

MASTER EN BIG DATA APLICADO AL SCOUTING EN FÚTBOL

Módulo 11. Trabajo de Fin de Máster



UCAM
UNIVERSIDAD
CATÓLICA DE MURCIA

ÍNDICE

0. Objetivo	3
1. Introducción	3
2. Extracción y tratamiento inicial	4
2.0. Cargar librerías	4
2.1. Extraer Datasets	4
2.2. Limpiar datos	5
2.3. Unión de los datasets	9
2.4. Resultado	10
3. Procesamiento de los datos	11
3.0. Cargar librerías	11
3.1. Cargar datasets	11
3.2. Procesamiento de los datos	11
3.3. Guardado de tablas	13
3.4. Resultado	13
4. Creación de los gráficos y despliegue en App	14
4.0. Cargar librerías	14
4.1. Cargar datasets	14
4.2. Definición de los input de la app	15
4.3. Creación de los gráficos	16
4.3.1 Gráfico de máximo y mínimos	16
4.3.2 Algoritmo y gráfico de similitud	19
4.4. Definición de los widgets de la app	20
4.5. Resultado	21
5. Puesta en producción	21
5.1 Resultado	22
6. Ejemplos de uso	23
6.1 Ejemplo de uso 1. Director Deportivo del Girona	23
6.2 Ejemplo de uso 2. Analista del Arsenal.	26

0. Objetivo

En el presente estudio se trata de comparar cualquier equipo de las 5 grandes ligas europeas en la temporada actual, con los datos de todas las variables recogidas por FBREF de los equipos campeones de estas competiciones desde la temporada 2017-2018. Se busca que el output final sea una página web interactiva en la que el usuario elija distintos parámetros y pueda de forma muy sencilla ver e incluso descargar dichas gráficas comparativas.

1. Introducción

Para crear un dashboard interactivo donde el usuario pueda comparar al equipo que quiera (de la temporada 2023-2024 y de las 5 grandes ligas europeas) con los campeones de estas ligas desde la temporada 2017-2018 se ha dividido el proceso en 3 partes, todas ellas realizadas usando Python:

1. **Extracción y tratamiento inicial de las tablas originales:** Se hace un scrapeo de las tablas de FBREF. Se renombran las variables para que al unir las diferentes tablas de este proveedor no haya nombres duplicados que induzcan a error, así como para que los nombres sean autoexplicativos. Se guarda toda la información en una tabla maestra.
2. **Tratamiento de los datos enfocados al problema a resolver y creación de las gráficas.** Estudio de los datos y tratamiento de los mismos para no incurrir en errores estadísticos. Representación de las gráficas y creación del algoritmo de similitud.
3. **Creación del dashboard que las contiene.** Automatización del proceso y creación del dashboard interactivo mediante streamlit.

Posteriormente se ha puesto en **producción** la **aplicación web** para que pueda ser usada por cualquiera que tenga acceso a internet.

Por último se añaden dos **ejemplos** de utilización de la misma tanto desde el punto de vista del Scouting como desde el punto de vista de un miembro del cuerpo técnico.

2. Extracción y tratamiento inicial

- Este código es semiautomático y no automático ya que la propia página de FBREF capta la posibilidad de scrapear todas las tablas en una sola ejecución. Es por esto que se debe ejecutar el código una vez por cada liga.
- Posteriormente el código las junta en una tabla maestra que será usada como input en el resto del proceso.

2.0. Cargar librerías

Este bloque carga las bibliotecas necesarias para el proyecto. En este caso, se utiliza Pandas para manipular y analizar datos tabulares.

0. Cargar librerías

```
[ ] import pandas as pd
    pd.set_option('display.max_columns', None)
```

2.1. Extraer Datasets

- Este bloque define una serie de URLs de FBREF, donde se extraen los datos relacionados con diferentes temporadas de las ligas de fútbol. Se tienen URLs para temporadas desde 2017-2018 hasta la temporada actual (2023-2024). Luego, se itera sobre estas URLs, se leen las tablas HTML y se almacenan en un diccionario llamado `tables_ligas`, utilizando nombres de clave que identifican tanto la temporada como el tipo de datos extraídos.

1. Extraer Datasets

```
URL = {
    'LALIGA_2023_2024': 'https://fbref.com/en/comps/12/La-Liga-Stats',
    'LALIGA_2022_2023': 'https://fbref.com/en/comps/12/2022-2023/2022-2023-La-Liga-Stats',
    'LALIGA_2021_2022': 'https://fbref.com/en/comps/12/2021-2022/2021-2022-La-Liga-Stats',
    'LALIGA_2020_2021': 'https://fbref.com/en/comps/12/2020-2021/2020-2021-La-Liga-Stats',
    'LALIGA_2019_2020': 'https://fbref.com/en/comps/12/2019-2020/2019-2020-La-Liga-Stats',
    'LALIGA_2018_2019': 'https://fbref.com/en/comps/12/2018-2019/2018-2019-La-Liga-Stats',
    'LALIGA_2017_2018': 'https://fbref.com/en/comps/12/2017-2018/2017-2018-La-Liga-Stats',
    'PREMIER_2023_2024': 'https://fbref.com/en/comps/9/Premier-League-Stats',
    ...
}
```

```
[ ] names_list = pd.Series(URL).index.tolist()
url_list = pd.Series(URL).tolist()
lista_datasets = ['Clasificacion_Home_Away', 'Standard_Stats', 'Opponent_Standard_Stats',
                  'Squad_Advance_Goalkeeping', 'Squad_Advance_Goalkeeping_Opponent_Stats',
                  'Squad_Goalkeeping', 'Squad_Goalkeeping_Opponent_Stats', 'Shooting_Stats', 'Shooting_Opponent_Stats',
                  'Passing_Stats', 'Passing_Opponent_Stats', 'Pass_Types_Opponent_Stats', 'Goal_Shot_Creation_Stats',
                  'Goal_Shot_Creation_Opponent_Stats', 'Defensive_Stats', 'Defensive_Opponent_Stats',
                  'Possession_Stats', 'Possession_Opponent_Stats', 'Playing_Time_Stats', 'Playing_Time_Opponent_Stats',
                  'Miscellaneous_Stats', 'Miscellaneous_Opponent_Stats']

tables_ligas = {}

# Iterar sobre las listas
for name, url in zip(names_list, url_list):
    # Leer las tablas desde la URL
    tables_0 = pd.read_html(url, header=0)
    tables = pd.read_html(url, header=1)
    tables_ligas['Clasificacion' + '_' + name] = tables_0[0]

    for nombre_df, tabla in zip(lista_datasets, tables[1:]):
        # Asignar nombres a las tablas en el diccionario
        tables_ligas[nombre_df + '_' + name] = tabla

[ ] print("Claves del diccionario:", tables_ligas.keys())

Claves del diccionario: dict_keys(['Clasificacion_LALIGA_2023_2024', 'Clasificacion_Home_Away_LALIGA_2023_2024', 'Standard_Stats_LALIGA_2023_2024', 'Opponent_Standard_Stats_LALIGA_2023_2024',
...
```

2.2. Limpiar datos

Este bloque se encarga de limpiar los datos recién extraídos y hacer algunos ajustes.

