In [1]:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn import metrics
from sklearn.metrics import accuracy_score
from sklearn.feature_selection import SelectKBest, chi2, f_regression
from sklearn.neighbors import NearestNeighbors
from sklearn.neighbors import KNeighborsClassifier
from sklearn import svm
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import KFold
from sklearn.ensemble import GradientBoostingRegressor
from sklearn.metrics import r2_score
from sklearn.model_selection import GridSearchCV
```

In [2]:

```
df_results = pd.read_csv("results-0.2.csv")
```

In [3]:

```
df_results.head(21)
```

Out[3]:

	League	Year	position	team	matches	wins	draws	loses	scored	missed	p
0	Bundesliga	2014	1	Bayern Munich	34	25	4	5	80	18	
1	Bundesliga	2014	2	Wolfsburg	34	20	9	5	72	38	
2	Bundesliga	2014	3	Borussia M.Gladbach	34	19	9	6	53	26	
3	Bundesliga	2014	4	Bayer Leverkusen	34	17	10	7	62	37	
4	Bundesliga	2014	5	Augsburg	34	15	4	15	43	43	
5	Bundesliga	2014	6	Schalke 04	34	13	9	12	42	40	
6	Bundesliga	2014	7	Borussia Dortmund	34	13	7	14	47	42	
7	Bundesliga	2014	8	Hoffenheim	34	12	8	14	49	55	
8	Bundesliga	2014	9	Werder Bremen	34	11	10	13	50	65	
9	Bundesliga	2014	10	Eintracht Frankfurt	34	11	10	13	56	62	
10	Bundesliga	2014	11	FC Cologne	34	9	13	12	34	40	
11	Bundesliga	2014	12	Mainz 05	34	9	13	12	45	47	
12	Bundesliga	2014	13	Hannover 96	34	9	10	15	40	56	
13	Bundesliga	2014	14	VfB Stuttgart	34	9	9	16	42	60	
14	Bundesliga	2014	15	Hamburger SV	34	9	8	17	25	50	
15	Bundesliga	2014	16	Hertha Berlin	34	9	8	17	36	52	
16	Bundesliga	2014	17	Freiburg	34	7	13	14	36	47	
17	Bundesliga	2014	18	Paderborn	34	7	10	17	31	65	
18	Bundesliga	2015	1	Bayern Munich	34	28	4	2	80	17	
19	Bundesliga	2015	2	Borussia Dortmund	34	24	6	4	82	34	
20	Bundesliga	2015	3	Bayer Leverkusen	34	18	6	10	56	40	

In [4]:

```
df_results.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 570 entries, 0 to 569
Data columns (total 11 columns):
League      570 non-null object
Year        570 non-null int64
position    570 non-null int64
team        570 non-null object
matches     570 non-null int64
wins        570 non-null int64
draws       570 non-null int64
loses       570 non-null int64
scored      570 non-null int64
missed      570 non-null int64
pts         570 non-null int64
dtypes: int64(9), object(2)
memory usage: 49.1+ KB
```

In [5]:

```
df_results['League'].unique()
```

Out[5]:

```
array(['Bundesliga', 'EPL', 'La_liga', 'Ligue_1', 'RFPL', 'Serie_A'],
      dtype=object)
```

In [6]:

```
df_results.describe()
```

Out[6]:

	Year	position	matches	wins	draws	loses	scored
count	570.000000	570.000000	570.000000	570.000000	570.000000	570.000000	570.000000
mean	2016.000000	10.061404	36.245614	13.531579	9.182456	13.531579	48.385965
std	1.415456	5.580982	2.906152	5.935200	2.927064	5.540700	17.634599
min	2014.000000	1.000000	30.000000	2.000000	2.000000	1.000000	13.000000
25%	2015.000000	5.000000	34.000000	9.000000	7.000000	10.000000	36.000000
50%	2016.000000	10.000000	38.000000	12.000000	9.000000	14.000000	45.000000
75%	2017.000000	15.000000	38.000000	17.000000	11.000000	17.000000	56.000000
max	2018.000000	20.000000	38.000000	32.000000	18.000000	29.000000	118.000000

In [7]:

```
df_results['LeagueFill'] = df_results['League'].map({'Bundesliga':0, 'EPL':1, 'La_liga'
:2, 'Ligue_1':3,
                                                         'RFPL': 4, 'Serie_A':5})
```

In [8]:

```
team_list = df_results['team'].unique()
numer_list = []

count = df_results['team'].nunique()

for i in range(count):
    numer_list.append(i+1)

mapped_frame = pd.DataFrame(team_list)
number_frame = pd.DataFrame(numer_list)

zipObj = zip(team_list, numer_list)
diction = dict(zipObj)
df_results['TeamFill'] = df_results['team'].map(diction)
df_results.head(30)
```

Out[8]:

	League	Year	position	team	matches	wins	draws	loses	scored	missed	p
0	Bundesliga	2014	1	Bayern Munich	34	25	4	5	80	18	
1	Bundesliga	2014	2	Wolfsburg	34	20	9	5	72	38	
2	Bundesliga	2014	3	Borussia M.Gladbach	34	19	9	6	53	26	
3	Bundesliga	2014	4	Bayer Leverkusen	34	17	10	7	62	37	
4	Bundesliga	2014	5	Augsburg	34	15	4	15	43	43	
5	Bundesliga	2014	6	Schalke 04	34	13	9	12	42	40	
6	Bundesliga	2014	7	Borussia Dortmund	34	13	7	14	47	42	
7	Bundesliga	2014	8	Hoffenheim	34	12	8	14	49	55	
8	Bundesliga	2014	9	Werder Bremen	34	11	10	13	50	65	
9	Bundesliga	2014	10	Eintracht Frankfurt	34	11	10	13	56	62	
10	Bundesliga	2014	11	FC Cologne	34	9	13	12	34	40	
11	Bundesliga	2014	12	Mainz 05	34	9	13	12	45	47	
12	Bundesliga	2014	13	Hannover 96	34	9	10	15	40	56	
13	Bundesliga	2014	14	VfB Stuttgart	34	9	9	16	42	60	
14	Bundesliga	2014	15	Hamburger SV	34	9	8	17	25	50	
15	Bundesliga	2014	16	Hertha Berlin	34	9	8	17	36	52	
16	Bundesliga	2014	17	Freiburg	34	7	13	14	36	47	
17	Bundesliga	2014	18	Paderborn	34	7	10	17	31	65	
18	Bundesliga	2015	1	Bayern Munich	34	28	4	2	80	17	
19	Bundesliga	2015	2	Borussia Dortmund	34	24	6	4	82	34	
20	Bundesliga	2015	3	Bayer Leverkusen	34	18	6	10	56	40	
21	Bundesliga	2015	4	Borussia M.Gladbach	34	17	4	13	67	50	
22	Bundesliga	2015	5	Schalke 04	34	15	7	12	51	49	
23	Bundesliga	2015	6	Mainz 05	34	14	8	12	46	42	
24	Bundesliga	2015	7	Hertha Berlin	34	14	8	12	42	42	
25	Bundesliga	2015	8	Wolfsburg	34	12	9	13	47	49	
26	Bundesliga	2015	9	FC Cologne	34	10	13	11	38	42	
27	Bundesliga	2015	10	Hamburger SV	34	11	8	15	40	46	

	League	Year	position	team	matches	wins	draws	loses	scored	missed	p
28	Bundesliga	2015	11	Ingolstadt	34	10	10	14	33	42	4
29	Bundesliga	2015	12	Darmstadt	34	9	11	14	38	53	3

In [9]:

```
df_bundes_2014 = df_results.loc[(df_results['League'] == 'Bundesliga')&(df_results['Year']==2014)]
df_bundes_2015 = df_results.loc[(df_results['League'] == 'Bundesliga')&(df_results['Year']==2015)]
df_bundes_2016 = df_results.loc[(df_results['League'] == 'Bundesliga')&(df_results['Year']==2016)]
df_bundes_2017 = df_results.loc[(df_results['League'] == 'Bundesliga')&(df_results['Year']==2017)]
df_bundes_2018 = df_results.loc[(df_results['League'] == 'Bundesliga')&(df_results['Year']==2018)]

df_epl_2014 = df_results.loc[(df_results['League'] == 'EPL')&(df_results['Year']==2014)]
df_epl_2015 = df_results.loc[(df_results['League'] == 'EPL')&(df_results['Year']==2015)]
df_epl_2016 = df_results.loc[(df_results['League'] == 'EPL')&(df_results['Year']==2016)]
df_epl_2017 = df_results.loc[(df_results['League'] == 'EPL')&(df_results['Year']==2017)]
df_epl_2018 = df_results.loc[(df_results['League'] == 'EPL')&(df_results['Year']==2018)]

df_laliga_2014 = df_results.loc[(df_results['League'] == 'La_liga')&(df_results['Year']==2014)]
df_laliga_2015 = df_results.loc[(df_results['League'] == 'La_liga')&(df_results['Year']==2015)]
df_laliga_2016 = df_results.loc[(df_results['League'] == 'La_liga')&(df_results['Year']==2016)]
df_laliga_2017 = df_results.loc[(df_results['League'] == 'La_liga')&(df_results['Year']==2017)]
df_laliga_2018 = df_results.loc[(df_results['League'] == 'La_liga')&(df_results['Year']==2018)]

df_ligue1_2014 = df_results.loc[(df_results['League'] == 'Ligue_1')&(df_results['Year']==2014)]
df_ligue1_2015 = df_results.loc[(df_results['League'] == 'Ligue_1')&(df_results['Year']==2015)]
df_ligue1_2016 = df_results.loc[(df_results['League'] == 'Ligue_1')&(df_results['Year']==2016)]
df_ligue1_2017 = df_results.loc[(df_results['League'] == 'Ligue_1')&(df_results['Year']==2017)]
df_ligue1_2018 = df_results.loc[(df_results['League'] == 'Ligue_1')&(df_results['Year']==2018)]

df_RFPL_2014 = df_results.loc[(df_results['League'] == 'RFPL')&(df_results['Year']==2014)]
df_RFPL_2015 = df_results.loc[(df_results['League'] == 'RFPL')&(df_results['Year']==2015)]
df_RFPL_2016 = df_results.loc[(df_results['League'] == 'RFPL')&(df_results['Year']==2016)]
df_RFPL_2017 = df_results.loc[(df_results['League'] == 'RFPL')&(df_results['Year']==2017)]
df_RFPL_2018 = df_results.loc[(df_results['League'] == 'RFPL')&(df_results['Year']==2018)]

