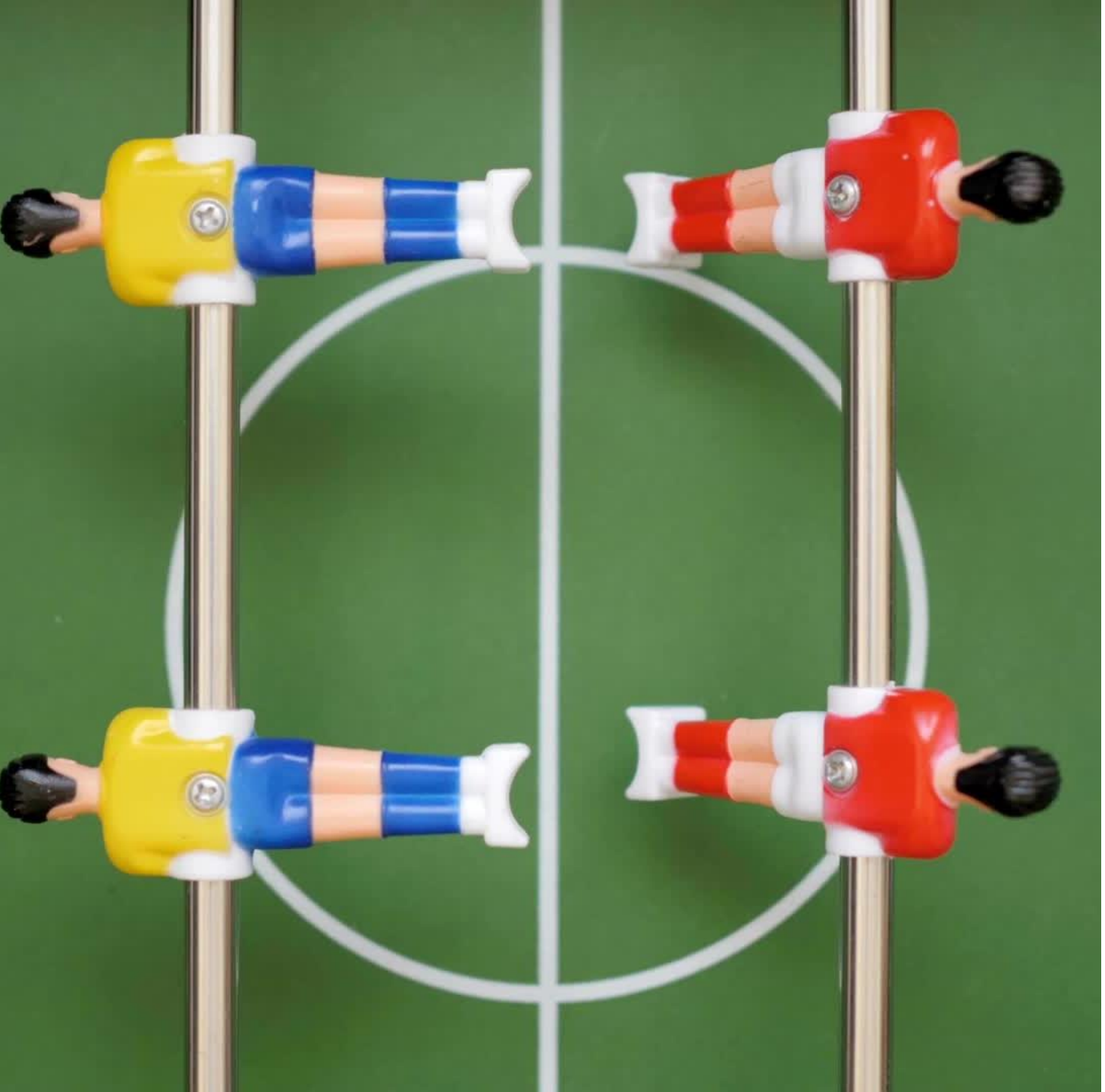


Análisis y Agrupamiento de Equipos de Fútbol

RASSHID

ORTIZ

RODRÍGUEZ

Objetivo

- El objetivo principal de este proyecto es aplicar técnicas de aprendizaje no supervisado, específicamente el algoritmo K-Means, para agrupar los 299 equipos del fútbol europeo en función de su rendimiento en diferentes aspectos del juego, esto para intentar hacer ligas más competitivas y así aumentar la popularidad y evitar que los ganadores sean los equipos de siempre. Además, buscamos identificar patrones emergentes y proporcionar una visión más completa de las dinámicas presentes en el fútbol.

