# FOOTBALL GAME ANALYSIS

## Final Document-J component

Prithin Devotta (19BCE0663)
Keerthana Rajeev (19BCE2448)
Oruganti Shiva charan (19BCE0545)
Aditi Adgaonkar (19BCE2427)
Kanishk Srivastava (19BCE2408)

# Introduction to the domain

Football teams are described as groups that interact in a dynamic and interdependent way to achieve their common goal. Individual performance of players does matter a lot. We will visualize a dataset and generate relevant graphs (barchart, heatmap and piechart) to find out the best player who can play for the team at a tactical position. This will increase the overall performance of the team. This will include python and libraries.

# Problem Statement

Analysis of each team member’s individual performance is very difficult. The way the different team members' integrate together is not easily observed. Not knowing how the team members integrate together and not finding out the team and the players strengths and weaknesses will cause problems. There would be a decrease in the overall performance of the team. It can also be time consuming trying to do hit and trial for assigning team members to the right position. This problem is observed when narrowing the top 11 and choosing the right players to place on the football field. This problem affects the team building decision and leads to poor salary allocation. This also affects the players as they are placed in positions which could be bettered. This problem is observed when there is less performance of the team, poor passing and less accuracy of the players.

# Part 1 - Prediction

## Data Abstraction

**Method-**

- https://www.kaggle.com/louischen7/football-results-and-betting-odds-data-of-epl
- We went through various news channel articles and we picked the one which has the highest views.

- **DATA REPRESENTATION-**

- The data set taken is presented in a table format consisting of a total of 26 attributes. These present the various positions, points and arithmetic difference between two points.Below is a table consisting of few of them :

| | FTR | HTP | ATP | HM1 | HM2 | HM3 | AM1 | AM2 | AM3 | HTGD | ATGD | DiffFormPts | DiffLP |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **30** | H | 1.25 | 1.00 | D | D | W | D | W | L | 0.50 | 0.25 | 0.25 | -16.0 |
| **31** | NH | 0.75 | 0.25 | L | L | W | D | L | L | -0.50 | -0.75 | 0.50 | -2.0 |
| **32** | H | 1.00 | 1.00 | L | D | W | D | W | L | 0.00 | 0.25 | 0.00 | -3.0 |
| **33** | NH | 0.75 | 0.50 | L | L | W | D | L | D | -0.25 | -0.25 | 0.25 | 3.0 |
| **34** | NH | 1.00 | 1.50 | D | L | W | W | W | L | 0.00 | 0.75 | -0.50 | 3.0 |

-

- **ATTRIBUTES SIMANTICS**

*CATEGORICAL ATTRIBUTES:*

#HM1, #HM2,#HM3,#AM1,#AM2,#AM3

#ATP - Away team points

#DiffFormPts Diff in points

#FTR and Res = Full-Time Result (H=Home Win, D=Draw, A=Away Win)

*ORDINAL :*

#ATGD - away team goal difference

#HTP - Home team points

#HTGD - Home team goal difference

#DiffLP - Difference in last years prediction

These have been further converted into binary values as shown below:

Feature values:

| | HTP | ATP | HM1_D | HM1_L | HM1_W | HM2_D | HM2_L | HM2_W | HM3_D | HM3_L | ... | AM2_D | AM2_L | AM2_W | AM3_D | AM3_L | AM3_W | HTGD | ATGD | DiffFormPts | DiffLP |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **30** | -0.043829 | -0.611968 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | ... | 0 | 0 | 1 | 0 | 1 | 0 | 0.753719 | 0.355995 | 0.25 | -1.989216 |
| **31** | -1.120644 | -2.238746 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | ... | 0 | 1 | 0 | 0 | 1 | 0 | -0.737082 | -1.138834 | 0.50 | -0.248963 |
| **32** | -0.582236 | -0.611968 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | ... | 0 | 0 | 1 | 0 | 1 | 0 | 0.008318 | 0.355995 | 0.00 | -0.373267 |
| **33** | -1.120644 | -1.696487 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | ... | 0 | 1 | 0 | 1 | 0 | 0 | -0.364382 | -0.391419 | 0.25 | 0.372556 |
| **34** | -0.582236 | 0.472551 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | ... | 0 | 0 | 1 | 0 | 1 | 0 | 0.008318 | 1.103409 | -0.50 | 0.372556 |

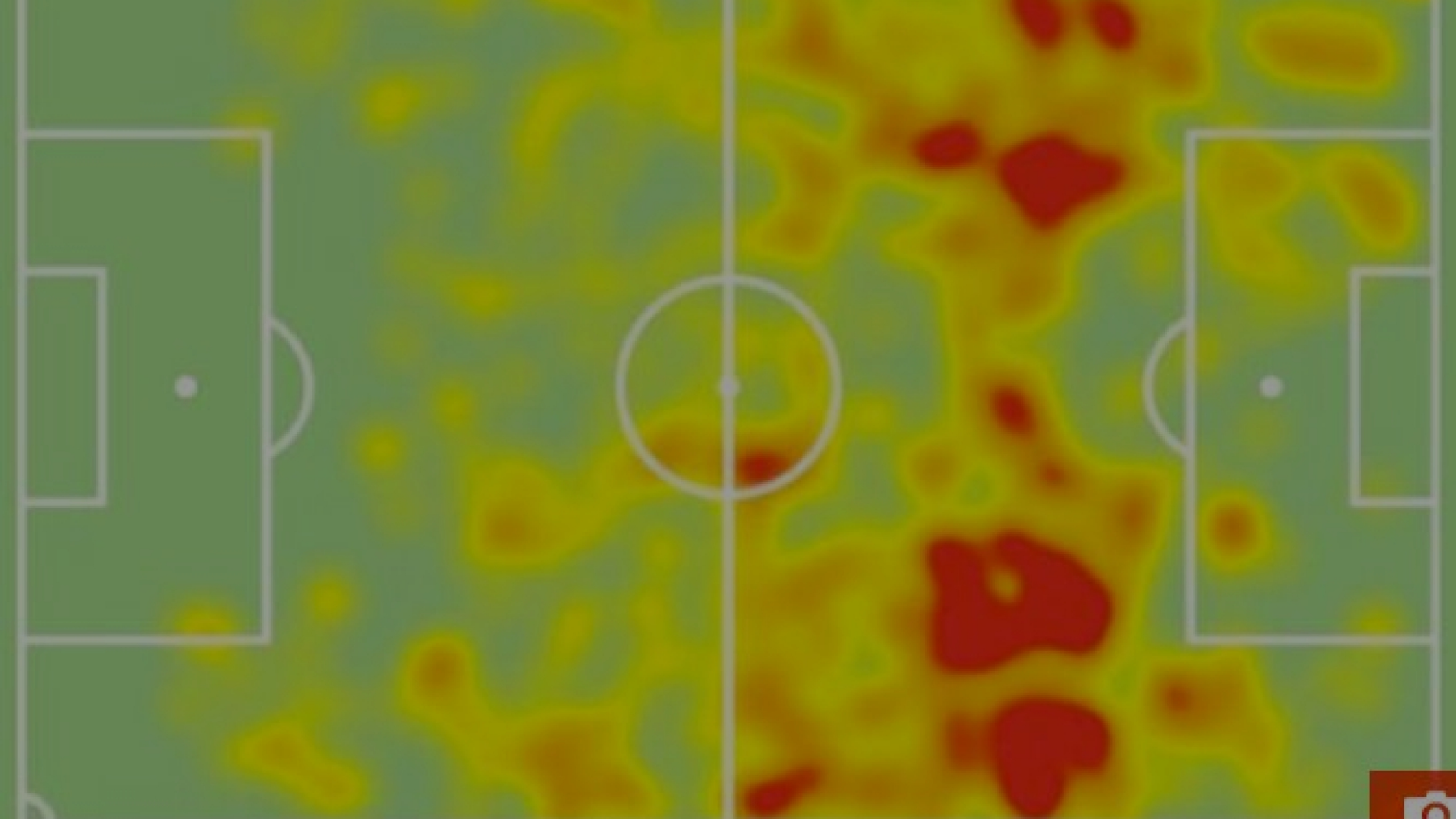
5 rows × 24 columns

# PART-2-Player Heatmaps

## Data Abstraction

**Target Identification-**

- FTR is the target variable of categorical type.
- By shuffling and splitting data into training and testing we can apply all three classifiers one by one on the training models. Few metrics will be applied on the target variable to analyze.



- **Method of abstraction:** These kinds of data of players positions and passing are extracted by analyzing and observing players position on the pitch and the time stamp of the pass made.
- We have made this part of the project around the dataset of a footballer called

  Lionel Messi in a league game by plotting his passes during the whole match.

- A study in a research paper around football players performance proves that the best

  analysis of a team's performance can be done around the player playing in the no. 10 position for the team. So, we chose Lionel Messi's dataset against Real Betis where he played in a right wing + central attaching midfield position.
- The link for the dataset:

- passmap/messibetis.csv at main · mckayjohns/passmap · GitHub

| player | minute | second | x | y | type | outcome | endX | endY |
|---|---|---|---|---|---|---|---|---|
| messi | 45 | 0 | 50 | 50 | Pass | Successful | 40 | 43 |
| messi | 45 | 25 | 63 | 48 | Pass | Successful | 75 | 83 |
| messi | 46 | 4 | 74 | 58 | Pass | Successful | 71 | 65 |
| messi | 46 | 7 | 76 | 68 | Pass | Successful | 95 | 78 |
| messi | 46 | 55 | 100 | 1 | Pass | Unsuccess | 96 | 38 |
| messi | 47 | 42 | 73 | 25 | Pass | Successful | 70 | 54 |
| messi | 49 | 47 | 75 | 43 | Pass | Unsuccess | 82 | 38 |
| messi | 50 | 10 | 79 | 31 | Pass | Successful | 87 | 24 |

- The data is represented in the form of tables with rows and columns. With the cells containing the values.

- With the minute and the seconds we get the exact time at which the player passes the ball and a vector can be plot on the field with (x,y) and (endx, endy) coordinates with the outcome giving the success in the action.

# Attribute Semantics

- The dataset contains 9 different attributes. Name, minute, seconds, x, y, type, outcome, endx, endy.
- Name is the name of the player in our case Messi, categorical datatype.

- Minute and Second: the time at which the pass was made, quantitative datatype.
- Type: represents the type of action on the ball like pass, shoot etc, categorical datatype.
- Outcome: represents the outcome of the action as successful or not, categorical datatype.
- X and Y: x,y coordinates of the player at the time of passing, quantitative datatype.
- Endx and endy: coordinates of the player at the receiving end, quantitative datatype.

# Target Identification

- The target is to visualize the area covered by the player and the amount of passes made at various positions on the pitch.
- The vector representing the passes can be categorized into first and second half passes by giving them different colors.
- The outcome attribute gives the amount of successful passes made by the player.

- With the vectors plot on the pitch it can be easily analyzed in which position the player dominates the most.
- And with the data a heat map can be made on the pitch to visualize the positions and amount of area covered by the player.

# Part 3 - Barplots

**We have constructed Bar Plots for the following : -**

- Top 10 countries of the world with most playing footballers.
- Number of players playing at a particular age.
- Top 5 players for various football position.
- In the final part we have concluded the best possible playing 11 for the team.

# Data Abstraction

**Method of Abstraction:**

- We took the dataset from below link

  https://www.kaggle.com/stefanoleone992/fifa-20-complete-player-

  dataset

- We have taken a dataset and converted it into a CSV file as it is easy to analyze, which had information and data about a particular match. Instead of having information in the form of matrix, which would cause a big chaos because at different timestamps as we have different players at one particular position.

# Data Representation

- The dataset is represented in the form of tables with rows and columns. With the cells containing the values.

| sofifa_id | player_url | short_name | long_name | age | dob | height_cm | weight_kg | nationality | club | player_pos... |
|---|---|---|---|---|---|---|---|---|---|---|
| 158023 | https://sofifa.com/player/158023/lionel-messi/15/157759 | L. Messi | Lionel Andrés Messi Cuccittini | 27 | 1987-06-24 | 169 | 67 | Argentina | FC Barcelona | CF |
| 20801 | https://sofifa.com/player/20801/c-ronaldo-dos-santos-aveiro/15/157759 | Cristiano Ronaldo | Cristiano Ronaldo dos Santos Aveiro | 29 | 1985-02-05 | 185 | 80 | Portugal | Real Madrid | LW, LM |
| 9014 | https://sofifa.com/player/9014/arjen-robben/15/157759 | A. Robben | Arjen Robben | 30 | 1984-01-23 | 180 | 80 | Netherlands | FC Bayern München | RM, LM, RW |
| 41236 | https://sofifa.com/player/41236/zlatan-ibrahimovic/15/157759 | Z. Ibrahimović | Zlatan Ibrahimović | 32 | 1981-10-03 | 195 | 95 | Sweden | Paris Saint-Germain | ST |
| 167495 | https://sofifa.com/player/167495/manuel-neuer/15/157759 | M. Neuer | Manuel Neuer | 28 | 1986-03-27 | 193 | 92 | Germany | FC Bayern München | GK |
| 176580 | https://sofifa.com/player/176580/luis-suarez/15/157759 | L. Suárez | Luis Alberto Suárez Díaz | 27 | 1987-01-24 | 181 | 81 | Uruguay | FC Barcelona | ST, CF |

# Attribute Semantics

- The dataset contains 11 Attributes sofifa_id, Player url , short_name, long_name, Age, Dob, height_cm, weight_kg, Nationality , Club and player_positions.
- Sofifa_id represents the player id, an ordinal datatype.

- Age is a categorical datatype.

- Player_url is the Url for the site which contains the details of the player.

