Football Player Dataset Analysis

Levent Ergul

Libraries

Pandas: is a fast, powerful, flexible and easy to use open source data analysis and manipulation tool, built on top of the Python programming language.

Numpy: is a library for the Python programming language, adding support for large, multi-dimensional arrays and matrices, along with a large collection of high-level mathematical functions to operate on these arrays

Seaborn: Seaborn is a data visualization library built on top of matplotlib and closely integrated with pandas data structures in Python. Visualization is the central part of Seaborn which helps in exploration and understanding of data.

Dataset

General Information

Dataset have 89 features and 18207 examples. Features type is different.

- 38 variables are float,
- 6 variables are integer,
- 45 variables object.

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 18207 entries, 0 to 18206
Data columns (total 89 columns):
 #   Column                    Non-Null Count  Dtype
---  ------                    --------------  -----
 0   Unnamed: 0                18207 non-null  int64
 1   ID                        18207 non-null  int64
 2   Name                      18207 non-null  object
 3   Age                       18207 non-null  int64
 4   Photo                     18207 non-null  object
 5   Nationality               18207 non-null  object
 6   Flag                      18207 non-null  object
 7   Overall                   18207 non-null  int64
 8   Potential                 18207 non-null  int64
 9   Club                      17966 non-null  object
 10  Club Logo                 18207 non-null  object
 11  Value                     18207 non-null  object
 12  Wage                      18207 non-null  object
 13  Special                   18207 non-null  int64
 14  Preferred Foot            18159 non-null  object
 15  International Reputation  18159 non-null  float64
 16  Weak Foot                 18159 non-null  float64
 17  Skill Moves               18159 non-null  float64
 18  Work Rate                 18159 non-null  object
 19  Body Type                 18159 non-null  object
 20  Real Face                 18159 non-null  object
 21  Position                  18147 non-null  object
 22  Jersey Number             18147 non-null  float64
 23  Joined                    16654 non-null  object
 24  Loaned From               1264 non-null   object
 25  Contract Valid Until      17918 non-null  object
 26  Height                    18159 non-null  object
 27  Weight                    18159 non-null  object
 28  LS                        16122 non-null  object
 29  ST                        16122 non-null  object
 30  RS                        16122 non-null  object
 31  LW                        16122 non-null  object
 32  LF                        16122 non-null  object
 33  CF                        16122 non-null  object
 34  RF                        16122 non-null  object
 35  RW                        16122 non-null  object
 36  LAM                       16122 non-null  object
 37  CAM                       16122 non-null  object
 38  RAM                       16122 non-null  object
 39  LM                        16122 non-null  object
 40  LCM                       16122 non-null  object
 41  CM                        16122 non-null  object
 42  RCM                       16122 non-null  object
 43  RM                        16122 non-null  object
 44  LWB                       16122 non-null  object
 45  LDM                       16122 non-null  object
 46  CDM                       16122 non-null  object
 47  RDM                       16122 non-null  object
 48  RWB                       16122 non-null  object
 49  LB                        16122 non-null  object
 50  LCB                       16122 non-null  object
 51  CB                        16122 non-null  object
 52  RCB                       16122 non-null  object
 53  RB                        16122 non-null  object
 54  Crossing                  18159 non-null  float64
 55  Finishing                 18159 non-null  float64
 56  HeadingAccuracy           18159 non-null  float64
 57  ShortPassing              18159 non-null  float64
 58  Volleys                   18159 non-null  float64
 59  Dribbling                 18159 non-null  float64
 60  Curve                     18159 non-null  float64
 61  FKAccuracy                18159 non-null  float64
 62  LongPassing               18159 non-null  float64
 63  BallControl               18159 non-null  float64
 64  Acceleration              18159 non-null  float64
 65  SprintSpeed               18159 non-null  float64
 66  Agility                   18159 non-null  float64
 67  Reactions                 18159 non-null  float64
 68  Balance                   18159 non-null  float64
 69  ShotPower                 18159 non-null  float64
 70  Jumping                   18159 non-null  float64
 71  Stamina                   18159 non-null  float64
 72  Strength                  18159 non-null  float64
 73  LongShots                 18159 non-null  float64
 74  Aggression                18159 non-null  float64
 75  Interceptions             18159 non-null  float64
 76  Positioning               18159 non-null  float64
 77  Vision                    18159 non-null  float64
 78  Penalties                 18159 non-null  float64
 79  Composure                 18159 non-null  float64
 80  Marking                   18159 non-null  float64
 81  StandingTackle            18159 non-null  float64
 82  SlidingTackle             18159 non-null  float64
 83  GKDiving                  18159 non-null  float64
 84  GKHandling                18159 non-null  float64
 85  GKKicking                 18159 non-null  float64
 86  GKPositioning             18159 non-null  float64
 87  GKReflexes                18159 non-null  float64
 88  Release Clause            16643 non-null  object
dtypes: float64(38), int64(6), object(45)
memory usage: 12.4+ MB
```

Dataset

Missing Data

One of the most important things is checking the dataset before analyzing it.

Loaned From feature has highest missing values equal to 16943.

Also some features have the missing values. Such as RCM 2085 or Club 241.

	Total	%
Loaned From	16943	93.1
LWB	2085	11.5
LM	2085	11.5
CB	2085	11.5
LCB	2085	11.5
LB	2085	11.5
RWB	2085	11.5
RDM	2085	11.5
CDM	2085	11.5
LDM	2085	11.5
RM	2085	11.5
RCM	2085	11.5
CM	2085	11.5
LCM	2085	11.5
RAM	2085	11.5
RB	2085	11.5
CAM	2085	11.5
LAM	2085	11.5
RW	2085	11.5
RF	2085	11.5
CF	2085	11.5
LF	2085	11.5
LW	2085	11.5
RS	2085	11.5
ST	2085	11.5
LS	2085	11.5
RCB	2085	11.5
Release Clause	1564	8.6
Joined	1553	8.5
Contract Valid Until	289	1.6
Club	241	1.3
Position	60	0.3
Jersey Number	60	0.3
Marking	48	0.3
StandingTackle	48	0.3
SlidingTackle	48	0.3
Dribbling	48	0.3
GKHandling	48	0.3
GKKicking	48	0.3
Weight	48	0.3
Height	48	0.3
Curve	48	0.3
GKPositioning	48	0.3
Penalties	48	0.3
Real Face	48	0.3
Body Type	48	0.3
Work Rate	48	0.3
Skill Moves	48	0.3
Weak Foot	48	0.3
International Reputation	48	0.3
Preferred Foot	48	0.3
Composure	48	0.3
GKDiving	48	0.3
Vision	48	0.3
Agility	48	0.3
Volleys	48	0.3
ShortPassing	48	0.3
HeadingAccuracy	48	0.3
Finishing	48	0.3
Positioning	48	0.3

Dataset

Missing Data Elimination

```
"""Club has 241 missing values and GKKicking 48 val
CAM some specific positions have 2085 missing value
and this is make sense. Be careful. We are dropping

f_players = f_players.loc[f_players['Club'].notnull
f_players = f_players.loc[f_players['GKKicking'].no
f_players.drop(['Loaned From'] ,axis =1 , inplace =
## And checking missing values again
total = f_players.isnull().sum().sort_values(ascend
percent= f_players.isnull().sum()/f_players.isnull(
percent_2 = (round(percent, 1))
missing_data = pd.concat([total, percent_2], axis=1
missing_data.head(30)
```

	Total	%
RWB	1989	11.1
RB	1989	11.1
CB	1989	11.1
LCB	1989	11.1
LB	1989	11.1
RDM	1989	11.1
CDM	1989	11.1
LDM	1989	11.1
LWB	1989	11.1
RM	1989	11.1
RCM	1989	11.1
CM	1989	11.1
LCM	1989	11.1
LM	1989	11.1
RAM	1989	11.1

RB	1992	11.1
CAM	1992	11.1
LAM	1992	11.1
RW	1992	11.1
RF	1992	11.1
CF	1992	11.1
LF	1992	11.1
LW	1992	11.1
RS	1992	11.1
ST	1992	11.1
LS	1992	11.1
RCB	1992	11.1
Release Clause	1275	7.1
Joined	1264	7.1
Curve	0	0.0

Dataset

Converting some features

Some features in the dataset are in the string. For the analysis part, we need to convert them to float or integer.

```
def str2float(amount):
    """
    This function help to us for convert string to float.