Conjunto de datos

Utilizaremos el conjunto de datos "European Soccer Database“, el cual contiene información de los equipos contenidos en las 11 ligas europeas. Para nuestro estudio extraeremos características esenciales de cada equipo:

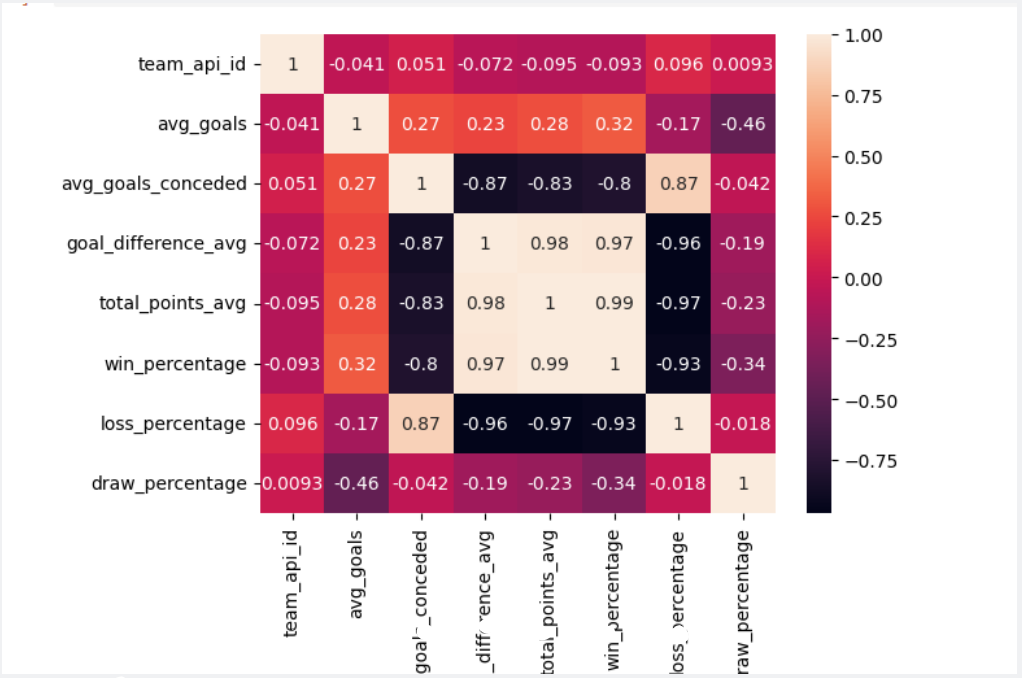
- Promedio de goles por partido
- Promedio de goles en contra por partido
- Promedio de la diferencia de goles por partido
- Promedio de los puntos obtenidos en sus respectivas ligas por partido
- Tasas de victorias
- Tasa de derrotas
- Tasa de empates

Esto porque las ligas en el fútbol toman todo esto en cuenta para determinar a los ganadores de sus respectivas ligas, así como los equipos que descienden por temporada. Consideramos el promedio porque no todos los equipos han jugado el mismo número de partidos.

```
# Visualizar las primeras filas del DataFrame
df_teams.head(20)
```

Out[24]:

	team_long_name	team_api_id	avg_goals	avg_goals_conceded	goal_difference_avg	total_points_avg	win_percentage	loss_percentage	draw_percentage
0	1. FC Kaiserslautern	8350	2.602941	1.544118	-0.485294	1.014706	0.250000	0.485294	0.264706
1	1. FC Köln	8722	2.632353	1.524510	-0.416667	1.147059	0.294118	0.441176	0.264706
2	1. FC Nürnberg	8165	2.717647	1.582353	-0.447059	1.117647	0.288235	0.458824	0.252941
3	1. FSV Mainz 05	9905	2.684874	1.340336	0.004202	1.382353	0.369748	0.357143	0.273109
4	AC Ajaccio	8576	2.631579	1.614035	-0.596491	0.929825	0.192982	0.456140	0.350877
5	AC Arles-Avignon	108893	2.394737	1.842105	-1.289474	0.526316	0.078947	0.631579	0.289474
6	AC Bellinzona	6493	3.203704	2.018519	-0.833333	0.925926	0.231481	0.537037	0.231481
7	ADO Den Haag	10217	3.066176	1.720588	-0.375000	1.143382	0.290441	0.437500	0.272059
8	AJ Auxerre	8583	2.171053	1.065789	0.039474	1.375000	0.348684	0.322368	0.328947
9	AS Monaco	9829	2.298246	1.039474	0.219298	1.578947	0.416667	0.254386	0.328947
10	AS Nancy-Lorraine	8481	2.405263	1.336842	-0.268421	1.163158	0.294737	0.426316	0.278947
11	AS Saint-Étienne	9853	2.292763	1.072368	0.148026	1.463816	0.398026	0.332237	0.269737
12	AZ	10229	3.088235	1.279412	0.529412	1.738971	0.514706	0.290441	0.194853
13	Aberdeen	8485	2.375000	1.174342	0.026316	1.430921	0.394737	0.358553	0.246711
14	Académica de Coimbra	10215	2.362903	1.395161	-0.427419	1.008065	0.225806	0.443548	0.330645
15	Ajax	8593	3.246324	0.867647	1.511029	2.213235	0.665441	0.117647	0.216912
16	Amadora	10213	2.133333	1.266667	-0.400000	1.133333	0.266667	0.400000	0.333333
17	Angers SCO	8121	2.052632	1.000000	0.052632	1.315789	0.342105	0.368421	0.289474



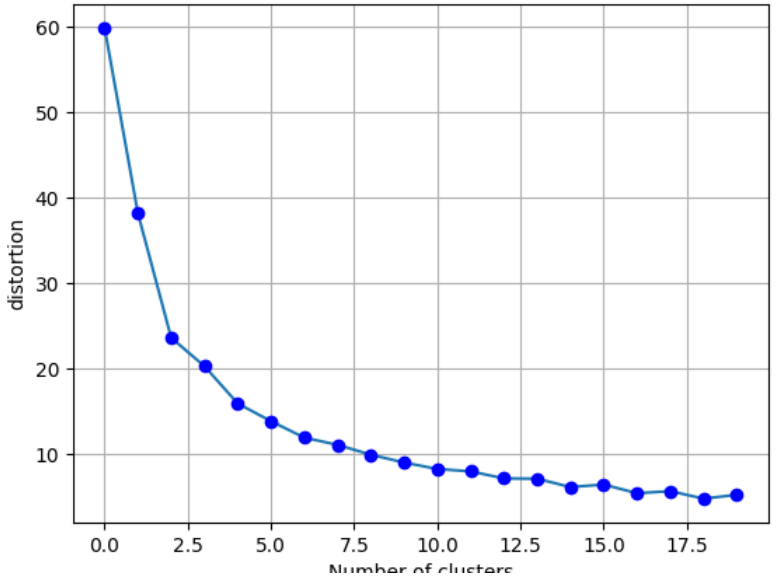
In [20]:
```
# Estadísticas descriptivas del data frame
print("\nEstadísticas descriptivas:")
print(df_teams.describe())
```

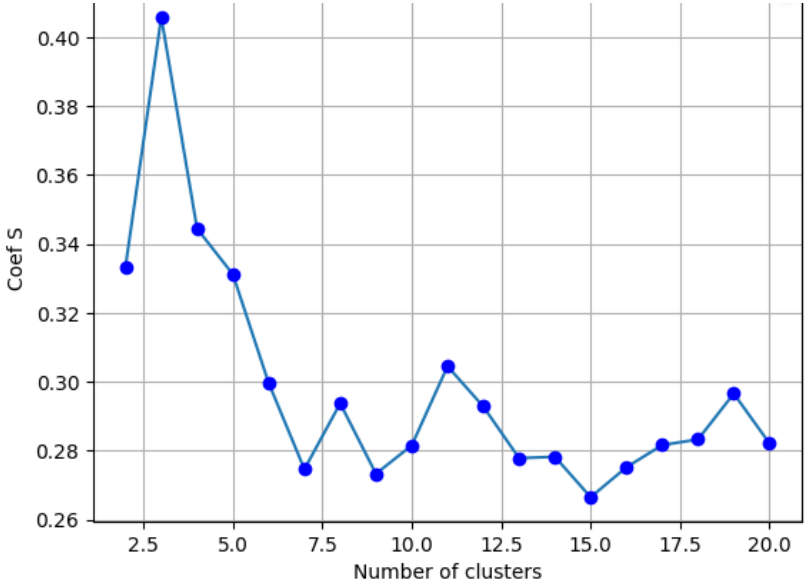
```
Estadísticas descriptivas:
         team_api_id   avg_goals  avg_goals_conceded  goal_difference_avg  \
count     299.000000  299.000000          299.000000           299.000000
mean    12340.521739    2.678256            1.444912            -0.211567
std     25940.411135    0.289769            0.290647             0.575348
min      1601.000000    2.013158            0.649194            -1.529412
25%      8349.000000    2.474342            1.272446            -0.581140
50%      8655.000000    2.639860            1.433333            -0.309211
75%      9886.500000    2.855263            1.631579             0.024187
max    274581.000000    3.773026            2.264706             2.029605

       total_points_avg  win_percentage  loss_percentage  draw_percentage
count        299.000000      299.000000       299.000000       299.000000
mean           1.231828        0.325681         0.419535         0.254784
std            0.368931        0.127333         0.119697         0.045579
min            0.526316        0.078947         0.088816         0.105263
25%            0.973684        0.236842         0.364384         0.233333
50%            1.159664        0.296703         0.431818         0.253333
75%            1.395482        0.375465         0.500000         0.280976
max            2.450658        0.769737         0.736842         0.470588
```

