Football in Europe

October 16, 2021

0.1 Table of Contents

Introduction

This dataset describes football in europe for several seasons;

It has data about players, players' attributes, teams, matches and leagues.

I am curious to know about a few things, let me state them below:

1 - If I am a coach who is looking for the characteristics of a high attacking work rate player, what are the factors that I should look for?

2 - Do teams score more at their homeland or away across leagues?

3 - Do teams score more at their homeland or away across seasons?

4 - What's is the distribution of attacking work rate for each preferred foot? And which foot has higher attacking work rate?

[10]:
```
# Use this cell to set up import statements for all of the packages that you
#   plan to use.
import sqlite3;
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
import math

conn = sqlite3.connect('D:\Python\database.sqlite')

df = pd.read_sql_query("SELECT * FROM Player_Attributes", conn)
df1 = pd.read_sql_query("SELECT * FROM Player", conn)
```

```
df2 = pd.read_sql_query("SELECT * FROM Match", conn)
df3 = pd.read_sql_query("SELECT * FROM League", conn)
df4 = pd.read_sql_query("SELECT * FROM Team", conn)
df5 = pd.read_sql_query("SELECT * FROM Team_Attributes", conn)
df6 = pd.read_sql_query("SELECT * FROM Country", conn)

# Remember to include a 'magic word' so that your visualizations are plotted
#   inline with the notebook. See this page for more:
#   http://ipython.readthedocs.io/en/stable/interactive/magics.html
```

Data Wrangling & Cleaning

0.1.1 General Properties

[11]:

```
# merging Match, League and Country tables
df_country_league = df6.merge(df3, left_on='id',  right_on='country_id',␣
↪how='inner')
df_country_league.rename(columns= {'name_x': 'country_name', 'name_y':
↪'league','id_x':'id', 'id_y':'id'}, inplace=True)



#Removing duplicate columns
df_country_league.drop(['id','id'], axis=1, inplace=True)
df_country_league_match = df_country_league.
↪merge(df2,left_on='country_id',right_on='country_id',how='inner')
df_country_league_match.drop(['id'], axis=1, inplace=True)

#Removing unuseful columns
df_country_league_match.
↪drop(['home_player_X1','home_player_X2','home_player_X3','home_player_X4','home_player_X5',
↪inplace=True)
df_country_league_match.drop(df_country_league_match.iloc[:, 12:34], axis=1,␣
↪inplace=True)
df_country_league_match.drop(df_country_league_match.iloc[:,-38:], axis=1,␣
↪inplace=True)
```

[12]:

```
#converting date column data type to date
df_country_league_match['date']= pd.to_datetime(df_country_league_match['date'])
df_country_league_match.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 25979 entries, 0 to 25978
Data columns (total 34 columns):
 #   Column            Non-Null Count  Dtype
---  ------            --------------  -----
 0   country_name      25979 non-null  object
 1   country_id        25979 non-null  int64
 2   league            25979 non-null  object
```

```
 3   league_id         25979 non-null  int64
 4   season            25979 non-null  object
 5   stage             25979 non-null  int64
 6   date              25979 non-null  datetime64[ns]
 7   match_api_id      25979 non-null  int64
 8   home_team_api_id  25979 non-null  int64
 9   away_team_api_id  25979 non-null  int64
 10  home_team_goal    25979 non-null  int64
 11  away_team_goal    25979 non-null  int64
 12  home_player_1     24755 non-null  float64
 13  home_player_2     24664 non-null  float64
 14  home_player_3     24698 non-null  float64
 15  home_player_4     24656 non-null  float64
 16  home_player_5     24663 non-null  float64
 17  home_player_6     24654 non-null  float64
 18  home_player_7     24752 non-null  float64
 19  home_player_8     24670 non-null  float64
 20  home_player_9     24706 non-null  float64
 21  home_player_10    24543 non-null  float64
 22  home_player_11    24424 non-null  float64
 23  away_player_1     24745 non-null  float64
 24  away_player_2     24701 non-null  float64
 25  away_player_3     24686 non-null  float64
 26  away_player_4     24658 non-null  float64
 27  away_player_5     24644 non-null  float64
 28  away_player_6     24666 non-null  float64
 29  away_player_7     24744 non-null  float64
 30  away_player_8     24638 non-null  float64
 31  away_player_9     24651 non-null  float64
 32  away_player_10    24538 non-null  float64
 33  away_player_11    24425 non-null  float64
dtypes: datetime64[ns](1), float64(22), int64(8), object(3)
memory usage: 6.9+ MB
```