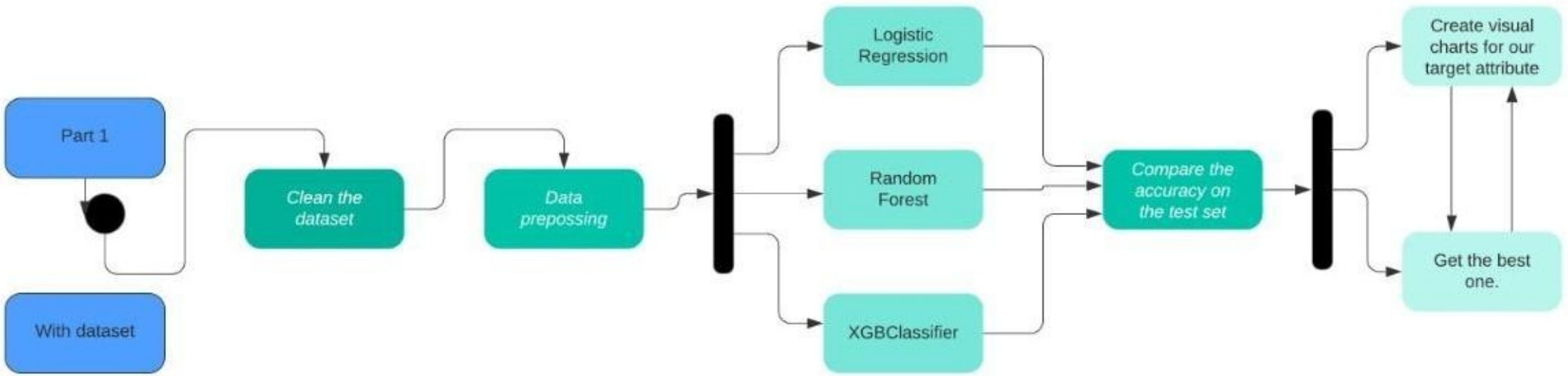
- Player_positions represents the position on the field where the player plays. For eg: - Striking, Defending, Goalkeeper etc., a categorical datatype.
- Nationality and club are categorical datatype.

# Target Identification

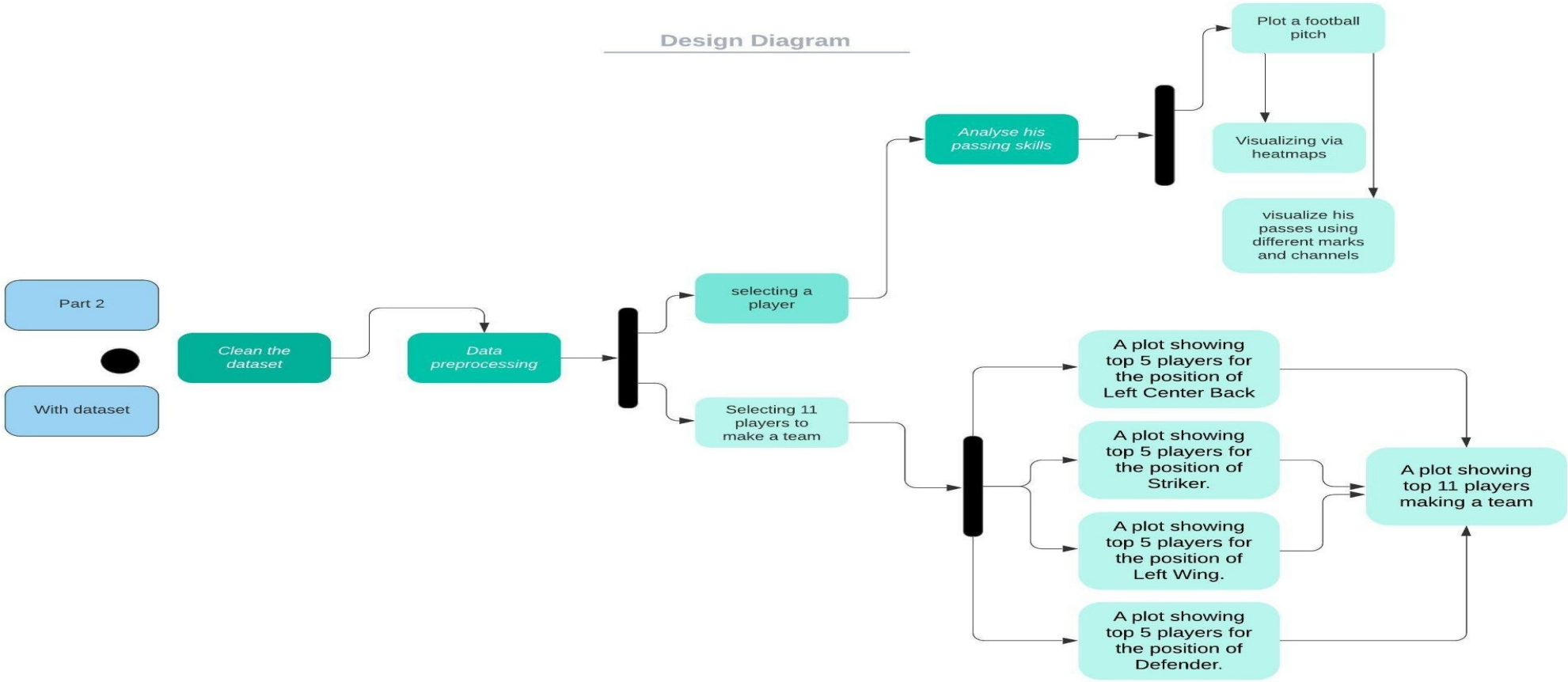
- The target of this experiment is to compare Various players on the basis of their Nationality, Age, and their positions.
- The target attributes are long_name, Age, Dob, Nationality , Club and player_positions.

| long_name | age | dob | nationality | club | player_pos... |
|---|---|---|---|---|---|
| Lionel Andrés Messi Cuccittini | 27 | 1987-06-24 | Argentina | FC Barcelona | CF |
| Cristiano Ronaldo dos Santos Aveiro | 29 | 1985-02-05 | Portugal | Real Madrid | LW, LM |
| Arjen Robben | 30 | 1984-01-23 | Netherlands | FC Bayern München | RM, LM, RW |
| Zlatan Ibrahimović | 32 | 1981-10-03 | Sweden | Paris Saint-Germain | ST |
| Manuel Neuer | 28 | 1986-03-27 | Germany | FC Bayern München | GK |
| Luis Alberto Suárez Díaz | 27 | 1987-01-24 | Uruguay | FC Barcelona | ST, CF |
| Eden Hazard | 23 | 1991-01-07 | Belgium | Chelsea | LM, RM |

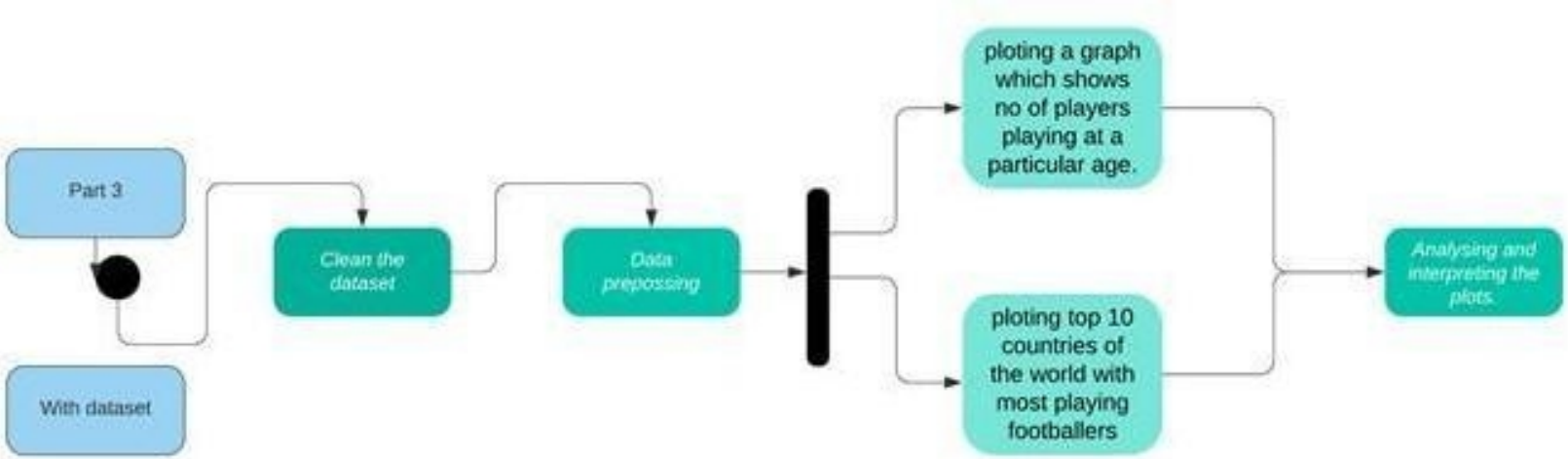
# Design of proposed systems



# Design of proposed systems



# Design of proposed systems



# Part 1 - Prediction

## **TASKS** :

Will the home team win on its home ground?

## **ACTIONS**:

### ANALYSE:

Produce(Record) : the record goals uses existing information to produce artifacts. These artifacts can be in the form of logs, screenshots, calculations. We will be training classifiers based on existing datasets to derive best results and predict whether the home team wins or not.

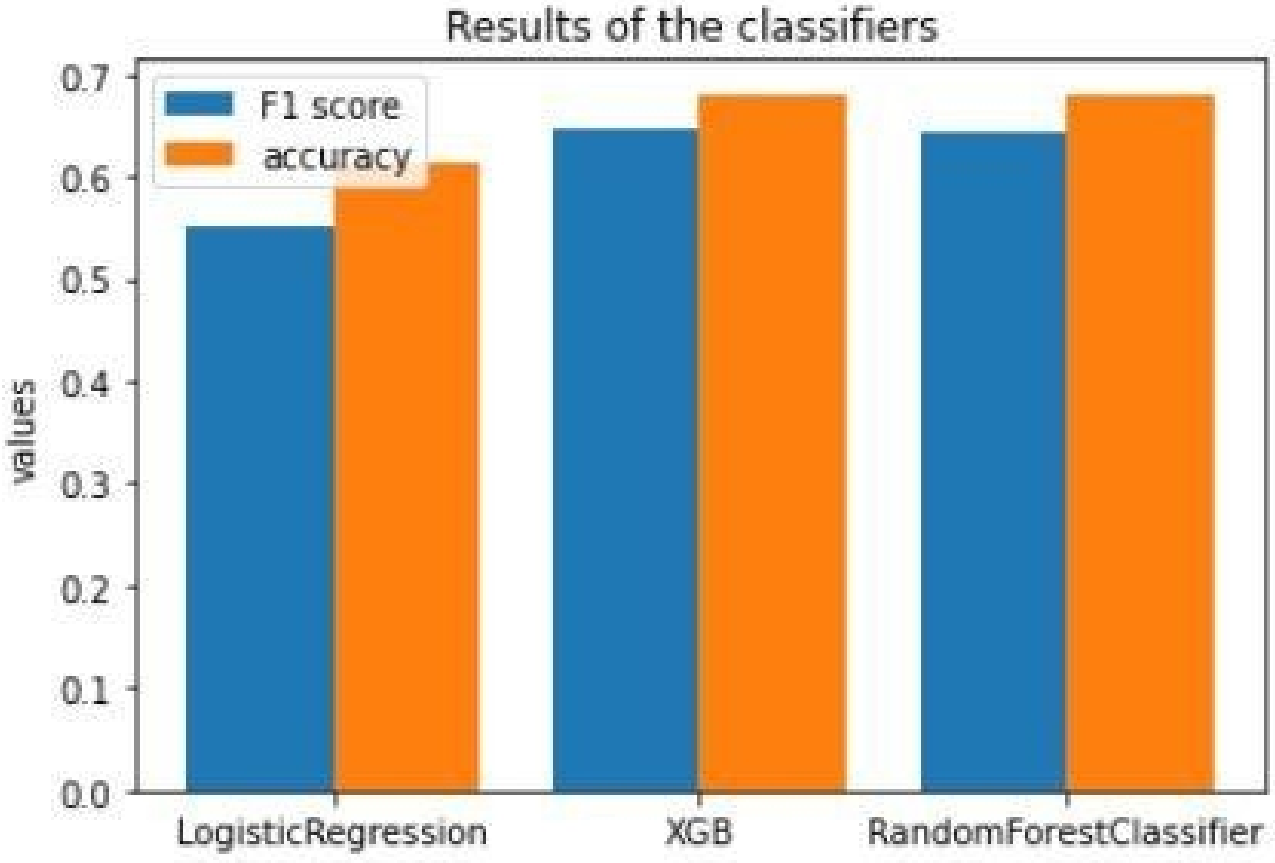
### SEARCH:

Locate: The users have to locate the classifiers . We have supported this by displaying the percentages in the bar graph..

### QUERY:

Users can identify and compare the results. For that we have used different colours for different classifiers.

# Visualization Screenshots



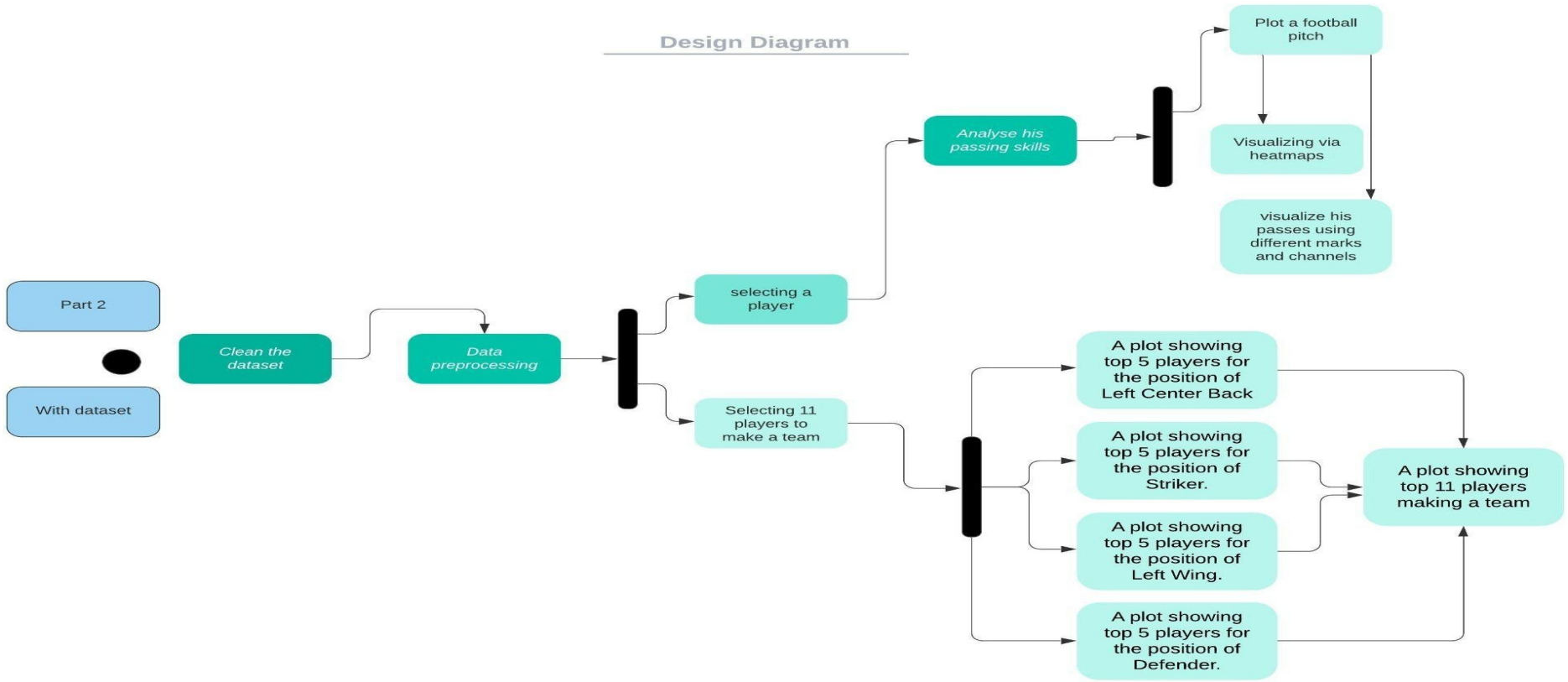
# Output

```
Training a LogisticRegression using a training set sizeof 5560. . .
0.5512927439532945 0.6129496402877698
F1 score and accuracy score for training set: 0.5513 , 0.6129.
F1 score and accuracy score for test set:0.6471 , 0.7000.

Training a XGBClassifier using a training set sizeof 5560. . .
[11:27:32] WARNING: ../src/learner.cc:1095: Starting in XGBoost 1.3.0, the default evaluati
on metric used with the objective 'binary:logistic' was changed from 'error' to 'logloss'.
Explicitly set eval_metric if you'd like to restore the old behavior.
```

```
0.6481884775292021 0.6803956834532374
F1 score and accuracy score for training set: 0.6482 , 0.6804.
F1 score and accuracy score for test set:0.5128 , 0.5250.

Training a RandomForestClassifier using a training set sizeof 5560. . .
0.6422422422422422 0.6785971223021583
F1 score and accuracy score for training set: 0.6422 , 0.6786.
F1 score and accuracy score for test set:0.5263 , 0.5500.
```

# Part 2 - Heat maps

## Algorithm design:



# Task Abstraction

**Task of this visualization:**

Task is to understand and analyze the players in a football team. It is necessary to understand how players in a team are performing, and to know if a player is actually performing in the area he is made for.

