ECS784P - Data Analytics - Coursework - 1

Clustering of Football Players Similar to 'Kylian Mbappé using K-means and GMM Algorithms

Sayed Sohail Pasha Peerzade(220541549)

1. INTRODUCTION

The aim of this project is to cluster the aggregated football performance data of Top 5 European Leagues (Premier League in England, the Bundesliga in Germany, La Liga in Spain, Serie A in Italy, and Ligue 1 in France) across across 5 seasons (2018-19 to 2022-2023), to find players similar to 'Kylian Mbappé'.

Clustering is a machine learning technique that groups similar data points based on their features. It can be used to discover patterns and insights from unlabeled data. In this project, we will explore how to cluster football players similar to Kylian Mbappé using two popular clustering algorithms: K-means and Gaussian Mixture Model (GMM).

To cluster football players similar to Kylian Mbappé, data was collected from FBref, including various features such as age, position, goals scored, assists made, and xG. After preprocessing and scaling the data, K-means and GMM algorithms were applied to cluster players based on their feature similarity. The clustering quality was evaluated using metrics such as Silhouette Score, Davies-Bouldin Index, and Calinski Harabasz Score.

By clustering football players similar to Kylian Mbappé using K-means or GMM algorithms, we can identify potential candidates who have similar playing styles or attributes as him. This can help us in scouting new talents or comparing different players across leagues or teams. Clustering can also reveal interesting patterns or insights about how different features affect player performance or similarity.

2. LITERATURE REVIEW

The paper by Pierpaolo D'Urso proposes a robust fuzzy clustering model for mixed data. The application to the clustering of football players on the basis of distance matrices of different attributes has shown the need to manage separately mixed type data. The obtained weights allow us to understand which is the most relevant set of attributes in partitioning the players. The paper suggests a multi-attributes approach for better clustering, In our project we are considering multi attributes to characterise a players performance.[1]

[2]The paper investigates the clustering capability of two unsupervised learning clustering methods: K-means and Expectation Maximization (EM). The methods are trained on soccer match data of the Spanish competition La Liga, which contains matches from 2004 to 2019. The paper classifies both cluster methods based on their ability to cluster soccer players. The clustering results are visualised using Principal Component Analysis (PCA) and show a correlation between player position. The authors found that K-means produced better results compared to EM which is similar to observations in this project.

The authors[3] discuss the relative importance of performance indicators to explain match outcomes in Australian Rules football. Prior studies have identified the relative importance of 'Inside-50's, kicks and goal conversion rates in explaining match success in the Australian Football League 1. Similarly in this project we have utilised performance features like Passes Into Final Third, Passes Into Penalty Area, Crosses Into Penalty Area to better analyse a forwards performance[5].

The paper[4] analyses football (soccer) player performance data with mixed type variables from the 2014-15 season of eight European major leagues. The authors cluster these data based on a tailor-made dissimilarity measure.

3. DATA PROCESSING

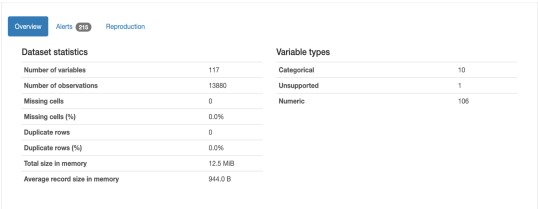
3.1 DATA SOURCE

The FBref dataset was used to analyse aggregated player performance across over 5,264 outfield players from the 'Big 5' European leagues for the past five seasons. FBref provided a vast range of comprehensive statistics and data on football, covering various leagues and competitions worldwide. The website's user-friendly features allowed for easy data filtering and sorting based on specific criteria, such as season, league, player, and team. Additionally, FBref.com's inclusion of historical data enabled performance comparisons across different seasons and competitions, making it an ideal data source for this project.

Player performance data is scraped from FBref website, saved to a csv file, and converted to a pandas DataFrame.

3.2 DATA DESCRIPTION

The raw dataset has one hundred and seventeen features(columns) and 13880 rows. There are mainly 3 different Data Types (object, float64, int64). No null and missing values are to be found in the Data Set.



For exploratory data analysis, a summary report was created using pandas Profiling Report, containing descriptive statistics, missing values, correlations, etc., in

an HTML file, making it easy to share and visualise the dataset

The quality of the data set was verified by checking the first and last roots in pandas using head( ) and tail( ) methods.

```
In [71]: df_fbref_outfield_raw.head()
```

Out[71]:

	player	nationality	position	team	comp_level	age	birth_year	games	games_starts	minutes	...	fouls	fouled	offsides	pens_won	pens_conceded
1533	Aaron Martin	es ESP	DF	Mainz 05	de Bundesliga	21	1997	33.0	32.0	2878.0	...	29.0	29.0	6.0	1.0	0
639	Abdou Diallo	sn SEN	DF	Dortmund	de Bundesliga	22	1996	28.0	27.0	2379.0	...	21.0	12.0	1.0	0.0	0
1043	Achraf Hakimi	ma MAR	DF	Dortmund	de Bundesliga	19	1998	21.0	19.0	1740.0	...	14.0	36.0	0.0	0.0	0
395	Adam Bodzek	de GER	MF,DF	Düsseldorf	de Bundesliga	32	1985	21.0	20.0	1696.0	...	32.0	41.0	0.0	0.0	2
2646	Adam Zrelak	sk SVK	FW	Nürnberg	de Bundesliga	24	1994	14.0	10.0	941.0	...	19.0	7.0	10.0	0.0	0

5 rows × 117 columns

```
In [72]: df_fbref_outfield_raw.tail()
```

Out[72]:

	player	nationality	position	team	comp_level	age	birth_year	games	games_starts	minutes	...	fouls	fouled	offsides	pens_won	pens_conceded
10310	Zlatan Ibrahimović	se SWE	FW	Milan	it Serie A	41-156	1981	2.0	0.0	42.0	...	2.0	0.0	0.0	0.0	0
11811	Angel Di Maria	ar ARG	FW,MF	Juventus	it Serie A	30-304	1988	16.0	11.0	876.0	...	4.0	11.0	4.0	0.0	0
11917	Éderson	br BRA	MF	Atalanta	it Serie A	22-240	1999	22.0	14.0	1266.0	...	18.0	22.0	0.0	0.0	0
12371	Hörður Jóhann Helgason	is ISL	MF	Lecce	it Serie A	22-160	2000	8.0	2.0	162.0	...	2.0	1.0	0.0	0.0	0
13477	Łukasz Skorupski	pl POL	GK	Bologna	it Serie A	31-309	1991	25.0	25.0	2250.0	...	0.0	6.0	0.0	0.0	0

5 rows × 117 columns

The shape property of a pandas DataFrame was used to identify the dimensionality of the data set.

