SerieA_ML

July 24, 2022

1 Serie A Machine Learnin Project

1.1 Importing Data

[1]:
```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import matplotlib
import numpy as np
%matplotlib inline
```

Data from: https://fbref.com/en/comps/11/Serie-A-Stats

[2]:
```
table_df = pd.read_csv('seriea_table.csv')
```

[3]:
```
table_df
```

[3]:

	Rk	Squad	MP	W	D	L	GF	GA	GD	Pts	Pts/MP	xG	xGA	\
0	1	Milan	38	26	8	4	69	31	38	86	2.26	63.1	34.8	
1	2	Inter	38	25	9	4	84	32	52	84	2.21	81.4	39.2	
2	3	Napoli	38	24	7	7	74	31	43	79	2.08	59.4	31.9	
3	4	Juventus	38	20	10	8	57	37	20	70	1.84	51.5	38.0	
4	5	Lazio	38	18	10	10	77	58	19	64	1.68	55.8	48.3	
5	6	Roma	38	18	9	11	59	43	16	63	1.66	63.7	38.5	
6	7	Fiorentina	38	19	5	14	59	51	8	62	1.63	58.8	44.1	
7	8	Atalanta	38	16	11	11	65	48	17	59	1.55	66.2	45.2	
8	9	Hellas Verona	38	14	11	13	65	59	6	53	1.39	51.8	48.9	
9	10	Torino	38	13	11	14	46	41	5	50	1.32	48.8	39.9	
10	11	Sassuolo	38	13	11	14	64	66	-2	50	1.32	56.0	67.0	
11	12	Udinese	38	11	14	13	61	58	3	47	1.24	53.1	52.0	
12	13	Bologna	38	12	10	16	44	55	-11	46	1.21	44.1	54.2	
13	14	Empoli	38	10	11	17	50	70	-20	41	1.08	46.6	67.8	
14	15	Sampdoria	38	10	6	22	46	63	-17	36	0.95	37.0	58.5	
15	16	Spezia	38	10	6	22	41	71	-30	36	0.95	39.3	67.6	
16	17	Salernitana	38	7	10	21	33	78	-45	31	0.82	37.7	65.9	
17	18	Cagliari	38	6	12	20	34	68	-34	30	0.79	39.5	61.5	
18	19	Genoa	38	4	16	18	27	60	-33	28	0.74	37.8	51.6	
19	20	Venezia	38	6	9	23	34	69	-35	27	0.71	36.1	72.6	

```
    xGD  xGD/90  Attendance                  Top Team Scorer  \
0   28.3    0.74       44015  Olivier Giroud Rafael Leão - 11
1   42.2    1.11       44473           Lautaro Martínez - 21
2   27.6    0.73       28119             Victor Osimhen - 14
3   13.4    0.35       22621               Paulo Dybala - 10
4    7.6    0.20       23263              Ciro Immobile - 27
5   25.2    0.66       41929              Tammy Abraham - 17
6   14.7    0.39       21107             Dušan Vlahović - 17
7   20.9    0.55       10447              Mario Pašalić - 13
8    2.9    0.08       13894           Giovanni Simeone - 17
9    8.9    0.23        9846             Andrea Belotti - 8
10 -10.9   -0.29        8362          Gianluca Scamacca - 16
11   1.0    0.03       12144           Gerard Deulofeu - 13
12 -10.1   -0.27       14158          Marko Arnautović - 14
13 -21.2   -0.56        6356          Andrea Pinamonti - 13
14 -21.5   -0.57        9417          Francesco Caputo - 11
15 -28.3   -0.74        6709              Daniele Verde - 8
16 -28.2   -0.74       15073        Federico Bonazzoli - 10
17 -22.0   -0.58        9718                João Pedro - 13
18 -13.8   -0.36       12326              Mattia Destro - 9
19 -36.5   -0.96        6648               Thomas Henry - 9

                Goalkeeper                                      Notes
0              Mike Maignan        → Champions League via league finish
1          Samir Handanović        → Champions League via league finish
2              David Ospina        → Champions League via league finish
3        Wojciech Szczęsny        → Champions League via league finish
4          Thomas Strakosha           → Europa League via league finish
5              Rui Patrício           → Europa League via league finish
6        Pietro Terracciano → Europa Conference League via league finish
7                Juan Musso                                         NaN
8          Lorenzo Montipò                                          NaN
9    Vanja Milinković-Savić                                         NaN
10          Andrea Consigli                                         NaN
11          Marco Silvestri                                         NaN
12          Łukasz Skorupski                                        NaN
13        Guglielmo Vicario                                         NaN
14             Emil Audero                                          NaN
15           Ivan Provedel                                          NaN
16               Vid Belec                                          NaN
17          Alessio Cragno                                    Relegated
18        Salvatore Sirigu                                    Relegated
19             Niki Mäenpää                                   Relegated
```

