COLLEGE OF ENGINEERING & TECHNOLOGY

(Techno Campus, PO- Ghatikia, Mahalaxmi Vihar, Bhubaneswar-751029)

MINOR PROJECT REPORT

ON

"MACHINE LEARNING PROJECT ON FIFA 22 ANALYSIS"

Submitted by:

SURAJ KUMAR ROY

Regd. No. : **2005106029**

Under the guidance of

Dr. Jibitesh Mishra

(HOD, DEPT. OF CSA)

DEPARTMENT OF COMPUTER SCIENCE AND APPLICATIONS

(COLLEGE OF ENGINEERING AND TECHNOLOGY, BHUBANESWAR)

2020-2021

❁❁❁❁❁

COLLEGE OF ENGINEERING AND TECHNOLOGY

(Techno Campus, PO-Ghatikia, Mahalaxmi Vihar, Bhubaneswar, 751029)

CERTIFICATE

*This is to certify that the work embodied in the project work entitled "**MACHINE LEARNING PROJECT ON FIFA 22 ANALYSIS**" being submitted by **SURAJ KUMAR ROY** for the partial fulfillment of "Master of Computer Application" degree is a record of bonafide work carried out by him under my supervision. The matter embodied in the project has not been submitted to any other University for the award of any degree or diploma to the best of my knowledge.*

Sign of HOD & Internal Guide **Sign of Student**

DEPARTMENT OF COMPUTER SCIENCE & APPLICATION

(COLLEGE OF ENGINEERING AND TECHNOLOGY, BHUBANESWAR)

❋❋❋❋❋

COLLEGE OF ENGINEERING AND TECHNOLOGY

(Techno Campus, PO-Ghatikia, Mahalaxmi Vihar, Bhubaneswar, 751029)

DECLARATION

I Suraj Kumar Roy bearing Regd. no.: 2005106029 hereby declare that the Project work entitled "MACHINE LEARNING PROJECT ON FIFA 22 ANALYSIS", is a record of an original work done by me under the guidance of Dr. Jibitesh Mishra, HOD of CSA Department. This project work is submitted in the partial fulfillment of the requirements for Masters degree & not submitted for the award of my degree.

Suraj Kumar Roy

Regd. No.: 2005106029

COLLEGE OF ENGINEERING AND TECHNOLOGY

(Techno Campus, PO-Ghatikia, Mahalaxmi Vihar, Bhubaneswar, 751029)

ACKNOWLEDGEMENT

I wish to express my heartful gratitude, indebtedness & sincere thanks to my guide Dr. Jibitesh Mishra (HOD, CSA Dept.) for suggesting me the topic. I am also thankful to him for his able guidance and untired cooperation which encouraged me a lot to finish this study properly.

I owe my sense of gratitude to all other faculty members of Dept. of CSA for their persistent encouragement for the completion of this dissertation.

I would also like to thank my parents and classmates who helped me a lot in finalizing this project within the limited time frame.

Date: ***Suraj Kumar Roy***

Place: CET BBSR ***Regd. No.: 2005106029***

ABSTRACT:

Sports analytics has emerged as a field of research with increasing popularity propelled, in part, by the real-world. Analysis of teams, clubs and players performance data has continued to revolutionize the sports industry on the field, court, and ice as well as in living rooms among fantasy sports audiences and online sports gaming.

The legendary **FIFA** series (currently the largest video game franchise, produced by **EA SPORTS**) brings **The World's Game** to life, letting us play with the biggest leagues, clubs, and players in football world, all with incredible detail and realism. This allows customization such as whether we want to build our dream squad in "FIFA Ultimate Team", lead our favorite club to glory in "Career Mode", take the game back to the streets with "EA SPORTS VOLTA FOOTBALL", or get bragging rights over a friend in "Kick-Off Mode" etc.

Analysis of the game's data will help the gamers as well as the football fans to have a wider perspective about the player's performances. Predicting your dream squad through simple comparison will also have an impact on the real-life game. This project focuses on analyzing the players' data using traditional machine learning algorithms and its steady implementation in real world data using python.

TABLE OF CONTENTS

CHAPTER 1

INTRODUCTION

INTRODUCTION:

Sports analytics has emerged as a field of research with increasing popularity propelled, in part, by the real-world. Analysis of teams, clubs and players performance data has continued to revolutionize the sports industry on the field, court, and ice as well as in living rooms among fantasy sports audiences and online sports gaming.

EA SPORTS FIFA is an association football simulation video game developed by EA Mobile and EA Canada and published by EA Sports for iOS and Android. It was released on 11 October 2016, for iOS and Android. Microsoft Windows was also included until 2017. It was announced on August 16, 2016 during Gamescom 2016.

The legendary **FIFA** series (currently the largest video game franchise, produced by **EA SPORTS**) brings **The World's Game** to life, letting us play with the biggest leagues, clubs, and players in football world, all with incredible detail and realism. This allows customization such as whether we want to build our dream squad in "FIFA Ultimate Team", lead our favorite club to glory in "Career Mode", take the game back to the streets with "EA SPORTS VOLTA FOOTBALL", or get bragging rights over a friend in "Kick-Off Mode" etc.

Analysis of the game's data will help the gamers as well as the football fans to have a wider perspective about the player's performances. Predicting your dream squad through simple comparison will also have an impact on the real-life game. This project focuses on analyzing the players' data using traditional machine learning algorithms and its steady implementation in real world data using python.

CHAPTER 2

SOFTWARE & HARDWARE REQUIREMENTS

SOFTWARE & HARDWARE REQUIREMENTS :

2.1. SOFTWARE REQUIREMENTS

2.1.1. Anaconda

2.1.2. Jupyter Book

2.1.3. Python

2.1.4. Python Libraries

2.1.4.1. Pandas

2.1.4.2. Seaborn

2.1.4.3. NumPy

2.1.4.4. Matplotlib

2.1.4.5. Plotly

2.2. HARDWARE REQUIREMENTS

2.2.1. Windows 10- 64 Bit

2.2.2. Intel Processor i5 10^{th} Generation

2.2.3. RAM 8 GB

CHAPTER 3

SOFTWARE REQUIREMENT ANALYSIS

SOFTWARE REQUIREMENT ANALYSIS:

4.1 PROBLEM STATEMENT

Rainfall prediction remains a serious concern and has attracted the attention of governments, industries, risk management entities, as well as the scientific community. Rainfall is a climatic factor that affects many human activities like agricultural production, construction, power generation, forestry and tourism, among others. As we know heavy and irregular rainfall can have many impacts like destruction of crops and farms, damage of property so a better forecasting model is essential for an early warning that can minimize risks to life and property and also managing the agricultural farms in better way. Therefore, having an appropriate approach for rainfall prediction is also required to counter these problems. This project is made focussing these problems.

4.2 PROPOSED METHOD FOR SOLUTION

The predictive model is used to prediction of the precipitation. The first step is converting data in to the correct format to conduct experiments then make a good analysis of data and observe variation in the patterns of rainfall. We predict the rainfall by separating the dataset into training set and testing set then we apply different machine learning approaches (MLR, SVR, etc.) and statistical techniques and compare and draw analysis over various approaches used. With the help of numerous approaches we attempt to minimize the error.

DATASET DESCRIPTION:

The dataset [10] consists of the measurement of rainfall from year 1901-2015 for each state.

- Data consists of 19 attributes (individual months, annual, and combinations of 3 consecutive months) for 36 sub divisions.
- The data is available only from 1950 to 2015 for some of the subdivisions
- The attributes are the amount of rainfall measured in mm

As the dataset is very large, feature reduction is done so that it improves the accuracy, reduces the computation time and also storage. Principal Component Analysis (PCA) is a technique of extracting necessary variables from a huge set of variables. It extracts low dimensional set with a motive to capture the maximum amount of information. With few variables,

visualization becomes more significant. It is done by using covariance matrix and by obtaining Eigen values from it. In our dataset by using PCA it has reduced the attributes by considering only the rainfall data of combination of three consecutive months and annual data from every subdivision.

Techniques used: Multiple Linear Regression: Multiple regression tries to model the connection between two or additional variables and a response by fitting an equation to determined information. Clearly, it's nothing however an extension of straight forward regression toward the mean. The general form of multivariable linear regression model is: $y=\alpha+\beta 1x1+ \beta 2x2+\ldots+ \beta kxk+\varepsilon$ where y = dependent variable and x1, x2… xk are independent variables,α,β are coefficients. Multiple regression will model additional complicated relationship that comes from numerous options along they should to be employed in cases wherever one explicit variable isn't evident enough to map the link between the independent and also the variable quantity.

SUPPORT VECTOR REGRESSION:

Support Vector regression machine learning and data science with the term SVM or support vector machine but SVR that is support vector regression is a bit different from SVM that is support vector machine as the name suggests that is integration algorithm so we can use SVR for working with continuous value instead of classification which is SVM Support Vector Machines support linear and nonlinear regression that we can to as Support Vector Regression. Instead of trying to fit the largest possible street between two classes while limiting margin violations, Support Vector Regression tries to fit as many instances as possible on the street while limiting margin violations. The size of the lane is measured by a hyper parameter Epsilon.

KERNEL- THE FUNCTION USED TO MAP A LOW DIMENSIONAL DATA INTO HIGHER DIMENSIONAL DATA.

