str(pc) + ": " + str(total_var))
```

```
Variance captured by PC 1: 54.16353030161864
Variance captured by PC 2: 84.24326976488847
Variance captured by PC 3: 93.02475658023847
Variance captured by PC 4: 96.39686927921211
Variance captured by PC 5: 99.13649063172303
Variance captured by PC 6: 99.7009523257863
Variance captured by PC 7: 99.87933453704714
Variance captured by PC 8: 99.90926815641104
Variance captured by PC 9: 99.91562263893668
Variance captured by PC 10: 99.91576934536371
```

```
pca = decomposition.PCA(n_components=7)
imgTrans = pca.fit(img_norm).transform(img_norm)
np.set_printoptions(precision=2,suppress=True)
print(imgTrans)
```

```
[[ 1.11  0.9   0.49 ...  0.02  0.93  0.62]
 [-0.29  1.11  0.43 ... -0.01  0.82  0.21]
 [-0.3   0.91  0.28 ... -0.07  0.88  0.2 ]
 ...
 [-0.34 -0.37 -0.34 ...  0.06  0.15  0.01]
 [-0.34 -0.35 -0.29 ...  0.33  0.12 -0.01]
 [-0.35 -0.45 -0.31 ...  0.12  0.15 -0.01]]
```

Data Clustering - KMeans and DBScan
Data Preprocessing

I began the first step in the clustering portion of our analysis with importing the necessary libraries. I then read in the data, or the csv file into a pandas dataframe. I began preprocessing the data by removing the unwanted columns from the dataframe such as 'Name', 'Age', etc. The columns removed were chosen using domain knowledge of the data set and trying to reduce the data set to attributes we thought were relevant to determining a player's position. 51 columns were removed including the 'Overall' metric as this measurement was a possible combination of other features and a potential source of noise. The remaining 38 columns were mostly player attributes measuring attributes such as 'Crossing', 'Finishing' and 'Dribbling'. I continued the data preprocessing by dropping the missing rows from the dataset and saving to a new dataframe.

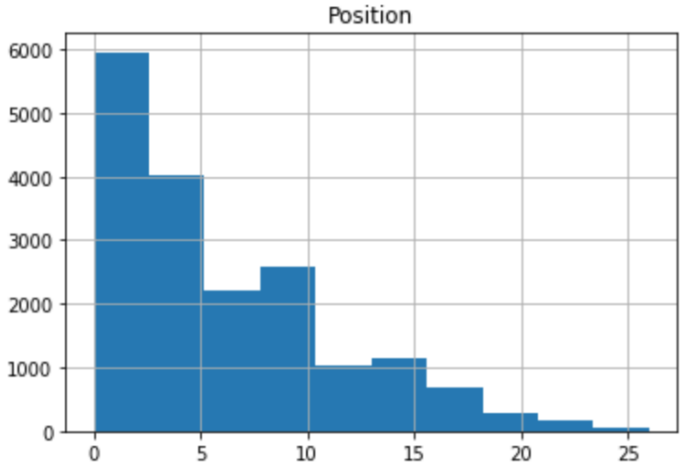
I then focused the data preprocessing on data types such as the Weight column, which was of type string. This column needed to have the 'lbs' characters stripped from the ends of the strings and be converted into float types. I then created an array for the 27 distinct player positions found in the data set, and assigned each a number, positions being the target variable and thus replacing the string character representation, such as ‘GK’ for goal keeper, to a number. I created a new data frame and replaced the position types with the corresponding array values as described above. I stored the target variable values, player position, corresponding to each entry in the data set in a numpy array. I stored the column names for all the attributes in a list and dropped the target variable, 'Position' from the list. I dropped the target variable from the data frame and stored two copies, one in a new dataframe and on as an array of values.

Kmeans clustering on raw data