    """

    if amount[-1] == 'M':
        return float (amount[1:-1])*1000000
    elif amount[-1] == 'K':
        return float (amount[1:-1])*1000
    else:
        return float(amount[1:])

f_players['Value_Float'] = f_players['Value'].apply(lambda x: str2float(x))
f_players['Wage_Float'] = f_players ['Wage'].apply(lambda x: str2float(x))
```

```
print(f_players['Wage_Float'].dtypes)
```

```
float64
```

Data Visualization

Create Function

In this part, we will look at some features distribution. I will create functions to implementation for these features.

This function is suited for the integer variables.

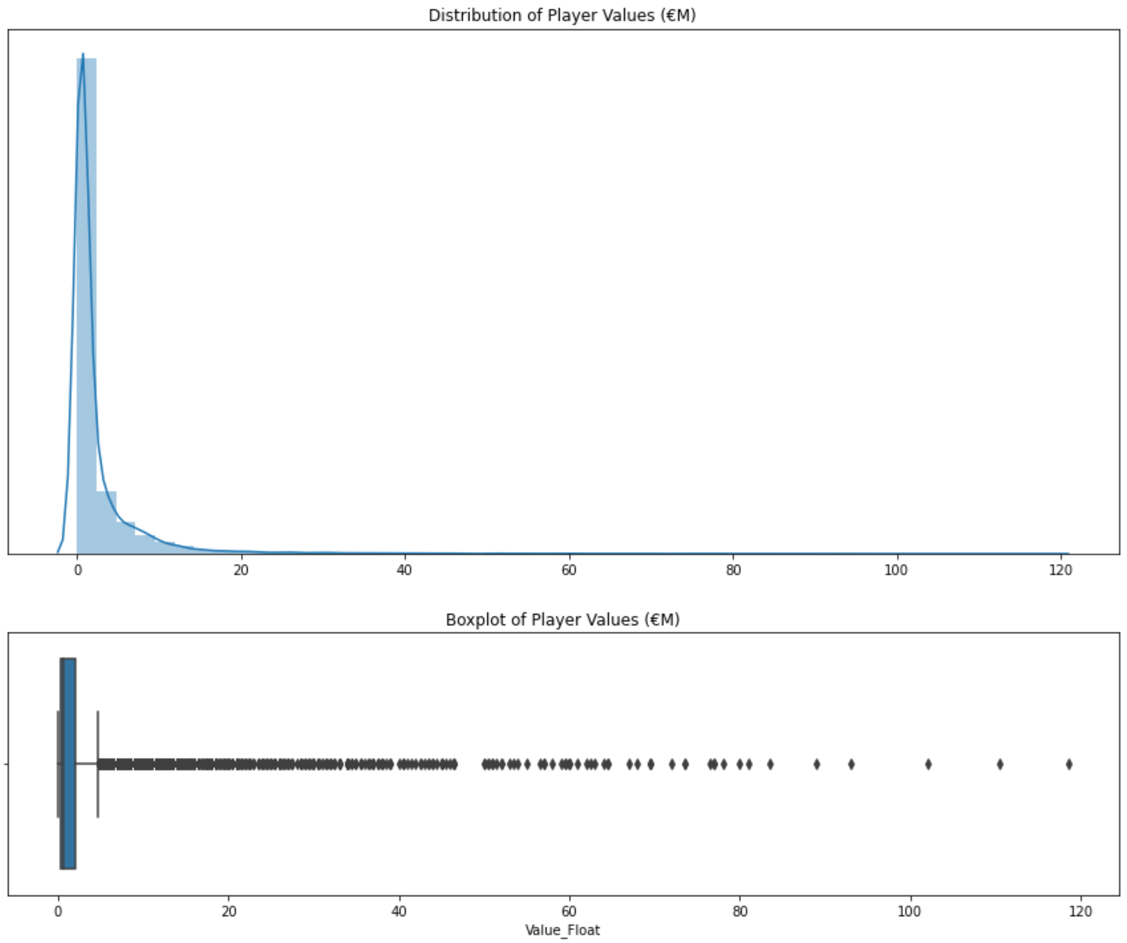
```
def distribution_and_stats_plot(variable,Name):
    """For better understanding for players value distribution.I will draw a plot charts"""

    fig, ax = plt.subplots(nrows=2,figsize =(15,12),gridspec_kw={'height_ratios': [2,1]})
    sns.distplot(variable, ax = ax[0])
    ax[0].set(title = f"Distribution of {Name}",xlabel='',ylabel='')
    ax[0].set_yticks([])

    ax[1]=sns.boxplot(x=variable.columns[0],y=None,data=variable)
    ax[1].set_title(f"Boxplot of {Name}")
    plt.show()
```

Data Visualization

Market Values

Players generally have low market value and for this reason, we can not see general the distribution of that. High prices players are almost not the chart. For that, we c logarithmic distribution cha shows us more clearly.

	Value_Float
count	17918.000000
mean	2.448629
std	5.631804
min	0.000000
25%	0.325000
50%	0.700000
75%	2.100000
max	118.500000



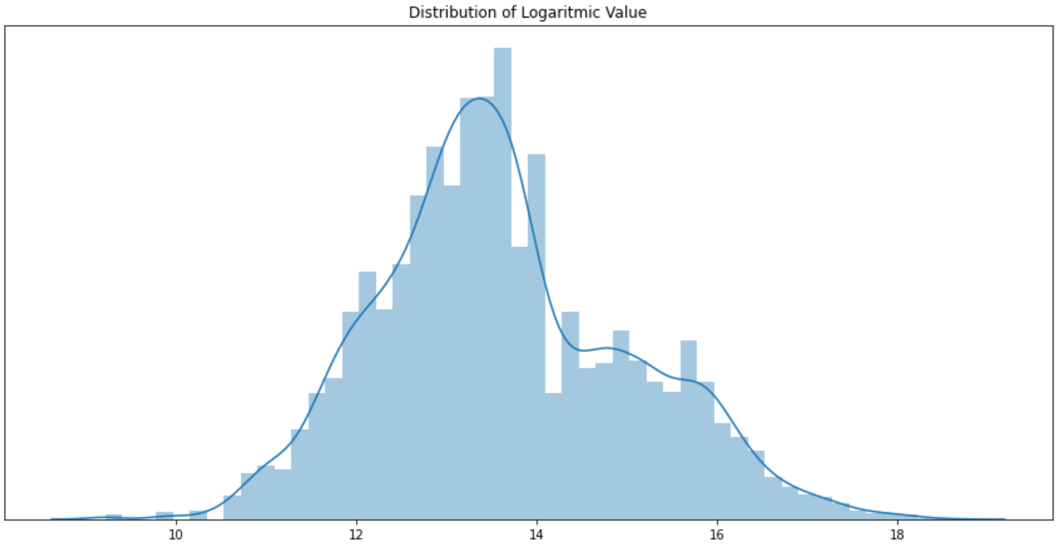
Data Visualization

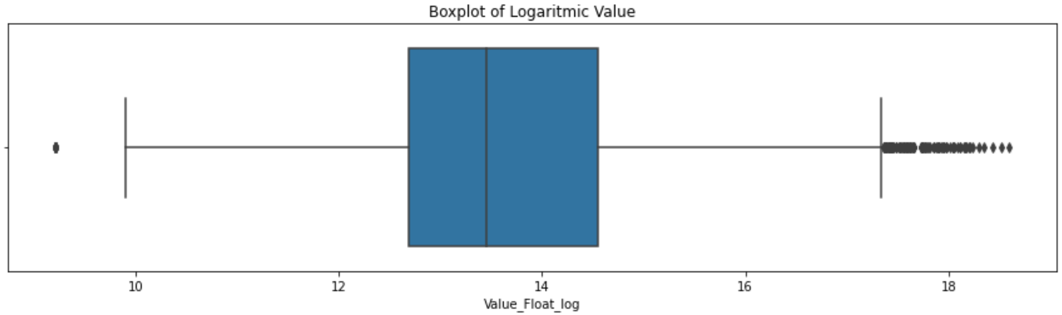
Market Values

There are 11 values equal
dataset, which is why we
these variables. They are n
too much.

	Value_Float_log
count	17907.000000
mean	13.622829
std	1.407122
min	9.210440
25%	12.691584
50%	13.458837
75%	14.557448
max	18.590424

```
#Convert to log

f_players["Value_Float_log"] = np.log(f_players["Value_Float"]+1)

#delete value_log=0
f_players =f_players[f_players["Value_Float_log"] !=0]