[13]:
```python
#Merging players with players' attributes
df_players = df1.merge(df,left_on='player_api_id', right_on='player_api_id',␣
↪how='inner')
df_players.drop(['id_y','player_fifa_api_id_y'], axis=1,inplace=True)
to_be_dropped = ['None','y','norm','stoc']
```

[149]:
```python
#dropping inaccurate rows
df_players = df_players.query('attacking_work_rate not in␣
↪[None,"y","norm","stoc","le"]')
df_players.attacking_work_rate.unique()
```

[149]:
```
array(['high', 'low', 'medium'], dtype=object)
```

Exploratory Data Analysis

[41]:
```
#General exploration of the dataset to identify usable columns and null values
df_country_league_match.hist(figsize = (20,15));
```

[171]:
```
#General exploration of the dataset to identify usable columns and null values
df_players.hist(figsize = (20,15));
```

[177]:
```
df_players.describe()
```

[177]:
```
                id_x  player_api_id  player_fifa_api_id_x         height  \
count  176462.000000  176462.000000         176462.000000  176462.000000
mean     5512.387727  138311.692115         167332.695657     181.879399
std      3192.412054  137876.007387          52353.997365       6.408034
min         1.000000    2625.000000              2.000000     157.480000
25%      2742.000000   35497.000000         157305.000000     177.800000
50%      5523.000000   91503.000000         183900.000000     182.880000
75%      8249.000000  193176.000000         200271.000000     185.420000
max     11075.000000  750584.000000         234141.000000     208.280000

              weight  overall_rating      potential       crossing  \
count  176462.000000   176462.000000  176462.000000  176462.000000
mean      168.765428       68.683178      73.513612      55.213825
std        15.130114        7.026800       6.581890      17.256186
min       117.000000       33.000000      39.000000       1.000000
25%       159.000000       64.000000      69.000000      45.000000
50%       168.000000       69.000000      74.000000      59.000000
75%       179.000000       73.000000      78.000000      68.000000
max       243.000000       94.000000      97.000000      95.000000
```

```
         finishing  heading_accuracy  …         vision     penalties  \
count  176462.000000     176462.000000  …  176462.000000  176462.000000
mean       50.046299         57.256707  …      57.886871      54.938927
std        19.030948         16.490416  …      15.160865      15.554081
min         1.000000          1.000000  …       1.000000       2.000000
25%        34.000000         49.000000  …      49.000000      45.000000
50%        53.000000         60.000000  …      60.000000      57.000000
75%        65.000000         68.000000  …      69.000000      67.000000
max        97.000000         98.000000  …      97.000000      96.000000

           marking  standing_tackle  sliding_tackle     gk_diving  \
count  176462.000000    176462.000000   176462.000000  176462.000000
mean       46.720739        50.335143       48.019404      14.712148
std        21.238263        21.516804       21.615942      16.852731
min         1.000000         1.000000        2.000000       1.000000
25%        25.000000        29.000000       25.000000       7.000000
50%        50.000000        56.000000       53.000000      10.000000
75%        66.000000        69.000000       67.000000      13.000000
max        94.000000        95.000000       95.000000      94.000000

        gk_handling     gk_kicking  gk_positioning   gk_reflexes
count  176462.000000  176462.000000    176462.00000  176462.000000
mean       15.885448      20.243339        15.95586      16.268324
std        15.852140      20.949841        16.08037      17.205300
min         1.000000       1.000000         1.00000       1.000000
25%         8.000000       8.000000         8.00000       8.000000
50%        11.000000      11.000000        11.00000      11.000000
75%        15.000000      15.000000        15.00000      15.000000
max        93.000000      97.000000        96.00000      96.000000

[8 rows x 40 columns]
```

