FIFA 21 PLAYER VALUE PREDICTION AND EDA

TABLE OF CONTENT

1. IMPORTING LIBRARIES

2. LOADING DATASET

3. DATA DESCRIPTION

4. EXPLORATORY DATA ANALYSIS

5. FEATURE ENGINEERING

6. MISSING VALUES

7. DATA VISUALIZATION

8. OUTLIER DETECTION

9. DATA PREPROCESSING

10. MODEL TRAINING AND EVALUATING

11 PCA AND LDA

12. MODEL TUNING

13. CONCLUSION

IMPORTING LIBRARIES

In [ ]:
```
#importing libraries
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
import plotly.express as px
from matplotlib import colors
import plotly.graph_objs as go
from plotly.offline import iplot
from plotly.subplots import make_subplots
```

```
from plotly.offline import init_notebook_mode, iplot
init_notebook_mode(connected=True)
from scipy import stats
from scipy.stats import norm, skew
from sklearn.preprocessing import LabelEncoder, OneHotEncoder
from sklearn.preprocessing import StandardScaler
from sklearn.preprocessing import MinMaxScaler
from sklearn.preprocessing import RobustScaler
from sklearn.model_selection import train_test_split,KFold
from sklearn.model_selection import GridSearchCV
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import cross_validate
from sklearn.metrics import confusion_matrix, accuracy_score, classification_report,roc_auc_score,roc_curve
from sklearn.metrics import mean_absolute_error,mean_squared_error,r2_score,f1_score,precision_score,recall_sco
from sklearn.linear_model import LinearRegression
from sklearn.svm import SVR
from sklearn.tree import DecisionTreeRegressor
from sklearn.ensemble import RandomForestRegressor,GradientBoostingRegressor
from sklearn.neighbors import KNeighborsRegressor
from sklearn.neural_network import MLPRegressor
from sklearn.linear_model import Ridge,Lasso,ElasticNet
from sklearn.gaussian_process import GaussianProcessRegressor
import warnings
warnings.filterwarnings("ignore")
warnings.simplefilter(action='ignore', category=FutureWarning)
pd.set_option('display.max_columns', None)
pd.set_option('display.max_rows', None)
```

LOADING DATASET

In [ ]:
```
data=pd.read_csv("players_21.csv")
df=data.copy()
df.drop(["ls","st","rb","rcb","cb","lcb","lb","rwb","rdm","cdm","ldm","lwb","rm","rcm","cm","rs","lcm","lm","ra
df.head().style.set_properties(**{'background-color': 'black',
                           'color': 'lawngreen',
                           'border-color': 'white'})
```

Out[ ]:

	sofifa_id	player_url	short_name	long_name	age	dob	height_cm	weight_kg	nationality	club_name	leagu
0	158023	https://sofifa.com/player/158023/lionel-messi/210002	L. Messi	Lionel Andrés Messi Cuccittini	33	1987-06-24	170	72	Argentina	FC Barcelona	Spain
1	20801	https://sofifa.com/player/20801/c-ronaldo-dos-santos-aveiro/210002	Cristiano Ronaldo	Cristiano Ronaldo dos Santos Aveiro	35	1985-02-05	187	83	Portugal	Juventus	Italia
2	200389	https://sofifa.com/player/200389/jan-oblak/210002	J. Oblak	Jan Oblak	27	1993-01-07	188	87	Slovenia	Atlético Madrid	Spain
3	188545	https://sofifa.com/player/188545/robert-lewandowski/210002	R. Lewandowski	Robert Lewandowski	31	1988-08-21	184	80	Poland	FC Bayern München	G Bu
4	190871	https://sofifa.com/player/190871/neymar-da-silva-santos-jr/210002	Neymar Jr	Neymar da Silva Santos Júnior	28	1992-02-05	175	68	Brazil	Paris Saint-Germain	Fren

DATA DESCRIPTION

DATA CONTENT:

- Every player available in FIFA 21
- 100+ attributes
- URL of the scraped players
- Player positions, with the role in the club and in the national team
- Player attributes with statistics as Attacking, Skills, Defense, Mentality, GK Skills, etc.
- Player personal data like Nationality, Club, DateOfBirth, Wage, Salary, etc.

EXPLORATORY ANALYSIS

In [ ]:
```
print('Shape of the data:',df.shape)
print("*****************************")
print('Columns of the data:',df.columns)
print("*****************************")
print('Number of unique values:',df.nunique())
print("*****************************")
print('Number of duplicated values:',df.duplicated().sum())
```

```
Shape of the data: (18944, 80)
*****************************
Columns of the data: Index(['sofifa_id', 'player_url', 'short_name', 'long_name', 'age', 'dob',
       'height_cm', 'weight_kg', 'nationality', 'club_name', 'league_name',
       'league_rank', 'overall', 'potential', 'value_eur', 'wage_eur',
       'player_positions', 'preferred_foot', 'international_reputation',
       'weak_foot', 'skill_moves', 'work_rate', 'body_type', 'real_face',
       'release_clause_eur', 'player_tags', 'team_position',
       'team_jersey_number', 'loaned_from', 'joined', 'contract_valid_until',
       'nation_position', 'nation_jersey_number', 'pace', 'shooting',
       'passing', 'dribbling', 'defending', 'physic', 'gk_diving',
       'gk_handling', 'gk_kicking', 'gk_reflexes', 'gk_speed',
       'gk_positioning', 'player_traits', 'attacking_crossing',
       'attacking_finishing', 'attacking_heading_accuracy',
       'attacking_short_passing', 'attacking_volleys', 'skill_dribbling',
       'skill_curve', 'skill_fk_accuracy', 'skill_long_passing',
       'skill_ball_control', 'movement_acceleration', 'movement_sprint_speed',
       'movement_agility', 'movement_reactions', 'movement_balance',
       'power_shot_power', 'power_jumping', 'power_stamina', 'power_strength',
       'power_long_shots', 'mentality_aggression', 'mentality_interceptions',
       'mentality_positioning', 'mentality_vision', 'mentality_penalties',
       'mentality_composure', 'defending_marking', 'defending_standing_tackle',
       'defending_sliding_tackle', 'goalkeeping_diving',
       'goalkeeping_handling', 'goalkeeping_kicking',
       'goalkeeping_positioning', 'goalkeeping_reflexes'],
      dtype='object')
*****************************
Number of unique values: sofifa_id                   18944
player_url                  18944
short_name                  17884
long_name                   18896
age                            29
dob                          6236
height_cm                      50
weight_kg                      56
nationality                   162
club_name                     681
league_name                    52
league_rank                     4
overall                        47
potential                      48
value_eur                     214
wage_eur                      142
player_positions              611
preferred_foot                  2
international_reputation        5
weak_foot                       5
skill_moves                     5
work_rate                       9
body_type                     118
real_face                       2
release_clause_eur           1225
player_tags                    75
team_position                  29
team_jersey_number             99
loaned_from                   278
joined                       1822
contract_valid_until            9
nation_position                26
nation_jersey_number           25
```

```
pace                            70
shooting                        77
passing                         68
dribbling                       69
defending                       77
physic                          63
gk_diving                       46
gk_handling                     47
gk_kicking                      52
gk_reflexes                     47
gk_speed                        52
gk_positioning                  53
player_traits                  915
attacking_crossing              89
attacking_finishing             93
attacking_heading_accuracy      89
attacking_short_passing         86
attacking_volleys               88
skill_dribbling                 91
skill_curve                     91
skill_fk_accuracy               90
skill_long_passing              86
skill_ball_control              91
movement_acceleration           85
movement_sprint_speed           84
movement_agility                81
movement_reactions              69
movement_balance                83
power_shot_power                76
power_jumping                   75
power_stamina                   85
power_strength                  77
power_long_shots                91
mentality_aggression            88
mentality_interceptions         89
mentality_positioning           94
mentality_vision                86
mentality_penalties             87
mentality_composure             85
defending_marking                0
defending_standing_tackle       87
defending_sliding_tackle        85
goalkeeping_diving              70
goalkeeping_handling            71
goalkeeping_kicking             80
goalkeeping_positioning         77
goalkeeping_reflexes            71
dtype: int64
**************************
Number of duplicated values: 0
```

In [ ]:
```
df.dtypes
```

Out[ ]:
```
sofifa_id                      int64
player_url                    object
short_name                    object
long_name                     object
age                            int64
dob                           object
height_cm                      int64
weight_kg                      int64
nationality                   object
club_name                     object
league_name                   object
league_rank                  float64
overall                        int64
potential                      int64
value_eur                      int64
wage_eur                       int64
player_positions              object
preferred_foot                object
international_reputation       int64
weak_foot                      int64
skill_moves                    int64
work_rate                     object
body_type                     object
real_face                     object
release_clause_eur           float64
player_tags                   object
team_position                 object
team_jersey_number           float64
loaned_from                   object
joined                        object
contract_valid_until         float64
nation_position               object
nation_jersey_number         float64
pace                         float64
shooting                     float64
passing                      float64
dribbling                    float64
defending                    float64
physic                       float64
gk_diving                    float64
gk_handling                  float64
gk_kicking                   float64
gk_reflexes                  float64
gk_speed                     float64
gk_positioning               float64
player_traits                 object
attacking_crossing             int64
attacking_finishing            int64
attacking_heading_accuracy     int64
attacking_short_passing        int64
attacking_volleys              int64
skill_dribbling                int64
skill_curve                    int64
skill_fk_accuracy              int64
skill_long_passing             int64
skill_ball_control             int64
movement_acceleration          int64
movement_sprint_speed          int64
movement_agility               int64
movement_reactions             int64
movement_balance               int64
power_shot_power               int64
power_jumping                  int64
power_stamina                  int64
power_strength                 int64
power_long_shots               int64
mentality_aggression           int64
mentality_interceptions        int64
mentality_positioning          int64
mentality_vision               int64
mentality_penalties            int64
mentality_composure            int64
defending_marking            float64
defending_standing_tackle      int64
defending_sliding_tackle       int64
goalkeeping_diving             int64
goalkeeping_handling           int64
goalkeeping_kicking            int64
goalkeeping_positioning        int64
goalkeeping_reflexes           int64
dtype: object
```

In [ ]:
```
# describe the data
df.describe()
```

Out[ ]:

	sofifa_id	age	height_cm	weight_kg	league_rank	overall	potential	value_eur_m	wage_eur_m	inte
count	18944.000000	18944.000000	18944.000000	18944.000000	18944.000000	18944.000000	18944.000000	18944.000000	18944.000000	
mean	226242.402872	25.225823	181.190773	75.016892	1.357070	65.677787	71.086729	2.224813	0.416441	
std	27171.091056	4.697354	6.825672	7.057140	0.734923	7.002278	6.109985	5.102486	0.943429	
min	41.000000	16.000000	155.000000	50.000000	1.000000	47.000000	47.000000	0.000000	0.000000	
25%	210030.500000	21.000000	176.000000	70.000000	1.000000	61.000000	67.000000	0.300000	0.048000	
50%	232314.500000	25.000000	181.000000	75.000000	1.000000	66.000000	71.000000	0.650000	0.144000	
75%	246760.250000	29.000000	186.000000	80.000000	1.000000	70.000000	75.000000	1.800000	0.336000	
max	258970.000000	53.000000	206.000000	110.000000	4.000000	93.000000	95.000000	105.500000	26.880000	

In [ ]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 18944 entries, 0 to 18943
Data columns (total 80 columns):
 #   Column                       Non-Null Count  Dtype  
---  ------                       --------------  -----  
 0   sofifa_id                    18944 non-null  int64  
 1   player_url                   18944 non-null  object 
 2   short_name                   18944 non-null  object 
 3   long_name                    18944 non-null  object 
 4   age                          18944 non-null  int64  
 5   dob                          18944 non-null  object 
 6   height_cm                    18944 non-null  int64  
 7   weight_kg                    18944 non-null  int64  
 8   nationality                  18944 non-null  object 
 9   club_name                    18719 non-null  object 
 10  league_name                  18719 non-null  object 
 11  league_rank                  18719 non-null  float64
 12  overall                      18944 non-null  int64  
 13  potential                    18944 non-null  int64  
 14  value_eur                    18944 non-null  int64  
 15  wage_eur                     18944 non-null  int64  
 16  player_positions             18944 non-null  object 
 17  preferred_foot               18944 non-null  object 
 18  international_reputation     18944 non-null  int64  
 19  weak_foot                    18944 non-null  int64  
 20  skill_moves                  18944 non-null  int64  
 21  work_rate                    18944 non-null  object 
 22  body_type                    18944 non-null  object 
 23  real_face                    18944 non-null  object 
 24  release_clause_eur           17949 non-null  float64
 25  player_tags                  1408 non-null   object 
 26  team_position                18719 non-null  object 
 27  team_jersey_number           18719 non-null  float64
 28  loaned_from                  758 non-null    object 
 29  joined                       17961 non-null  object 
 30  contract_valid_until         18719 non-null  float64
 31  nation_position              1127 non-null   object 
 32  nation_jersey_number         1127 non-null   float64
 33  pace                         16861 non-null  float64
 34  shooting                     16861 non-null  float64
 35  passing                      16861 non-null  float64
 36  dribbling                    16861 non-null  float64
 37  defending                    16861 non-null  float64
 38  physic                       16861 non-null  float64
 39  gk_diving                    2083 non-null   float64
 40  gk_handling                  2083 non-null   float64
 41  gk_kicking                   2083 non-null   float64
 42  gk_reflexes                  2083 non-null   float64
 43  gk_speed                     2083 non-null   float64
 44  gk_positioning               2083 non-null   float64
 45  player_traits                8315 non-null   object 
 46  attacking_crossing           18944 non-null  int64  
 47  attacking_finishing          18944 non-null  int64  
 48  attacking_heading_accuracy   18944 non-null  int64  
 49  attacking_short_passing      18944 non-null  int64  
 50  attacking_volleys            18944 non-null  int64  
 51  skill_dribbling              18944 non-null  int64  
 52  skill_curve                  18944 non-null  int64  
 53  skill_fk_accuracy            18944 non-null  int64  
 54  skill_long_passing           18944 non-null  int64  
 55  skill_ball_control           18944 non-null  int64  
 56  movement_acceleration        18944 non-null  int64  
 57  movement_sprint_speed        18944 non-null  int64  
 58  movement_agility             18944 non-null  int64  
 59  movement_reactions           18944 non-null  int64  
 60  movement_balance             18944 non-null  int64  
 61  power_shot_power             18944 non-null  int64  
 62  power_jumping                18944 non-null  int64  
 63  power_stamina                18944 non-null  int64  
 64  power_strength               18944 non-null  int64  
 65  power_long_shots             18944 non-null  int64  
 66  mentality_aggression         18944 non-null  int64  
 67  mentality_interceptions      18944 non-null  int64  
 68  mentality_positioning        18944 non-null  int64  
 69  mentality_vision             18944 non-null  int64  
 70  mentality_penalties          18944 non-null  int64  
 71  mentality_composure          18944 non-null  int64  
 72  defending_marking            0 non-null      float64
 73  defending_standing_tackle    18944 non-null  int64  
 74  defending_sliding_tackle     18944 non-null  int64  
 75  goalkeeping_diving           18944 non-null  int64  
 76  goalkeeping_handling         18944 non-null  int64  
 77  goalkeeping_kicking          18944 non-null  int64  
 78  goalkeeping_positioning      18944 non-null  int64  
 79  goalkeeping_reflexes         18944 non-null  int64  
dtypes: float64(18), int64(44), object(18)
memory usage: 11.6+ MB
```

In [ ]:
```
import researchpy as rp
```