[4]:

```
table_df.drop(columns=['Notes', 'Goalkeeper', 'Top Team Scorer', 'MP'],
 ↪inplace=True)
```

[5]:
```
table_df
```

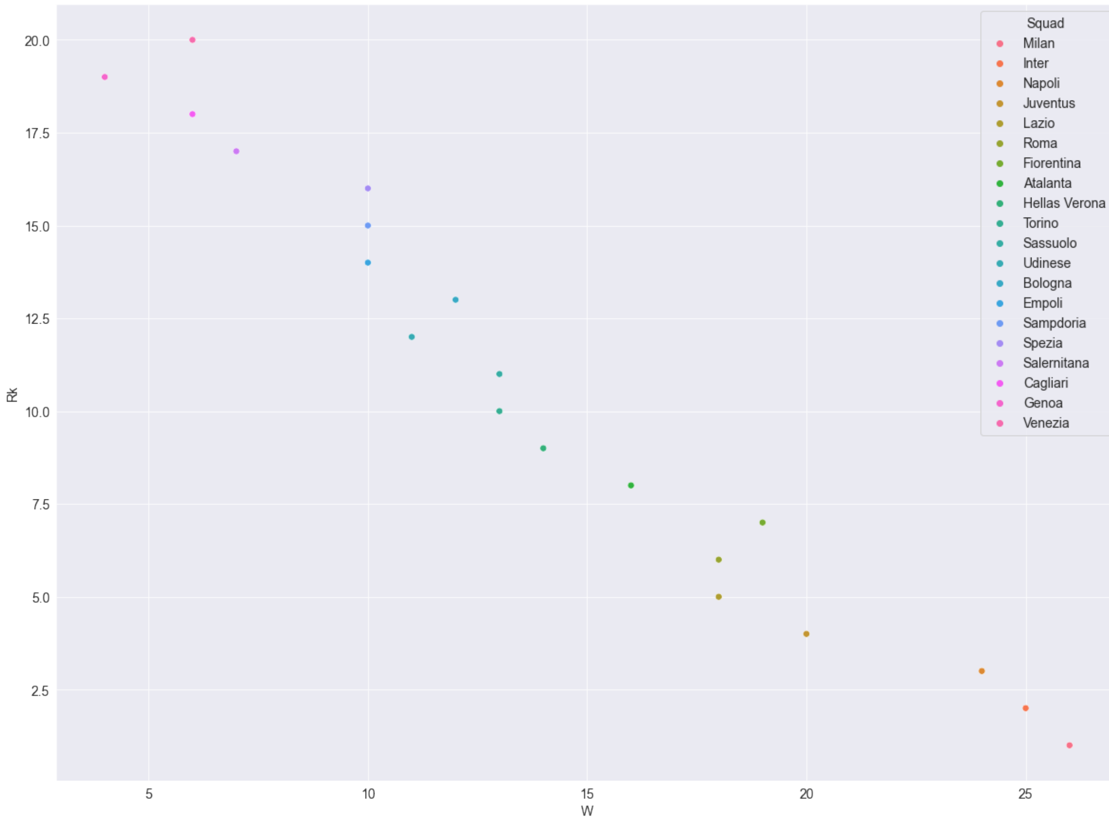
[5]:
```
    Rk          Squad   W   D   L  GF  GA  GD  Pts  Pts/MP    xG   xGA   xGD  \
0    1          Milan  26   8   4  69  31  38   86    2.26  63.1  34.8  28.3
1    2          Inter  25   9   4  84  32  52   84    2.21  81.4  39.2  42.2
2    3         Napoli  24   7   7  74  31  43   79    2.08  59.4  31.9  27.6
3    4       Juventus  20  10   8  57  37  20   70    1.84  51.5  38.0  13.4
4    5          Lazio  18  10  10  77  58  19   64    1.68  55.8  48.3   7.6
5    6           Roma  18   9  11  59  43  16   63    1.66  63.7  38.5  25.2
6    7     Fiorentina  19   5  14  59  51   8   62    1.63  58.8  44.1  14.7
7    8       Atalanta  16  11  11  65  48  17   59    1.55  66.2  45.2  20.9
8    9  Hellas Verona  14  11  13  65  59   6   53    1.39  51.8  48.9   2.9
9   10         Torino  13  11  14  46  41   5   50    1.32  48.8  39.9   8.9
10  11       Sassuolo  13  11  14  64  66  -2   50    1.32  56.0  67.0 -10.9
11  12        Udinese  11  14  13  61  58   3   47    1.24  53.1  52.0   1.0
12  13        Bologna  12  10  16  44  55 -11   46    1.21  44.1  54.2 -10.1
13  14         Empoli  10  11  17  50  70 -20   41    1.08  46.6  67.8 -21.2
14  15      Sampdoria  10   6  22  46  63 -17   36    0.95  37.0  58.5 -21.5
15  16         Spezia  10   6  22  41  71 -30   36    0.95  39.3  67.6 -28.3
16  17    Salernitana   7  10  21  33  78 -45   31    0.82  37.7  65.9 -28.2
17  18       Cagliari   6  12  20  34  68 -34   30    0.79  39.5  61.5 -22.0
18  19          Genoa   4  16  18  27  60 -33   28    0.74  37.8  51.6 -13.8
19  20        Venezia   6   9  23  34  69 -35   27    0.71  36.1  72.6 -36.5

    xGD/90  Attendance
0     0.74       44015
1     1.11       44473
2     0.73       28119
3     0.35       22621
4     0.20       23263
5     0.66       41929
6     0.39       21107
7     0.55       10447
8     0.08       13894
9     0.23        9846
10   -0.29        8362
11    0.03       12144
12   -0.27       14158
13   -0.56        6356
14   -0.57        9417
15   -0.74        6709
16   -0.74       15073
17   -0.58        9718
18   -0.36       12326
19   -0.96        6648
```