- Se agrega la temporada correspondiente a cada DataFrame del diccionario `tables_ligas`.

2. Clean Data

```
[ ] tables_ligas.keys()

[ ] # Creo la variable temporada
for key, df in tables_ligas.items():
    last_9_chars = key[-9:]
    df = df.assign(Temporada=last_9_chars)
    tables_ligas[key] = df

[ ] tables_ligas = {key: value for key, value in tables_ligas.items() if 'Playing_Time' not in key}
```

- Se eliminan ciertas columnas de los DataFrames según un patrón. Por ejemplo, elimina columnas como 'Attendance', 'Top Team Scorer', 'Goalkeeper' y 'Notes' de los DataFrames de clasificación (cuando aplicable), ya que estas columnas no son relevantes para el análisis.
- Se renombran y eliminan columnas específicas de ciertos DataFrames para que los nombres de las columnas sean más descriptivos y coherentes, y para eliminar redundancias.

- Se realizan ajustes similares para varios otros tipos de datos, como estadísticas de

```
for df_name in tables_ligas:
    if 'Playing_Time' in df_name:
        tables_ligas[df_name].drop(columns=['Attendance', 'Top Team Scorer', 'Goalkeeper', 'Notes'], inplace=True)
    elif 'Classification_L' in df_name:
        print(df_name)
        tables_ligas[df_name].drop(columns=['Attendance', 'Top Team Scorer', 'Goalkeeper', 'Notes'], inplace=True)
    elif 'Last_5' in tables_ligas[df_name].columns:
        tables_ligas[df_name].drop(columns=['Last_5'], inplace=True)
    elif 'Classification_Home_Away_' in df_name:
        print(df_name)
        tables_ligas[df_name].drop(columns=['Temporada'], inplace=True)
        tables_ligas[df_name].rename(columns={
            "Rk": "Rk_Home",
            "HP": "Partidos_Home",
            "W": "W_Home",
            "D": "D_Home",
            "L": "L_Home",
            "GF": "GF_Home",
            "GA": "GA_Home",
            "GD": "GD_Home",
            "Pts": "Pts_Home",
            "Pts/HP": "Pts_Partido_Home",
            "xG": "xG_Home",
            "xGA": "xGA_Home",
            "xGD": "xGD_Home",
            "xGD/90": "xGD/90_Home",
            "HP_1": "Partidos_Away",
            "W_1": "W_Away",
            "D_1": "D_Away",
            "L_1": "L_Away",
            "GF_1": "GF_Away",
            "GA_1": "GA_Away",
            "GD_1": "GD_Away",
            "Pts_1": "Pts_Away",
            "Pts/HP_1": "Pts_Partido_Away",
            "xG_1": "xG_Away",
            "xGA_1": "xGA_Away",
            "xGD_1": "xGD_Away",
            "xGD/90_1": "xGD/90_Away"
        }, inplace=True)
    elif 'Standard_Stats_' in df_name:
        print(df_name)
        tables_ligas[df_name].drop(columns=['HP', 'Starts', 'Win', 'QoC', 'Gls', 'xG', 'xGA', 'xGD', 'Temporada'], inplace=True)
```

porteros, estadísticas de disparos, estadísticas de pases, etc.

- Se tienen varios conjuntos de datos, cada uno representando una liga de fútbol y una temporada específica. Estos conjuntos de datos se almacenan en variables como DF_LALIGA, DF_PREMIER, DF_SERIEA, DF_BUNDESLIGA, DF_LIGUE1, que contienen información para las ligas respectivas y para diferentes temporadas.
- Cada DataFrame contiene estadísticas y métricas de diferentes equipos durante una temporada específica de la liga.
- Se agregan columnas adicionales a cada DataFrame para indicar la competición (liga) a la que pertenecen.
- Se tratan algunos nombres de columnas que habían quedado confusos.

✓ Juntar los Datasets

```
# Crear un diccionario para almacenar los DataFrames fusionados con sufijo
result_dict = {}

# DISTINGO LAS TABLAS POR LIGA Y AÑO
for liga_anio in names_list:
    tables_Prop = {}
    for df_name in tables_ligas.keys():
        if (liga_anio in df_name) and ('Opponent_' not in df_name):
            tables_Prop[df_name] = tables_ligas[df_name]
    tables_Opp = {}
    for df_name in tables_ligas.keys():
        if (liga_anio in df_name) and ('Opponent_' in df_name):
            tables_Opp[df_name] = tables_ligas[df_name]

    merged_df = None

    # Iterar sobre las claves del diccionario
    for df_name in tables_Prop.keys():
        df = tables_Prop[df_name]

        # Fusionar el DataFrame actual con el DataFrame acumulado
        if merged_df is None:
            merged_df = df
        else:
            merged_df = merged_df.merge(df, on='Squad', how='inner')

    # Agregar el DataFrame fusionado a la lista
    Df_Propia = merged_df.copy()

    merged_df = None

    # Iterar sobre las claves del diccionario
    for df_name in tables_Opp.keys():
        df = tables_Opp[df_name]

        # Fusionar el DataFrame actual con el DataFrame acumulado
        if merged_df is None:
            merged_df = df
        else:
            merged_df = merged_df.merge(df, on='Squad', how='inner')

    # Agregar el DataFrame fusionado a la lista
    Df_Opponent = merged_df.copy()

    Df_Opponent['Squad'] = Df_Opponent['Squad'].str.replace('vs ', '')
    Df_Season_Total = Df_Propia.merge(Df_Opponent, how='inner', on='Squad')

    # Agregar el DataFrame fusionado al diccionario con sufijo
    result_dict[liga_anio] = Df_Season_Total
```

```
[ ] DF_FINAL = pd.DataFrame()
for df_name in names_list:
    if df_name in result_dict:
        DF_FINAL = pd.concat([DF_FINAL, result_dict[df_name]], axis=0)
    else:
        print(f"DataFrame {df_name} not found in result_dict.")

# Reset index if needed
DF_FINAL.reset_index(drop=True, inplace=True)
```