df_SerieA_2014 = df_results.loc[(df_results['League'] == 'Serie_A')&(df_results['Year']==2014)]
df_SerieA_2015 = df_results.loc[(df_results['League'] == 'Serie_A')&(df_results['Year']==2015)]
```

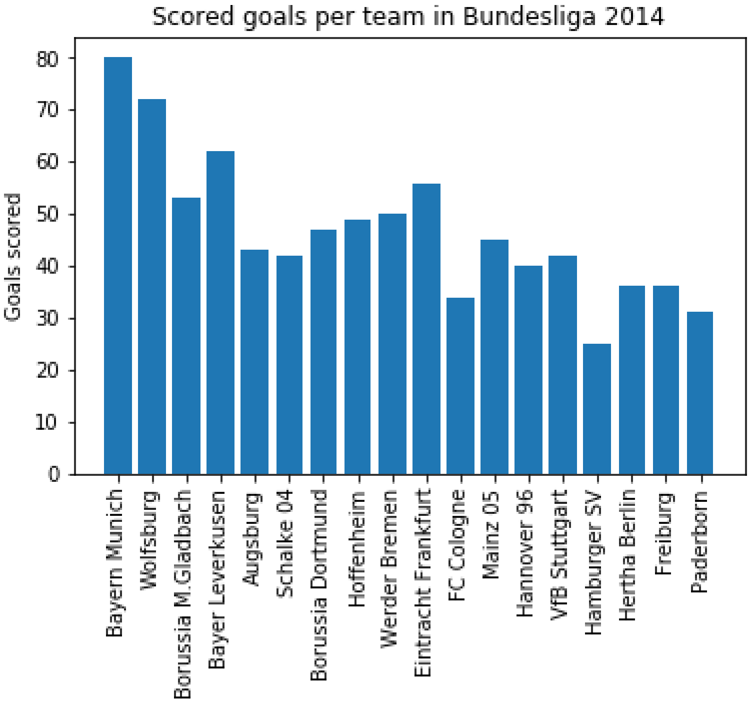
```
df_SerieA_2016 = df_results.loc[(df_results['League'] == 'Serie_A')&(df_results['Year']
==2016)]
df_SerieA_2017 = df_results.loc[(df_results['League'] == 'Serie_A')&(df_results['Year']
==2017)]
df_SerieA_2018 = df_results.loc[(df_results['League'] == 'Serie_A')&(df_results['Year']
==2018)]
```

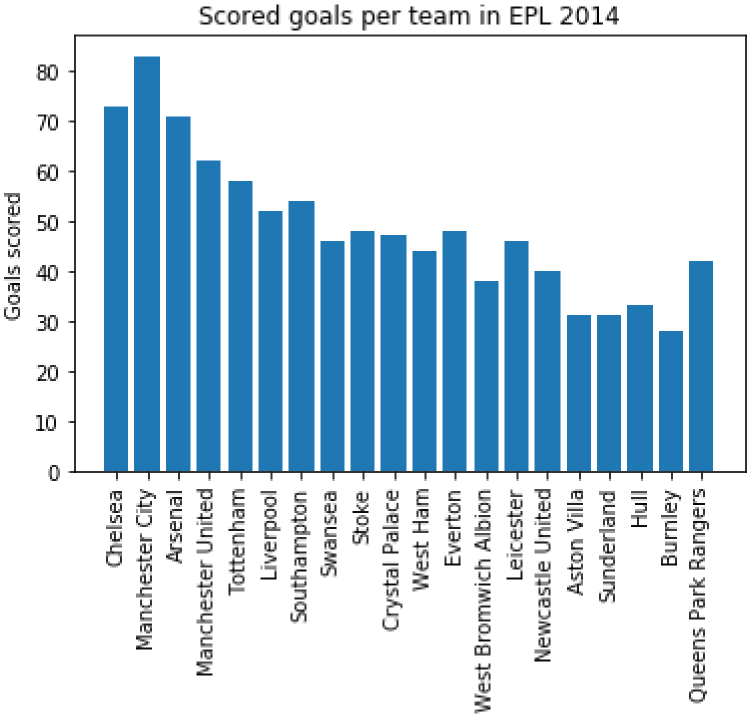
In [10]:

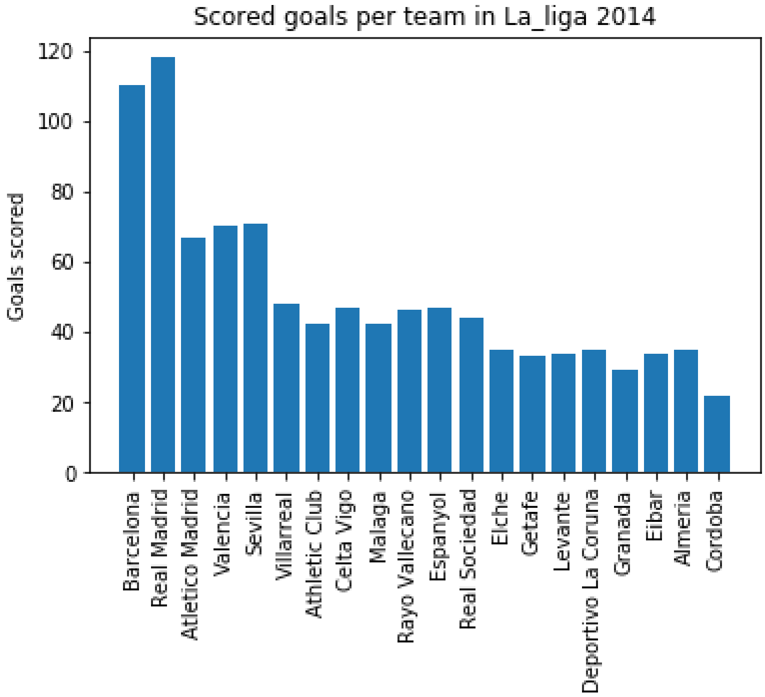
```
color_list = [df_SerieA_2014['position']]
```

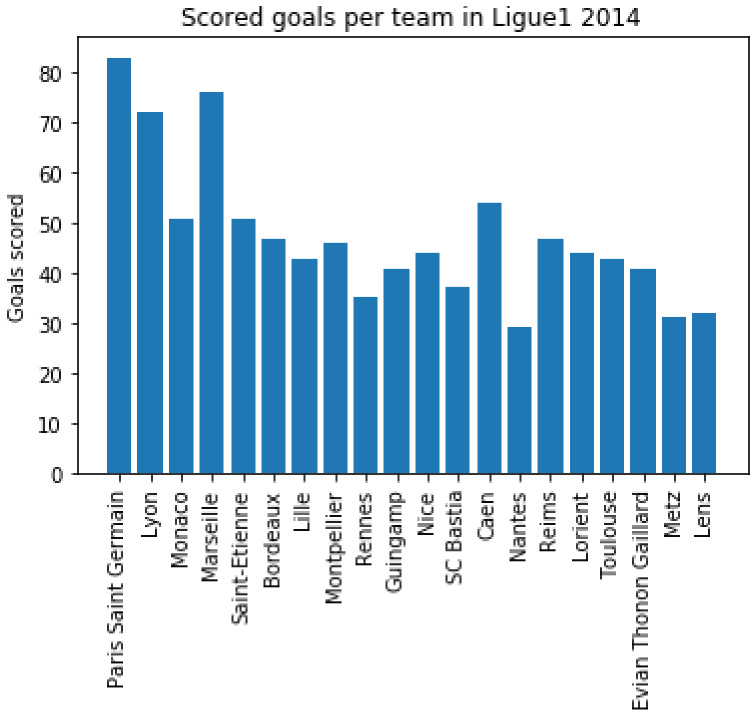
In [11]:

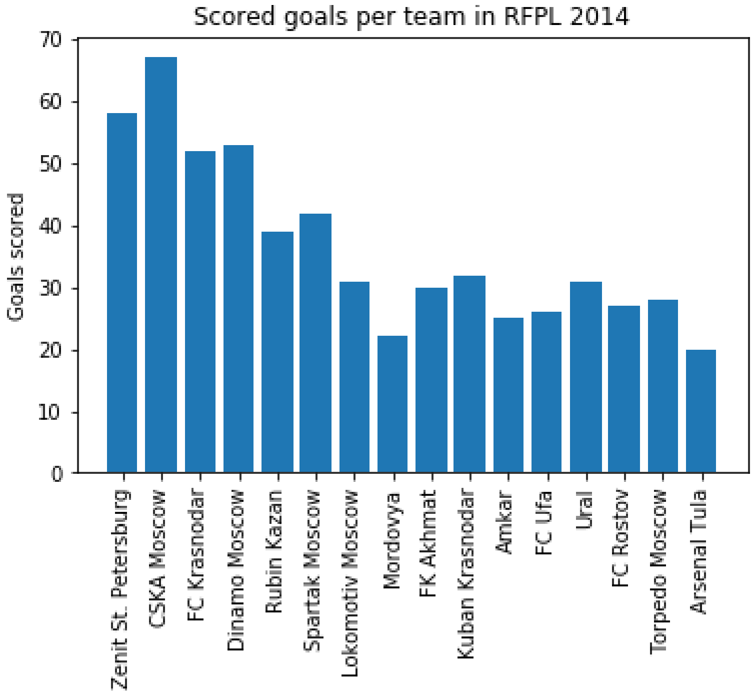
```
##Goals scored per team per league 2014
plt.bar(df_bundes_2014['team'], df_bundes_2014['scored'])
plt.ylabel("Goals scored")
plt.title("Scored goals per team in Bundesliga 2014")
plt.xticks(rotation=90)
plt.show()
plt.bar(df_epl_2014['team'], df_epl_2014['scored'])
plt.ylabel("Goals scored")
plt.title("Scored goals per team in EPL 2014")
plt.xticks(rotation=90)
plt.show()
plt.bar(df_laliga_2014['team'], df_laliga_2014['scored'])
plt.ylabel("Goals scored")
plt.title("Scored goals per team in La_liga 2014")
plt.xticks(rotation=90)
plt.show()
plt.bar(df_ligue1_2014['team'], df_ligue1_2014['scored'])
plt.ylabel("Goals scored")
plt.title("Scored goals per team in Ligue1 2014")
plt.xticks(rotation=90)
plt.show()
plt.bar(df_RFPL_2014['team'], df_RFPL_2014['scored'])
plt.ylabel("Goals scored")
plt.title("Scored goals per team in RFPL 2014")
plt.xticks(rotation=90)
plt.show()
plt.bar(df_SerieA_2014['team'], df_SerieA_2014['scored'])
plt.ylabel("Goals scored")
plt.title("Scored goals per team in Serie_A 2014")
plt.xticks(rotation=90)
plt.show()
```

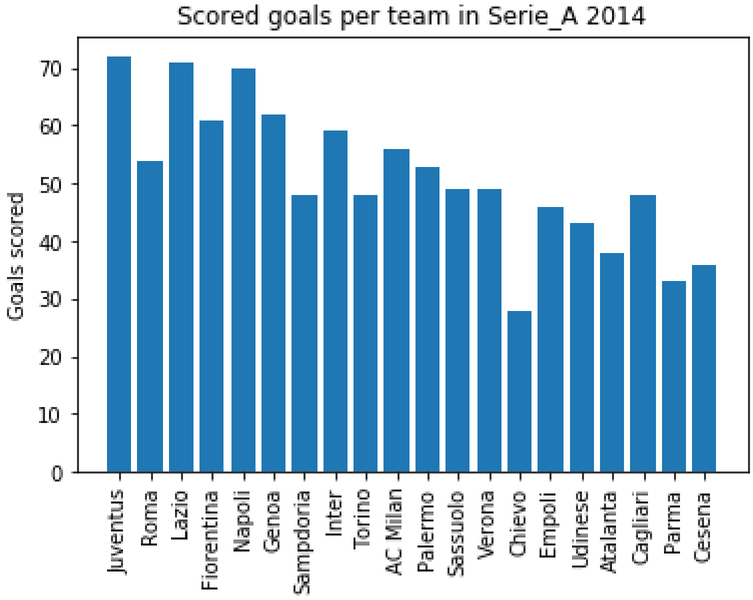

Scored goals per team in Bundesliga 2014
Goals scored
0
10
20
30
40
50
60
70
80
Bayern Munich
Wolfsburg
Borussia M.Gladbach
Bayer Leverkusen
Augsburg
Schalke 04
Borussia Dortmund
Hoffenheim
Werder Bremen
Eintracht Frankfurt
FC Cologne
Mainz 05
Hannover 96
VfB Stuttgart
Hamburger SV
Hertha Berlin
Freiburg
Paderborn


Scored goals per team in EPL 2014
Goals scored
0
10
20
30
40
50
60
70
80
Chelsea
Manchester City
Arsenal
Manchester United
Tottenham
Liverpool
Southampton
Swansea
Stoke
Crystal Palace
West Ham
Everton
West Bromwich Albion
Leicester
Newcastle United
Aston Villa
Sunderland
Hull
Burnley
Queens Park Rangers


Scored goals per team in La_liga 2014
Goals scored
0
20
40
60
80
100
120
Barcelona
Real Madrid
Atletico Madrid
Valencia
Sevilla
Villarreal
Athletic Club
Celta Vigo
Malaga
Rayo Vallecano
Espanyol
Real Sociedad
Elche
Getafe
Levante
Deportivo La Coruna
Granada
Eibar
Almeria
Cordoba

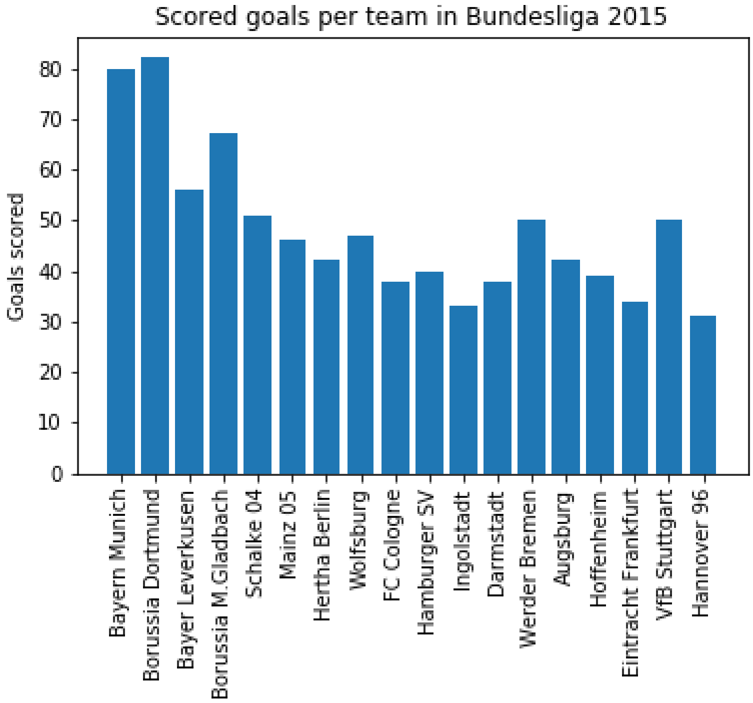

Scored goals per team in Ligue1 2014
Goals scored
0
10
20
30
40
50
60
70
80
Paris Saint Germain
Lyon
Monaco
Marseille
Saint-Etienne
Bordeaux
Lille
Montpellier
Rennes
Guingamp
Nice
SC Bastia
Caen
Nantes
Reims
Lorient
Toulouse
Evian Thonon Gaillard
Metz
Lens

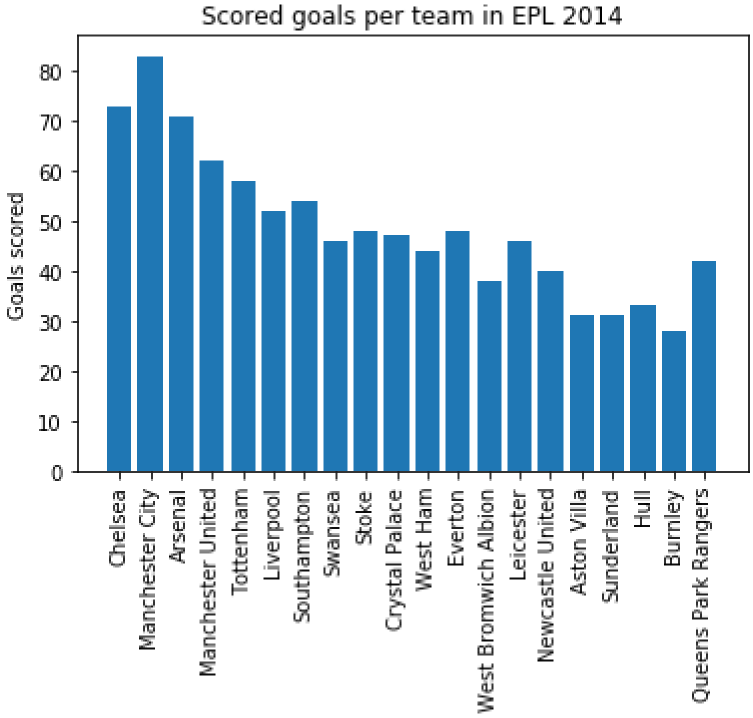

Scored goals per team in RFPL 2014
Goals scored
0