Análisis de los datos

Construcción del modelo

- Primeramente utilizaremos los métodos del codo y la silueta para analizar el óptimo de los clusters a formar y ver si es factible utilizar ese valor para aplicar al K-means que construimos con el criterio de paro para los centroides.
- Después de hacer las predicciones necesitaremos también una forma de mapear las etiquetas numéricas a sus respectivas ligas predichas.





Método del codo y silueta

In [48]:
```
# construir modelo
y_pred=ML.k_means_Criterio_1(X,11)
y_pred
```

```
iteraciones 23
```

Out[48]:
```
array([ 8, 10,  3, 10,  8,  8,  7,  9,  1,  1,  4,  1,  2,  1,  4,  0,  6,
        1,  6,  5, 10,  4, 10,  5,  2,  4,  5,  3,  4,  3,  7,  6,  4,  3,
        0,  2,  9,  6,  9,  4, 10,  9,  7,  4, 10, 10,  9,  8,  4,  0,  8,
        5,  6,  0, 10,  4,  8,  8,  9,  8,  3,  2,  7,  9,  8,  3,  6,  4,
        4, 10,  5,  7,  9,  4, 10,  0,  0,  0,  7,  8,  2,  6,  8, 10,  2,
        8,  6, 10,  9,  0,  5, 10,  4,  3,  7, 10,  5,  2,  9,  7,  2,  9,
        6,  8,  5,  5,  9,  9, 10,  8,  1, 10, 10,  8,  1,  9,  8,  2,  6,
        4,  3,  4,  9,  4,  9,  7, 10, 10,  8,  8,  5, 10,  4,  5,  5,  8,
        5,  3,  9,  4,  2,  2,  3,  8, 10,  4,  1, 10,  8,  8,  3,  1,  4,
        5,  5,  6, 10,  8,  5,  9,  7,  5,  5,  6,  5,  1,  8,  2, 10,  3,
        3,  5,  6,  3,  3,  3,  9,  4,  6,  5,  1,  7,  1,  2,  0,  3,  5,
       10, 10,  7,  4, 10,  4,  4,  4,  8,  8,  9,  7, 10, 10,  4,  8, 10,
       10,  9,  5, 10,  0,  7,  7,  3,  0,  2,  3,  3,  8,  8,  6,  7,  5,
        3, 10,  9,  4,  6,  4,  4,  5,  9,  3,  7,  9,  8,  0, 10,  3,  7,
       10, 10, 10,  3,  2,  4,  8, 10,  8,  7,  1, 10, 10,  4,  8,  6,  1,
       10,  5,  4, 10, 10,  2, 10,  8, 10,  5,  1,  6,  7,  3, 10,  8, 10,
        6,  8,  7,  5,  4,  2,  9,  2, 10,  2,  4,  8,  9,  4,  3, 10,  6,
        8,  3,  7,  1,  9,  9,  4,  3, 10,  4])
```

Predicciones

APLICAMOS EL ALGORITMO DE APRENDIZAJE NO SUPERVISADO DE K-MEANS PARA OBTENER LAS PREDICCIONES.