3.3 Checking Shape returns a tuple representing the dimensionality of the DataFrame

```
# Print the shape of the raw DataFrame, df_fbref_outfield_raw
print(df_fbref_outfield_raw.shape)
```

```
(13880, 117)
```

The info method was used to get a quick description of the data, in particular the total number of rows, and each attribute's type and number of non-null values.

```
df_fbref_outfield_raw.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 83280 entries, 0 to 82877
Columns: 117 entries, player to aerials_won_pct
dtypes: float64(109), int64(1), object(7)
memory usage: 77.0+ MB
```

The describe method was used to show some useful statistics for each numerical column in the DataFrame.

```
In [77]: # Description of the raw DataFrame, df_fbref_outfield_raw, showing some summary statistics for each numberical column i
         df_fbref_outfield_raw.describe()
```

Out[77]:

	birth_year	games	games_starts	minutes	goals	assists	pens_made	pens_att	cards_yellow	cards_red	...
count	13880.000000	13880.000000	13880.000000	13880.000000	13880.000000	13880.000000	13880.000000	13880.000000	13880.000000	13880.000000	...
mean	1994.386671	17.731772	13.331556	1196.824784	1.632637	1.134942	0.161167	0.203098	2.495605	0.119020	...
std	4.742461	11.099607	10.933905	944.788889	3.140143	1.913247	0.739042	0.866409	2.648048	0.355028	...
min	1977.000000	1.000000	0.000000	1.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	...
25%	1991.000000	7.000000	3.000000	318.750000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	...
50%	1995.000000	18.000000	12.000000	1064.000000	0.000000	0.000000	0.000000	0.000000	2.000000	0.000000	...
75%	1998.000000	27.000000	22.000000	1919.000000	2.000000	2.000000	0.000000	0.000000	4.000000	0.000000	...
max	2007.000000	38.000000	38.000000	3420.000000	41.000000	21.000000	14.000000	15.000000	17.000000	3.000000	...

8 rows × 110 columns

3.3 DATA PRE-PROCESSING

The raw dataset was then cleaned and transformed into a format that is suitable for analysis. The following steps were performed as part of data pre-processing:

- Creating a data frame to store names of all the teams in Top 5 European Leagues

```
df_teams.head()
```

	0
0	Mainz 05
1	Dortmund
2	Düsseldorf
3	Nürnberg
4	Wolfsburg

- Removing Special Characters and lower casing player names.
  Special European characters were removed from player names, and names were converted to lowercase in the dataset.
- String Cleaning: Mapping Countries obtained from CSV to Player Nationality in the dataset.
  A new column 'Nationality Cleaned' was created in the dataset by matching FIFA codes with country names using a separate CSV file.

```
df_fbref_outfield['Nationality Cleaned'] = df_fbref_outfield['Nationality Code'].map(dict_countries)
df_fbref_outfield.head()
```

	birth_year	games	games_starts	minutes	...	ball_recoveries	aerials_won	aerials_lost	aerials_won_pct	Player Lower	First Name Lower	Last Name Lower	First Initial Lower	Nationality Code	Nationality Cleaned
	1997	33.0	32.0	2878.0	...	166.0	23.0	36.0	39.0	aaron martin	aaron	martin	a	ESP	Spain
	1996	28.0	27.0	2379.0	...	153.0	49.0	40.0	55.1	abdou diallo	abdou	diallo	a	SEN	Senegal
	1998	21.0	19.0	1740.0	...	137.0	15.0	18.0	45.5	achraf hakimi	achraf	hakimi	a	MAR	Morocco
	1985	21.0	20.0	1696.0	...	63.0	34.0	21.0	61.8	adam bodzek	adam	bodzek	a	GER	Germany
	1994	14.0	10.0	941.0	...	15.0	53.0	88.0	37.6	adam zrelak	adam	zrelak	a	SVK	Slovakia

- String Cleaning: Cleaning League names in the Dataset.
  The 'comp_level' column in the raw dataset contains both league and country code, so the country code was removed and a new column 'Comp' was created to store cleaned league names.

4.4 String Cleaning: Cleaning League names in the dataset

```
In [97]: ## Define league names and cleaned league names
         dict_league_names = {'eng Premier League': 'Premier League',
                              'fr Ligue 1': 'Ligue 1',
                              'de Bundesliga': 'Bundaliga',
                              'it Serie A': 'Serie A',
                              'es La Liga': 'La Liga'
                             }
         df_fbref_outfield['Comp'] = df_fbref_outfield['comp_level'].map(dict_league_names)
```

```
In [98]: df_fbref_outfield.tail()
```

Out[98]:

	birth_year	games	games_starts	minutes	...	aerials_won	aerials_lost	aerials_won_pct	Player Lower	First Name Lower	Last Name Lower	First Initial Lower	Nationality Code	Nationality Cleaned	Comp
	1981	2.0	0.0	42.0	...	3.0	2.0	60.0	zlatan ibrahimovic	zlatan	ibrahimovic	z	SWE	Sweden	Serie A
	1988	16.0	11.0	876.0	...	1.0	1.0	50.0	angel di maria	angel	maria	a	ARG	Argentina	Serie A
	1999	22.0	14.0	1266.0	...	10.0	15.0	40.0	ederson	ederson	ederson	e	BRA	Brazil	Serie A
	2000	8.0	2.0	162.0	...	1.0	4.0	20.0	orir johann helgason	orir	helgason	o	ISL	Iceland	Serie A
	1991	25.0	25.0	2250.0	...	4.0	0.0	100.0	ukasz skorupski	ukasz	skorupski	u	POL	Poland	Serie A

- Grouping Player Positions and Defining a grouped position
  As every footballer plays in multiple positions during the course of the season, the dataset included multiple positions for each player. It becomes difficult to analyse player performances as some stats depend on the position which a player plays. Therefore the player positions were grouped into Goalkeeper, Defender , Midfielder and Forward. Two new columns 'Primary Pos' and 'Position Grouped' were created.

```
## Define positions and their grouped position names
dict_positions_grouped = {'DF': 'Defender',
                          'DF,FW': 'Defender',
                          'DF,GK': 'Defender',
                          'DF,MF': 'Defender',
                          'FW': 'Forward',
                          'FW,DF': 'Forward',
                          'FW,MF': 'Forward',
                          'GK': 'Goalkeeper',
                          'GK,FW': 'Goalkeeper',
                          'MF': 'Midfielder',
                          'MF,DF': 'Midfielder',
                          'MF,FW': 'Midfielder',
                          'MF,GK': 'Midfielder',
                         }
df_fbref_outfield['Primary Pos'] = df_fbref_outfield['position'].str[:2]
# Map grouped positions to DataFrame
df_fbref_outfield['Position Grouped'] = df_fbref_outfield['position'].map(dict_positions_grouped)
df_fbref_outfield.head()
```