```
num_cols = df.select_dtypes(include = np.number).columns
rp.summary_cont(df[num_cols]).style.set_properties(**{'background-color': 'black',
                            'color': 'lawngreen',
                            'border-color': 'white'})
```

Out[ ]:

	Variable	N	Mean	SD	SE	95% Conf.	Interval
0	sofifa_id	18944.000000	226242.402900	27171.091100	197.410900	225855.459800	226629.345900
1	age	18944.000000	25.225800	4.697400	0.034100	25.158900	25.292700
2	height_cm	18944.000000	181.190800	6.825700	0.049600	181.093600	181.288000
3	weight_kg	18944.000000	75.016900	7.057100	0.051300	74.916400	75.117400
4	league_rank	18719.000000	1.357100	0.739300	0.005400	1.346500	1.367700
5	overall	18944.000000	65.677800	7.002300	0.050900	65.578100	65.777500
6	potential	18944.000000	71.086700	6.110000	0.044400	70.999700	71.173700
7	value_eur	18944.000000	2224813.291800	5102485.993600	37071.996800	2152148.870400	2297477.713200
8	wage_eur	18944.000000	8675.852500	19654.774900	142.801300	8395.949200	8955.755800
9	international_reputation	18944.000000	1.091800	0.361800	0.002600	1.086700	1.097000
10	weak_foot	18944.000000	2.936600	0.667100	0.004800	2.927100	2.946100
11	skill_moves	18944.000000	2.363000	0.766500	0.005600	2.352100	2.373900
12	release_clause_eur	17949.000000	4296353.055900	10059682.784200	75086.897000	4149175.516800	4443530.595000
13	team_jersey_number	18719.000000	20.589700	17.057600	0.124700	20.345300	20.834000
14	contract_valid_until	18719.000000	2021.983300	1.260400	0.009200	2021.965300	2022.001400
15	nation_jersey_number	1127.000000	12.029300	6.674600	0.198800	11.639200	12.419400
16	pace	16861.000000	67.668100	10.984900	0.084600	67.502300	67.833900
17	shooting	16861.000000	52.275000	13.991400	0.107800	52.063800	52.486200
18	passing	16861.000000	57.139400	10.273700	0.079100	56.984400	57.294500
19	dribbling	16861.000000	62.455400	10.048700	0.077400	62.303700	62.607100
20	defending	16861.000000	51.316300	16.405200	0.126300	51.068700	51.563900
21	physic	16861.000000	64.459000	9.746700	0.075100	64.311900	64.606100
22	gk_diving	2083.000000	65.164200	7.710500	0.168900	64.832900	65.495500
23	gk_handling	2083.000000	62.889100	7.328900	0.160600	62.574200	63.204000
24	gk_kicking	2083.000000	61.719600	7.575200	0.166000	61.394100	62.045100
25	gk_reflexes	2083.000000	66.108000	8.126600	0.178100	65.758800	66.457200
26	gk_speed	2083.000000	37.203600	10.714500	0.234800	36.743200	37.663900
27	gk_positioning	2083.000000	63.171900	8.566000	0.187700	62.803800	63.539900
28	attacking_crossing	18944.000000	49.612900	18.153100	0.131900	49.354300	49.871400
29	attacking_finishing	18944.000000	45.796100	19.592300	0.142300	45.517100	46.075100
30	attacking_heading_accuracy	18944.000000	51.874600	17.321700	0.125900	51.627900	52.121300
31	attacking_short_passing	18944.000000	58.705400	14.572400	0.105900	58.497900	58.913000
32	attacking_volleys	18944.000000	42.669600	17.661600	0.128300	42.418000	42.921100
33	skill_dribbling	18944.000000	55.546400	18.787100	0.136500	55.278900	55.813900
34	skill_curve	18944.000000	47.187300	18.231200	0.132500	46.927700	47.446900
35	skill_fk_accuracy	18944.000000	42.356200	17.239600	0.125300	42.110600	42.601700
36	skill_long_passing	18944.000000	52.653000	15.205300	0.110500	52.436500	52.869600
37	skill_ball_control	18944.000000	58.482800	16.598000	0.120600	58.246400	58.719200
38	movement_acceleration	18944.000000	64.285100	14.926500	0.108400	64.072500	64.497700
39	movement_sprint_speed	18944.000000	64.326500	14.689400	0.106700	64.117300	64.535700
40	movement_agility	18944.000000	63.333200	14.625200	0.106300	63.124900	63.541500
41	movement_reactions	18944.000000	61.609400	9.113000	0.066200	61.479700	61.739200
42	movement_balance	18944.000000	63.918500	14.075200	0.102300	63.718100	64.119000
43	power_shot_power	18944.000000	57.752800	13.330600	0.096900	57.563000	57.942600
44	power_jumping	18944.000000	64.590500	11.888500	0.086400	64.421200	64.759800
45	power_stamina	18944.000000	62.596000	15.878800	0.115400	62.369900	62.822100
46	power_strength	18944.000000	64.742900	12.519800	0.091000	64.564600	64.921200
47	power_long_shots	18944.000000	46.758800	19.305300	0.140300	46.483800	47.033700
48	mentality_aggression	18944.000000	55.488900	17.203000	0.125000	55.243900	55.733900
49	mentality_interceptions	18944.000000	46.250000	20.738100	0.150700	45.954700	46.545300

50	mentality_positioning	18944.000000	50.255800	19.443700	0.141300	49.978900	50.532700
51	mentality_vision	18944.000000	53.828200	13.724100	0.099700	53.632800	54.023700
52	mentality_penalties	18944.000000	48.050400	15.671700	0.113900	47.827200	48.273600
53	mentality_composure	18944.000000	57.978700	12.118400	0.088000	57.806100	58.151300
54	defending_marking	0.000000	nan	nan	nan	nan	nan
55	defending_standing_tackle	18944.000000	47.581800	21.402500	0.155500	47.277000	47.886600
56	defending_sliding_tackle	18944.000000	45.546500	20.954000	0.152200	45.248100	45.844900
57	goalkeeping_diving	18944.000000	16.446100	17.577300	0.127700	16.195700	16.696400
58	goalkeeping_handling	18944.000000	16.236500	16.845500	0.122400	15.996600	16.476400
59	goalkeeping_kicking	18944.000000	16.103400	16.519400	0.120000	15.868100	16.338600
60	goalkeeping_positioning	18944.000000	16.226000	17.017300	0.123600	15.983600	16.468300
61	goalkeeping_reflexes	18944.000000	16.551300	17.878100	0.129900	16.296700	16.805900

FEATURE ENGINEERING

In [ ]:

```
# convert object features to datetime

from datetime import datetime as dt

df["dob"]=pd.to_datetime(df["dob"])
df["joined"]=pd.to_datetime(df["joined"])

df["born_year"]=df["dob"].dt.year

# converting value features:

df["value_eur"]=df["value_eur"]/1000000
df["wage_eur"]=(df["wage_eur"]*48)/1000000
df["release_clause_eur"]=df["release_clause_eur"]/1000000

#RENAME THE  COLUMNS:
df.rename(columns={"value_eur":"value_eur_m","wage_eur":"wage_eur_m","release_clause_eur":"release_clause_eur_m

# CREATE NEW FEATURES:

# league value
df["league_value_B"]=df.groupby("league_name")["value_eur_m"].transform("sum")/1000

# position_area(attack,defense,midfield,goalkeeper):

df["team_position"]=df["team_position"].apply(lambda x: "attack" if x in ["LS","ST","RS","LW","LF","CF","RF","R
                                              else ("defense" if x in ["LWB","LB","LCB","CB","RCB","RB"] else
                                                    ("midfield" if x in ["LDM","CDM","RDM","LM","LCM","CM","RCM
                                                     else("goalkeeper" if x in ["GK"] else "none"))))


# loyalty feature: how many years the player has been in the club
df["loyalty"]=2020-df["joined"].dt.year

# body type availablity

df["body_type"]=df["body_type"].apply(lambda x: "not famous" if x.startswith("PLAYER_BODY_TYPE_") else "famous"


# number of player traits:

df["player_traits"]=df["player_traits"].apply(lambda x: len(x.split(",")) if type(x)==str else 0)

# team quelity:

vf=df.groupby("club_name")[["value_eur_m"]].sum().sort_values(by="value_eur_m",ascending=False).head(500)

df["team_quality"]=df["club_name"].apply(lambda x: "hight_quality" if x in vf.index else "low_quality")


df.head()
```

Out[ ]:

	sofifa_id	player_url	short_name	long_name	age	dob	height_cm	weight_kg	nationality	club_name	leagu
0	158023	https://sofifa.com/player/158023/lionel-messi/...	L. Messi	Lionel Andrés Messi Cuccittini	33	1987-06-24	170	72	Argentina	FC Barcelona	Spair
1	20801	https://sofifa.com/player/20801/c-ronaldo-dos-...	Cristiano Ronaldo	Cristiano Ronaldo dos Santos Aveiro	35	1985-02-05	187	83	Portugal	Juventus	Italia
2	200389	https://sofifa.com/player/200389/jan-oblak/210002	J. Oblak	Jan Oblak	27	1993-01-07	188	87	Slovenia	Atlético Madrid	Spair
3	188545	https://sofifa.com/player/188545/robert-lewand...	R. Lewandowski	Robert Lewandowski	31	1988-08-21	184	80	Poland	FC Bayern München	G Bu
4	190871	https://sofifa.com/player/190871/neymar-da-sil...	Neymar Jr	Neymar da Silva Santos Júnior	28	1992-02-05	175	68	Brazil	Paris Saint-Germain	Frer

MISSING VALUES

In [ ]:
```
df.tail().style.set_properties(**{'background-color': 'black',
                                  'color': 'lawngreen',
                                  'border-color': 'white'})
```

Out[ ]:

	sofifa_id	player_url	short_name	long_name	age	dob	height_cm	weight_kg	nationality	club_nam
18939	256679	https://sofifa.com/player/256679/kevin-angulo/210002	K. Angulo	Kevin Angulo	24	1996-04-13 00:00:00	176	73	Colombia	América C
18940	257710	https://sofifa.com/player/257710/mengxuan-zhang/210002	Zhang Mengxuan	Mengxuan Zhang	21	1999-04-26 00:00:00	177	70	China PR	Chongqi Dango Lifan F SWM Tea
18941	250989	https://sofifa.com/player/250989/zhenghao-wang/210002	Wang Zhenghao	王政豪	20	2000-06-28 00:00:00	185	74	China PR	Tian TEDA F
18942	257697	https://sofifa.com/player/257697/zitong-chen/210002	Chen Zitong	Zitong Chen	23	1997-02-20 00:00:00	186	80	China PR	Shijiazhua Ever Brig F.
18943	257936	https://sofifa.com/player/257936/yue-song/210002	Song Yue	Yue Song	28	1991-11-20 00:00:00	185	79	China PR	Tian TEDA F

In [ ]:
```
# check the missing values
df.isnull().sum()
```

Out[ ]:

```
sofifa_id                      0
player_url                     0
short_name                     0
long_name                      0
age                            0
dob                            0
height_cm                      0
weight_kg                      0
nationality                    0
club_name                    225
league_name                  225
league_rank                  225
overall                        0
potential                      0
value_eur_m                    0
wage_eur_m                     0
player_positions               0
preferred_foot                 0
international_reputation       0
weak_foot                      0
skill_moves                    0
work_rate                      0
body_type                      0
real_face                      0
release_clause_eur_m         995
player_tags                17536
team_position                  0
team_jersey_number           225
loaned_from                18186
joined                       983
contract_valid_until         225
nation_position            17817
nation_jersey_number       17817
pace                        2083
shooting                    2083
passing                     2083
dribbling                   2083
defending                   2083
physic                      2083
gk_diving                  16861
gk_handling                16861
gk_kicking                 16861
gk_reflexes                16861
gk_speed                   16861
gk_positioning             16861
player_traits                  0
attacking_crossing             0
attacking_finishing            0
attacking_heading_accuracy     0
attacking_short_passing        0
attacking_volleys              0
skill_dribbling                0
skill_curve                    0
skill_fk_accuracy              0
skill_long_passing             0
skill_ball_control             0
movement_acceleration          0
movement_sprint_speed          0
movement_agility               0
movement_reactions             0
movement_balance               0
power_shot_power               0
power_jumping                  0
power_stamina                  0
power_strength                 0
power_long_shots               0
mentality_aggression           0
mentality_interceptions        0
mentality_positioning          0
mentality_vision               0
mentality_penalties            0
mentality_composure            0
defending_marking          18944
defending_standing_tackle      0
defending_sliding_tackle       0
goalkeeping_diving             0
goalkeeping_handling           0
goalkeeping_kicking            0
goalkeeping_positioning        0
goalkeeping_reflexes           0
born_year                      0
league_value_B               225
loyalty                      983
team_quality                   0
dtype: int64
```

In [ ]:

```
# show the missing values in dataset with ratio
def missing_values_tabl(df):
```

```
    na_columns = [col for col in df.columns if df[col].isnull().sum() > 0]
    n_miss = df[na_columns].isnull().sum().sort_values(ascending=False)
    ratio = (df[na_columns].isnull().sum() / df.shape[0] * 100).sort_values(ascending=False)
    missing_df = pd.concat([n_miss, np.round(ratio,2)], axis=1, keys=['n_miss', 'ratio'])
    missing_df = pd.DataFrame(missing_df)
    return missing_df

missing_values_tabl(df)
```

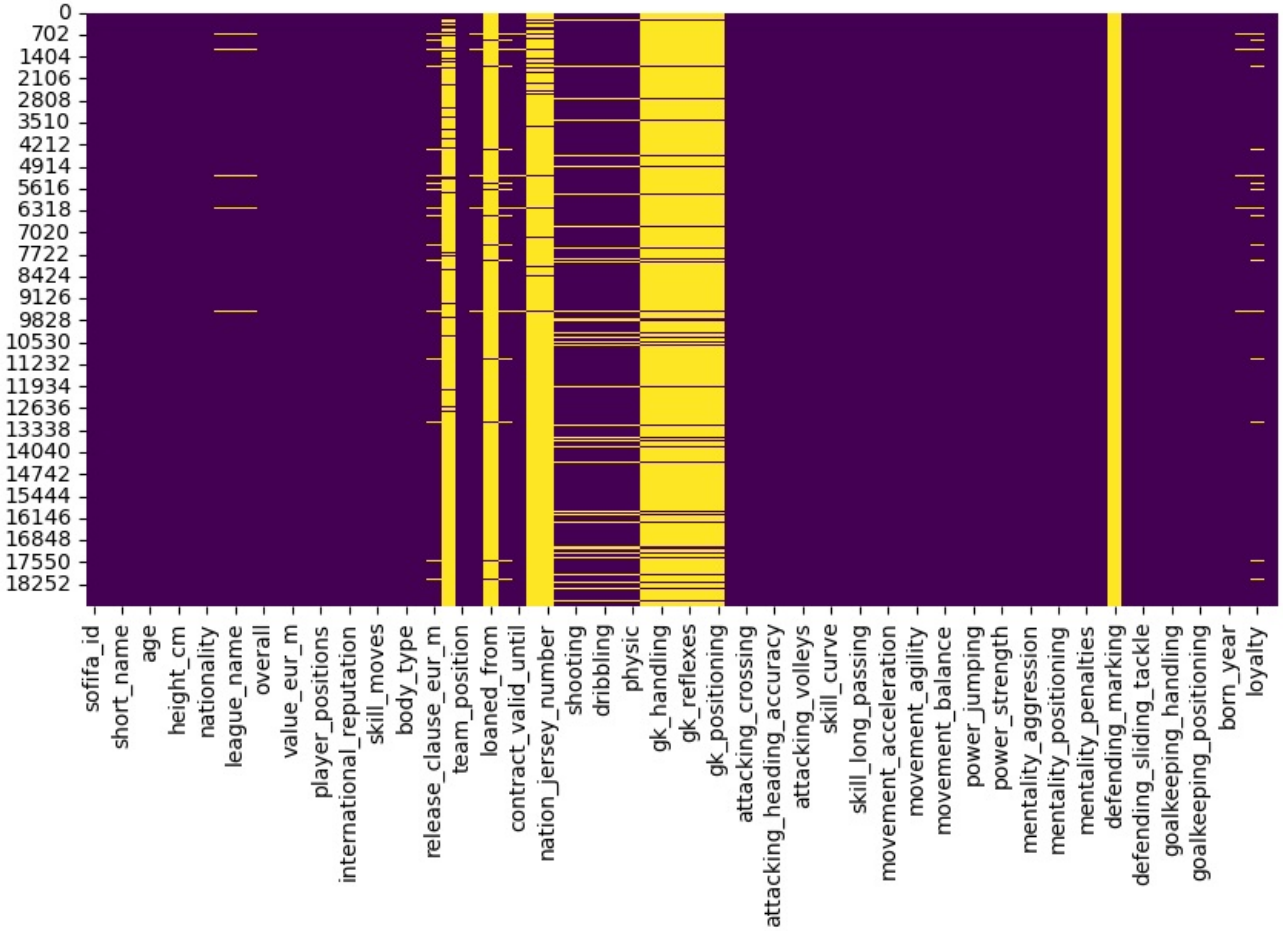
Out[ ]:

	n_miss	ratio
defending_marking	18944	100.00
loaned_from	18186	96.00
nation_position	17817	94.05
nation_jersey_number	17817	94.05
player_tags	17536	92.57
gk_reflexes	16861	89.00
gk_kicking	16861	89.00
gk_handling	16861	89.00
gk_diving	16861	89.00
gk_speed	16861	89.00
gk_positioning	16861	89.00
pace	2083	11.00
shooting	2083	11.00
dribbling	2083	11.00
defending	2083	11.00
physic	2083	11.00
passing	2083	11.00
release_clause_eur_m	995	5.25
loyalty	983	5.19
joined	983	5.19
league_name	225	1.19
contract_valid_until	225	1.19
team_jersey_number	225	1.19
league_rank	225	1.19
league_value_B	225	1.19
club_name	225	1.19

In [ ]:

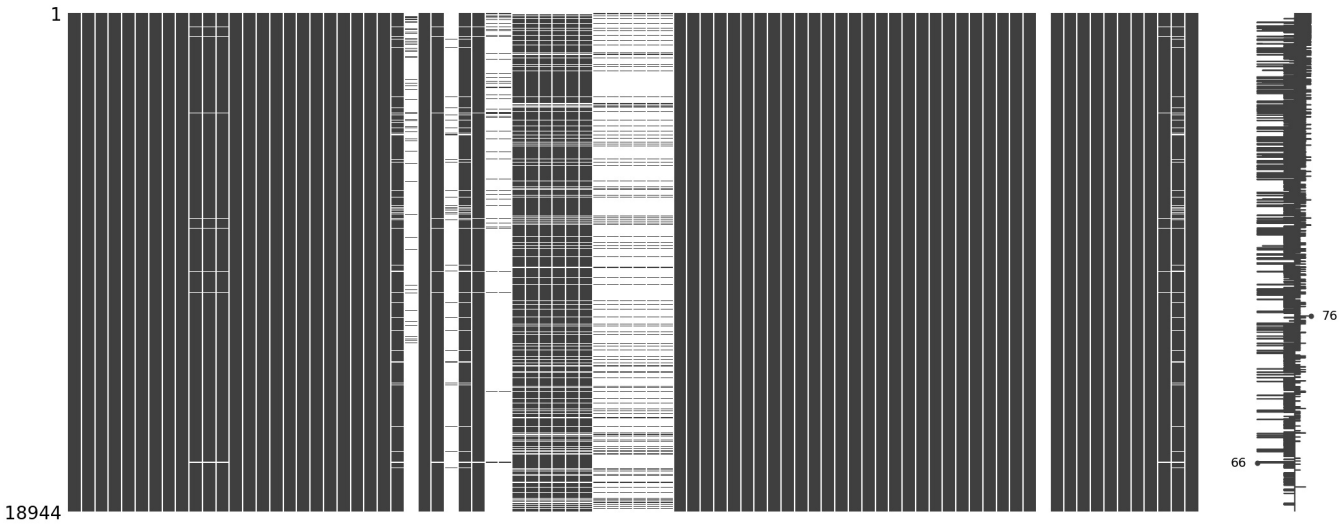
```
# visualize the missing values with heatmap

plt.figure(figsize=(10,5))
sns.heatmap(df.isnull(),cbar=False,cmap='viridis')
plt.xticks(rotation=90)
```

Out[ ]:
```
(array([ 0.5,  2.5,  4.5,  6.5,  8.5, 10.5, 12.5, 14.5, 16.5, 18.5, 20.5,
        22.5, 24.5, 26.5, 28.5, 30.5, 32.5, 34.5, 36.5, 38.5, 40.5, 42.5,
        44.5, 46.5, 48.5, 50.5, 52.5, 54.5, 56.5, 58.5, 60.5, 62.5, 64.5,
        66.5, 68.5, 70.5, 72.5, 74.5, 76.5, 78.5, 80.5, 82.5]),
 [Text(0.5, 0, 'sofifa_id'),
  Text(2.5, 0, 'short_name'),
  Text(4.5, 0, 'age'),
  Text(6.5, 0, 'height_cm'),
  Text(8.5, 0, 'nationality'),
  Text(10.5, 0, 'league_name'),
  Text(12.5, 0, 'overall'),
  Text(14.5, 0, 'value_eur_m'),
  Text(16.5, 0, 'player_positions'),
  Text(18.5, 0, 'international_reputation'),
  Text(20.5, 0, 'skill_moves'),
  Text(22.5, 0, 'body_type'),
  Text(24.5, 0, 'release_clause_eur_m'),
  Text(26.5, 0, 'team_position'),
  Text(28.5, 0, 'loaned_from'),
  Text(30.5, 0, 'contract_valid_until'),
  Text(32.5, 0, 'nation_jersey_number'),
  Text(34.5, 0, 'shooting'),
  Text(36.5, 0, 'dribbling'),
  Text(38.5, 0, 'physic'),
  Text(40.5, 0, 'gk_handling'),
  Text(42.5, 0, 'gk_reflexes'),
  Text(44.5, 0, 'gk_positioning'),
  Text(46.5, 0, 'attacking_crossing'),
  Text(48.5, 0, 'attacking_heading_accuracy'),
  Text(50.5, 0, 'attacking_volleys'),
  Text(52.5, 0, 'skill_curve'),
  Text(54.5, 0, 'skill_long_passing'),
  Text(56.5, 0, 'movement_acceleration'),
  Text(58.5, 0, 'movement_agility'),
  Text(60.5, 0, 'movement_balance'),
  Text(62.5, 0, 'power_jumping'),
  Text(64.5, 0, 'power_strength'),
  Text(66.5, 0, 'mentality_aggression'),
  Text(68.5, 0, 'mentality_positioning'),
  Text(70.5, 0, 'mentality_penalties'),
  Text(72.5, 0, 'defending_marking'),
  Text(74.5, 0, 'defending_sliding_tackle'),
  Text(76.5, 0, 'goalkeeping_handling'),
  Text(78.5, 0, 'goalkeeping_positioning'),
  Text(80.5, 0, 'born_year'),
  Text(82.5, 0, 'loyalty')])
```



In [ ]:
```
# visualize the missing values with barplot
import missingno as msno
msno.matrix(df);
```