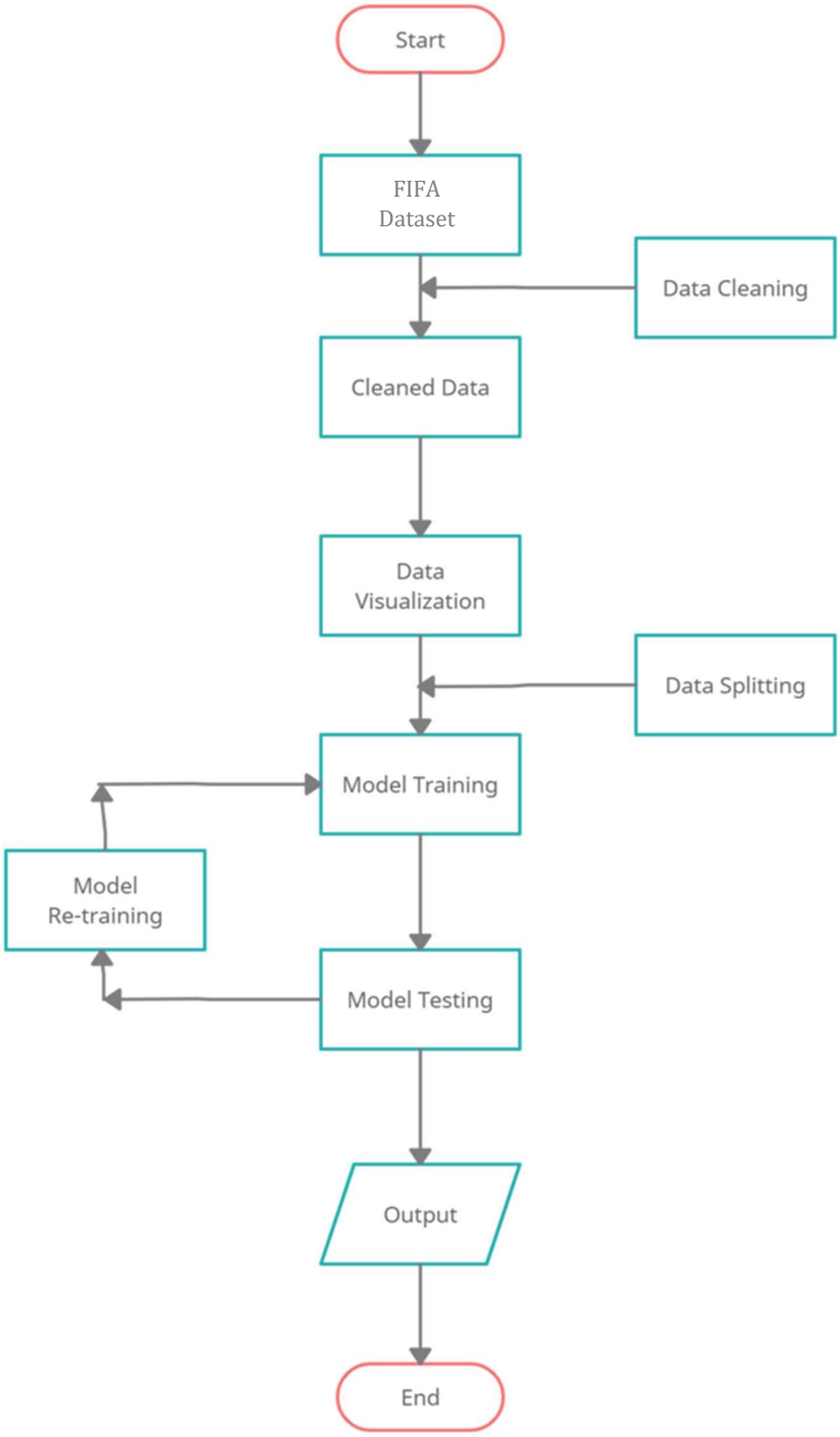
Hyper plane- in SVM this is a basically The Separation line between the data classes also in SVR we are going to define it is as the line that will that will help us to predict the continuous value or target value. Boundary line - the SVM plane which creates imagine the support vector can be on boundary lines or outside the boundary line separates two classes in the concept same. Vectors-these are the data points which are closest to the boundary the distance of the point is minimum. SVR performs linear regression in higher dimensional space. We can think of SVR as if each data point in the training represents its own dimension. When we evaluate kernel between a test point and a point in the training set the resulting value gives

you the coordinate of your test point in that dimension. The vector we get when we evaluate the test point for all points in the training set, k is the representation of the test point in the higher dimensional space. The equation of the hyper plane is wx+b=0 and the two equations of boundary lines is Wx+b=+e, Wx+b=-e Equation that satisfy our SVR is e<=y-Wx-b<=+e SVR has a different regression goal compared to linear regression in linear regression, we are trying to minimize the error between the prediction and data whereas in SVR a goal is to make sure that error do not exceed the threshold.

CHAPTER 4

SOFTWARE DESIGN

DATA-FLOW DIAGRAM:



CHAPTER 5

CODE TEMPLATES AND RESULTS

CODE TEMPLATES AND RESULTS:

6.1 DATASET

- **data.csv:**

Content

- 17,000+ players
- 50+ attributes per player ranging from ball skills aggression etc.
- Player's attributes sourced from EA Sports' FIFA video game series, including the weekly updates
- Players from all around the globe
- URLs to their homepage
- Club logos
- Player images male and female
- National and club team data

Weekly Updates would include :

- Real life data (Match events etc.)
- The fifa generated player dataset
- Betting odds
- Growth

- **fulldata.csv**

Data description

- The file *FullData.csv* contains attributes describing the in game play style and also some of the real statistics such as Nationality etc.
- The file *PlayerNames.csv* contains URLs for different players from their profiles on fifaindex.com. Append the URLs after the base url fifaindex.com.
- The compressed file Pictures.zip contains pictures for top 1000 players in Fifa 22.
- The compressed file Pictures_f.zip contains pictures for top 139 female players in Fifa 22.
- The compressed file ClubPictures.zip contains pictures for emblems of some major clubs in Fifa 22.

Exploring the data

For starters you can become a scout:

- Create attribute dependent or overall best teams
- Create the fastest/slowest teams
- See which areas of the world provide which attributes (like Africa : Stamina, Pace)
- See which players are the best at each position
- See which outfield players can play a better role at some other position
- See which youngsters have attributes which can be developed

```
#Lets start by importing libraries

# Data Analysis
import numpy as np
import pandas as pd

# Data visualisations
import matplotlib.pyplot as plt
import seaborn as sns
import plotly.graph_objs as go
import plotly.express as px
%matplotlib inline

import warnings
warnings.simplefilter (action = 'ignore', category = Warning
)

#Reading the dataset
data = pd.read_csv('data.csv')
print(data.shape)

data.head(10)
```

	Unnamed: 0	ID	Name	Age	Photo	Nationality	Flag	Overall	Potential	Club	...	Composure	Marking	StandingTackle	SlidingTackle	GKDiving	GKHandling	GKKicking	GKPositioning	GKReflexes	Release Clause
0	0	158023	L. Messi	31	https://cdn.sofifa.org/players/4/19/158023.png	Argentina	https://cdn.sofifa.org/flags/52.png	94	94	FC Barcelona	...	96.0	33.0	28.0	26.0	6.0	11.0	15.0	14.0	8.0	€226.5M
1	1	20801	Cristiano Ronaldo	33	https://cdn.sofifa.org/players/4/19/20801.png	Portugal	https://cdn.sofifa.org/flags/38.png	94	94	Juventus	...	95.0	28.0	31.0	23.0	7.0	11.0	15.0	14.0	11.0	€127.1M
2	2	190871	Neymar Jr	26	https://cdn.sofifa.org/players/4/19/190871.png	Brazil	https://cdn.sofifa.org/flags/54.png	92	93	Paris Saint-Germain	...	94.0	27.0	24.0	33.0	9.0	9.0	15.0	15.0	11.0	€228.1M
3	3	193080	De Gea	27	https://cdn.sofifa.org/players/4/19/193080.png	Spain	https://cdn.sofifa.org/flags/45.png	91	93	Manchester United	...	68.0	15.0	21.0	13.0	90.0	85.0	87.0	88.0	94.0	€138.6M
4	4	192985	K. De Bruyne	27	https://cdn.sofifa.org/players/4/19/192985.png	Belgium	https://cdn.sofifa.org/flags/7.png	91	92	Manchester City	...	88.0	68.0	58.0	51.0	15.0	13.0	5.0	10.0	13.0	€196.4M
5	5	183277	E. Hazard	27	https://cdn.sofifa.org/players/4/19/183277.png	Belgium	https://cdn.sofifa.org/flags/7.png	91	91	Chelsea	...	91.0	34.0	27.0	22.0	11.0	12.0	6.0	8.0	8.0	€172.1M
6	6	177003	L. Modrić	32	https://cdn.sofifa.org/players/4/19/177003.png	Croatia	https://cdn.sofifa.org/flags/10.png	91	91	Real Madrid	...	84.0	60.0	76.0	73.0	13.0	9.0	7.0	14.0	9.0	€137.4M
7	7	176580	L. Suárez	31	https://cdn.sofifa.org/players/4/19/176580.png	Uruguay	https://cdn.sofifa.org/flags/60.png	91	91	FC Barcelona	...	85.0	62.0	45.0	38.0	27.0	25.0	31.0	33.0	37.0	€164M
8	8	155862	Sergio Ramos	32	https://cdn.sofifa.org/players/4/19/155862.png	Spain	https://cdn.sofifa.org/flags/45.png	91	91	Real Madrid	...	82.0	87.0	92.0	91.0	11.0	8.0	9.0	7.0	11.0	€104.6M
9	9	200389	J. Oblak	25	https://cdn.sofifa.org/players/4/19/200389.png	Slovenia	https://cdn.sofifa.org/flags/44.png	90	93	Atlético Madrid	...	70.0	27.0	12.0	18.0	86.0	92.0	78.0	88.0	89.0	€144.5M

```
#Describing the data
data.describe()
```

	Unnamed: 0	ID	Age	Overall	Potential	Special	International Reputation	Weak Foot	Skill Moves	Jersey Number	...	Penalties	Composure	Marking	StandingTackle	SlidingTackle	GKDiving	GKHandling	GKKicking	GKPositioning	GKF
count	18207.000000	18207.000000	18207.000000	18207.000000	18207.000000	18207.000000	18159.000000	18159.000000	18159.000000	18147.000000	...	18159.000000	18159.000000	18159.000000	18159.000000	18159.000000	18159.000000	18159.000000	18159.000000	18159.000000	18159
mean	9103.000000	214298.338606	25.122206	66.238699	71.307299	1597.809908	1.113222	2.947299	2.361308	19.546096	...	48.548598	58.648274	47.281623	47.697836	45.661435	16.616223	16.391596	16.232061	16.388898	16
std	5256.052511	29965.244204	4.669943	6.908930	6.136496	272.586016	0.394031	0.660456	0.756164	15.947765	...	15.704053	11.436133	19.904397	21.664004	21.289135	17.695349	16.906900	16.502864	17.034669	17
min	0.000000	16.000000	16.000000	46.000000	48.000000	731.000000	1.000000	1.000000	1.000000	1.000000	...	5.000000	3.000000	3.000000	2.000000	3.000000	1.000000	1.000000	1.000000	1.000000	1
25%	4551.500000	200315.500000	21.000000	62.000000	67.000000	1457.000000	1.000000	3.000000	2.000000	8.000000	...	39.000000	51.000000	30.000000	27.000000	24.000000	8.000000	8.000000	8.000000	8.000000	8
50%	9103.000000	221759.000000	25.000000	66.000000	71.000000	1635.000000	1.000000	3.000000	2.000000	17.000000	...	49.000000	60.000000	53.000000	55.000000	52.000000	11.000000	11.000000	11.000000	11.000000	11
75%	13654.500000	236529.500000	28.000000	71.000000	75.000000	1787.000000	1.000000	3.000000	3.000000	26.000000	...	60.000000	67.000000	64.000000	66.000000	64.000000	14.000000	14.000000	14.000000	14.000000	14
max	18206.000000	246620.000000	45.000000	94.000000	95.000000	2346.000000	5.000000	5.000000	5.000000	99.000000	...	92.000000	96.000000	94.000000	93.000000	91.000000	90.000000	92.000000	91.000000	90.000000	94

```
#Analysing the Club, Let's analyse my favourite club "Real M
adrid"

def club(x):
    return data[data['Club'] == x][['Name','Jersey Number','
    Position','Overall','Nationality','Age','Wage','Value','
    Contract Valid Until']]

club('Real Madrid')
```