10
20
30
40
50
60
70
Zenit St. Petersburg
CSKA Moscow
FC Krasnodar
Dinamo Moscow
Rubin Kazan
Spartak Moscow
Lokomotiv Moscow
Mordovya
FK Akhmat
Kuban Krasnodar
Amkar
FC Ufa
Ural
FC Rostov
Torpedo Moscow
Arsenal Tula

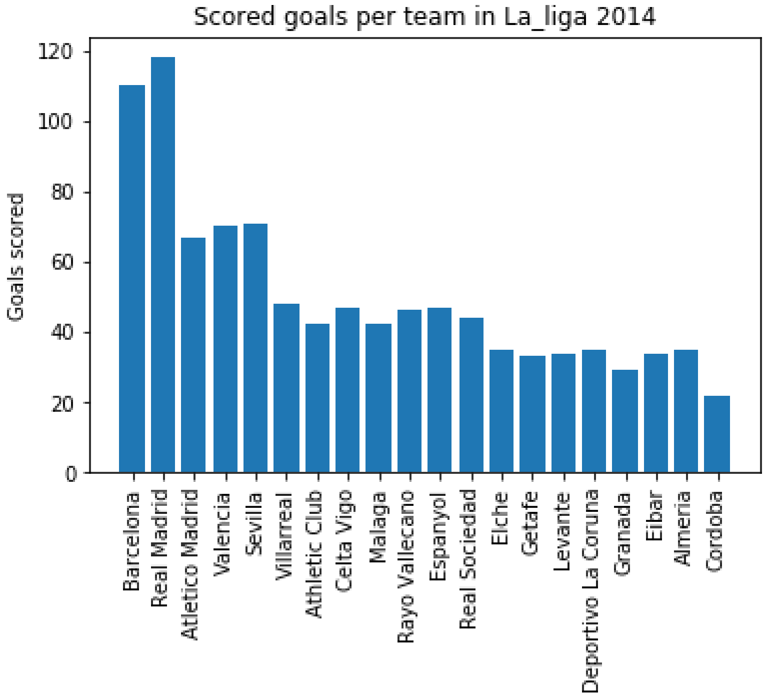

Scored goals per team in Serie_A 2014
Goals scored
0
10
20
30
40
50
60
70
Juventus
Roma
Lazio
Fiorentina
Napoli
Genoa
Sampdoria
Inter
Torino
AC Milan
Palermo
Sassuolo
Verona
Chievo
Empoli
Udinese
Atalanta
Cagliari
Parma
Cesena

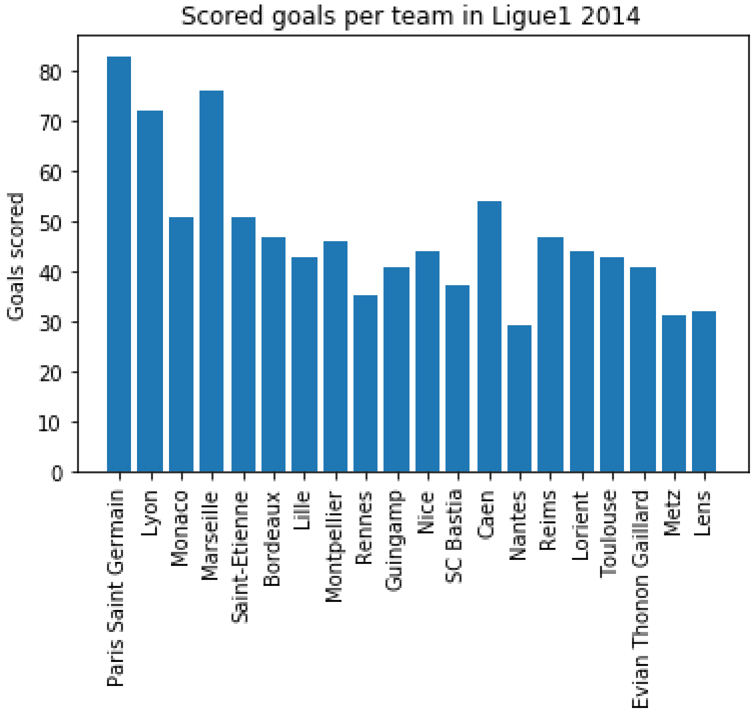
In [12]:

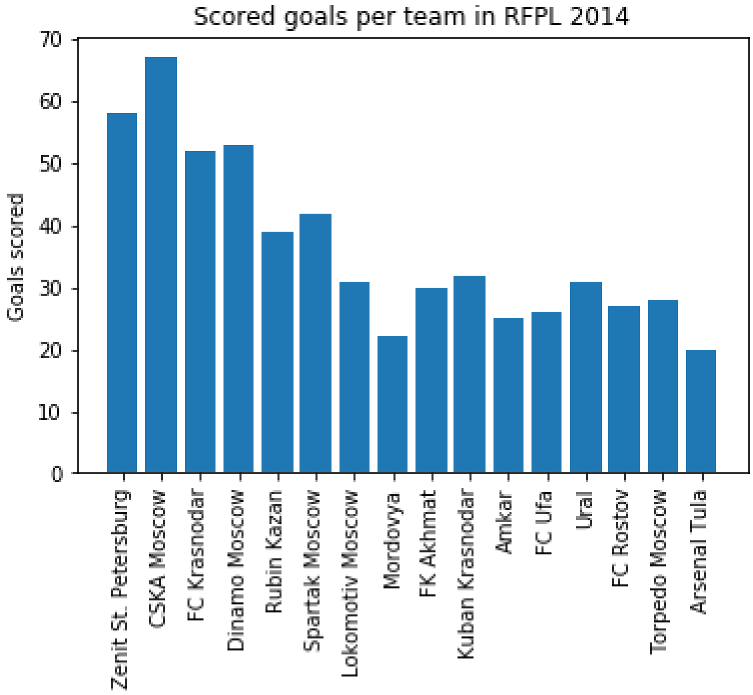
```
plt.bar(df_bundes_2015['team'], df_bundes_2015['scored'])
plt.ylabel("Goals scored")
plt.title("Scored goals per team in Bundesliga 2015")
plt.xticks(rotation=90)
plt.show()
plt.bar(df_epl_2014['team'], df_epl_2014['scored'])
plt.ylabel("Goals scored")
plt.title("Scored goals per team in EPL 2014")
plt.xticks(rotation=90)
plt.show()
plt.bar(df_laliga_2014['team'], df_laliga_2014['scored'])
plt.ylabel("Goals scored")
plt.title("Scored goals per team in La_liga 2014")
plt.xticks(rotation=90)
plt.show()
plt.bar(df_ligue1_2014['team'], df_ligue1_2014['scored'])
plt.ylabel("Goals scored")
plt.title("Scored goals per team in Ligue1 2014")
plt.xticks(rotation=90)
plt.show()
plt.bar(df_RFPL_2014['team'], df_RFPL_2014['scored'])
plt.ylabel("Goals scored")
plt.title("Scored goals per team in RFPL 2014")
plt.xticks(rotation=90)
plt.show()
plt.bar(df_SerieA_2014['team'], df_SerieA_2014['scored'])
plt.ylabel("Goals scored")
plt.title("Scored goals per team in Serie_A 2014")
plt.xticks(rotation=90)
plt.show()
```

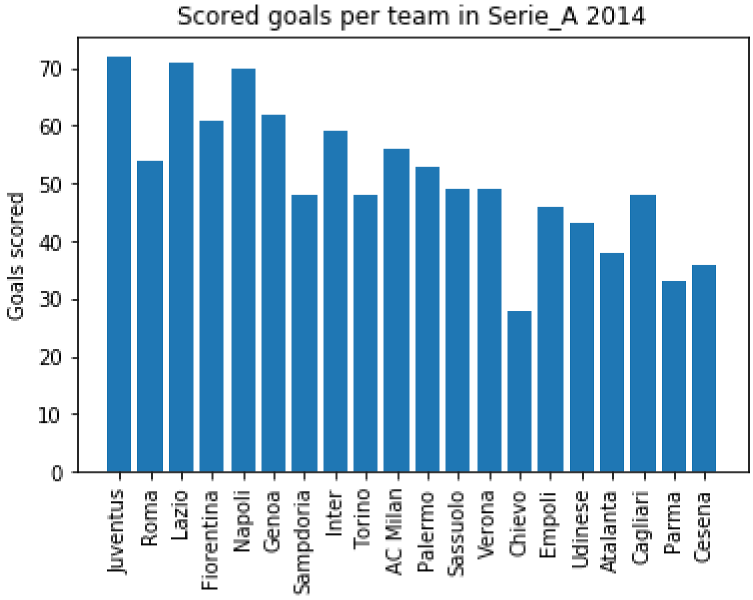

Scored goals per team in Bundesliga 2015
Goals scored
0
10
20
30
40
50
60
70
80
Bayern Munich
Borussia Dortmund
Bayer Leverkusen
Borussia M.Gladbach
Schalke 04
Mainz 05
Hertha Berlin
Wolfsburg
FC Cologne
Hamburger SV
Ingolstadt
Darmstadt
Werder Bremen
Augsburg
Hoffenheim
Eintracht Frankfurt
VfB Stuttgart
Hannover 96


Scored goals per team in EPL 2014
Goals scored
0
10
20
30
40
50
60
70
80
Chelsea
Manchester City
Arsenal
Manchester United
Tottenham
Liverpool
Southampton
Swansea
Stoke
Crystal Palace
West Ham
Everton
West Bromwich Albion
Leicester
Newcastle United
Aston Villa
Sunderland
Hull
Burnley
Queens Park Rangers


Scored goals per team in La_liga 2014
Goals scored
0
20
40
60
80
100
120
Barcelona
Real Madrid
Atletico Madrid
Valencia
Sevilla
Villarreal
Athletic Club
Celta Vigo
Malaga
Rayo Vallecano
Espanyol
Real Sociedad
Elche
Getafe
Levante
Deportivo La Coruna
Granada
Eibar
Almeria
Cordoba

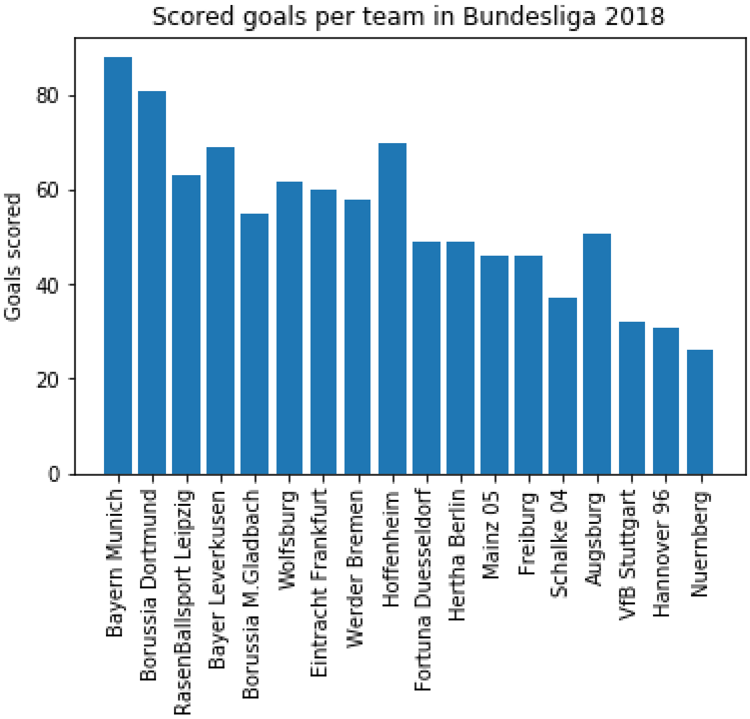

Scored goals per team in Ligue1 2014
Goals scored
0
10
20
30
40
50
60
70
80
Paris Saint Germain
Lyon
Monaco
Marseille
Saint-Etienne
Bordeaux
Lille
Montpellier
Rennes
Guingamp
Nice
SC Bastia
Caen
Nantes
Reims
Lorient
Toulouse
Evian Thonon Gaillard
Metz
Lens

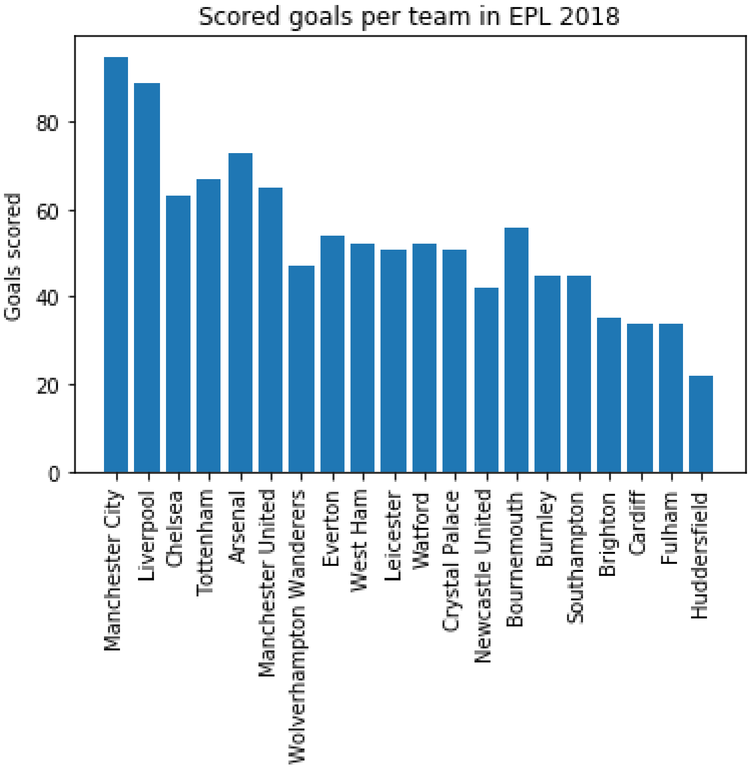

Scored goals per team in RFPL 2014
Goals scored
0
10
20
30
40
50
60
70
Zenit St. Petersburg
CSKA Moscow
FC Krasnodar
Dinamo Moscow
Rubin Kazan
Spartak Moscow
Lokomotiv Moscow
Mordovya
FK Akhmat
Kuban Krasnodar
Amkar
FC Ufa
Ural
FC Rostov
Torpedo Moscow
Arsenal Tula