[6]:
```
table_df.shape
```

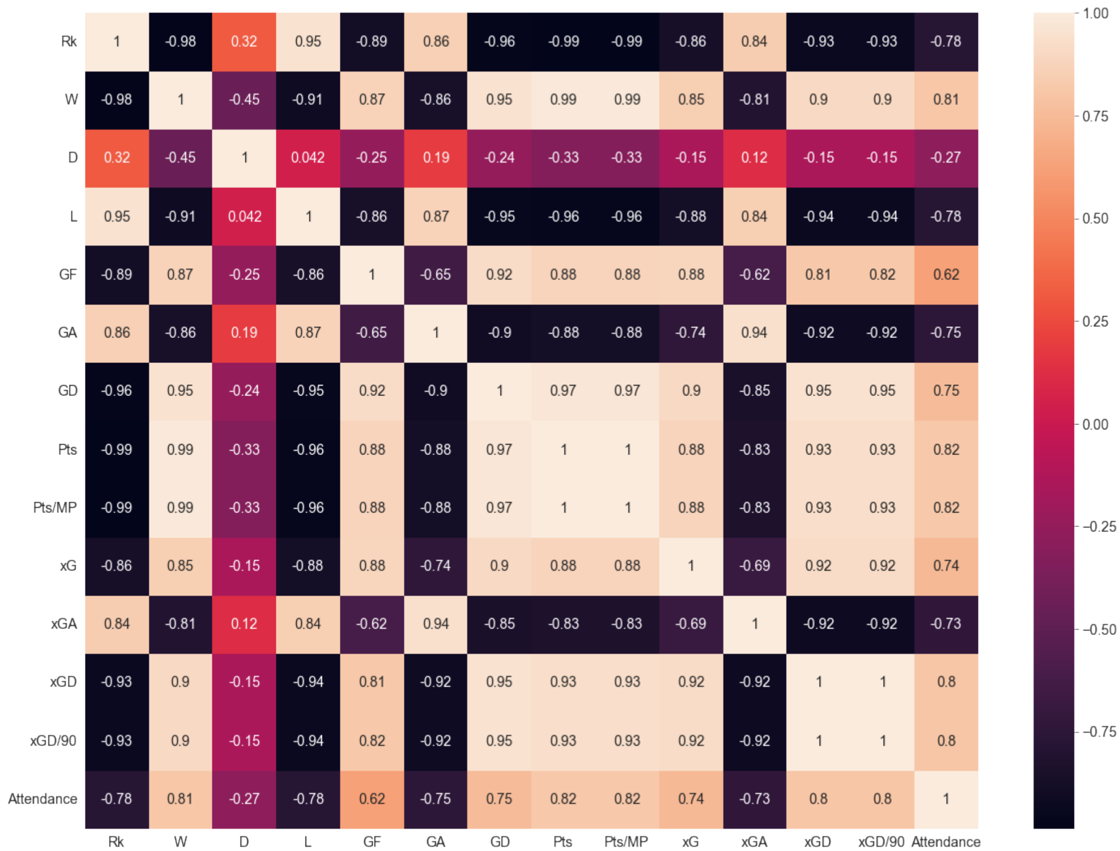
[6]: (20, 15)

[19]:
```
sns.set_style('darkgrid')
matplotlib.rcParams['font.size'] = 14
matplotlib.rcParams['figure.figsize'] = (20, 15)
matplotlib.rcParams['figure.facecolor'] = '#00000000'
```

[20]:
```
sns.scatterplot(data=table_df, x='W', y='Rk', hue='Squad', s=50);
```



[21]:
```
sns.heatmap(table_df.corr(), annot=True);
```



1.2 Training Preparation

Defining inputs and targets

[22]:
```
table_df
```

[22]:

	Rk	Squad	W	D	L	GF	GA	GD	Pts	Pts/MP	xG	xGA	xGD	\
0	1	Milan	26	8	4	69	31	38	86	2.26	63.1	34.8	28.3	
1	2	Inter	25	9	4	84	32	52	84	2.21	81.4	39.2	42.2	
2	3	Napoli	24	7	7	74	31	43	79	2.08	59.4	31.9	27.6	
3	4	Juventus	20	10	8	57	37	20	70	1.84	51.5	38.0	13.4	
4	5	Lazio	18	10	10	77	58	19	64	1.68	55.8	48.3	7.6	
5	6	Roma	18	9	11	59	43	16	63	1.66	63.7	38.5	25.2	
6	7	Fiorentina	19	5	14	59	51	8	62	1.63	58.8	44.1	14.7	
7	8	Atalanta	16	11	11	65	48	17	59	1.55	66.2	45.2	20.9	
8	9	Hellas Verona	14	11	13	65	59	6	53	1.39	51.8	48.9	2.9	
9	10	Torino	13	11	14	46	41	5	50	1.32	48.8	39.9	8.9	
10	11	Sassuolo	13	11	14	64	66	-2	50	1.32	56.0	67.0	-10.9	