**Action**

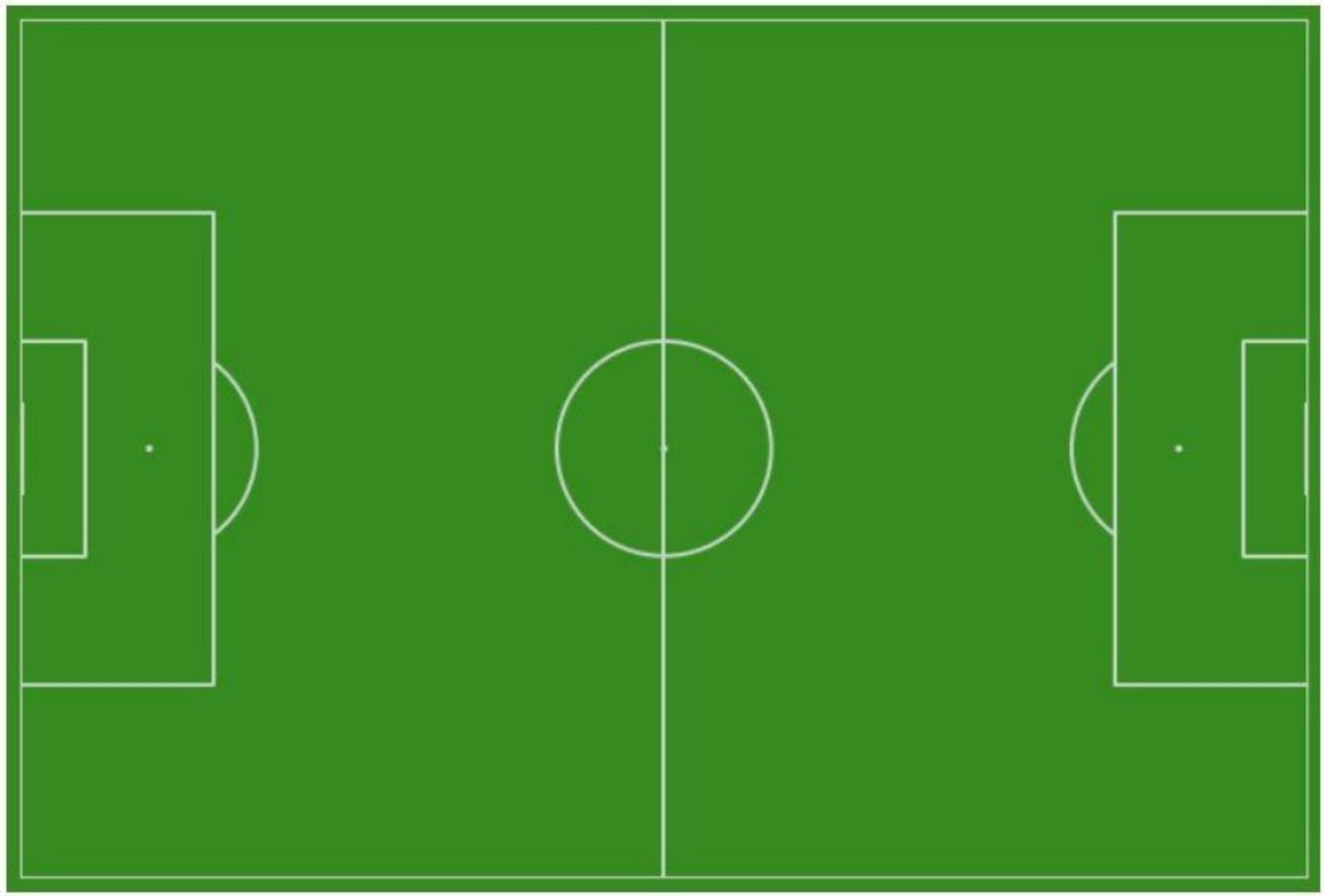
- Our task is to visualize the existing data in such a way that it will be made easy to **analyze** and **present** the relevant information about the player to the user.
- The visualization will let the user to just **look-up** for the stats he is searching for.
- Finally, the user will be able to **compare** between various players using their corresponding heatmaps.

**Visual idioms:**

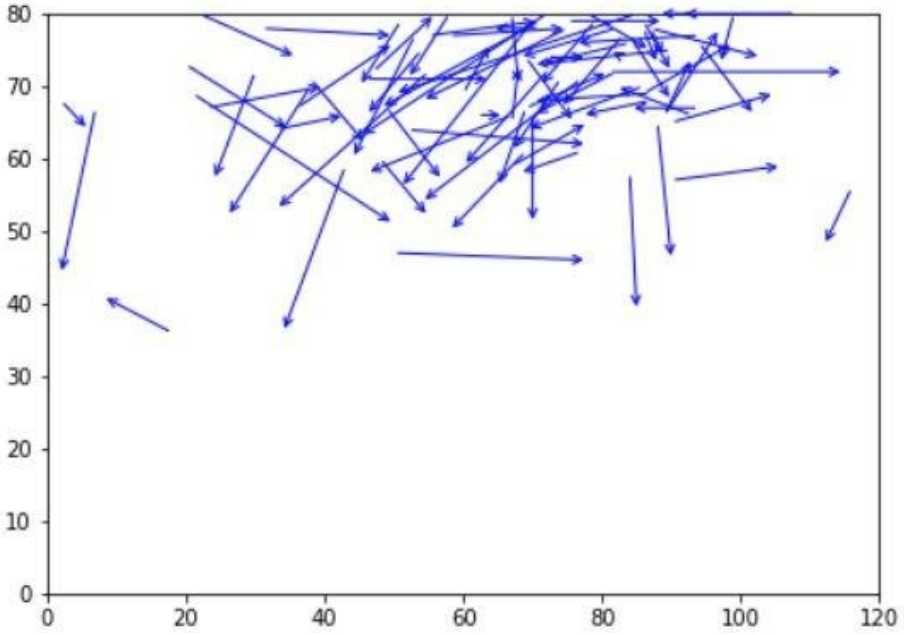
- We'll be using lines and arrows to represent the passes made by a player and differentiate them into 1st half and 2nd half using colors.

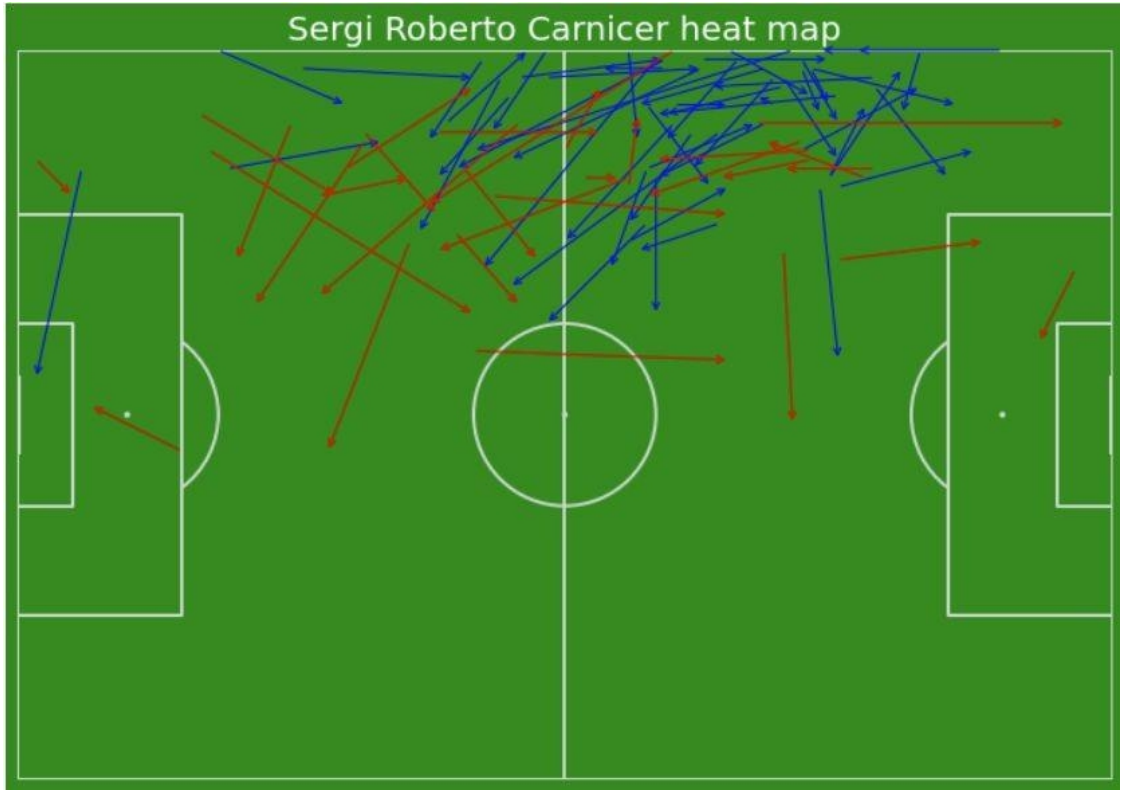
We have plotted a football pitch with dimensions 120m X 80m using Matplotlib library. The football pitch includes the pitch outline, center line, left wing and right wing penalty area, 6 yard boxes.

```
Out[116]: (0.0, 80.0)
```

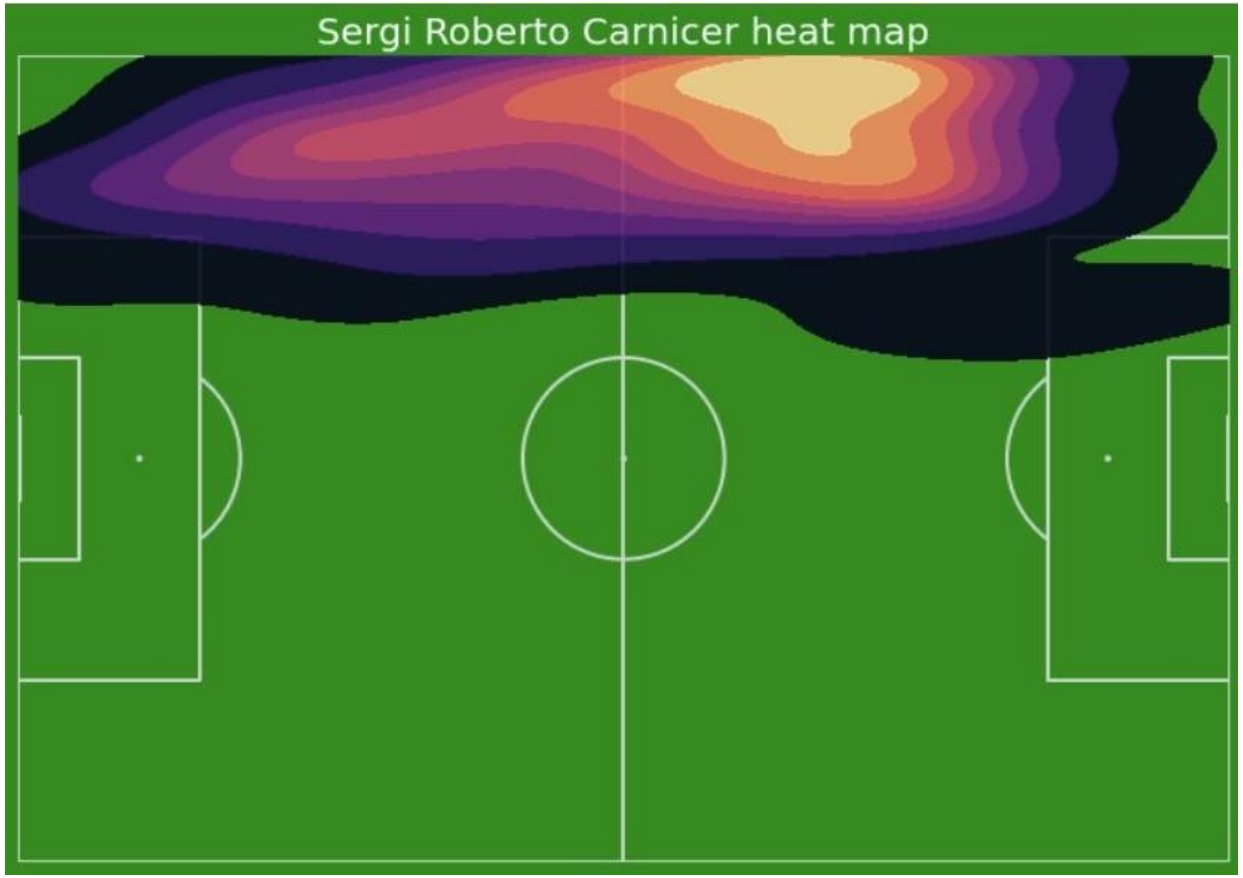


We have considered a player named Ivan sergi roberto and we have plotted his passes during the whole match.