_level	age	birth_year	games	games_starts	minutes	...	aerials_won_pct	Player Lower	First Name Lower	Last Name Lower	First Initial Lower	Nationality Code	Nationality Cleaned	Comp	Primary Pos	Position Grouped
de esliga	21	1997	33.0	32.0	2878.0	...	39.0	aaron martin	aaron	martin	a	ESP	Spain	Bundeliga	DF	Defender
de esliga	22	1996	28.0	27.0	2379.0	...	55.1	abdou diallo	abdou	diallo	a	SEN	Senegal	Bundeliga	DF	Defender
de esliga	19	1998	21.0	19.0	1740.0	...	45.5	achraf hakimi	achraf	hakimi	a	MAR	Morocco	Bundeliga	DF	Defender
de esliga	32	1985	21.0	20.0	1696.0	...	61.8	adam bodzek	adam	bodzek	a	GER	Germany	Bundeliga	MF	Midfielder
de esliga	24	1994	14.0	10.0	941.0	...	37.6	adam zrelak	adam	zrelak	a	SVK	Slovakia	Bundeliga	FW	Forward

- Converting Data Types
  Data Types for age and birth_year were converted from string object to integer to allow us to filter the data set easily.
- Creating Per 90 Stats
  Standardising stats to per 90 to allow players that have had less minutes on the field or had less time on the ball to be compared to first team regular players.The following stats from the original Dataset have been converted to per 90 stats:
  - **Goals Per Shot**
  - **Goals Per Shot on Target**
  - **Non Goals XG per Shot**
  - **XG NET**

- **Passes Into Final Third**
- **Passes Into Penalty Area**
- **Crosses Into Penalty Area**
- **Progressive Passes**
- **Through Balls**
- **Carries Into Penalty Area**

```
In [132]: ## Creating per 90 stats for later use
df_players = df_fbref_outfield
df_players['goals_per_shot_per90'] = (df_players['goals_per_shot'] / df_players['minutes']) * 90
df_players['goals_per_shot_on_target_per90'] = (df_players['goals_per_shot_on_target'] / df_players['minutes']) * 90
df_players['npxg_per_shot_per90'] = (df_players['npxg_per_shot'] / df_players['minutes']) * 90
df_players['xg_net_per90'] = (df_players['xg_net'] / df_players['minutes']) * 90
df_players['passes_into_final_third_per90'] = (df_players['passes_into_final_third'] / df_players['minutes']) * 90
df_players['passes_into_penalty_area_per90'] = (df_players['passes_into_penalty_area'] / df_players['minutes']) * 90
df_players['crosses_into_penalty_area_per90'] = (df_players['crosses_into_penalty_area'] / df_players['minutes']) * 90
df_players['progressive_passes_per90'] = (df_players['progressive_passes'] / df_players['minutes']) * 90
df_players['through_balls_per90'] = (df_players['through_balls'] / df_players['minutes']) * 90
df_players['carries_into_penalty_area_per90'] = (df_players['carries_into_penalty_area'] / df_players['minutes']) * 90
df_players.head()
```

Out[132]:

...	goals_per_shot_per90	goals_per_shot_on_target_per90	npxg_per_shot_per90	xg_net_per90	passes_into_final_third_per90	passes_into_penalty_area_per90	cross
...	0.000000	0.000000	0.001251	-0.025017	2.314107	1.188325	
...	0.003783	0.009458	0.009836	-0.022699	4.653216	0.416141	
...	0.005172	0.020690	0.004138	0.015517	4.913793	1.396552	
...	0.000000	0.000000	0.002256	-0.005639	2.424812	0.225564	
...	0.011477	0.031562	0.021041	-0.162593	0.669501	0.095643	

- Subset Data of only forwards with at least 10 league matches.
  A subset of the dataset was created for forwards as Kylian Mbappe's primary position is a Forward, henceforth only for next steps, dataset is filtered to players with primary position as forward. The dataset is further filtered to include players who have played at least 10 league matches in a season (900 minutes), as players with lower game time can skew the clustering.

Filter for Forwards that have played at least 10 Matches or 900 minutes during a season

```
## Subset Data
# Filter for Forwards that have played at least 10 matches (900 minutes)
df_players_fw = df_players[
                           (df_players['Primary Pos'] ==  'FW') &
                           (df_players['Season'].isin(['2018-2019', '2019-2020', '2020-2021', '2021-202
                           (df_players['minutes'] >= 90 * 10)
                          ]
```

```
# Check the variance in the dataset before and after filtration
print('No. rows in Players DataFrame BEFORE filtration: {}'.format(len(df_players)))
print('No. rows AFTER filtration: {}'.format(len(df_players_fw)))
print('-'*10+'\n')
```

```
No. rows in Players DataFrame BEFORE filtration: 13880
No. rows AFTER filtration: 1606
----------
```

3.4 FEATURE SELECTION

The dataset contains 117 features which depict various aspects of player performance, However some stats like 'Clean sheets made' are irrelevant to our position of interest (Forward). Hence we need to select a subset of most important features from the dataset so that it improves the accuracy and efficiency of the analysis and reduces the risk of overfitting.

The columns of interest selected for analysis are:

```
In [516]: ## Selecting colums of interest
          ## String Columns:
          cols_str = ['Player Lower',
                      'Season',
                      'Comp',
                      'Primary Pos',
                      'position',
                      'age',
                      'birth_year',
                      'Nationality Cleaned',
                      'team'
          ]
          cols_stats = ['games',
                        'games_starts',
                        'minutes',
                        'goals',
                        'assists',
                        'goals_per90',
                        'assists_per90',
                        'goals_assists_per90',
                        'goals_pens_per90',
                        'goals_assists_pens_per90',
                        'xg_per90',
                        'npxg_per90',
                        'minutes_90s',
                        'shots_on_target_pct',
                        'shots_on_target_per90',
                        'goals_per_shot_per90',
                        'goals_per_shot_on_target_per90',
                        'npxg_per_shot_per90',
                        'xg_net_per90',
                        'passes_into_final_third_per90',
                        'passes_into_penalty_area_per90',
                        'crosses_into_penalty_area_per90',
                        'progressive_passes_per90',
                        'through_balls_per90',
                        'carries_into_penalty_area_per90',
                        'progressive_carries'
          ]
```

3.5 DATA NORMALISATION

All the features in the data set are not in a common scale. Data has to be normalised to bring all the features to the same level of importance so that no feature dominates over the others.