distribution_and_stats_plot(f_players[["Value_Float_log"]],'Logaritmic Value')
```



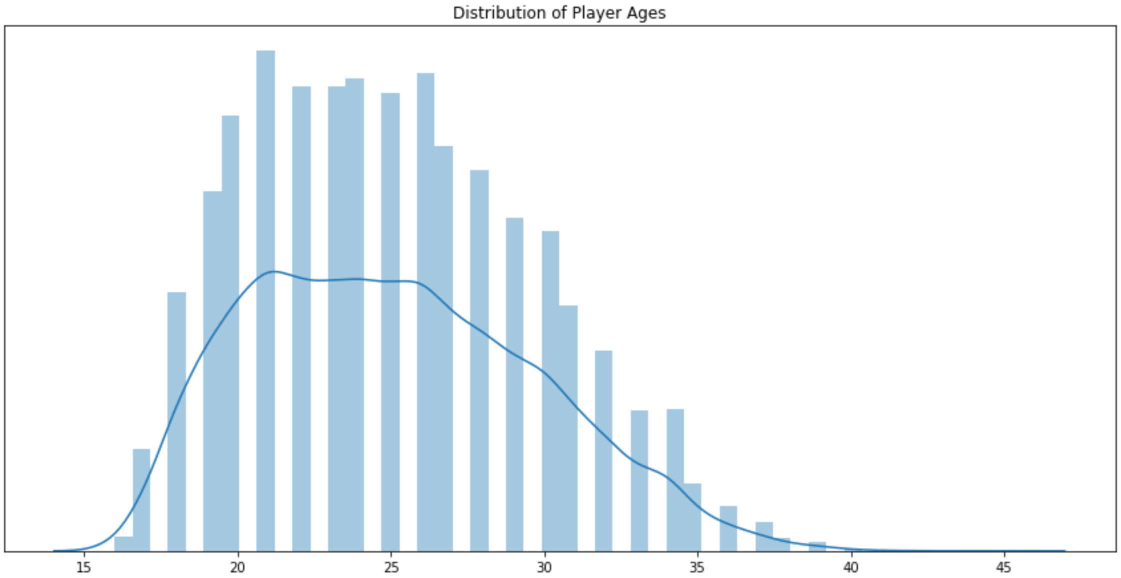


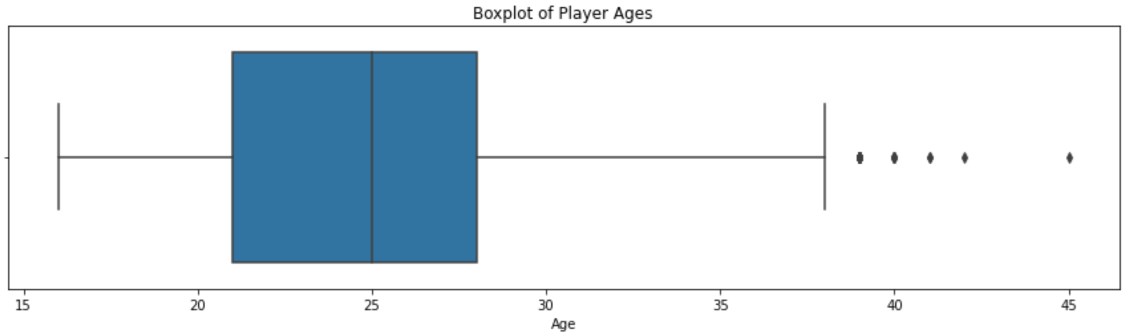
Data Visualization

Age

The age of each player.

	Age
count	17907.000000
mean	25.095605
std	4.660388
min	16.000000
25%	21.000000
50%	25.000000
75%	28.000000
max	45.000000



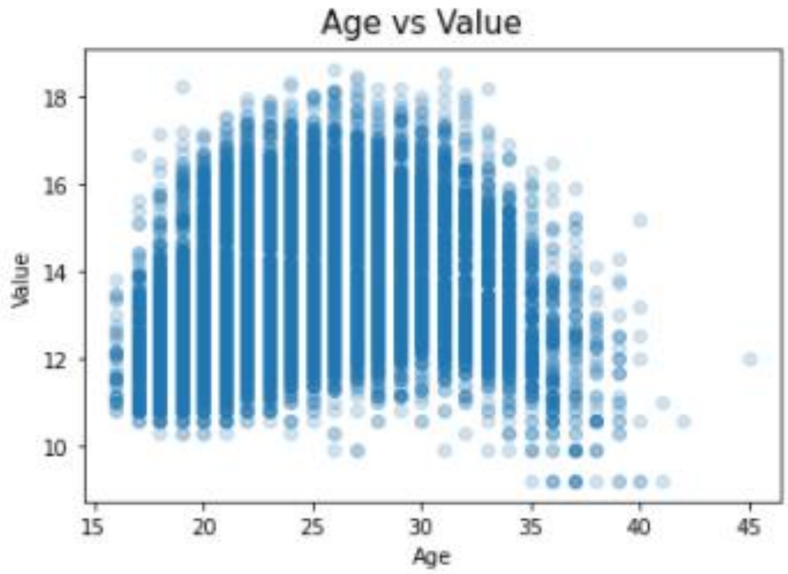


Data Visualization

Age vs Market Value

There seems to be slight positive correlation between age and value.

When the player age is approximately >35 then the values tend to decline.



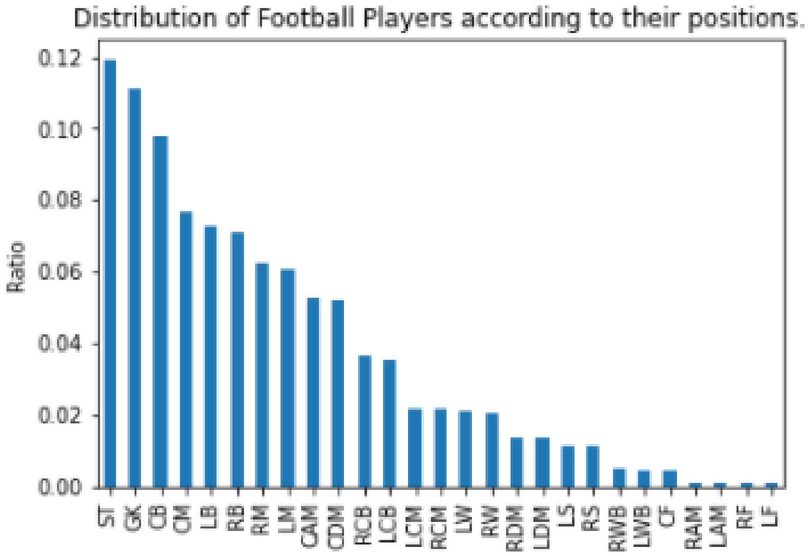
Pearson correlation = 0.07845203553213996

Data Visualization

Position

In this step we will examine distribution of the players position.

```
f_players['Position'].value_counts(normalize= True).plot(kind='bar')
plt.title("Distribution of Football Players according to their positions.")
plt.ylabel("Ratio")
plt.show()

print(f"Number of positions= {len(f_players['Position'].unique())}")
```



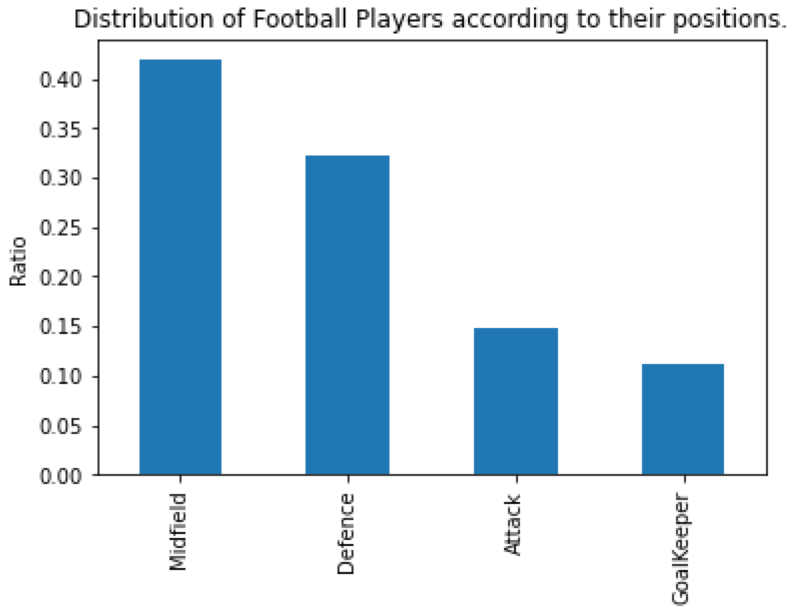
Number of positions= 27

Data Visualization

Position

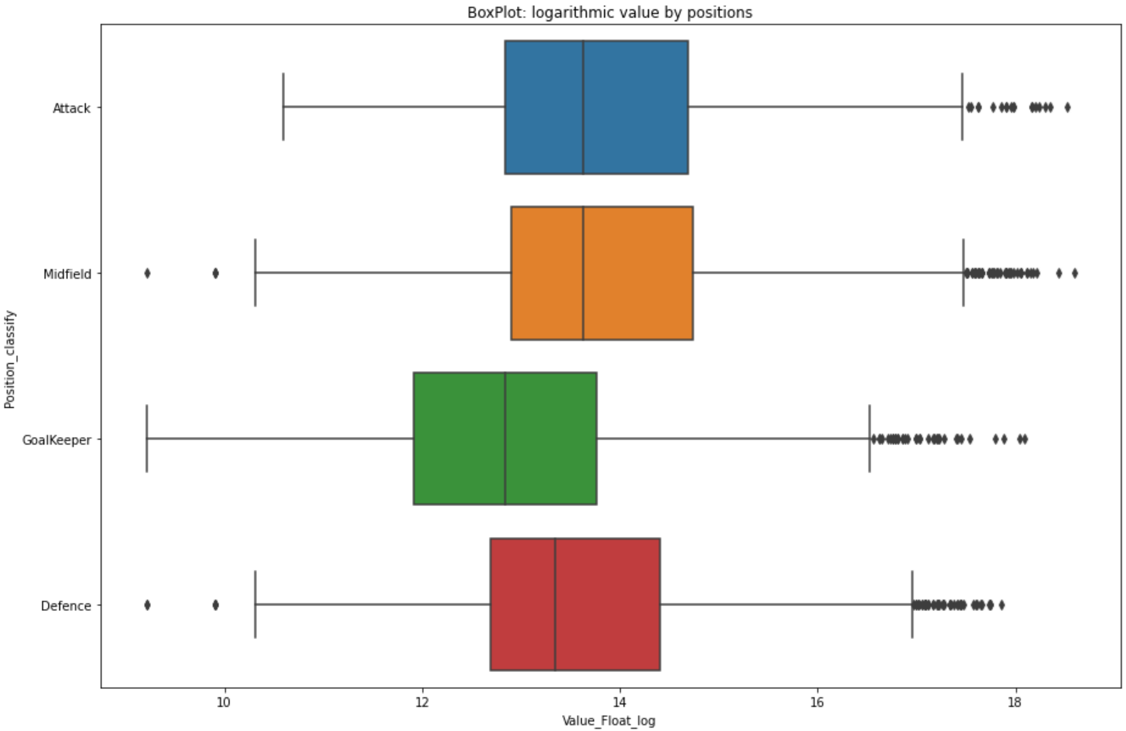
In this step, I'll classify the players' positions. Such as Attacker, midfielder, defender and goalkeeper. I will create a function for this.

```
def classify_position(position):
    """This function help to you for classification football player position."""