11	12	Udinese	11	14	13	61	58	3	47	1.24	53.1	52.0	1.0	
12	13	Bologna	12	10	16	44	55	-11	46	1.21	44.1	54.2	-10.1	

```
13  14         Empoli  10  11  17  50  70 -20   41     1.08  46.6  67.8 -21.2
14  15      Sampdoria  10   6  22  46  63 -17   36     0.95  37.0  58.5 -21.5
15  16         Spezia  10   6  22  41  71 -30   36     0.95  39.3  67.6 -28.3
16  17    Salernitana   7  10  21  33  78 -45   31     0.82  37.7  65.9 -28.2
17  18       Cagliari   6  12  20  34  68 -34   30     0.79  39.5  61.5 -22.0
18  19          Genoa   4  16  18  27  60 -33   28     0.74  37.8  51.6 -13.8
19  20        Venezia   6   9  23  34  69 -35   27     0.71  36.1  72.6 -36.5

    xGD/90  Attendance
0     0.74       44015
1     1.11       44473
2     0.73       28119
3     0.35       22621
4     0.20       23263
5     0.66       41929
6     0.39       21107
7     0.55       10447
8     0.08       13894
9     0.23        9846
10   -0.29        8362
11    0.03       12144
12   -0.27       14158
13   -0.56        6356
14   -0.57        9417
15   -0.74        6709
16   -0.74       15073
17   -0.58        9718
18   -0.36       12326
19   -0.96        6648
```