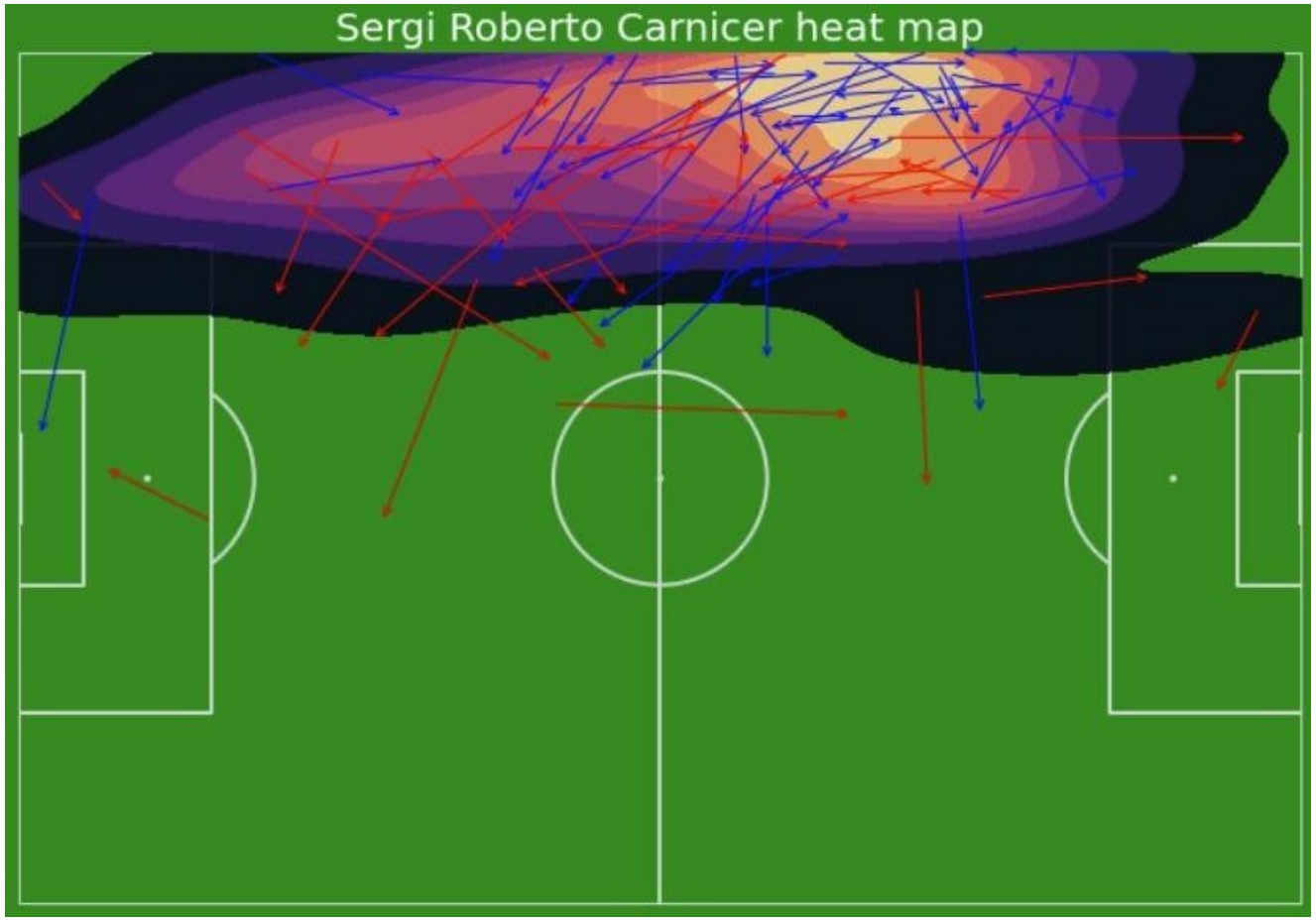


Then we have plotted the heat map for the player sergi roberto considering his position during the entire duration of the football game.



We have combined both the heat map and pass plots into one plot. This helps us to analyze the main position that the player plays at and where the player misses to pass the ball successfully under one single function.



# Results and Analysis

So now interplay in each ball possession is examined separately instead of evaluating an aggregated passing matrix at match-level. Playing positions are tracked to facilitate an evaluation of the individual contribution of players in our study.

# PART-3
# Task Abstraction

**Task 1:** To know the no of players playing at a particular age.

**Actions:**

Present: The data is already understood by users. Presentation using vis can take place within the context of decision making. We have used a Bar graph as we have a target varible age.

Locate: The users have to locate the no of players at a particular age. We have supported this by displaying the age attribute on x axis with there respective values in ascending order.

**Compare**:

Users may compare Between two targets so, we have supported this by associating different range of colours to the age attribute.

**Targets:**

Only one target attribute which is the age of type distribution.

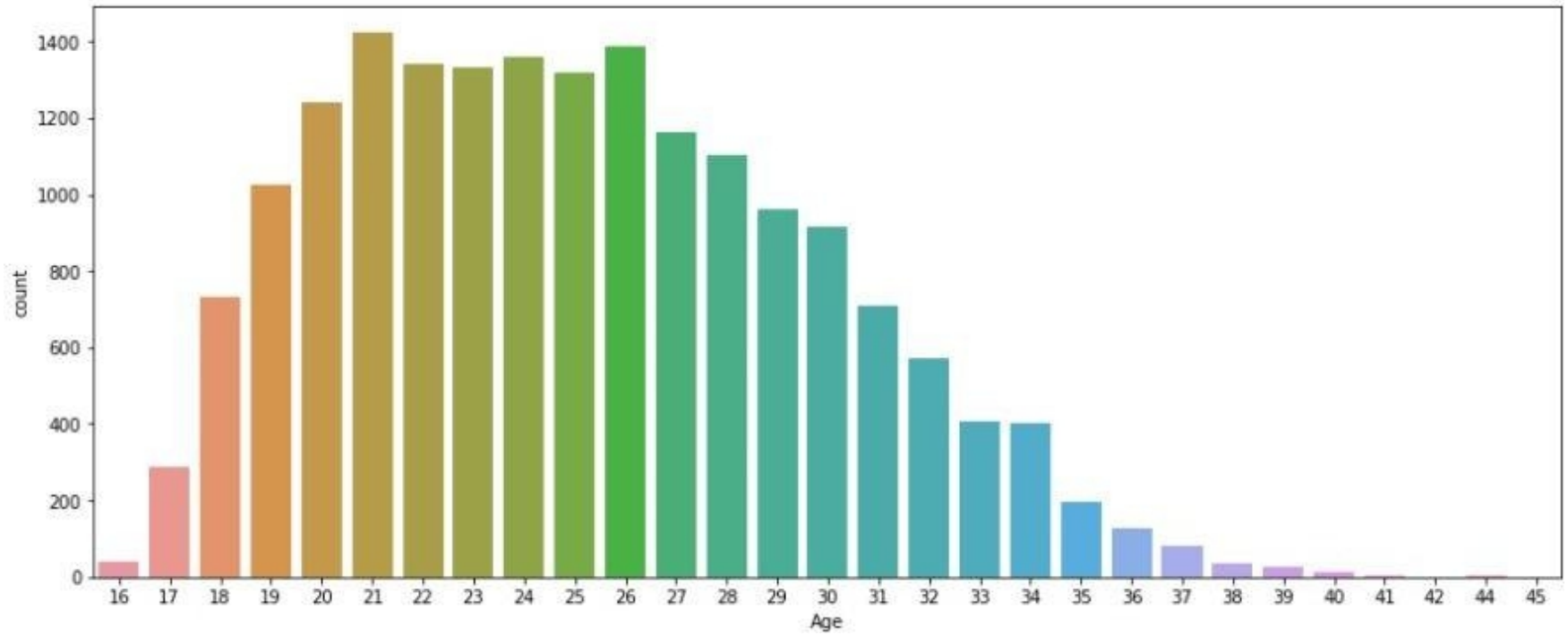
**Trends:**

They would like to see the fluctuations in target attribute.

**Outliers:**

Graph is equipped to see all the outliers.

# Visualization



**Task 2: -**

Top 10 Countries with most playing footballers:-

**Actions:**

Present:

The data is already understood by users. Presentation using vis can take place within the context of decision making. We have used a pie graph as we have the target varible as no of people playing.

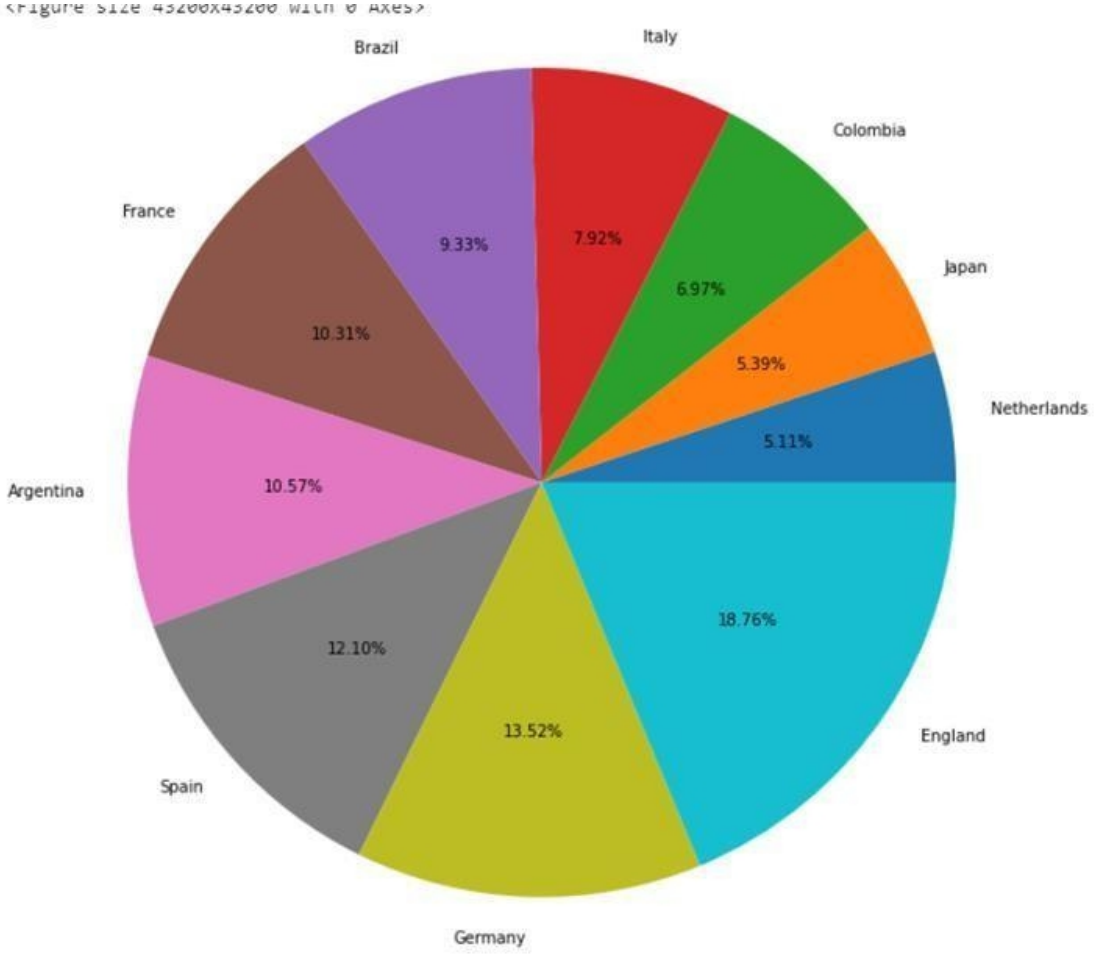
Locate:

The users have to locate the no of players playing at a particular countries . We have supported this by displaying the percentages instead of numbers in there respective blocks..

Compare:

Users may compare Between two targets so, we have supported this by associating different range of colors to each block in the pie graph .

# Visualization

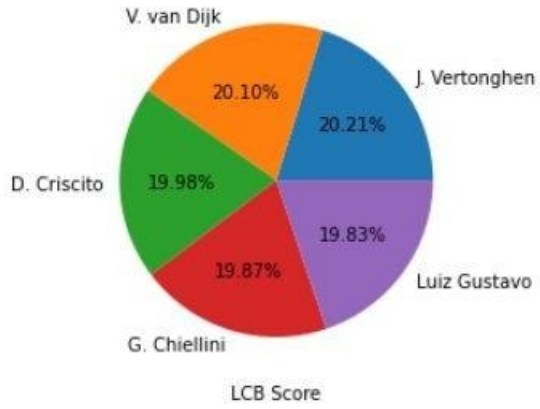


# 1. This is a plot showing top 5 players for the position of Left Center Back.

Task:

Compare and identify the composition of the top 5 players from the Left Center Back and visualize each player as part of a whole. Determining the percentage of each player's left centre back score.

```
[42]: Text(0.5, 0, 'LCB Score')
      <Figure size 1080x432 with 0 Axes>
```