	Name	Jersey Number	Position	Overall	Nationality	Age	Wage	Value	Contract Valid Until
6	L. Modrić	10.0	RCM	91	Croatia	32	€420K	€67M	2020
8	Sergio Ramos	15.0	RCB	91	Spain	32	€380K	€51M	2020
11	T. Kroos	8.0	LCM	90	Germany	28	€355K	€76.5M	2022
19	T. Courtois	1.0	GK	89	Belgium	26	€240K	€53.5M	2024
27	Casemiro	14.0	CDM	88	Brazil	26	€285K	€59.5M	2021
30	Isco	22.0	LW	88	Spain	26	€315K	€73.5M	2022
35	Marcelo	12.0	LB	88	Brazil	30	€285K	€43M	2022
36	G. Bale	11.0	ST	88	Wales	28	€355K	€60M	2022
46	K. Navas	1.0	GK	87	Costa Rica	31	€195K	€30.5M	2020
62	R. Varane	4.0	RCB	86	France	25	€210K	€50M	2022
79	Marco Asensio	10.0	RW	85	Spain	22	€215K	€54M	2023
105	K. Benzema	9.0	ST	85	France	30	€240K	€37M	2021
123	Carvajal	2.0	RB	84	Spain	26	€185K	€31.5M	2022
172	Lucas Vázquez	17.0	RW	83	Spain	27	€205K	€27M	2021
188	Nacho Fernández	12.0	CB	83	Spain	28	€180K	€24.5M	2021
328	Dani Ceballos	21.0	LCM	81	Spain	21	€120K	€25M	2023
417	Odriozola	19.0	RB	80	Spain	22	€115K	€18.5M	2024
430	Mariano	7.0	ST	80	Dominican Republic	24	€140K	€20M	2023
573	Marcos Llorente	18.0	CDM	79	Spain	23	€110K	€16M	2021
697	Kiko Casilla	13.0	GK	79	Spain	31	€105K	€7.5M	2020
754	Vallejo	3.0	CB	78	Spain	21	€81K	€13.5M	2021
1143	Vinícius Júnior	28.0	LW	77	Brazil	17	€66K	€17.5M	2025
2513	F. Valverde	5.0	CM	74	Uruguay	19	€46K	€8.5M	2021
6724	Reguilón	23.0	LB	68	Spain	21	€28K	€1.4M	2020
8732	Javi Sánchez	31.0	CB	67	Spain	21	€24K	€1.2M	2019
9141	Cristo González	27.0	ST	66	Spain	20	€26K	€1.2M	2019
10178	F. Feuillassier	34.0	LW	65	Argentina	20	€23K	€1.2M	2019
10269	Fidalgo	36.0	CM	65	Spain	21	€20K	€875K	2019
11163	Sergio López	32.0	RB	64	Spain	19	€9K	€875K	2019
11327	Fran García	37.0	LB	64	Spain	18	€9K	€825K	2019
11877	Manu Hernando	33.0	CB	64	Spain	19	€9K	€700K	2021
12504	Dani Gómez	38.0	ST	63	Spain	19	€12K	€800K	2022
13687	L. Zidane	30.0	GK	62	France	20	€9K	€350K	2019

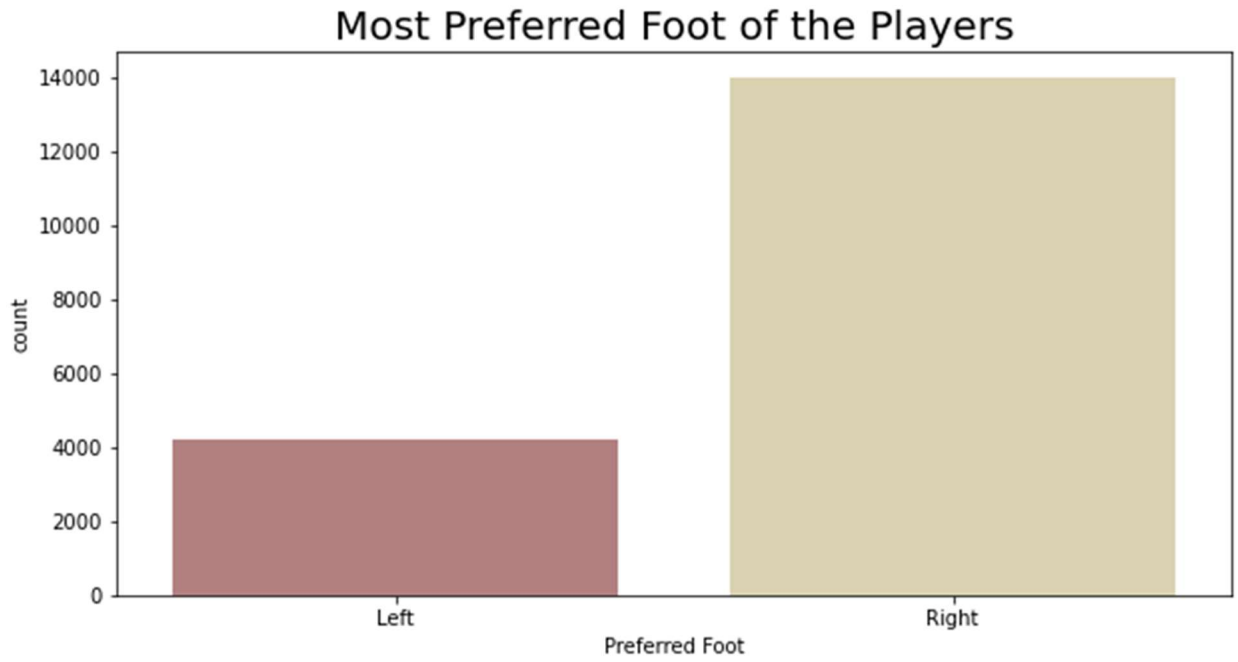
```
# Filling the missing value for the continuous variables for proper data visualiz
ation

data['ShortPassing'].fillna(data['ShortPassing'].mean(), inplace = True)
data['Volleys'].fillna(data['Volleys'].mean(), inplace = True)
data['Dribbling'].fillna(data['Dribbling'].mean(), inplace = True)
data['Curve'].fillna(data['Curve'].mean(), inplace = True)
data['FKAccuracy'].fillna(data['FKAccuracy'], inplace = True)
data['LongPassing'].fillna(data['LongPassing'].mean(), inplace = True)
data['BallControl'].fillna(data['BallControl'].mean(), inplace = True)
data['HeadingAccuracy'].fillna(data['HeadingAccuracy'].mean(), inplace = True)
data['Finishing'].fillna(data['Finishing'].mean(), inplace = True)
data['Crossing'].fillna(data['Crossing'].mean(), inplace = True)
data['Weight'].fillna('200lbs', inplace = True)
data['Contract Valid Until'].fillna(2019, inplace = True)
data['Height'].fillna("5'11", inplace = True)
data['Loaned From'].fillna('None', inplace = True)
data['Joined'].fillna('Jul 1, 2018', inplace = True)
data['Jersey Number'].fillna(8, inplace = True)
data['Body Type'].fillna('Normal', inplace = True)
data['Position'].fillna('ST', inplace = True)
data['Club'].fillna('No Club', inplace = True)
data['Work Rate'].fillna('Medium/ Medium', inplace = True)
data['Skill Moves'].fillna(data['Skill Moves'].median(), inplace = True)
data['Weak Foot'].fillna(3, inplace = True)
data['Preferred Foot'].fillna('Right', inplace = True)
data['International Reputation'].fillna(1, inplace = True)
data['Wage'].fillna('€200K', inplace = True)
data.fillna(0, inplace = True)

def defending(data):
    return int(round((data[['Marking', 'StandingTackle',
                            'SlidingTackle']].mean()).mean()))

def general(data):
    return int(round((data[['HeadingAccuracy', 'Dribbling', 'Curve',
                            'BallControl']].mean()).mean()))

def mental(data):
    return int(round((data[['Aggression', 'Interceptions', 'Positioning',
                            'Vision','Composure']].mean()).mean()))

def passing(data):
    return int(round((data[['Crossing', 'ShortPassing',
```

```
                                'LongPassing']].mean()).mean()))

def mobility(data):
    return int(round((data[['Acceleration', 'SprintSpeed',
                                'Agility','Reactions']].mean()).mean()))
def power(data):
    return int(round((data[['Balance', 'Jumping', 'Stamina',
                                'Strength']].mean()).mean()))

def rating(data):
    return int(round((data[['Potential', 'Overall']].mean()).mean()))

def shooting(data):
    return int(round((data[['Finishing', 'Volleys', 'FKAccuracy',
                                'ShotPower','LongShots', 'Penalties']].mean()).mea
n()))

#Renaming the columns

data.rename(columns={'Club Logo':'Club_Logo'}, inplace=True)
data['Defending'] = data.apply(defending, axis = 1)
data['General'] = data.apply(general, axis = 1)
data['Mental'] = data.apply(mental, axis = 1)
data['Passing'] = data.apply(passing, axis = 1)
data['Mobility'] = data.apply(mobility, axis = 1)
data['Power'] = data.apply(power, axis = 1)
data['Rating'] = data.apply(rating, axis = 1)
data['Shooting'] = data.apply(shooting, axis = 1)

players = data[['Name','Nationality', 'Club', 'Age', 'Rating', 'General',
                'Passing', 'Power', 'Shooting', 'Defending']]

players.head(11)
```