[27]:
```python
input_cols = list(table_df.columns[2:])
target_col = 'Rk'
```

[31]:
```python
print('Input:', input_cols)
print('Target:', target_col)
```

```
Input: ['W', 'D', 'L', 'GF', 'GA', 'GD', 'Pts', 'Pts/MP', 'xG', 'xGA', 'xGD',
'xGD/90', 'Attendance']
Target: Rk
```

Creating the Two Different Dataframes

[43]:
```python
inputs_df = table_df[input_cols].copy()
target_df = table_df[target_col]
```

[44]:
```python
inputs_df
```

[44]:

	W	D	L	GF	GA	GD	Pts	Pts/MP	xG	xGA	xGD	xGD/90	Attendance
0	26	8	4	69	31	38	86	2.26	63.1	34.8	28.3	0.74	44015
1	25	9	4	84	32	52	84	2.21	81.4	39.2	42.2	1.11	44473
2	24	7	7	74	31	43	79	2.08	59.4	31.9	27.6	0.73	28119
3	20	10	8	57	37	20	70	1.84	51.5	38.0	13.4	0.35	22621
4	18	10	10	77	58	19	64	1.68	55.8	48.3	7.6	0.20	23263
5	18	9	11	59	43	16	63	1.66	63.7	38.5	25.2	0.66	41929
6	19	5	14	59	51	8	62	1.63	58.8	44.1	14.7	0.39	21107
7	16	11	11	65	48	17	59	1.55	66.2	45.2	20.9	0.55	10447
8	14	11	13	65	59	6	53	1.39	51.8	48.9	2.9	0.08	13894
9	13	11	14	46	41	5	50	1.32	48.8	39.9	8.9	0.23	9846
10	13	11	14	64	66	-2	50	1.32	56.0	67.0	-10.9	-0.29	8362
11	11	14	13	61	58	3	47	1.24	53.1	52.0	1.0	0.03	12144
12	12	10	16	44	55	-11	46	1.21	44.1	54.2	-10.1	-0.27	14158
13	10	11	17	50	70	-20	41	1.08	46.6	67.8	-21.2	-0.56	6356
14	10	6	22	46	63	-17	36	0.95	37.0	58.5	-21.5	-0.57	9417
15	10	6	22	41	71	-30	36	0.95	39.3	67.6	-28.3	-0.74	6709
16	7	10	21	33	78	-45	31	0.82	37.7	65.9	-28.2	-0.74	15073
17	6	12	20	34	68	-34	30	0.79	39.5	61.5	-22.0	-0.58	9718
18	4	16	18	27	60	-33	28	0.74	37.8	51.6	-13.8	-0.36	12326
19	6	9	23	34	69	-35	27	0.71	36.1	72.6	-36.5	-0.96	6648

[37]:

```
target_df
```

[37]:

```
0      1
1      2
2      3
3      4
4      5
5      6
6      7
7      8
8      9
9     10
10    11
11    12
12    13
13    14
14    15
15    16
16    17
17    18
18    19
19    20
Name: Rk, dtype: int64
```