Min-max normalisation was used to normalise the features in the dataset to a range between 0 and 1 by subtracting the minimum value of the feature and dividing the result by the difference between the maximum and minimum values.

```
## Standardise Data
x = df_players_fw_stats.values    # NumPy array
scaler = preprocessing.MinMaxScaler()
x_scaled = scaler.fit_transform(x)
X_norm = pd.DataFrame(x_scaled)
```

3.6 Dimensionality Reduction using PCA

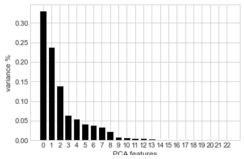
After selecting features that are relevant for our analysis we still have 23 features. It becomes very difficult to visualise these 23 features, hence dimensionality reduction has been implemented to retain most important information and to simplify the data for visualisations.

PCA is a technique that reduces the dimensions of a dataset while retaining important information. It transforms correlated variables into uncorrelated principal components sorted by variance explained. The first component explains the most variance, followed by the second, and so on.

By selecting a subset of these principal components, we can reduce the dimensionality of the data while retaining most of the information.

To Determine the optimal number of components that capture the greatest amount of variance in the data the following scree plot was plotted.

```
In [586]: # Plot the explained variances
          features = range(pca.n_components_)
          plt.bar(features, pca.explained_variance_ratio_, color='black')
          plt.xlabel('PCA features')
          plt.ylabel('variance %')
          plt.xticks(features)
          plt.show()
```



The figure shows that the first two components explain the majority of the variance in the data. For the use case of visualisation this works well as 2-dimensions can be easily plotted visually.

4. Learning Methods

To identify players similar to Kylian Mbappe, we have used two clustering algorithms: K-means and Gaussian Mixture Model(GMM). Clustering is a type of unsupervised machine learning technique used to group similar data points together based on some similarity measure or distance metric.

4.1 K-Means

K-Means clustering is a clustering algorithm that involves the following steps:

- First, a user specifies the number of clusters (K) and randomly selects initial centroids.
- Then, the algorithm iteratively assigns each observation to one of the K clusters until the assignments stop changing.
- Next, the mean is calculated for each of the K clusters. Finally, each observation is assigned to the cluster whose mean is closest to it.
- The objective is to form clusters in a way that maximises the similarity of observations within the same cluster.

K-Means clustering measures similarity using the

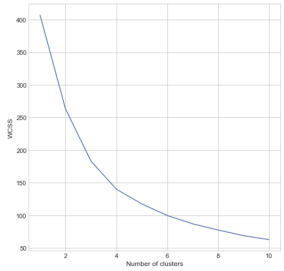
Squared Euclidean distance.

Determining Number of Clusters for K-Means

The number of clusters for K-Means algorithm for our problem was determined by two methods:

By Determining the Elbow

To determine the optimal number of clusters, the Within Cluster Sum of Squares (WCSS) for different cluster solutions was computed and plotted on an Elbow plot. The elbow point, where the WCSS reduction rate begins to level off, was chosen to identify the number of clusters that would not enhance clustering performance..

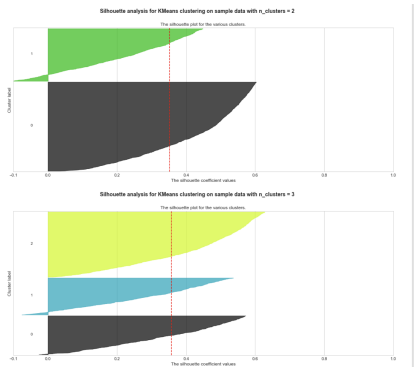


However, no distinct elbow can be identified from the plot for our dataset. The plot shows that there is a slight elbow near 4, hence 4 clusters have been considered for implementing K-Means.

Silhouette Score to determine optimal number of clusters

The silhouette score measures how well each data point fits its cluster. A score close to 1 indicates a good match, while -1 means a data point may be in the wrong cluster. Scores for 2-10 clusters were plotted.Silhouette scores for 2 to 10 clusters were calculated and the coefficient values were plotted against the cluster labels.

```
For n_clusters = 2 The average silhouette_score is : 0.3517643638160773
For n_clusters = 3 The average silhouette_score is : 0.3579423672179299
For n_clusters = 4 The average silhouette_score is : 0.3465608526411299
For n_clusters = 5 The average silhouette_score is : 0.3210605029601038
For n_clusters = 6 The average silhouette_score is : 0.32695659588683545
For n_clusters = 7 The average silhouette_score is : 0.32950588845850814
For n_clusters = 8 The average silhouette_score is : 0.3206865282670317
For n_clusters = 9 The average silhouette_score is : 0.3282346817062359
For n_clusters = 10 The average silhouette_score is : 0.32452984247592154
```

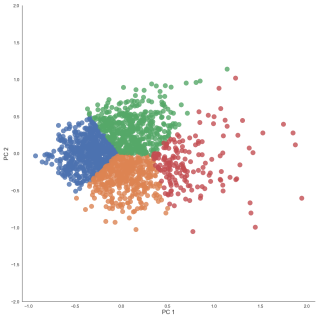


Average of all the values was taken and rounded off to 4. Hence 4 clusters were considered for K-means Implementation.

5.1.3 Clustering the Data using K-Means

```
In [592]: # K-Means Clustering

          ## Specify the number of clusters
          kmeans = KMeans(n_clusters=4, init='k-means++', max_iter=600, random_state=42)

          ## Fit the input data
          kmeans = kmeans.fit(df_reduced)

          ## Get the cluster labels
          labels = kmeans.predict(df_reduced)

          ## Centroid values
          centroid = kmeans.cluster_centers_

          ## Cluster values
          clusters = kmeans.labels_.tolist()
```

K-Means clustering is applied on the entire dataset to obtain cluster labels, centroids and values. The 2-dimensional data points obtained after PCA are used to plot the features and the labels are marked.



We can see that K-means was able to separate the data points into 4 different clusters.

4.2 GMM

The Gaussian Mixture Model (GMM) algorithm estimates parameters of multiple Gaussian distributions to assign each data point to a particular cluster. It involves initialization, expectation step and maximisation step, which is repeated until convergence. GMM can be used for clustering and density estimation, where data points are assigned to a cluster based on their highest probability of belonging to that cluster.

Determining Optimal Number of Clusters using BIC and AIC

The Bayesian Information Criterion (BIC) and Akaike Information Criterion (AIC) are two statistical metrics used in Gaussian Mixture Model (GMM) clustering to determine the optimal number of clusters.