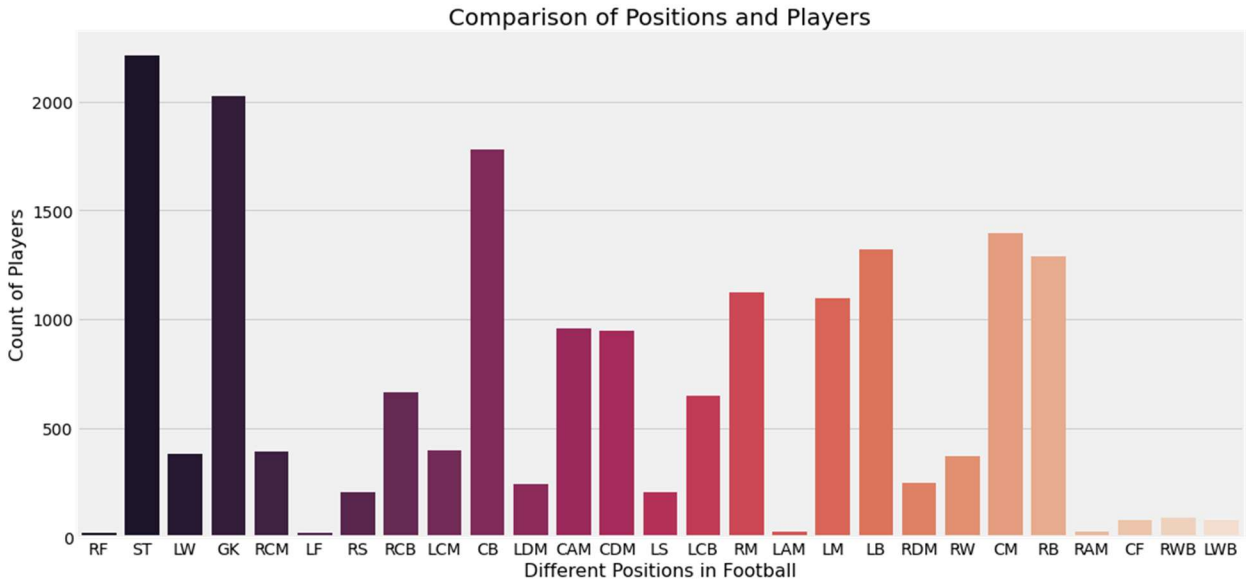
	Name	Nationality	Club	Age	Rating	General	Passing	Power	Shooting	Defending
0	L. Messi	Argentina	FC Barcelona	31	94	89	87	74	88	29
1	Cristiano Ronaldo	Portugal	Juventus	33	94	88	81	83	88	27
2	Neymar Jr	Brazil	Paris Saint-Germain	26	92	85	80	69	84	28
3	De Gea	Spain	Manchester United	27	92	26	39	54	21	16
4	K. De Bruyne	Belgium	Manchester City	27	92	79	92	76	85	59
5	E. Hazard	Belgium	Chelsea	27	91	83	84	75	82	28
6	L. Modrić	Croatia	Real Madrid	32	91	81	89	77	78	70
7	L. Suárez	Uruguay	FC Barcelona	31	91	85	74	81	87	48
8	Sergio Ramos	Spain	Real Madrid	32	91	78	74	82	68	90
9	J. Oblak	Slovenia	Atlético Madrid	25	92	14	23	61	14	19
10	R. Lewandowski	Poland	FC Bayern München	29	90	84	70	81	88	32

- **EXPLORATORY DATA ANALYSIS**

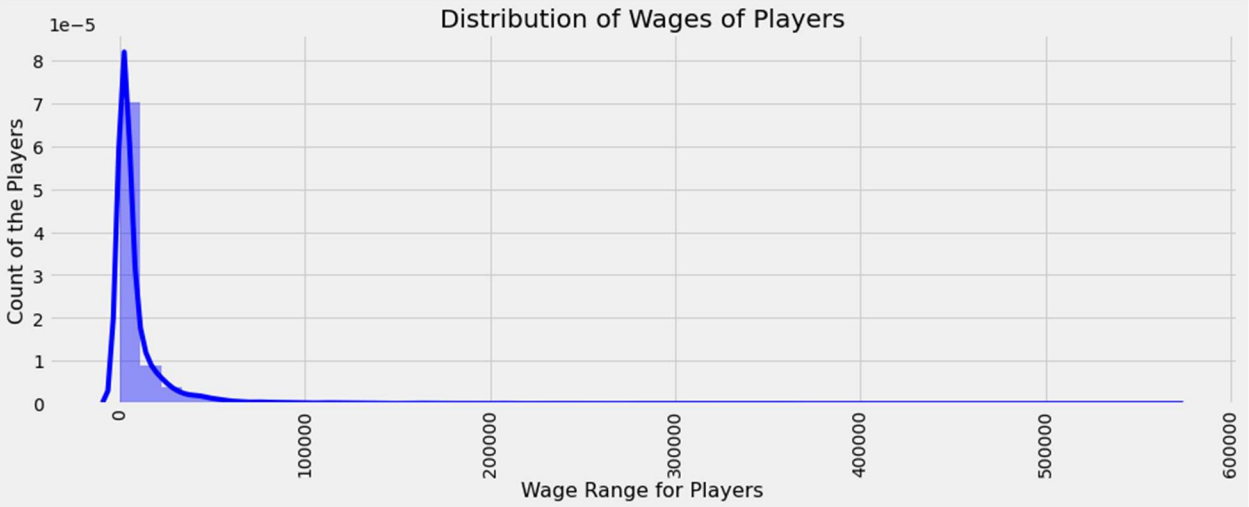
```
#Eliminating certain columns from analysis

data.drop(['ID', 'Photo', 'Flag', 'Real Face', 'Jersey Number',
           'Loaned From'], axis = 1, inplace = True)

#Comparison of preferred foot over the different players

plt.rcParams['figure.figsize'] = (10, 5)
sns.countplot(data['Preferred Foot'], palette = 'pink')
plt.title('Most Preferred Foot of the Players', fontsize = 20)
plt.show()
```



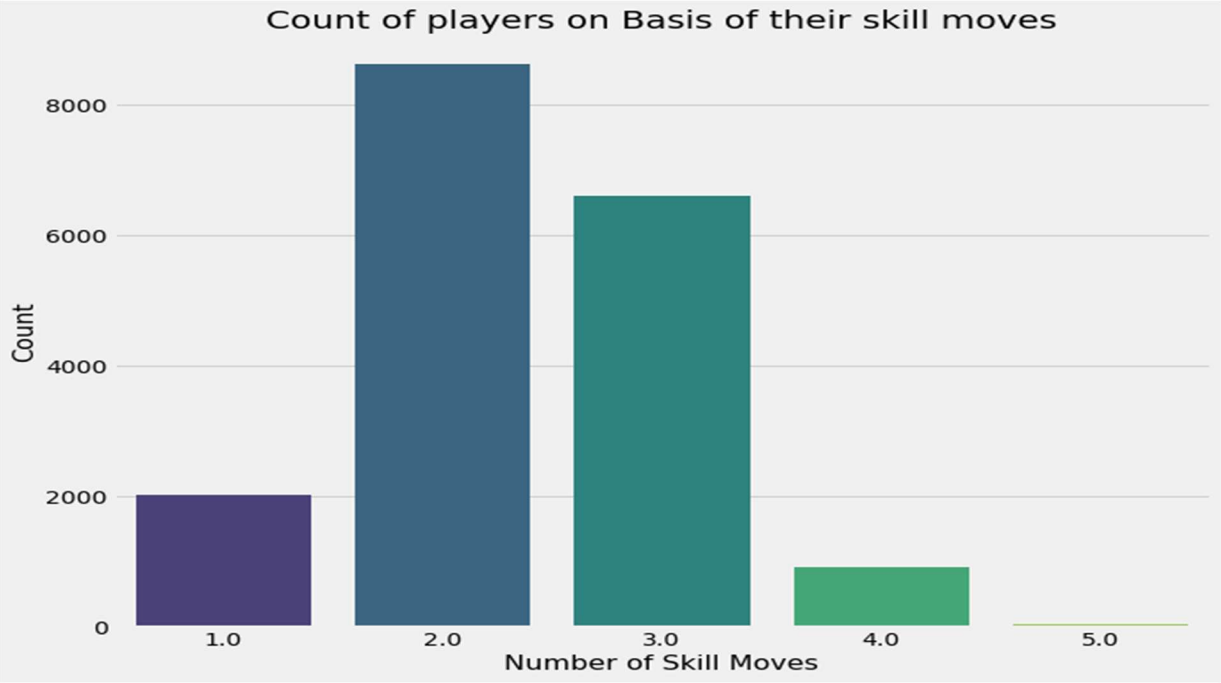

```
#Different positions acquired by the players

plt.figure(figsize = (18, 8))
plt.style.use('fivethirtyeight')
ax = sns.countplot('Position', data = data, palette = 'rocket')
ax.set_xlabel(xlabel = 'Different Positions in Football', fontsize = 16)
ax.set_ylabel(ylabel = 'Count of Players', fontsize = 16)
ax.set_title(label = 'Comparison of Positions and Players', fontsize = 20)
plt.show()
```



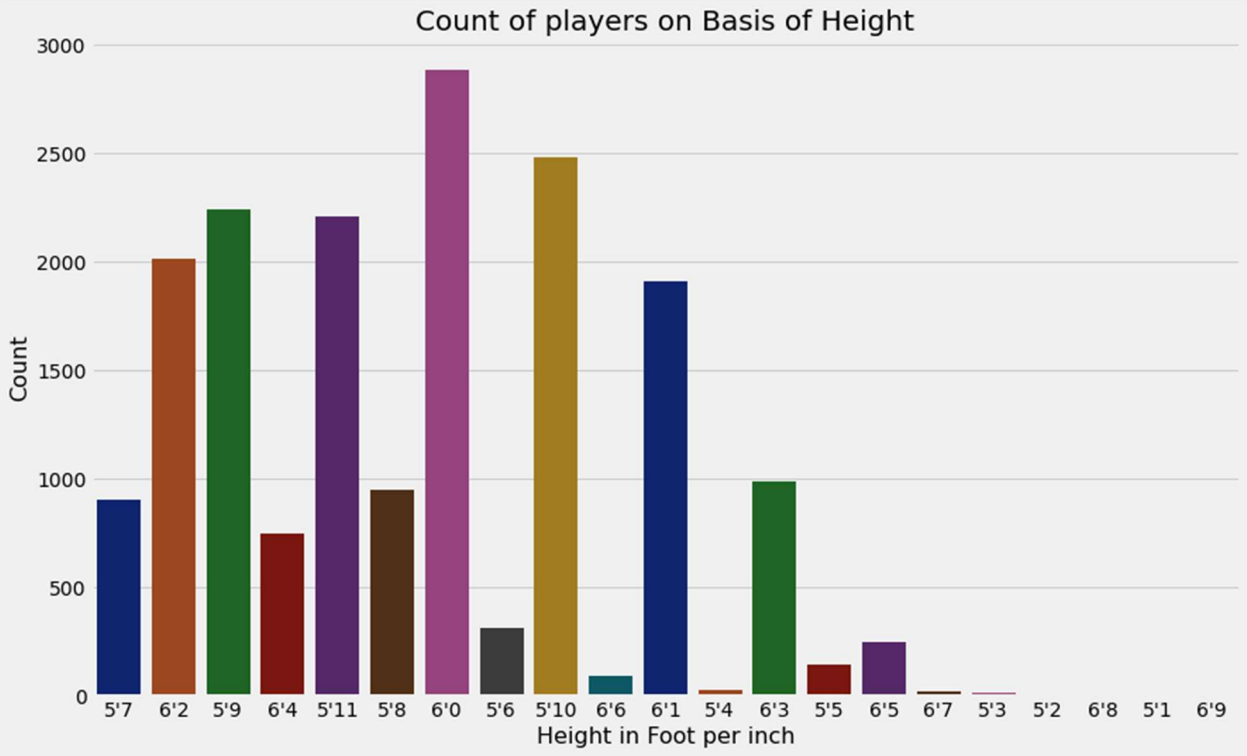

```
#Defining a function for cleaning the Weight data

def extract_value_from(value):
  out = value.replace('lbs', '')
  return float(out)

# applying the function to weight column
#data['value'] = data['value'].apply(lambda x: extract_value_from(x))
data['Weight'] = data['Weight'].apply(lambda x : extract_value_from(x))

data['Weight'].head()
OUTPUT: 0 159.0 1 183.0 2 150.0 3 168.0 4 154.0 Name: Weight, dtype: float64

#Comparing the players' Wages
import warnings
warnings.filterwarnings('ignore')
plt.rcParams['figure.figsize'] = (15, 5)
sns.distplot(data['Wage'], color = 'blue')
plt.xlabel('Wage Range for Players', fontsize = 16)
plt.ylabel('Count of the Players', fontsize = 16)
plt.title('Distribution of Wages of Players', fontsize = 20)
plt.xticks(rotation = 90)
plt.show()
```



