In [1]:

```python
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]:

```python
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]:

```python
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]:

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

Out[5]:

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

In [6]:

```python
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]:

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

In [8]:

```python
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]:

```python
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)]
```
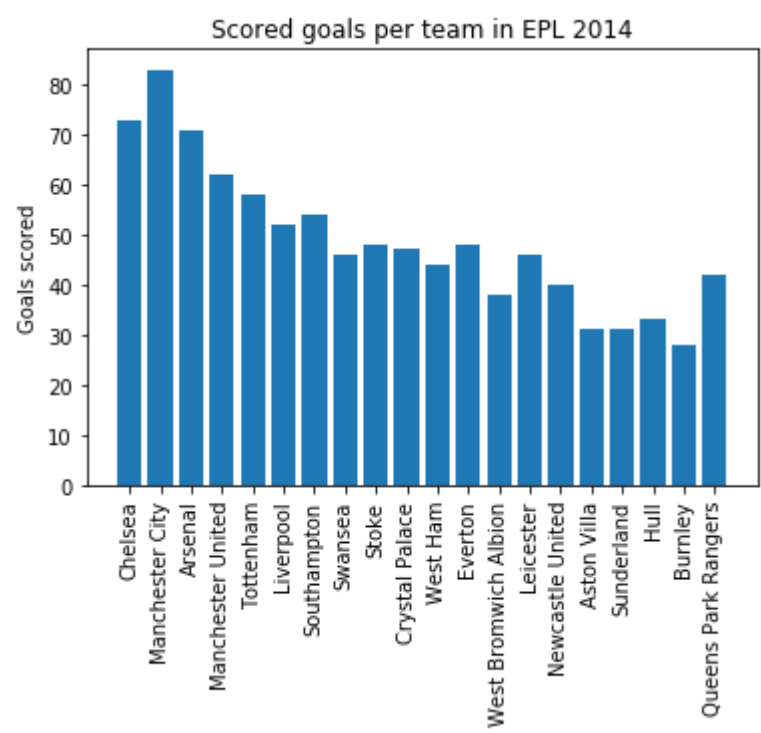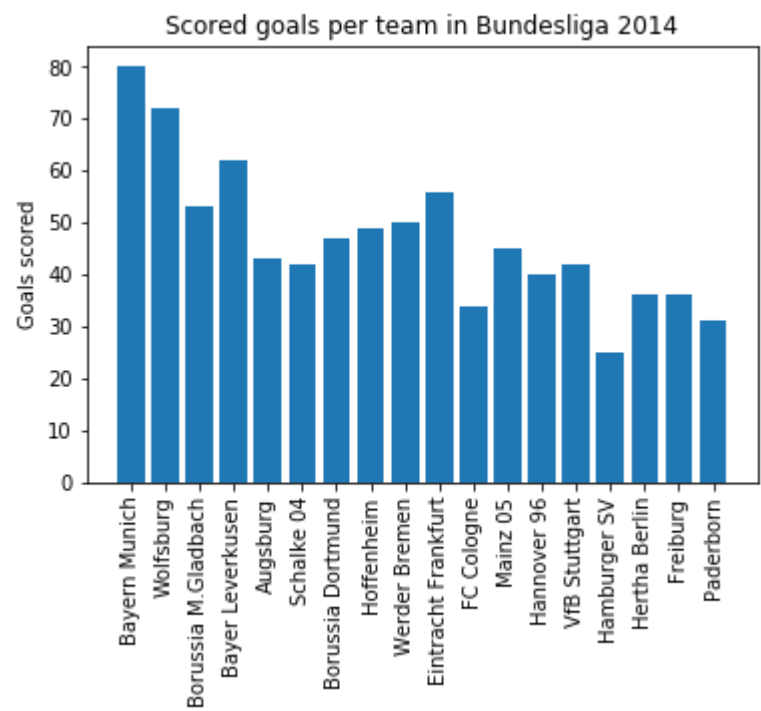
```python
df_SerieA_2016 = df_results.loc[(df_results['League'] == 'Serie_A')&(df_results['Year']
==2016)]
df_SerieA_2017 = df_results.loc[(df_results['League'] == 'Serie_A')&(df_results['Year']
==2017)]
df_SerieA_2018 = df_results.loc[(df_results['League'] == 'Serie_A')&(df_results['Year']
==2018)]
```
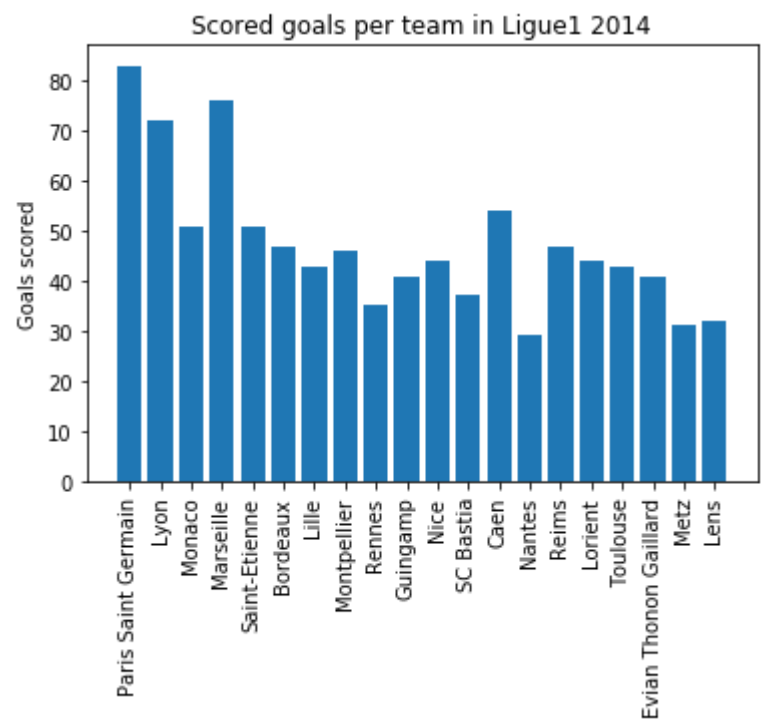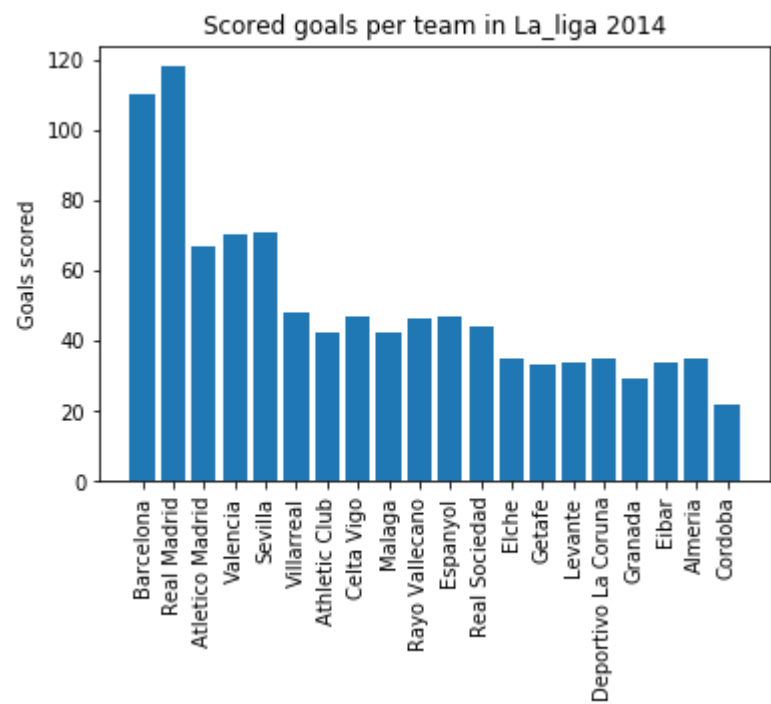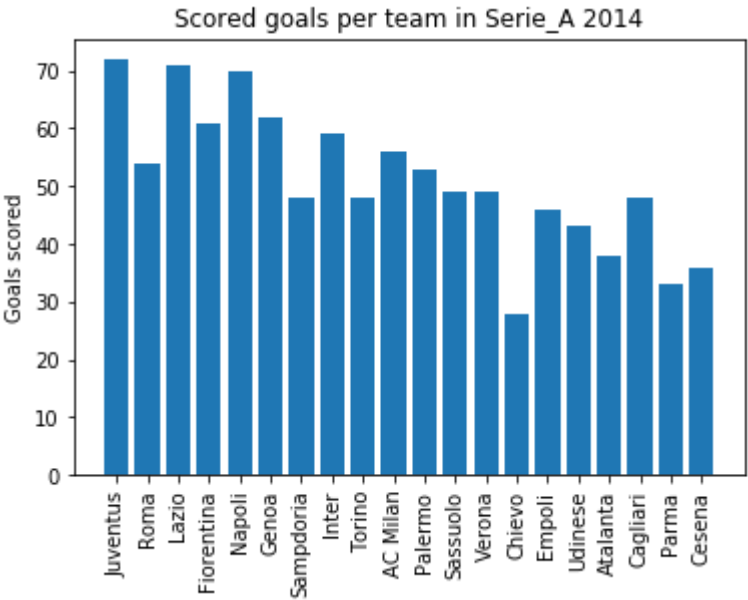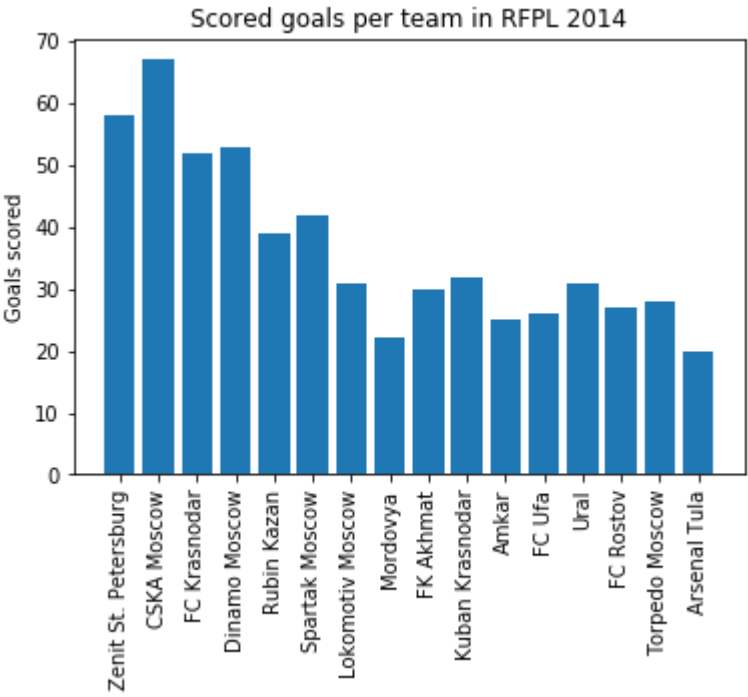
In [10]:

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

In [11]:

```python
##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()
```
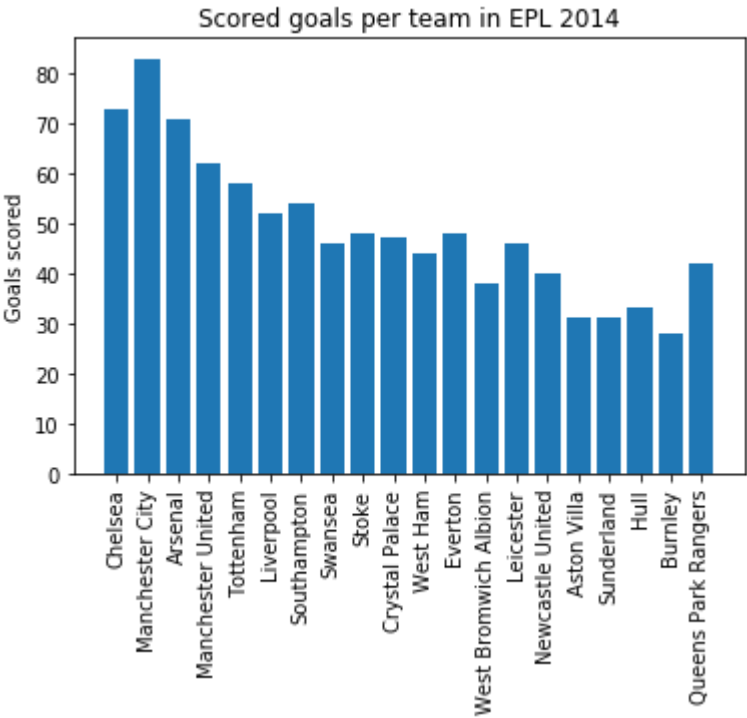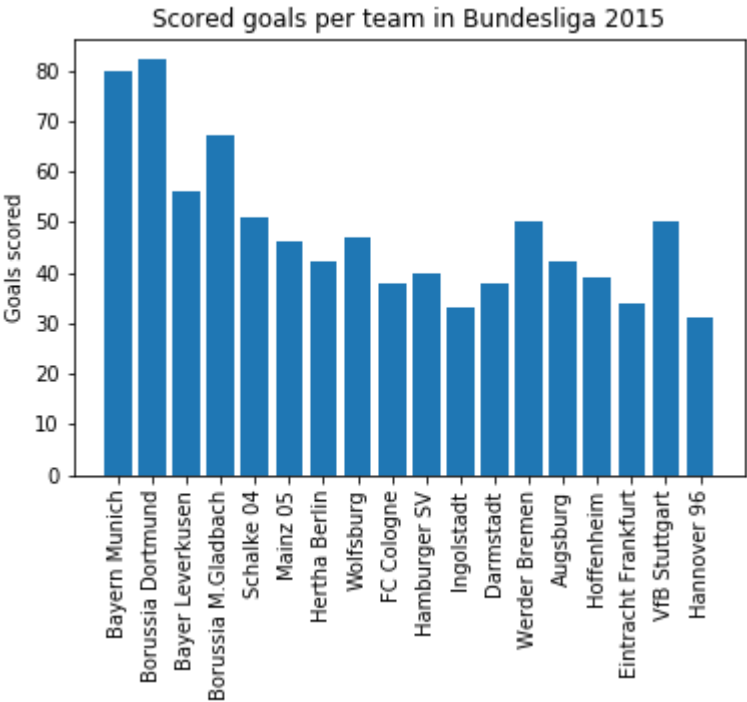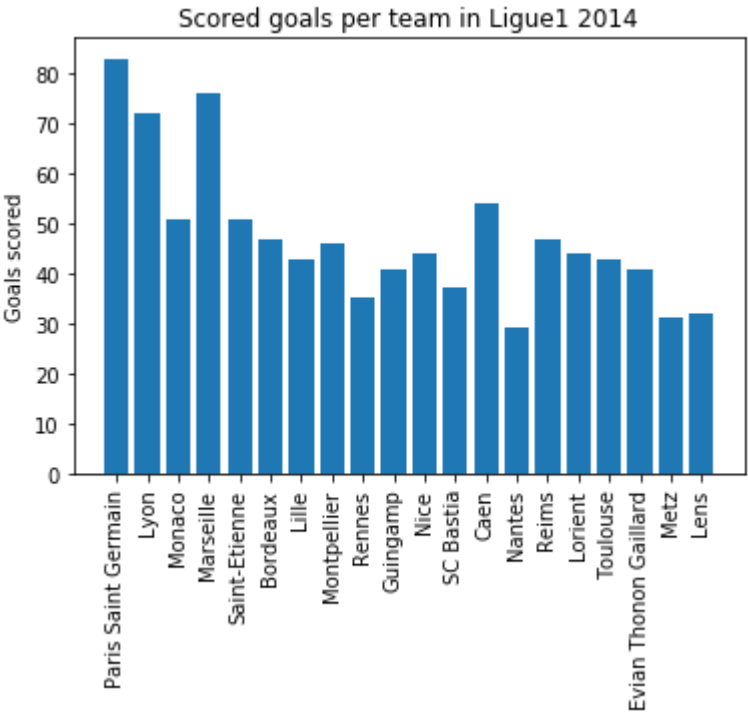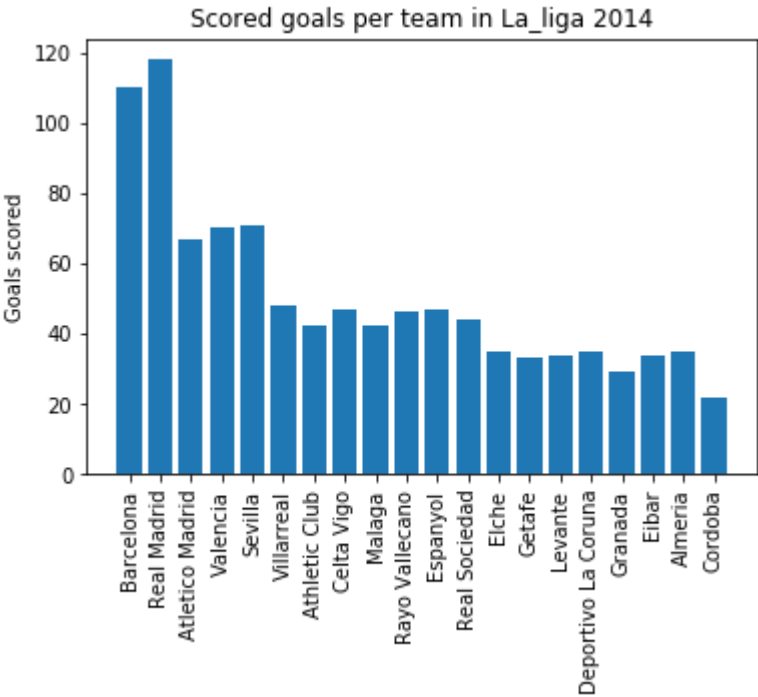
Scored goals per team in Bundesliga 2014

Scored goals per team in EPL 2014

Scored goals per team in La_liga 2014

Scored goals per team in Ligue1 2014

Scored goals per team in RFPL 2014
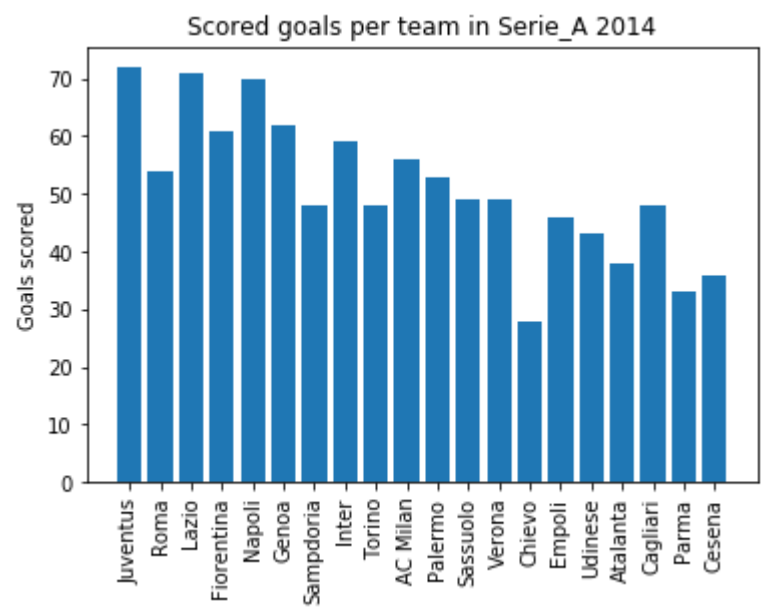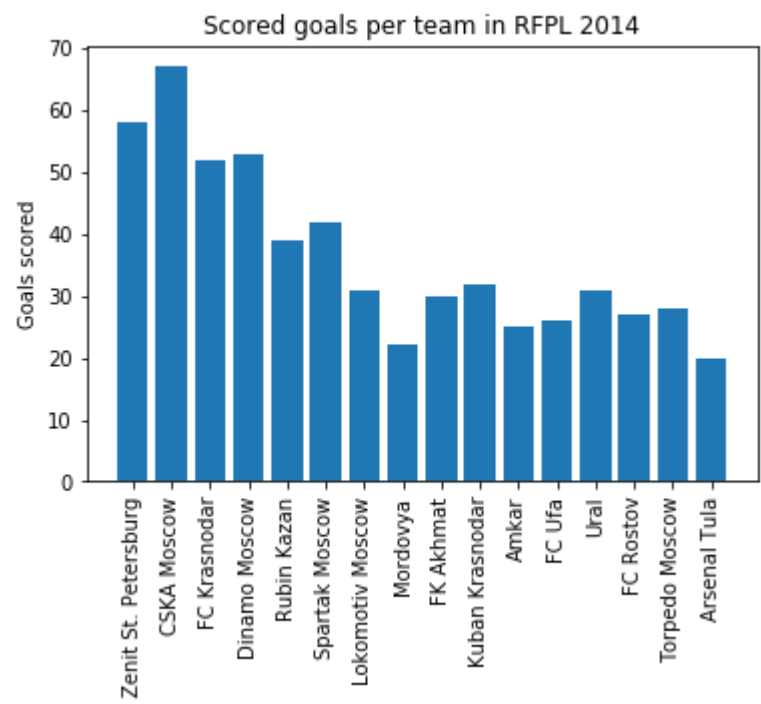

Scored goals per team in Serie_A 2014

In [12]:

```python
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

Scored goals per team in EPL 2014

Scored goals per team in La_liga 2014

Scored goals per team in Ligue1 2014

Scored goals per team in RFPL 2014


Scored goals per team in Serie_A 2014

In [13]:

```python
##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

Scored goals per team in EPL 2018

Scored goals per team in La_liga 2018

Scored goals per team in Ligue1 2018

Scored goals per team in RFPL 2018

Scored goals per team in Serie_A 2018
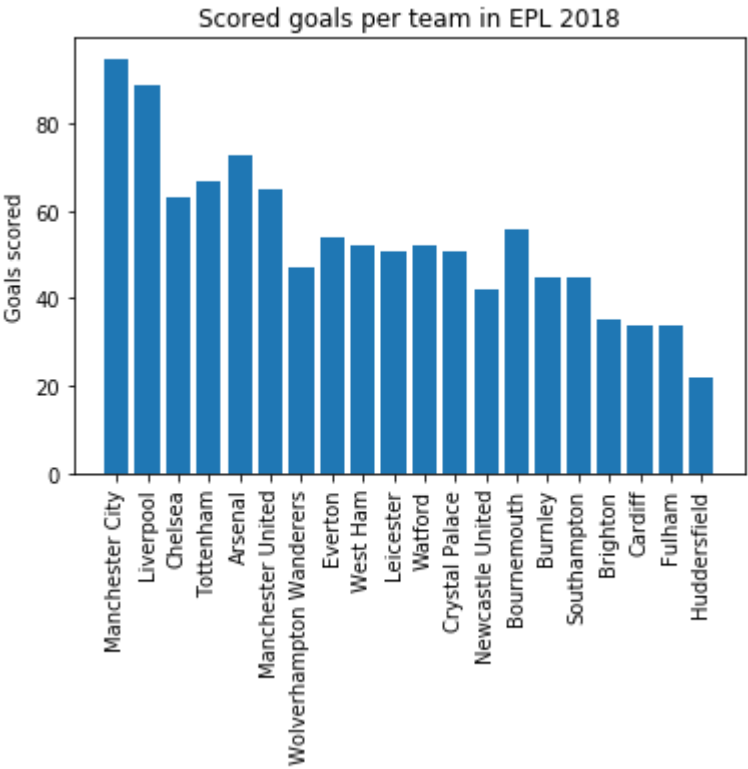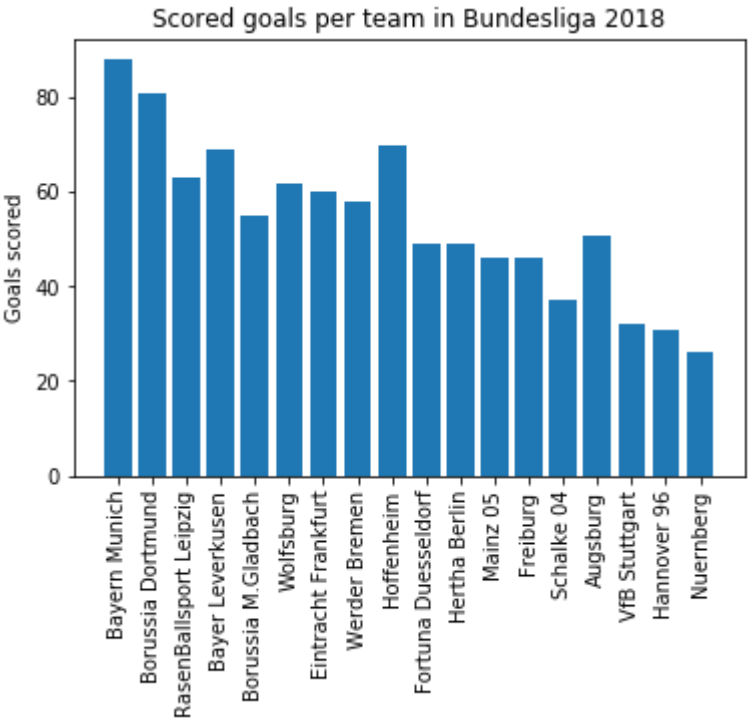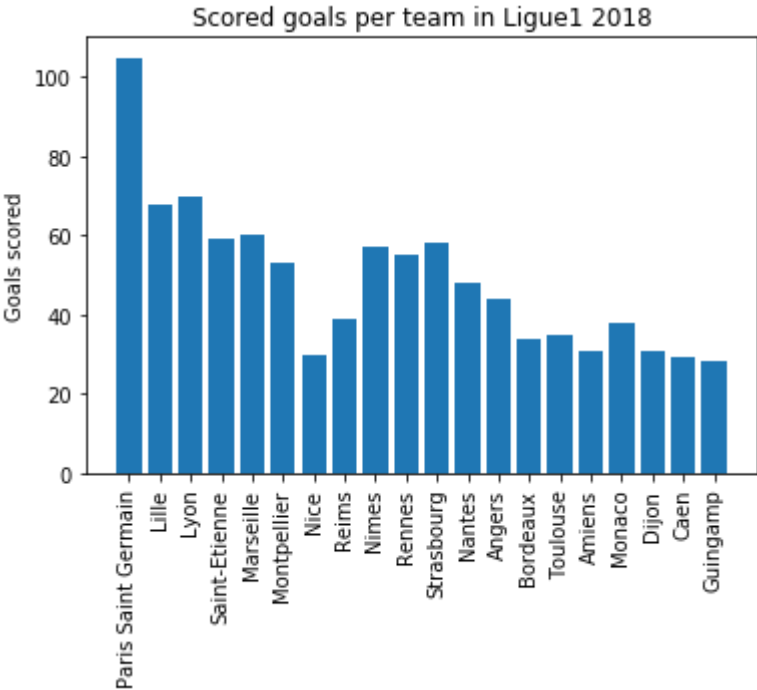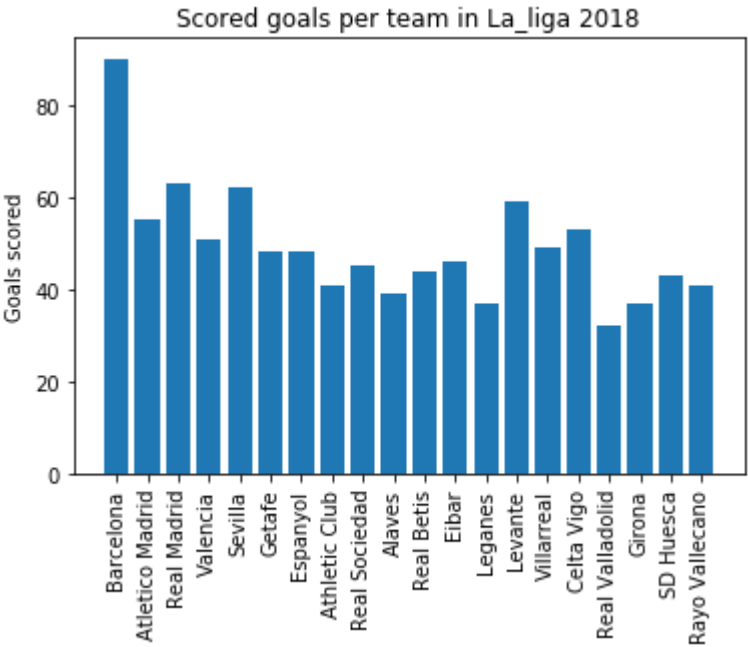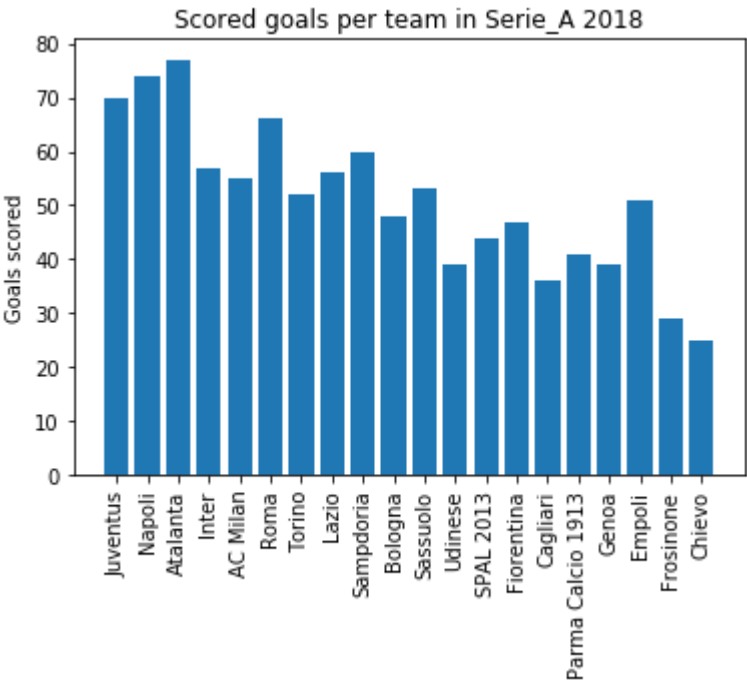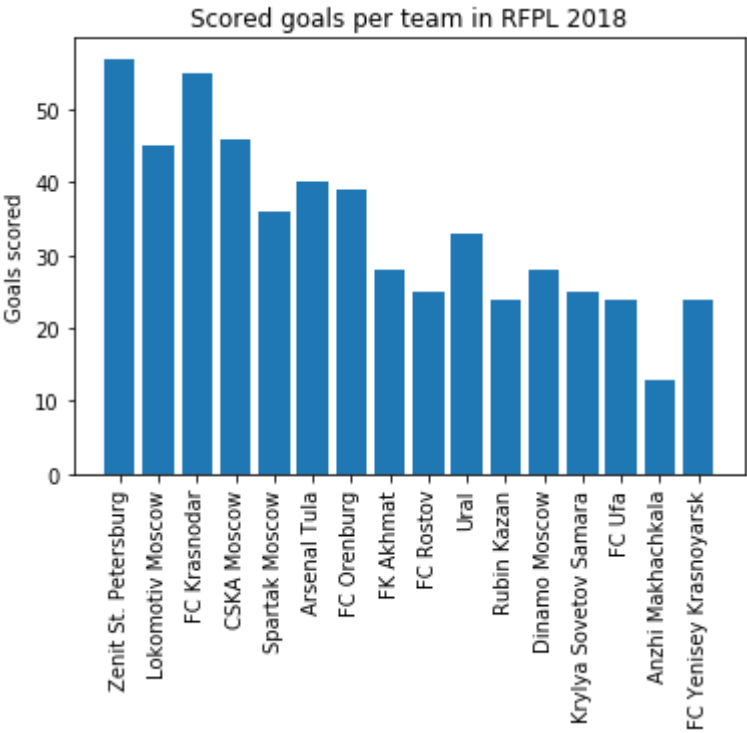
In [14]:

```python
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]:

```python
#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_2
017, 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	
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	

In [18]:

```python
#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
```
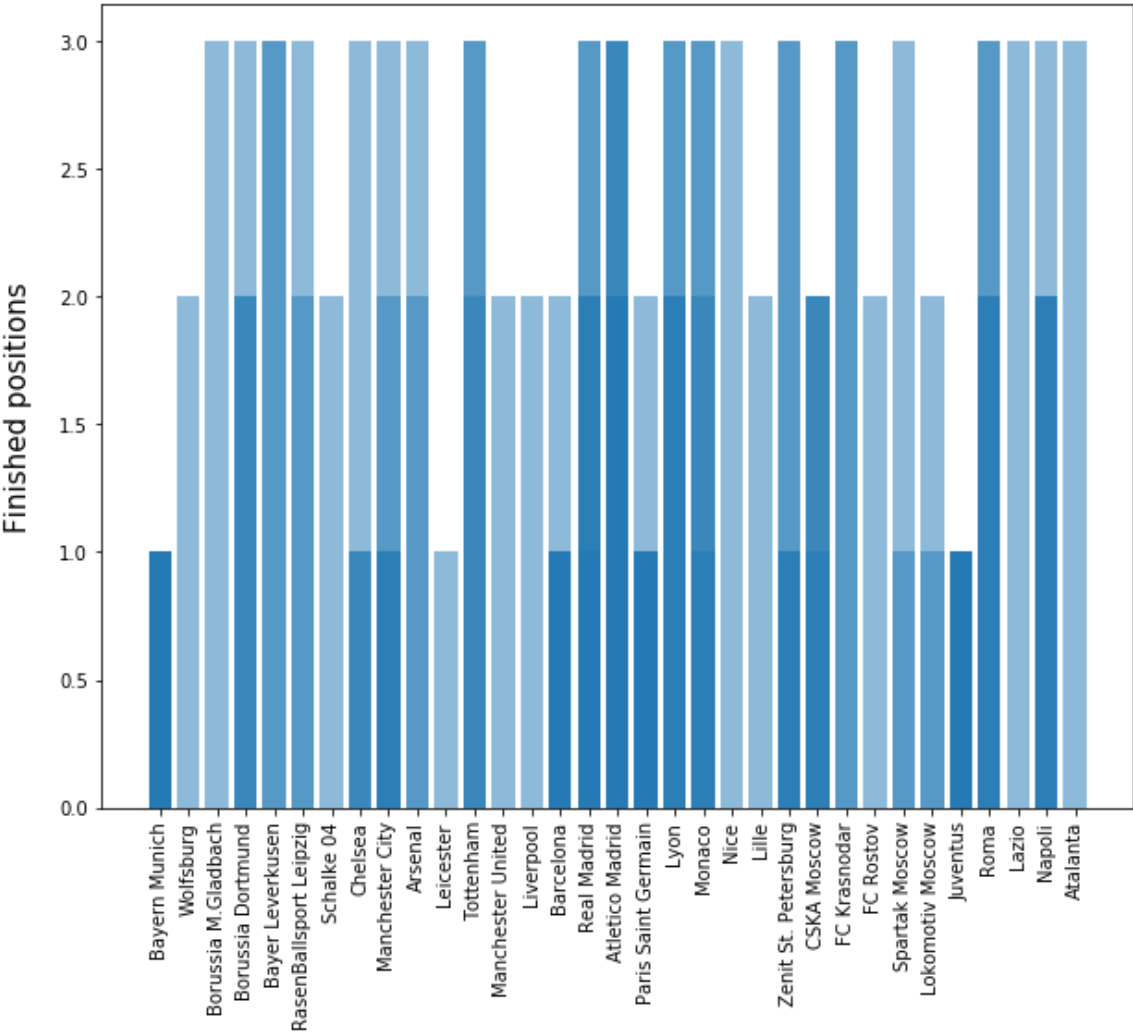
In [19]:

```python
#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

Counting 2nd place per team

Counting 3rd place per team
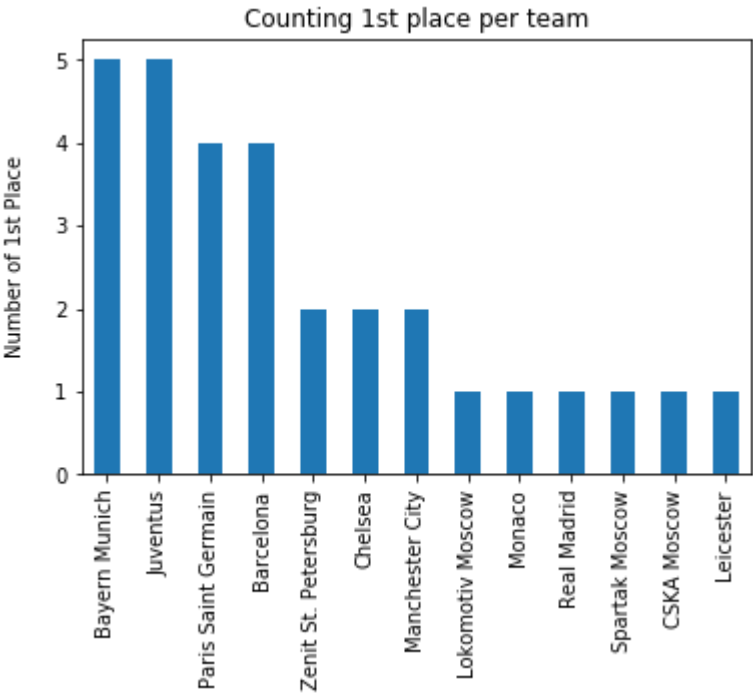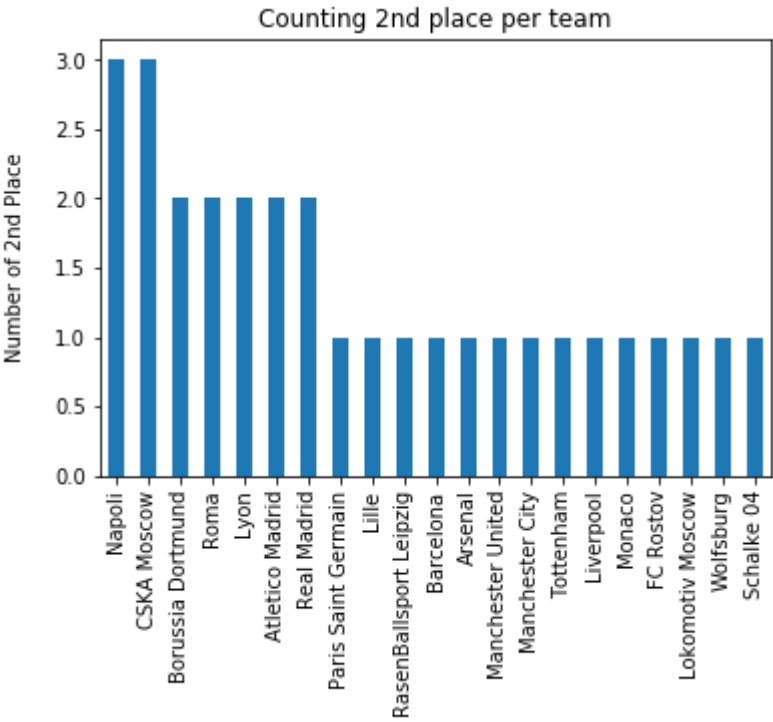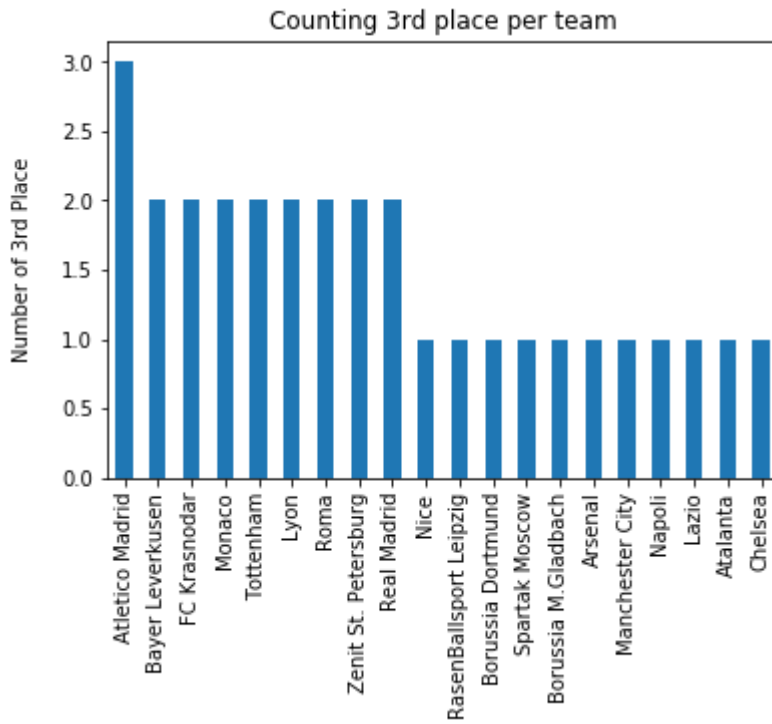
In [20]:

```python
#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]:

```python
#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',r
ot=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',r
ot=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',r
ot=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=9
0)
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',r
ot=90)
plt.ylabel("Total points", labelpad=14)
plt.title("Total points for all years per team in Serie A")
plt.show()
```
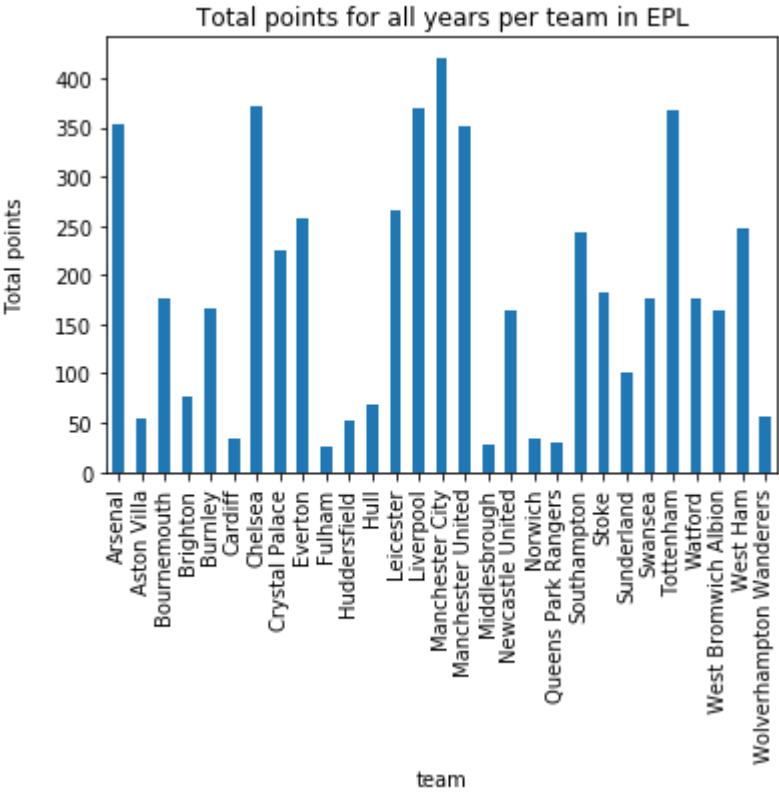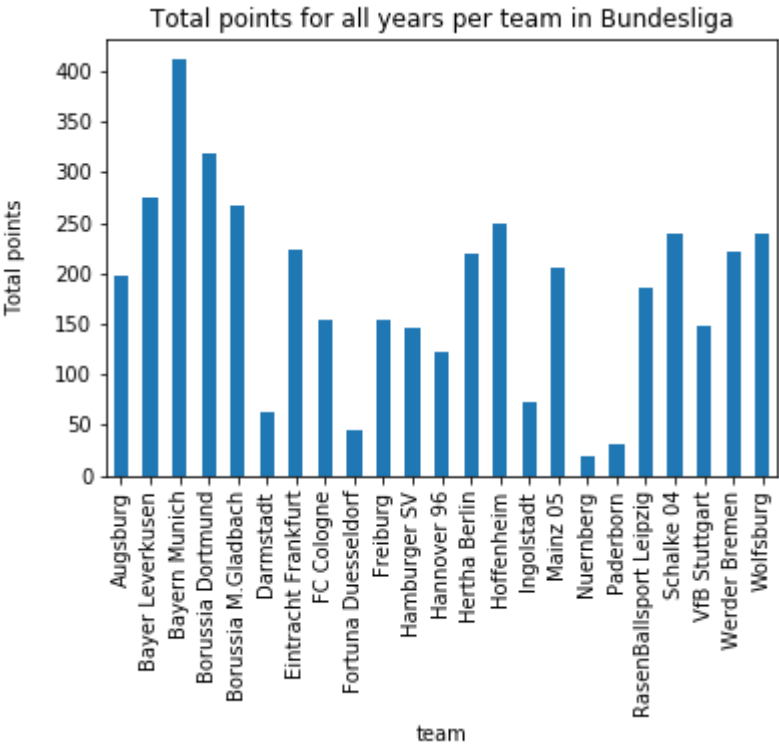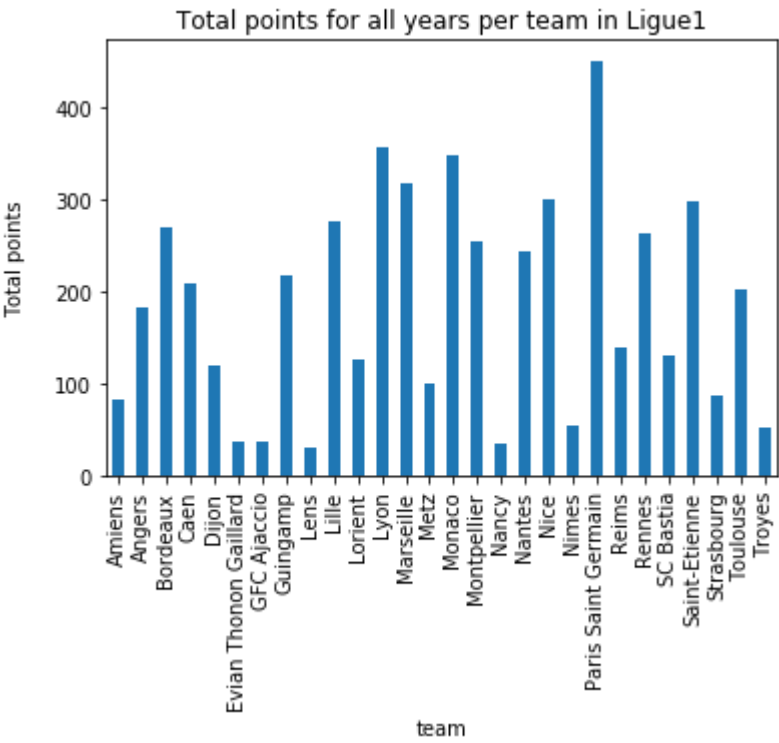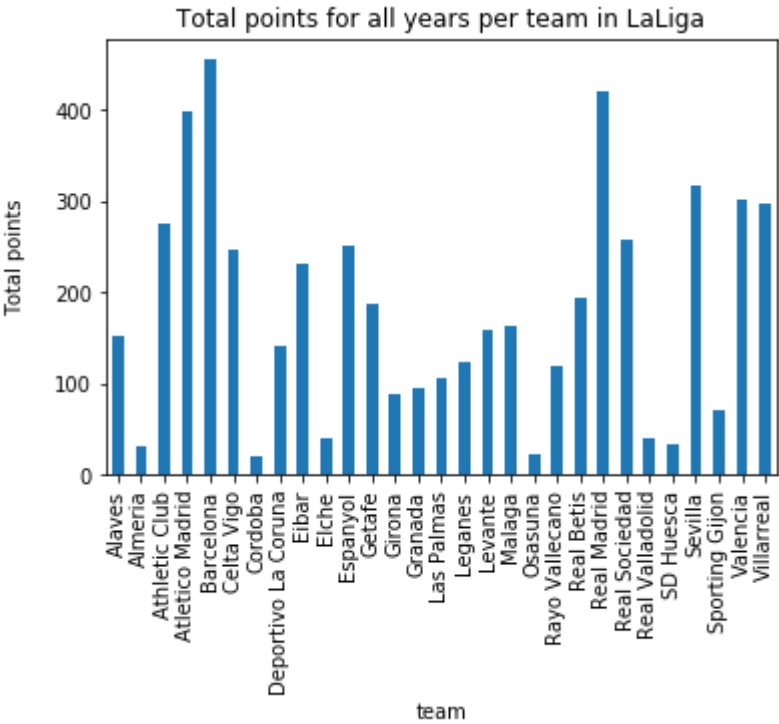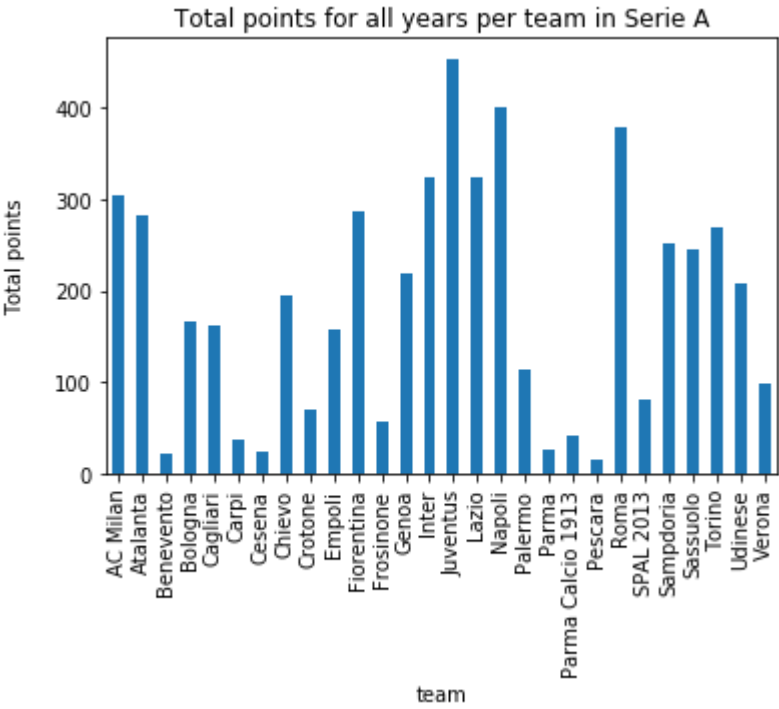
Total points for all years per team in Bundesliga

Total points for all years per team in EPL

Total points for all years per team in LaLiga

Total points for all years per team in Ligue1

Total points for all years per team in RFPL

Total points for all years per team in Serie A
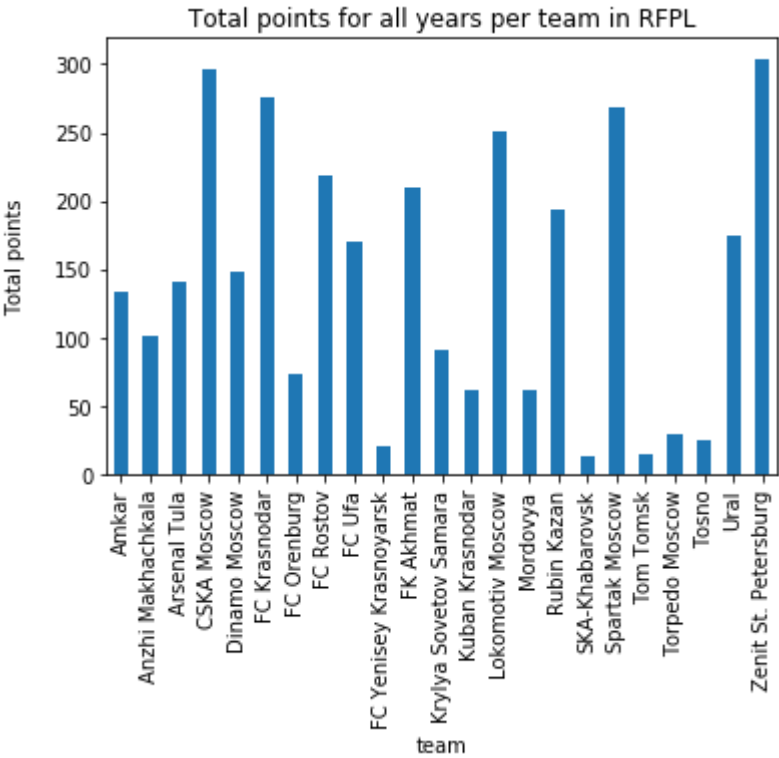
In [22]:

```
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','missed','pts']]
features_position = df_results[['Year','matches', 'wins','draws','loses','scored','missed','pts']]
```

In [25]:

```python
#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]:

```python
#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]:

```python
#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]:

```python
#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)
#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)
```

In [29]:

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

In [30]:

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

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

In [31]:

```python
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]:

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

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

In [33]:

```python
#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)
```

```
0.7736527025945974
[ 36  70  34  65  35  40  46  40  24  40  35  35  41  57  44  83  42  40
  25  56  40  53  25  25  63  38  49  70  53  35  47  46  42  31  34  71
  36  37  40  90  49  49  58 107  25  80 110  44  45  37  40  41  37  49
  51  48  40  39  65  48  31  40  35  37  51  52  37  41  37  37  28  32
  35  47  41  36  41  40  45  45  35 110  38  41  35  57  57  25  35  39
  71  71  39  45  37  48  41  45  20  41  47  36  56  30  37  56  71  47
  22  41  62  41  48  47]
```

In [34]:

```python
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]:

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

In [36]:

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

In [37]:

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

In [38]:

```python
#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)
```

```
0.93484602195551
[ 9  8 14  3  8 13 12  1 11 10  6  8 18  4 15 20 17 13 20 17  4  8 13  1
 11  9 12 14 10  6 16 14 19  2 16  6  9  7 19 11 12  4 18 15  3 15 17 13
 16  6 11 16  6  9  5 18 11 14  9 18  6 14 20  1  6  3  9  7 20  9 18  9
  3  6  7 12  3  7  5 16 13  7 11  6  2  1 17 12 16 15  9  7 14 19  4 11
  6 14 11  4  1  3  4  2  1  3 10  3  6  6 18  7 11 17]
```

In [39]:

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

Out[39]:

```
0.9362969305659015
```

In [40]:

```python
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]:

```python
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: 0.82 (+/- 0.02)
```

In [42]:

```python
#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)
[3 3 1]
```

In [43]:

```python
#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_goals
2, 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
[88 53 53]

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)
```