    Position_dict= { 'Attack' : ['ST','LS','RS','CF','RF','LF'],
                    'Midfield': ['CM','RM','LM','CAM','CDM','LCM','RCM','LW',
                                 'RW','RDM','LDM','LAM','RAM'],
                     'Defence': ['CB','LB','RB','RCB','LCB','RWB','LWB'],
                   'GoalKeeper':['GK']}

    for group , position_list in Position_dict.items():

        if position in position_list:
            return group

    return np.nan
```



Data Visualization

Position vs Market Value

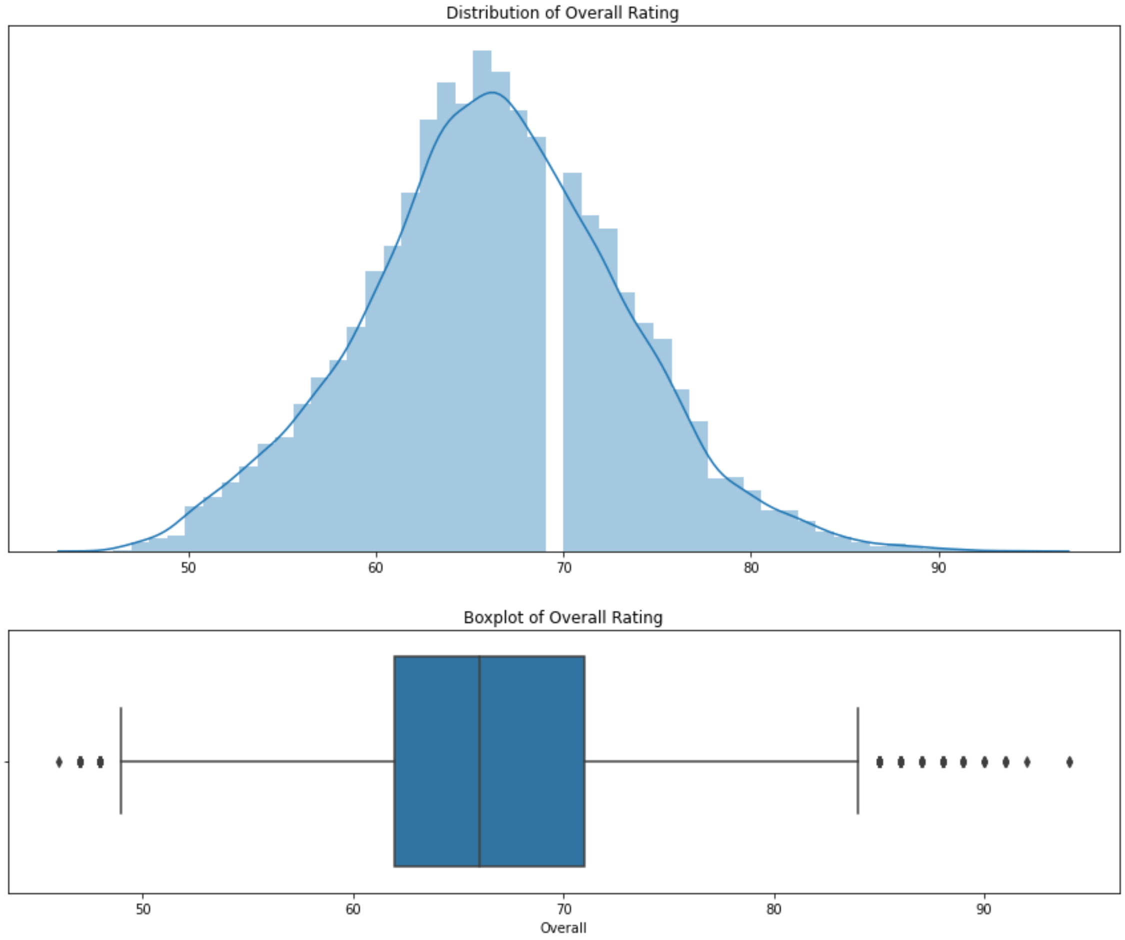



```
plt.figure(figsize=(15,10))
sns.boxplot(x="Value_Float_log",y="Position_classify",data=f_players)
plt.title("BoxPlot: logarithmic value by positions")
plt.show()
```

Data Visualization

Overall Rating

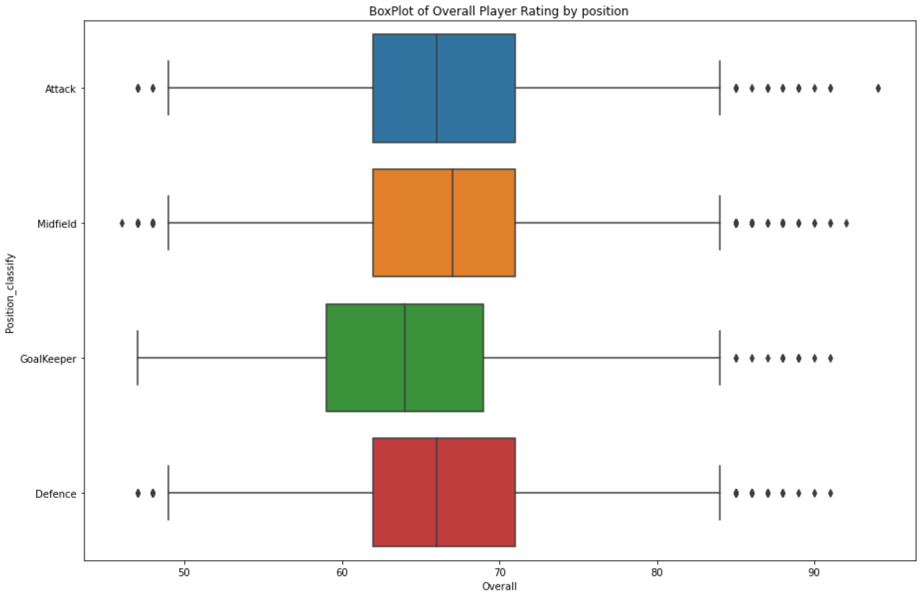
	Overall
count	17907.000000
mean	66.239571
std	6.926818
min	46.000000
25%	62.000000
50%	66.000000
75%	71.000000
max	94.000000



Data Visualization

Overall Rating vs Position

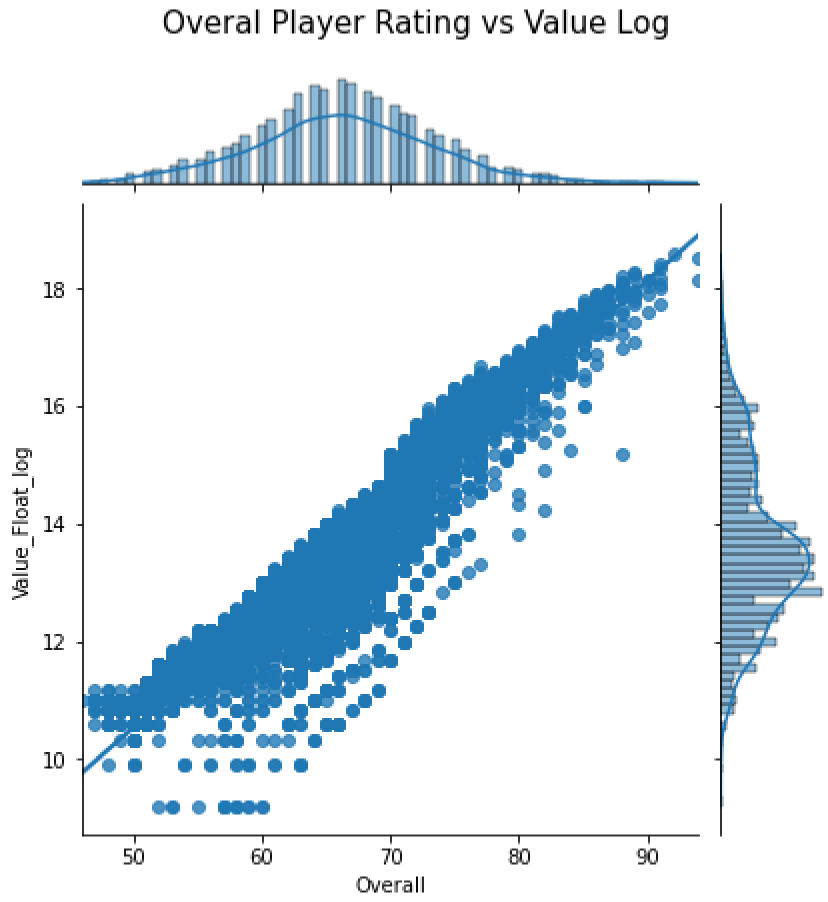
- Goalkeepers less overall rating.
- Others are similar to each ones.



Data Visualization

Overall Rating vs Value

Overall rating is highly correlated with log value with an R2 of 0.89 and is likely to be the most predictive feature.

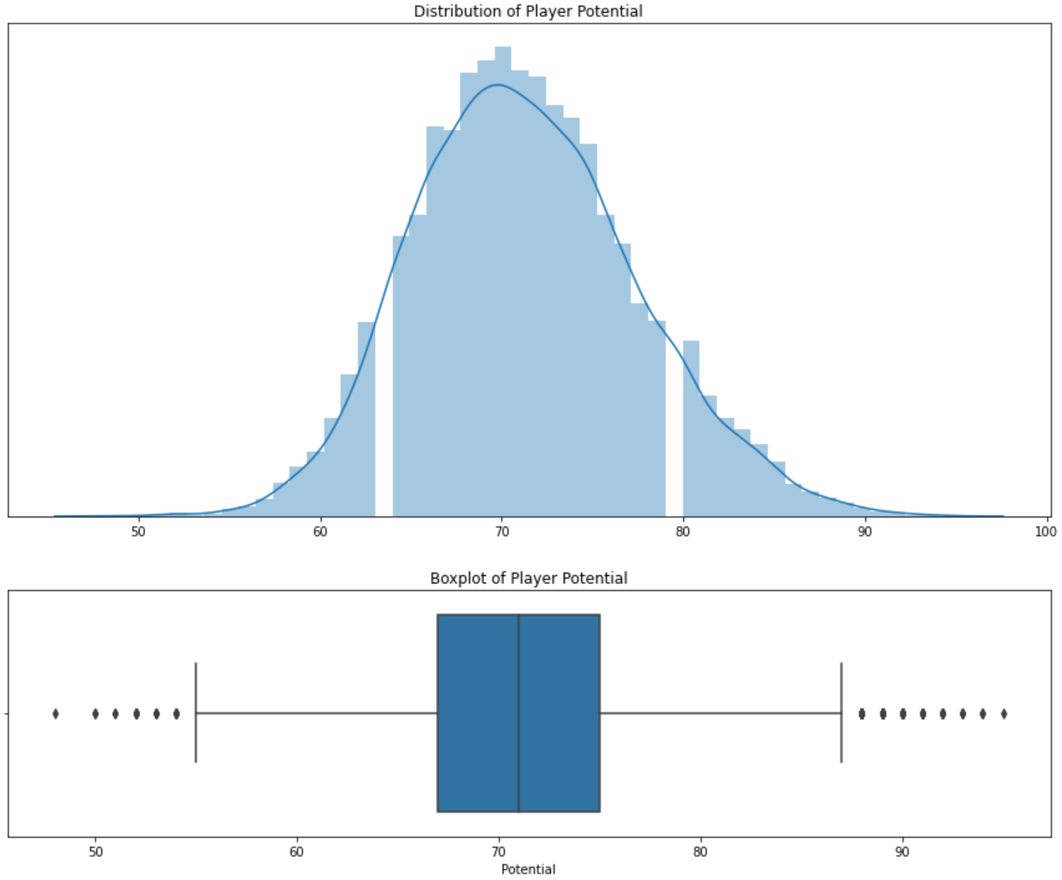