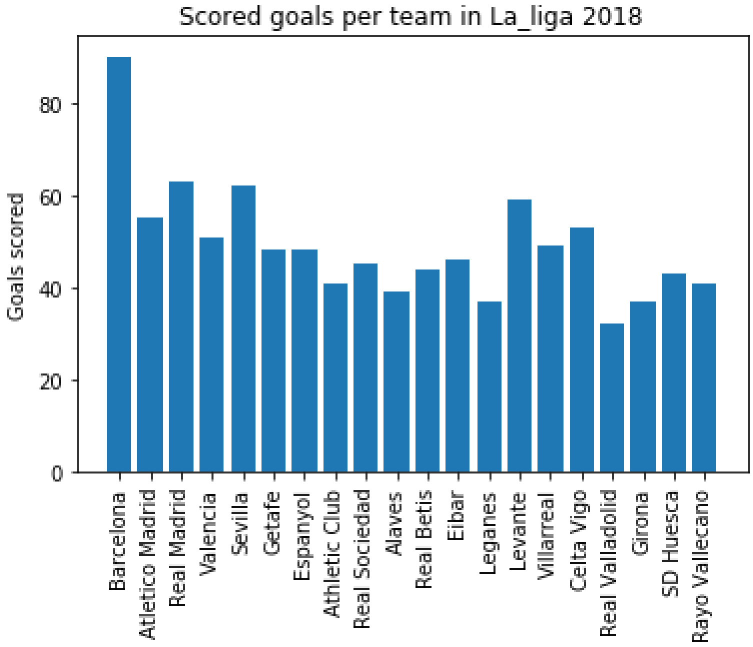

Scored goals per team in Serie_A 2014
Goals scored
0
10
20
30
40
50
60
70
Juventus
Roma
Lazio
Fiorentina
Napoli
Genoa
Sampdoria
Inter
Torino
AC Milan
Palermo
Sassuolo
Verona
Chievo
Empoli
Udinese
Atalanta
Cagliari
Parma
Cesena

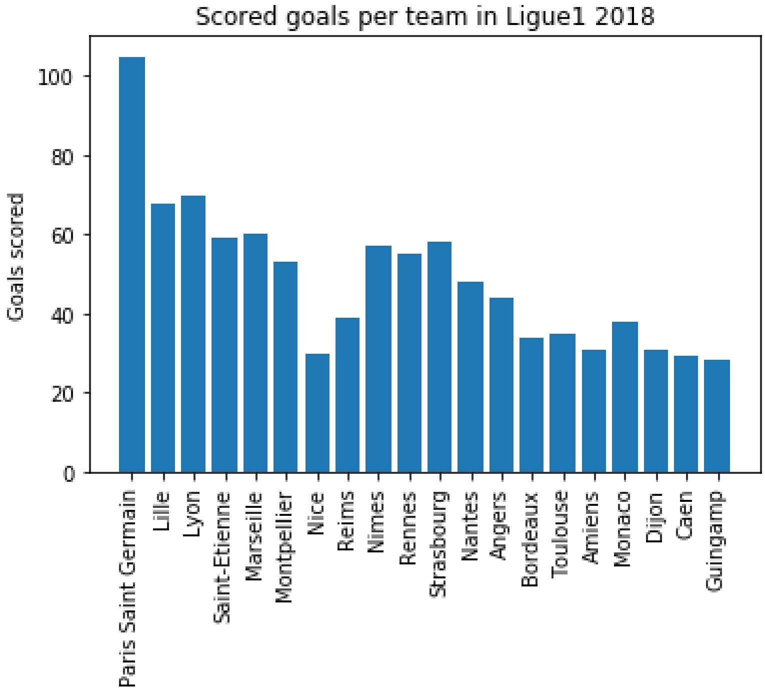
In [13]:

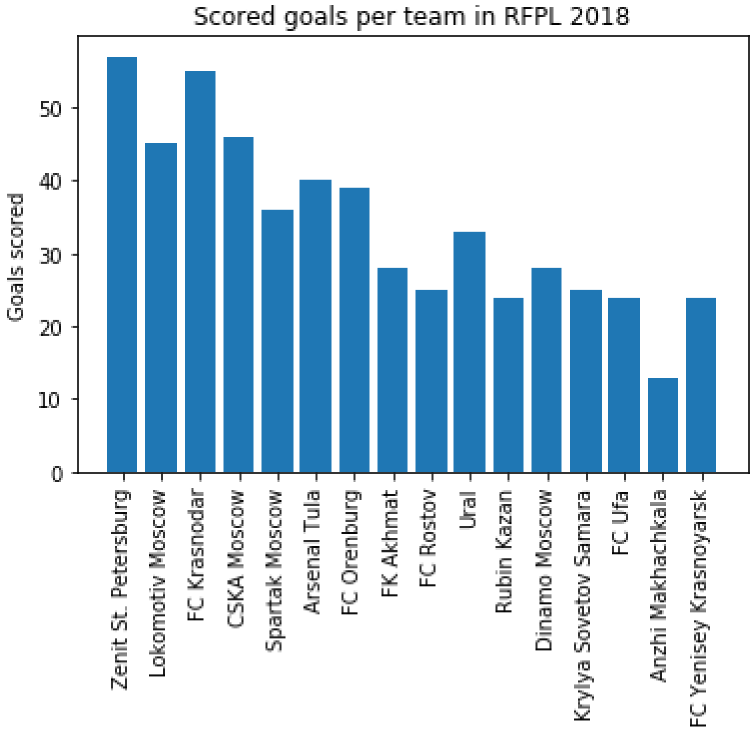
```
##Goals scored per team per league 2018
plt.bar(df_bundes_2018['team'], df_bundes_2018['scored'])
plt.ylabel("Goals scored")
plt.title("Scored goals per team in Bundesliga 2018")
plt.xticks(rotation=90)
plt.show()
plt.bar(df_epl_2018['team'], df_epl_2018['scored'])
plt.ylabel("Goals scored")
plt.title("Scored goals per team in EPL 2018")
plt.xticks(rotation=90)
plt.show()
plt.bar(df_laliga_2018['team'], df_laliga_2018['scored'])
plt.ylabel("Goals scored")
plt.title("Scored goals per team in La_liga 2018")
plt.xticks(rotation=90)
plt.show()
plt.bar(df_ligue1_2018['team'], df_ligue1_2018['scored'])
plt.ylabel("Goals scored")
plt.title("Scored goals per team in Ligue1 2018")
plt.xticks(rotation=90)
plt.show()
plt.bar(df_RFPL_2018['team'], df_RFPL_2018['scored'])
plt.ylabel("Goals scored")
plt.title("Scored goals per team in RFPL 2018")
plt.xticks(rotation=90)
plt.show()
plt.bar(df_SerieA_2018['team'], df_SerieA_2018['scored'])
plt.ylabel("Goals scored")
plt.title("Scored goals per team in Serie_A 2018")
plt.xticks(rotation=90)
plt.show()
```

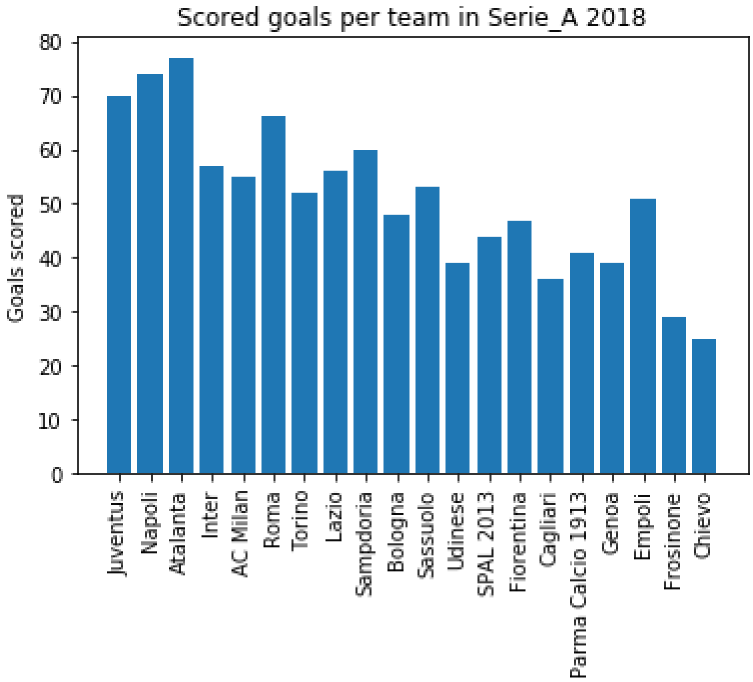

Scored goals per team in Bundesliga 2018
Goals scored
0
20
40
60
80
Bayern Munich
Borussia Dortmund
RasenBallsport Leipzig
Bayer Leverkusen
Borussia M.Gladbach
Wolfsburg
Eintracht Frankfurt
Werder Bremen
Hoffenheim
Fortuna Duesseldorf
Hertha Berlin
Mainz 05
Freiburg
Schalke 04
Augsburg
VfB Stuttgart
Hannover 96
Nuernberg


Scored goals per team in EPL 2018
Goals scored
0
20
40
60
80
Manchester City
Liverpool
Chelsea
Tottenham
Arsenal
Manchester United
Wolverhampton Wanderers
Everton
West Ham
Leicester
Watford
Crystal Palace
Newcastle United
Bournemouth
Burnley
Southampton
Brighton
Cardiff
Fulham
Huddersfield


Scored goals per team in La_liga 2018
Goals scored
0
20
40
60
80
Barcelona
Atletico Madrid
Real Madrid
Valencia
Sevilla
Getafe
Espanyol
Athletic Club
Real Sociedad
Alaves
Real Betis
Eibar
Leganes
Levante
Villarreal
Celta Vigo
Real Valladolid
Girona
SD Huesca
Rayo Vallecano


Scored goals per team in Ligue1 2018
Goals scored
0
20
40
60
80
100
Paris Saint Germain
Lille
Lyon
Saint-Etienne
Marseille
Montpellier
Nice
Reims
Nimes
Rennes
Strasbourg
Nantes
Angers
Bordeaux
Toulouse
Amiens
Monaco
Dijon
Caen
Guingamp


Scored goals per team in RFPL 2018
Goals scored
0
10
20
30
40
50
Zenit St. Petersburg
Lokomotiv Moscow
FC Krasnodar
CSKA Moscow
Spartak Moscow
Arsenal Tula
FC Orenburg
FK Akhmat
FC Rostov
Ural
Rubin Kazan
Dinamo Moscow
Krylya Sovetov Samara
FC Ufa
Anzhi Makhachkala
FC Yenisey Krasnoyarsk

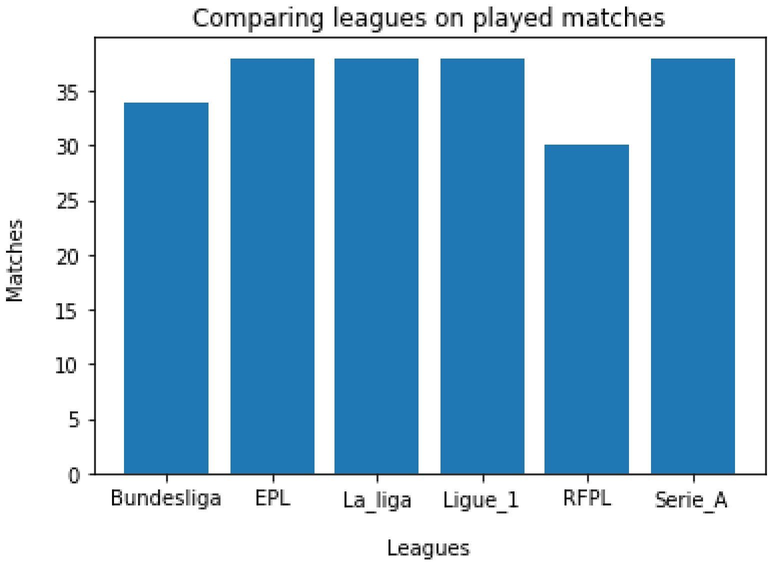

Scored goals per team in Serie_A 2018
Goals scored
0
10
20
30
40
50
60
70
80
Juventus
Napoli
Atalanta
Inter
AC Milan
Roma
Torino
Lazio
Sampdoria
Bologna
Sassuolo
Udinese
SPAL 2013
Fiorentina
Cagliari
Parma Calcio 1913
Genoa
Empoli
Frosinone
Chievo

In [14]:

```
plt.bar(df_results['League'],df_results['matches'],align='center', alpha=0.5)
plt.xlabel("Leagues", labelpad=14)
plt.ylabel("Matches", labelpad=14)
plt.title("Comparing leagues on played matches")
```

Out[14]:

```
Text(0.5, 1.0, 'Comparing leagues on played matches')
```



In [15]:

```
#Making dataframes for each league every year
df_bundes_top3_2014_rows = df_results.loc[(df_results['League'] == 'Bundesliga')&
                                     (df_results['Year']==2014)]
df_bundes_top3_2014 = df_bundes_top3_2014_rows[0:3]
df_bundes_top3_2015_rows = df_results.loc[(df_results['League'] == 'Bundesliga')&
                                     (df_results['Year']==2015)]
df_bundes_top3_2015 = df_bundes_top3_2015_rows[0:3]
df_bundes_top3_2016_rows = df_results.loc[(df_results['League'] == 'Bundesliga')&
                                     (df_results['Year']==2016)]
df_bundes_top3_2016 = df_bundes_top3_2016_rows[0:3]
df_bundes_top3_2017_rows = df_results.loc[(df_results['League'] == 'Bundesliga')&
                                     (df_results['Year']==2017)]
df_bundes_top3_2017 = df_bundes_top3_2017_rows[0:3]
df_bundes_top3_2018_rows = df_results.loc[(df_results['League'] == 'Bundesliga')&
                                     (df_results['Year']==2018)]
df_bundes_top3_2018 = df_bundes_top3_2018_rows[0:3]

df_epl_top3_2014_rows = df_results.loc[(df_results['League'] == 'EPL')&
                                     (df_results['Year']==2014)]
df_epl_top3_2014 = df_epl_top3_2014_rows[0:3]
df_epl_top3_2015_rows = df_results.loc[(df_results['League'] == 'EPL')&
                                     (df_results['Year']==2015)]
df_epl_top3_2015 = df_epl_top3_2015_rows[0:3]
df_epl_top3_2016_rows = df_results.loc[(df_results['League'] == 'EPL')&
                                     (df_results['Year']==2016)]
df_epl_top3_2016 = df_epl_top3_2016_rows[0:3]
df_epl_top3_2017_rows = df_results.loc[(df_results['League'] == 'EPL')&
                                     (df_results['Year']==2017)]
df_epl_top3_2017 = df_epl_top3_2017_rows[0:3]
df_epl_top3_2018_rows = df_results.loc[(df_results['League'] == 'EPL')&
                                     (df_results['Year']==2018)]
df_epl_top3_2018 = df_epl_top3_2018_rows[0:3]

df_laliga_top3_2014_rows = df_results.loc[(df_results['League'] == 'La_liga')&
                                     (df_results['Year']==2014)]
df_laliga_top3_2014 = df_laliga_top3_2014_rows[0:3]
df_laliga_top3_2015_rows = df_results.loc[(df_results['League'] == 'La_liga')&
                                     (df_results['Year']==2015)]
df_laliga_top3_2015 = df_laliga_top3_2015_rows[0:3]
df_laliga_top3_2016_rows = df_results.loc[(df_results['League'] == 'La_liga')&
                                     (df_results['Year']==2016)]
df_laliga_top3_2016 = df_laliga_top3_2016_rows[0:3]
df_laliga_top3_2017_rows = df_results.loc[(df_results['League'] == 'La_liga')&
                                     (df_results['Year']==2017)]
df_laliga_top3_2017 = df_laliga_top3_2017_rows[0:3]
df_laliga_top3_2018_rows = df_results.loc[(df_results['League'] == 'La_liga')&
                                     (df_results['Year']==2018)]
df_laliga_top3_2018 = df_laliga_top3_2018_rows[0:3]

df_ligue1_top3_2014_rows = df_results.loc[(df_results['League'] == 'Ligue_1')&
                                     (df_results['Year']==2014)]
df_ligue1_top3_2014 = df_ligue1_top3_2014_rows[0:3]
df_ligue1_top3_2015_rows = df_results.loc[(df_results['League'] == 'Ligue_1')&
                                     (df_results['Year']==2015)]
df_ligue1_top3_2015 = df_ligue1_top3_2015_rows[0:3]
df_ligue1_top3_2016_rows = df_results.loc[(df_results['League'] == 'Ligue_1')&
                                     (df_results['Year']==2016)]
df_ligue1_top3_2016 = df_ligue1_top3_2016_rows[0:3]
df_ligue1_top3_2017_rows = df_results.loc[(df_results['League'] == 'Ligue_1')&
```

```
                                     (df_results['Year']==2017)]
df_ligue1_top3_2017 = df_ligue1_top3_2017_rows[0:3]
df_ligue1_top3_2018_rows = df_results.loc[(df_results['League'] == 'Ligue_1')&
                                     (df_results['Year']==2018)]
df_ligue1_top3_2018 = df_ligue1_top3_2018_rows[0:3]

df_RFPL_top3_2014_rows = df_results.loc[(df_results['League'] == 'RFPL')&
                                     (df_results['Year']==2014)]
df_RFPL_top3_2014 = df_RFPL_top3_2014_rows[0:3]
df_RFPL_top3_2015_rows = df_results.loc[(df_results['League'] == 'RFPL')&
                                     (df_results['Year']==2015)]
df_RFPL_top3_2015 = df_RFPL_top3_2015_rows[0:3]
df_RFPL_top3_2016_rows = df_results.loc[(df_results['League'] == 'RFPL')&
                                     (df_results['Year']==2016)]
df_RFPL_top3_2016 = df_RFPL_top3_2016_rows[0:3]
df_RFPL_top3_2017_rows = df_results.loc[(df_results['League'] == 'RFPL')&
                                     (df_results['Year']==2017)]
df_RFPL_top3_2017 = df_RFPL_top3_2017_rows[0:3]
df_RFPL_top3_2018_rows = df_results.loc[(df_results['League'] == 'RFPL')&
                                     (df_results['Year']==2018)]
df_RFPL_top3_2018 = df_RFPL_top3_2018_rows[0:3]

df_SerieA_top3_2014_rows = df_results.loc[(df_results['League'] == 'Serie_A')&
                                     (df_results['Year']==2014)]
df_SerieA_top3_2014 = df_SerieA_top3_2014_rows[0:3]
df_SerieA_top3_2015_rows = df_results.loc[(df_results['League'] == 'Serie_A')&
                                     (df_results['Year']==2015)]
df_SerieA_top3_2015 = df_SerieA_top3_2015_rows[0:3]
df_SerieA_top3_2016_rows = df_results.loc[(df_results['League'] == 'Serie_A')&
                                     (df_results['Year']==2016)]
df_SerieA_top3_2016 = df_SerieA_top3_2016_rows[0:3]
df_SerieA_top3_2017_rows = df_results.loc[(df_results['League'] == 'Serie_A')&
                                     (df_results['Year']==2017)]
df_SerieA_top3_2017 = df_SerieA_top3_2017_rows[0:3]
df_SerieA_top3_2018_rows = df_results.loc[(df_results['League'] == 'Serie_A')&
                                     (df_results['Year']==2018)]
df_SerieA_top3_2018 = df_SerieA_top3_2018_rows[0:3]
```

In [16]:

```
#Combining top 3 teams from all years into one DataFrame
frames = [df_bundes_top3_2014, df_bundes_top3_2015, df_bundes_top3_2016, df_bundes_top3_2017, df_bundes_top3_2018,
         df_epl_top3_2014, df_epl_top3_2015, df_epl_top3_2016, df_epl_top3_2017, df_epl_top3_2018,
         df_laliga_top3_2014,df_laliga_top3_2015, df_laliga_top3_2016, df_laliga_top3_2017, df_laliga_top3_2018,
         df_ligue1_top3_2014, df_ligue1_top3_2015, df_ligue1_top3_2016, df_ligue1_top3_2017, df_ligue1_top3_2018,
         df_RFPL_top3_2014, df_RFPL_top3_2015, df_RFPL_top3_2016, df_RFPL_top3_2017, df_RFPL_top3_2018,
         df_SerieA_top3_2014, df_SerieA_top3_2015, df_SerieA_top3_2016, df_SerieA_top3_2017, df_SerieA_top3_2018]