DF_FINAL.shape

```
DF_FINAL.rename(columns={
    'Jugadores_Involucrados_x': 'Jugadores_Involucrados',
    'Age_media_x': 'Age_media_equipo_propio',
    'Posesion_media_x': 'Posesion_media_a_favor',
    'Ast_x': 'Ast_a_favor',
    'G+A_x': 'G+A_a_favor',
    'Goles_Sin_Penaltis_x': 'Goles_Sin_Penaltis_a_favor',
    'Penaltis_Intentados_x': 'Penaltis_Intentados_a_favor',
    'Tarjetas_Amarillas_x': 'Tarjetas_Amarillas_propias',
    'Tarjetas_Rojas_x': 'Tarjetas_Rojas_propias',
    'xAst_x': 'xAst_a_favor',
    'npxG+xAG_x': 'npxG+xAG_a_favor',
    'Prog_con_Conduccion_x': 'Prog_con_Conduccion_a_favor',
    'Prog_con_Pase_x': 'Prog_con_Pase_a_favor',
    'Gls/90_x': 'Gls/90_a_favor',
    'Ast/90_x': 'Ast/90_a_favor',
    'G+A.1_x': 'G+A.1_a_favor',
    'G-PK/90_x': 'G-PK/90_a_favor',
    'xG/90_x': 'xG/90_a_favor',
    'xAst/90_x': 'xAst/90_a_favor',
    'xG+xAG/90_x': 'xG+xAG/90_a_favor',
    'npxG/90_x': 'npxG/90_a_favor',
    'npxG+xAst/90_x': 'npxG+xAst/90_a_favor',
    'CS_x': 'CS_a_favor',
    'CS%_x': 'CS%_a_favor',
    'Thr_x': 'Centros_Interceptados_a_favor',
    'G-xG_x': 'G-xG_a_favor',
    'A-xAG_x': 'A-xAG_a_favor',
    'Pases_Bloqueados_en_contra_x': 'Pases_Bloqueados_por_rival',
    'Tackle_a_favor_x': 'Tackle_a_favor',
    'Tacle_Ganado_a_favor_x': 'Tackle_Ganado_a_favor',
    'Tackle_1_Tercio_a_favor_x': 'Tackle_1_Tercio_a_favor',
    'Tackle_2_Tercio_a_favor_x': 'Tackle_2_Tercio_a_favor',
    'Tackle_3_Tercio_a_favor_x': 'Tackle_3_Tercio_a_favor',
    'Tackle_Regate_a_favor_x': 'Tackle_Regate_a_favor',
    'Tackle_Regate_Intentados_a_favor_x': 'Tackle_Regate_Intentados_a_favor',
    'Porc_Tackle_Ganado_a_favor_x': 'Porc_Tackle_Ganado_a_favor',
    'Bloqueos_a_favor_x': 'Bloqueos_a_favor',
    'Tiros_Bloqueos_a_favor_x': 'Tiros_Bloqueos_a_favor',
    'Pases_Bloqueos_a_favor_x': 'Pases_Bloqueos_a_favor',
    'Intercepciones_a_favor_x': 'Intercepciones_a_favor',
    'Tackles_Intercepciones_a_favor_x': 'Tackles_Intercepciones_a_favor',
    'Despejes_a_favor_x': 'Despejes_a_favor',
    'Errores_equipo_propio_x': 'Errores_equipo_propio',
    'Tarjetas_Amarillas_equipo_propio_x': 'Eliminar_1',
    'Tarjetas_Rojas_equipo_propio_x': 'Eliminar_2',
    '2_Tarjetas_Amarillas_equipo_propio_x': '2_Tarjetas_Amarillas_equipo_propio',
    'Faltas_cometidas_equipo_propio_x': 'Faltas_cometidas_equipo_propio',
    'Fueras_Juego_equipo_propio_x': 'Fueras_Juego_equipo_propio',
    'Duelos_Aereos_ganados_equipo_propio_x': 'Duelos_Aereos_ganados_equipo_propio',
    'Duelos_Aereos_perdidos_equipo_propio_x': 'Duelos_Aereos_perdidos_equipo_propio',
    'Porc_Duelos_Aereos_ganados_equipo_propio_x': 'Porc_Duelos_Aereos_ganados_equipo_propio'
}, inplace=True)

DF_FINAL.drop(columns=['Eliminar_1', 'Eliminar_2'], inplace=True)
```

- Se agrega la columna que indica la temporada a la que hace referencia cada registro como primera columna de la tabla.
- Se guarda la tabla de la cada competición como punto intermedio.


```
[ ] col_Temporada = DF_FINAL.iloc[:, 15]
DF_FINAL.drop(DF_FINAL.columns[15], axis=1, inplace=True)
DF_FINAL.insert(0, 'Temporada', col_Temporada)
```

```
[ ] DF_FINAL.head()
```

	Temporada	Rk	Squad	MP	W	D	L	GF	GA	GD	Pts	Pts/MP	xG	xGA	xGD	xGD/90	Rk_Home	Partidos_Home	W_Home	D_Home	L_Home	GF_Home	GA_Home	G
0	2023_2024	1	Real Madrid	28	21	6	1	80	18	42	69	2.46	52.2	26.2	26.0	0.93	1	14	12	2	0	35	7	
1	2023_2024	2	Girona	28	19	5	4	59	33	26	62	2.21	47.8	38.0	9.8	0.35	2	14	11	2	1	35	14	
2	2023_2024	3	Barcelona	28	18	7	3	57	34	23	61	2.18	61.0	29.0	32.0	1.14	3	15	11	1	3	33	19	
3	2023_2024	4	Atlético Madrid	28	17	4	7	54	31	23	55	1.96	49.2	29.6	19.6	0.70	4	14	13	1	0	34	13	
4	2023_2024	5	Athletic Club	28	15	8	5	48	26	22	53	1.89	42.8	27.7	15.1	0.54	5	14	10	3	1	34	14	
5																								

```
[ ] DF_FINAL.to_csv('/content/drive/MyDrive/Master Analisis de datos futbol/Master/Tablas_FBREF/LALIGA_FBREF_201617_202324_J28.csv', index=False)
```

2.3. Unión de los datasets

- A continuación se utilizan las funciones `pd.concat()` y `ignore_index=True` para concatenar los DataFrames de las diferentes ligas en uno solo llamado `GRANDES_5_LIGAS_EUR_FBREF_201617_202324_J28`. Esto crea un DataFrame que contiene toda la información de las cinco principales ligas de fútbol europeas.