```
Pearson correlation = 0.9384379114329584
R2= 0.8806657136146531
```

Data Visualization

Potential Rating

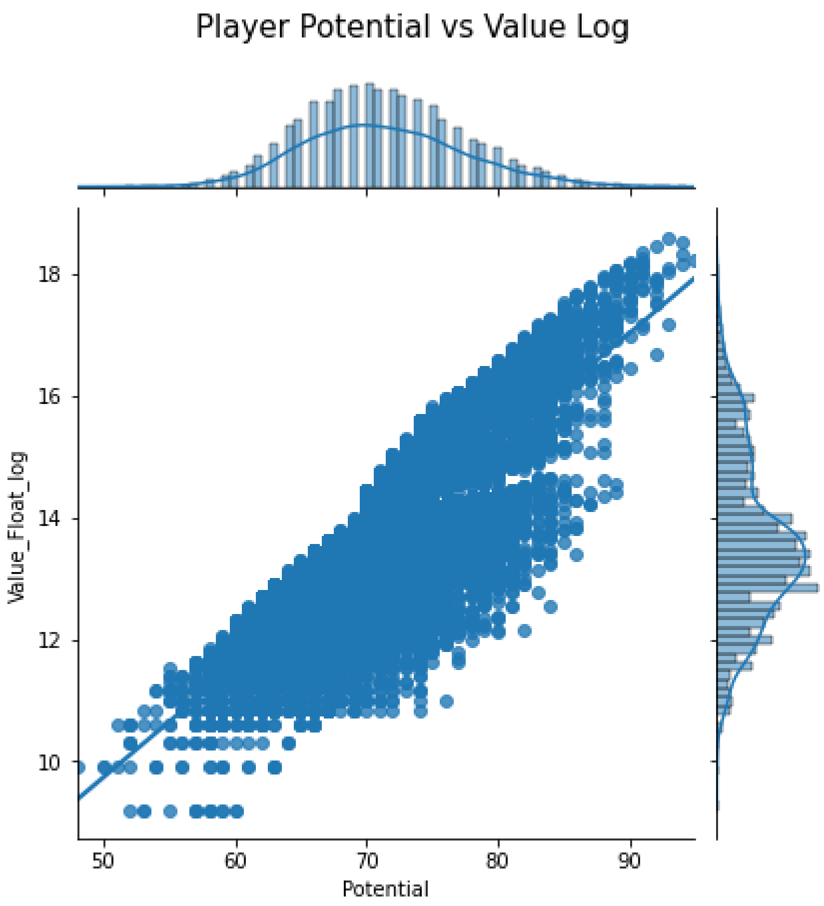
	Potential
count	17907.000000
mean	71.335232
std	6.137251
min	48.000000
25%	67.000000
50%	71.000000
75%	75.000000
max	95.000000



Data Visualization

Potential Rating vs Market Value

If a player has high potential it also has high market values.



Data Visualization

Best Player for each Position

Except for the Aubameyang, all of them are from Europe or America(South).

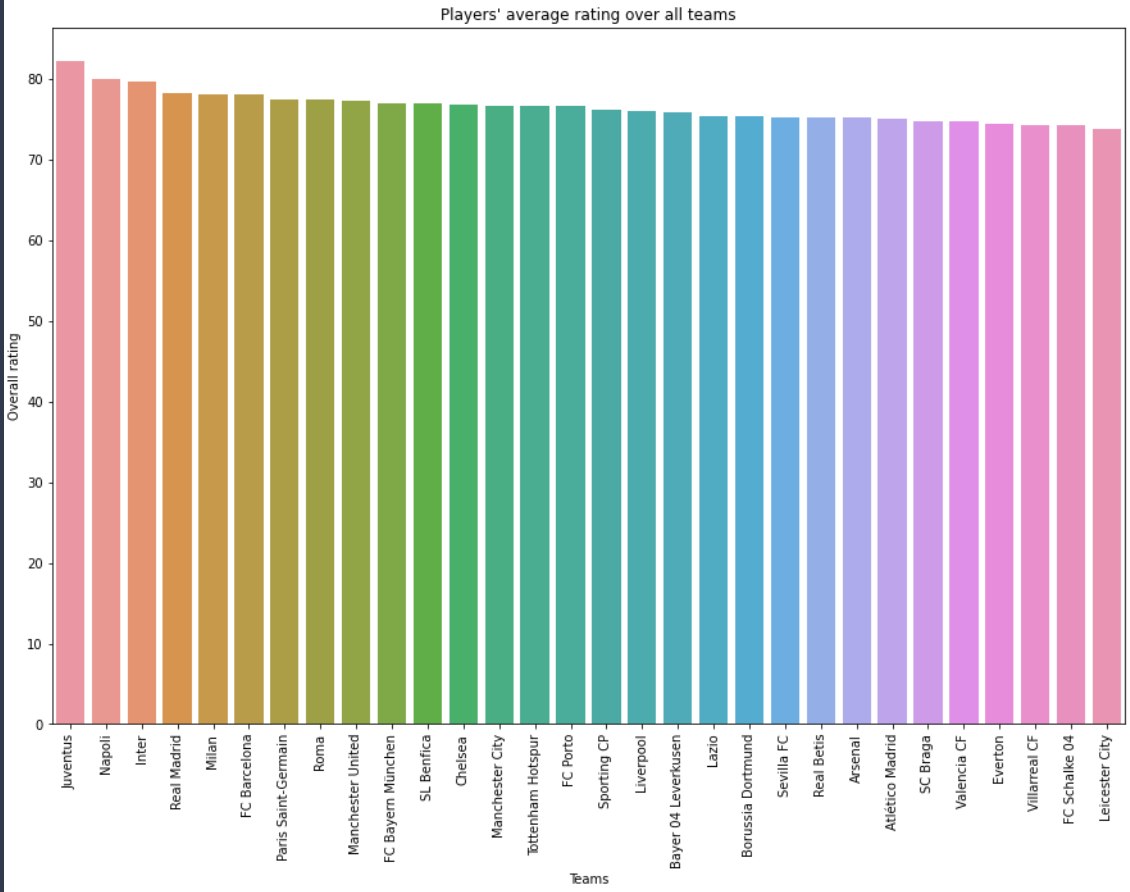
```
f_players.iloc[f_players.groupby(f_players['Positio
                    ascending=False).reset_index()[
```

	Position	Name	Age	Club	Nationality	Overall	Position_classify	Wage	Value
0	ST	Cristiano Ronaldo	33	Juventus	Portugal	94	Attack	€405K	€77M
1	RF	L. Messi	31	FC Barcelona	Argentina	94	Attack	€565K	€110.5M
2	LW	Neymar Jr	26	Paris Saint-Germain	Brazil	92	Midfield	€290K	€118.5M
3	LF	E. Hazard	27	Chelsea	Belgium	91	Attack	€340K	€93M
4	RS	L. Suárez	31	FC Barcelona	Uruguay	91	Attack	€455K	€80M
5	RCM	K. De Bruyne	27	Manchester City	Belgium	91	Midfield	€355K	€102M
6	GK	De Gea	27	Manchester United	Spain	91	GoalKeeper	€260K	€72M
7	RCB	Sergio Ramos	32	Real Madrid	Spain	91	Defence	€380K	€51M
8	CB	D. Godín	32	Atlético Madrid	Uruguay	90	Defence	€125K	€44M
9	LCM	T. Kroos	28	Real Madrid	Germany	90	Midfield	€355K	€76.5M
10	CAM	A. Griezmann	27	Atlético Madrid	France	89	Midfield	€145K	€78M