0.1.2

0.2 Question 1

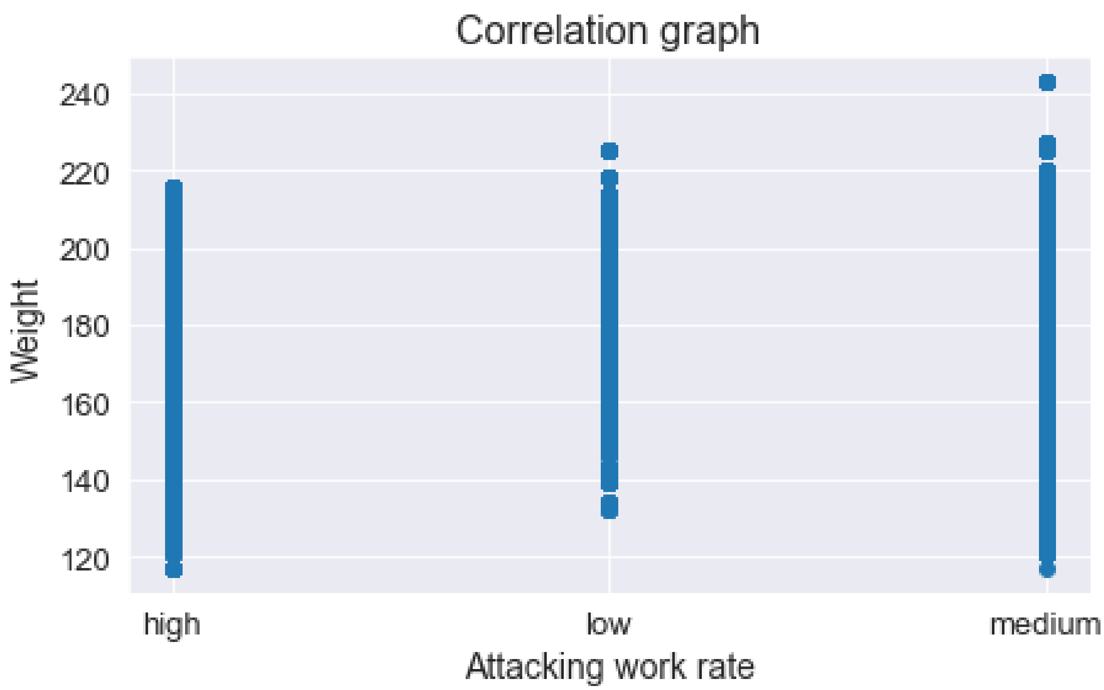
0.3 If I am a coach who is looking for the characteristics of a high attacking work rate player,

0.4 What are the factors that I should look for?

[160]:
```python
# df_players['num_attacking_work_rate'].unique()
def scatter_plot(x,y,xlabel,ylabel,title):
    plt.figure(figsize=(7,4))
    sns.set_style('darkgrid')
    plt.scatter(x,y)
    plt.xticks(size='13',rotation='horizontal')
```

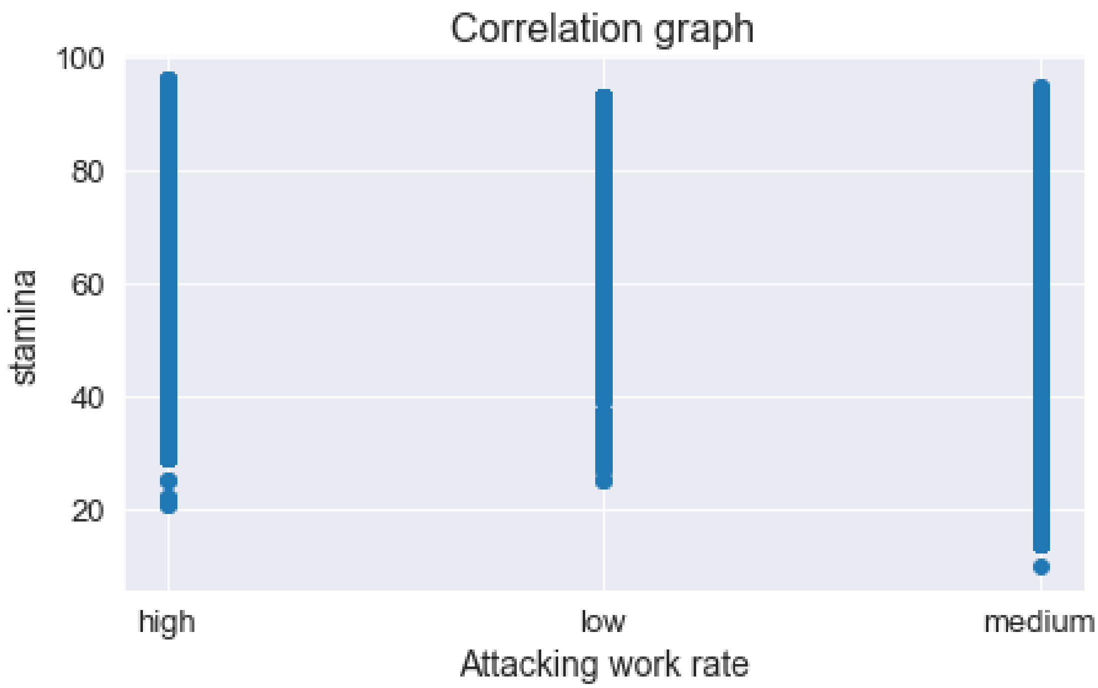
```
    plt.yticks(size='13')
    plt.xlabel(xlabel,size='14')
    plt.ylabel(ylabel,size='14')
    plt.title(title,size='16')
    return

scatter_plot(df_players['attacking_work_rate'],df_players['weight'],'Attacking␣
↪work rate','Weight','Correlation graph')
```