Next I normalized the remaining attributes so that everything is in [0,1] scale. Normalization ensures that no attribute has disproportionate weight based on its range. The normalized values are a function of their distribution. For example, normalization prevents an attribute of height in inches having less weight than annual salary whereas a difference of 12 would be significant in the first but not the second. I then defined the Kmeans function as having 27 clusters, the predicted number of K or positions, fit the x_scaled data and clustered it. I then calculated and printed the centroids for each cluster. The centroids provide an aggregate representation and a characterization of each cluster. I then provided a function for calculating the cluster sizes. However, before providing the cluster sizes I wanted to provide the number of players in the data set that belong to each position such that it can be compared to the cluster sizes.

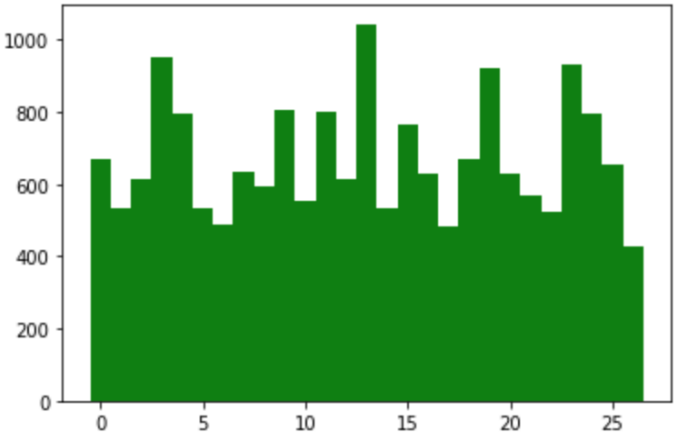
```
Out[18]: ST      2152
         GK      2025
         CB      1778
         CM      1394
         LB      1322
         RB      1291
         RM      1124
         LM      1095
         CAM      958
         CDM      948
         RCB      662
         LCB      648
         LCM      395
         RCM      391
         LW       381
         RW       370
         RDM      248
         LDM      243
         LS       207
         RS       203
         RWB       87
         LWB       78
         CF        74
         RAM       21
         LAM       21
         RF        16
         LF        15
         Name: Position, dtype: int64
```

Additionally, I provided a histogram distribution of the positions as they are represented in the data. The number of position players at each position is not equally distributed across the data as illustrated in the histogram provided below.



The clusters generated using Kmeans with a K of 27, as illustrated by the cluster sizes and the histogram provided below, are closer to an even distribution than that of the actual data thus it is reasonable to think that the clusters generated using Kmeans are not separated solely by position, or attributes that are likely determinative of position.

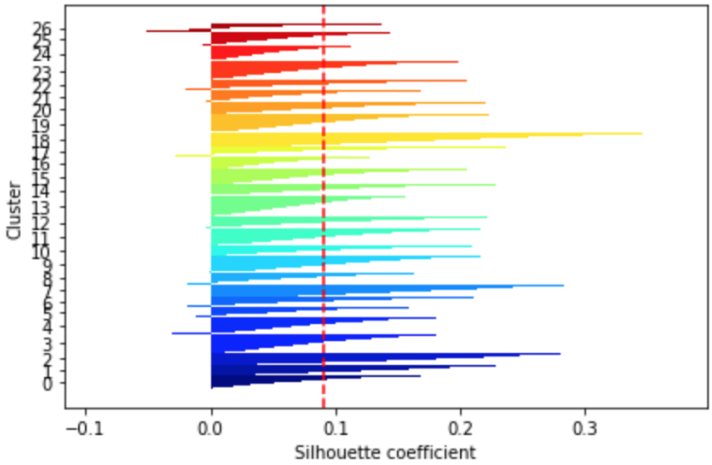
```
Out[20]: {0: 667,
          1: 532,
          2: 611,
          3: 950,
          4: 797,
          5: 535,
          6: 489,
          7: 634,
          8: 592,
          9: 803,
          10: 554,
          11: 799,
          12: 611,
          13: 1043,
          14: 534,
          15: 766,
          16: 630,
          17: 481,
          18: 671,
          19: 920,
          20: 629,
          21: 566,
          22: 525,
          23: 933,
          24: 797,
          25: 653,
          26: 425}
```



I then calculated the Silhouette coefficient of the generated clusters. The Silhouette Coefficient is useful in determining if clusters are clearly defined, indifferent, or have been assigned incorrectly. The measurements meanings are defined as follows: 1 - clusters clearly defined; 0 - clusters indifferent; and -1 clusters assigned wrong. The mean silhouette value of the KMeans K=27 clusters is close to 0 suggesting that the clusters are very similar to neighboring clusters. I then defined a function to plot the Silhouette coefficient for each cluster. The plot illustrated the clusters being curving upwards at just before Silhouette coefficient .1.

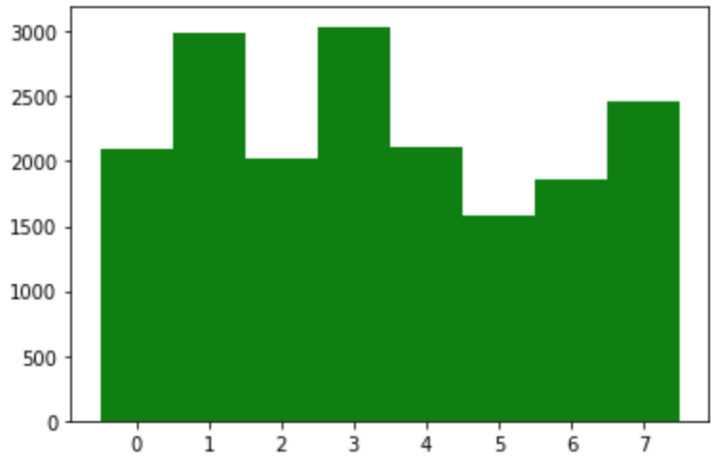
```
print(silhouettes.mean())
```