3. Junto los datasets por paises

```
[ ] DF_LALIGA = pd.read_csv('/content/drive/MyDrive/Master Analisis de datos futbol/Master/Tablas_FBREF/LALIGA_FBREF_201617_202324_J28.csv', sep=',', index_col=False)
DF_PREMIER = pd.read_csv('/content/drive/MyDrive/Master Analisis de datos futbol/Master/Tablas_FBREF/PREMIER_FBREF_201617_202324_J28.csv', sep=',', index_col=False)
DF_SERIEA = pd.read_csv('/content/drive/MyDrive/Master Analisis de datos futbol/Master/Tablas_FBREF/SERIEA_FBREF_201617_202324_J28.csv', sep=',', index_col=False)
DF_BUNDESLIGA = pd.read_csv('/content/drive/MyDrive/Master Analisis de datos futbol/Master/Tablas_FBREF/BUNDESLIGA_FBREF_201617_202324_J28.csv', sep=',', index_col=False)
DF_LIGUE1 = pd.read_csv('/content/drive/MyDrive/Master Analisis de datos futbol/Master/Tablas_FBREF/LIGUE1_FBREF_201617_202324_J28.csv', sep=',', index_col=False)
```

```
[ ] DF_LALIGA['COMPETITION'] = 'LALIGA'
DF_PREMIER['COMPETITION'] = 'PREMIER'
DF_SERIEA['COMPETITION'] = 'SERIEA'
DF_BUNDESLIGA['COMPETITION'] = 'BUNDESLIGA'
DF_LIGUE1['COMPETITION'] = 'LIGUE1'
```

```
[ ] col_COMPETITION = DF_LALIGA.iloc[:, 353]
DF_LALIGA.drop(DF_LALIGA.columns[353], axis=1, inplace=True)
DF_LALIGA.insert(0, 'Competition', col_COMPETITION)

col_COMPETITION = DF_PREMIER.iloc[:, 353]
DF_PREMIER.drop(DF_PREMIER.columns[353], axis=1, inplace=True)
DF_PREMIER.insert(0, 'Competition', col_COMPETITION)

col_COMPETITION = DF_SERIEA.iloc[:, 353]
DF_SERIEA.drop(DF_SERIEA.columns[353], axis=1, inplace=True)
DF_SERIEA.insert(0, 'Competition', col_COMPETITION)

col_COMPETITION = DF_BUNDESLIGA.iloc[:, 353]
DF_BUNDESLIGA.drop(DF_BUNDESLIGA.columns[353], axis=1, inplace=True)
DF_BUNDESLIGA.insert(0, 'Competition', col_COMPETITION)

col_COMPETITION = DF_LIGUE1.iloc[:, 353]
DF_LIGUE1.drop(DF_LIGUE1.columns[353], axis=1, inplace=True)
DF_LIGUE1.insert(0, 'Competition', col_COMPETITION)
```

```
[ ] GRANDES_5_LIGAS_EUR_FBREF_201617_202324_J28 = pd.concat([DF_LALIGA, DF_PREMIER, DF_SERIEA, DF_BUNDESLIGA, DF_LIGUE1], ignore_index=True)
```

```
[ ] GRANDES_5_LIGAS_EUR_FBREF_201617_202324_J28.to_csv('/content/drive/MyDrive/Master Analisis de datos futbol/Master/Tablas_FBREF/GRANDES_5_LIGAS_EUR_FBREF_201617_202324_J28.csv', sep=',', index=False)
```

- Se utiliza la función `to_csv()` para guardar el DataFrame combinado en un archivo CSV en una ubicación específica. Esto permite almacenar los datos fusionados para su posterior análisis o uso. Este DataFrame será el resultado final de las ejecuciones de este código.

```
[ ] GRANDES_5_LIGAS_EUR_FBREF_201617_202324_J28.to_csv('/content/drive/MyDrive/Master Analisis de datos futbol/Master/Tablas_FBREF/GRANDES_5_LIGAS_EUR_FBREF_201617_202324_J28.csv', sep=',', index=False)
```

2.4. Resultado

Como resultado de este código se tiene:

- El DataFrame `GRANDES_5_LIGAS_EUR_FBREF_201617_202324_J28` contiene información de las cinco principales ligas de fútbol europeas para múltiples temporadas.
- Cada fila del DataFrame representa estadísticas de un equipo en una temporada específica de una liga específica.
- Las columnas contienen diversas métricas y estadísticas relacionadas con el rendimiento de los equipos en el campo.

3. Procesamiento de los datos

Usando un archivo .ipynb se ejecuta un procesamiento intermedio de las tablas. Este paso intermedio se realiza después de la extracción para poder tener a parte las tablas brutas por si se quieren utilizar para otro tipo de estudio y antes de la creación del output final ya que así cuando la aplicación lance el código este se hace menos pesado y el usuario de la app tiene que esperar menos tiempo.

3.0. Cargar librerías

Importación de bibliotecas:

- **pandas**: Para la manipulación y análisis de datos.
- **numpy**: Para cálculos numéricos.
- **MinMaxScaler** de **sklearn.preprocessing**: Para escalar características en un rango específico, en este caso entre 0 y 100.

0. Cargar librerías

```
[8] import pandas as pd
import numpy as np
from sklearn.preprocessing import MinMaxScaler
pd.set_option('display.max_columns',None)
```

3.1. Cargar datasets

Se leen el archivo CSV creado anteriormente y se almacena en un DataFrame.

1. Cargar Datasets

```
] GRANDES_5_LIGAS_EUR_FBREF_201617_202324_328 = pd.read_csv('/content/drive/MyDrive/Master Analisis de datos futbol/Master/Tablas_FBREF/GRANDES_5_LIGAS_EUR_FBREF_201617_202324_328.csv', sep=',', index_col=False)
print(GRANDES_5_LIGAS_EUR_FBREF_201617_202324_328.shape)
GRANDES_5_LIGAS_EUR_FBREF_201617_202324_328.head()
```

(684, 354)

	Competition	Temporada	Rk	Squad	MP	W	D	L	GF	GA	GD	Pts	Pts/MP	xG	xGA	xGD	xGD/90	Rk_Home	Partidos_Home	W_Home	D_Home	L_Home	GF_Home	GA_Home	GD_Home	Pts_Home	Pts_Partido_Home	xG_Home
0	LALIGA	2023_2024	1	Real Madrid	28	21	6	1	60	18	42	69	2.46	52.2	26.2	26.0	0.93	1	14	12	2	0	35	7	28	38	2.71	31.0
1	LALIGA	2023_2024	2	Girona	28	19	5	4	59	33	26	62	2.21	47.8	38.0	9.8	0.35	2	14	11	2	1	35	14	21	35	2.50	23.8
2	LALIGA	2023_2024	3	Barcelona	28	18	7	3	57	34	23	61	2.18	61.0	29.0	32.0	1.14	3	15	11	1	3	33	19	14	34	2.27	35.3
3	LALIGA	2023_2024	4	Atlético Madrid	28	17	4	7	54	31	23	55	1.96	49.2	29.6	19.6	0.70	4	14	13	1	0	34	13	21	40	2.86	30.7
4	LALIGA	2023_2024	5	Athletic Club	28	15	8	5	48	26	22	53	1.89	42.8	27.7	15.1	0.54	5	14	10	3	1	34	14	20	33	2.36	27.9

3.2. Procesamiento de los datos

En esta parte del código se realizan varias operaciones:

- **Creación de listas de variables:** Se definen tres listas que contienen las variables del conjunto de datos: `variables_porcentuales`, `variables_por_partido` y `variables_absolutas`. Estas listas son utilizadas posteriormente para separar y procesar las variables.