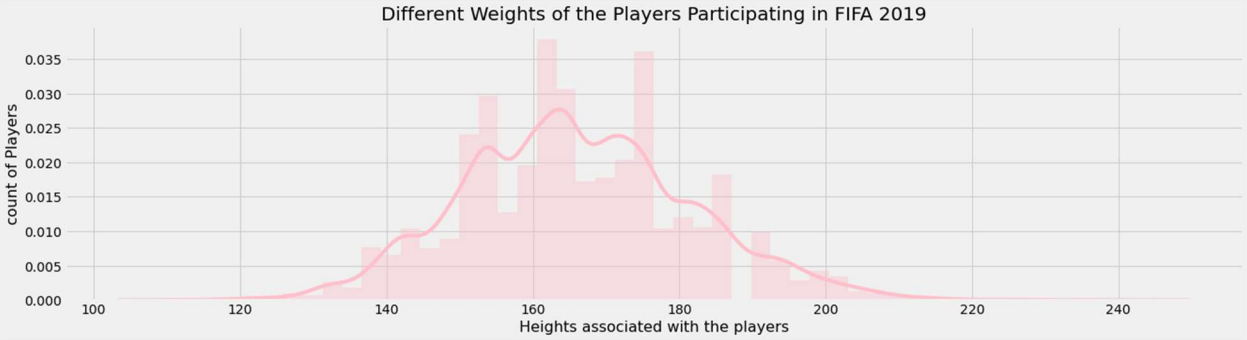

```
#Skill Moves of Players
plt.figure(figsize = (10, 8))
ax = sns.countplot(x = 'Skill Moves', data = data, palette = 'viridis')
ax.set_title(label = 'Count of players on Basis of their skill moves', fontsize =
 20)
ax.set_xlabel(xlabel = 'Number of Skill Moves', fontsize = 16)
ax.set_ylabel(ylabel = 'Count', fontsize = 16)
plt.show()
```



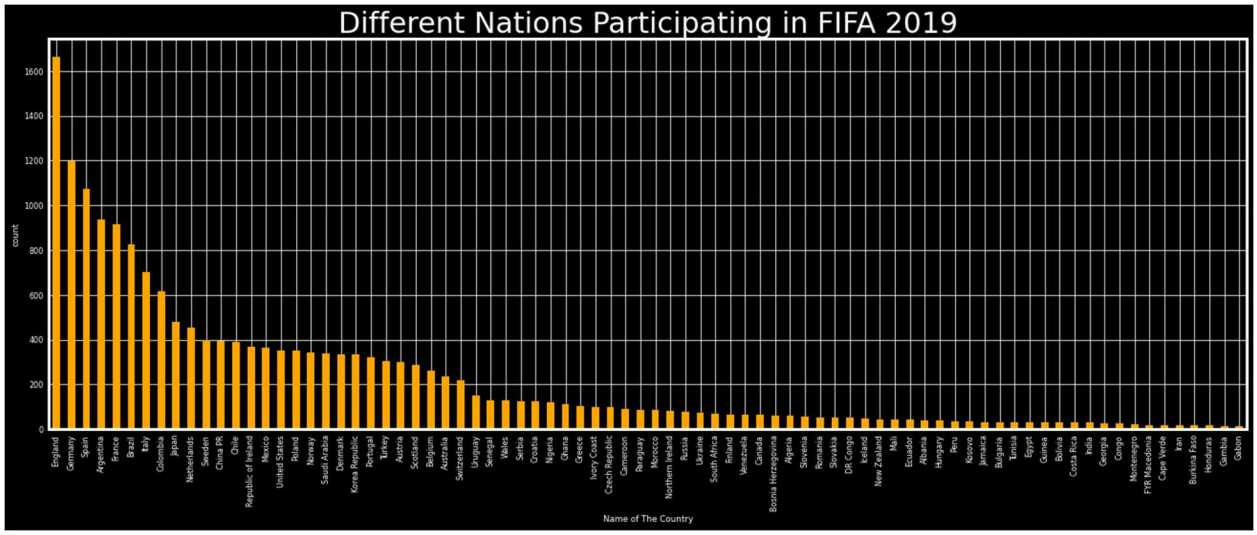
```
#Height of Players
plt.figure(figsize = (13, 8))
ax = sns.countplot(x = 'Height', data = data, palette = 'dark')
ax.set_title(label = 'Count of players on Basis of Height', fontsize = 20)
ax.set_xlabel(xlabel = 'Height in Foot per inch', fontsize = 16)
ax.set_ylabel(ylabel = 'Count', fontsize = 16)
plt.show()
```




```
#Weight of Players
plt.figure(figsize = (20, 5))
sns.distplot(data['Weight'], color = 'pink')
plt.title('Different Weights of the Players Participating in FIFA 2019', fontsize
 = 20)
plt.xlabel('Heights associated with the players', fontsize = 16)
plt.ylabel('count of Players', fontsize = 16)
plt.show()
```

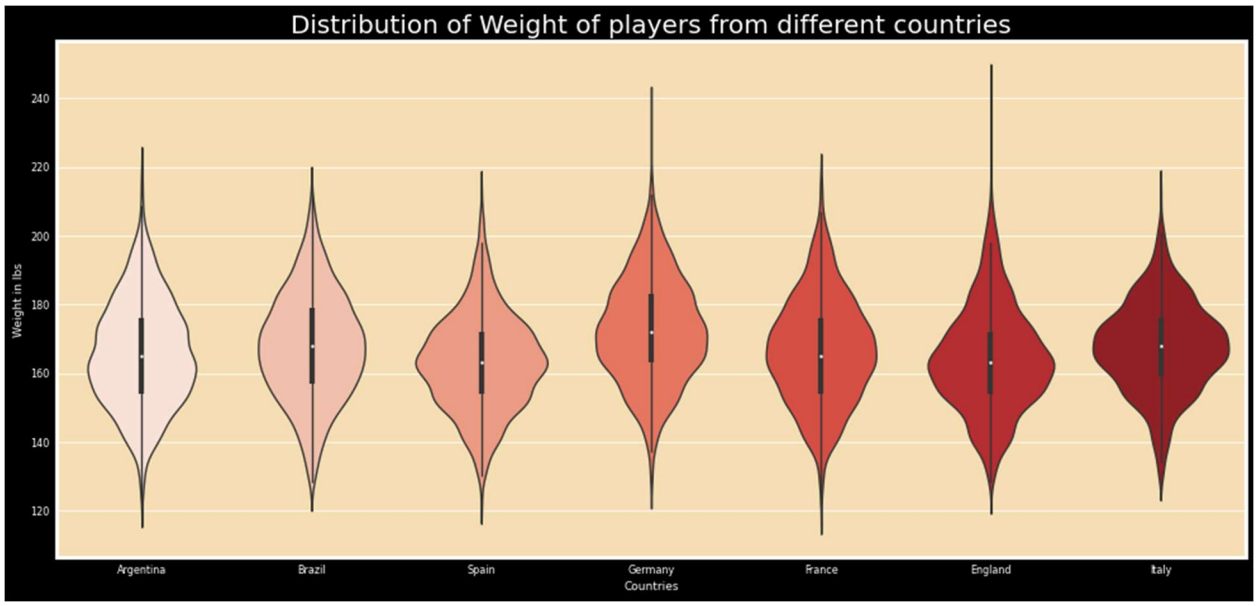



```
# Nations
plt.style.use('dark_background')
data['Nationality'].value_counts().head(80).plot.bar(color = 'orange', figsize =
(20, 7))
plt.title('Different Nations Participating in FIFA 2019', fontsize = 30, fontweig
ht = 20)
plt.xlabel('Name of The Country')
plt.ylabel('count')
ax.set(facecolor = "wheat")
plt.show()
```

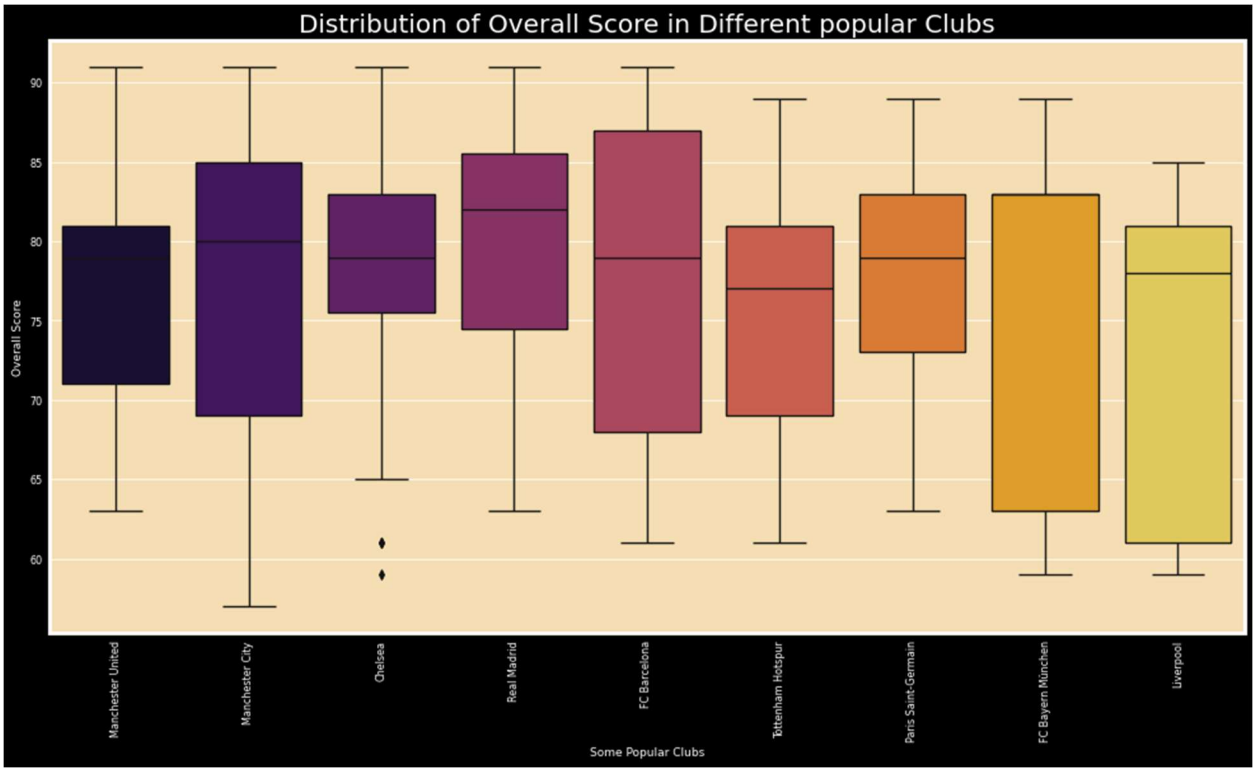



```
# Every Nations' Player and their Weights

some_countries = ('England', 'Germany', 'Spain', 'Argentina', 'France', 'Brazil',
 'Italy', 'Columbia')
data_countries = data.loc[data['Nationality'].isin(some_countries) & data['Weight
']]
```