```
0.09024599173454685
```



I then retried Kmeans with a lower cluster number of 8, under the hypothesis that may provide better results as the various forward positions and defensive positions likely benefit from similar player attributes and calculated all the same metrics as done above with 27 clusters. The histogram of the clusters again illustrated that the number of players in each cluster was generally evenly distributed.

Out[30]: <BarContainer object of 8 artists>

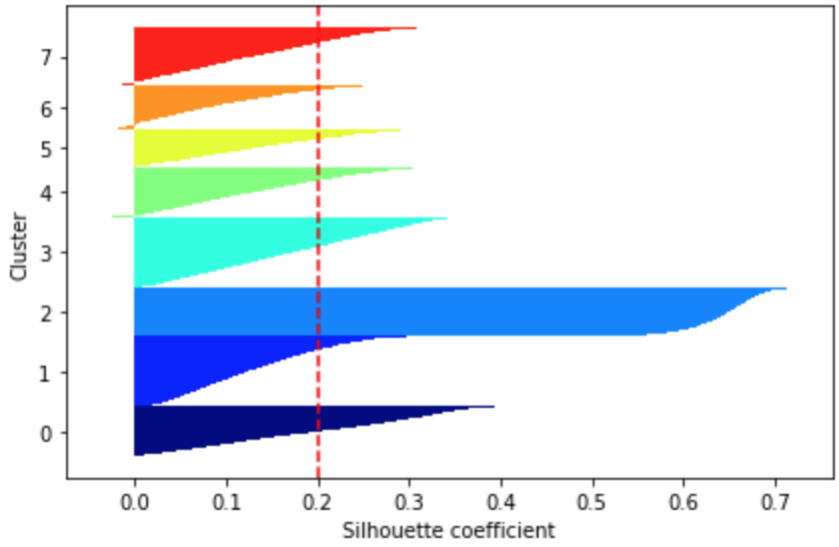


Lowering the number of clusters by 19 increased the silhouette coefficient, thus indicating that the eight clusters are more meaningful and representative than the 27 clusters.