```
plt.show()
```



In [ ]:

```
# handling the missing values by using KNN imputer

num=df.select_dtypes(include=['float64','int64'])


for i in df.columns:
    if df[i].isnull().sum()>len(df)*0.90:
        df.drop(i,axis=1,inplace=True)

df.isnull().sum()
```

Out[ ]:

```
sofifa_id                      0
player_url                     0
short_name                     0
long_name                      0
age                            0
dob                            0
height_cm                      0
weight_kg                      0
nationality                    0
club_name                    225
league_name                  225
league_rank                  225
overall                        0
potential                      0
value_eur_m                    0
wage_eur_m                     0
player_positions               0
preferred_foot                 0
international_reputation       0
weak_foot                      0
skill_moves                    0
work_rate                      0
body_type                      0
real_face                      0
release_clause_eur_m         995
team_position                  0
team_jersey_number           225
joined                       983
contract_valid_until         225
pace                        2083
shooting                    2083
passing                     2083
dribbling                   2083
defending                   2083
physic                      2083
gk_diving                  16861
gk_handling                16861
gk_kicking                 16861
gk_reflexes                16861
gk_speed                   16861
gk_positioning             16861
player_traits                  0
attacking_crossing             0
attacking_finishing            0
attacking_heading_accuracy     0
attacking_short_passing        0
attacking_volleys              0
skill_dribbling                0
skill_curve                    0
skill_fk_accuracy              0
skill_long_passing             0
skill_ball_control             0
movement_acceleration          0
movement_sprint_speed          0
movement_agility               0
movement_reactions             0
movement_balance               0
power_shot_power               0
power_jumping                  0
power_stamina                  0
power_strength                 0
power_long_shots               0
mentality_aggression           0
mentality_interceptions        0
mentality_positioning          0
mentality_vision               0
mentality_penalties            0
mentality_composure            0
defending_standing_tackle      0
defending_sliding_tackle       0
goalkeeping_diving             0
goalkeeping_handling           0
goalkeeping_kicking            0
goalkeeping_positioning        0
goalkeeping_reflexes           0
born_year                      0
league_value_B               225
loyalty                      983
team_quality                   0
dtype: int64
```

In [ ]:

```
from sklearn.impute import KNNImputer
knn=KNNImputer(n_neighbors=15)
num=df.select_dtypes(include=['float64','int64'])
for i in num.columns:
   df[i]=knn.fit_transform(df[[i]])

df.isnull().sum()
```

Out[ ]:

```
sofifa_id                      0
player_url                     0
short_name                     0
long_name                      0
age                            0
dob                            0
height_cm                      0
weight_kg                      0
nationality                    0
club_name                    225
league_name                  225
league_rank                    0
overall                        0
potential                      0
value_eur_m                    0
wage_eur_m                     0
player_positions               0
preferred_foot                 0
international_reputation       0
weak_foot                      0
skill_moves                    0
work_rate                      0
body_type                      0
real_face                      0
release_clause_eur_m           0
team_position                  0
team_jersey_number             0
joined                       983
contract_valid_until           0
pace                           0
shooting                       0
passing                        0
dribbling                      0
defending                      0
physic                         0
gk_diving                      0
gk_handling                    0
gk_kicking                     0
gk_reflexes                    0
gk_speed                       0
gk_positioning                 0
player_traits                  0
attacking_crossing             0
attacking_finishing            0
attacking_heading_accuracy     0
attacking_short_passing        0
attacking_volleys              0
skill_dribbling                0
skill_curve                    0
skill_fk_accuracy              0
skill_long_passing             0
skill_ball_control             0
movement_acceleration          0
movement_sprint_speed          0
movement_agility               0
movement_reactions             0
movement_balance               0
power_shot_power               0
power_jumping                  0
power_stamina                  0
power_strength                 0
power_long_shots               0
mentality_aggression           0
mentality_interceptions        0
mentality_positioning          0
mentality_vision               0
mentality_penalties            0
mentality_composure            0
defending_standing_tackle      0
defending_sliding_tackle       0
goalkeeping_diving             0
goalkeeping_handling           0
goalkeeping_kicking            0
goalkeeping_positioning        0
goalkeeping_reflexes           0
born_year                      0
league_value_B                 0
loyalty                        0
team_quality                   0
dtype: int64
```

DATA VISUALIZATION

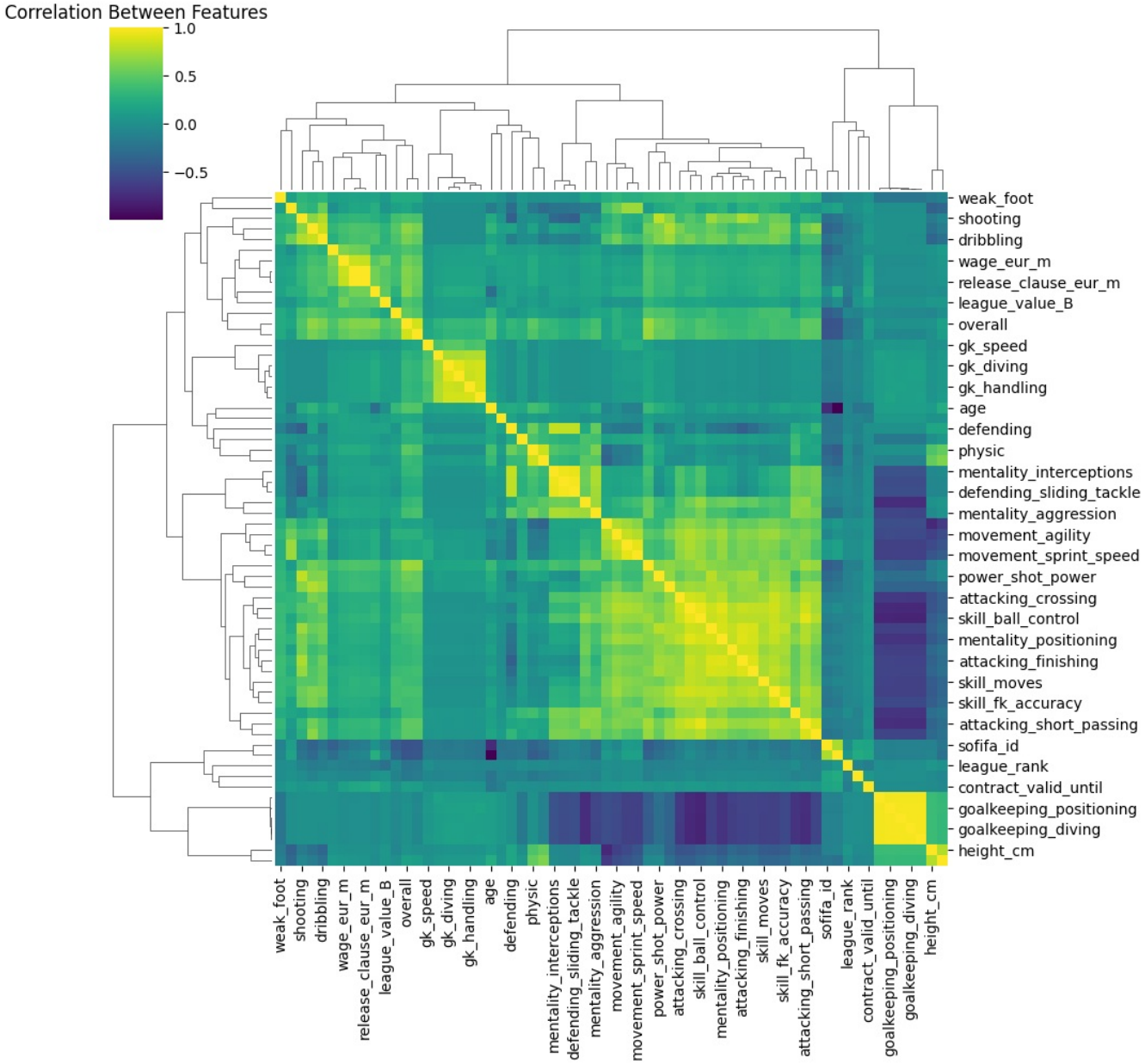
In [ ]:

```
# check the correlation between the features
```

```
# correlation matrix:
corr=df.iloc[:,0:78].corr()
plt.figure(figsize=(40,20))
sns.clustermap(corr,cmap="viridis")
plt.title("Correlation Between Features")
```

Out[ ]:

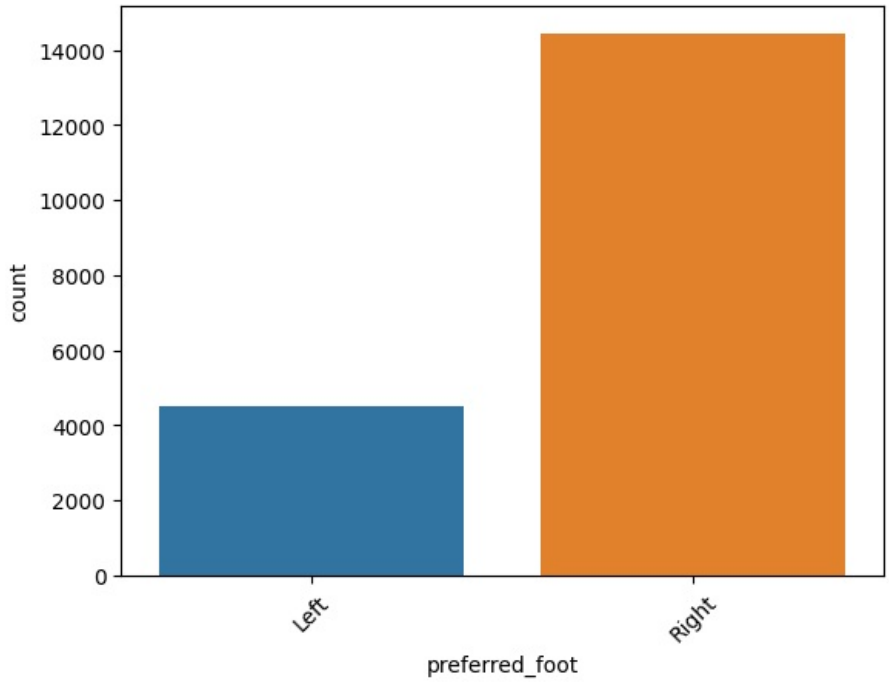
```
Text(0.5, 1.0, 'Correlation Between Features')

<Figure size 4000x2000 with 0 Axes>
```



In [ ]:

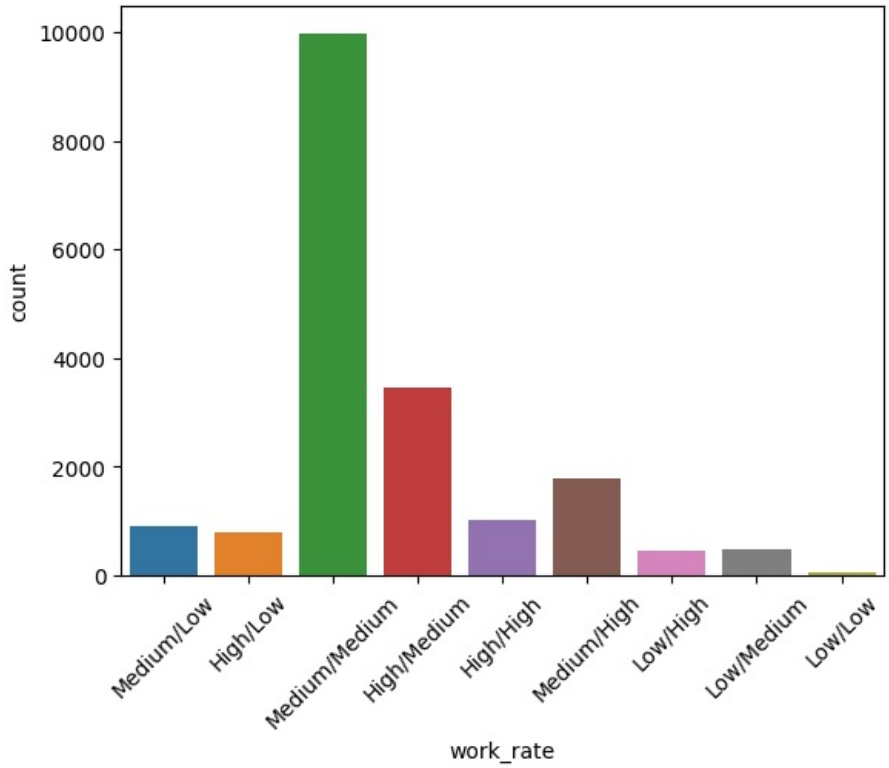
```
cat_cols = [col for col in df.columns if df[col].dtypes == "O" and df[col].nunique() <= 10]
num_cols = [col for col in df.columns if df[col].dtypes != "O"]



def cat_summary(df,col,plot=False):
    print(pd.DataFrame({col:df[col].value_counts(),
                        "Ratio":100*df[col].value_counts()/len(df)}))
    print("##########################################")
    if plot:
        sns.countplot(x=col,data=df)
        plt.xticks(rotation=45)
        plt.show()