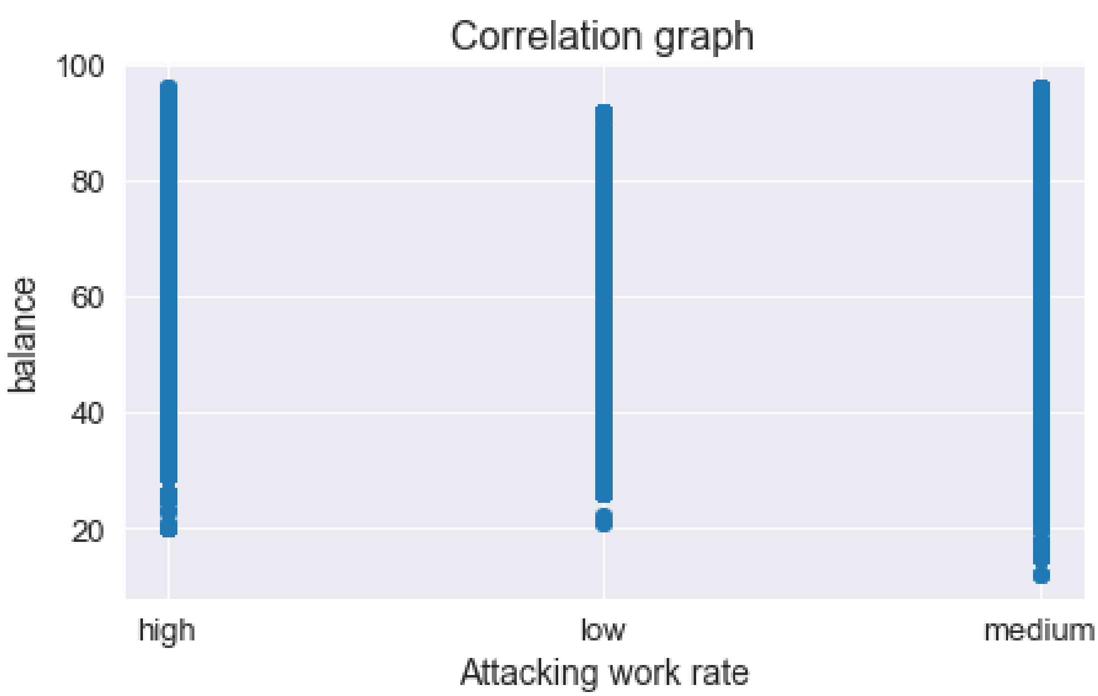
0.4.1 It seems that players with less weight tend to have more attacking work rate

[162]:
```
scatter_plot(df_players['attacking_work_rate'],df_players['stamina'],'Attacking␣
↪work rate','stamina','Correlation graph')
```