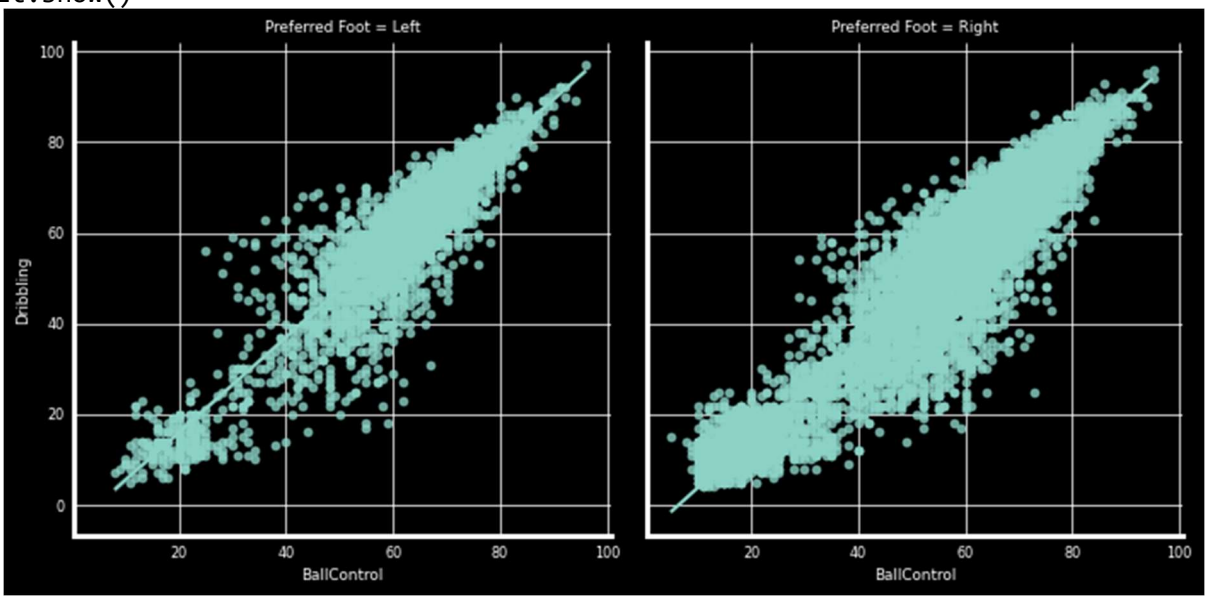
```
plt.rcParams['figure.figsize'] = (15, 7)
ax = sns.violinplot(x = data_countries['Nationality'], y = data_countries['Weight
'], palette = 'Reds')
ax.set_xlabel(xlabel = 'Countries', fontsize = 9)
ax.set_ylabel(ylabel = 'Weight in lbs', fontsize = 9)
ax.set_title(label = 'Distribution of Weight of players from different countries'
, fontsize = 20)
ax.set(facecolor = "wheat")
plt.show()
```



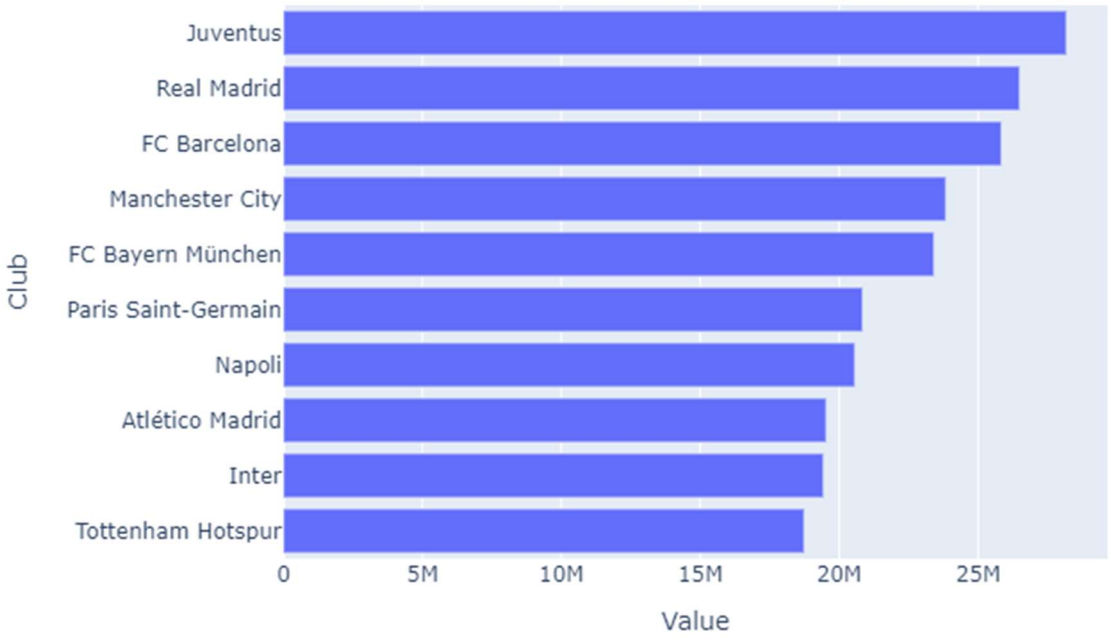

```
some_clubs = ('Manchester City', 'FC Bayern München', 'Chelsea', 'Liverpool',
              'Manchester United', 'Manchestar City',
             'Tottenham Hotspur', 'FC Barcelona', 'Paris Saint-Germain',
              'Chelsea', 'Real Madrid')

data_clubs = data.loc[data['Club'].isin(some_clubs) & data['Overall']]

plt.rcParams['figure.figsize'] = (15, 8)
ax = sns.boxplot(x = data_clubs['Club'], y = data_clubs['Overall'], palette = 'in
ferno')
ax.set_xlabel(xlabel = 'Some Popular Clubs', fontsize = 9)
ax.set_ylabel(ylabel = 'Overall Score', fontsize = 9)
ax.set_title(label = 'Distribution of Overall Score in Different popular Clubs',
fontsize = 20)
plt.xticks(rotation = 90)
ax.set(facecolor = "wheat")
plt.show()
```




```
#Comparing the performance of left-footed and right-footed footballers
# ballcontrol vs dribbing

sns.lmplot(x = 'BallControl', y = 'Dribbling', data = data, col = 'Preferred Foot
')
ax.set(facecolor = "Wheat")
plt.show()
```



```
Top 20 teams with highest player average value
club = data.groupby('Club')['Value'].mean().reset_index().sort_values('Value',
                                                    ascending = True).tail(10)
fig = px.bar(club, x = 'Value', y = 'Club', orientation = 'h')
fig.show()
```




```
#Top Skills of every Playing Positions

player_features = ["Crossing", "Finishing", "HeadingAccuracy", "ShortPassing",
                   "Volleys", "Dribbling", "Curve", "FKAccuracy", "LongPassing",
 "BallControl", "Acceleration", "SprintSpeed", "Agility", "Reactions",
                   "Balance", "ShotPower", "Jumping", "Stamina", "Strength",
                   "LongShots", "Aggression", "Interceptions", "Positioning",
                   "Vision", "Penalties", "Composure", "Marking",
                   "StandingTackle", "SlidingTackle", "GKDiving", "GKHandling",
                   "GKKicking", "GKPositioning", "GKReflexes"]
df_position = pd.DataFrame()
for position_name, features in data.groupby(data['Position'])[player_features].me
an().iterrows():
    top_features = dict(features.nlargest(5))
    df_position[position_name] = tuple(top_features)
df_position.head()
```

	CAM	CB	CDM	CF	CM	GK	LAM	LB	LCB	LCM	...	RB	RCB	RCM	RDM	RF	RM	RS	RW	RWB	ST
0	Balance	Strength	Stamina	Agility	Balance	GKReflexes	Agility	SprintSpeed	Strength	Stamina	...	SprintSpeed	Strength	Stamina	Stamina	Agility	Acceleration	SprintSpeed	Acceleration	SprintSpeed	SprintSpeed
1	Agility	Jumping	Aggression	Balance	ShortPassing	GKDiving	Balance	Acceleration	Jumping	ShortPassing	...	Stamina	Jumping	ShortPassing	ShortPassing	Acceleration	SprintSpeed	Strength	SprintSpeed	Acceleration	Strength
2	Acceleration	StandingTackle	Strength	Acceleration	Agility	GKPositioning	SprintSpeed	Stamina	StandingTackle	Balance	...	Acceleration	Aggression	Agility	Aggression	Balance	Agility	Acceleration	Agility	Stamina	Acceleration
3	SprintSpeed	Aggression	ShortPassing	SprintSpeed	Stamina	GKHandling	Acceleration	Balance	Aggression	Agility	...	Balance	StandingTackle	Balance	Strength	BallControl	Balance	Agility	Balance	Agility	Jumping
4	BallControl	HeadingAccuracy	Jumping	Dribbling	Acceleration	GKKicking	Dribbling	Agility	HeadingAccuracy	BallControl	...	Jumping	HeadingAccuracy	BallControl	Jumping	SprintSpeed	Dribbling	ShotPower	Dribbling	Balance	Finishing

5 rows × 27 columns

```
osition = []
player = []
club_l = []
for col in df_position.columns:
    tmp_df = pd.DataFrame()
    l = [df_position[col].values]
    l = l[0]
    l = list(l)
    l.append("Name")
    tmp_df = pd.DataFrame.copy(data[data["Position"] == col][l])
    tmp_df['mean'] = np.mean(tmp_df.iloc[: , :-1], axis = 1)
    name = tmp_df["Name"][tmp_df["mean"] == tmp_df["mean"].max()].values[0]
    club = data['Club'][data["Name"] == str(name)].values[0]
    position.append(col)
    player.append(name)
    club_l.append(club)

gk = ["GK"]
forward = ["LS", "ST", "RS", "LF", "CF", "RF"]
midfielder = ["LW", "RW", "LAM", "CAM", "RAM", "LM", "LCM",
              "CM", "RCM", "RM", "LDM", "CDM", "RDM"]
defenders = ["LWB", "RWB", "LB", "LCB", "CB", "RCB", "RB"]

print ("Goalkeeper: ")
for p, n, c in zip(position, player, club_l):
    if p in gk:
        print("{} [Club : {}, Position: {}]".format(n, c, p))
print ("\nFORWARD: ")
for p, n, c in zip(position, player, club_l):
    if p in forward:
        print("{} [Club : {}, Position: {}]".format(n, c, p))
print ("\nMIDFIELDER: ")
for p, n, c in zip(position, player, club_l):
    if p in midfielder:
        print("{} [Club : {}, Position: {}]".format(n, c, p))
print ("\nDEFENDER: ")
for p, n, c in zip(position, player, club_l):
    if p in defenders:
        print("{} [Club : {}, Position: {}]".format(n, c, p))
```