```
print(silhouettes.mean())
```

```
0.20025641008563527
```

```
plot_silhouettes(x_scaled, clusters2)
```



I then calculated the Homogeneity and Completeness scores of the the cluster values when K=8. Homogeneity is a measurement of whether a cluster contains only data points which are members of a single class. Completeness is a measurement of whether all the data points that are members of a given class are elements of the same cluster. Both scores have positive values

between 0.0 and 1.0, larger values being more desirable. While the completeness and homogeneity scores are better when K=8 as opposed to when K=27, these scores are not close enough to 1 to indicate that a cluster represents all of the values of a class or that a cluster only contains values of one class.

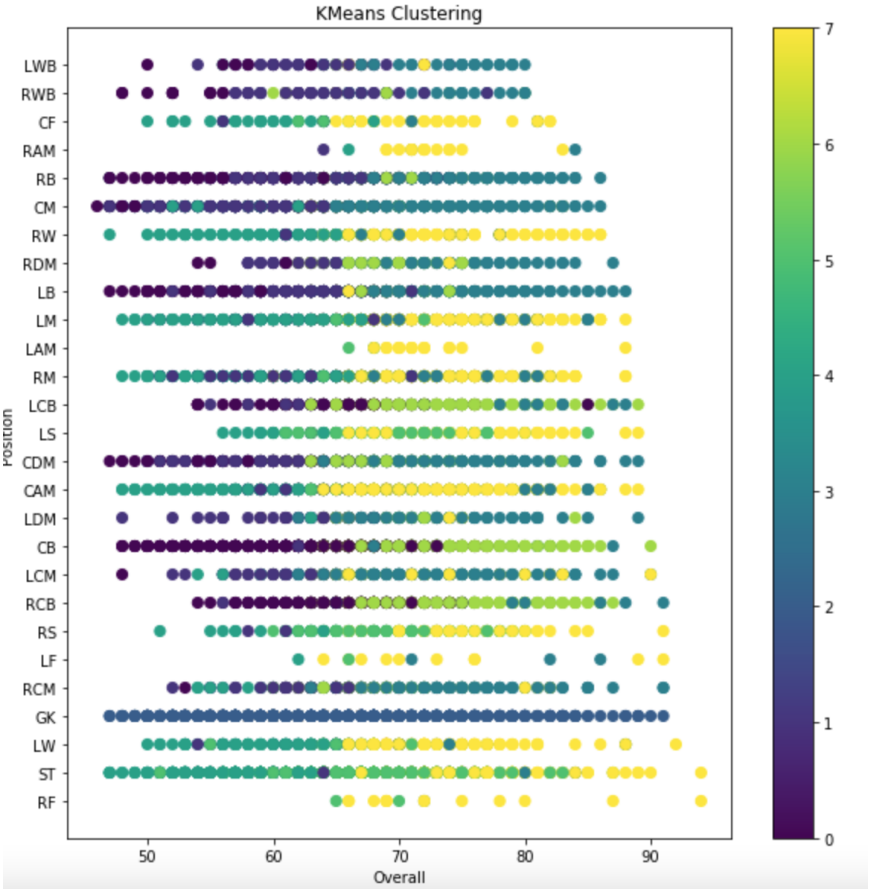
```
print(completeness_score(y,clusters2))
```