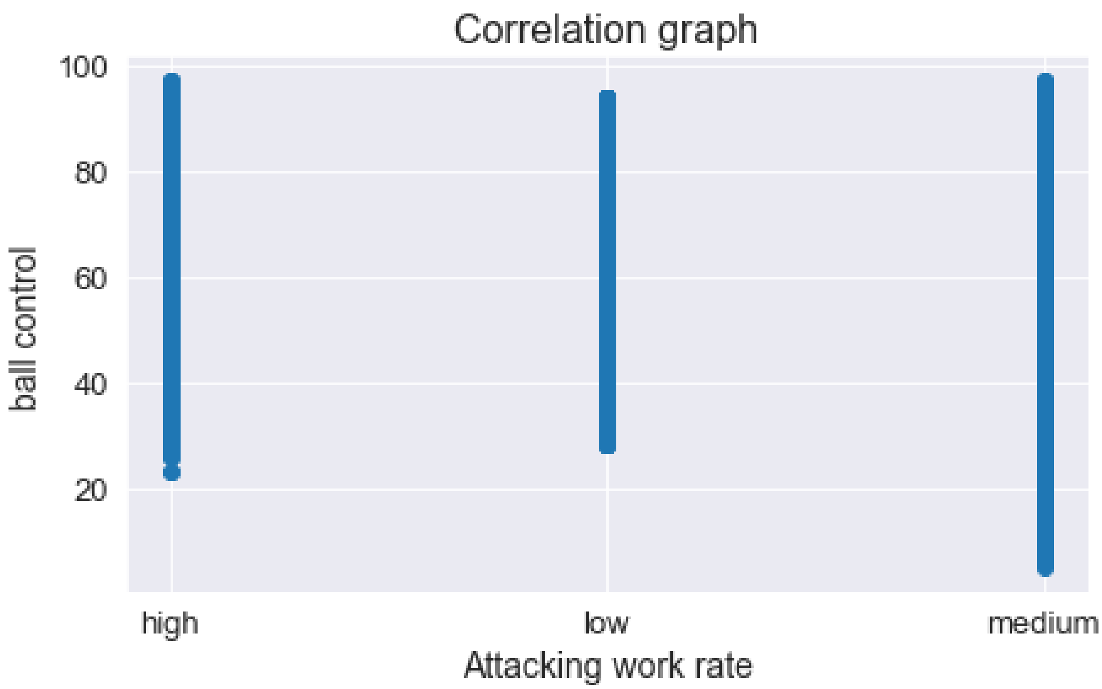
0.4.2 Also, stamina seems to be a factor that correlates with high attacking work rate

[163]:
```
scatter_plot(df_players['attacking_work_rate'],df_players['balance'],'Attacking␣
↪work rate','balance','Correlation graph')
```



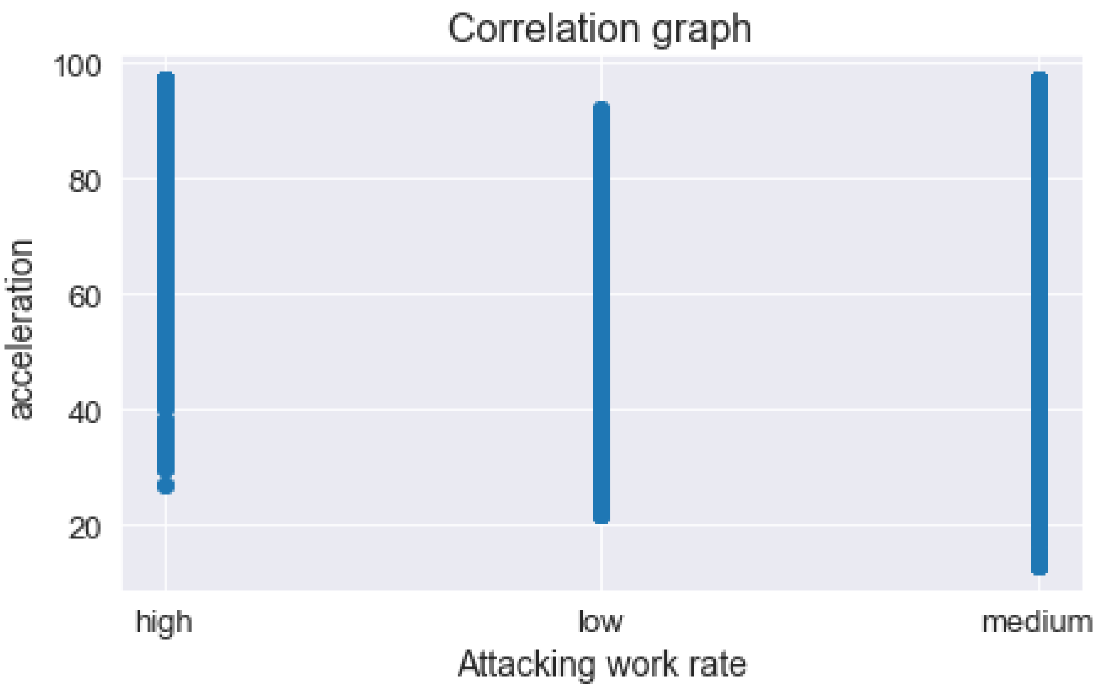
0.4.3 Players with high attacking work rate seems to have higher balance; makes sense as the play under pressure.