df_leagues_top3 = pd.concat(frames)
```

In [17]:

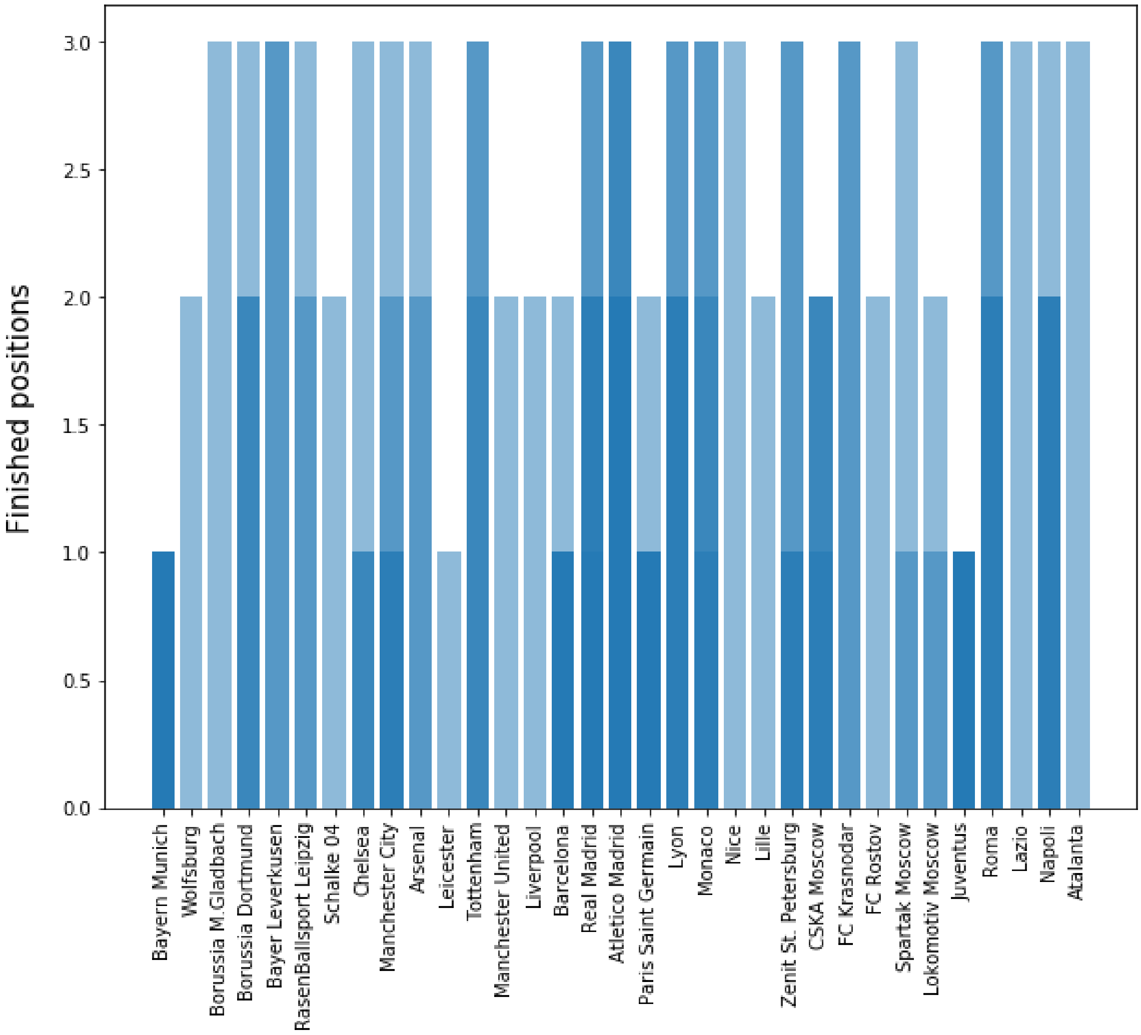
```
df_leagues_top3.head()
```

Out[17]:

	League	Year	position	team	matches	wins	draws	loses	scored	missed	p
0	Bundesliga	2014	1	Bayern Munich	34	25	4	5	80	18	7
1	Bundesliga	2014	2	Wolfsburg	34	20	9	5	72	38	6
2	Bundesliga	2014	3	Borussia M.Gladbach	34	19	9	6	53	26	6
18	Bundesliga	2015	1	Bayern Munich	34	28	4	2	80	17	8
19	Bundesliga	2015	2	Borussia Dortmund	34	24	6	4	82	34	7

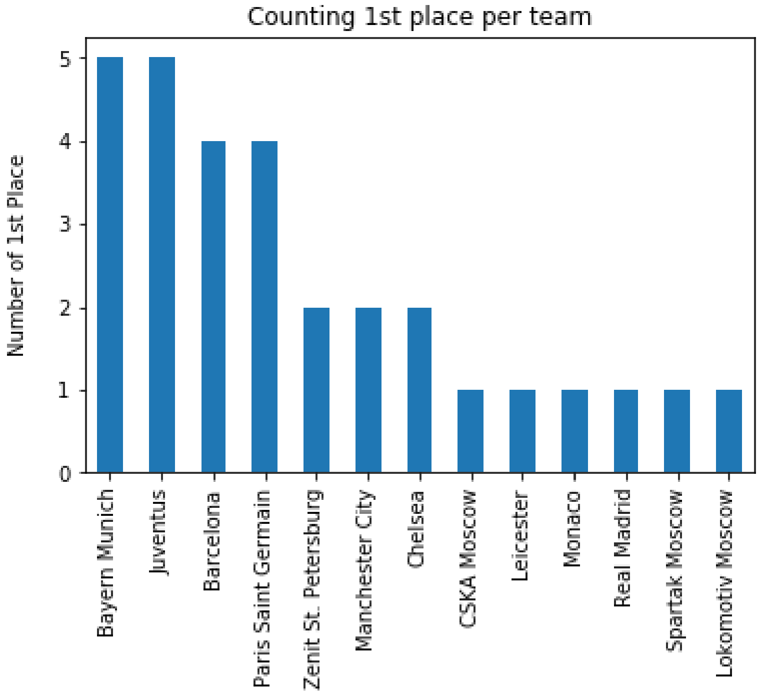
In [18]:

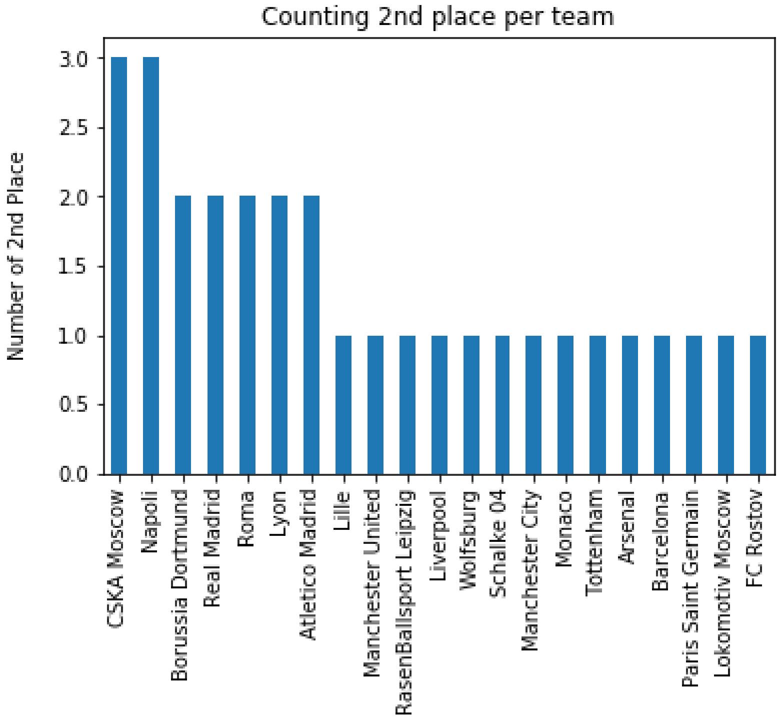
```
#Brief overview of how which teams appear on which position for all years
plt.bar(df_leagues_top3['team'], df_leagues_top3['position'], align='center', alpha=0.5
)
plt.ylabel("Finished positions", labelpad=14, fontsize = 15)
plt.xticks(rotation=90)
plt.gcf().set_size_inches(10, 8)
plt.show()
#plt.legend(df_leagues_top3['position'].value_counts()) doesn't work as wanted
```

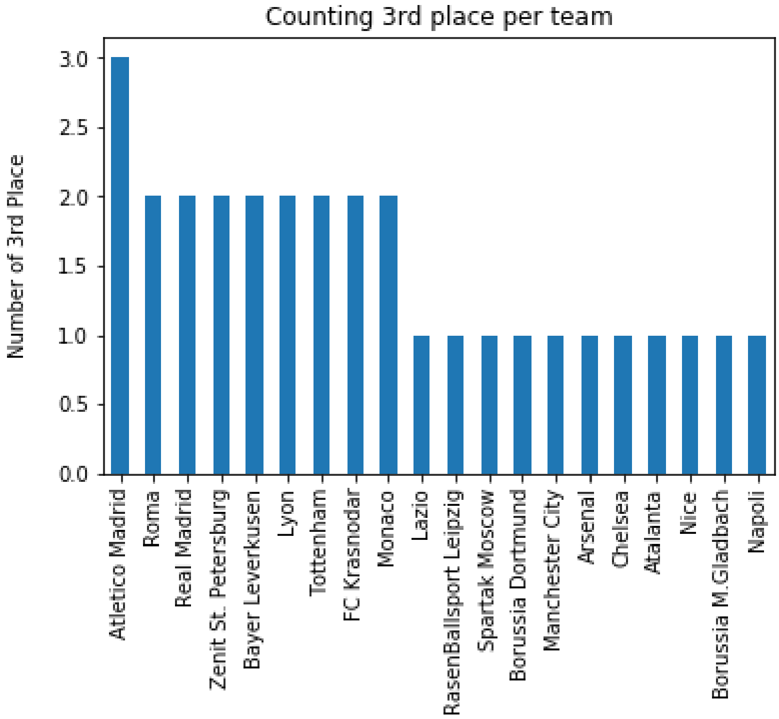

Finished positions
0.0
0.5
1.0
1.5
2.0
2.5
3.0
Bayern Munich
Wolfsburg
Borussia M.Gladbach
Borussia Dortmund
Bayer Leverkusen
RasenBallsport Leipzig
Schalke 04
Chelsea
Manchester City
Arsenal
Leicester
Tottenham
Manchester United
Liverpool
Barcelona
Real Madrid
Atletico Madrid
Paris Saint Germain
Lyon
Monaco
Nice
Lille
Zenit St. Petersburg
CSKA Moscow
FC Krasnodar
FC Rostov
Spartak Moscow
Lokomotiv Moscow
Juventus
Roma
Lazio
Napoli
Atalanta

In [19]:

```
#Counting the number of times teams have finished on first, second and third place
df_leagues_1 = df_leagues_top3.loc[(df_leagues_top3['position'] == 1)]
df_leagues_1['team'].value_counts().plot(kind = 'bar',rot=90)
plt.ylabel("Number of 1st Place", labelpad=14)
plt.title("Counting 1st place per team")
plt.show()

df_leagues_2 = df_leagues_top3.loc[(df_leagues_top3['position'] == 2)]
df_leagues_2['team'].value_counts().plot(kind = 'bar',rot=90)
plt.ylabel("Number of 2nd Place", labelpad=14)
plt.title("Counting 2nd place per team")
plt.show()

df_leagues_3 = df_leagues_top3.loc[(df_leagues_top3['position'] == 3)]
df_leagues_3['team'].value_counts().plot(kind = 'bar',rot=90)
plt.ylabel("Number of 3rd Place", labelpad=14)
plt.title("Counting 3rd place per team")
plt.show()
```


Counting 1st place per team
Number of 1st Place
0
1
2
3
4
5
Bayern Munich
Juventus
Barcelona
Paris Saint Germain
Zenit St. Petersburg
Manchester City
Chelsea
CSKA Moscow
Leicester
Monaco
Real Madrid
Spartak Moscow
Lokomotiv Moscow


Counting 2nd place per team
Number of 2nd Place
3.0
2.5
2.0
1.5
1.0
0.5
0.0
CSKA Moscow
Napoli
Borussia Dortmund
Real Madrid
Roma
Lyon
Atletico Madrid
Lille
Manchester United
RasenBallsport Leipzig
Liverpool
Wolfsburg
Schalke 04
Manchester City
Monaco
Tottenham
Arsenal
Barcelona
Paris Saint Germain
Lokomotiv Moscow
FC Rostov

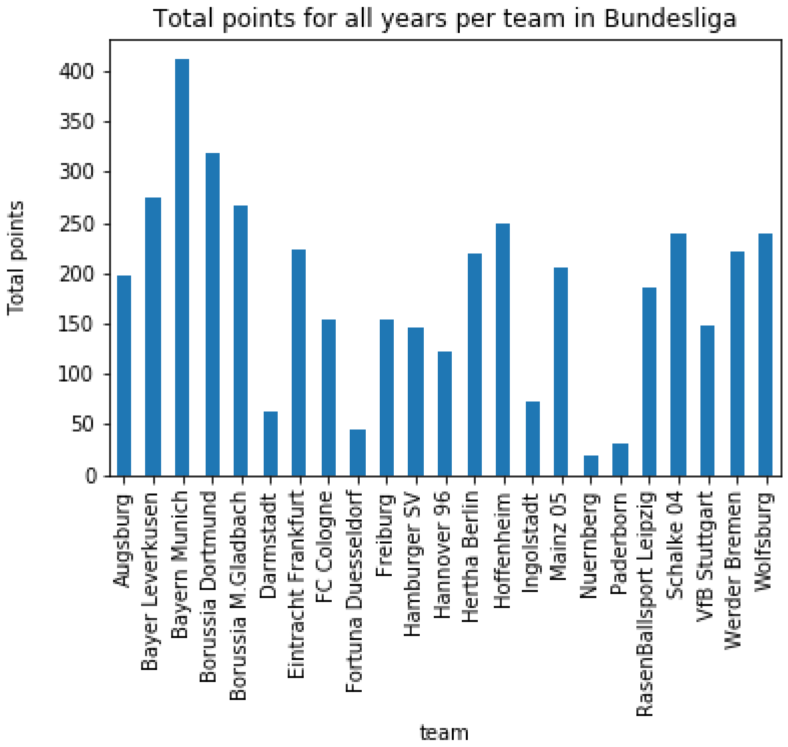


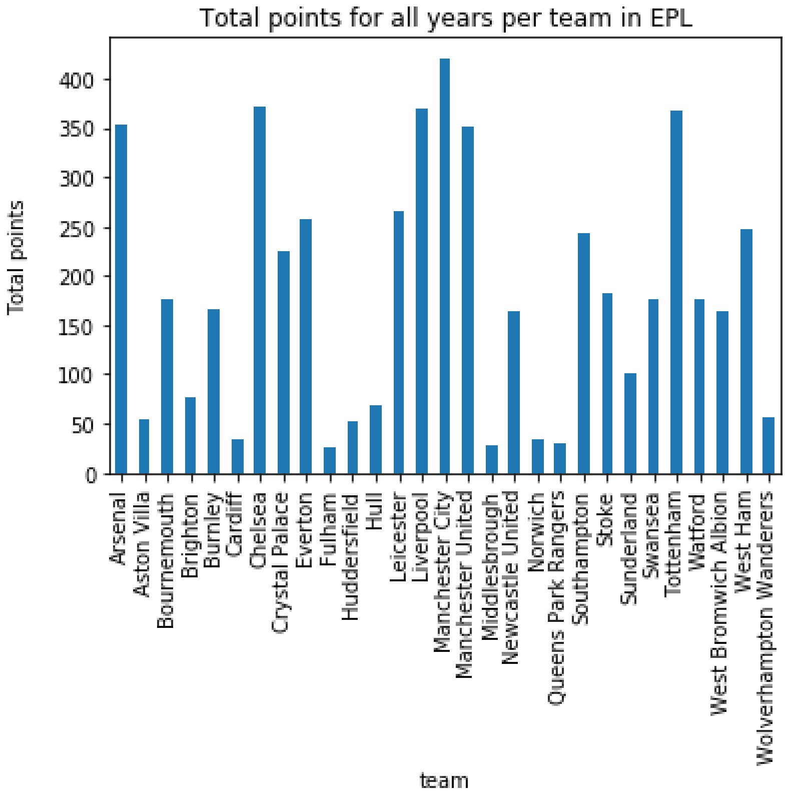
In [20]:

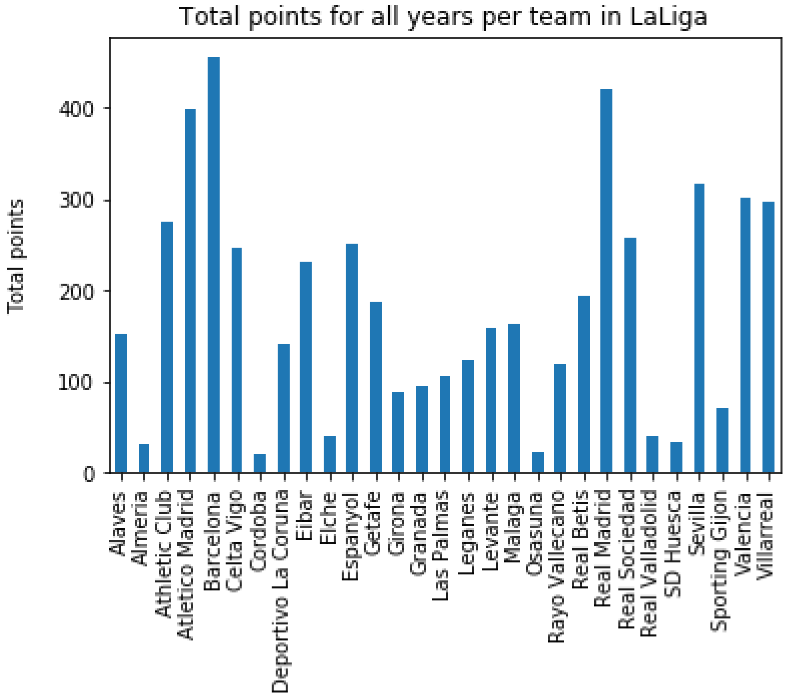
```
#Making datasets for all top3 teams per league
df_bundes_top3_alltime = df_leagues_top3.loc[(df_leagues_top3['League']=='Bundesliga')]
df_epl_top3_alltime = df_leagues_top3.loc[(df_leagues_top3['League']=='EPL')]
df_laliga_top3_alltime = df_leagues_top3.loc[(df_leagues_top3['League']=='La_liga')]
df_ligue1_top3_alltime = df_leagues_top3.loc[(df_leagues_top3['League']=='Ligue_1')]
df_RFPL_top3_alltime = df_leagues_top3.loc[(df_leagues_top3['League']=='RFPL')]
df_SerieA_top3_alltime = df_leagues_top3.loc[(df_leagues_top3['League']=='Serie_A')]