In [ ]:
```
#diccionario de las ligas
mapeo_ligas = {0: 'Belgium Jupiler League', 1: 'England Premier League', 2: 'France Ligue 1',
               3: "Germany 1. Bundesliga",4: "Italy Serie A",5: "Netherlands Eredivisie", 6:"Poland Ekstraklasa", 7:"Portugal Liga ZON Sagres",
               8:"Scotland Premier League",9:"Spain LIGA BBVA", 10: "Switzerland Super League"}
# Decodificación de números de etiqueta a nombres de ligas del vector de etiquetas original
nombres_ligas = [mapeo_ligas[num] for num in etiquetas]
nombres_ligas
```

In [ ]:
```
#Ligas para las etiquetas predichas
nombres_ligas_new = [mapeo_ligas[num] for num in y_pred]
nombres_ligas_new
```

In [51]:
```
#Mostrar predicciones
df_teams['liga_predicha'] = nombres_ligas_new
df_teams.head(20)
```

Out[51]:

	team_long_name	team_api_id	avg_goals	avg_goals_conceded	goal_difference_avg	total_points_avg	win_percentage	loss_percentage	draw_percentage	liga_predicha
0	1. FC Kaiserslautern	8350	2.602941	1.544118	-0.485294	1.014706	0.250000	0.485294	0.264706	Scotland Premier League
1	1. FC Köln	8722	2.632353	1.524510	-0.416667	1.147059	0.294118	0.441176	0.264706	Switzerland Super League
2	1. FC Nürnberg	8165	2.717647	1.582353	-0.441059	1.176471	0.288235	0.459824	0.252941	Germany 1. Bundesliga

Mapeo de las etiquetas

Evaluación del modelo

- Para medir los resultados lo que haremos será un análisis estadístico de algunos grupos, específicamente las estadísticas del mejor grupo vs las del peor grupo.
- Para ello buscaremos los grupos donde está el equipo con mayor porcentaje de victoria y el equipo con el peor porcentaje de victoria.

In [53]:
```
#Grupo donde está el equipo con mayor porcentaje de victorias.
index=df_teams["win_percentage"].argmax()
team_max_win_rate=df_teams.iloc[index]
print("Mostrar estadísticas del equipo con más win rate \n", team_max_win_rate)
```

```
Mostrar estadísticas del equipo con más win rate
 team_long_name                  FC Barcelona
team_api_id                            8634
avg_goals                          3.555921
avg_goals_conceded                 0.763158
goal_difference_avg                2.029605
total_points_avg                   2.450658
win_percentage                     0.769737
loss_percentage                    0.088816
draw_percentage                    0.141447
liga_predicha          Belgium Jupiler League
Name: 75, dtype: object
```

In [54]:
```
#Grupo donde está el equipo con menor porcentaje de victorias.
index=df_teams["win_percentage"].argmin()
team_min_win_rate=df_teams.iloc[index]
print("Mostrar estadísticas del equipo con más win rate \n", team_min_win_rate)
```

```
Mostrar estadísticas del equipo con más win rate
 team_long_name               AC Arles-Avignon
team_api_id                            108893
avg_goals                          2.394737
avg_goals_conceded                 1.842105
goal_difference_avg               -1.289474
total_points_avg                   0.526316
win_percentage                     0.078947
loss_percentage                    0.631579
draw_percentage                    0.289474
liga_predicha          Scotland Premier League
Name: 5, dtype: object
```

Mejor grupo vs peor grupo

In [55]:
```
#Data frame de los equipos con mejor win_rate
df_group_max_win=df_teams.loc[df_teams.loc[:,"liga_predicha"]== "Belgium Jupiler League"]
df_group_max_win.head(10)
```

Out[55]:

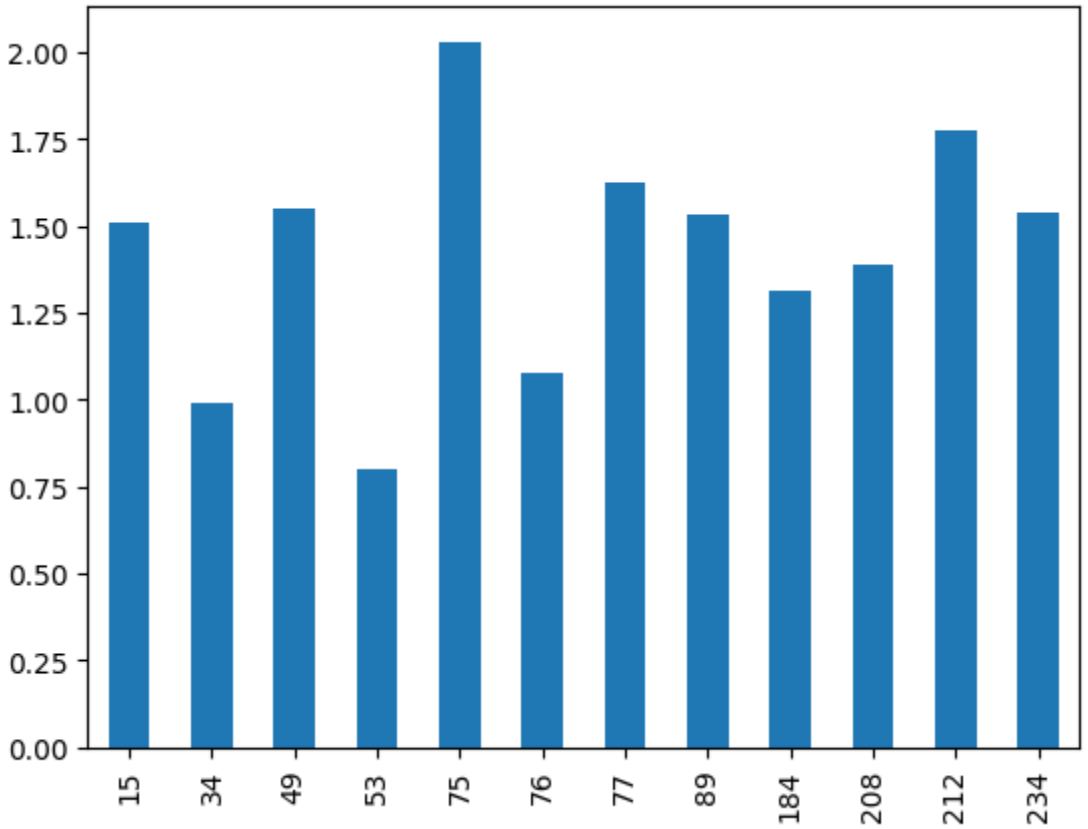
	team_long_name	team_api_id	avg_goals	avg_goals_conceded	goal_difference_avg	total_points_avg	win_percentage	loss_percentage	draw_percentage	liga_predicha
15	Ajax	8593	3.246324	0.867647	1.511029	2.213235	0.665441	0.117647	0.216912	Belgium Jupiler League
34	Borussia Dortmund	9789	3.062500	1.036765	0.988971	1.959559	0.577206	0.194853	0.227941	Belgium Jupiler League
49	Celtic	9925	3.019737	0.733553	1.552632	2.315789	0.717105	0.118421	0.164474	Belgium Jupiler League
53	Club Brugge KV	8342	3.169811	1.183962	0.801887	1.929245	0.580189	0.231132	0.188679	Belgium Jupiler League
75	FC Barcelona	8634	3.555921	0.763158	2.029605	2.450658	0.769737	0.088816	0.141447	Belgium Jupiler League
76	FC Basel	9931	3.251748	1.087413	1.076923	2.111888	0.629371	0.146853	0.223776	Belgium Jupiler League
77	FC Bayern Munich	9823	3.176471	0.775735	1.625000	2.290441	0.709559	0.128676	0.161765	Belgium Jupiler League