2. Procesamiento de los datos

```
[3] # Hay 3 tipos de variables: absolutas, por partido y porcentuales. Las identifico porque las voy a tratar para escalarlas todas entre 0 y 100
columns = GRANDES_5_LIGAS_EUR_FBREF_201617_202324_J28.columns.tolist()
variables_porcentuales = ['Posesion_media_a_favor', 'SaveK', 'CSK_a_favor', 'Penaltis_SaveK', 'PSxG/SoT', 'Porc_Pases_Largos_Completados_Portero', 'Porc_Pases_Largos_Portero', 'Porc_Saques_Puerta_Largo', 'Porc_Centr',
'Porc_Tackle_Ganado_a_favor', 'Posesion_a_favor', 'Porc_Regates_Completados_a_favor', 'Porcentaje_Tackles_Regate_en_contra', 'Porc_Duelos_Aereos_ganados_equipo_propio',
'Posesion_media_en_contra', 'Porc_SoT_Detenidos_en_contra', 'CSK_en_contra', 'Porc_PK_Detenidos_en_contra', 'PSxG/SoT_a_favor', 'Porc_Pases_Largos_Completados_Portero_rival', 'Porc_Pases',
'Porc_Tackle_Ganado_en_contra', 'Posesion_en_contra', 'Porc_Regates_Completados_en_contra', 'Porcentaje_Tackles_Regate_a_favor', 'Porc_Duelos_Aereos_ganados_equipo_rival', 'Porc_Pases',
'xG/90', 'Gls/90_a_favor', 'Ast/90_a_favor', 'G-PK/90_a_favor', 'xG/90_en_contra', 'xG/90_a_favor', 'xGxAG/90_a_favor', 'npG/90_a_favor', 'npGxAG/90_a_favor', 'GA/90', 'PSxG_por_partido',
'Gls/90_en_contra', 'Ast/90_en_contra', 'G-PK/90_en_contra', 'xG/90_en_contra', 'xG/90_a_favor', 'xGxAG/90_en_contra', 'npG/90_en_contra', 'npGxAG/90_en_contra', 'PSxG_por_pa']
variables_absolutas = [columna for columna in columns if columna not in variables_porcentuales and columna not in variables_por_partido]
variables_absolutas = variables_absolutas[4:]
```

- **Procesamiento de variables absolutas:** Las variables absolutas se dividen por el número de partidos jugados (MP) para convertirlas en variables por partido. Luego, estas variables se combinan con las variables por partido y las variables porcentuales para formar un DataFrame llamado `GRANDES_5_LIGAS_EUR_FBREF_201617_202324_J28_var_abs_std`.

```
[4] # Pongo las variables absolutas como variables por partido
GRANDES_5_LIGAS_EUR_FBREF_201617_202324_J28_var_abs_std = pd.DataFrame()
for col in variables_absolutas:
    GRANDES_5_LIGAS_EUR_FBREF_201617_202324_J28_var_abs_std[col] = GRANDES_5_LIGAS_EUR_FBREF_201617_202324_J28[col]/GRANDES_5_LIGAS_EUR_FBREF_201617_202324_J28['MP']

[5] # Uno las variables por partido y las reordeno
GRANDES_5_LIGAS_EUR_FBREF_201617_202324_J28_var_abs_std = pd.concat([GRANDES_5_LIGAS_EUR_FBREF_201617_202324_J28_var_abs_std, GRANDES_5_LIGAS_EUR_FBREF_201617_202324_J28[variables_por_partido], GRANDES_5_LIGAS_EUR_FBREF_201617_202324_J28[variables_porcentuales]], axis=1)
```

- **Corrección de variables porcentuales:** Algunas variables porcentuales que tienen valores entre 0 y 1 se multiplican por 100 para convertirlas a un rango de 0 a 100.

```
[6] # Corrijo las variables porcentuales que tienen valores entre 0 y 1 convirtiendolas a entre 0 y 100
variables_porcentuales_corr = ['PSxG/SoT', 'Goles_por_disparo_a_puerta', 'xG_sin_penaltis_por_tiro', 'PSxG/SoT_a_favor', 'Goles_por_disparo_en_contra.1', 'xG_en_contra_sin_penaltis_por_tiro']
GRANDES_5_LIGAS_EUR_FBREF_201617_202324_J28_var_abs_std[variables_porcentuales_corr] = 100*GRANDES_5_LIGAS_EUR_FBREF_201617_202324_J28_var_abs_std[variables_porcentuales_corr]
```

- **Escalado de variables:** Todas las variables se escalan para que estén en el rango de 0 a 100 utilizando la clase `MinMaxScaler` de `scikit-learn`. El resultado se almacena en el DataFrame `GRANDES_5_LIGAS_EUR_FBREF_201617_202324_J28_scaled`.

```
[9] # Reescalo todas las variables para que estén entre 0 y 100
scaler = MinMaxScaler(feature_range=(0, 100))

# Escalar los datos
GRANDES_5_LIGAS_EUR_FBREF_201617_202324_J28_scaled = pd.DataFrame(scaler.fit_transform(GRANDES_5_LIGAS_EUR_FBREF_201617_202324_J28_var_abs_std), columns=GRANDES_5_LIGAS_EUR_FBREF_201617_202324_J28_var_abs_std.columns)

[10] # Uno las variables ya escaladas con los identificadores
GRANDES_5_LIGAS_EUR_FBREF_201617_202324_J28_scaled = pd.concat([GRANDES_5_LIGAS_EUR_FBREF_201617_202324_J28.iloc[:,0:4], GRANDES_5_LIGAS_EUR_FBREF_201617_202324_J28_scaled], axis=1)
```

- **Verificación de valores escalados:** Se verifica que todas las variables escaladas están en el rango deseado (entre 0 y 100).

```
[11] # Chequeamos que todas las variables han quedado entre 0 y 100
for col in GRANDES_5_LIGAS_EUR_FBREF_201617_202324_J28_scaled.columns[4:]:
    if (GRANDES_5_LIGAS_EUR_FBREF_201617_202324_J28_scaled[col].min() != 0):
        print(f"Minimum value of {col} is different than 0")
    if (GRANDES_5_LIGAS_EUR_FBREF_201617_202324_J28_scaled[col].max() < 99) or (GRANDES_5_LIGAS_EUR_FBREF_201617_202324_J28_scaled[col].max() > 101):
        print(f"Maximum value of {col} is much different than 100")
```

Maximum value of MP is much different than 100

Se observa que la única variable que no está siempre entre 0 y 100 es MP que siempre vale 0 por su propia definición.