[165]:
```
scatter_plot(df_players['attacking_work_rate'],df_players['ball_control'],'Attacking␣
↪work rate','ball control','Correlation graph')
```



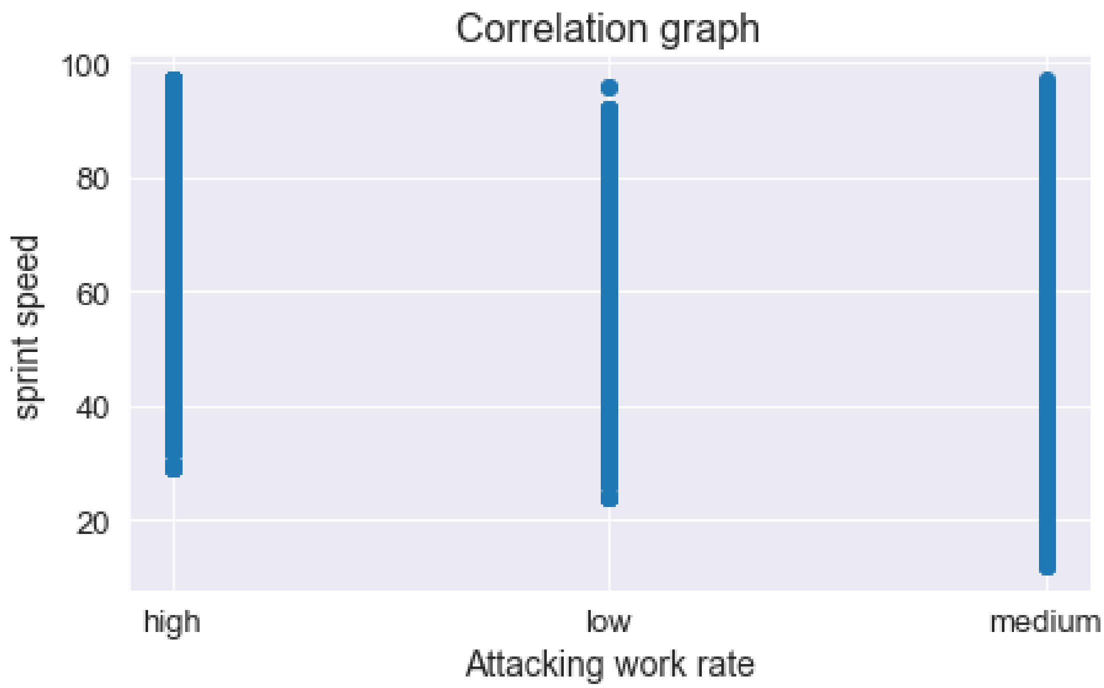
0.4.4 Ball control is higher, also, seems to be relatively higher for high attacking work rate players

[168]:
```
scatter_plot(df_players['attacking_work_rate'],df_players['acceleration'],'Attacking␣
↪work rate','acceleration','Correlation graph')
```



0.4.5 It seems that players with higher acceleration tend to have more attacking work rate

[166]:
```
scatter_plot(df_players['attacking_work_rate'],df_players['sprint_speed'],'Attacking␣
↪work rate','sprint speed','Correlation graph')
```