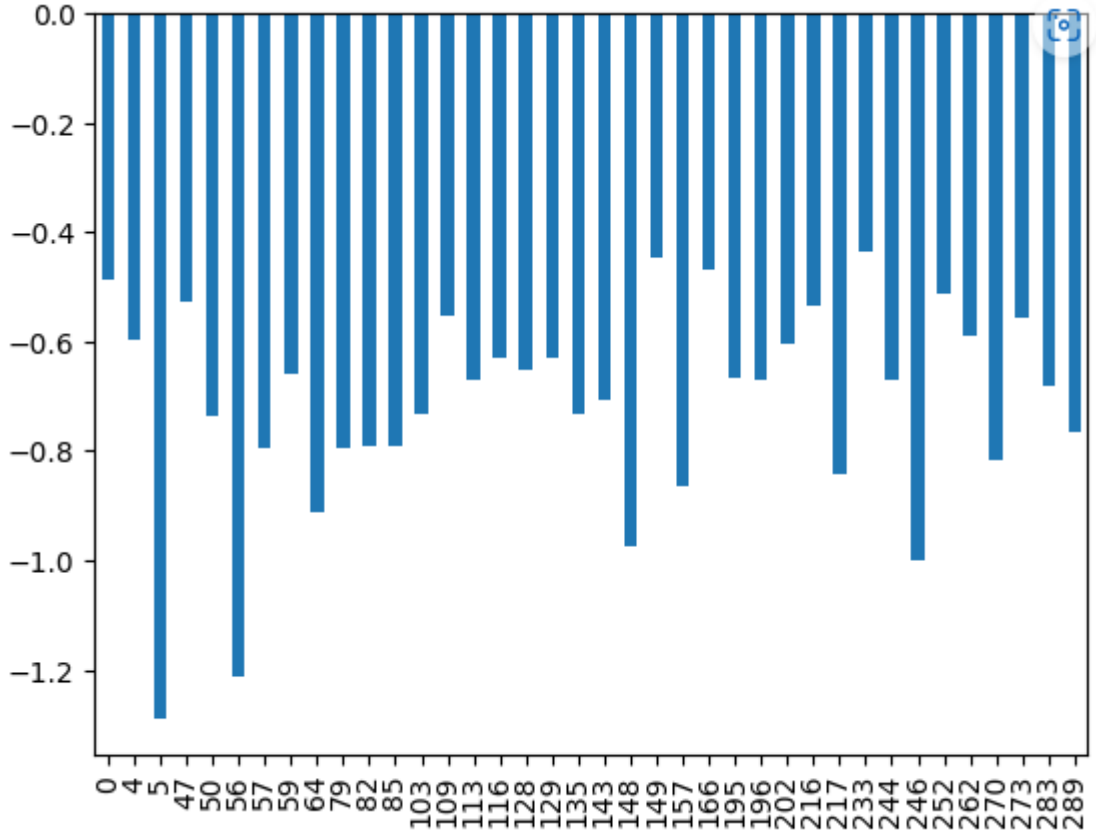
In [57]:
```
#Data frame de los equipos con mejor win_rate
df_group_min_win=df_teams.loc[df_teams.loc[:,"liga_predicha"]== "Scotland Premier League"]
df_group_min_win.head(10)
```

Out[57]:

	team_long_name	team_api_id	avg_goals	avg_goals_conceded	goal_difference_avg	total_points_avg	win_percentage	loss_percentage	draw_percentage	liga_predicha
0	1. FC Kaiserslautern	8350	2.602941	1.544118	-0.485294	1.014706	0.250000	0.485294	0.264706	Scotland Premier League
4	AC Ajaccio	8576	2.631579	1.614035	-0.596491	0.929825	0.192982	0.456140	0.350877	Scotland Premier League
5	AC Arles-Avignon	108893	2.394737	1.842105	-1.289474	0.526316	0.078947	0.631579	0.289474	Scotland Premier League
47	Carpi	208931	2.473684	1.500000	-0.526316	1.000000	0.236842	0.473684	0.289474	Scotland Premier League
50	Cesena	9880	2.518182	1.627273	-0.736364	0.790909	0.172727	0.554545	0.272727	Scotland Premier League
56	Córdoba CF	7869	2.368421	1.789474	-1.210526	0.526316	0.078947	0.631579	0.289474	Scotland Premier League
57	DSC Arminia Bielefeld	9912	2.500000	1.647059	-0.794118	0.823529	0.117647	0.411765	0.470588	Scotland Premier League

Promedio de goles de diferencia





Conclusiones

Sobre los datos: Nos dimos cuenta de que varias características guardaban una alta correlación por lo que las eliminamos, también que el agrupamiento ideal según nuestro modelo no era formar 11 grupos (en este caso ligas) sin embargo para efectos prácticos mantuvimos el mismo número de grupos.

Sobre la mejora del modelo: Extender y actualizar los datos; agregar otras características importantes; mejorar el agrupamiento de equipos; actualizar los resultados del algoritmo por año o temporada en el fútbol; mejorar el análisis estadístico de los resultados.

Bibliografía

Mathien, H. (2016). European soccer database [Data set].

(S/f). Tibco.com. Recuperado el 2 de diciembre de 2023, de https://www.tibco.com/es/reference-center/what-is-unsupervised-learning

Ortega, C. (2022, febrero 21). Análisis estadístico: Qué es, usos y cómo realizarlo. QuestionPro. https://www.questionpro.com/blog/es/analisis-estadistico/