Output:

```
Goalkeeper:
De Gea [Club : Manchester United, Position: GK]

FORWARD:
S. Giovinco [Club : Toronto FC, Position: CF]
E. Hazard [Club : Chelsea, Position: LF]
J. Martínez [Club : Atlanta United, Position: LS]
L. Messi [Club : FC Barcelona, Position: RF]
A. Saint-Maximin [Club : OGC Nice, Position: RS]
Cristiano Ronaldo [Club : Juventus, Position: ST]

MIDFIELDER:
H. Nakagawa [Club : Kashiwa Reysol, Position: CAM]
Casemiro [Club : Real Madrid, Position: CDM]
N. Keïta [Club : Liverpool, Position: CM]
Paulo Daineiro [Club : Ceará Sporting Club, Position: LAM]
David Silva [Club : Manchester City, Position: LCM]
N. Kanté [Club : Chelsea, Position: LDM]
Douglas Costa [Club : Juventus, Position: LM]
Neymar Jr [Club : Paris Saint-Germain, Position: LW]
J. Cuadrado [Club : Juventus, Position: RAM]
L. Modrić [Club : Real Madrid, Position: RCM]
P. Pogba [Club : Manchester United, Position: RDM]
Gelson Martins [Club : Atlético Madrid, Position: RM]
R. Sterling [Club : Manchester City, Position: RW]
G. Chiellini [Club : Juventus, Position: LCB] M. Pedersen [Club : Strømsgodset
IF, Position: LWB] Nélson Semedo [Club : FC Barcelona, Position: RB] Sergio Ramos
[Club : Real Madrid, Position: RCB] M. Millar [Club : Central Coast Mariners,
Position: RWB]
```

BEST FORMATIONS:

```
plt.figure(1 , figsize = (15 , 7))
create_football_formation(formation = [ 4 , 2 ] ,
                          label_4 = [LWB , LCB , RCB , RWB],
                          label_4W = [LW , LCM , CM , RW],
                          label_2 = [LF , RF],
                          )
plt.title('Best Fit for formation 4-4-2')
plt.show()

plt.figure(1 , figsize = (15 , 7))
```

```
create_football_formation(formation = [ 4 , 2 ] ,
                          label_4 = [LB , CB , RCB , RB],
                          label_4W = [LAM , LDM , RDM , RAM],
                          label_2 = [LS , RS],
                          )
plt.title('OR\nBest Fit for formation 4-4-2')
plt.show()


plt.figure(1 , figsize = (15 , 7))
create_football_formation(formation = [ 4 , 2 ] ,
                          label_4 = [LB , CB , RCB , RB],
                          label_4W = [LW , LDM , RDM , RW],
                          label_2 = [CF , ST],
                          )
plt.title('OR\nBest Fit for formation 4-4-2')
plt.show()


plt.figure(1 , figsize = (15 , 7))
create_football_formation(formation = [ 4 , 2 ] ,
                          label_4 = [LB , CB , RCB , RB],
                          label_4W = [LW , LCM , RCM , RW],
                          label_2 = [CF , ST],
                          )
plt.title('OR\nBest Fit for formation 4-4-2')
plt.show()

plt.figure(1 , figsize = (15 , 7))
create_football_formation(formation = [ 4 , 2 ] ,
                          label_4 = [LWB , LCB , RCB , RWB],
                          label_4W = [LW , LCM , CM , RW],
                          label_2 = [LF , RF],
                          )
plt.title('OR\nBest Fit for formation 4-4-2')
plt.show()


plt.figure(1 , figsize = (15 , 7))
create_football_formation(formation = [ 4 , 2 , 3 , 1] ,
                          label_4 = [LWB , LCB , RCB , RWB],
                          label_2 = [LCM , RCM],
                          label_3 = [LF , CAM , RF],
                          label_1 = [ST])
```

```
plt.title('Best Fit for formation 4-2-3-1')
plt.show()

plt.figure(1 , figsize = (15 , 7))
create_football_formation(formation = [ 4 , 2 , 3 , 1] ,
                          label_4 = [LWB , LB , RB , RWB],
                          label_2 = [LAM , RAM],
                          label_3 = [LW , CF , RW],
                          label_1 = [ST])
plt.title('OR\nBest Fit for formation 4-2-3-1')
plt.show()

plt.figure(1 , figsize = (15 , 7))
create_football_formation(formation = [ 4 , 2 , 3 , 1] ,
                          label_4 = [LWB , CB , RCB , RWB],
                          label_2 = [CM , CAM],
                          label_3 = [LF , CM , RF],
                          label_1 = [ST])
plt.title('OR\nBest Fit for formation 4-2-3-1')

plt.show()

plt.figure(1 , figsize = (15 , 7))
create_football_formation(formation = [ 4 , 2 , 3 , 1] ,
                          label_4 = [LWB , LCB , RCB , RWB],
                          label_2 = [LCM , RCM],
                          label_3 = [LDM , CAM , RDM],
                          label_1 = [ST])
plt.title('OR\nBest Fit for formation 4-2-3-1')
plt.show()

plt.figure(1 , figsize = (15 , 7))
create_football_formation(formation = [ 5, 4 , 1 ] ,
                          label_5 = [LWB , LCB , CB , RCB , RWB],
                          label_4 = [LW, LDM , RDM , RW],
                          label_1 = [ST])
plt.title('Best Fit for formation 5-4-1')
plt.show()

plt.figure(1 , figsize = (15 , 7))
create_football_formation(formation = [ 4 , 3 ] ,
                          label_4 = [LWB , LCB , RCB , RWB],
                          label_3 = [LW, CAM , RW],
                          label_3W = [LF , ST , RF])
```

```
plt.title('Best Fit for formation 4-3-3')
plt.show()


plt.figure(1 , figsize = (15 , 7))
create_football_formation(formation = [ 4 , 3 ] ,
                          label_4 = [LWB , CB , RB , RWB],
                          label_3 = [LAM, CM , RAM],
                          label_3W = [LS , CF , RS])
plt.title('OR\nBest Fit for formation 4-3-3')
plt.show()

plt.figure(1 , figsize = (15 , 7))
create_football_formation(formation = [ 4 , 3 ] ,
                          label_4 = [LB , LCB , RCB , RB],
                          label_3 = [LDM, CDM , RDM],
                          label_3W = [LF , CF , RF])
plt.title('OR\nBest Fit for formation 4-3-3')
plt.show()

plt.figure(1 , figsize = (15 , 7))
create_football_formation(formation = [ 4 , 3] ,
                          label_4 = [LWB , CB , RB , RWB],
                          label_3 = [LAM, CAM , RAM],
                          label_3W = [LS , ST , RS])
plt.title('OR\nBest Fit for formation 4-3-3')
plt.show()


plt.figure(1 , figsize = (15 , 7))
create_football_formation(formation = [ 4 , 3] ,
                          label_4 = [LWB , CB , RB , RWB],
                          label_3 = [LCM, CAM , RCM],
                          label_3W = [LF , ST , RF])
plt.title('OR\nBest Fit for formation 4-3-3')
plt.show()
```

WORLD'S BEST PLAYING XI:

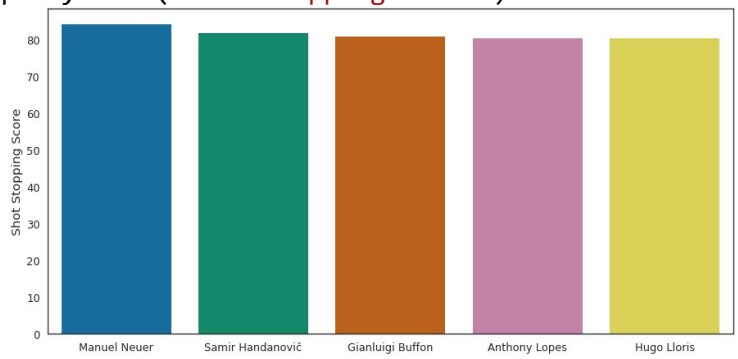
BEST GOALKEEPER:

In order to get the best goalkeeper, I'll be analyzing the data for the below men tioned parameters:

Shot Stopper: A goalkeeper who is strong in stopping shots taken by opponents.
Sweeper: A goalkeeper who is strong in playing with his feet and making passes.

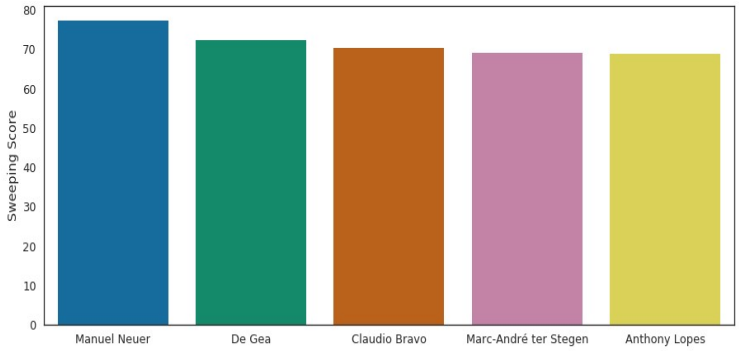
```
df = pd.read_csv("FullData.csv")
df.head(7)
```

```
#weights
a = 0.5
b = 1
c= 2
d = 3
#GoalKeeping Characterstics
df['gk_Shot_Stopper'] = (b*df.Reactions + b*df.Composure + a*df.Speed + a*df.Stre
ngth + c*df.Jumping + b*df.GK_Positioning + c*df.GK_Diving + d*df.GK_Reflexes + b
*df.GK_Handling)/(2*a + 4*b + 2*c + 1*d)
df['gk_Sweeper'] = (b*df.Reactions + b*df.Composure + b*df.Speed + a*df.Short_Pas
s + a*df.Long_Pass + b*df.Jumping + b*df.GK_Positioning + b*df.GK_Diving + d*df.G
K_Reflexes + b*df.GK_Handling + d*df.GK_Kicking + 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)[:5]
x1 = np.array(list(sd['Name']))
y1 = np.array(list(sd['gk_Shot_Stopper']))
sns.barplot(x1, y1, palette= "afmhot")
ax.set(facecolor = "wheat")
plt.ylabel("Shot Stopping Score")
```



Based on the shot-stopper characteristics, it can be inferred that Manuel Neuer is the best goalkeeper as you can see he tops the above list. Let us now plot the other parameter(Sweeper) as well.

```
plt.figure(figsize=(15,6))
sd = df.sort_values('gk_Sweeper', ascending=False)[:5]
x2 = np.array(list(sd['Name']))
y2 = np.array(list(sd['gk_Sweeper']))
sns.barplot(x2, y2, palette= "colorblind")
plt.ylabel("Sweeping Score")
```



Manuel Neuer tops the chart here as well. Based on the two parameters we used, we can conclude that Manuel Neuer would be the best choice goalkeeper for our squad.