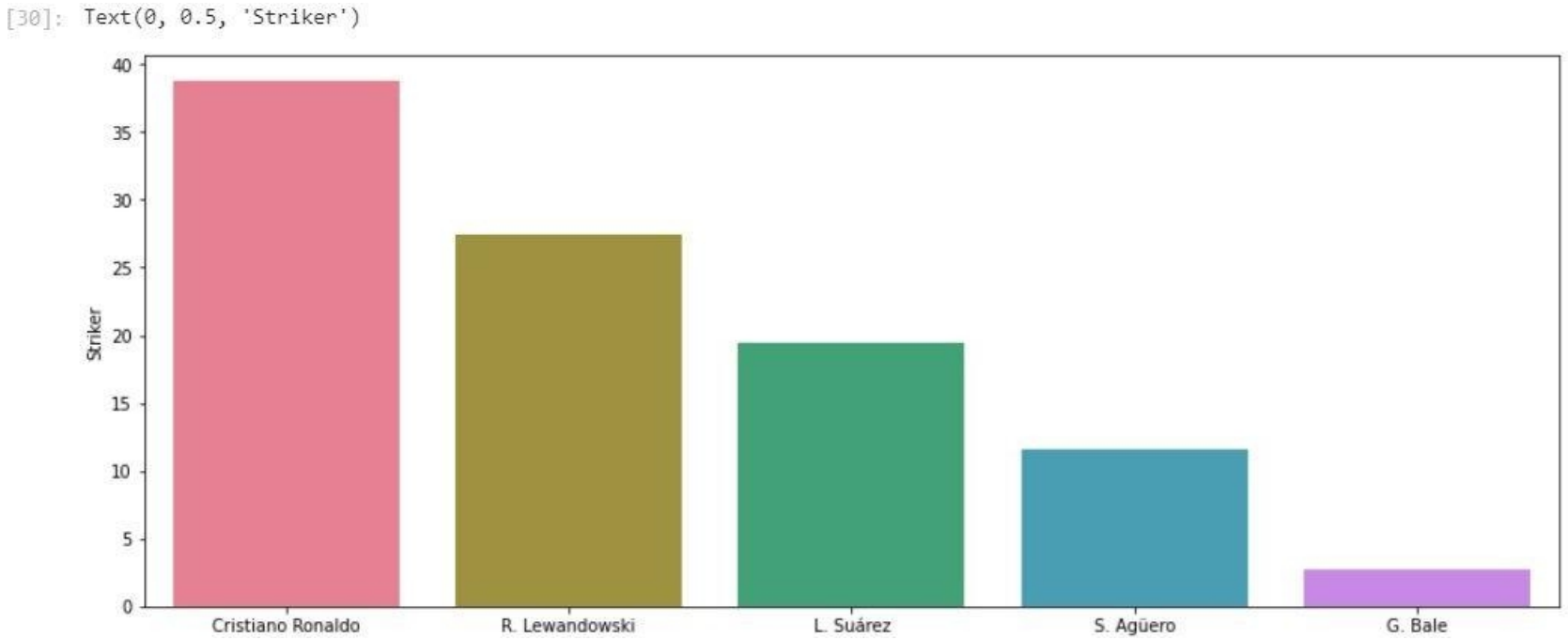

+ Code + Markdown

```
#Left center back
df['df_centre_backs'] = ( d*df['Reactions'] +
c*df['Interceptions'] + d*df['SlidingTackle']  +
d*df['StandingTackle'] + b*df['Vision']+ b*df['Composure'] +
b*df['Crossing'] +a*df['ShortPassing'] + b*df['LongPassing']+
c*df['Acceleration'] + b*df['SprintSpeed']
+ d*df['Stamina'] + d*df['Jumping'] + d*df['HeadingAccuracy'] +
b*df['LongShots'] + d*df['Marking'] +
c*df['Aggression'])/(6*b + 3*c + 7*d)
df['df_wb_Wing_Backs'] = (b*df['BallControl'] +
a*df['Dribbling'] + a*df['Marking'] + d*df['SlidingTackle'] +
d*df['StandingTackle'] + a*df['Positioning'] +
c*df['Vision'] + c*df['Crossing'] + b*df['ShortPassing'] +
c*df['LongPassing'] + d*df['Acceleration'] +d*df['SprintSpeed'] +
c*df['Stamina'] + a*df['Finishing'])/(4*a + 2*b + 4*c +
4*d)

plt.figure(figsize=(15,6))
sd = df[(df['Position'] =='LCB')].sort_values('df_centre_backs',
ascending=False)[:5]
x2 = np.array(list(sd['Name']))
y2 = np.array(list(sd['df_centre_backs']))
fig1, ax1 = plt.subplots()
ax1.pie(y2,labels=x2, autopct='%1.2f%%')
plt.xlabel("LCB Score")
```

## 2. This is a plot showing top 5 players for the position of Striker.

Task:

Compare the skills of the top 5 players from the Striker position and visualize which player has a higher striking score.



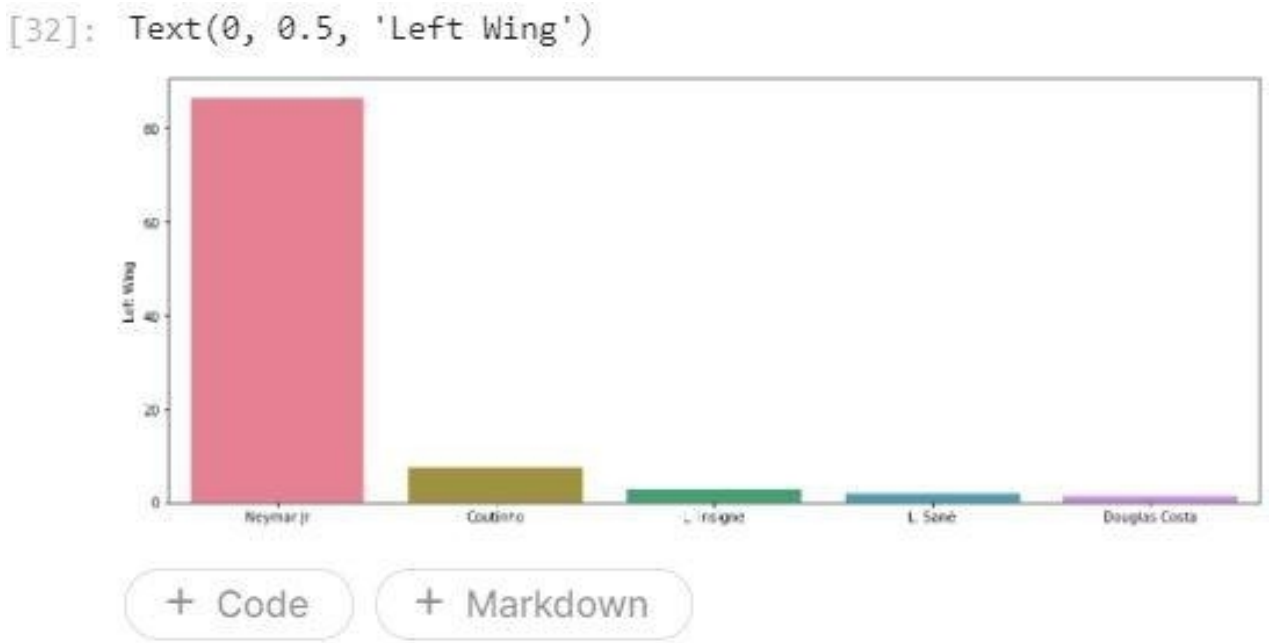
[33]:

```
#Strikers
df['att_striker'] = (b*df['Weak Foot'] +
b*df['BallControl'] + a*df['Vision'] + b*df['Aggression'] +
b*df['Agility'] + a*df['Curve'] + a*df['LongShots'] +
d*df['Balance'] + d*df['Finishing'] + d*df['FKAccuracy'] + d*df['HeadingAccuracy'] +
c*df['Jumping'] + c*df['Dribbling'])/(3*a + 4*b + 2*c +
3*d)

plt.figure(figsize=(15,6))
ss = df[(df['Position'] == 'ST') |
(df['Position'] == 'LS') | (df['Position']
== 'RS') | (df['Position'] ==
'CF')].sort_values('att_striker',
ascending=False)[:5]
x3 = np.array(list(ss['Name']))
y3 = np.array(list(ss['att_striker']))
yyy1 = softmax(y3)*100
yy1 = (y3 - np.mean(y3)) + (np.mean(y3) - np.min(y3))
sns.barplot(x3, yyy1, palette= "husl")
plt.ylabel("Striker")
```

## 3. This is a plot showing top 5 players for the position of Left Wing.

Task:

Compare the skills of the top 5 players from the Left wing position and visualize which player has a higher score in the left wing and which has lower.



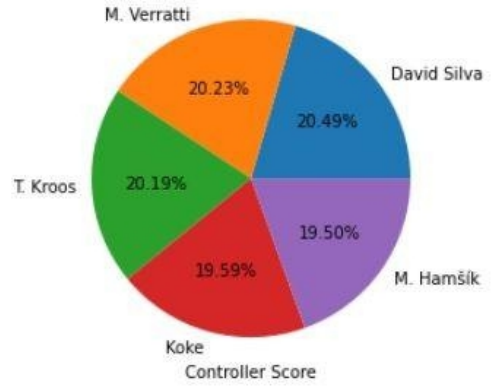
```
#Attackers left wing
df['att_left_wing'] = (c*df['Weak Foot'] +
c*df['BallControl'] + c*df['Dribbling'] + c*df['SprintSpeed'] +
d*df['Acceleration'] + b*df['Vision'] + c*df['Crossing'] +
b*df['ShortPassing'] + b*df['LongPassing'] + b*df['Aggression'] +
b*df['Agility'] + a*df['Curve'] + c*df['LongShots'] +
b*df['HeadingAccuracy'] + d*df['Finishing'] + b*df['FKAccuracy'])/(a + 6*b +
6*c + 2*d)

plt.figure(figsize=(15,6))
ss = df[(df['Position'] == 'LW') |
(df['Position'] == 'LM') | (df['Position']
== 'LS')].sort_values('att_left_wing',
ascending=False)[:5]
x1 = np.array(list(ss['Name']))
y1 = np.array(list(ss['att_left_wing']))
yyy1 = softmax(y1)*100
yy1 = (y1 - np.mean(y1)) + (np.mean(y1) - np.min(y1))
sns.barplot(x1, yyy1, palette= "husl")
plt.ylabel("Left Wing")
```

## 4. This is a plot showing top 5 players for the position of Defender.

Task:

Compare and identify the composition of the top 5 players from the Defender position and visualize each player as part of a whole. Determining the percentage of each player's defending score.

```
[45]: Text(0.5, 0, 'Controller Score')
      <Figure size 1080x432 with 0 Axes>
```



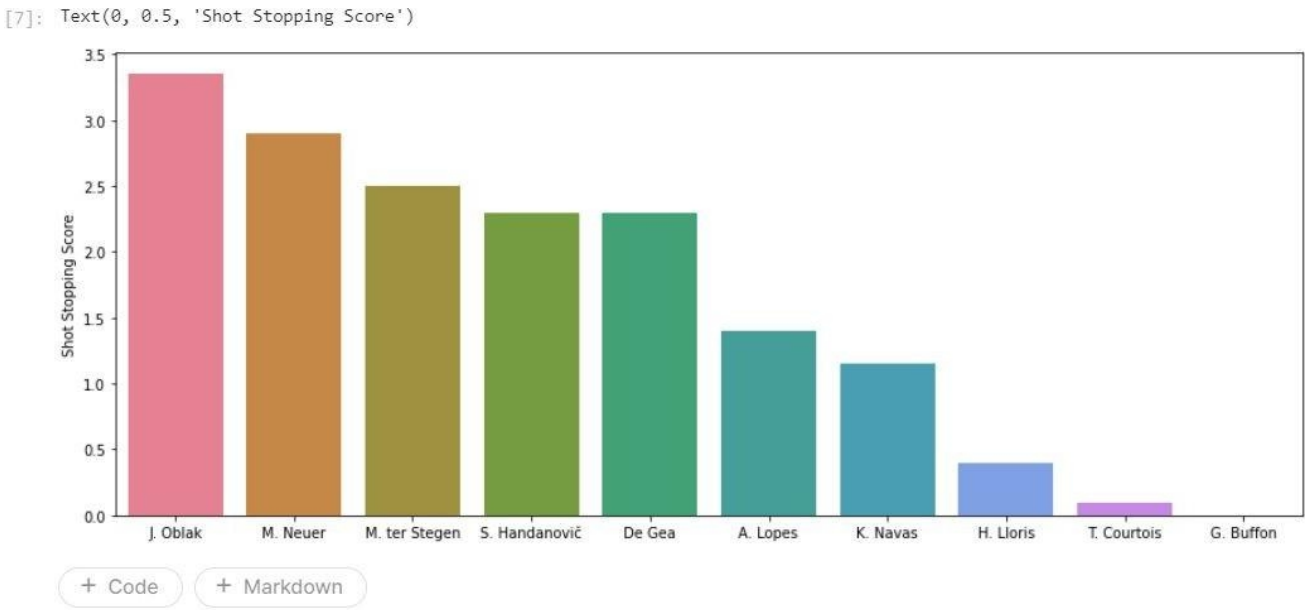

+ Code + Markdown

```
#Defender
df['mf_controller'] = (b*df['Weak Foot'] +
d*df['BallControl'] + a*df['Dribbling'] + a*df['Marking'] +
a*df['Reactions'] + c*df['Vision'] + c*df['Composure'] +
d*df['ShortPassing'] + d*df['LongPassing'])/(2*c + 3*d + 4*a)

import matplotlib.pyplot as plt
plt.figure(figsize=(15,6))

ss = df[(df['Position'] == 'LCM') |
(df['Position'] ==
'LM')].sort_values('mf_controller',
ascending=False)[:5]
x1 = np.array(list(ss['Name']))
y1 = np.array(list(ss['mf_controller']))
fig1, ax1 = plt.subplots()
ax1.pie(y1,labels=x1, autopct='%1.2f%%')
ax1.axis('equal')
plt.xlabel("Controller Score")
```

## 5. This is a plot showing top 5 players for the position of Left Center Back.

Task:

Compare and identify the composition of 10 players from the Left Center Back and visualize each player as part of a whole. Determining the percentage of each player's score.



[7]:

```
a = 1
b = 2
c = 3
d = 4
#GoalKeeping Characteristics
df['gk_Shot_Stopper'] = ( d*df['GKReflexes'] +
c*df['GKDiving'] + c*df['Jumping'] + b*df['Composure'] +
b*df['Reactions'] + a*df['Strength'] + a*df['SprintSpeed'] +
b*df['GKPositioning']+ b*df['GKHandling'] )/(2*a + 4*b +2*c + 1*d)

df['gk_Sweeper'] = (b*df['Reactions'] + b*df['Composure'] +
b*df['SprintSpeed'] + a*df['ShortPassing'] + a*df['LongPassing'] +
b*df['Jumping'] + b*df['GKPositioning'] + b*df['GKDiving'] +
d*df['GKReflexes'] + b*df['GKHandling'] + d*df['GKKicking']
+ c*df['Vision'])/(2*a + 4*b + 3*c + 2*d)

plt.figure(figsize=(15,6))
# Generate sequential data and plot
sd = df.sort_values('gk_Shot_Stopper',ascending=False)[:10]
x1 = np.array(list(sd['Name']))
y1 = np.array(list(sd['gk_Shot_Stopper']))
yyy1 = softmax(y1)*100
yy1 = (y1 - np.mean(y1)) + (np.mean(y1) - np.min(y1))
sns.barplot(x1, yy1, palette= "husl")
plt.ylabel("Shot Stopping Score")
```

## 6. This is a plot showing top 11 players making a team.

Task:

Identify the top 11 players that can make a team and show which players suit which position for the team.