- **Selección de temporadas y campeones:** Se separan las temporadas anteriores a 2023-2024 en un DataFrame llamado `GRANDES_5_LIGAS_EUR_FBREF_201617_202223`, mientras que la temporada 2023-2024 se guarda en el DataFrame `GRANDES_5_LIGAS_EUR_FBREF_202324_J28`. Además, se filtran los campeones de cada liga para un análisis específico en el DataFrame `CHAMP_GRANDES_5_LIGAS_EUR_FBREF_201617_202223`.

```
[12] # Me quedo por un lado con las temporadas completas para que sean la base del estudio
      GRANDES_5_LIGAS_EUR_FBREF_201617_202223 = GRANDES_5_LIGAS_EUR_FBREF_201617_202324_J28_scaled[GRANDES_5_LIGAS_EUR_FBREF_201617_202324_J28_scaled['Temporada']<'2023_2024']
      # Me quedo con la temporada en curso para aplicar los resultados del estudio
      GRANDES_5_LIGAS_EUR_FBREF_202324_J28 = GRANDES_5_LIGAS_EUR_FBREF_201617_202324_J28_scaled[GRANDES_5_LIGAS_EUR_FBREF_201617_202324_J28_scaled['Temporada']=='2023_2024']

[15] # Me quedo con los campeones de cada liga
      CHAMP_GRANDES_5_LIGAS_EUR_FBREF_201617_202223 = GRANDES_5_LIGAS_EUR_FBREF_201617_202223[GRANDES_5_LIGAS_EUR_FBREF_201617_202223['Rk']==1]
```

3.3. Guardado de tablas

Se almacenan las tres tablas en la ruta de Drive C:/Users/carlo/Documents/Master Scouting/Master/Modulo 11. TFM/.

3. Guardado de tablas

```
[18] GRANDES_5_LIGAS_EUR_FBREF_201617_202223.to_csv("/content/drive/MyDrive/Master Analisis de datos futbol/Master/Tablas_FBREF/GRANDES_5_LIGAS_EUR_FBREF_201617_202223_std.csv", sep=',', index=False)
      GRANDES_5_LIGAS_EUR_FBREF_202324_J28.to_csv("/content/drive/MyDrive/Master Analisis de datos futbol/Master/Tablas_FBREF/GRANDES_5_LIGAS_EUR_FBREF_202324_J28_std.csv", sep=',', index=False)
      CHAMP_GRANDES_5_LIGAS_EUR_FBREF_201617_202223.to_csv("/content/drive/MyDrive/Master Analisis de datos futbol/Master/Tablas_FBREF/CHAMP_GRANDES_5_LIGAS_EUR_FBREF_201617_202223_std.csv", sep=',', index=False)
```

3.4. Resultado

Tras la ejecución de este código se obtienen las siguientes tres tablas:

- **GRANDES_5_LIGAS_EUR_FBREF_201617_202223** que contiene todos los datos de todos los equipos de las 5 grandes ligas europeas desde la temporada 2017-2018 hasta la 2022-2023.
- **GRANDES_5_LIGAS_EUR_FBREF_202324_J28** que contiene todos los datos de todos los equipos de las 5 grandes ligas europeas de la temporada 2023-2024.
- **CHAMP_GRANDES_5_LIGAS_EUR_FBREF_201617_202223** que contiene todos los datos de los equipos campeones de liga de las 5 grandes ligas europeas desde la temporada 2017-2018 hasta la 2022-2023.

4. Creación de los gráficos y despliegue en App

Usando un archivo .py para poder usar la librería Streamlit con éxito se ejecuta el siguiente código.

4.0. Cargar librerías

Importación de bibliotecas:

- **streamlit**: Una biblioteca para crear aplicaciones web interactivas con Python.
- **pandas**: Para la manipulación y análisis de datos.
- **numpy**: Para cálculos numéricos.
- **io**: Para la manipulación de datos de entrada y salida.
- **matplotlib.pyplot**: Para trazar gráficos y visualizaciones.
- **StandardScaler** y **MinMaxScaler** de **sklearn.preprocessing**: Para normalización de datos.
- **cosine** y **euclidean** de **scipy.spatial.distance**: Para calcular distancias entre vectores.
- **cosine_similarity** de **sklearn.metrics.pairwise**: Para calcular similitud coseno entre matrices.

Repositorio de GitHub:

- **repo_path**: Una cadena que contiene la URL del repositorio en GitHub donde se subirá todo lo necesario para productivizar la aplicación web.

```
# 0. Importar librerías

import streamlit as st
import pandas as pd
import numpy as np
import io
import matplotlib.pyplot as plt
from sklearn.preprocessing import StandardScaler
from sklearn.preprocessing import MinMaxScaler
from scipy.spatial.distance import cosine, euclidean
from sklearn.metrics.pairwise import cosine_similarity

# Cargo tambien el repositorio de Github donde subiré todo lo necesario para productivizar la app web
repo_path = "https://github.com/Carlossp5/TFM_Carlos_Alvarez.git"
```

4.1. Cargar datasets

En primer lugar se descargan las tablas almacenadas anteriormente en Drive y se trasladan a la carpeta local C:/Users/carlo/Documents/Master Scouting/Master/Modulo 11. TFM/.

Posteriormente, se leen los tres archivos CSV distintos y se almacenan en tres DataFrames diferentes:

- GRANDES_5_LIGAS_EUR_FBREF_201617_202223: este archivo contiene todos los datos estandarizados de todas las temporadas pasadas de los equipos de las 5 grandes ligas europeas.
- GRANDES_5_LIGAS_EUR_FBREF_202324_J28: este archivo contiene todos los datos estandarizados de la temporada actual hasta la jornada 28 de todos los equipos de las 5 grandes ligas europeas.
- CHAMP_GRANDES_5_LIGAS_EUR_FBREF_201617_202223: este archivo contiene todos los datos estandarizados de los equipos que han sido campeones de las 5 grandes ligas en temporadas pasadas.

```
# 1. Lectura de ficheros
# leo las tablas anteriormente guardadas
GRANDES_5_LIGAS_EUR_FBREF_201617_202223 = pd.read_csv("C:/Users/carlo/Documents/Master Scouting/Master/Modulo 11. TFM/GRANDES_5_LIGAS_EUR_FBREF_201617_202223_std.csv", sep=',', index_col=False)
GRANDES_5_LIGAS_EUR_FBREF_202324_J28 = pd.read_csv("C:/Users/carlo/Documents/Master Scouting/Master/Modulo 11. TFM/GRANDES_5_LIGAS_EUR_FBREF_202324_J28_std.csv", sep=',', index_col=False)
CHAMP_GRANDES_5_LIGAS_EUR_FBREF_201617_202223 = pd.read_csv("C:/Users/carlo/Documents/Master Scouting/Master/Modulo 11. TFM/CHAMP_GRANDES_5_LIGAS_EUR_FBREF_201617_202223_std.csv", sep=',', index_col=False)
```

4.2. Definición de los input de la app

En este paso se están definiendo diferentes grupos de variables que se utilizarán como entradas en la aplicación web. Estos grupos de variables están basados en los mismos grupos que utiliza FBREF.