for i in cat_cols:
    cat_summary(df,i,plot=True)
```

```
       preferred_foot      Ratio
Right           14448  76.266892
Left             4496  23.733108
##########################################
```



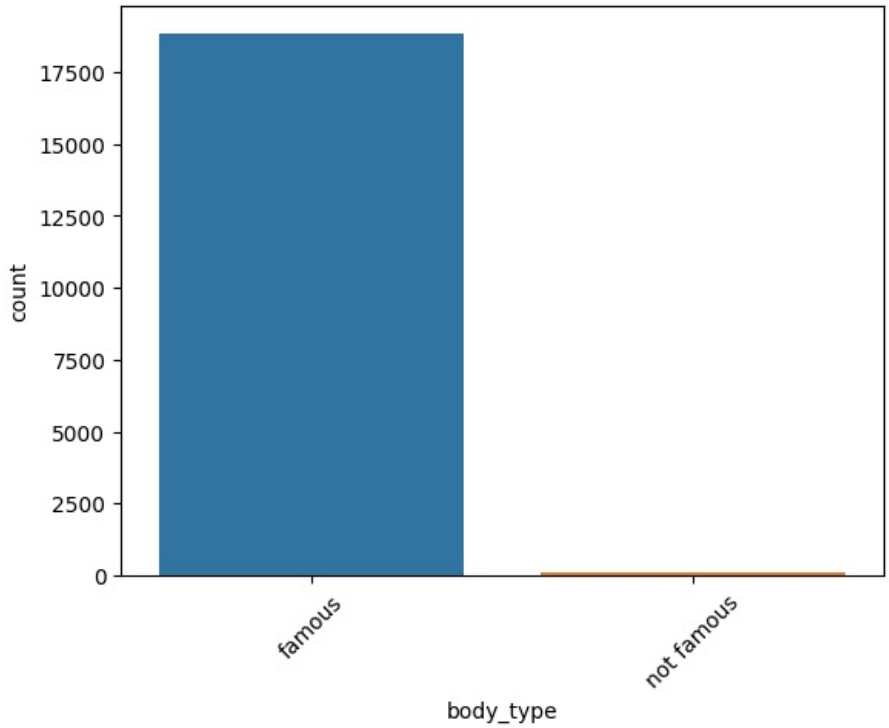
	work_rate	Ratio
Medium/Medium	9980	52.681588
High/Medium	3455	18.237965
Medium/High	1798	9.491132
High/High	1031	5.442356
Medium/Low	913	4.819468
High/Low	786	4.149071
Low/Medium	482	2.544341
Low/High	446	2.354307
Low/Low	53	0.279772

##



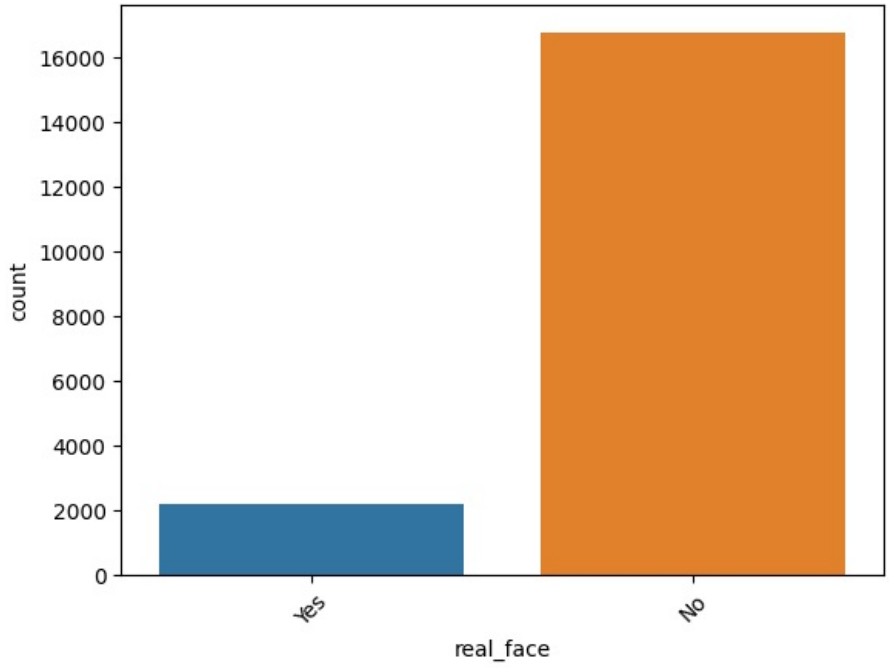
	body_type	Ratio
famous	18836	99.429899
not famous	108	0.570101

##



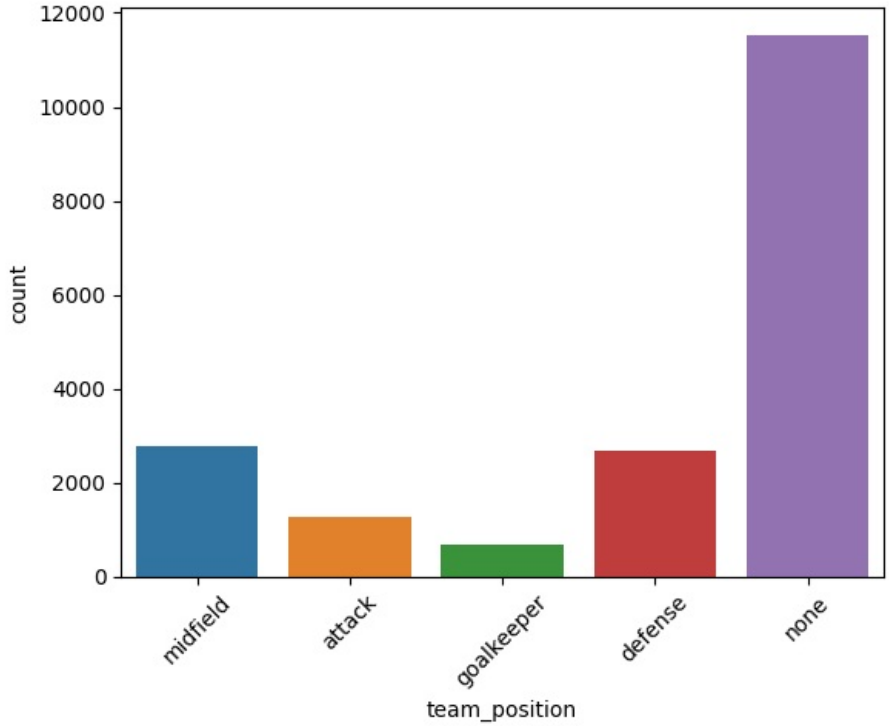
	real_face	Ratio
No	16746	88.397382
Yes	2198	11.602618

##

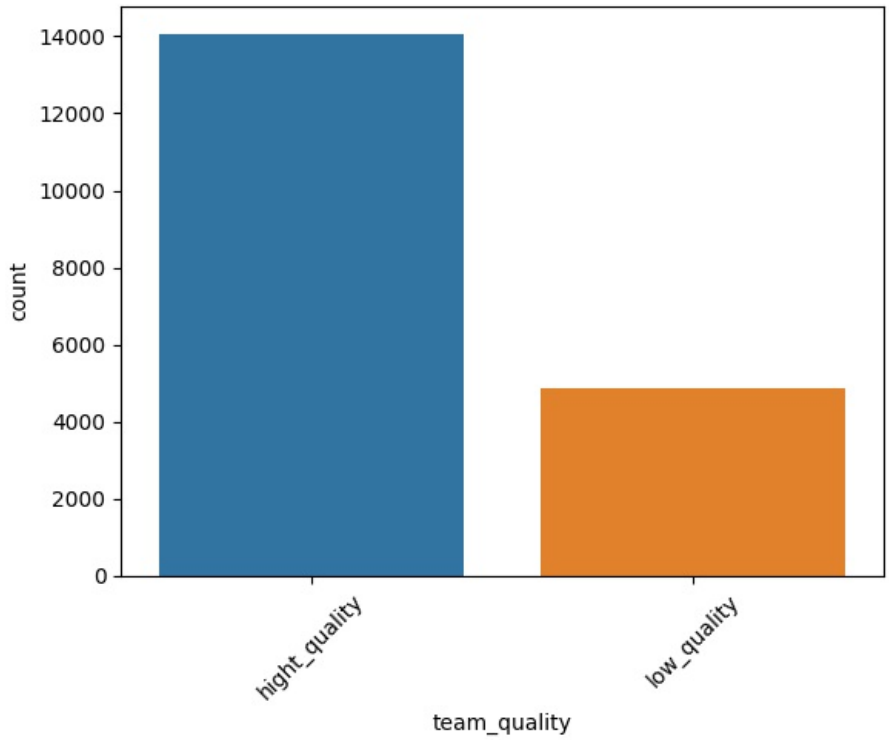


	team_position	Ratio
none	11531	60.868877
midfield	2779	14.669552
defense	2674	14.115287
attack	1279	6.751478
goalkeeper	681	3.594806

##

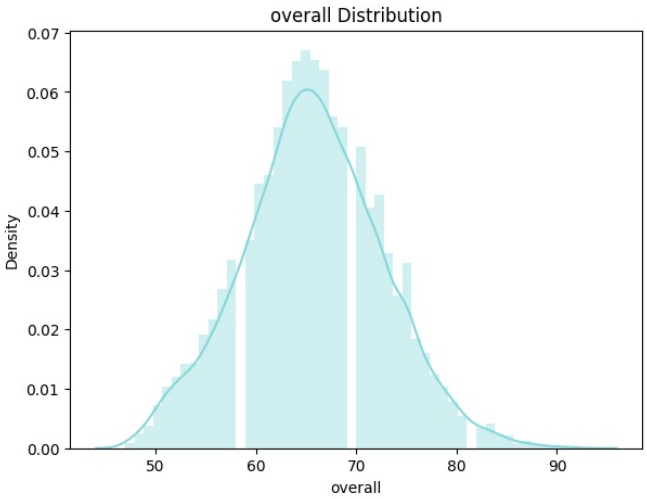


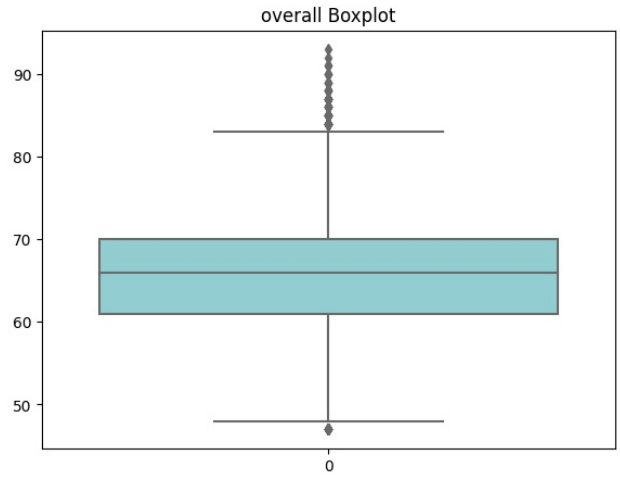

```
               team_quality      Ratio
hight_quality         14075  74.297931
low_quality            4869  25.702069
##########################################
```



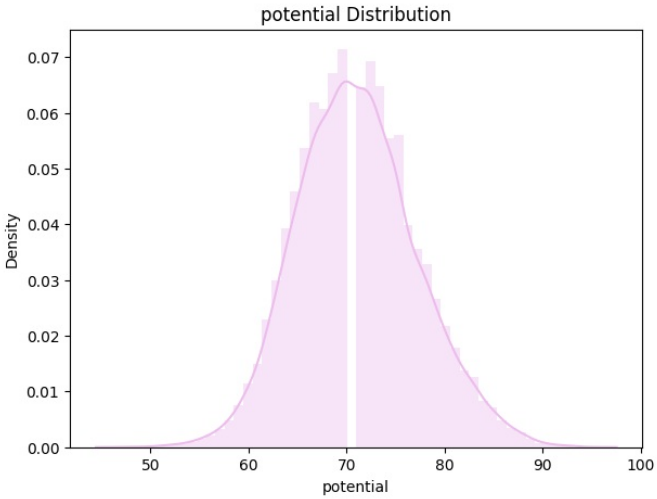
In [ ]:
```
# distribution of the features

colors = ['#7DBCE6','#EEBDEE','#EAEAAF','#8FE195','#E28181',
          '#87D8DB','#C2E37D','#DF93A4','#DCB778','#C497DE']

features=['overall','potential','value_eur_m','wage_eur_m','international_reputation','weak_foot','skill_moves'
            'pace','shooting','passing','dribbling','defending']
```

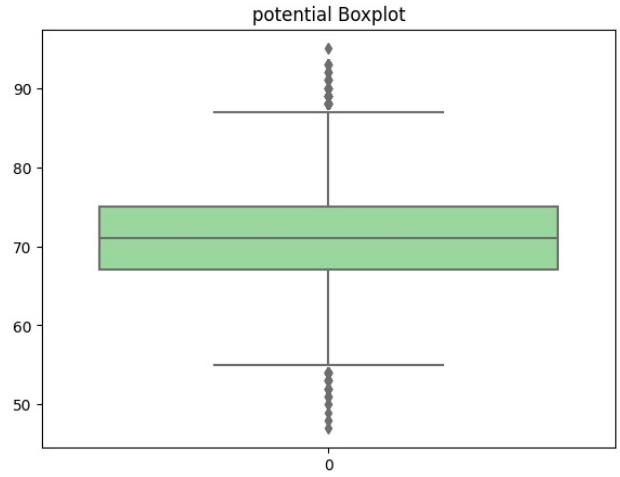
```
def num_summary(dataframe, col_name):
    fig, ax = plt.subplots(1,2,figsize=(15,5))
    sns.distplot(dataframe[col_name], ax=ax[0], color=np.random.choice(colors))
    ax[0].set_title(col_name + ' Distribution')
    sns.boxplot(dataframe[col_name], ax=ax[1], color=np.random.choice(colors))
    ax[1].set_title(col_name + ' Boxplot')
    plt.show()
    print(dataframe[col_name].describe().T)
    print("Skewness: %f" % dataframe[col_name].skew())
    print("Kurtosis: %f" % dataframe[col_name].kurt())
    print("****************************")


for i in df[features].columns:
    num_summary(df,i)
```



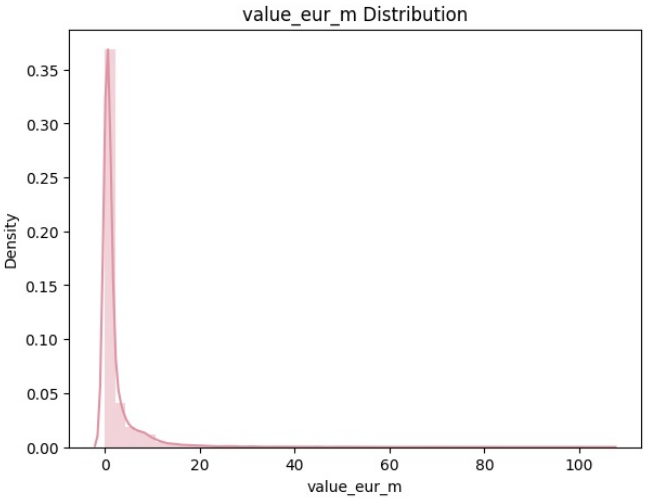


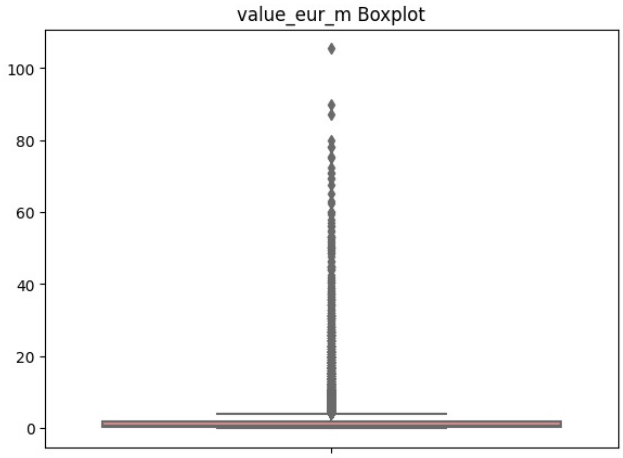

```
count    18944.000000
mean        65.677787
std          7.002278
min         47.000000
25%         61.000000
50%         66.000000
75%         70.000000
max         93.000000
Name: overall, dtype: float64
Skewness: 0.089313
Kurtosis: -0.002538
****************************
```



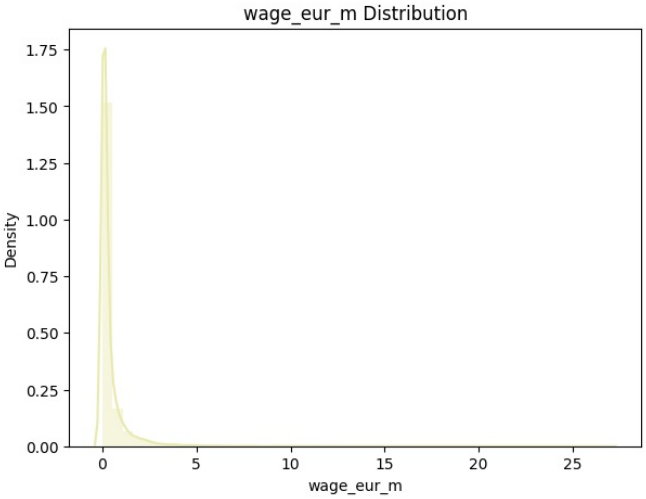


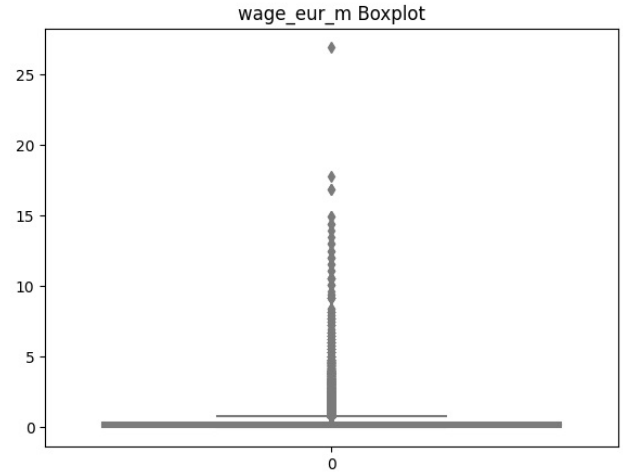

```
count    18944.000000
mean        71.086729
std          6.109985
min         47.000000
25%         67.000000
50%         71.000000
75%         75.000000
max         95.000000
Name: potential, dtype: float64
Skewness: 0.217394
Kurtosis: 0.092657
****************************
```



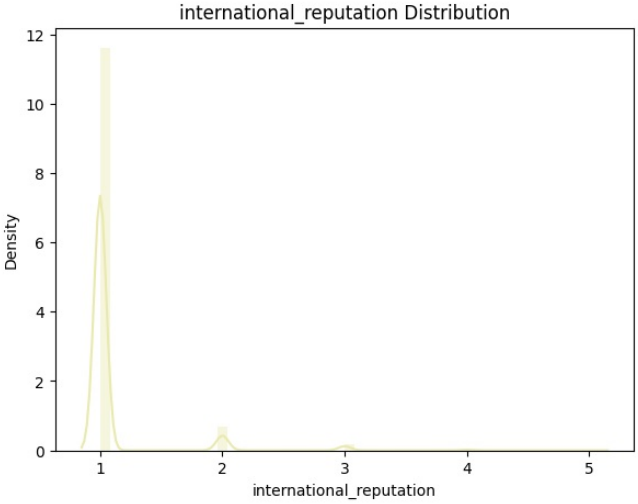


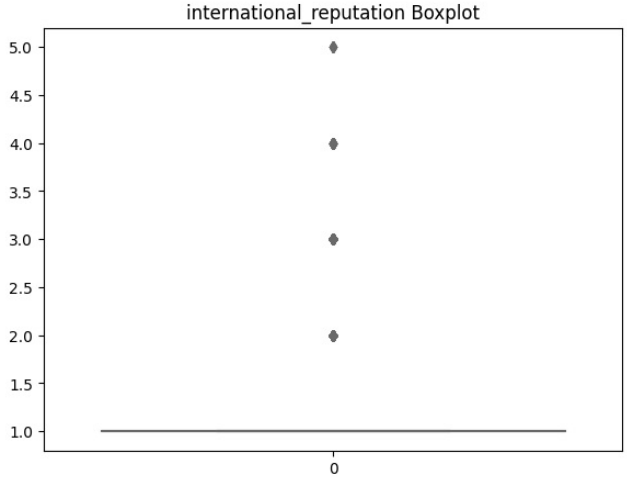