+ Markdown

+ Code

```
def draw_pitch(pitch, line):

 line = line
 pitch = pitch

 fig,ax = plt.subplots(figsize=(6.8,10.4))
 plt.ylim(-1,105)
 plt.xlim(-1,69)
 ax.axis('off') # this hides the x and y ticks
 # side and goal lines # 68*104 playing field
 lx1 = [0,0,68,68,0]
 ly1 = [0,104,104,0,0]
 plt.plot(lx1,ly1,color=line,zorder=5)
 # boxes, 6 yard box and goals
 #outer boxes#
 lx2 = [13.84,13.84,54.16,54.16]
 ly2 = [104,87.5,87.5,104]
 plt.plot(lx2,ly2,color=line,zorder=5)
 lx3 = [13.84,13.84,54.16,54.16]
 ly3 = [0,16.5,16.5,0]
 plt.plot(lx3,ly3,color=line,zorder=5)

 #goals#
 lx4 = [30.34,30.34,37.66,37.66]
 ly4 = [104,104.2,104.2,104]
 plt.plot(lx4,ly4,color=line,zorder=5)
 lx5 = [30.34,30.34,37.66,37.66]
 ly5 = [0,-0.2,-0.2,0]
 plt.plot(lx5,ly5,color=line,zorder=5)
 #6 yard boxes#
 lx6 = [24.84,24.84,43.16,43.16]
 ly6 = [104,99.5,99.5,104]
```

```
 ly6 = [104,99.5,99.5,104]
 plt.plot(lx6,ly6,color=line,zorder=5)
 lx7 = [24.84,24.84,43.16,43.16]
 ly7 = [0,4.5,4.5,0]
 plt.plot(lx7,ly7,color=line,zorder=5)
 #Halfway line, penalty spots, and kickoff spot
 lx8 = [0,68]
 ly8 = [52,52]
 plt.plot(lx8,ly8,color=line,zorder=5)
 plt.scatter(34,93,color=line,zorder=5)
 plt.scatter(34,11,color=line,zorder=5)
 plt.scatter(34,52,color=line,zorder=5)
 circle1 = plt.Circle((34,93.5),
9.15,ls='solid',lw=1.5,color=line, fill=False,
zorder=1,alpha=1)
 circle2 = plt.Circle((34,10.5),
9.15,ls='solid',lw=1.5,color=line, fill=False,
zorder=1,alpha=1)
 circle3 = plt.Circle((34,52),
9.15,ls='solid',lw=1.5,color=line, fill=False,
zorder=2,alpha=1)

 ## Rectangles in boxes
 rec1 = plt.Rectangle((20, 87.5), 30,16.5,ls='-',color=pitch, zorder=1,alpha=1)
 rec2 = plt.Rectangle((20, 0), 30,16.5,ls='-',color=pitch, zorder=1,alpha=1)
 ## Pitch rectangle
 rec3 = plt.Rectangle((-1, -1), 70,106,ls='-',color=pitch, zorder=1,alpha=1)

 #Add Player Photos
 arr = mpimg.imread('../input/messi-pic/messi.jpg')
 imagebox = OffsetImage(arr, zoom=0.2)
 ab = AnnotationBbox(imagebox, (60, 80))
 ax.add_artist(ab)
```

```
arr1 = mpimg.imread('../input/ronaldo-pic/ronaldo.jpg')
imagebox = OffsetImage(arr1,0.2)
ab = AnnotationBbox(imagebox, (10, 80))
ax.add_artist(ab)

arr2 = mpimg.imread('../input/roberto-pic/roberto.png')
imagebox = OffsetImage(arr2,0.2)
ab = AnnotationBbox(imagebox, (35, 80))
ax.add_artist(ab)

arr3 = mpimg.imread('../input/antonio/michail-antonio-c9742575-869a-4a2f-a82b-403b8ba910d-resize-750.jpeg')
imagebox = OffsetImage(arr3,0.2)
ab = AnnotationBbox(imagebox, (60, 45))
ax.add_artist(ab)

arr4 = mpimg.imread('../input/picture/Manuel-Lanzini-thumb.gif')
imagebox = OffsetImage(arr4,0.2)
ab = AnnotationBbox(imagebox, (35, 10))
ax.add_artist(ab)

arr5 = mpimg.imread('../input/silva-picturr/silva.jpg')
imagebox = OffsetImage(arr5,0.2)
ab = AnnotationBbox(imagebox, (10, 25))
ax.add_artist(ab)

arr6 = mpimg.imread('../input/ceasra/220px-Cesar_Azpilicueta.jpg')
imagebox = OffsetImage(arr6,0.2)
ab = AnnotationBbox(imagebox, (60, 25))
ax.add_artist(ab)

arr7 = mpimg.imread('../input/picturee/np_file_10609-200x200.jpeg')
imagebox = OffsetImage(arr7,0.2)
ab = AnnotationBbox(imagebox, (25, 50))
```

```
arr6 = mpimg.imread('../input/ceasra/220px-Cesar_Azpilicueta.jpg')
imagebox = OffsetImage(arr6,0.2)
ab = AnnotationBbox(imagebox, (60, 25))
ax.add_artist(ab)

arr7 = mpimg.imread('../input/picturee/np_file_10609-200x200.jpeg')
imagebox = OffsetImage(arr7,0.2)
ab = AnnotationBbox(imagebox, (25, 50))
ax.add_artist(ab)

arr8 = mpimg.imread('../input/picture1/n-golo-kante.jpeg')
imagebox = OffsetImage(arr8,0.2)
ab = AnnotationBbox(imagebox, (35, 25))
ax.add_artist(ab)

arr9 = mpimg.imread('../input/picture2/Ivan-Rakitic.jpg')
imagebox = OffsetImage(arr9,0.2)
ab = AnnotationBbox(imagebox, (45, 50))
ax.add_artist(ab)

arr10 = mpimg.imread('../input/ramos1/download.jpg')
imagebox = OffsetImage(arr10,0.2)
ab = AnnotationBbox(imagebox, (10, 45))
ax.add_artist(ab)

ax.add_artist(rec3)
ax.add_artist(circle1)
ax.add_artist(circle2)
ax.add_artist(rec1)
ax.add_artist(rec2)
ax.add_artist(circle3)

draw_pitch("#195905","#faf0e6")
```

# Appendix - Codes

1) Prediction

```
import pandas as pd import numpy as np
import xgboost as xgb
from sklearn.linear_model import LogisticRegression from sklearn.ensemble import RandomForestClassifier from
IPython.display import display get_ipython().run_line_magic('matplotlib', 'inline')
%matplotlib inline
import matplotlib.pyplot as plt import numpy as np

data = pd.read_csv('../input/football-results-and-betting-odds-data-of-epl/final_dataset.csv') # Remove first 3 match weeks
data = data[data.MW > 3]
data.drop(['Unnamed: 0','HomeTeam', 'AwayTeam', 'Date', 'MW', 'HTFormPtsStr', 'ATFormPtsStr', 'FTHG',
'FTAG','HomeTeamLP',
'AwayTeamLP',
'HTGS', 'ATGS', 'HTGC',
'ATGC','DiffPts','HTFormPts','ATFormPts', 'HM4','HM5','AM4','AM5','HTLossStreak5','ATLossStreak5','HTWinStreak5','ATWinStreak5',
'HTWinStreak3','HTLossStreak3','ATWinStreak3','ATLossStreak3'],1, inplace=True)
display(data.head())

# Total number of matches. n_matches = data.shape[0]
# Calculate number of features. n_features = data.shape[1] - 1
# Calculate matches won by the home team. n_homewins = len(data[data.FTR == 'H'])
# Calculate win rate for home team.
```

```
win_rate = (float(n_homewins) / (n_matches)) * 100 # Print the results

print ("Total number of matches:{}".format(n_matches)) print ("Number of features: {}".format(n_features))
print ("Number of matches won by home team:{}".format(n_homewins)) print ("Win rate of home team:{:.2f}%".format(win_rate))

# Separate into feature set and target variable X_all = data.drop(['FTR'],1)
y_all = data['FTR']
# Standardising the data.
from sklearn.preprocessing import scale cols = [['HTGD','ATGD','HTP','ATP','DiffLP']]
for col in cols:
X_all[col] = scale(X_all[col])


X_all.HM1 = X_all.HM1.astype('str') X_all.HM2 = X_all.HM2.astype('str') X_all.HM3 = X_all.HM3.astype('str')
X_all.AM1 = X_all.AM1.astype('str') X_all.AM2 = X_all.AM2.astype('str')
```

```
X_all.AM3 = X_all.AM3.astype('str') def preprocess_features(X):
output = pd.DataFrame(index = X.index) for col, col_data in X.iteritems():
if col_data.dtype == object:
col_data = pd.get_dummies(col_data,prefix = col) output = output.join(col_data)
return output X_all = preprocess…
from sklearn.model_selection import train_test_split # Shuffle and split the dataset into training and

X_train, X_test, y_train, y_test = train_test_split(X_all, y_all,test_size = 40,random_state = 2,stratify = y_all)
# ## Training and Evaluating Models from sklearn.metrics import f1_score def
train_classifier(clf, X_train, y_train):
"Fits a classifier to the training data." # train the classifier
clf.fit(X_train, y_train)


def predict_labels(clf, features, target):
"Makes predictions using a fit classifier based on F1 score." # make predictions
y_pred = clf.predict(features)
return f1_score(target, y_pred, pos_label='H'),sum(target == y_pred) / float(len(y_pred))

def train_predict(clf, X_train, y_train, X_test,y_test): "Tr…
# Initialize the three models clf_A = LogisticRegression()
clf_B = xgb.XGBClassifier(seed = 82)
clf_C = RandomForestClassifier(criterion='gini',n_estimators=700, min_samples_split=10,
min_samples_leaf=1, max_features='auto', oob_score=True,
```

```
random_state=1)
train_predict(clf_A, X_train, y_train, X_test,y_test) print ('')
train_predict(clf_B, X_train, y_train, X_test,y_test) print ('')
train_predict(clf_C, X_train, y_train, X_test,y_test) print ('')

classif = ['LogisticRegression','XGB','RandomForestClassifier'] f11, acc1 = predict_labels(clf_A, X_train, y_train)
f12, acc2 = predict_labels(clf_B, X_train, y_train) f13, acc3 = predict_labels(clf_C, X_train,
y_train) f1_score = [f11,f12,f13]
accuracy = [acc1,acc2,acc3] xpos = np.arange(len(classif))
plt.bar(xpos-0.2,f1_score, width=0.4, label="F1 score") plt.bar(xpos+0.2,accuracy, width=0.4,label="accuracy")

plt.xticks(xpos,classif)
```

```
plt.ylabel("values") plt.title('Results of the classifiers ') plt.legend()
```

2)

```
import numpy as np #to work with arrays containing player info import seaborn as sns
import pandas as pd
import matplotlib.pyplot as plt
from matplotlib.patches import Arc, Rectangle, ConnectionPatch from matplotlib.offsetbox import
OffsetImage
from mplsoccer.pitch import Pitch import squarify
from functools import reduce
%matplotlib inline import json
from pandas.io.json import json_normalize def draw_pitch(ax):
plt.plot([0,0],[0,80], color="black")
plt.plot([0,120],[80,80], color="black")
plt.plot([120,120],[80,0], color="black")
plt.plot([120,0],[0,0], color="black")
plt.plot([60,60],[0,80], color="black") #Left wing Penalty Area
plt.plot([14.6,14.6],[57.8,22.2],color="black")
plt.plot([0,14.6],[57.8,57.8],color="black")
plt.plot([0,14.6],[22.2,22.2],color="black") #Right wing Penalty Area
plt.plot([120,105.4],[57.8,57.8],color="black")
plt.plot([105.4,105.4],[57.8,22.5],color="black")
plt.plot([120, 105.4],[22.5,22.5],color="black") #Left 6-yard Box
```

```
plt.plot([0,4.9],[48,48],color="black")
plt.plot([4.9,4.9],[48,32],color="black")
plt.plot([0,4.9],[32,32],color="black") #Right 6-yard Box
plt.plot([120,115.1],[48,48],color="black")
plt.plot([115.1,115.1],[48,32],color="black")
plt.plot([120,115.1],[32,32],color="black") #Prepare Circles
centreCircle = plt.Circle((60,40),8.1,color="black",fill=False) centreSpot =
plt.Circle((60,40),0.71,color="black") leftPenSpot = plt.Circle((9.7,40),0.71,color="black")
rightPenSpot = plt.Circle((110.3,40),0.71,color="black") #Draw CirclesP a g e | 25
ax.add_patch(centreCircle) ax.add_patch(centreSpot)
ax.add_patch(leftPenSpot) ax.add_patch(rightPenSpot) #Preparing Arcs
leftArc = Arc((9.7,40),height=16.2,width=16.2,angle=0,theta1=310,theta2=50,color="black") rightArc =
Arc((110.3,40),height=16.2,width=16.2,angle=0,theta1=130,theta2=230,color="black") #Draw Arcs
ax.add_patch(leftArc) ax.add_patch(rightArc)
#using the above function to plot the football field.
```

```
fig=plt.figure() #create a figure object for plotting purpose fig.set_size_inches(7, 5) #specify the width
and height. ax=fig.add_subplot(1,1,1) #to draw a 1x1 plot at position 1. draw_pitch(ax) #pass the ax
object as a paramter in the above function plt.show() #display the plot (football field)
df = pd.read_csv('messibetis.csv') df['x'] = df['x']*1.2
df['y'] = df['y']*0.8
df['endX'] = df['endX']*1.2
df['endY'] = df['endY']*0.8
#using the above function to plot the football field. fig,ax=plt.subplots(figsize = (13.5, 8))
fig.set_facecolor('#228B22') ax.patch.set_facecolor('#228B22')
pitch =Pitch(pitch_type = 'statsbomb', orientation = 'horizontal', pitch_color = '#228B22', line_color = '#c7d5cc', figsize = (16,
11), constrained_layout = True, tight_layout = False)
pitch.draw(ax = ax) plt.gca().invert_yaxis()

#create heatmap kde = sns.kdeplot(
df['x'],
df['y'],
shade = True, shade_lowest = False, alpha = 0.8,
n_levels = 10, cmap = 'magma'
)


for x in range(len(df['x'])):
if df['outcome'][x] == 'Successful':
plt.plot((df['x'][x], df['endX'][x]), (df['y'][x], df['endY'][x]), color='green') plt.scatter(df['x'][x], df['y'][x], color
```

```
= 'green')
if df['outcome'][x] == 'Unsuccessful':
plt.plot((df['x'][x], df['endX'][x]), (df['y'][x], df['endY'][x]), color='red') plt.scatter(df['x'][x], df['y'][x], color =
'red')

plt.xlim(0, 120) plt.ylim(0,80)
plt.title("Lionel Andres Messi heatmap", color = 'white', size = 20) with open('15956.json') as
data_file:
data = json.load(data_file)
df = pd.json_normalize(data, sep = "_")
sergi_pass = df[(df['type_name'] == "Pass") & (df['player_name']=='Sergi Roberto Carnicer')] # get passing information of
Ozil
pass_column = [i for i in df.columns if i.startswith("pass")]
sergi_pass = sergi_pass[["id", "period", "timestamp", "location", "pass_end_location", "pass_recipient_name"]]
sergi_pass.head()
fig, ax = plt.subplots() fig.set_size_inches(7, 5) ax.set_xlim([0,120])
ax.set_ylim([0,80])
for i in range(len(sergi_pass)):
# can also differentiate by color
```

```
color = "blue" if sergi_pass.iloc[i]['period'] == 1 else "red" ax.annotate("", xy =
(sergi_pass.iloc[i]['pass_end_location'][0],
sergi_pass.iloc[i]['pass_end_location'][1]), xycoords = 'data',
xytext = (sergi_pass.iloc[i]['location'][0], sergi_pass.iloc[i]['location'][1]), textcoords
= ‘data’, arrowprops=dict(arrowstyle="->",connectionstyle="arc3", color = "blue"),)

plt.show()

# extract players involvement in the entire game
sergi_action = df[(df['player_name']=='Sergi Roberto Carnicer')][["id", "type_name","period", "timestamp", "location"]]
sergi_action.head() fig,ax=plt.subplots(figsize = (13.5, 8))
fig.set_facecolor('#228B22') ax.patch.set_facecolor('#228B22')
pitch =Pitch(pitch_type = 'statsbomb', orientation = 'horizontal', pitch_color = '#228B22', line_color = '#c7d5cc', figsize = (16,
11), constrained_layout = True, tight_layout = False)
pitch.draw(ax = ax) plt.gca().invert_yaxis()

x_coord = [i[0] for i in sergi_action["location"]] y_coord = [i[1] for i in
sergi_action["location"]]

#shades: give us the heat map we desire
# n_levels: draw more lines, the larger n, the more bluerry it loos
sns.kdeplot(x_coord, y_coord, shade = True, shade_lowest = False, alpha = 0.9, cmap = "magma", n_levels = 10)
plt.xlim(0, 120) plt.ylim(0,80)
plt.title("Sergi Roberto heat map", color = 'white', size = 20)
```