1.2.1 Scaling Numerical Values

[46]:
```python
numerical_cols = list(inputs_df.columns)
```

[47]:
```python
numerical_cols
```

[47]:
```
['W',
 'D',
 'L',
 'GF',
 'GA',
 'GD',
 'Pts',
 'Pts/MP',
 'xG',
 'xGA',
 'xGD',
 'xGD/90',
 'Attendance']
```

[38]:
```python
from sklearn.preprocessing import MinMaxScaler
```

[49]:
```python
scaler = MinMaxScaler().fit(inputs_df[numerical_cols])
```

[50]:
```python
inputs_df[numerical_cols] = scaler.transform(inputs_df[numerical_cols])
```

[51]:
```python
inputs_df
```

[51]:

	W	D	L	GF	GA	GD	Pts	\
0	1.000000	0.272727	0.000000	0.736842	0.000000	0.855670	1.000000	
1	0.954545	0.363636	0.000000	1.000000	0.021277	1.000000	0.966102	
2	0.909091	0.181818	0.157895	0.824561	0.000000	0.907216	0.881356	
3	0.727273	0.454545	0.210526	0.526316	0.127660	0.670103	0.728814	
4	0.636364	0.454545	0.315789	0.877193	0.574468	0.659794	0.627119	
5	0.636364	0.363636	0.368421	0.561404	0.255319	0.628866	0.610169	
6	0.681818	0.000000	0.526316	0.561404	0.425532	0.546392	0.593220	
7	0.545455	0.545455	0.368421	0.666667	0.361702	0.639175	0.542373	
8	0.454545	0.545455	0.473684	0.666667	0.595745	0.525773	0.440678	
9	0.409091	0.545455	0.526316	0.333333	0.212766	0.515464	0.389831	
10	0.409091	0.545455	0.526316	0.649123	0.744681	0.443299	0.389831	
11	0.318182	0.818182	0.473684	0.596491	0.574468	0.494845	0.338983	
12	0.363636	0.454545	0.631579	0.298246	0.510638	0.350515	0.322034	
13	0.272727	0.545455	0.684211	0.403509	0.829787	0.257732	0.237288	
14	0.272727	0.090909	0.947368	0.333333	0.680851	0.288660	0.152542	
15	0.272727	0.090909	0.947368	0.245614	0.851064	0.154639	0.152542	
16	0.136364	0.454545	0.894737	0.105263	1.000000	0.000000	0.067797	
17	0.090909	0.636364	0.842105	0.122807	0.787234	0.113402	0.050847	
18	0.000000	1.000000	0.736842	0.000000	0.617021	0.123711	0.016949	

19	0.090909	0.363636	1.000000	0.122807	0.808511	0.103093	0.000000

	Pts/MP	xG	xGA	xGD	xGD/90	Attendance
0	1.000000	0.596026	0.071253	0.823380	0.821256	0.987984
1	0.967742	1.000000	0.179361	1.000000	1.000000	1.000000
2	0.883871	0.514349	0.000000	0.814485	0.816425	0.570953
3	0.729032	0.339956	0.149877	0.634053	0.632850	0.426712
4	0.625806	0.434879	0.402948	0.560356	0.560386	0.443555
5	0.612903	0.609272	0.162162	0.783990	0.782609	0.933258
6	0.593548	0.501104	0.299754	0.650572	0.652174	0.386993
7	0.541935	0.664459	0.326781	0.729352	0.729469	0.107327
8	0.438710	0.346578	0.417690	0.500635	0.502415	0.197760
9	0.393548	0.280353	0.196560	0.576874	0.574879	0.091560
10	0.393548	0.439294	0.862408	0.325286	0.323671	0.052627
11	0.341935	0.375276	0.493857	0.476493	0.478261	0.151848
12	0.322581	0.176600	0.547912	0.335451	0.333333	0.204686
13	0.238710	0.231788	0.882064	0.194409	0.193237	0.000000
14	0.154839	0.019868	0.653563	0.190597	0.188406	0.080305
15	0.154839	0.070640	0.877150	0.104193	0.106280	0.009261
16	0.070968	0.035320	0.835381	0.105464	0.106280	0.228691
17	0.051613	0.075055	0.727273	0.184244	0.183575	0.088202
18	0.019355	0.037528	0.484029	0.288437	0.289855	0.156623
19	0.000000	0.000000	1.000000	0.000000	0.000000	0.007661