#Making datasets for all teams per league
df_bundes_alltime = df_results.loc[(df_results['League']=='Bundesliga')]
df_epl_alltime = df_results.loc[(df_results['League']=='EPL')]
df_laliga_alltime = df_results.loc[(df_results['League']=='La_liga')]
df_ligue1_alltime = df_results.loc[(df_results['League']=='Ligue_1')]
df_RFPL_alltime = df_results.loc[(df_results['League']=='RFPL')]
df_SerieA_alltime = df_results.loc[(df_results['League']=='Serie_A')]
```

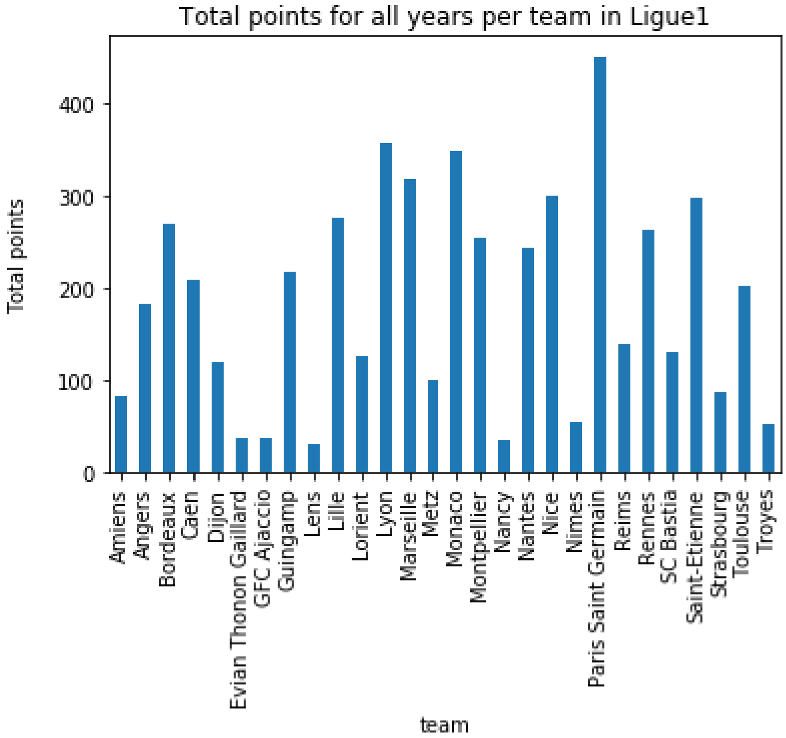
In [21]:

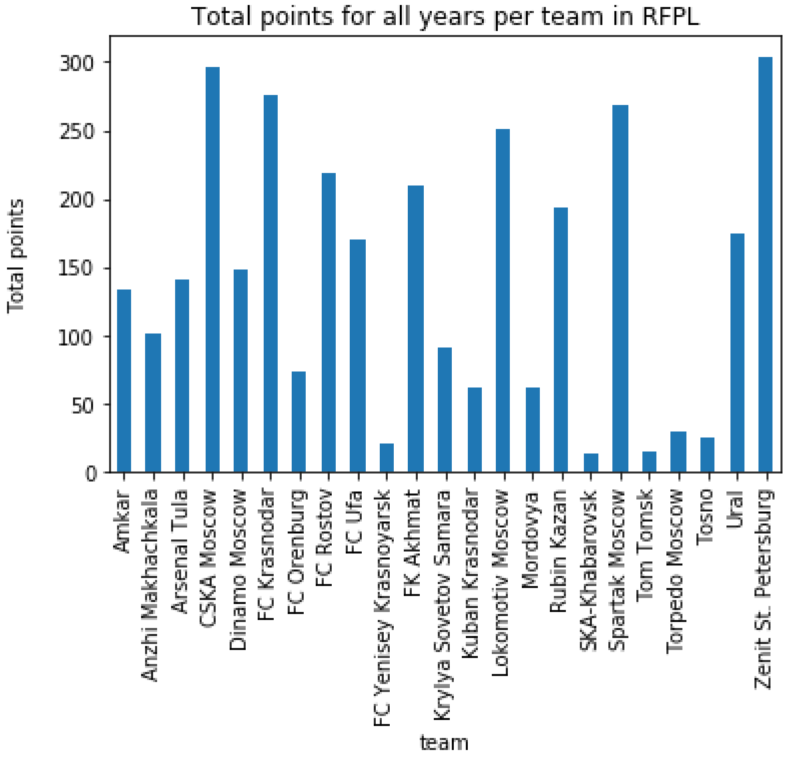
```
#Counting the amount of goals each team has scored through all seasons
df_bundes_alltime['pts'].groupby([df_bundes_alltime['team']]).sum().plot(kind = 'bar',rot=90)
plt.ylabel("Total points", labelpad=14)
plt.title("Total points for all years per team in Bundesliga")
plt.show()

df_epl_alltime['pts'].groupby([df_epl_alltime['team']]).sum().plot(kind = 'bar',rot=90)
plt.ylabel("Total points", labelpad=14)
plt.title("Total points for all years per team in EPL")
plt.show()

df_laliga_alltime['pts'].groupby([df_laliga_alltime['team']]).sum().plot(kind = 'bar',rot=90)
plt.ylabel("Total points", labelpad=14)
plt.title("Total points for all years per team in LaLiga")
plt.show()

df_ligue1_alltime['pts'].groupby([df_ligue1_alltime['team']]).sum().plot(kind = 'bar',rot=90)
plt.ylabel("Total points", labelpad=14)
plt.title("Total points for all years per team in Ligue1")
plt.show()

df_RFPL_alltime['pts'].groupby([df_RFPL_alltime['team']]).sum().plot(kind = 'bar',rot=90)
plt.ylabel("Total points", labelpad=14)
plt.title("Total points for all years per team in RFPL")
plt.show()

df_SerieA_alltime['pts'].groupby([df_SerieA_alltime['team']]).sum().plot(kind = 'bar',rot=90)
plt.ylabel("Total points", labelpad=14)
plt.title("Total points for all years per team in Serie A")
plt.show()
```

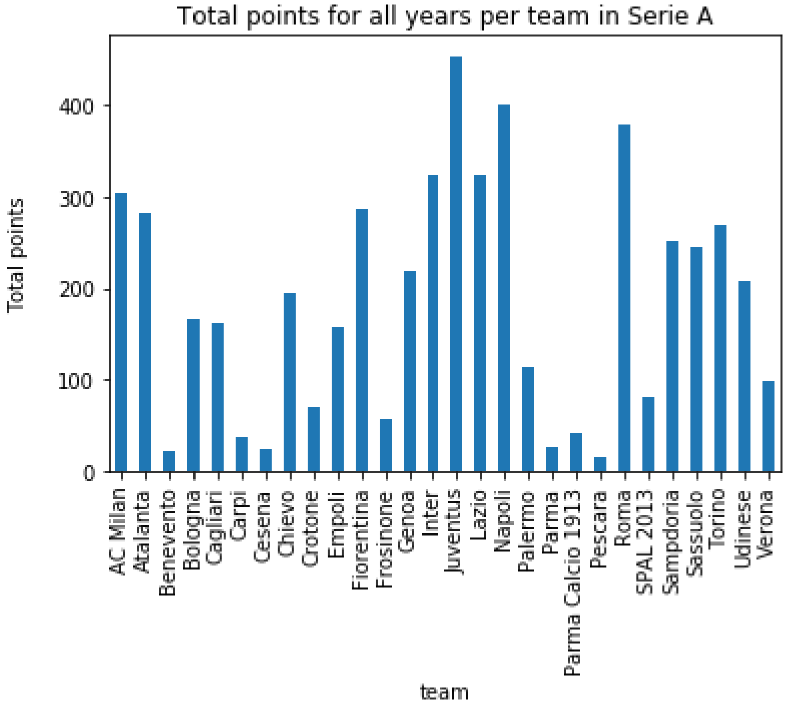

Total points for all years per team in Bundesliga
Total points
0
50
100
150
200
250
300
350
400
Augsburg
Bayer Leverkusen
Bayern Munich
Borussia Dortmund
Borussia M.Gladbach
Darmstadt
Eintracht Frankfurt
FC Cologne
Fortuna Duesseldorf
Freiburg
Hamburger SV
Hannover 96
Hertha Berlin
Hoffenheim
Ingolstadt
Mainz 05
Nuernberg
Paderborn
RasenBallsport Leipzig
Schalke 04
VfB Stuttgart
Werder Bremen
Wolfsburg
team


Total points for all years per team in EPL
Total points
0
50
100
150
200
250
300
350
400
Arsenal
Aston Villa
Bournemouth
Brighton
Burnley
Cardiff
Chelsea
Crystal Palace
Everton
Fulham
Huddersfield
Hull
Leicester
Liverpool
Manchester City
Manchester United
Middlesbrough
Newcastle United
Norwich
Queens Park Rangers
Southampton
Stoke
Sunderland
Swansea
Tottenham
Watford
West Bromwich Albion
West Ham
Wolverhampton Wanderers
team


Total points for all years per team in LaLiga
Total points
0
100
200
300
400
team
Alaves
Almeria
Athletic Club
Atletico Madrid
Barcelona
Celta Vigo
Cordoba
Deportivo La Coruna
Eibar
Elche
Espanyol
Getafe
Girona
Granada
Las Palmas
Leganes
Levante
Malaga
Osasuna
Rayo Vallecano
Real Betis
Real Madrid
Real Sociedad
Real Valladolid
SD Huesca
Sevilla
Sporting Gijon
Valencia
Villarreal


Total points for all years per team in Ligue1
Total points
0
100
200
300
400
team
Amiens
Angers
Bordeaux
Caen
Dijon
Evian Thonon Gaillard
GFC Ajaccio
Guingamp
Lens
Lille
Lorient
Lyon
Marseille
Metz
Monaco
Montpellier
Nancy
Nantes
Nice
Nimes
Paris Saint Germain
Reims
Rennes
SC Bastia
Saint-Etienne
Strasbourg
Toulouse
Troyes


Total points for all years per team in RFPL
Total points
300
250
200
150
100
50
0
Amkar
Anzhi Makhachkala
Arsenal Tula
CSKA Moscow
Dinamo Moscow
FC Krasnodar
FC Orenburg
FC Rostov
FC Ufa
FC Yenisey Krasnoyarsk
FK Akhmat
Krylya Sovetov Samara
Kuban Krasnodar
Lokomotiv Moscow
Mordovya
Rubin Kazan
SKA-Khabarovsk
Spartak Moscow
Tom Tomsk
Torpedo Moscow
Tosno
Ural
Zenit St. Petersburg
team


Total points for all years per team in Serie A
Total points
400
300
200
100
0
AC Milan
Atalanta
Benevento
Bologna
Cagliari
Carpi
Cesena
Chievo
Crotone
Empoli
Fiorentina
Frosinone
Genoa
Inter
Juventus
Lazio
Napoli
Palermo
Parma
Parma Calcio 1913
Pescara
Roma
SPAL 2013
Sampdoria
Sassuolo
Torino
Udinese
Verona
team

In [22]:

```
df_bundes_top3_alltime.head(20)
```

Out[22]:

	League	Year	position	team	matches	wins	draws	loses	scored	missed
0	Bundesliga	2014	1	Bayern Munich	34	25	4	5	80	18
1	Bundesliga	2014	2	Wolfsburg	34	20	9	5	72	38
2	Bundesliga	2014	3	Borussia M.Gladbach	34	19	9	6	53	26
18	Bundesliga	2015	1	Bayern Munich	34	28	4	2	80	17
19	Bundesliga	2015	2	Borussia Dortmund	34	24	6	4	82	34
20	Bundesliga	2015	3	Bayer Leverkusen	34	18	6	10	56	40
36	Bundesliga	2016	1	Bayern Munich	34	25	7	2	89	22
37	Bundesliga	2016	2	RasenBallsport Leipzig	34	20	7	7	66	39
38	Bundesliga	2016	3	Borussia Dortmund	34	18	10	6	72	40
54	Bundesliga	2017	1	Bayern Munich	34	27	3	4	92	28
55	Bundesliga	2017	2	Schalke 04	34	18	9	7	53	37
56	Bundesliga	2017	3	Bayer Leverkusen	34	15	10	9	58	44
72	Bundesliga	2018	1	Bayern Munich	34	24	6	4	88	32
73	Bundesliga	2018	2	Borussia Dortmund	34	23	7	4	81	44
74	Bundesliga	2018	3	RasenBallsport Leipzig	34	19	9	6	63	29

In [23]:

```
# features = tuple(df_bundes_top3_alltime[['position','TeamFill']].values)
# df_bundes_top3_alltime.boxplot(column=features, by='TeamFill', figsize=(15,8));
```

In [24]:

```
features_goals = df_results[['Year','position','matches', 'wins','draws','loses','misse
d','pts']]
features_position = df_results[['Year','matches', 'wins','draws','loses','scored','miss
ed','pts']]
```

In [25]:

```
#SelectKBest for goals
from sklearn.feature_selection import SelectKBest, chi2, f_regression, f_classif
column_goals = SelectKBest(score_func=f_classif,k=5).fit_transform(features_goals,df_re
sults['scored'])
print(column_goals)
```

```
[[ 1 34 25  5 79]
 [ 2 34 20  5 69]
 [ 3 34 19  6 66]
 ...
 [18 38 10 20 38]
 [19 38  5 23 25]
 [20 38  2 22 20]]
```

In [26]:

```
#SelectKBest for position
from sklearn.feature_selection import SelectKBest, chi2, f_regression, f_classif
column_position = SelectKBest(score_func=f_classif,k=5).fit_transform(features_position
,df_results['position'])
print(column_position)
```

```
[[25  5 80 18 79]
 [20  5 72 38 69]
 [19  6 53 26 66]
 ...
 [10 20 51 70 38]
 [ 5 23 29 69 25]
 [ 2 22 25 75 20]]
```

In [27]:

```
#Defining the columns(features) to use for training the algorithm and which column
#I want to predict(X is for features and Y is for the predicted column)
X_goals = df_results[['wins','loses','pts','missed','draws']]
y_goals = df_results['scored']
X_position = df_results[['wins','loses','scored','missed','pts']]
y_position = df_results['position']
```

In [28]:

```
#Splitting the data for predicting goals into test and train sets
X_train_goals, X_test_goals, y_train_goals, y_test_goals = train_test_split(X_goals, y_
goals, test_size=0.20, random_state=5)
#Splitting the data for predicting position into test and train sets
X_train_position, X_test_position, y_train_position, y_test_position = train_test_split
(X_position, y_position, test_size=0.20, random_state=5)
```

In [29]:

```
k_range = range(1, 31)
weight_options =('uniform', 'distance')
```

In [30]:

```
param_grid = dict(n_neighbors = k_range, weights = weight_options)
print(param_grid)
```

```
{'n_neighbors': range(1, 31), 'weights': ('uniform', 'distance')}
```

In [31]:

```
knn = KNeighborsClassifier()
grid = GridSearchCV(knn,param_grid, cv=10, scoring='r2')
grid.fit(X_goals,y_goals)
```

```
C:\Users\plame\Anaconda3\lib\site-packages\sklearn\model_selection\_split.
py:657: Warning: The least populated class in y has only 1 members, which
is too few. The minimum number of members in any class cannot be less than
n_splits=10.
  % (min_groups, self.n_splits)), Warning)
C:\Users\plame\Anaconda3\lib\site-packages\sklearn\model_selection\_searc
h.py:813: DeprecationWarning: The default of the `iid` parameter will chan
ge from True to False in version 0.22 and will be removed in 0.24. This wi
ll change numeric results when test-set sizes are unequal.
  DeprecationWarning)
```

Out[31]:

```
GridSearchCV(cv=10, error_score='raise-deprecating',
             estimator=KNeighborsClassifier(algorithm='auto', leaf_size=3
0,
                                            metric='minkowski',
                                            metric_params=None, n_jobs=Non
e,
                                            n_neighbors=5, p=2,
                                            weights='uniform'),
             iid='warn', n_jobs=None,
             param_grid={'n_neighbors': range(1, 31),
                         'weights': ('uniform', 'distance')},
             pre_dispatch='2*n_jobs', refit=True, return_train_score=Fals
e,
             scoring='r2', verbose=0)
```

In [32]:

```
print(grid.best_score_)
print(grid.best_params_)
```

```
0.6938860381587232
{'n_neighbors': 30, 'weights': 'distance'}
```

In [33]:

```
#KNN algorithm for predicting goals
knn = KNeighborsClassifier(n_neighbors=30, weights = 'distance')
knn.fit(X_train_goals, y_train_goals)
pred = knn.predict(X_test_goals)
accuracy = r2_score(y_test_goals, pred)
print(accuracy)
print(pred)
```

```
0.7779518049647038
[ 37  48  41  56  77  44  38  62  51  40  39  52  31  48  83  63  37  35
  51  45  44  29  25  51  36  39  45  41  41  37  71  46  44  83  42  25
  31  38  25  31  31  31  57  37  71  62  44  71  47  31  57  56  37  41
  39  34 112  51  41  62  45  47  38 110  35  36  70  31  42  83  42  37
  83  39  40  47  59  44  35  30  57  38  48  74  47  58  62  83  40  37
  50  56  51  40  51  33  59  42  40  28  57  45 110  38  55  48  54  45
  28  25  62  40  53  36]
```

In [34]:

```
scores_goals = cross_val_score(knn, X_goals, y_goals, scoring='r2', cv=5)
scores_goals.mean()
```

```
C:\Users\plame\Anaconda3\lib\site-packages\sklearn\model_selection\_split.
py:657: Warning: The least populated class in y has only 1 members, which
is too few. The minimum number of members in any class cannot be less than
n_splits=5.
  % (min_groups, self.n_splits)), Warning)
```

Out[34]:

```
0.7268087749092776
```

In [35]:

```
# param = {'kernel':('linear','poly','rbf','sigmoid'),
#          'C':[1,52,10],
#          'degree':[3,8],
#          'coef0':[0.001,10,0.5],
#          'gamma':('auto','scale')}
```

In [36]:

```
# SVM_ = svm.SVC()
# grid_svm = GridSearchCV(SVM_,param, cv=5)
# grid_svm.fit(X_position,y_position)
```

In [37]:

```
# print(grid_svm.best_score_)
# print(grid_svm.best_params_)
```

In [38]:

```
#SVM algorithm for predicting position
clf_position=svm.SVC(kernel='linear',C=5).fit(X_train_position,y_train_position)
predict = clf_position.predict(X_test_position)
score_position=r2_score(y_test_position,predict)
print(score_position)
print(predict)
```

```
0.9416802095800547
[11 14 11 11  1  9 16  7  1 14 14  8 20  6  2  5 15 20  8 15 10 16 13  7
  9  8  7 11  6 12  3  4 11  1 18 13 18 19 14 19 17 20  4 11  5  4 11  3
 11  7  5  5 18  2  9  6  1  2 18  5 18 12 12  1 15  9  2 20  7  2  8 15
  2  5 18 10  8 11 20 20  5 13 14  2 10  5  6  2 16 15 10  8  2 12  9 19
  5  5 13 19  6  9  1 13 10 15  4 15 15 12  5 18  4  9]
```

In [39]:

```
scores_position = cross_val_score(clf_position, X_position, y_position, scoring='r2', c
v=5)
scores_position.mean()
```

Out[39]:

```
0.9362969305659015
```

In [40]:

```
clf_gb = GradientBoostingRegressor(n_estimators=500,min_samples_split=90)
clf_gb.fit(X_train_position,y_train_position)

y_pred_gb = clf_gb.predict(X_test_position)
scores = cross_val_score(clf_gb, X_position, y_position, cv=5)
print("Accuracy: %0.2f (+/- %0.2f)" % (scores.mean(), scores.std() * 2))
```

```
Accuracy: 0.92 (+/- 0.04)
```

In [41]:

```
from sklearn.ensemble import RandomForestClassifier
clf1 = RandomForestClassifier(n_estimators = 250, max_depth=50, min_samples_split= 20,
min_samples_leaf= 13)
clf1.fit(X_train_position, y_train_position)
pred1 = clf1.predict(X_test_position)
acc1 = accuracy_score(pred1,y_test_position)
print(acc1)
```

```
0.20175438596491227
```

In [42]:

```
clf_gb_goals = GradientBoostingRegressor(n_estimators=180,min_samples_split=50)
clf_gb_goals.fit(X_train_goals,y_train_goals)

y_pred_gb_goals = clf_gb_goals.predict(X_test_goals)
scores_gb_goals = cross_val_score(clf_gb_goals, X_goals, y_goals, cv=2)
print("Accuracy: %0.2f (+/- %0.2f)" % (scores_gb_goals.mean(), scores_gb_goals.std() *
2))
```

```
Accuracy: 0.83 (+/- 0.02)
```

In [43]:

```
from sklearn.linear_model import LogisticRegression
model = LogisticRegression(multi_class='auto', C=4, random_state= 250, solver='lbfgs',
max_iter=1000)
clf_Reg = model.fit(X_train_goals, y_train_goals)
predicted_classes = model.predict(X_test_goals)
accuracy = r2_score(y_test_goals,predicted_classes)
accuracy
```

```
C:\Users\plame\Anaconda3\lib\site-packages\sklearn\linear_model\logistic.p
y:947: ConvergenceWarning: lbfgs failed to converge. Increase the number o
f iterations.
  "of iterations.", ConvergenceWarning)
```

Out[43]:

```
0.6144812599590312
```

In [44]:

```
#Testing SVM for predicting position on smaller DataFrame(Top 3 teams from Bundesliga f
or all time)
X_position2 = df_bundes_top3_alltime[['wins','loses','scored','missed','pts']]
y_position2 = df_bundes_top3_alltime['position']
clf_position2=svm.SVC(kernel='linear',C=5).fit(X_train_position,y_train_position)
X_train_position2, X_test_position2, y_train_position2, y_test_position2 = train_test_s
plit(X_position2, y_position2, test_size=0.20)
predict2 = clf_position2.predict(X_test_position2)
scores_svc2_position = cross_val_score(clf_position2, X_position, y_position, cv=4, sco
ring='accuracy')
print("Accuracy: %0.2f (+/- %0.2f)" % (scores_svc2_position.mean(), scores_svc2_positio
n.std() * 2))
print(predict2)
```

```
Accuracy: 0.32 (+/- 0.08)
[1 1 1]
```

In [45]:

```
#Testing KNN for predicting goals on smaller DataFrame(Top 3 teams from Bundesliga for all time)
X_goals2 = df_bundes_top3_alltime[['wins','loses','pts','missed','draws']]
y_goals2 = df_bundes_top3_alltime['scored']
X_train_goals2, X_test_goals2, y_train_goals2, y_test_goals2 = train_test_split(X_goals2, y_goals2, test_size=0.20)
knn2 = KNeighborsClassifier(n_neighbors=2)
knn2.fit(X_train_goals2, y_train_goals2)
pred2 = knn2.predict(X_test_goals2)
scores_goals2 = cross_val_score(knn2, X_goals2, y_goals2, cv=2,scoring='accuracy')
print(scores_goals2.mean())
print(pred2)
```

```
0.20833333333333331
[53 88 80]

C:\Users\plame\Anaconda3\lib\site-packages\sklearn\model_selection\_split.
py:657: Warning: The least populated class in y has only 1 members, which
is too few. The minimum number of members in any class cannot be less than
n_splits=2.
  % (min_groups, self.n_splits)), Warning)
```