0.4.6 It seems that players with higher sprint speed tend to have more attacking work rate

0.5 Those are the factors a coach should look for while hunting a player with high attacking work rate!

0.5.1 ---

0.6 Question 2

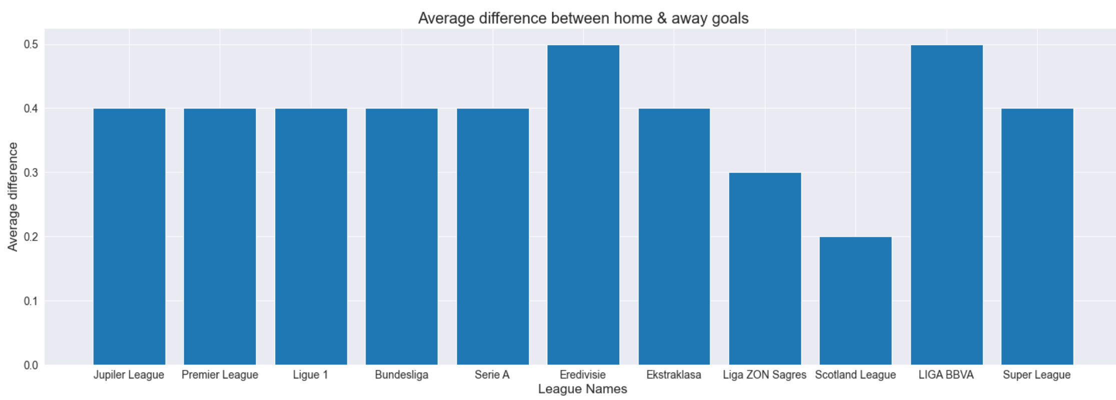
0.6.1 Do teams score more at their homeland or away across leagues?

[15]:
```
df_country_league_match['goals_difference'] =␣
↪df_country_league_match['home_team_goal'] -␣
↪df_country_league_match['away_team_goal']
average_goals_difference = df_country_league_match.
↪groupby(['league'])['goals_difference'].mean().round(1)
league_names = ['Jupiler League', 'Premier League', 'Ligue 1', 'Bundesliga',␣
↪'Serie A',
                'Eredivisie', 'Ekstraklasa', 'Liga ZON Sagres', 'Scotland␣
↪League',
                'LIGA BBVA', 'Super League']
```

[167]:
```
def display_barplot(x,y, xlabel, ylabel, title):
    """ A function that utilizes inputs to visualize them with certain standards
        x = X-axis
```

```
        y = Y-axis
        xlabel = X-axis title
        ylabel = Y-axis title
        title = Chart's title
    """
    plt.figure(figsize=(25,8))
    sns.set_style('darkgrid')
    plt.bar(x,y)
    plt.xticks(size='14',rotation='horizontal')
    plt.yticks(size='14')
    plt.xlabel(xlabel,size='17')
    plt.ylabel(ylabel,size='17')
    plt.title(title,size='20');
    return

display_barplot(league_names,average_goals_difference,'League Names','Average␣
↪difference','Average difference between home & away goals')
```



0.7 Answer is Homeland

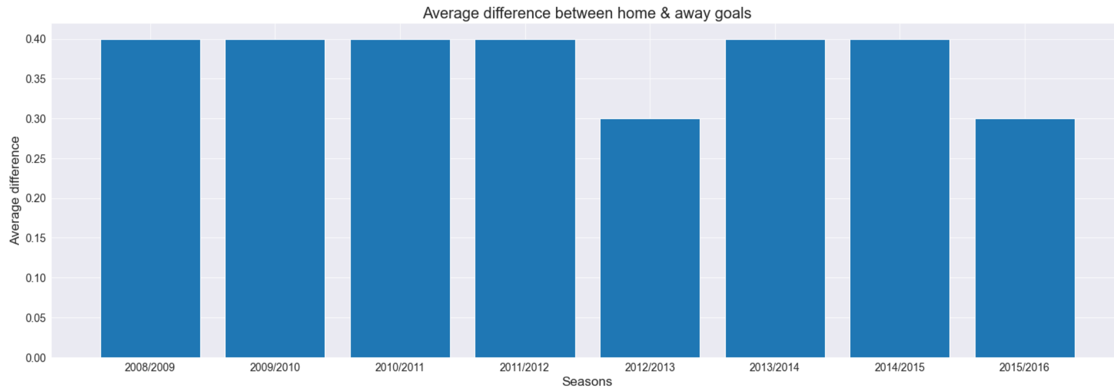
0.7.1 ---

0.8 Question 3

0.8.1 Do teams score more at their homeland or away across seasons?

[169]:
```
# Continue to explore the data to address your additional research
#   questions. Add more headers as needed if you have more questions to
#   investigate.
seasons = ['2008/2009', '2009/2010', '2010/2011', '2011/2012', '2012/2013',
       '2013/2014', '2014/2015', '2015/2016']
average_goals_difference_season = df_country_league_match.
↪groupby(['season'])['goals_difference'].mean().round(1)
```

```
display_barplot(seasons,average_goals_difference_season,'Seasons','Average␣
↪difference','Average difference between home & away goals')
```