BIC and AIC are both measures of the quality of the model fit and take into account the number of parameters in the model. They penalise models with a large number of parameters, thus avoiding overfitting.

The AIC is defined as AIC = 2k - 2ln(L), where k is the number of parameters in the model and L is the likelihood function of the data. The BIC is defined as BIC = kln(n) - 2ln(L), where n is the number of data points.

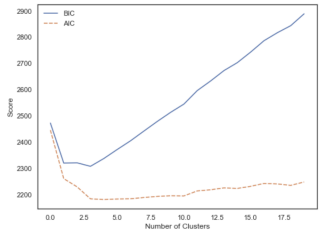
In both cases, the lower the value of the metric, the better the model fits. When comparing GMM models with different numbers of clusters, the model with the lowest AIC or BIC value is typically selected as the optimal model.

```
In [606]: n_components = np.arange(1, 21)
          models = [GMM(n,covariance_type='full', random_state=0).fit(gmm_df_reduced) for n in n_components]
          gmm_model_comparisons=pd.DataFrame({"n_components" : n_components,
                                              "BIC" : [m.bic(gmm_df_reduced) for m in models],
                                              "AIC" : [m.aic(gmm_df_reduced) for m in models]})
          gmm_model_comparisons.head()
```

Out[606]:

	n_components	BIC	AIC
0	1	2473.474276	2446.566767
1	2	2320.961238	2261.764717
2	3	2321.916713	2230.431181
3	4	2308.612846	2184.838302
4	5	2338.261262	2182.197707

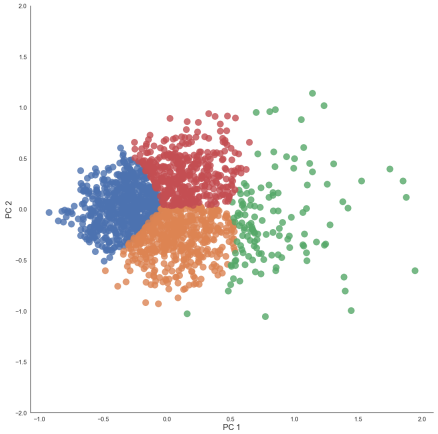
```
In [607]: plt.figure(figsize=(8,6))
          sns.lineplot(data=gmm_model_comparisons[["BIC","AIC"]])
          plt.xlabel("Number of Clusters")
          plt.ylabel("Score")
          plt.savefig("GMM_model_comparison_with_AIC_BIC_Scores_Python.png",
                      format='png',dpi=150)
```



As we can see from the above table and graph the BIC and AIC values are lower when Number of Clusters is 4. Hence Selecting 4 Clusters for GMM Clustering.
GMM clustering is then implemented on the entire data set containing all the features and the cluster and label values are computed.

```
In [608]: gmm = GMM(n_components=4)
          gmm = gmm.fit(gmm_df_reduced)
          gmm_labels = gmm.predict(gmm_df_reduced)
          gmm_labels_list = gmm_labels.tolist()
```

The 2-dimensional data points obtained after PCA are used to plot the features and cluster labels are then marked on the plot.



We can see that GMM was able to separate the data points into 4 different clusters.

4.3 COMPARING K-MEANS AND GMM

Silhouette Score, Davies Bouldin Score, and Calinski Harabasz Score are all metrics used to evaluate the quality of clustering results.

Silhouette Score: measures how well each data point fits into its assigned cluster, and how far it is from the neighbouring clusters. The score ranges from -1 to 1, where a score closer to 1 indicates a better clustering result. Higher scores indicate that the data points are well-clustered and separated from each other.

Davies Bouldin Score: measures the average similarity between each cluster and its most similar cluster, and compares this to the average distance between each cluster and its most dissimilar cluster. Lower scores indicate better clustering results, with values closer to 0 indicating better separation between the clusters.

Calinski Harabasz Score: measures the ratio of the between-cluster dispersion to the within-cluster dispersion. Higher scores indicate better clustering results, with larger differences in dispersion between the clusters and smaller dispersion within each cluster.

Silhouette Score, Davies Bouldin Score and Calinski Harabasz score for K-means and GMM have been implemented.

```
In [593]: ## Silhouette score of Kmeans
          km_score = silhouette_score(df_reduced, clusters)
          print('Silhouetter Score: %.3f' % km_score)

          Silhouetter Score: 0.347

In [594]: ## Davies Bouldin Score of Kmeans
          km_davies_bouldin_score = davies_bouldin_score(df_reduced,clusters)
          km_davies_bouldin_score

Out[594]: 0.8777551503192457

In [595]: ## Calinski Score for Kmeans
          km_calinski_score = calinski_harabasz_score(df_reduced, clusters)
          km_calinski_score

Out[595]: 1016.7282138852587

In [616]: ## Silhouette Score for GMM
          gmm_score = silhouette_score(gmm_norm, gmm_labels)
          print('Silhouette Score: %.3f' % gmm_score)

          Silhouette Score: 0.160

In [610]: ## Davies Bouldin Score for GMM
          gmm_davies_bouldin_score = davies_bouldin_score(gmm_norm,gmm_labels)
          gmm_davies_bouldin_score

Out[610]: 1.6807706185997255

In [611]: ## Calinski Score for GMM
          gmm_calinski_score = calinski_harabasz_score(gmm_norm, gmm_labels)
          gmm_calinski_score

Out[611]: 314.36191877045707
```

	Silhouette Score	Davies Bouldin Score	Calinski Harabasz Score
K-Means	0.3465608526411299	0.8777551503192457	1016.7282138852587
GMM	0.15989146361332376	1.6807706185997255	314.36191877045707

By observing the scores for both K-means and GMM we can see that K-means has a better Silhouette, Davies Bouldin and Calinski Harabasz Scores. This indicates that K-means is a better algorithm for the given data set.

This could be because the data points are well separated and belong to distinct clusters: K-means works well when the clusters are well-separated and have a spherical shape. If the data points belong to distinct clusters with little overlap, K-means can quickly and accurately group them together. Also, K-means works well with uniformly distributed data, where the cluster centres are equidistant from each other. If the data is uniformly distributed, the K-means algorithm can quickly converge to an optimal solution.

5. ANALYSIS

To find players similar to Kylian Mbappe, the following investigations have been made on the data set using both the K-Means and GMM clusters which have been obtained in the previous step.