11	LDM	N. Kanté	27	Chelsea	France	89	Midfield	€225K	€63M
12	LCB	G. Chiellini	33	Juventus	Italy	89	Defence	€215K	€27M
13	CDM	Sergio Busquets	29	FC Barcelona	Spain	89	Midfield	€315K	€51.5M
14	LS	E. Cavani	31	Paris Saint-Germain	Uruguay	89	Attack	€200K	€60M
15	LM	P. Aubameyang	29	Arsenal	Gabon	88	Midfield	€265K	€59M
16	LB	Marcelo	30	Real Madrid	Brazil	88	Defence	€285K	€43M
17	LAM	J. Rodríguez	26	FC Bayern München	Colombia	88	Midfield	€315K	€69.5M
18	RM	K. Mbappé	19	Paris Saint-Germain	France	88	Midfield	€100K	€81M
19	RDM	P. Pogba	25	Manchester United	France	87	Midfield	€210K	€64M
20	RB	Azpilicueta	28	Chelsea	Spain	86	Defence	€175K	€35M
21	CM	Thiago	27	FC Bayern München	Spain	86	Midfield	€130K	€45.5M
22	RW	Bernardo Silva	23	Manchester City	Portugal	86	Midfield	€180K	€59.5M
23	RAM	J. Cuadrado	30	Juventus	Colombia	84	Midfield	€150K	€29.5M
24	CF	Luis Alberto	25	Lazio	Spain	82	Attack	€67K	€28.5M
25	LB	L. Digne	24	Everton	France	80	Defence	€79K	€16M
26	RWB	M. Ginter	24	Borussia Mönchengladbach	Germany	80	Defence	€28K	€15.5M

Data Visualization

Club Teams Overall

This plot show us top 30 teams according to their average rating.

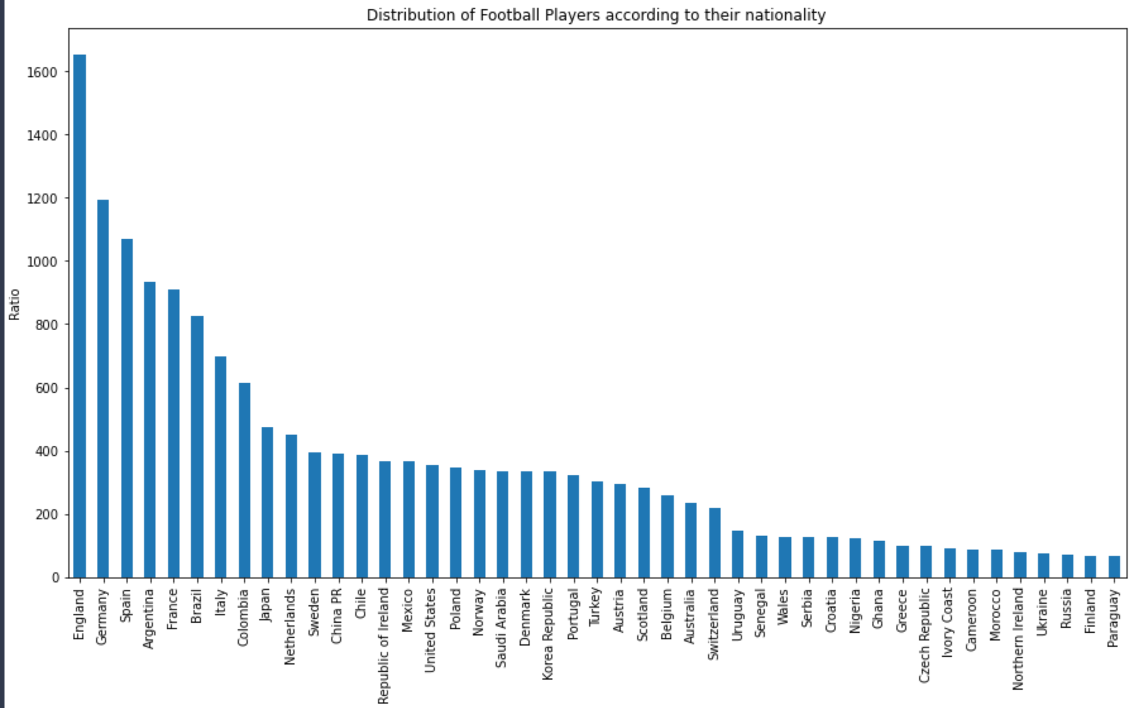


Data Visualization

Nationality

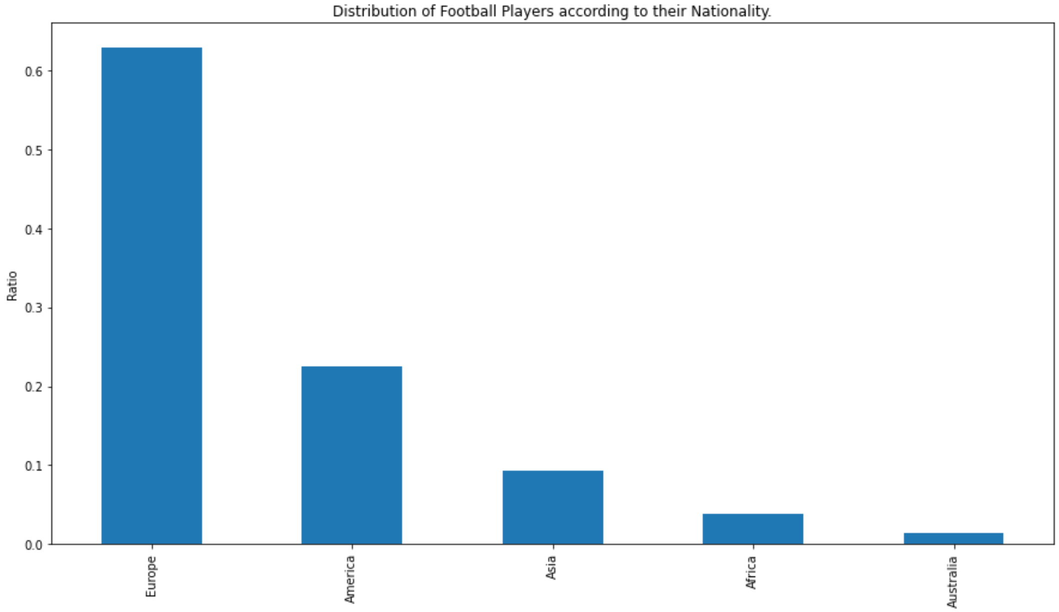
Number of countries = 43

This number is a lot, we can classify country like their region. Such as Europe, America etc.



Data Visualization

Nationality



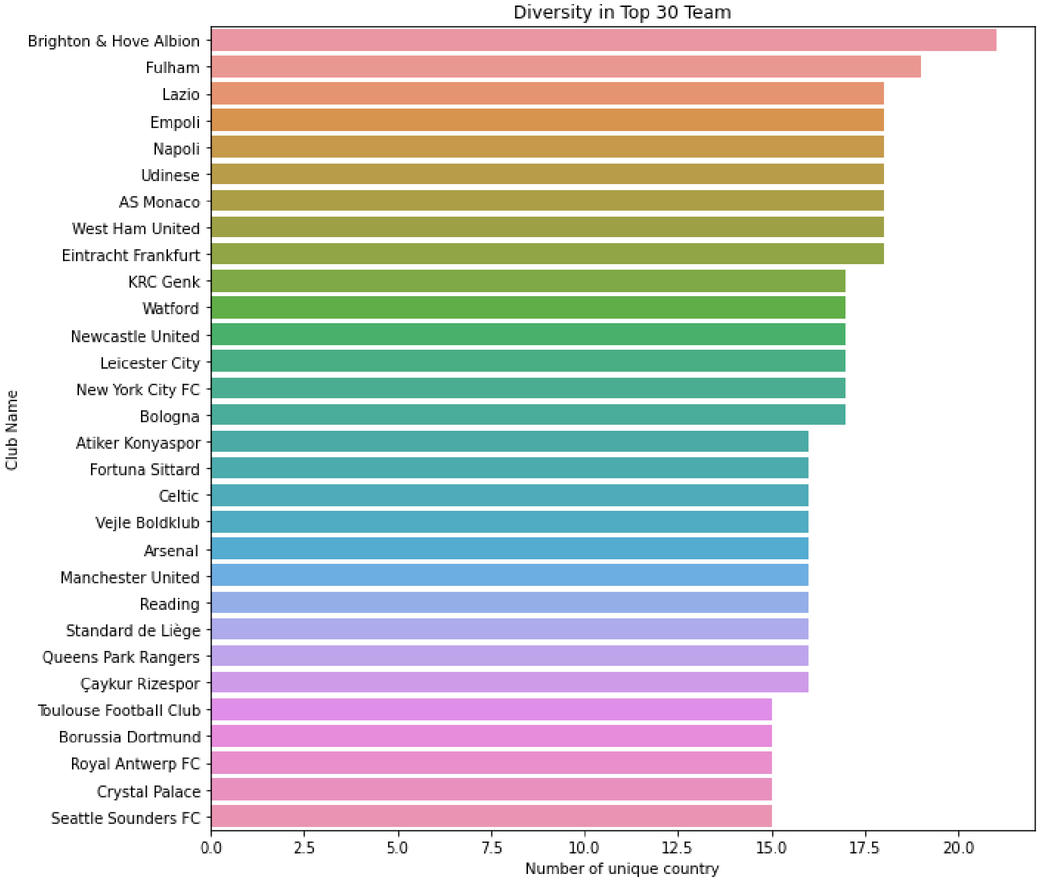