**def heat_pass_map(data, player_name):**

pass_data = data[(data['type_name'] == "Pass") & (data['player_name'] == player_name)] action_data = data[(data['player_name']==player_name)]

fig,ax=plt.subplots(figsize = (13.5, 8)) fig.set_facecolor('#228B22')
ax.patch.set_facecolor('#228B22')
pitch =Pitch(pitch_type = 'statsbomb', orientation = 'horizontal', pitch_color = '#228B22', line_color = '#c7d5cc', figsize = (16, 11), constrained_layout = True, tight_layout = False)
pitch.draw(ax = ax) plt.gca().invert_yaxis()

for i in range(len(pass_data)):
# we also differentiate different half by different color color = "blue" if pass_data.iloc[i]['period'] == 1
else "red"
ax.annotate("", xy = (pass_data.iloc[i]['pass_end_location'][0], pass_data.iloc[i]['pass_end_location'][1]),
xycoords = 'data',
xytext = (pass_data.iloc[i]['location'][0], pass_data.iloc[i]['location'][1]), textcoords
= 'data',
arrowprops=dict(arrowstyle="->",connectionstyle="arc3", color = color),)
x_coord = [i[0] for i in action_data["location"]] y_coord = [i[1] for i in
action_data["location"]]
sns.kdeplot(x_coord, y_coord, shade = True, shade_lowest = False, alpha = 0.9, cmap = "magma", n_levels = 10)

```
plt.ylim(0, 80) # need this, otherwise kde plot will go outside plt.xlim(0, 120)
plt.title( player_name+" heat map", color = 'white', size = 20) plt.show()
heat_pass_map(df, 'Jordi Alba Ramos') heat_pass_map(df, 'Sergi Roberto Carnicer')
heat_pass_map(df, 'Sergio Busquets i Burgos')
```

3)

```
import pandas as pd import seaborn as sns
import matplotlib.pyplot as plt
from matplotlib.offsetbox import TextArea,DrawingArea, OffsetImage, AnnotationBbox import matplotlib.image as
mpimg
import numpy as np
from scipy.special import softmax

get_ipython().run_line_magic('matplotlib', 'inline') df =
pd.read_csv("../input/fifa19/data.csv") df.head(7)

df.replace(' ', None)

df1=df[['Nationality']].groupby(['Nationality'])['Nationality'].count().reset_index(name='count').sort_v
alues(['count'],ascending='false')
df2=df1[df1.shape[0]-10:df1.shape[0]] plt.figure(figsize=(600,600))
fig1, ax1 = plt.subplots() ax1.pie(df2['count'],labels=df2['Nationality'],autopct='%1.2f%%',radius=3)
```

# Top 10 Countries with most playingfootballers#To visualize how many players are playing at a particular ageplt.figure(figsize=(15,6))sns.countplot(x="Age",data=df)# We can see most players play at the age of 25df.drop(['Height', 'Weight'],axis=1)

4)

```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from matplotlib.offsetbox import TextArea,DrawingArea, OffsetImage, AnnotationBbox
import matplotlib.image as mpimg
import numpy as np
from scipy.special import softmax
get_ipython().run_line_magic('matplotlib', 'inline')
df = pd.read_csv("../input/fifa19/data.csv")
df.head(7)
df.replace(' ', None)



a = 1
b = 2
c = 3
d = 4
#GoalKeeping Characteristics
df['gk_Shot_Stopper'] = ( d*df['GKReflexes'] +
```

```
c*df['GKDiving'] + c*df['Jumping'] + b*df['Composure'] +
b*df['Reactions'] + a*df['Strength'] + a*df['SprintSpeed'] +
b*df['GKPositioning']+ b*df['GKHandling'] )/(2*a + 4*b +2*c + 1*d)

df['gk_Sweeper'] = (b*df['Reactions'] + b*df['Composure'] +
b*df['SprintSpeed'] + a*df['ShortPassing'] + a*df['LongPassing'] +
b*df['Jumping'] + b*df['GKPositioning'] + b*df['GKDiving'] +
d*df['GKReflexes'] + b*df['GKHandling'] + d*df['GKKicking']
+ c*df['Vision'])/(2*a + 4*b + 3*c + 2*d)

plt.figure(figsize=(15,6))
# Generate sequential data and plot
sd = df.sort_values('gk_Shot_Stopper',ascending=False)[:10]
x1 = np.array(list(sd['Name']))
y1 = np.array(list(sd['gk_Shot_Stopper']))
yyy1 = softmax(y1)*100
yy1 = (y1 - np.mean(y1)) + (np.mean(y1) - np.min(y1))
sns.barplot(x1, yy1, palette= "husl")
plt.ylabel("Shot Stopping Score")
```

**#Left center back**

```
df['df_centre_backs'] = ( d*df['Reactions'] +
c*df['Interceptions'] + d*df['SlidingTackle'] +
```

```
d*df['StandingTackle'] + b*df[‘Vision’]+ b*df[‘Composure’] +
b*df[‘Crossing’] +a*df[‘ShortPassing’] + b*df.Long_Pass+
c*df.Acceleration + b*df.Speed
+ d*df.Stamina + d*df.Jumping + d*df.Heading +
b*df.Long_Shots + d*df.Marking +
c*df.Aggression)/(6*b + 3*c + 7*d)
df['df_wb_Wing_Backs'] = (b*df.Ball_Control +
a*df.Dribbling + a*df.Marking + d*df.Sliding_Tackle +
d*df.Standing_Tackle + a*df.Attacking_Position +
c*df.Vision + c*df.Crossing + b*df.Short_Pass +
c*df.Long_Pass + d*df.Acceleration +d*df.Speed +
c*df.Stamina + a*df.Finishing)/(4*a + 2*b + 4*c +
4*d)

plt.figure(figsize=(15,6))
sd = df[(df['Club_Position'] =='LCB')].sort_values('df_centre_backs',
ascending=False)[:5]
x2 = np.array(list(sd['Name']))
y2 = np.array(list(sd['df_centre_backs']))
fig1, ax1 = plt.subplots()
ax1.pie(y2,labels=x2, autopct='%1.2f%%')
plt.xlabel("LCB Score")
```

```
plt.figure(figsize=(15,6))
sd = df[(df['Club_Position'] =='RCB')].sort_values('df_centre_backs',ascending=False)[:5]
x2 = np.array(list(sd['Name']))
y2 = np.array(list(sd['df_centre_backs']))
```

**#Defender**

```
df['mf_controller'] = (b*df.Weak_foot +
d*df.Ball_Control + a*df.Dribbling + a*df.Marking +
a*df.Reactions + c*df.Vision + c*df.Composure +
d*df.Short_Pass + d*df.Long_Pass)/(2*c + 3*d + 4*a)

import matplotlib.pyplot as plt
plt.figure(figsize=(15,6))

# Generate some sequential data
ss = df[(df['Club_Position'] == 'LCM') |
(df['Club_Position'] ==
'LM')].sort_values('mf_controller',
ascending=False)[:5]
x1 = np.array(list(ss['Name']))
y1 = np.array(list(ss['mf_controller']))
fig1, ax1 = plt.subplots()
ax1.pie(y1,labels=x1, autopct='%1.2f%%')
```

```
ax1.axis('equal')
plt.xlabel("Controller Score")


#Strikers
df['att_striker'] = (b*df.Weak_foot +
b*df.Ball_Control + a*df.Vision + b*df.Aggression +
b*df.Agility + a*df.Curve + a*df.Long_Shots +
d*df.Balance + d*df.Finishing + d*df.Heading +
c*df.Jumping + c*df.Dribbling)/(3*a + 4*b + 2*c +
3*d)

plt.figure(figsize=(15,6))
ss = df[(df['Club_Position'] == 'ST') |
(df['Club_Position'] == 'LS') | (df['Club_Position']
== 'RS') | (df['Club_Position'] ==
'CF')].sort_values('att_striker',
ascending=False)[:5]
x3 = np.array(list(ss['Name']))
y3 = np.array(list(ss['att_striker']))
sns.barplot(x3, y3,
palette=sns.diverging_palette(255, 133, l=60, n=5,
center="dark"))
plt.ylabel("Striker")
```

```
#football pitch
def draw_pitch(pitch, line):

 line = line
 pitch = pitch

 fig,ax = plt.subplots(figsize=(6.8,10.4))
 plt.ylim(-1,105)
 plt.xlim(-1,69)
 ax.axis('off') # this hides the x and y ticks
 # side and goal lines # 68*104 playing field
 lx1 = [0,0,68,68,0]
 ly1 = [0,104,104,0,0]
 plt.plot(lx1,ly1,color=line,zorder=5)
 # boxes, 6 yard box and goals
 #outer boxes#
 lx2 = [13.84,13.84,54.16,54.16]
 ly2 = [104,87.5,87.5,104]
 plt.plot(lx2,ly2,color=line,zorder=5)
 lx3 = [13.84,13.84,54.16,54.16]
 ly3 = [0,16.5,16.5,0]
 plt.plot(lx3,ly3,color=line,zorder=5)
```

```
#goals#
lx4 = [30.34,30.34,37.66,37.66]
ly4 = [104,104.2,104.2,104]
plt.plot(lx4,ly4,color=line,zorder=5)
lx5 = [30.34,30.34,37.66,37.66]
ly5 = [0,-0.2,-0.2,0]
plt.plot(lx5,ly5,color=line,zorder=5)
#6 yard boxes#
lx6 = [24.84,24.84,43.16,43.16]
ly6 = [104,99.5,99.5,104]
plt.plot(lx6,ly6,color=line,zorder=5)
lx7 = [24.84,24.84,43.16,43.16]
ly7 = [0,4.5,4.5,0]
plt.plot(lx7,ly7,color=line,zorder=5)
#Halfway line, penalty spots, and kickoff spot
lx8 = [0,68]
ly8 = [52,52]
plt.plot(lx8,ly8,color=line,zorder=5)
plt.scatter(34,93,color=line,zorder=5)
plt.scatter(34,11,color=line,zorder=5)
plt.scatter(34,52,color=line,zorder=5)
circle1 = plt.Circle((34,93.5),
9.15,ls='solid',lw=1.5,color=line, fill=False,
```

```
zorder=1,alpha=1)
 circle2 = plt.Circle((34,10.5),
9.15,ls='solid',lw=1.5,color=line, fill=False,
zorder=1,alpha=1)
 circle3 = plt.Circle((34,52),
9.15,ls='solid',lw=1.5,color=line, fill=False,
zorder=2,alpha=1)

 ## Rectangles in boxes
 rec1 = plt.Rectangle((20, 87.5), 30,16.5,ls='-
',color=pitch, zorder=1,alpha=1)
 rec2 = plt.Rectangle((20, 0), 30,16.5,ls='-
',color=pitch, zorder=1,alpha=1)
 ## Pitch rectangle
 rec3 = plt.Rectangle((-1, -1), 70,106,ls='-
',color=pitch, zorder=1,alpha=1)

 #Add Player Photos
 arr = mpimg.imread('messi.jfif')
 imagebox = OffsetImage(arr, zoom=0.2)
 ab = AnnotationBbox(imagebox, (60, 80))
 ax.add_artist(ab)

 arr1 = mpimg.imread('ronaldo.jfif')
```

```
imagebox = OffsetImage(arr1,0.2)
ab = AnnotationBbox(imagebox, (10, 80))
ax.add_artist(ab)

arr2 = mpimg.imread('robert.jfif')
imagebox = OffsetImage(arr2,0.2)
ab = AnnotationBbox(imagebox, (35, 80))
ax.add_artist(ab)

arr3 = mpimg.imread('antonio.jfif')
imagebox = OffsetImage(arr3,0.2)
ab = AnnotationBbox(imagebox, (60, 45))
ax.add_artist(ab)

arr4 = mpimg.imread('manuel.jfif')
imagebox = OffsetImage(arr4,0.2)
ab = AnnotationBbox(imagebox, (35, 10))
ax.add_artist(ab)

arr5 = mpimg.imread('thiago.jfif')
imagebox = OffsetImage(arr5,0.2)
ab = AnnotationBbox(imagebox, (10, 25))
ax.add_artist(ab)
```

```
arr6 = mpimg.imread('azi.jfif')
imagebox = OffsetImage(arr6,0.2)
ab = AnnotationBbox(imagebox, (60, 25))
ax.add_artist(ab)

arr7 = mpimg.imread('iniesta.jfif')
imagebox = OffsetImage(arr7,0.2)
ab = AnnotationBbox(imagebox, (25, 50))
ax.add_artist(ab)

arr8 = mpimg.imread('kante.jfif')
imagebox = OffsetImage(arr8,0.2)
ab = AnnotationBbox(imagebox, (35, 25))
ax.add_artist(ab)

arr9 = mpimg.imread('rakitic.jfif')
imagebox = OffsetImage(arr9,0.2)
ab = AnnotationBbox(imagebox, (45, 50))
ax.add_artist(ab)

arr10 = mpimg.imread('ramos.jfif')
imagebox = OffsetImage(arr10,0.2)
ab = AnnotationBbox(imagebox, (10, 45))
ax.add_artist(ab)
```

```
 ax.add_artist(rec3)
 ax.add_artist(circle1)
 ax.add_artist(circle2)
 ax.add_artist(rec1)
 ax.add_artist(rec2)
 ax.add_artist(circle3)
draw_pitch("#195905","#faf0e6")
```