5.1 Analysing Kylian Mbappe's performance over last 5 seasons

```
In [629]: df_reduced['cluster'].value_counts()

Out[629]: 0    556
          2    457
          1    419
          3    174
          Name: cluster, dtype: int64
```

The results from K-means produces 4 distinct clusters, the most populous being cluster 0 with 556 players, followed by cluster 2 (457), 1 (419) and 3 (174).

```
In [633]: gmm_df_reduced['cluster'].value_counts()

Out[633]: 0    585
          1    500
          3    393
          2    128
          Name: cluster, dtype: int64
```

The results from GMM produced 4 distinct clusters, the most populous being cluster 0 with 585 players, followed by cluster 1 (500), 3 (393) and 2 (128).
Identifying which cluster Kylian Mbappe belongs to:

```
Out[568]:
```

	x	y	cluster	season
992	1.874440	0.119329	3	2018-2019
1067	1.526649	0.276810	3	2019-2020
1131	1.277519	-0.151959	3	2020-2021
1202	1.387960	-0.666055	3	2021-2022
1261	1.093180	0.008615	3	2022-2023

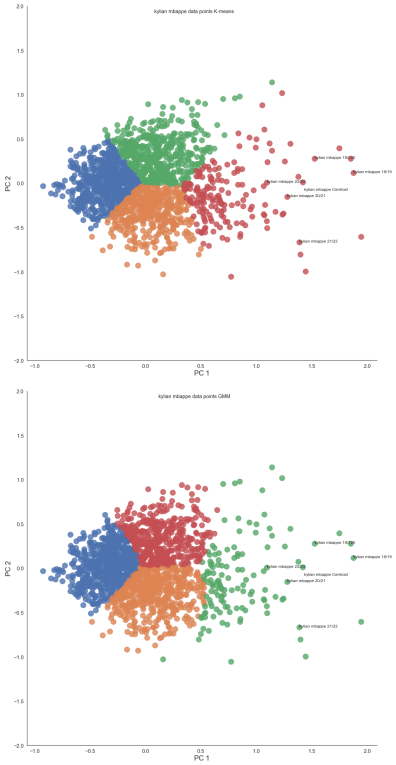
From the results we can see that Mbappe is found in the

cluster 3 for the past 5 seasons for K-Means Algorithm.

Out[634]:

	x	y	cluster	season
992	1.874440	0.119329	2	2018-2019
1067	1.526649	0.276810	2	2019-2020
1131	1.277519	-0.151959	2	2020-2021
1202	1.387960	-0.666055	2	2021-2022
1261	1.093180	0.008615	2	2022-2023

From the results we can see that Mbappe is found in the cluster 2 for the past 5 seasons for GMM Algorithm.
Visualising the data points for Kylian Mbappe:



By looking at the plots we can infer that inter cluster distance between Mbappe and other data points is significant. Indicating that Mbappe has been out performing all the other football players in that cluster.

5.2 Closest Performing Players in last 5 seasons

Finding top 5 players with similar performances as Mbappe over the last 5 seasons. The distance between each player and Mbappe points are calculated to find the closest performing players.

```
In [641]: # Determine the distance of each data point from each of the five mbappe data points
df_reduced['dist_mbappe_1819'] = np.sqrt( (mbappe_x_1819 - df_reduced['x'])**2 + (mbappe_y_1819 - df_reduced['y'])**2)
df_reduced['dist_mbappe_1920'] = np.sqrt( (mbappe_x_1920 - df_reduced['x'])**2 + (mbappe_y_1920 - df_reduced['y'])**2)
df_reduced['dist_mbappe_2021'] = np.sqrt( (mbappe_x_2021 - df_reduced['x'])**2 + (mbappe_y_2021 - df_reduced['y'])**2)
df_reduced['dist_mbappe_2122'] = np.sqrt( (mbappe_x_2122 - df_reduced['x'])**2 + (mbappe_y_2122 - df_reduced['y'])**2)
df_reduced['dist_mbappe_2223'] = np.sqrt( (mbappe_x_2223 - df_reduced['x'])**2 + (mbappe_y_2223 - df_reduced['y'])**2)

# Determine the distance of each data point from the 17/18-19/20 mbappe data points centroid
df_reduced['dist_mbappe_centroid'] = np.sqrt( (mbappe_x_centroid - df_reduced['x'])**2 + (mbappe_y_centroid - df_reduce
```

```
In [643]: termine the distance of each data point from each of the five mbappe data points
df_reduced['dist_mbappe_1819'] = np.sqrt( (gmm_mbappe_x_1819 - gmm_df_reduced['x'])**2 + (gmm_mbappe_y_1819 - gmm_df_re
df_reduced['dist_mbappe_1920'] = np.sqrt( (gmm_mbappe_x_1920 - gmm_df_reduced['x'])**2 + (gmm_mbappe_y_1920 - gmm_df_re
df_reduced['dist_mbappe_2021'] = np.sqrt( (gmm_mbappe_x_2021 - gmm_df_reduced['x'])**2 + (gmm_mbappe_y_2021 - gmm_df_re
df_reduced['dist_mbappe_2122'] = np.sqrt( (gmm_mbappe_x_2122 - gmm_df_reduced['x'])**2 + (gmm_mbappe_y_2122 - gmm_df_re
df_reduced['dist_mbappe_2223'] = np.sqrt( (gmm_mbappe_x_2223 - gmm_df_reduced['x'])**2 + (gmm_mbappe_y_2223 - gmm_df_re

termine the distance of each data point from the 17/18-19/20 mbappe data points centroid
df_reduced['dist_mbappe_centroid'] = np.sqrt( (gmm_mbappe_x_centroid - gmm_df_reduced['x'])**2 + (gmm_mbappe_y_centroid
```

Mbappe Cluster is identified and sorted by distance between Mbappe Points.

For K Means:

```
# Filter the DataFrame by players in the same cluster as Mbappe - K-means
df_mbappe_cluster = df_reduced[(df_reduced['cluster'] == mbappe_cluster)]

# Display the DataFrame
# df_mbappe_cluster.sort_values(by=['dist_mbappe_centroid'], ascending=True).head(10)
df_mbappe_cluster[~df_mbappe_cluster['name'].str.contains('kylian mbappe', case=False)].sort_values(by=['dist_mbappe_ce
## Closest performing players in all seasons
```

	x	y	cluster	name	season	minutes	matches_played	matches_started	comp	team	...	pos	age	birth_year	nationality
108	1.419252	0.012619	3	robert lewandowski	2019-2020	2759.0	31.0	31.0	Bundesliga	Bayern Munich	...	FW	30	1988	Polanc
225	1.377175	0.073961	3	robert lewandowski	2021-2022	2946.0	34.0	34.0	Bundesliga	Bayern Munich	...	FW	32	1988	Polanc
117	1.244581	-0.342463	3	timo werner	2019-2020	2795.0	34.0	33.0	Bundesliga	RB Leipzig	...	FW	23	1996	Germany
865	1.234378	-0.351267	3	karim benzema	2021-2022	2593.0	32.0	31.0	La Liga	Real Madrid	...	FW	33	1987	France
1365	1.171932	-0.322932	3	ciro immobile	2019-2020	3170.0	37.0	36.0	Serie A	Lazio	...	FW	29	1990	Italy