```
def classify_position(nation):
    """This function help to you for classification football player nation."""

    Nation_dictionary= { 'Europe' : ['England','Germany','Spain','France','Italy','Netherlands','Sweden','Republic of Ireland',
                                    'Poland','Norway','Denmark','Portugal','Turkey','Austria','Scotland','Belgium',
                                    'Switzerland','Wales','Serbia','Croatia','Greece','Czech Republic','Northern Ireland',
                                    'Ukraine','Russia','Finland'],
                    'America': ['Argentina','Brazil','Colombia','Chile','Mexico','United States','Uruguay','Paraguay'],
                     'Asia': ['Japan','China PR','Saudi Arabia','Korea Republic'],
                   'Australia':['Australia'],
                        'Africa':['Senegal','Nigeria','Ghana','Ivory Coast','Cameroon','Morocco']}

    for group , nation_list in Nation_dictionary.items():

        if nation in nation_list:
            return group

    return np.nan
```

Data Visualization

Diversity in Club Teams

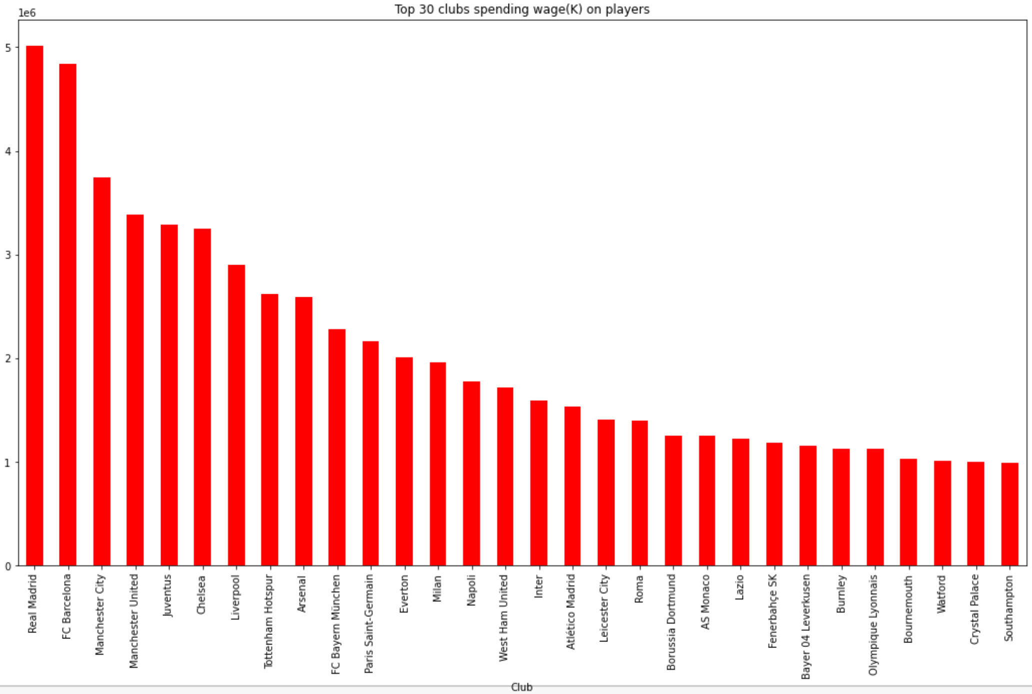
Average of players from different nationalities in teams: **7.86**

Number of unique club teams: **651**



Data Visualization

Wage



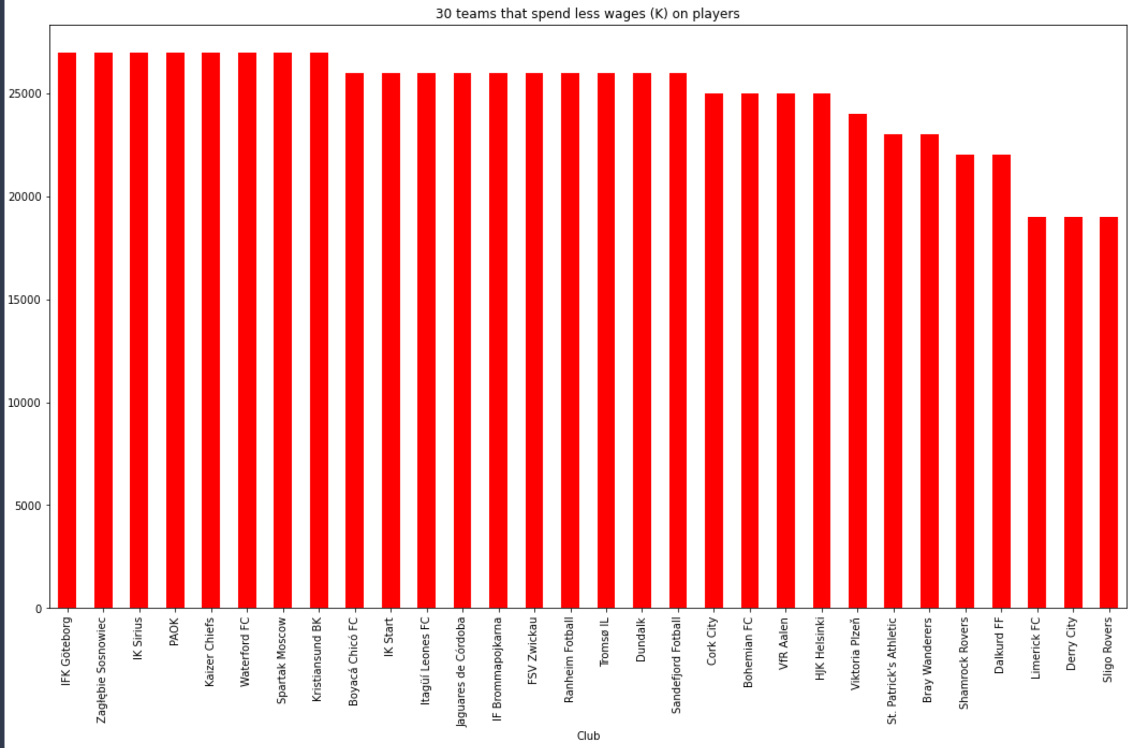

```
club_player = f_players.groupby('Club').sum()
# Number of clubs and average number of players in each club
print('Number of clubs is {}'.format(club_player.shape[0]))
print('Average number players in each club is {}'.format(round(club_player['Age'].mean()/f_players['Age'].mean(),2)))
print('Total Average wage(K) potential ratio is {}'.
      format(round(club_player['Wage_Float'].sum()/(club_player['Potential'].sum()*1000), 2)))
##Wage ve Value çevirirken çarptığım için 1000'e böldük
```