**#Attackers left wing**

```
df['att_left_wing'] = (c*df.Weak_foot +
c*df.Ball_Control + c*df.Dribbling + c*df.Speed +
d*df.Acceleration + b*df.Vision + c*df.Crossing +
b*df.Short_Pass + b*df.Long_Pass + b*df.Aggression +
b*df.Agility + a*df.Curve + c*df.Long_Shots +
b*df.Freekick_Accuracy + d*df.Finishing)/(a + 6*b +
6*c + 2*d)

plt.figure(figsize=(15,6))
ss = df[(df['Club_Position'] == 'LW') |
(df['Club_Position'] == 'LM') | (df['Club_Position']
== 'LS')].sort_values('att_left_wing',
ascending=False)[:5]
```

```
x1 = np.array(list(ss['Name']))
y1 = np.array(list(ss['att_left_wing']))
sns.barplot(x1, y1,
palette=sns.diverging_palette(255, 133, l=60, n=5,
center="dark"))
plt.ylabel("Left Wing")
```

5) Dashboard:

HTML:

```
<!DOCTYPE html>
<html>
<head>
<title>Dashboard</title>
<link rel="stylesheet" href="stylee.css">

</head>
<body>
```

```
<h1>Our Dashboard</h1>

<div class="main">


<hr style=" height:2px;border-width:0;color:gray;background-color:gray"">

<h2>Graphs and charts</h2>
<!-- <p>Resize the browser window to see the responsive effect.</p> -->

<!-- Portfolio Gallery Grid -->
<div class="row">
<div class="column">
<div class="content">
<img src="1.png" alt="Mountains" style="width:100%" class="imgg">

<p>First Heat Map</p>
</div>
</div>
<div class="column">
<div class="content">
<img src="2.png" alt="Mountains" style="width:100%" class="imgg">

<p>Second Heat Map</p>
```

```
</div>
</div>
<div class="column">
<div class="content">
<img src="3.png" alt="Mountains" style="width:100%" class="imgg">

<p>Chart</p>
</div>
</div>
<div class="column">
<div class="content">
<img src="gg.png" alt="Mountains" style="width:100%" class="imgg">

<p>Bar Graph</p>

</div></div>
<div class="column">
<div class="content">
<img src="GRAPH3.png" alt="Mountains" style="width:100%" class="imgg">

<p>Combination of heat and pass plots into one.</p>
</div>
</div>
<div class="column">
```

```
<div class="content">
<img src="GRAPH4.png" alt="Mountains" style="width:100%" class="imgg">

<p>Combination of heat and pass plots into one.</p>
</div>
</div>
<div class="column">
<div class="content">
<img src="GRAPH6.png" alt="Mountains" style="width:100%" class="img">
```

```
<p>First Bar Graph</p>
</div>
</div>
<div class="column">
<div class="content">
<img src="GRAPH7.png" alt="Mountains" style="width:100%">

<p>Second Bar Graph</p>
</div>
</div>
<div class="column">
<div class="content">
<img src="mm.png" alt="Mountains" style="width:100%" class="imgg">

<p>Third Bar Graph</p>
</div>
</div>
<div class="column">
<div class="content">
<img src="ms.png" alt="Mountains" style="width:100%" class="imgg">

<p>Fourth Bar Graph</p>
</div>
```

```
</div>
<div class="column">
<div class="content">
<img src="sm.png" alt="Mountains" style="width:100%" class="imgg">

<p>Fifth Bar Graph</p>
</div>
</div>

<div class="column">
<div class="content">
<img src="ff.png" alt="Mountains" style="width:100%" class="imgg">

<p>First Pie chart</p>
</div>
</div>
<div class="column">
<div class="content">
<img src="tt.png" alt="Mountains" style="width:100%" class="imggg">

<p>Second Pie chart</p>
</div>
</div>
```

```
<!-- END MAIN-->
</div>
</body>

</html>
```

CSS:-

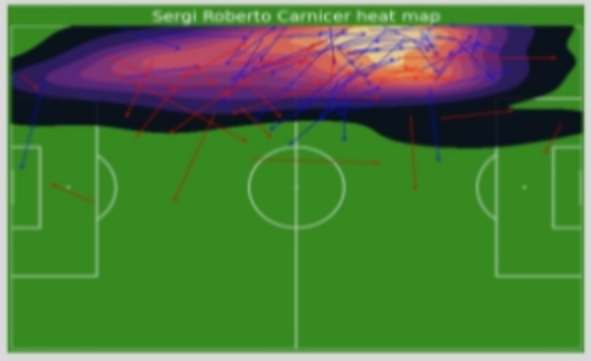
```
h1{
text-align: center;
}
* {
box-sizing: border-box;
}


body {
background-color:rgb(116, 114, 114); padding: 20px;
font-family: Arial; text-align: center;
}

/* Center website */
.main {
max-width: 1000px; margin: auto;
}

h1 {
font-size: 50px;
/* word-break: break-all; */
}
```

```
.row {
margin: 8px -16px;
}

/* Add padding BETWEEN each column (if you want) */
.row,
.row > .column { padding: 8px;
}

/* Create four equal columns that floats next to each other */
.column { float: left; width: 50%;
}

/* Clear floats after rows */
.row:after { content: ""; display: table; clear: both;
}

/* Content */
.content {
background-color:rgba(169, 169, 169, 0.427); padding: 10px;
box-shadow: 5px 10px rgba(0, 0, 0, 0.194);
}
.imgg{
```

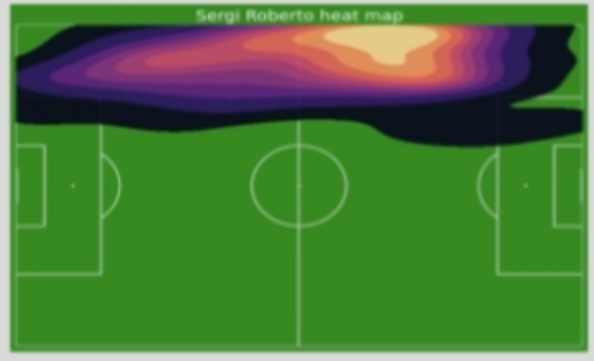
```
height: 300px; padding:20px;

}
p{
text-decoration: crimson;

}
```

# Our Dashboard

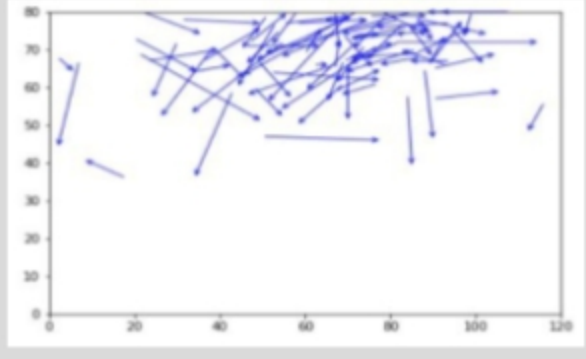
## Graphs and charts

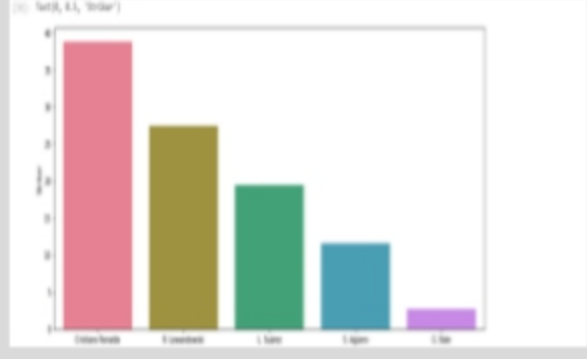


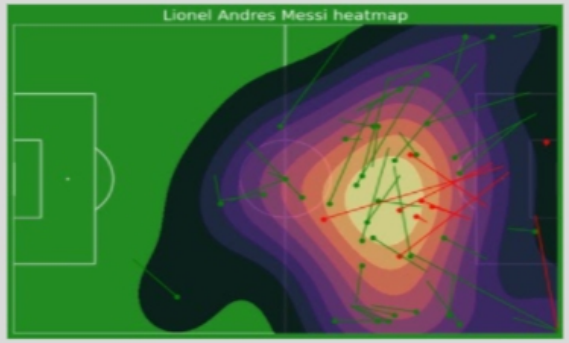

First Heat Map



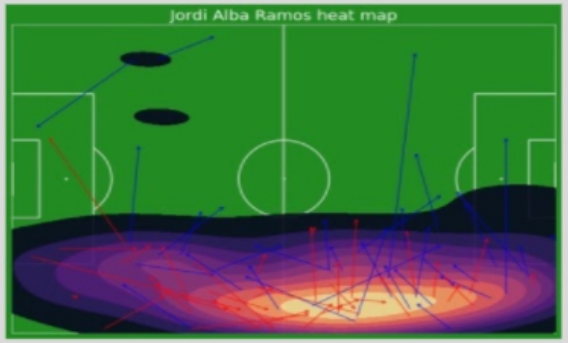

Second Heat Map



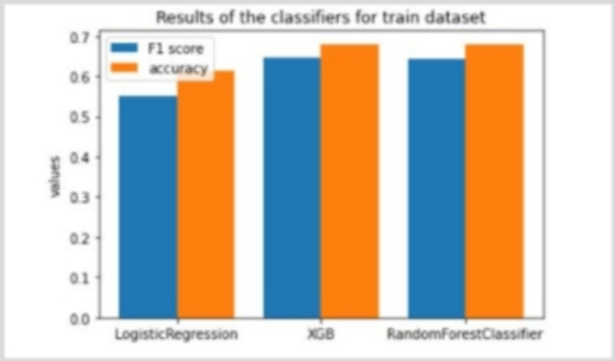





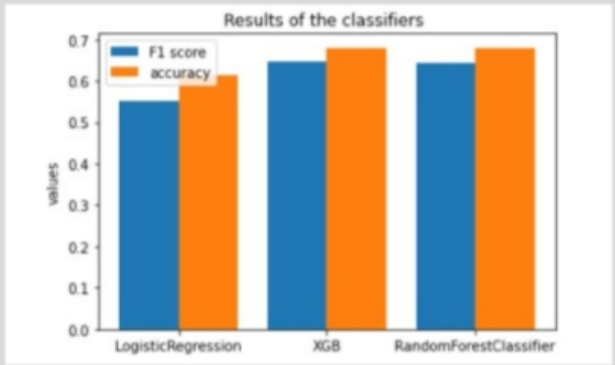

Combination of heat and pass plots into one.



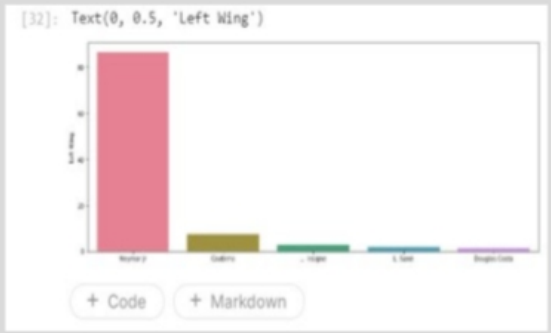

Combination of heat and pass plots into one.



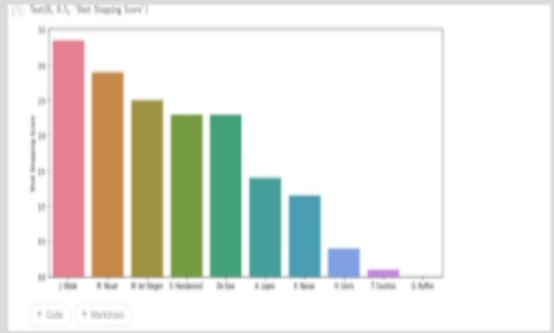

First Bar Graph




Second Bar Graph

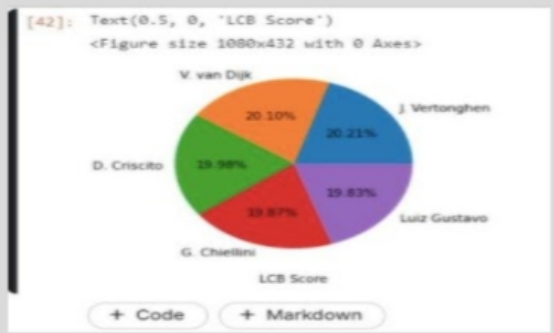



Third Bar Graph

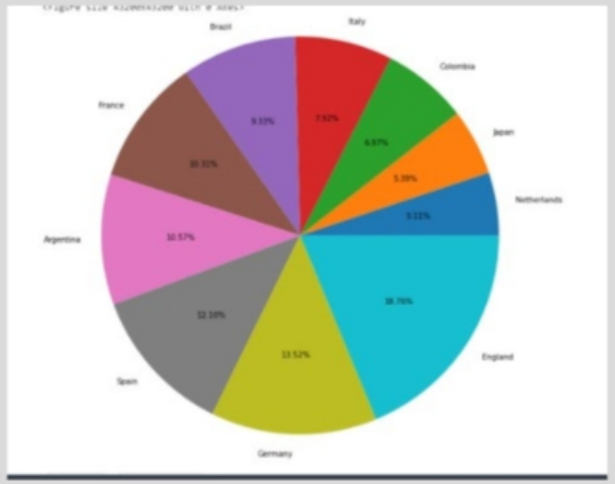


Fourth Bar Graph

Fifth Bar Graph




First Pie chart




Second Pie chart