For GMM:

```
In [705]: # Filter the DataFrame by players in the same cluster as Mbappe - GMM
gmm_df_mbappe_cluster = gmm_df_reduced[(gmm_df_reduced['cluster'] == gmm_mbappe_cluster)]

# Display the DataFrame
# df_mbappe_cluster.sort_values(by=['dist_mbappe_centroid'], ascending=True).head(10)
gmm_df_mbappe_cluster[~gmm_df_mbappe_cluster['name'].str.contains('kylian mbappe', case=False)].sort_values(by=['dist_m
## Closest performing players in all seasons
```

Out[705]:

	x	y	cluster	name	season	minutes	matches_played	matches_started	comp	team	...	pos	age	birth_year	national
108	1.419252	0.012619	2	robert lewandowski	2019-2020	2759.0	31.0	31.0	Bundesliga	Bayern Munich	...	FW	30	1988	Pola
225	1.377175	0.073961	2	robert lewandowski	2021-2022	2946.0	34.0	34.0	Bundesliga	Bayern Munich	...	FW	32	1988	Pola
117	1.244581	-0.342463	2	timo werner	2019-2020	2795.0	34.0	33.0	Bundesliga	RB Leipzig	...	FW	23	1996	Germa
865	1.234378	-0.351267	2	karim benzema	2021-2022	2593.0	32.0	31.0	La Liga	Real Madrid	...	FW	33	1987	Fran
1365	1.171932	-0.322932	2	ciro immobile	2019-2020	3170.0	37.0	36.0	Serie A	Lazio	...	FW	29	1990	It

5.3 Closest Performing Players in Current Season(22-23)

Finding top 5 players with similar performances as Mbappe for season 2022-2023:

Players in Similar Cluster to Mbappe for 22/23 season using Kmeans

```
[653]: #Closest Performing players in current season 2022-2023
# Filter the DataFrame by players in the same cluster as Mbappe
df_mbappe_cluster_2223 = df_reduced[(df_reduced['cluster'] == mbappe_cluster) & (df_reduced['season'] == '2022-2023')]

# Display the DataFrame
# df_mbappe_cluster.sort_values(by=['dist_mbappe_centroid'], ascending=True).head(10)
df_mbappe_cluster_2223[~df_mbappe_cluster_2223['name'].str.contains('kylian mbappe', case=False)].sort_values(by=['dist
## Closest performing players in all seasons
```

:[653]:

	x	y	cluster	name	season	minutes	matches_played	matches_started	comp	team	...	pos	age	birth_year	national
1263	1.094470	-0.426474	3	lionel messi	2022-2023	1847.0	21.0	21.0	Ligue 1	Paris S-G	...	FW,MF	35	1987	Argenti
603	1.307279	0.445813	3	erling haaland	2022-2023	2022.0	25.0	24.0	Premier League	Manchester City	...	FW	22	2000	Norw
941	0.994545	0.401820	3	robert lewandowski	2022-2023	1587.0	20.0	19.0	La Liga	Barcelona	...	FW	34	1988	Pola
1605	1.000987	0.497166	3	victor osimhen	2022-2023	1748.0	21.0	20.0	Serie A	Napoli	...	FW	24	1998	Nige
1282	1.072350	0.606469	3	wissam ben yedder	2022-2023	1333.0	22.0	18.0	Ligue 1	Monaco	...	FW	32	1990	Fran

Players in Similar Cluster to Mbappe for 22/23 season using GMM

```
In [656]: #Closest Performing players in current season 2022-2023
# Filter the DataFrame by players in the same cluster as Mbappe
gmm_df_mbappe_cluster_2223 = gmm_df_reduced[(gmm_df_reduced['cluster'] == gmm_mbappe_cluster) & (gmm_df_reduced['season
# Display the DataFrame
# df_mbappe_cluster.sort_values(by=['dist_mbappe_centroid'], ascending=True).head(10)
gmm_df_mbappe_cluster_2223[~gmm_df_mbappe_cluster_2223['name'].str.contains('kylian mbappe', case=False)].sort_values(b
## Closest performing players in all seasons
```

Out[656]:

	x	y	cluster	name	season	minutes	matches_played	matches_started	comp	team	...	pos	age	birth_year	national
1263	1.094470	-0.426474	2	lionel messi	2022-2023	1847.0	21.0	21.0	Ligue 1	Paris S-G	...	FW,MF	35	1987	Argenti
603	1.307279	0.445813	2	erling haaland	2022-2023	2022.0	25.0	24.0	Premier League	Manchester City	...	FW	22	2000	Norw
941	0.994545	0.401820	2	robert lewandowski	2022-2023	1587.0	20.0	19.0	La Liga	Barcelona	...	FW	34	1988	Pola
1605	1.000987	0.497166	2	victor osimhen	2022-2023	1748.0	21.0	20.0	Serie A	Napoli	...	FW	24	1998	Nige
1282	1.072350	0.606469	2	wissam ben yedder	2022-2023	1333.0	22.0	18.0	Ligue 1	Monaco	...	FW	32	1990	Fran

```
In [654]: df_mbappe_cluster_2223.shape
Out[654]: (18, 21)
```

As per K-means this season there are 18 players that are similar to Mbappe in terms of performance.

```
In [658]: gmm_df_mbappe_cluster_2223.shape
Out[658]: (14, 21)
```

As per GMM this season there are 14 players that are similar to Mbappe in terms of performance.