```
Number of clubs is 651
Average number players in each club is 27.51
Total Average wage(K) potential ratio is 0.14
```

Data Visualization

Wage

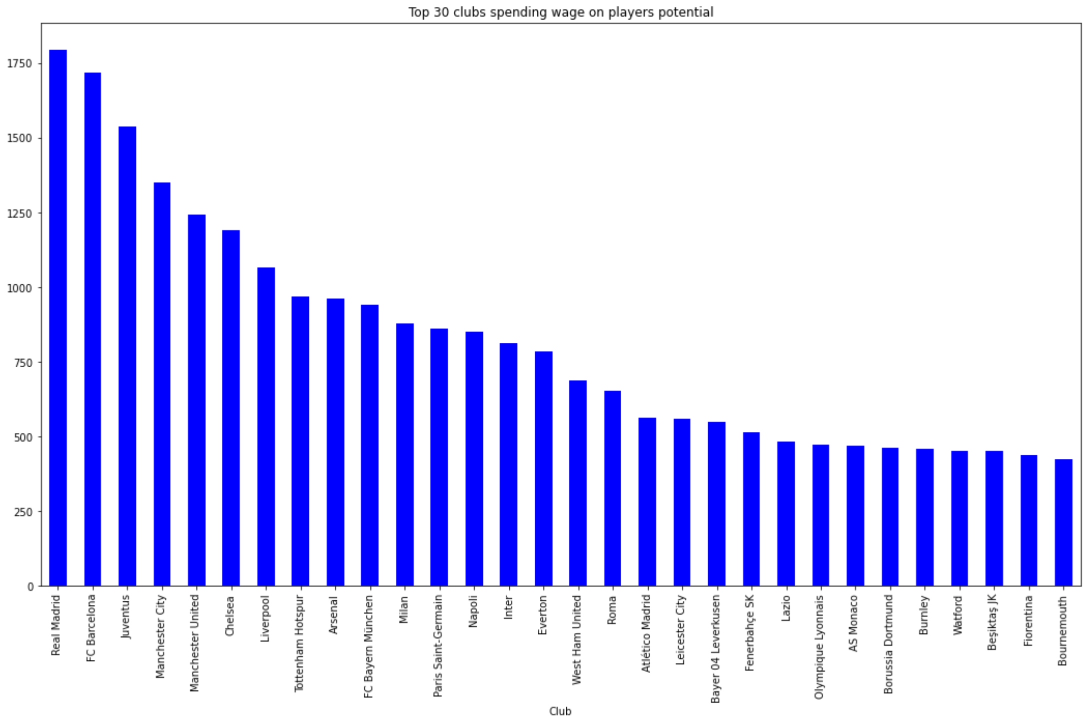
This plot shows us the last 30 teams in spending money for football players.



Data Visualization

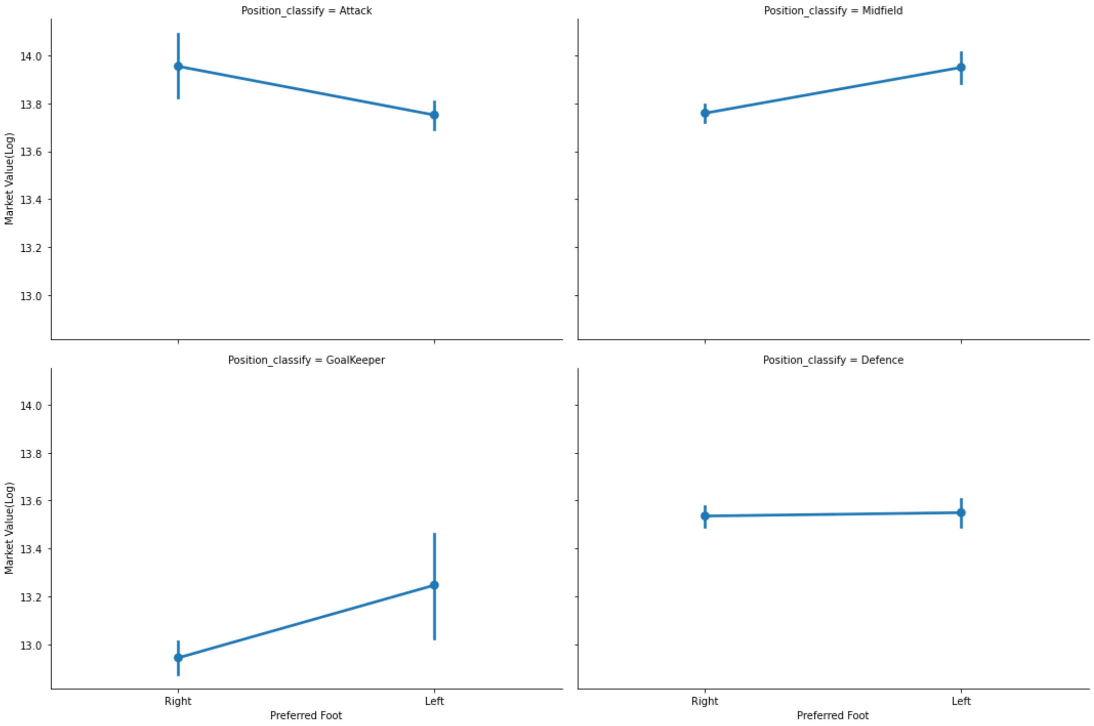
Wage vs Potential

This plot shows us the top 30 teams in spending money for football players.



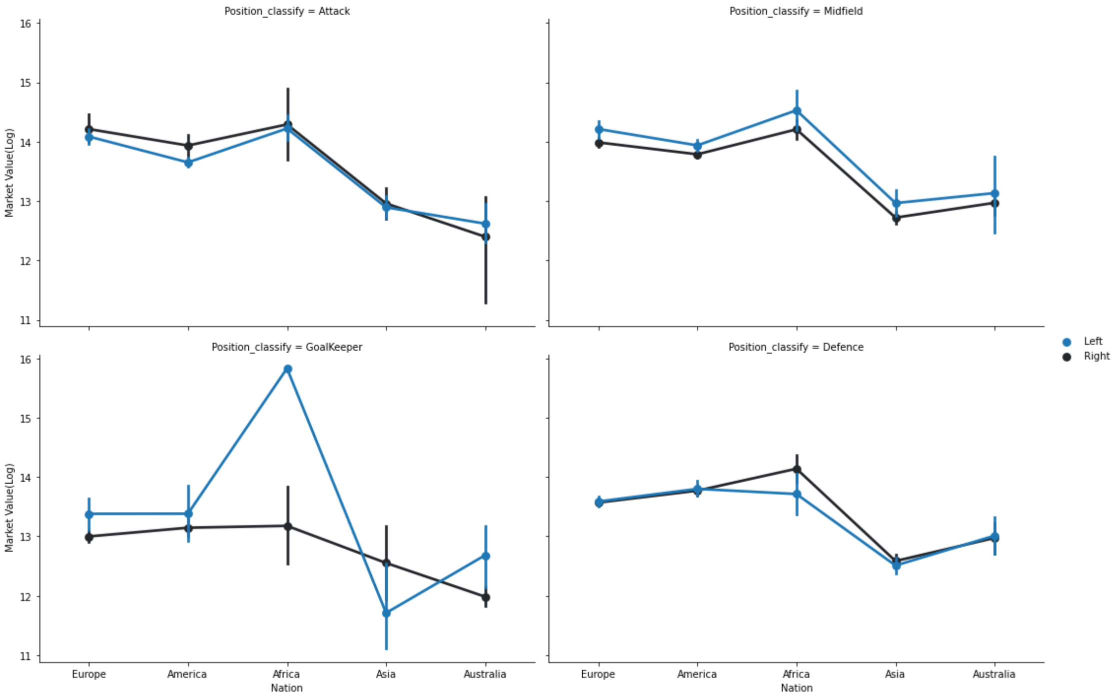
Data Visualization

Preferred Foot vs Value



Data Visualization

Preferred Foot – Value – Nationality

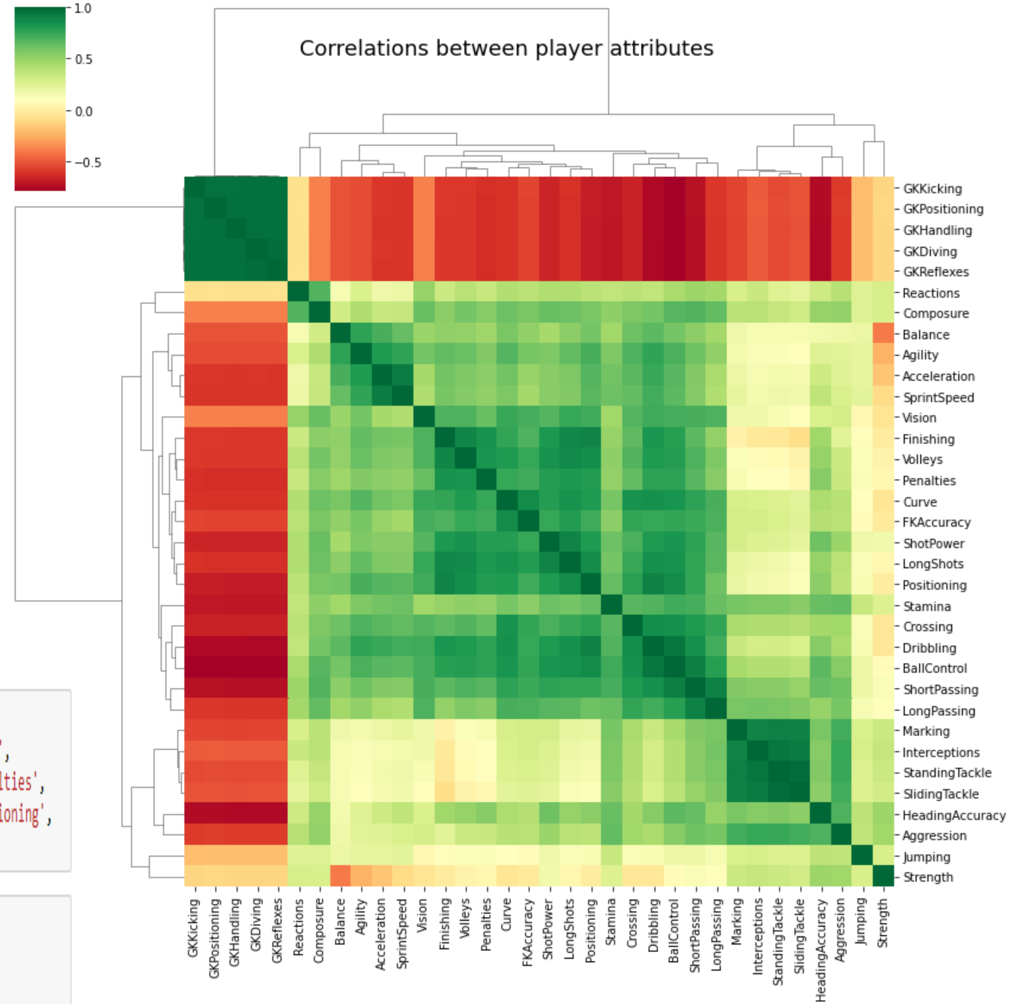


Data Visualization

Correlation

```python
ATTRIBUTES =['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',]
```

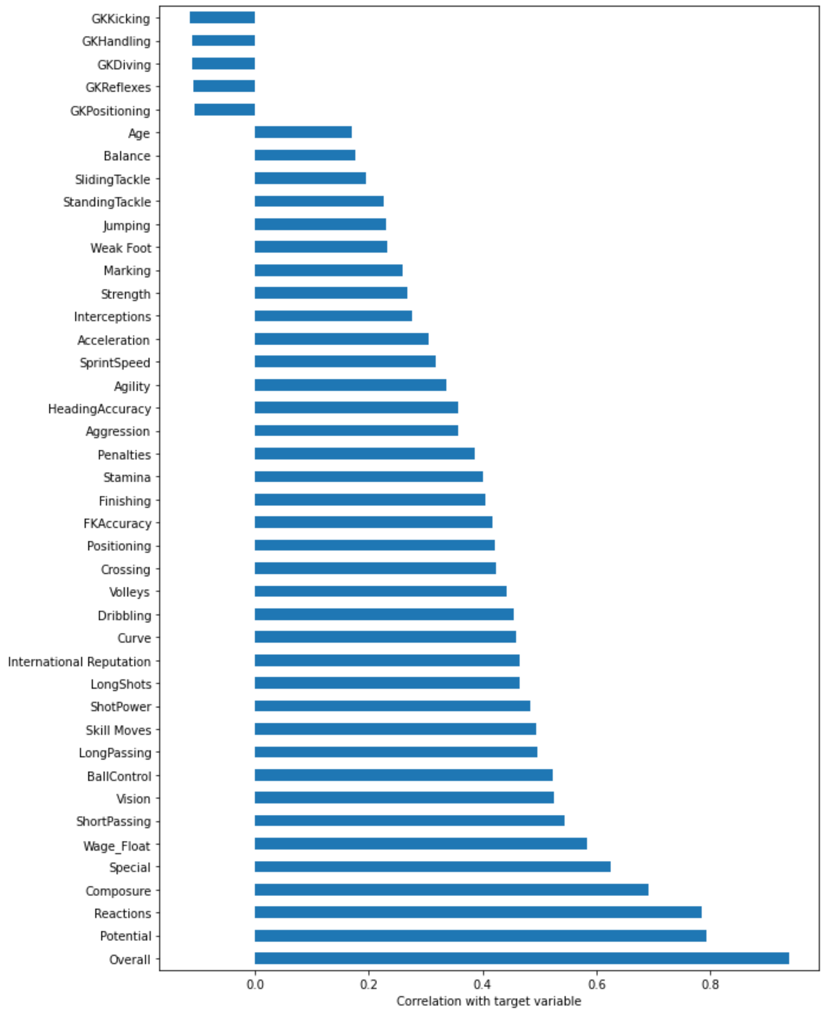
```python
sns.clustermap(f_players[ATTRIBUTES].corr(),cmap='RdYlGn',figsize=(12,12))
plt.suptitle("Correlations between player attributes",fontsize=18,y=0.95)
plt.show()
```



Data Visualization

Feature Importance

Feature Correlations to the target variable, log(Value_Float)



Thank you for listening

Are there any questions?