```
count    18944.000000
mean         2.224813
std          5.102486
min          0.000000
25%          0.300000
50%          0.650000
75%          1.800000
max        105.500000
Name: value_eur_m, dtype: float64
Skewness: 6.727281
Kurtosis: 68.855929
****************************
```



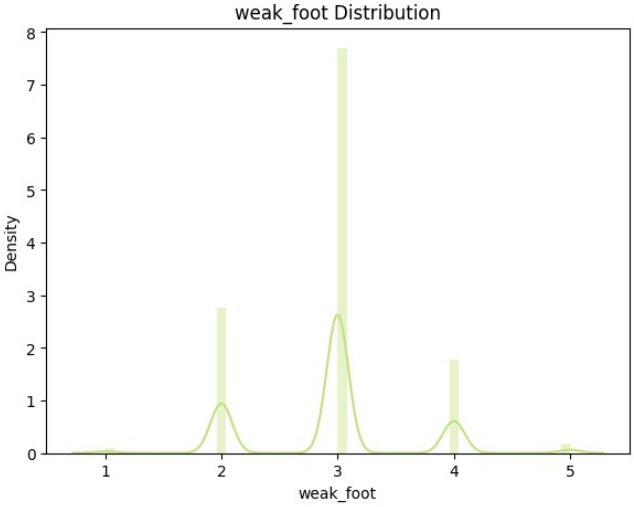


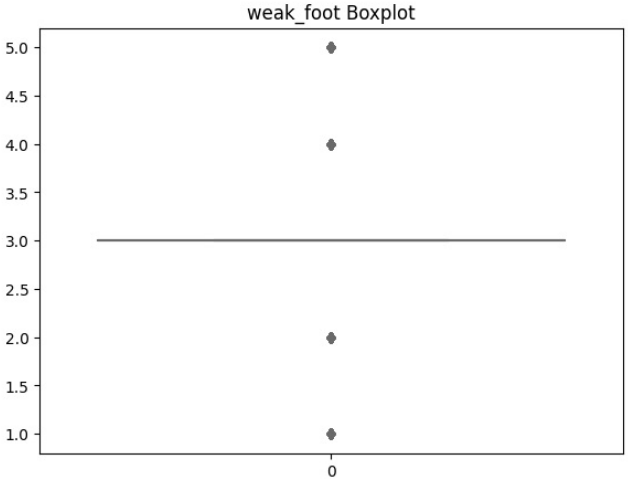

```
count    18944.000000
mean         0.416441
std          0.943429
min          0.000000
25%          0.048000
50%          0.144000
75%          0.336000
max         26.880000
Name: wage_eur_m, dtype: float64
Skewness: 7.484533
Kurtosis: 96.591958
****************************
```



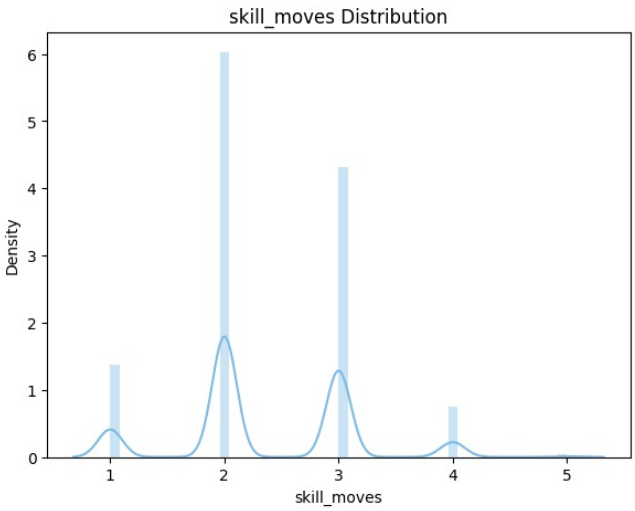


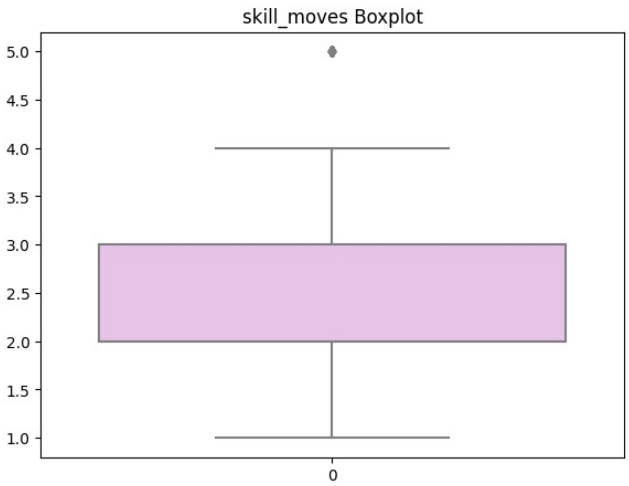

```
count    18944.000000
mean         1.091850
std          0.361841
min          1.000000
25%          1.000000
50%          1.000000
75%          1.000000
max          5.000000
Name: international_reputation, dtype: float64
Skewness: 4.618014
Kurtosis: 24.559540
**************************
```



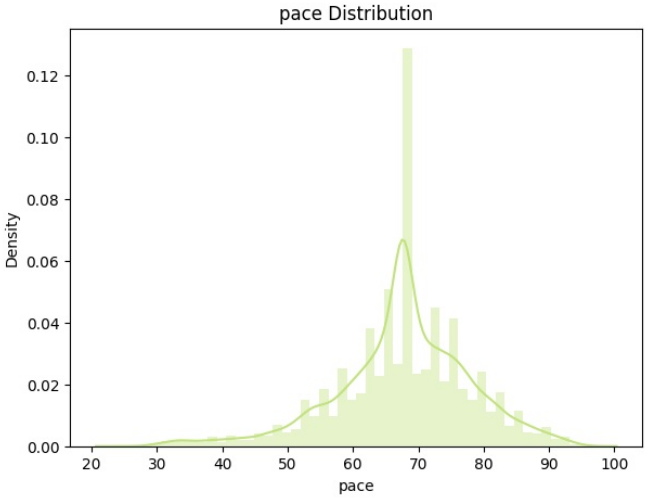


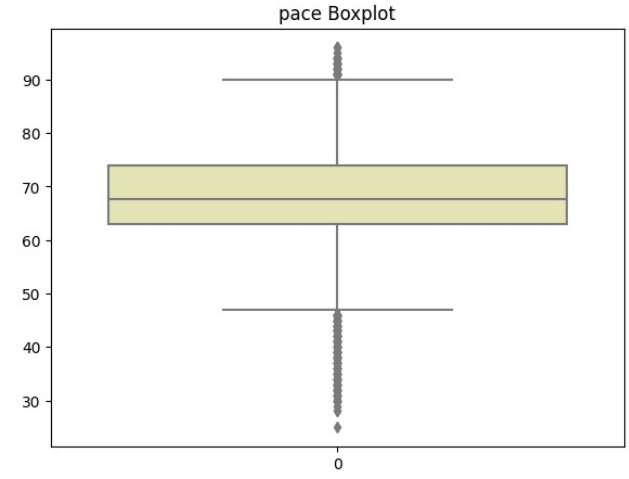

```
count    18944.000000
mean         2.936603
std          0.667132
min          1.000000
25%          3.000000
50%          3.000000
75%          3.000000
max          5.000000
Name: weak_foot, dtype: float64
Skewness: 0.219654
Kurtosis: 0.601850
**************************
```



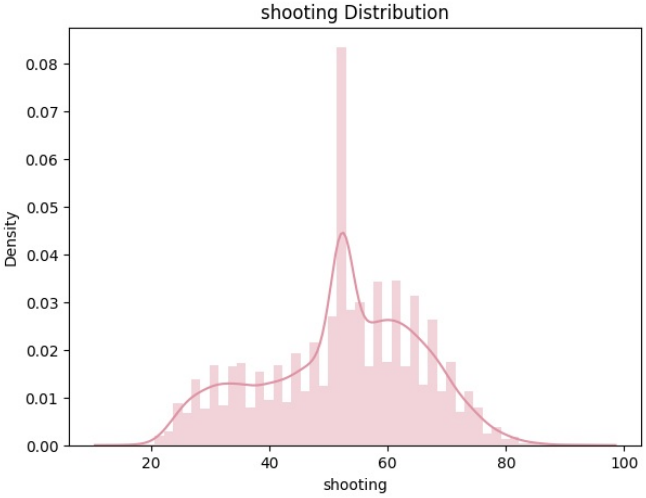


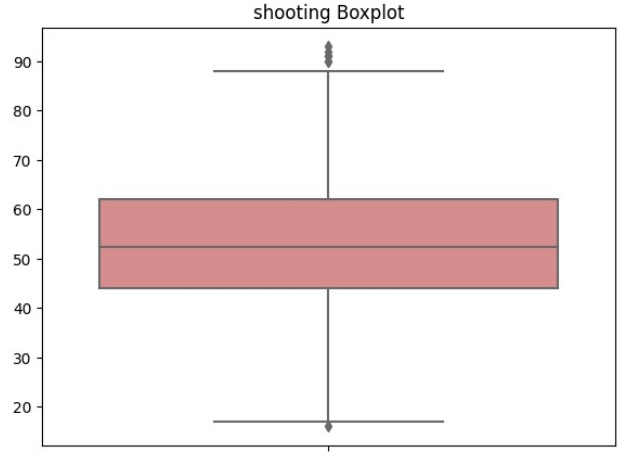

```
count    18944.000000
mean         2.363017
std          0.766469
min          1.000000
25%          2.000000
50%          2.000000
75%          3.000000
max          5.000000
Name: skill_moves, dtype: float64
Skewness: 0.225260
Kurtosis: -0.067779
****************************
```



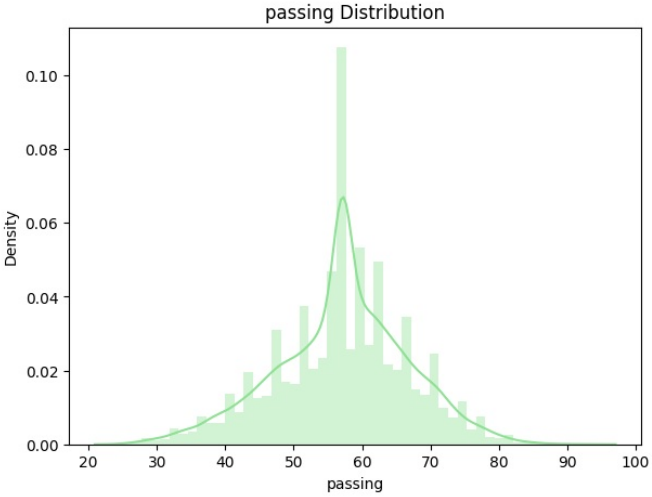


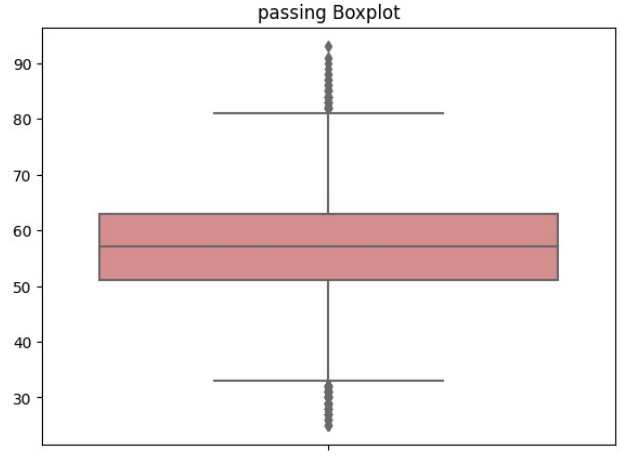

```
count    18944.000000
mean        67.668110
std         10.363322
min         25.000000
25%         63.000000
50%         67.668110
75%         74.000000
max         96.000000
Name: pace, dtype: float64
Skewness: -0.539655
Kurtosis: 1.101084
****************************
```



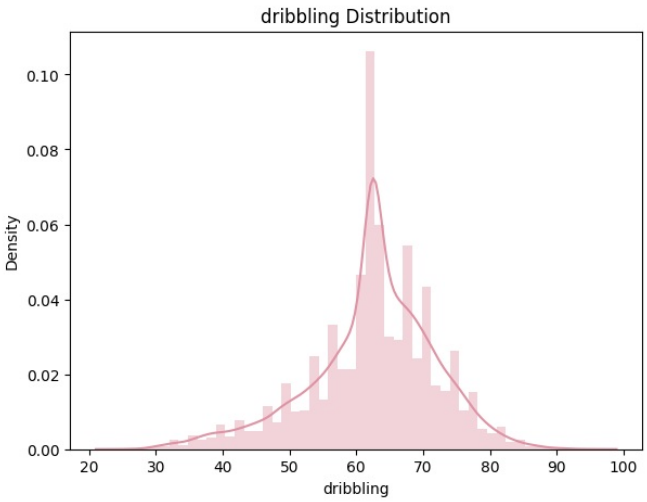


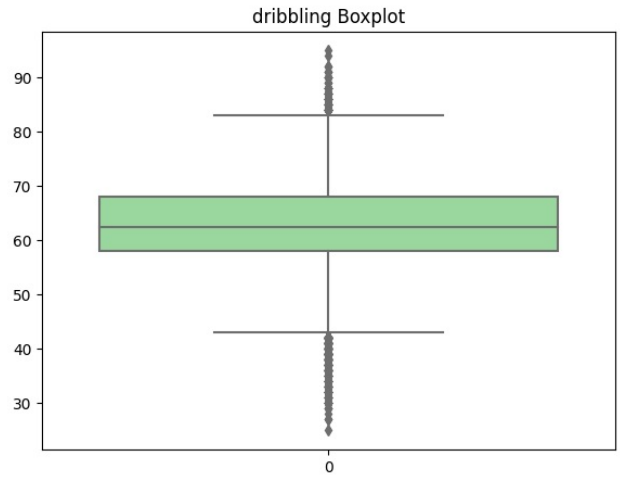

```
count    18944.000000
mean        52.274954
std         13.199766
min         16.000000
25%         44.000000
50%         52.274954
75%         62.000000
max         93.000000
Name: shooting, dtype: float64
Skewness: -0.287125
Kurtosis: -0.497940
****************************
```



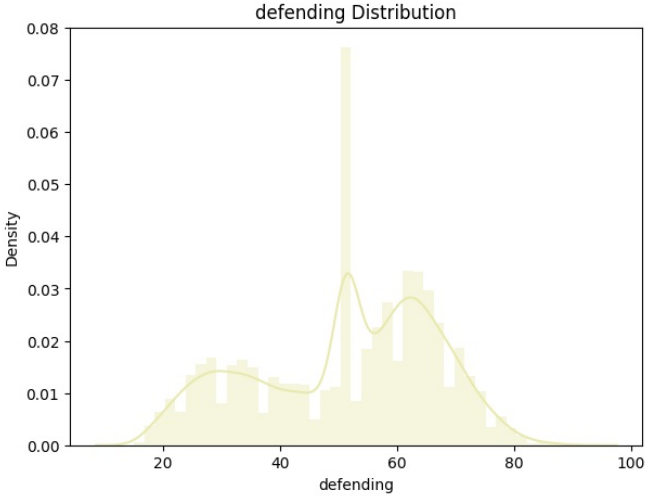


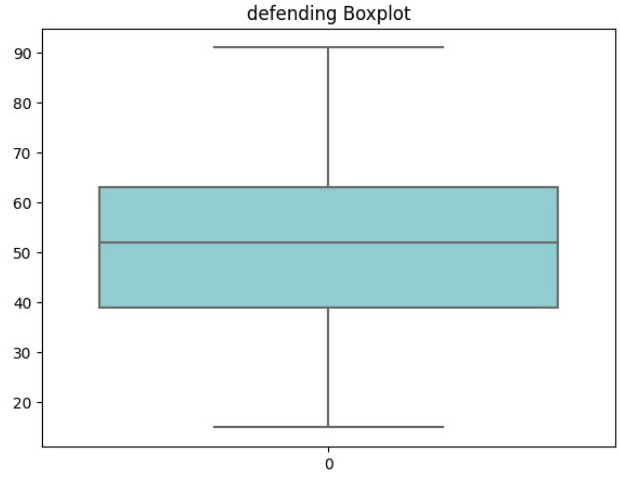

```
count    18944.000000
mean        57.139434
std          9.692357
min         25.000000
25%         51.000000
50%         57.139434
75%         63.000000
max         93.000000
Name: passing, dtype: float64
Skewness: -0.199215
Kurtosis: 0.218255
****************************
```