5.4 Closest Performing Young Players (u23) in Current Season(22-23)

Finding young players (u23) with similar performances as Mbappe for season 2022-2023:

Using K-means

```
In [666]: #Closest Performing Young players (Under 25) in current season 2022-2023
# Filter the DataFrame by players in the same cluster as Mbappe
df_mbappe_cluster_u23 = df_reduced[(df_reduced['cluster'] == mbappe_cluster) & (df_reduced['age'] <= 23) & (df_reduced[
# Display the DataFrame
# df_mbappe_cluster.sort_values(by=['dist_mbappe_centroid'], ascending=True).head(10)
df_mbappe_cluster_u23[~df_mbappe_cluster_u23['name'].str.contains('kylian mbappe', case=False)].sort_values(by=['dist_m
## Closest performing players in all seasons
```

Out[666]:

	x	y	cluster	name	season	minutes	matches_played	matches_started	comp	team	...	pos	age	birth_year	nationality
603	1.307279	0.445813	3	erling haaland	2022-2023	2022.0	25.0	24.0	Premier League	Manchester City	...	FW	22	2000	Norway
1584	0.587614	-0.134980	3	khvicha kvaratskhelia	2022-2023	1551.0	21.0	20.0	Serie A	Napoli	...	FW	22	2001	Georgia
1256	0.582418	0.080077	3	jonathan david	2022-2023	2112.0	25.0	24.0	Ligue 1	Lille	...	FW	23	2000	Canada

3 rows × 21 columns

```
In [663]: df_mbappe_cluster_u23.shape
Out[663]: (3, 21)
```

As per K Means there are 3 players Erling Haaland, khvicha kvaratskhelia and Jonathan David,that are similar to Mbappe .

Using GMM

```
In [664]: #Closest Performing Young players (Under 25) in current season 2022-2023
          # Filter the DataFrame by players in the same cluster as Mbappe
          gmm_df_mbappe_cluster_u23 = gmm_df_reduced[(gmm_df_reduced['cluster'] == gmm_mbappe_cluster) & (gmm_df_reduced['age'] <

          # Display the DataFrame
          # df_mbappe_cluster.sort_values(by=['dist_mbappe_centroid'], ascending=True).head(10)
          gmm_df_mbappe_cluster_u23[~gmm_df_mbappe_cluster_u23['name'].str.contains('kylian mbappe', case=False)].sort_values(by=
          ## Closest performing players in all seasons
```

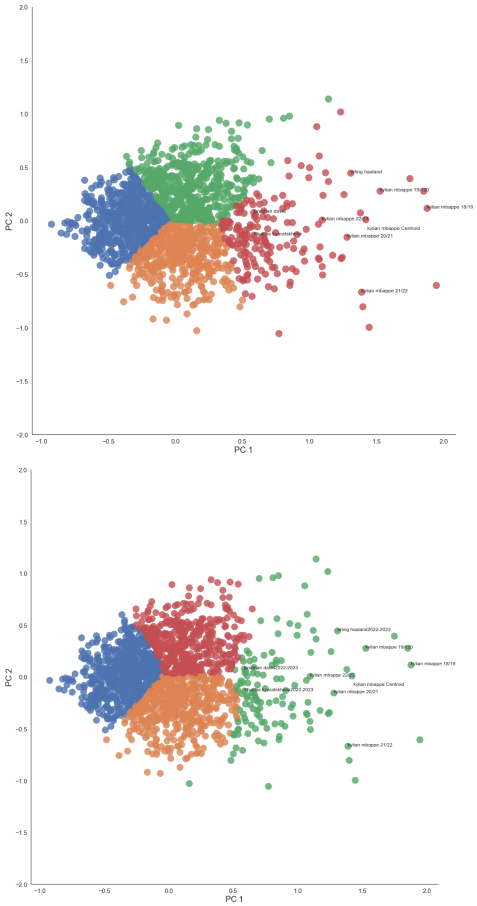
Out[664]:

	x	y	cluster	name	season	minutes	matches_played	matches_started	comp	team	...	pos	age	birth_year	nationality
603	1.307279	0.445813	2	erling haaland	2022-2023	2022.0	25.0	24.0	Premier League	Manchester City	...	FW	22	2000	Norway
1584	0.587614	-0.134980	2	khvicha kvaratskhelia	2022-2023	1551.0	21.0	20.0	Serie A	Napoli	...	FW	22	2001	Georgia
1256	0.582418	0.080077	2	jonathan david	2022-2023	2112.0	25.0	24.0	Ligue 1	Lille	...	FW	23	2000	Canada

3 rows × 21 columns

```
In [665]: gmm_df_mbappe_cluster_u23.shape
Out[665]: (3, 21)
```

As per K Means there are 3 players Erling Haaland, khvicha kvaratskhelia and Jonathan David, that are similar to Mbappe.



By looking at the plots we can see that Erling Haaland Performances have been closest to Kylian Mbappe this Year.

6. CONCLUSION

The outcome of this analysis was to identify players similar to Kylian Mbappe over past 5 seasons across top 5 European Football Leagues.

- We found that Erling Haaland, khvicha kvaratskhelia and Jonathan David are young players under 23 that have similar performances this season(2022-2023).
- Top 5 Players Lionel Messi, Robert Lewandowski, Erling Haaland, victor osimhen and wissam ben yedder are performing similar to Mbappe this season(2022-2023).
- Top 5 players over the five seasons 2018-2019 with similar performances like Mbappe are:
    1. robert lewandowski (2019-2020)
    2. robert lewandowski (2021-2022)
    3. timo werner (2019-2020)
    4. karim benzema (2021-2022)
    5. ciro immobile (2019-2020)
- Over Past 5 seasons K-Means algorithm clustered 174 players and GMM 128 Players into Mbappe's cluster.
- In the 2022-2023 season, As per GMM 14 players have similar performances to Mbappe.
- In the 2022-2023 season, As per K-Means 13 players have similar performances to Mbappe.

7. FUTURE IMPROVEMENTS OR DIRECTIONS

- The project could be enhanced to include transfer data and could be used in player recruitment, by comparing stats from similar players and use it to recruit next best players for the club. The player valuation model could be used to estimate a player's market value.
- This code currently measures proximity to clusters but can be converted to percentage score to find how similar a player is to the comparison player
- Implement other clustering algorithms like DBSCAN and Hierarchical Clustering.
- Improve the identification of the characteristics that influence Mbappe's playing style. Currently, the model simplifies the selected performance metrics into two dimensions, which are chosen arbitrarily for visualisation purposes. It doesn't explain why a player is achieving results similar to Mbappe, only that they belong to the same cluster and are in close proximity.

8. REFERENCES

1. A robust method for clustering football players with mixed attributes by Pierpaolo D'Urso, Livia De Giovanni & Vincenzina Vitale
   Annals of Operations Research (2022)
2. Clustering soccer players: investigating unsupervised learning on player positions
   - Gijs Wijngaard - Utrecht university
3. CLUSTERING TEAM PROFILES IN THE AUSTRALIAN FOOTBALL LEAGUE USING PERFORMANCE INDICATORS
   July 2016
   Conference: Proceedings of the 13th Australasian Conference on Mathematics and Computers in SportAt: Melbourne, Australia
4. Clustering of football players based on performance data and aggregated clustering validity indexes
   Serhat Akhanli, Christian Hennig
5. Data Analytics in Football - Positional Data Collection, Modelling and Analysis - Dominik Raabe, Daniel Memmert