1.3 Training, Validation and Test Set

[52]:
```
from sklearn.model_selection import train_test_split
```

[57]:
```
train_inputs, val_inputs, train_targets, val_targets =␣
↪train_test_split(inputs_df, target_df, test_size=0.20, random_state=42)
```

[58]:
```
train_inputs
```

[58]:

	W	D	L	GF	GA	GD	Pts \
8	0.454545	0.545455	0.473684	0.666667	0.595745	0.525773	0.440678
5	0.636364	0.363636	0.368421	0.561404	0.255319	0.628866	0.610169
11	0.318182	0.818182	0.473684	0.596491	0.574468	0.494845	0.338983
3	0.727273	0.454545	0.210526	0.526316	0.127660	0.670103	0.728814
18	0.000000	1.000000	0.736842	0.000000	0.617021	0.123711	0.016949
16	0.136364	0.454545	0.894737	0.105263	1.000000	0.000000	0.067797
13	0.272727	0.545455	0.684211	0.403509	0.829787	0.257732	0.237288
2	0.909091	0.181818	0.157895	0.824561	0.000000	0.907216	0.881356
9	0.409091	0.545455	0.526316	0.333333	0.212766	0.515464	0.389831
19	0.090909	0.363636	1.000000	0.122807	0.808511	0.103093	0.000000
4	0.636364	0.454545	0.315789	0.877193	0.574468	0.659794	0.627119
12	0.363636	0.454545	0.631579	0.298246	0.510638	0.350515	0.322034
7	0.545455	0.545455	0.368421	0.666667	0.361702	0.639175	0.542373

```
10  0.409091  0.545455  0.526316  0.649123  0.744681  0.443299  0.389831
14  0.272727  0.090909  0.947368  0.333333  0.680851  0.288660  0.152542
6   0.681818  0.000000  0.526316  0.561404  0.425532  0.546392  0.593220

      Pts/MP        xG       xGA       xGD    xGD/90  Attendance
8   0.438710  0.346578  0.417690  0.500635  0.502415    0.197760
5   0.612903  0.609272  0.162162  0.783990  0.782609    0.933258
11  0.341935  0.375276  0.493857  0.476493  0.478261    0.151848
3   0.729032  0.339956  0.149877  0.634053  0.632850    0.426712
18  0.019355  0.037528  0.484029  0.288437  0.289855    0.156623
16  0.070968  0.035320  0.835381  0.105464  0.106280    0.228691
13  0.238710  0.231788  0.882064  0.194409  0.193237    0.000000
2   0.883871  0.514349  0.000000  0.814485  0.816425    0.570953
9   0.393548  0.280353  0.196560  0.576874  0.574879    0.091560
19  0.000000  0.000000  1.000000  0.000000  0.000000    0.007661
4   0.625806  0.434879  0.402948  0.560356  0.560386    0.443555
12  0.322581  0.176600  0.547912  0.335451  0.333333    0.204686
7   0.541935  0.664459  0.326781  0.729352  0.729469    0.107327
10  0.393548  0.439294  0.862408  0.325286  0.323671    0.052627
14  0.154839  0.019868  0.653563  0.190597  0.188406    0.080305
6   0.593548  0.501104  0.299754  0.650572  0.652174    0.386993
```

[59]:
```
train_targets
```

[59]:
```
8      9
5      6
11    12
3      4
18    19
16    17
13    14
2      3
9     10
19    20
4      5
12    13
7      8
10    11
14    15
6      7
Name: Rk, dtype: int64
```