BEST DEFENDERS:

In order to find the best defenders, I'll be using following attributes to fetch the best defenders:

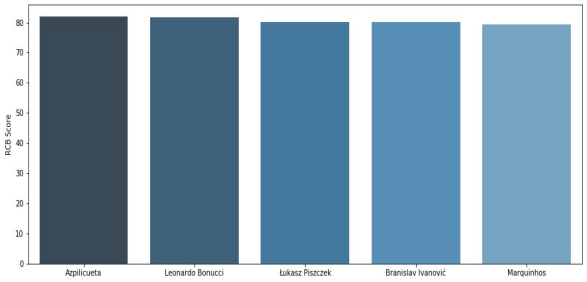
Centre Backs: We need two center-backs. One who plays LCB and the other who plays RCB.

Wing Backs: We again need two wing backs. One who plays on the Left and the other who plays on the right.

```
#Choosing Defenders
df['df_centre_backs'] = ( d*df.Reactions + c*df.Interceptions +
d*df.Sliding_Tackle + d*df.Standing_Tackle + b*df.Vision+ b*df.Composure +
b*df.Crossing +a*df.Short_Pass + 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)
```

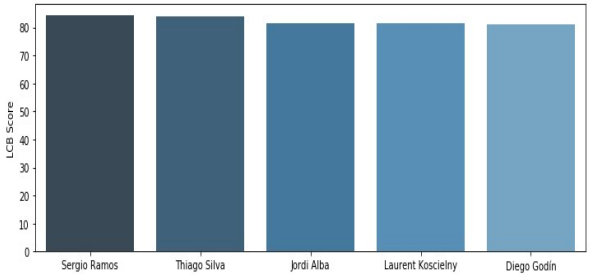
Based on the above parameters, I'll be predicting 4 best defenders: 2 Centre backs and 2 wing backs. Let us first plot left centre back and right centre back.

```
#LEFT CENTRAL DEFENDER:
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']))
sns.barplot(x2, y2, palette=sns.color_palette("Blues_d"))
plt.ylabel("LCB Score")
```



Based on the left centre back characteristics, it can be inferred that **Sergio Ramos** is the Best Left Central Defender. Let us now plot the RCB as well.

```
#RIGHT CENTRAL DEFENDER:
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']))
sns.barplot(x2, y2, palette=sns.color_palette("Blues_d"))
plt.ylabel("RCB Score")
```



Based on the right centre back characteristics, it can be inferred that Azpilicueta is the Best Right Central Defender.

Next, let us pick the World's best left wing back/ left back.

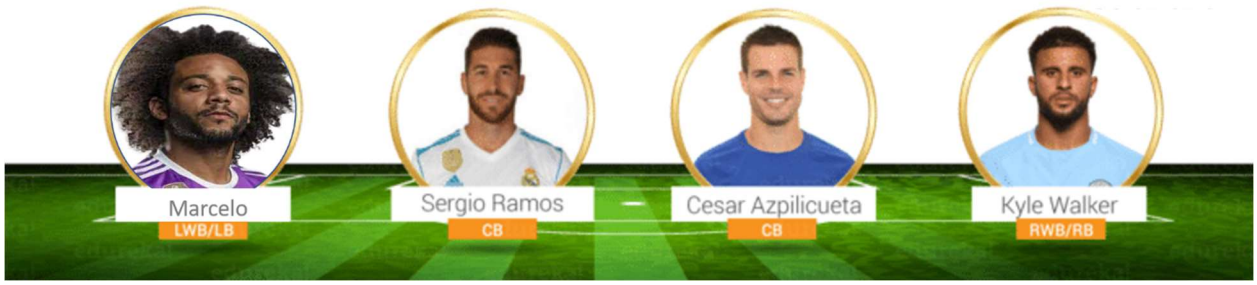
```
#LEFT WING BACK:
plt.figure(figsize=(15,6))
```

```
sd = df[(df['Club_Position'] == 'LWB') | (df['Club_Position'] ==
'LB')].sort_values('df_wb_Wing_Backs', ascending=False)[:5]
x4 = np.array(list(sd['Name']))
y4 = np.array(list(sd['df_wb_Wing_Backs']))
sns.barplot(x4, y4, palette=sns.color_palette("Blues_d"))
plt.ylabel("Left Back Score")
```

As per the above analysis, it is evident that Marcelo as the best LWB/LB defender.

```
#RIGHT WING BACK:
plt.figure(figsize=(15,6))
sd = df[(df['Club_Position'] == 'RWB') | (df['Club_Position'] ==
'RB')].sort_values('df_wb_Wing_Backs', ascending=False)[:5]
x5 = np.array(list(sd['Name']))
y5 = np.array(list(sd['df_wb_Wing_Backs']))
sns.barplot(x5, y5, palette=sns.color_palette("Blues_d"))
plt.ylabel("Right Back Score")
```

As per the above analysis, it is evident that Kyle Walker is the best RWB/RB defender.



BEST MIDFIELDERS:

As per my game formation 4-3-3, I have to choose 3 midfielders. In order to find these, I'll be analyzing the data for the below mentioned parameters:

Playmaker: A playmaker is someone who will move the ball to the attacking 3rd from defence or midfield.

Beast:A beast is a typical box-to-box player with loads of energy and who can boss the midfield.

Controller:A controller is the person who is orchestrating your midfield engine by either sitting back or going forward based on dynamic needs.

```
#Midfielding Indices
df['mf_playmaker'] = (d*df.Ball_Control + d*df.Dribbling + a*df.Marking +
d*df.Reactions + d*df.Vision + c*df.Attacking_Position + c*df.Crossing +
```

```
d*df.Short_Pass + c*df.Long_Pass + c*df.Curve + b*df.Long_Shots +
c*df.Freekick_Accuracy)/(1*a + 1*b + 4*c + 4*d)
df['mf_beast'] = (d*df.Agility + c*df.Balance + b*df.Jumping + c*df.Strength +
d*df.Stamina + a*df.Speed + c*df.Acceleration + d*df.Short_Pass + c*df.Aggression
+ d*df.Reactions + b*df.Marking + b*df.Standing_Tackle + b*df.Sliding_Tackle +
b*df.Interceptions)/(1*a + 5*b + 4*c + 4*d)
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)

#PLAYMAKER:
plt.figure(figsize=(15,6))
ss = df[(df['Club_Position'] == 'CAM') | (df['Club_Position'] == 'LAM') |
(df['Club_Position'] == 'RAM')].sort_values('mf_playmaker', ascending=False)[:5]
x3 = np.array(list(ss['Name']))
y3 = np.array(list(ss['mf_playmaker']))
sns.barplot(x3, y3, palette=sns.diverging_palette(145, 280, s=85, l=25, n=5))
plt.ylabel("PlayMaker Score")
As per the above analysis, I'll pick Mesut Ozil as the best Playmaker.

#BEAST:
plt.figure(figsize=(15,6))
ss = df[(df['Club_Position'] == 'RCM') | (df['Club_Position'] ==
'RM')].sort_values('mf_beast', ascending=False)[:5]
x2 = np.array(list(ss['Name']))
y2 = np.array(list(ss['mf_beast']))
sns.barplot(x2, y2, palette=sns.diverging_palette(145, 280, s=85, l=25, n=5))
plt.ylabel("Beast Score")
As per the above analysis, I'll pick N' Golo Kante as the best Beast/ Right
Central Midfielder.

#CONTROLLER:
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']))
sns.barplot(x1, y1, palette=sns.diverging_palette(145, 280, s=85, l=25, n=5))
plt.ylabel("Controller Score")
As per the above analysis, I'll pick Iniesta as the best controller/ Left Central
Midfielder.
```

BEST ATTACKERS:

In order to find the best attacker, I'll be analyzing the below mentioned parameters:

Attacking Left Wing: He is a player, attacking from the left flank.

Attacking Right Wing: He is a player, attacking from the right flank.

Striker: He is a player attacking from the center.

```
#Attackers
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)
df['att_right_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)
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'] == '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")
```

It's quite evident from the above plot that Ronaldo is the best Left Wing Attacker.

```
plt.figure(figsize=(15,6))
ss = df[(df['Club_Position'] == 'RW') | (df['Club_Position'] == 'RM') |
(df['Club_Position'] == 'RS')].sort_values('att_right_wing', ascending=False)[:5]
x2 = np.array(list(ss['Name']))
y2 = np.array(list(ss['att_right_wing']))
sns.barplot(x2, y2, palette=sns.diverging_palette(255, 133, l=60, n=5,
center="dark"))
plt.ylabel("Right Wing")
```

As per the above analysis, I'll pick Lionel Messi as the right wing attacker.

```
#STRIKER:
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")
```

As per the above analysis, the best striker would be Robert Lewandowski.

CHAPTER 6

OBSERVATIONS AND CONCLUSIONS

TEAM OF THE DECADE (TOTD): 4-3-3 Lineup

Similarly, we can predict the Team of the Season and All time Playing XI.

TEAM OF THE SEASON (UCL- 2021): 4-3-3 Lineup

- The best lineup was predicted on the basis of the player's performance in UEFA Champions League (2020-2021).

ALL TIME BEST XI (TEAM OF THE CENTURY): 4-3-3 Lineup

- It's really a tough task and controversial at the same time to predict the Team of the Century.

- Manager of the Century: **Sir Alex Ferguson**

CHAPTER 7

REFERENCES/ BIBILIOGRAPHY

References:

- Project File Link: https://github.com/surajkr7/ML-PROJECT-ON-FIFA-22-ANALYSIS.git
- FIFA Dataset Link: https://www.kaggle.com/artimous/complete-fifa-2017-player-dataset-global/data#FullData.csv
- FIFA official reference Link: https://sofifa.com/
- EA SPORTS FIFA 22 Link: https://www.ea.com/games/fifa/fifa-22
- Altman, D.G. (1991) Practical Statistics for Medical Research, London: Chapman & Hall.
- Ariel, G. (2007) 'Sports technologies from Mexico City Olympics to the future Olympics in Beijing 2008', paper presented at the 6th International Symposium of the International Association of Computer Science in Sport, Calgary, June.
- Armitage, P.J. (2006) An Analysis of the Knock Out Stages of the 2003 Rugby Union World Cup. Unpublished BSc Hons thesis, University of Wales Institute Cardiff.
- Atkinson, G. (2002) 'What is this thing called measurement error?', paper presented at the 12th Commonwealth International Sport Conference, Manchester, July.
- Atkinson, G. and Nevill, A.M. (1998) 'Statistical methods for addressing measurement error (reliability) in variables relevant to sports medicine', Sports Medicine, 26: 217–38.
- Australian Open (2009) 'Official website of the Australian Open', Available (accessed 27 July 2009). Baker, J., Ramsbottom, R. and Hazeldine, R. (1993) 'Maximal shuttle performance over 40m as a measure of anaerobic performance', British Journal of Sports Medicine, 27: 228–32.