0.9 Answer is Homeland as well

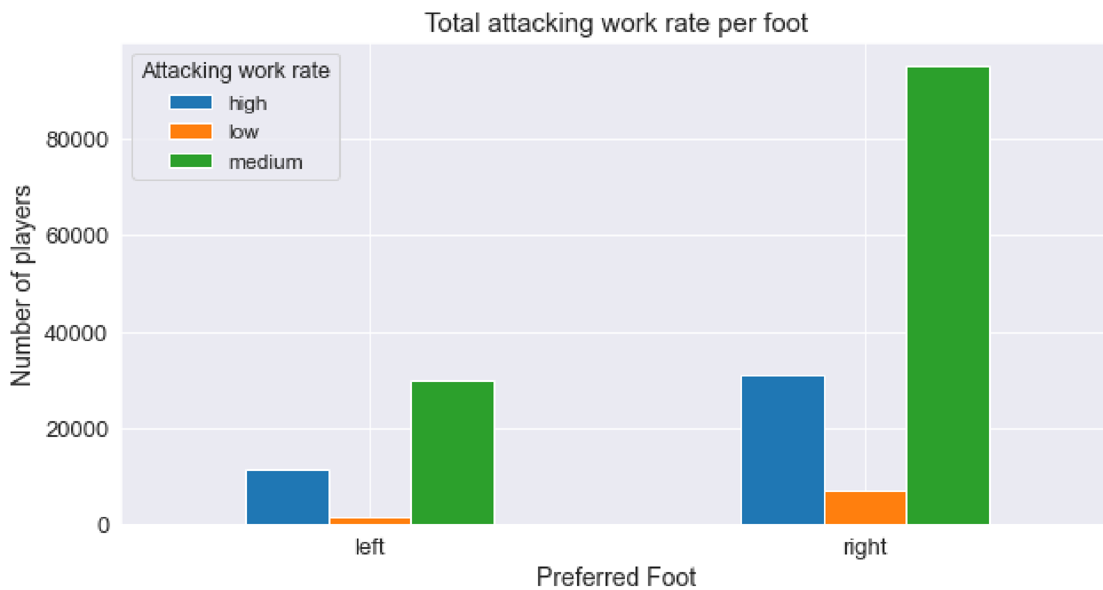
0.9.1 ——————————————————————————-

0.10 Question 4

0.11 What's is the distribution of attacking work rate for each preferred foot?

0.12 And which foot has higher attacking work rate?

[170]:
```
pd.crosstab(df_players['preferred_foot'],df_players['attacking_work_rate']).
↪plot.bar(figsize=(10,5));
sns.set_style('darkgrid');
plt.xticks(size='13',rotation='horizontal');
plt.yticks(size='13');
plt.xlabel('Preferred Foot',size='14');
plt.ylabel('Number of players',size='14');
plt.title('Total attacking work rate per foot',size='15');
plt.legend(loc='upper left',prop={'size': 12}, title='Attacking work rate',␣
↪title_fontsize='12.5');
```



0.12.1 Using proportions:

- Left footed players tend to have a higher percentage of medium & high attacking work rate with approximately 71% and 23%

- While right footed tend to have a higher percentage of low attacking rate with approximately 7.1%

0.12.2 ——————————————————————————-

Conclusions

1 - Weight, sprint speed, acceleration, balance, ball control and stamina seems to be higher for players with high attacking rate, so, coaches should look for it.

2 & 3 - Across all leagues and all seasons, teams tend to score more at their homeland.

4- left footed players tend to have a higher percentage of high & medium attacking work rate with approximately 71% and 23%, while right footed tend to have a higher percentage of low attacking rate with approximately 7.1%

0.12.3 Challenges & feedback:

- The dataset had many null values, especially in the attacking rate column. however, we still have a decent amount of inputs to have an accurate result.

- It would have been useful if it was possible to merge teams with df_players to see if teams with higher attacking work rate score more? or teams with higher defense work rate tend to receive less goal? what are the factors that lead to higher stages?

- The data provided was enough to answer my questions though.

- I believe hypothesis would be more accurate in identifying factors affecting high attacking work rate.