```
count    18944.000000
mean        62.455430
std          9.480136
min         25.000000
25%         58.000000
50%         62.455430
75%         68.000000
max         95.000000
Name: dribbling, dtype: float64
Skewness: -0.552649
Kurtosis: 0.782478
****************************
```

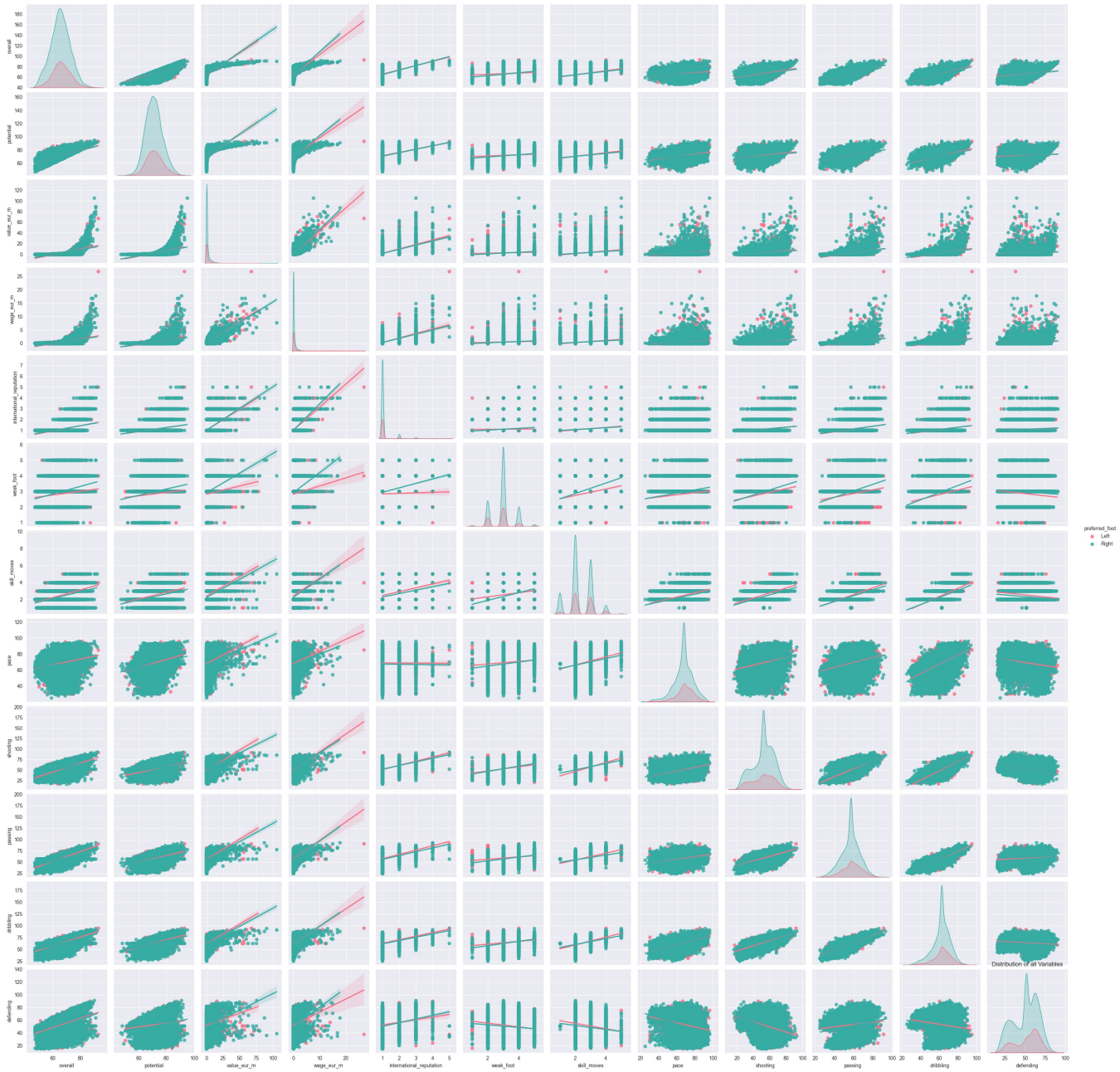




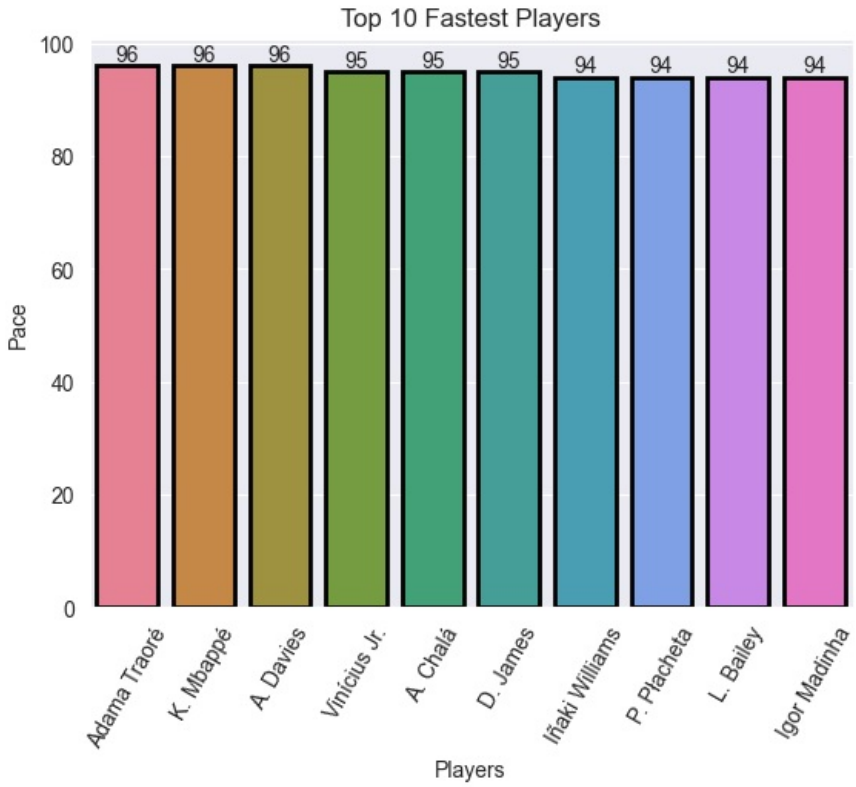

```
count    18944.000000
mean        51.316292
std         15.476950
min         15.000000
25%         39.000000
50%         52.000000
75%         63.000000
max         91.000000
Name: defending, dtype: float64
Skewness: -0.374699
Kurtosis: -0.801525
****************************
```

In [ ]:
```
# pairplot of the selected features

features=['overall','potential','value_eur_m','wage_eur_m','international_reputation','weak_foot','skill_moves'
          'pace','shooting','passing','dribbling','defending',"preferred_foot"]


plt.figure(figsize=(20,20))
sns.set_style("darkgrid")
sns.pairplot(df[features],kind="reg",diag_kind="kde",palette="husl",hue="preferred_foot");
plt.title("Distribution of all Variables")
```

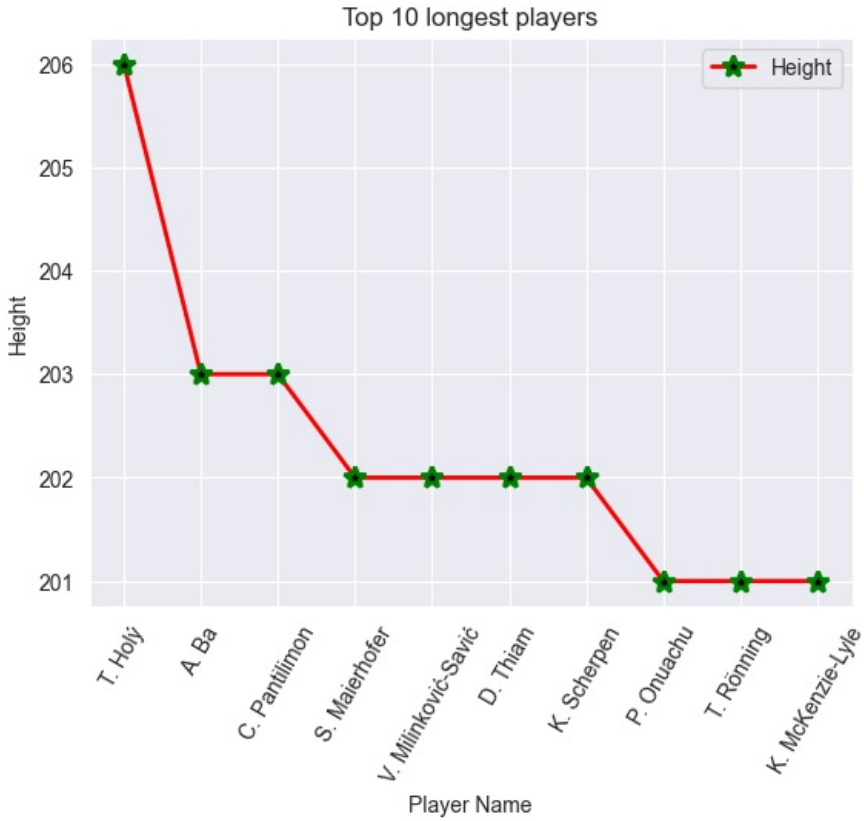
Out[ ]:
```
Text(0.5, 1.0, 'Distribution of all Variables')

<Figure size 2000x2000 with 0 Axes>
```



In [ ]:
```
# top 10 fastest players

fs=df.sort_values(by="pace",ascending=False).head(10)

sns.set_style("darkgrid")
ax=sns.barplot(x=fs["short_name"],y=fs["pace"],palette="husl",edgecolor="black",linewidth=2)
for i in ax.containers:
    ax.bar_label(i,)
plt.xticks(rotation=60)
plt.title("Top 10 Fastest Players")
plt.xlabel("Players")
plt.ylabel("Pace")
plt.show()
```

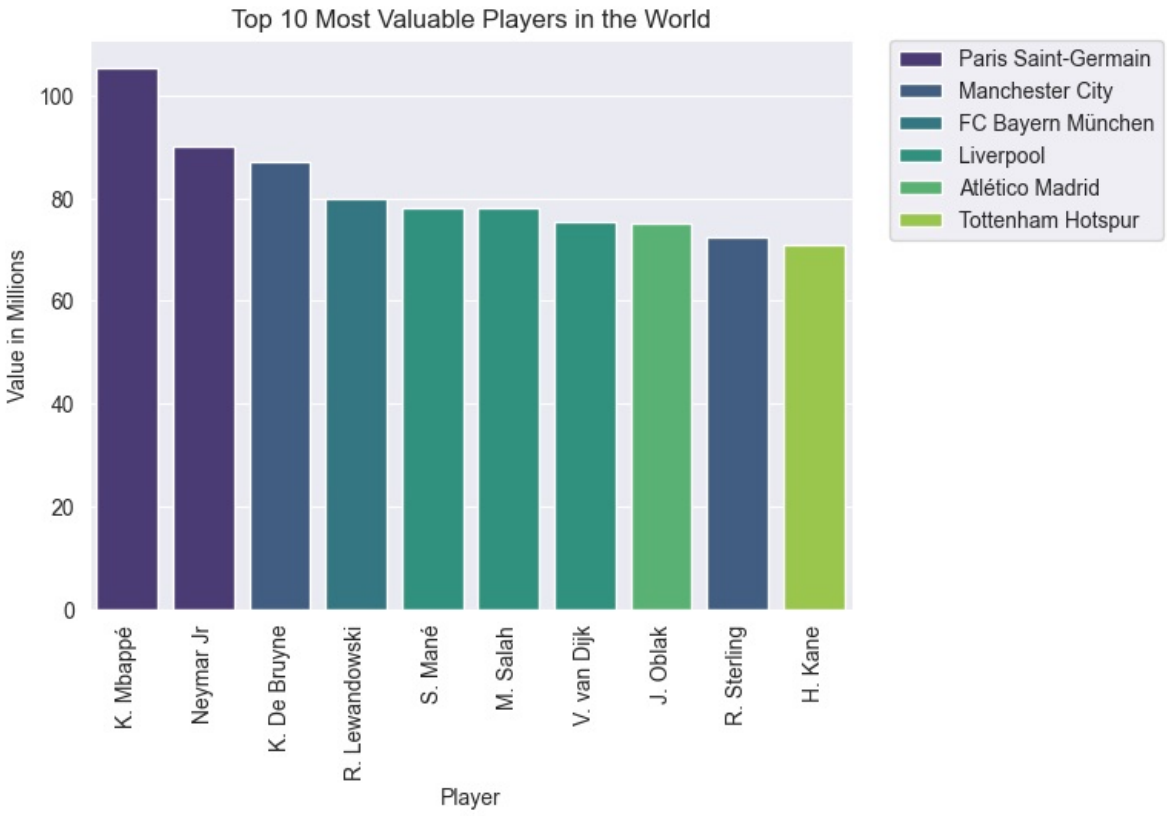


In [ ]:
```
# top 10 longest players:

lg=df.sort_values(by='height_cm', ascending=False).head(10)

sns.set_style('darkgrid')
sns.lineplot(x='short_name', y='height_cm', data=lg, color='red', label='Height', marker='*', markersize=10,
             markerfacecolor='black', markeredgewidth=2, markeredgecolor="green", linewidth=2)
plt.title('Top 10 longest players')
plt.ylabel('Height')
plt.xlabel('Player Name')
plt.xticks(rotation=60)
plt.show()
```



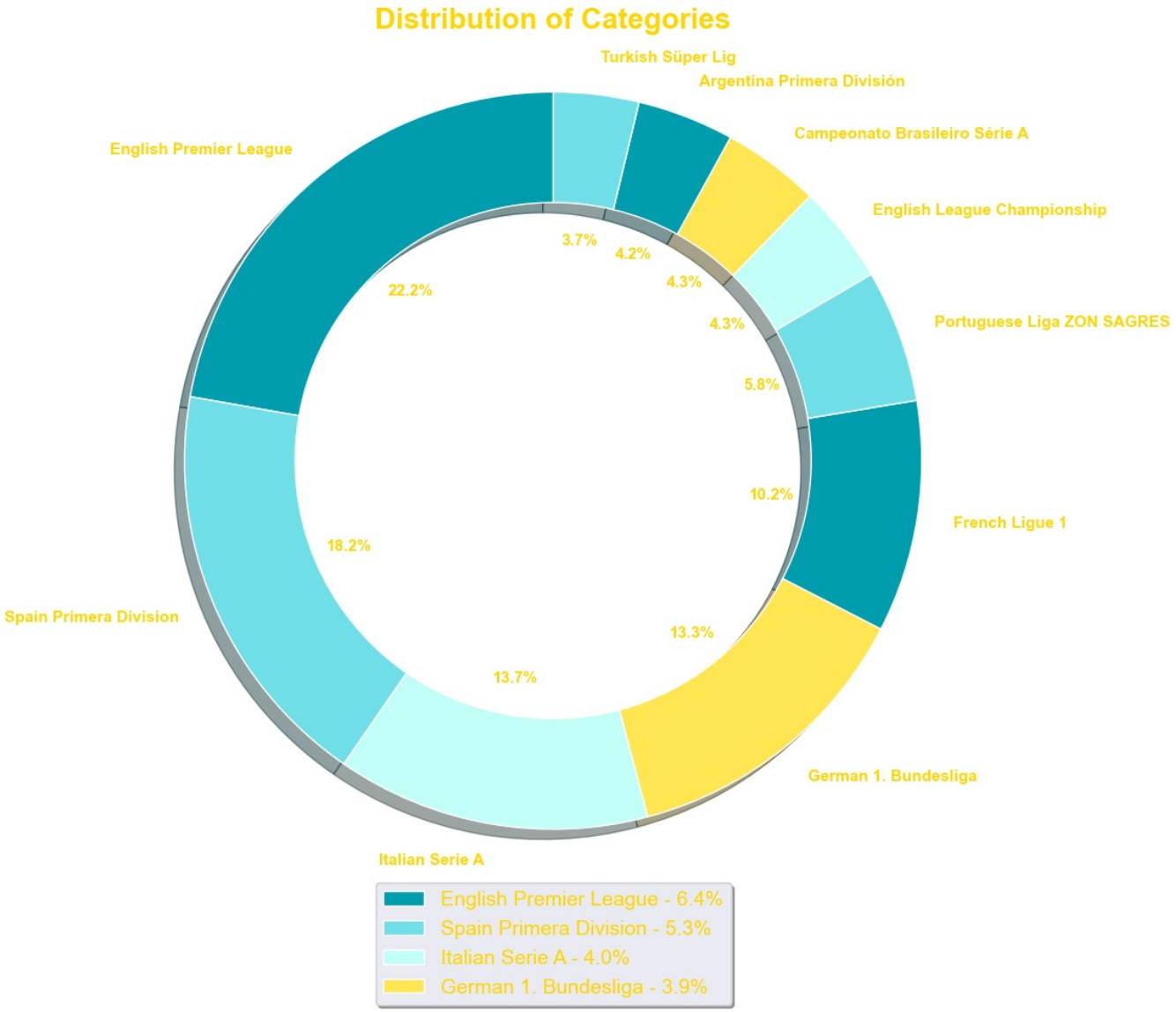
In [ ]:
```
# top 10 most valuable players in the world

vl= df.sort_values(by='value_eur_m', ascending=False).head(10)
sns.set_style('darkgrid')
sns.barplot(x='short_name', y='value_eur_m', data=vl, palette='viridis', hue='club_name', dodge=False, ci=None)
plt.legend(bbox_to_anchor=(1.05, 1), loc=2, borderaxespad=0.)
plt.xticks(rotation=90)
```