```
0.4985244472862511
```

```
print(homogeneity_score(y,clusters2))
```

```
0.3616282879392885
```

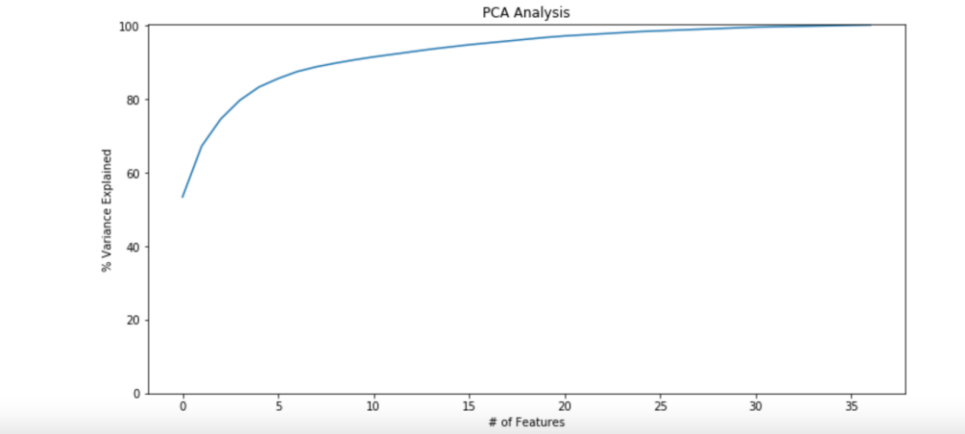
I then calculated a new data set but left in the 'Overall', dropped the NA values and added the clusters calculated during Kmeans with a value of K=8 for purposes of plotting the data by cluster. As illustrated in the plot provided below, For the most part,and with the exception of Goal Keepers, Kmeans did not cluster position players on opposite ends of the talent spectrum in the same cluster. Rather, it seems Kmeans may be clustering by talent level as opposed to position.



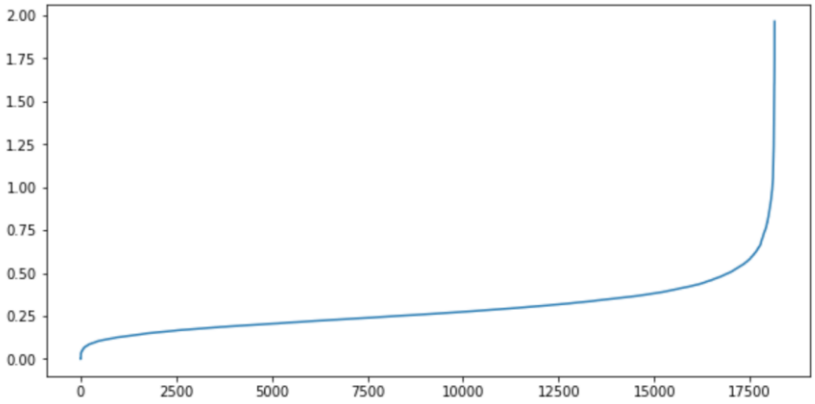
Density based clustering using DBScan¶

To begin I obtained a clean data value array scaled the values and put them in a data frame. I performed PCA of the scaled data frame and plotted the variance provided below which indicated that by selecting 4 features about 80% of the variance is explained.

```
Out[39]: [<matplotlib.lines.Line2D at 0x7fcf6605e790>]
```



DBScan generally does not perform well on high dimensional data so feature reduction was necessary. I then plotted the Nearest neighbors on the pca data frame, provided below which indicated that an eps of around .5 is preferable.



I then performed DBScan three times selecting various eps and min sample variables, resulting in the highest silhouette value of .484.

```
In [44]: db = DBSCAN(eps=1.2, min_samples=6).fit(pca_df)
         labels = db.labels_
         # Number of clusters in labels, ignoring noise if present.
         n_clusters_ = len(set(labels)) - (1 if -1 in labels else 0)
         n_noise_ = list(labels).count(-1)
         print('Estimated number of clusters: %d' % n_clusters_)
         print('Estimated number of noise points: %d' % n_noise_)
         print("Silhouette Coefficient: %0.3f" % metrics.silhouette_score(pca_df, labels))

         Estimated number of clusters: 2
         Estimated number of noise points: 42
         Silhouette Coefficient: 0.484
```

Kmeans clustering on PCA data

Lastly, I clustered the PCA data using Kmeans to see how it compared to Kmeans without PCA.

```
In [48]: print(silhouettes.mean())
0.33054466164410956

In [49]: print(completeness_score(y,clusters3))
0.4731062778011055

In [50]: print(homogeneity_score(y,clusters3))
0.3427193369853778
```

The results of Kmeans clustering on the PCA data were better than that of the Kmeans clustering on the raw data as illustrated by the higher silhouette, completeness and homogeneity values. Clustering algorithms generally have a hard time dealing with high dimensionality data, as this data set is, the Kmeans plot of clusters and positions seems to indicate that the data is clustered more by player potential and skill than by position.