[60]:
```
val_inputs
```

[60]:
```
           W         D         L        GF        GA        GD       Pts  \
0   1.000000  0.272727  0.000000  0.736842  0.000000  0.855670  1.000000
17  0.090909  0.636364  0.842105  0.122807  0.787234  0.113402  0.050847
```

```
15  0.272727  0.090909  0.947368  0.245614  0.851064  0.154639  0.152542
1   0.954545  0.363636  0.000000  1.000000  0.021277  1.000000  0.966102

      Pts/MP        xG       xGA       xGD    xGD/90  Attendance
0   1.000000  0.596026  0.071253  0.823380  0.821256    0.987984
17  0.051613  0.075055  0.727273  0.184244  0.183575    0.088202
15  0.154839  0.070640  0.877150  0.104193  0.106280    0.009261
1   0.967742  1.000000  0.179361  1.000000  1.000000    1.000000
```

[61]:
```
val_targets
```

[61]:
```
0      1
17    18
15    16
1      2
Name: Rk, dtype: int64
```

2 Train a Linear Regression Model

[78]:
```
from sklearn.linear_model import Ridge
from sklearn.metrics import mean_squared_error
```

2.0.1 Train

[79]:
```
%%time
model = Ridge().fit(train_inputs, train_targets)
```

```
CPU times: total: 0 ns
Wall time: 14.1 ms
```

[80]:
```
train_preds = model.predict(train_inputs)
```

[81]:
```
train_preds
```

[81]:
```
array([ 9.77093948,  5.88979012, 11.15823153,  6.04573319, 16.92544041,
       17.26138736, 14.43950518,  2.3812202 , 10.38990622, 18.16848165,
        6.86681754, 12.57885414,  7.52433463, 11.52068121, 14.79206149,
        7.28661564])
```

[82]:
```
train_targets
```

[82]:
```
8      9
5      6
11    12
3      4
18    19
16    17
```

```
13    14
2      3
9     10
19    20
4      5
12    13
7      8
10    11
14    15
6      7
Name: Rk, dtype: int64
```

[83]:
```python
train_rmse = mean_squared_error(train_preds, train_targets, squared=False )
print('The position is wrong of {} places'.format(train_rmse))
```

```
The position is wrong of 1.0677042461643216 places
```

2.0.2 Validation

[84]:
```python
model = Ridge().fit(val_inputs, val_targets)
```

[85]:
```python
val_preds = model.predict(val_inputs)
```

[86]:
```python
val_preds
```

[86]:
```
array([ 2.86574459, 16.28920059, 15.87623286,  1.96882195])
```

[87]:
```python
val_targets
```

[87]:
```
0      1
17    18
15    16
1      2
Name: Rk, dtype: int64
```

[88]:
```python
val_rmse = mean_squared_error(val_preds, val_targets, squared=False)
print('The position is wrong of {} places'.format(val_rmse))
```

```
The position is wrong of 1.2672931673486385 places
```

[89]:
```python
val_rmse
```

[89]:
```
1.2672931673486385
```

2.0.3 Evaluating the Weights

[102]:
```python
weights = model.coef_
```

[110]:
```
weights_df = pd.DataFrame({'Parameters': train_inputs.columns,
'Weights': weights})
```

[115]:
```
weights_df.sort_values('Weights', ascending=False)
```

[115]:

	Parameters	Weights
2	L	1.557926
4	GA	1.452797
9	xGA	1.225432
1	D	0.602440
8	xG	-0.943708
3	GF	-1.103084
11	xGD/90	-1.173862
10	xGD	-1.176938
5	GD	-1.352136
0	W	-1.646702
12	Attendance	-1.660106
7	Pts/MP	-1.728109
6	Pts	-1.729754

These are the different weights' parameters.

3 Making Predictions

[90]:
```
def make_preds(user_input):
    input_df = pd.DataFrame([user_input])
    input_df[numerical_cols] = scaler.transform(input_df[numerical_cols])
    return model.predict(input_df[numerical_cols])
```

Let's see if it can properly guess in what position Roma finished its 2018/2019 season.

[97]:
```
user_input = {'W':18,
'D':12, 'L':8, 'GF':66, 'GA':48, 'GD': 18,
'Pts':66, 'Pts/MP':1.74, 'xG':64.3, 'xGA':54.4, 'xGD':9.9,
'xGD/90':0.26, 'Attendance': 38622}
```

[98]:
```
make_preds(user_input)
```

[98]:
```
array([7.38742012])
```

Quite, good, Roma final position was 6.