```
plt.xlabel('Player')
plt.ylabel('Value in Millions')
plt.title('Top 10 Most Valuable Players in the World')
plt.show()
```



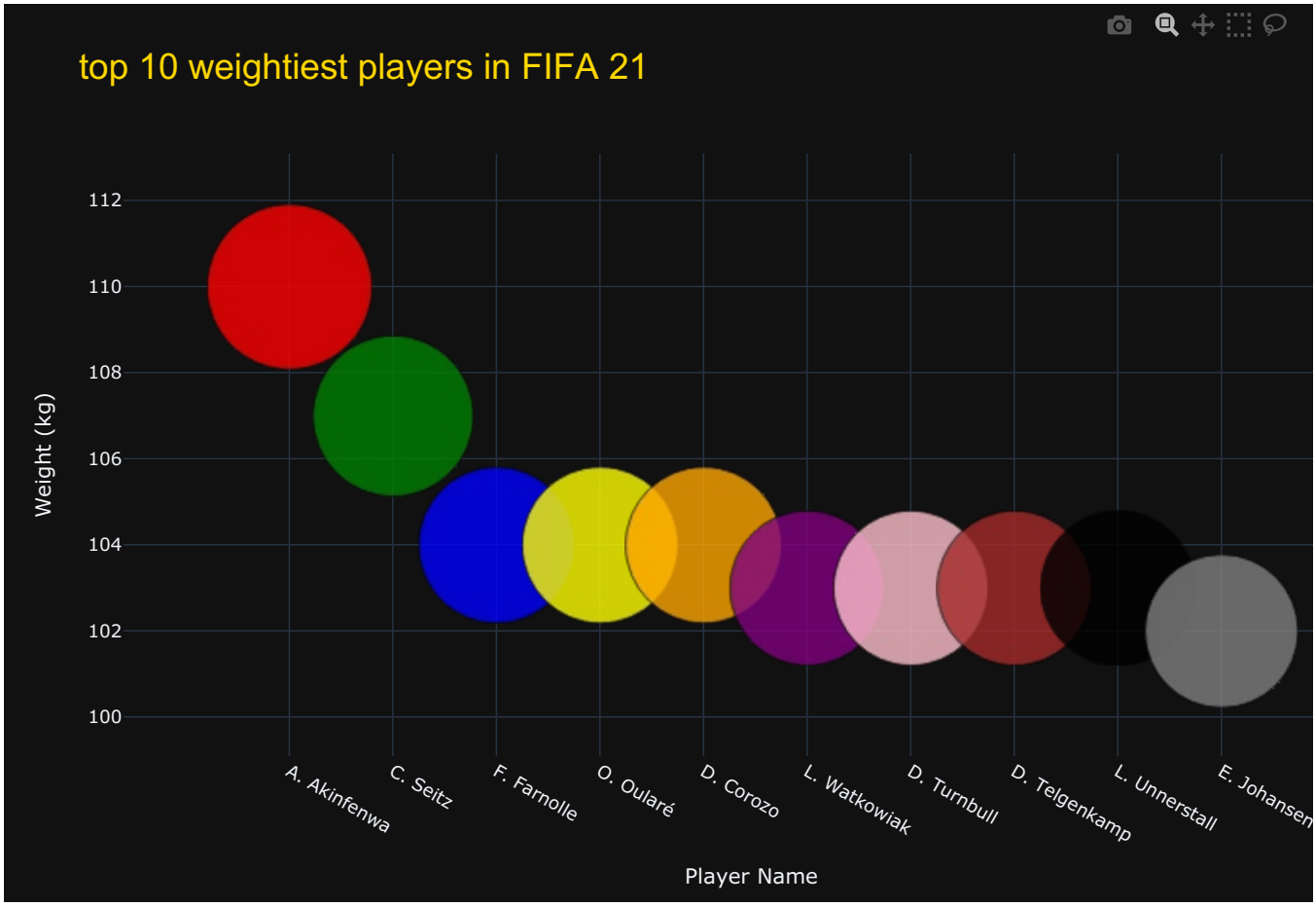
In [ ]:

```
# top 10 most valuable leagues

import matplotlib.patches as mpatches



lv=df.groupby('league_name').mean().sort_values(by='league_value_B',ascending=False).head(10).reset_index()



colors = ['#009DAE', '#71DFE7', '#C2FFF9', '#FFE652']  # Custom colors for each category
explode = (0.05, 0.05, 0.05, 0.05)  # Explode the 1st slice

# Create the pie chart
plt.figure(figsize=(10, 10))  # Set the figure size

# Plot the pie chart with customizations
wedges, texts, autotexts = plt.pie(lv['league_value_B'],  labels=lv['league_name'], colors=colors,
                                   autopct='%1.1f%%', startangle=90, shadow=True,
                                   wedgeprops=dict(width=0.3, edgecolor='w'))

# Add a title
plt.title('Distribution of Categories', fontsize=20, fontweight='bold', pad=20,color='gold')

# Customize font style for text and percentage labels
for text in texts + autotexts:
    text.set_fontsize(11)
    text.set_fontweight('semibold')
    text.set_color('gold')

# Create a legend with custom labels and colors
legend_labels = ['{} - {:.1f}%'.format(label, size) for label, size in zip(lv['league_name'], lv['league_value_
legend_colors = [mpatches.Patch(facecolor=color) for color in colors]
leg=plt.legend(legend_colors, legend_labels, loc='center', bbox_to_anchor=(0.5, -0.1),
           fontsize=14, title_fontsize=16, fancybox=True, shadow=True)
for text in leg.get_texts():
    text.set_color("gold")

# Equal aspect ratio ensures that the pie chart is drawn as a circle.
plt.axis('equal')

# Display the chart
plt.show()
```



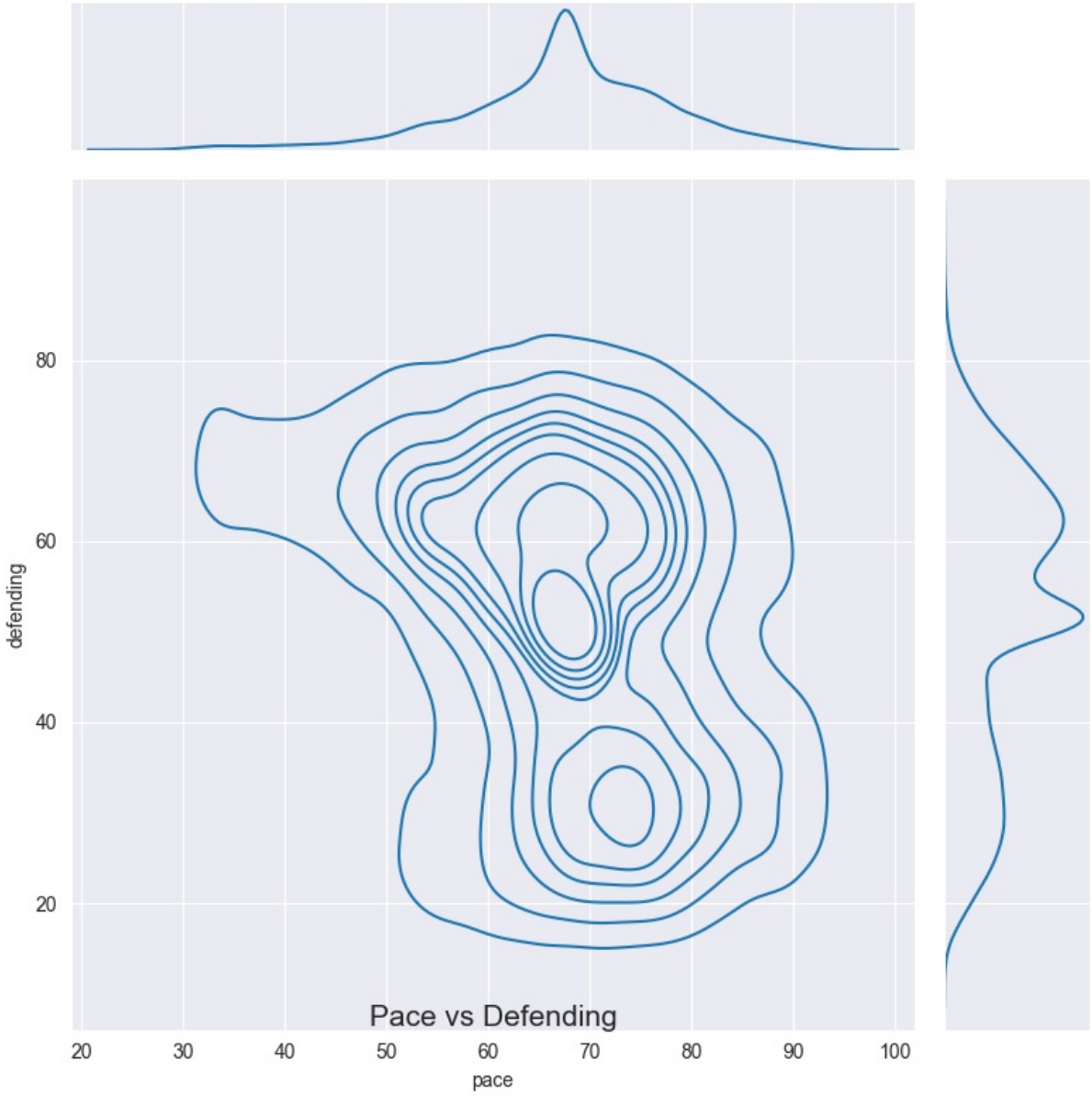
In [ ]:

```
# top 10 weightiest players:


wt=df.sort_values(by='weight_kg',ascending=False).head(10)


# Create the scatter plot
fig = go.Figure()

# Add scatter trace with customizations
fig.add_trace(go.Scatter(
    x=wt["short_name"],
    y=wt["weight_kg"],
    mode='markers',
    marker=dict(
        size=wt["weight_kg"],
        color=colors,
        colorscale='Blues',
        opacity=0.8,
        symbol='circle',
        line=dict(color='black', width=1)
    )
))

# Add colorbar to indicate the scale of the colors
fig.update_layout(coloraxis=dict(colorbar=dict(title='Color Scale', thickness=20)))

# Add axis labels and title
fig.update_layout(
    xaxis_title='Player Name',
    yaxis_title='Weight (kg)',
    title_text='top 10 weightiest players in FIFA 21',
    title_font=dict(size=24, family='Arial, sans-serif', color='gold'),
    width=1000,
    height=600,
    template='plotly_dark'
)

fig.show()
```

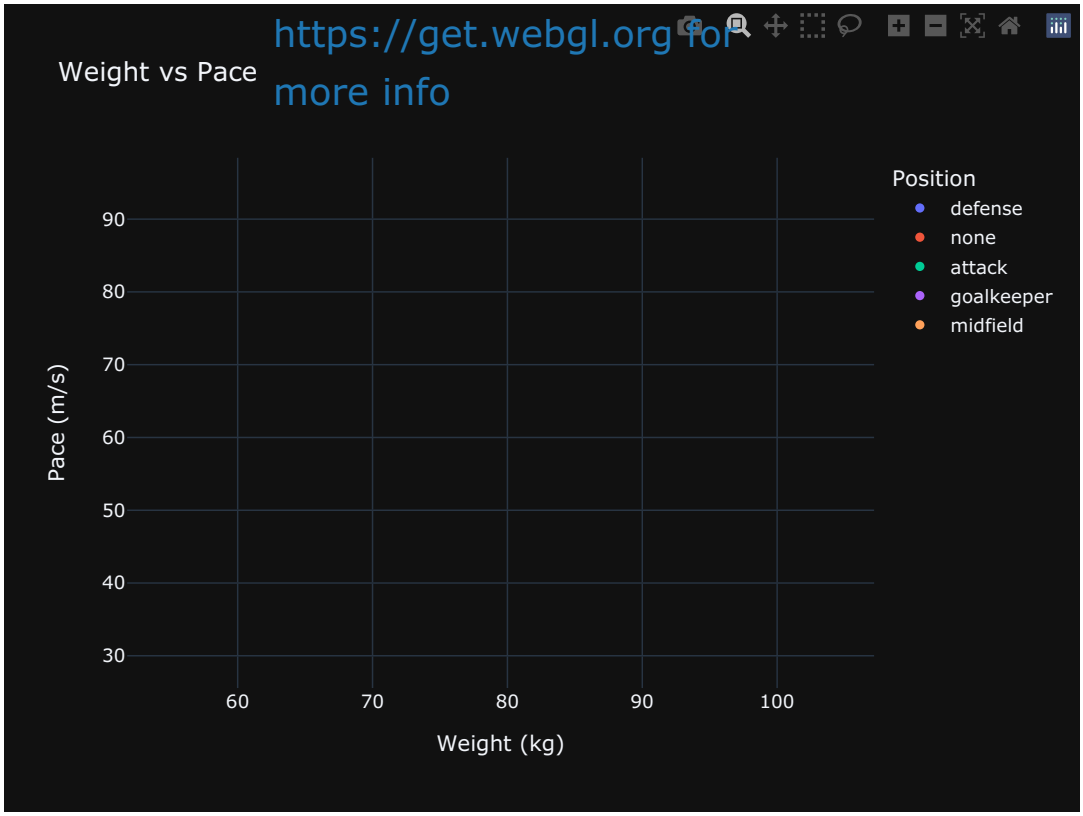


In [ ]:

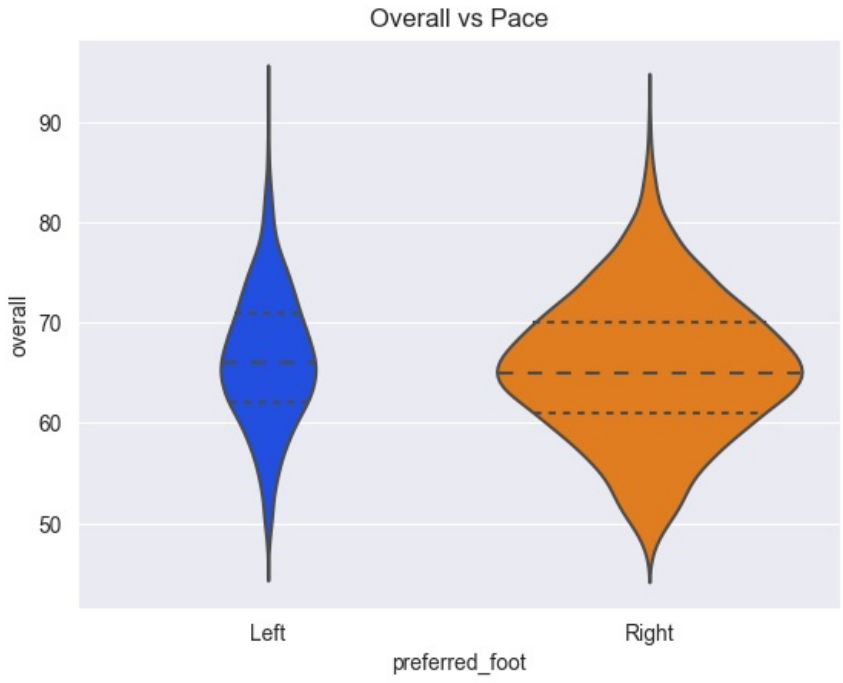
```
# PACE and DEFENDING:
sns.set_style("darkgrid")
sns.jointplot(x=df["pace"],y=df["defending"],kind="kde",palette="rocket",height=8)
plt.title("Pace vs Defending",fontsize=15,y=-0.01)
plt.show()
```



In [ ]:
```
# Weight vs Pace:
sam=df.sample(2000)
colors=["red","green","blue","yellow","orange","purple","pink","brown","black","grey"]
px.scatter(sam,x="weight_kg",y="pace",hover_data=["short_name"],color=sam["team_position"],title="Weight vs Pac
           labels={"weight_kg":"Weight (kg)","pace":"Pace (m/s)","team_position":"Position"},trendline="ols")
```



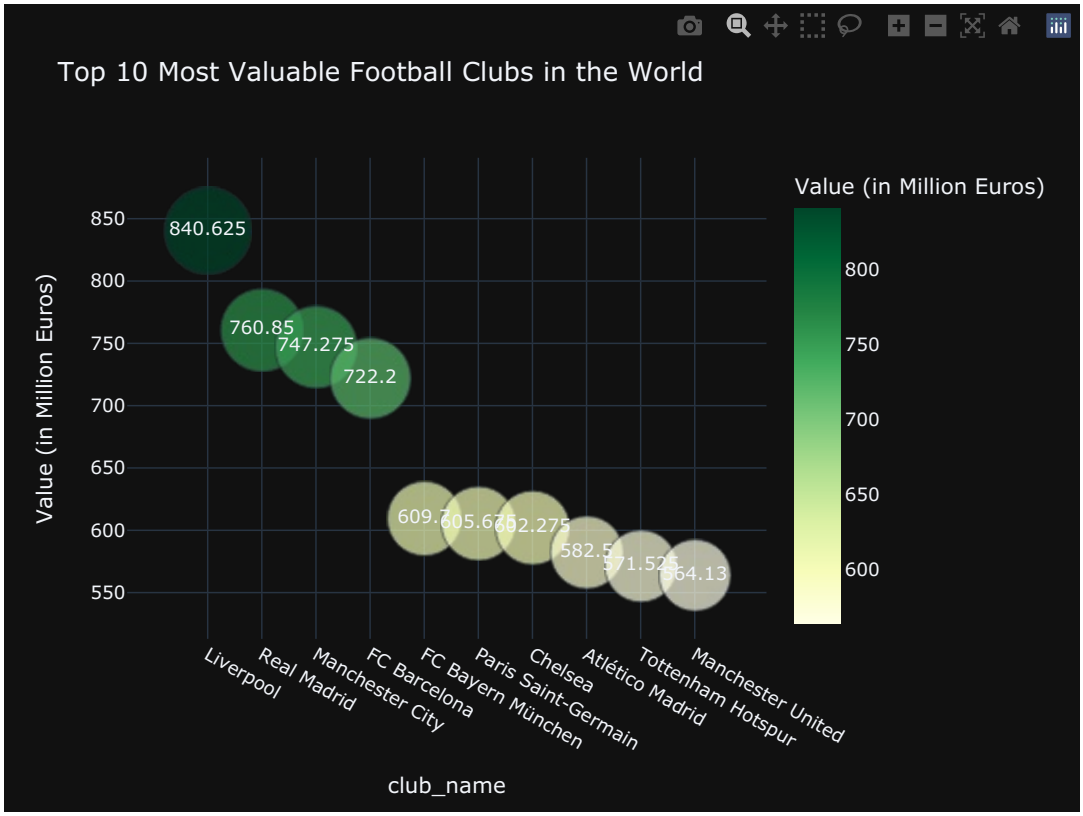
In [ ]:
```
fig=sns.violinplot(y="overall", x="preferred_foot", data=df, palette="bright", split=True, scale="count", inner
fig.set_title('Overall vs Pace')
fig.set_xlabel('preferred_foot')
fig.set_ylabel('overall')
plt.show()
```



In [ ]:
```
#top 10 player has the highest overall rating:
from wordcloud import WordCloud
ov= df.sort_values(by=['overall'],ascending=False).head(10)["short_name"]


plt.subplots(figsize=(8,8))
wordcloud = WordCloud(
                          background_color='white',
                          width=512,
                          height=384
                         ).generate(" ".join(ov))
plt.imshow(wordcloud)
plt.axis('off')
plt.savefig('graph.png')

plt.show()
```

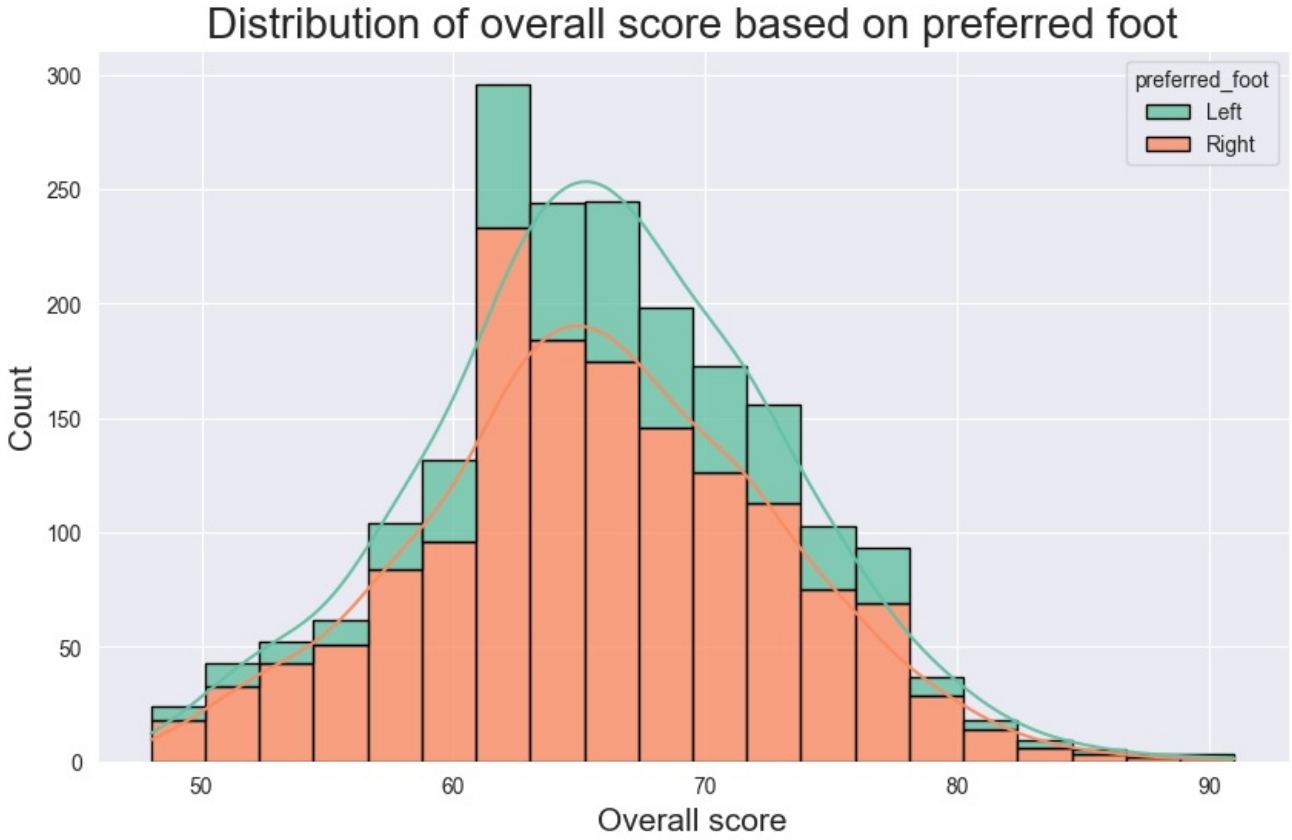
In [ ]:
```
# relationship between value and wage:
sam = df.sample(1000)

px.scatter(sam, y='value_eur_m', x='wage_eur_m', color='preferred_foot', title='Relationship between value and
           template="ggplot2",width=1000, height=600,labels={'value_eur_m':'Value (in millions)','wage_eur_m':'
```

In [ ]:
```
# most valuable 10 football clubs in the world:

vf=df.groupby("club_name")[["value_eur_m"]].sum().sort_values(by="value_eur_m",ascending=False).head(10)

px.scatter(vf,x=vf.index,y="value_eur_m",color="value_eur_m",size="value_eur_m",size_max=40,title="Top 10 Most
           template="plotly_dark",labels={"value_eur_m":"Value (in Million Euros)"},color_continuous_scale="ylg
```



In [ ]:
```
# distribution of overall score based on preferred foot:
sam=df.sample(2000)

fig=plt.figure(figsize=(10,6))
sns.histplot(sam,x="overall",hue="preferred_foot",multiple="stack",bins=20,kde=True,palette="Set2",alpha=0.8,ed
plt.title("Distribution of overall score based on preferred foot",fontsize=20)
plt.xlabel("Overall score",fontsize=15)
plt.ylabel("Count",fontsize=15)
```

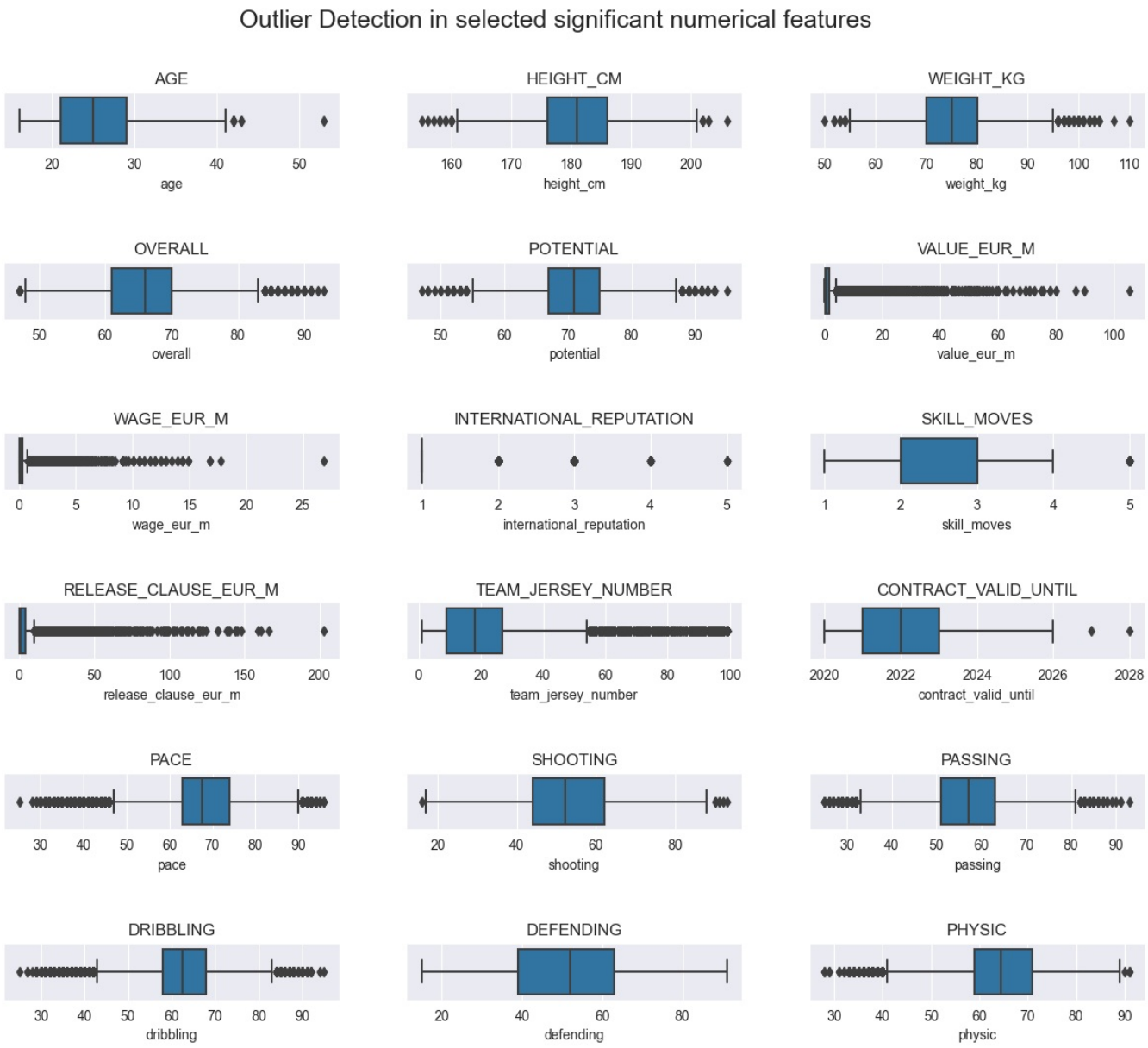
Out[ ]:
```
Text(0, 0.5, 'Count')
```



OUTLIER DETECTION

In [ ]:
```
# Selecting related features:
num=df.select_dtypes(include=['int64','float64'])
```

```
num.drop(["sofifa_id","league_rank","weak_foot"],axis=1,inplace=True)


fig, axs = plt.subplots(nrows=6, ncols=3, figsize=(15, 12))
plt.subplots_adjust(hspace=2)
fig.suptitle("Outlier Detection in selected significant numerical features", fontsize=18, y=0.95)

# loop through tickers and axes
for ticker, ax in zip(num.iloc[:,0:18], axs.ravel()):
    # filter df for ticker and plot on specified axes
    sns.boxplot(data=num, x=ticker, ax=ax)
    # chart formatting
    ax.set_title(ticker.upper())


plt.show()
```



In [ ]:
```
#  outlier threshold:
def outlier_thresholds(dataframe,col_name,q1=0.25,q3=0.75):
    q1=dataframe[col_name].quantile(q1)
    q3=dataframe[col_name].quantile(q3)
    IQR=q3-q1

    up_limit=q3+1.5*IQR
    low_limit=q1-1.5*IQR

    return low_limit,up_limit

# check outliers:
def check_outliers(dataframe,col_name):
    low_limit,up_limit=outlier_thresholds(dataframe,col_name)
    if dataframe[(dataframe[col_name]<low_limit) | (dataframe[col_name]>up_limit)].any(axis=None):
        return True

    else:
        return False
```

In [ ]:
```
# replacing outliers with thresholds:
```

```
def replace_with_thresholds(dataframe, col):
    low_limit, up_limit = outlier_thresholds(dataframe, col)
    if low_limit > 0:
        dataframe.loc[(dataframe[col] < low_limit), col] = low_limit
        dataframe.loc[(dataframe[col] > up_limit), col] = up_limit
    else:
        dataframe.loc[(dataframe[col] > up_limit), col] = up_limit

replace_with_thresholds(df,"age")

print("After filling with thresholds:")
print("Age:",check_outliers(df,"age"))
```

```
After filling with thresholds:
Age: False
```

DATA PREPROCESSING

In [ ]:
```
# label encoding:

df1=df.copy()
# if catorical column has 2 unique values, then use label encoder
# if categorical column has more than 2 unique values and less than 10 unique values, then use one hot encoder.
#df1.drop(["sofifa_id","player_url","short_name","long_name","dob","joined","club_name","league_name","player_p
def label_encoder(df1, column_name):
    if df1[column_name].dtype == 'object':
        if df1[column_name].nunique() <= 2:
            le = LabelEncoder()
            df1[column_name] = le.fit_transform(df1[column_name])
            return df1
        elif df1[column_name].nunique() > 2  and df1[column_name].nunique() <= 10:
            ohe = OneHotEncoder()
            ohe_df1 = pd.DataFrame(ohe.fit_transform(df1[[column_name]]).toarray())
            ohe_df1.columns = [column_name + "_" + str(i) for i in ohe_df1.columns]
            df1 = df1.join(ohe_df1)
            df1 = df1.drop(column_name, axis=1)
            return df1
        else:
            return df1

    else:
        return df1


for i in df1.columns:
    df1 = label_encoder(df1, i)
```

In [ ]:
```
X=df1.drop(["value_eur_m"],axis=1).values
y=df1["value_eur_m"].values.reshape(-1,1)

X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.2,random_state=42)

print("X_train shape: ",X_train.shape)
print("X_test shape: ",X_test.shape)
print("y_train shape: ",y_train.shape)
print("y_test shape: ",y_test.shape)
```

```
X_train shape:  (15155, 80)
X_test shape:  (3789, 80)
y_train shape:  (15155, 1)
y_test shape:  (3789, 1)
```

In [ ]:
```
# standization and normalization data

rb=RobustScaler()
X_train=rb.fit_transform(X_train)
X_test=rb.transform(X_test)
```

MODEL TRAINING AND EVALUATING

In [ ]:
```
# regression algorithms



lr=LinearRegression()
svr=SVR()
dt=DecisionTreeRegressor()
rf=RandomForestRegressor()
gr=GradientBoostingRegressor()
knn=KNeighborsRegressor()
gb=GaussianProcessRegressor()
```

```
models = [ lr, svr, dt, rf, gr, knn,gb]
overral=pd.DataFrame(columns=["Model","R2 Score",
                              "RMSE","MAE","MSE"])
for model in models:

    model.fit(X_train,y_train)
    y_pred=model.predict(X_test)
    r2=r2_score(y_test,y_pred)
    rmse=np.sqrt(mean_squared_error(y_test,y_pred))
    mae=mean_absolute_error(y_test,y_pred)
    mse=mean_squared_error(y_test,y_pred)


    overral=overral.append({"Model":model.__class__.__name__,
                            "R2 Score":r2,
                            "RMSE":rmse,
                            "MAE":mae,
                            "MSE":mse},ignore_index=True)


overral=overral.sort_values(by="R2 Score",ascending=False)
overral=overral.style.background_gradient(cmap="viridis")
overral
```

Out[ ]:

	Model	R2 Score	RMSE	MAE	MSE
4	GradientBoostingRegressor	0.985015	0.651072	0.183346	0.423894
3	RandomForestRegressor	0.982005	0.713455	0.124267	0.509018
2	DecisionTreeRegressor	0.970354	0.915755	0.160092	0.838608
0	LinearRegression	0.966599	0.972028	0.380998	0.944839
5	KNeighborsRegressor	0.960537	1.056548	0.415163	1.116293
1	SVR	0.659564	3.103228	0.454567	9.630023
6	GaussianProcessRegressor	-0.164612	5.739670	2.133242	32.943814

According to evaluation metrics, the best model is GradientBoostingRegressor with 0.985 R2 score and 0.65 RMSE score. I will use this model to predict the value of the players in the test dataset.

PCA / PRINCIPAL COMPONENT ANALYSINS

In [ ]:
```
from sklearn.decomposition import PCA

pca = PCA(n_components=2)
pca.fit(X_train)
X_train_pca = pca.transform(X_train)
X_test_pca = pca.transform(X_test)
```

In [ ]:
```
# model traning after PCA
lr=LinearRegression()
svr=SVR()
dt=DecisionTreeRegressor()
rf=RandomForestRegressor()
gr=GradientBoostingRegressor()
knn=KNeighborsRegressor()
gb=GaussianProcessRegressor()


models = [ lr, svr, dt, rf, gr, knn,gb]
overral=pd.DataFrame(columns=["Model","R2 Score",
                              "RMSE","MAE","MSE"])
for model in models:

    model.fit(X_train_pca,y_train)
    y_pred=model.predict(X_test_pca)
    r2=r2_score(y_test,y_pred)
    rmse=np.sqrt(mean_squared_error(y_test,y_pred))
    mae=mean_absolute_error(y_test,y_pred)
    mse=mean_squared_error(y_test,y_pred)


    overral=overral.append({"Model":model.__class__.__name__,
                            "R2 Score":r2,
                            "RMSE":rmse,
                            "MAE":mae,
                            "MSE":mse},ignore_index=True)


overral=overral.sort_values(by="R2 Score",ascending=False)
overral=overral.style.background_gradient(cmap="viridis")
overral
```

Out[ ]:

	Model	R2 Score	RMSE	MAE	MSE
4	GradientBoostingRegressor	0.891819	1.749331	0.724529	3.060159
5	KNeighborsRegressor	0.887130	1.786836	0.736437	3.192785
3	RandomForestRegressor	0.877282	1.863162	0.749468	3.471374
2	DecisionTreeRegressor	0.813468	2.297061	0.934373	5.276488
1	SVR	0.812307	2.304198	0.760785	5.309331
0	LinearRegression	0.126214	4.971628	2.531050	24.717087
6	GaussianProcessRegressor	-12339.495484	590.829968	25.921561	349080.050688

MODEL TUNING

In [ ]:
```
gb_params={'learning_rate': [0.01, 0.05, 0.1, 0.2, 0.3],
           'max_depth': [3, 5, 7, 9, 11],
           'min_samples_leaf': [1, 2, 3, 4, 5],
           'max_features': [0.1, 0.3, 0.5, 0.7, 0.9]}

gv=GridSearchCV(estimator=GradientBoostingRegressor(random_state=42),param_grid=gb_params,cv=5,n_jobs=-1,verbos
gv.fit(X_train_pca,y_train)

print(gv.best_params_)
print(gv.best_score_)
```

```
Fitting 5 folds for each of 625 candidates, totalling 3125 fits
{'learning_rate': 0.05, 'max_depth': 5, 'max_features': 0.1, 'min_samples_leaf': 3}
0.8868191962166199
```

In [ ]:
```
# final model:
gr=GradientBoostingRegressor(learning_rate=0.05,max_depth=5,max_features=0.1,min_samples_leaf=3,min_samples_spl
overral=pd.DataFrame(columns=["Model","R2 Score",
                              "RMSE","MAE","MSE"])


gr.fit(X_train_pca,y_train)
y_pred=gr.predict(X_test_pca)
r2=r2_score(y_test,y_pred)
rmse=np.sqrt(mean_squared_error(y_test,y_pred))
mae=mean_absolute_error(y_test,y_pred)
mse=mean_squared_error(y_test,y_pred)


overral=overral.append({"Model":model.__class__.__name__,
                            "R2 Score":r2,
                            "RMSE":rmse,
                            "MAE":mae,
                            "MSE":mse},ignore_index=True)


overral=overral.sort_values(by="R2 Score",ascending=False)
overral=overral.style.background_gradient(cmap="viridis")
overral
```

Out[ ]:

	Model	R2 Score	RMSE	MAE	MSE
0	GaussianProcessRegressor	0.892769	1.741635	0.724518	3.033291

CONCLUSION

- In this project, I aimed to predict the value of FIFA 21 players based on various features such as their age, overall rating, potential, and performance attributes. My predictive model achieved an impressive R2 score of 0.89, indicating that approximately 89% of the variance in the players' values can be explained by my model.