Cada grupo de variables está representado como una lista de cadenas que contienen los nombres de las variables dentro de ese grupo. Además, se crea un diccionario llamado Conjunto_Variables_dict que mapea el nombre del grupo de variables a su respectiva lista de variables.

También se definen los equipos de las 5 grandes ligas europeas de la temporada actual y los tipos de distancia con los que se puede medir la similitud (distancia coseno o distancia euclídea).

Además, se incluye un título para el dashboard utilizando la función st.write() de Streamlit para escribir HTML directamente en la aplicación web.

```
# 2. Definición de los input de la app
# Defino los diferentes paquetes de variables. Mismos grupos que usa FBREF
Clasificacion = ['MP', 'W', 'D', 'L', 'GF', 'GA', 'GD', 'Pts', 'Pts/MP', 'xG', 'xGA', 'xGD', 'xGD/90']
Clasificacion_Home = ['Rk_Home', 'Partidos_Home', 'W_Home', 'D_Home', 'L_Home', 'GF_Home', 'GA_Home', 'GD_Home', 'Pts_Home', 'Pts_Partido_Home', 'xG_Home', 'xGA_Home', 'xGD_Home', 'xGD/90_Home']
Clasificacion_Away = ['Partidos_Away', 'W_Away', 'D_Away', 'L_Away', 'GF_Away', 'GA_Away', 'GD_Away', 'Pts_Away', 'Pts_Partido_Away', 'xG_Away', 'xGA_Away', 'xGD_Away', 'xGD/90_Away']
Standard_Stats_Own = ['Jugadores_Involucrados', 'Age_media_equipo_propio', 'Posesion_media_a_favor', 'Ast_a_favor', 'G+A_a_favor', 'Goles_Sin_Penaltis_a_favor', 'PK', 'Penaltis_Intentados_a_favor',
'Tarjetas_Amarillas_propias', 'Tarjetas_Rojas_propias', 'npxG', 'xAst_a_favor', 'npxG+xAG_a_favor', 'Prog_con_Conduccion_a_favor', 'Prog_con_Pase_a_favor', 'Gls/90_a_favor',
'Ast/90_a_favor', 'G+A/90_a_favor', 'G-PK/90_a_favor', 'xG/90_a_favor', 'xAst/90_a_favor', 'xG+xAG/90_a_favor', 'npxG/90_a_favor', 'npxG+xAst/90_a_favor']
Standard_Stats_Against = ['Age_media_equipos_rivales', 'Posesion_media_en_contra', 'Ast_en_contra', 'G+A_en_contra', 'Goles_Sin_Penaltis_en_contra', 'PK_x', 'Penaltis_Intentados_en_contra',
'Tarjetas_Amarillas_en_contra', 'Tarjetas_Rojas_en_contra', 'npxG_x', 'xAst_en_contra', 'npxG+xAG_en_contra', 'Prog_con_Conduccion_en_contra', 'Prog_con_Pase_en_contra',
'Gls/90_en_contra', 'Ast/90_en_contra', 'G+A/90_en_contra', 'G-PK/90_en_contra', 'xG/90_en_contra', 'xAst/90_en_contra', 'xG+xAG/90_en_contra', 'npxG/90_en_contra',
'npxG+xAst/90_en_contra']
GoalKeeping_Own = ['Porteros_Participado', 'GA/90', 'SoTA', 'Saves', 'Save%', 'CS%_a_favor', 'CS%_en_contra', 'PKsv', 'PKm', 'Penaltis_Save%']
```

...

```
Miscellaneous_Own": ["2.Tarjetas_Amarillas_equipos_rivales","faltas.cometidas_equipos_rivales","fuera.de.juego_equipos_rivales","Goles.Aerios.ganados_equipos_rivales","Goles.Aerios.perdidos_equipos_rivales","Por.Goles.Aerios.ganados_equipos_rivales"]
# Crear los eigthletes
lista_Grandes_5_Ligas_EUR_FBREF_202324_328 = Grandes_5_Ligas_EUR_FBREF_202324_328["Squad"].to_list()
distancias = ["euclidean","cosine"]
Conjunto_Variables=dict={
    'Classification': Classification,
    'Classification_Home': Classification_Home,
    'Classification_Away': Classification_Away,
    'Standard_Stats_Own': Standard_Stats_Own,
    'Standard_Stats_Against': Standard_Stats_Against,
    'Goalkeeping_Own': Goalkeeping_Own,
    'Goalkeeping_Against': Goalkeeping_Against,
    'Goalkeeping_Advance_Own': Goalkeeping_Advance_Own,
    'Goalkeeping_Advance_Against': Goalkeeping_Advance_Against,
    'Shooting_Own': Shooting_Own,
    'Shooting_Against': Shooting_Against,
    'Passing_Own': Passing_Own,
    'Passing_Against': Passing_Against,
    'Pass_Types_Own': Pass_Types_Own,
    'Pass_Types_Against': Pass_Types_Against,
    'Goal_and_Shot_Creation_Own': Goal_and_Shot_Creation_Own,
    'Goal_and_Shot_Creation_Against': Goal_and_Shot_Creation_Against,
    'Defensive_Actions_Own': Defensive_Actions_Own,
    'Defensive_Actions_Against': Defensive_Actions_Against,
    'Possession_Own': Possession_Own,
    'Possession_Against': Possession_Against,
    'Miscellaneous_Own': Miscellaneous_Own
}

# Introduce el título del dashboard
st.write(html style="text-align: center;">GRÁFICOS CUSTOMIZADOS/his, unsafe_allow_html=True)
```

4.3. Creación de los gráficos

Se crea una función que será la encargada de pintar dos gráficos cada vez que se ejecute una selección en la app.

4.3.1 Gráfico de máximo y mínimos

En primer lugar se realiza una preparación de los datos:

- Se selecciona el DataFrame del equipo elegido para la temporada actual.
- Se calculan los valores mínimos y máximos de cada variable en el DataFrame de los campeones de las temporadas anteriores.
- Los datos se preparan para ser dibujados en el gráfico de radar.


```
# Me quedo con la temporada actual del equipo elegido
df_Equipo = GRANDES_5_LIGAS_EUR_FBREF_202324_J28[GRANDES_5_LIGAS_EUR_FBREF_202324_J28['Squad']==Equipo]

# PRIMER GRAFICO
# Saco los minimos y maximos de cada variable y los vuelco en un dataframe
# Crear un nuevo DataFrame vacío
df_min_max = pd.DataFrame()

# Iterar sobre las columnas del DataFrame original
for col in CHAMP_GRANDES_5_LIGAS_EUR_FBREF_201617_202223.columns:
    if CHAMP_GRANDES_5_LIGAS_EUR_FBREF_201617_202223[col].dtype in ['int64', 'float64']:
        # Calcular el valor máximo y mínimo de la columna actual
        max_valor = max(CHAMP_GRANDES_5_LIGAS_EUR_FBREF_201617_202223[col])
        min_valor = min(CHAMP_GRANDES_5_LIGAS_EUR_FBREF_201617_202223[col])

        # Agregar el valor máximo y mínimo al nuevo DataFrame
        df_min_max[col+'_max'] = [max_valor]
        df_min_max[col+'_min'] = [min_valor]

# Dividir el dataframe en dos filas
df_max = df_min_max[df_min_max.columns[df_min_max.columns.str.endswith('_max')]].reset_index(drop=True)
df_min = df_min_max[df_min_max.columns[df_min_max.columns.str.endswith('_min')]].reset_index(drop=True)

# Renombrar las filas
df_max.index = ['Maximo']
df_min.index = ['Minimo']

# Quitamos el sufijo '_min' o '_max'
df_max.columns = df_max.columns.str.replace('_min', '').str.replace('_max', '')
df_min.columns = df_min.columns.str.replace('_min', '').str.replace('_max', '')

df_min_max = pd.concat([df_min, df_max], axis=0)

df_Equipo.set_index('Squad', inplace=True)
df_min_max_equipo = pd.concat([df_min_max, df_Equipo.iloc[:,2:]])

# Definir los datos
labels = np.array(Variables)
min_stats = np.array(df_min_max_equipo[Variables].iloc[0]) # Valores mínimos para cada categoría
max_stats = np.array(df_min_max_equipo[Variables].iloc[1]) # Valores máximos para cada categoría
team_stats = np.array(df_min_max_equipo[Variables].iloc[2]) # Valores máximos para cada categoría
```

A continuación se dibuja el gráfico de radar:

- Se crea un gráfico de radar utilizando la biblioteca Matplotlib.
- Se dibujan las áreas sombreadas correspondientes a los valores mínimos y máximos permitidos.
- Se dibuja una línea para representar los valores del equipo seleccionado.
- Se etiquetan los ejes con el nombre de las variables.
- Se añade una leyenda para indicar qué representan las áreas sombreadas y la línea del equipo.

```
# Definir los datos
labels = np.array(Variables)
min_stats = np.array(df_min_max_equipo[Variables].iloc[0]) # Valores mínimos para cada categoría
max_stats = np.array(df_min_max_equipo[Variables].iloc[1]) # Valores máximos para cada categoría
team_stats = np.array(df_min_max_equipo[Variables].iloc[2]) # Valores máximos para cada categoría

# Número de variables
num_vars = len(labels)

# Ángulos para cada eje
angles = np.linspace(0, 2 * np.pi, num_vars, endpoint=False).tolist()

# El gráfico es un círculo, entonces cerramos el ciclo
min_stats = np.concatenate((min_stats, [min_stats[0]]))
max_stats = np.concatenate((max_stats, [max_stats[0]]))
team_stats = np.concatenate((team_stats, [team_stats[0]]))
angles += angles[:1]

# Crear la figura y el eje polar
fig1, ax = plt.subplots(figsize=(12, 12), subplot_kw=dict(polar=True))

# Dibujar el radar para los valores mínimos
ax.fill(angles, min_stats, color='red', alpha=0.25, label='Valores mínimos')
# Dibujar el radar para los valores máximos
ax.fill(angles, max_stats, color='blue', alpha=0.25, label='Valores máximos')
# Dibujar el radar para los valores máximos
ax.fill(angles, team_stats, color='yellow', alpha=0.25, label=Equipo)

# Establecer las etiquetas de las variables en el eje polar
ax.set_xticks(angles[:-1])
ax.set_xticklabels(labels)

# Añadir leyenda
plt.legend(loc='upper right', bbox_to_anchor=(1.25, 1))

# Para que la leyenda entre en los márgenes de la imagen al descargarla
plt.tight_layout()
```

Por último, se añade un botón a la app para descargar el gráfico:

- Se genera un botón para que el usuario pueda descargar el gráfico como una imagen JPG.

```
# Ponemos la opción de guardar el gráfico como imagen
bufferq = io.BytesIO()
fig1.savefig(bufferq, format='jpeg')
bufferq.seek(0)

# Mostrar el gráfico
columna1.pyplot(fig1)

# Botón de descargar
st.download_button(label="Descargar gráfico 1 como imagen",
                    data=bufferq.getvalue(),
                    file_name='Grafico_1.jpeg',
                    mime="image/jpeg")
```

4.3.2 Algoritmo y gráfico de similitud

Para el segundo gráfico que se representa en la app primero hay que usar el algoritmo de similitud para saber que equipo campeón ha sido el más parecido al equipo seleccionado en las variables seleccionadas y medido con la distancia seleccionada.

Primero se preparan los datos:

- Se selecciona el DataFrame del equipo elegido para la temporada actual.
- Se seleccionan los datos de los campeones de las temporadas anteriores para calcular la similitud con el equipo escogido.
- Se calcula la distancia entre el equipo seleccionado y todos los equipos campeones utilizando la medida de distancia seleccionada (coseno o euclídea).
- Se ordenan los resultados en función de la distancia para identificar el equipo más similar.

```
# SEGUNDO GRAFICO
# Filtrar el dataframe para obtener solo las filas correspondientes al equipo dado
team_data = df_Equipo[['Temporada'] + Variables]

# Filtrar el dataframe para obtener solo las filas correspondientes a los otros equipos
other_teams_data = CHAMP_GRANDES_5_LIGAS_EUR_FBREF_201617_202223[['Squad', 'Temporada'] + Variables]

if Distancia == 'cosine':
    # Calcular la distancia del coseno para cada equipo
    distances = cosine_similarity(other_teams_data[Variables].values, team_data[Variables].values)
    distances = distances[:, 0] # Seleccionamos la primera columna (similitud con el equipo dado)
elif Distancia == 'euclidean':
    # Calcular la distancia euclidiana para cada equipo
    distances = np.sqrt(np.sum((other_teams_data[Variables].values - team_data[Variables].values)**2, axis=1))
else:
    raise ValueError("Tipo de medida de distancia no válido. Utiliza 'cosine' o 'euclidean'.")

# Crear un nuevo dataframe con los equipos, temporadas y sus distancias
result_df = pd.DataFrame({'Temporada': other_teams_data['Temporada'], 'Equipo': CHAMP_GRANDES_5_LIGAS_EUR_FBREF_201617_202223['Squad'], 'Distancia': distances})

# Ordenar los resultados en función de la distancia (menor similitud a mayor similitud)
result_df = result_df.sort_values(by='Distancia', ascending=True)

# Guardar el primer registro en un nuevo dataframe llamado df_similar_team
df_similar_team = result_df.head(1)

Equipo_Similar = df_similar_team['Equipo'].iloc[0]
Temporada_Similar = df_similar_team['Temporada'].iloc[0]

df_similar_team = GRANDES_5_LIGAS_EUR_FBREF_201617_202223[(GRANDES_5_LIGAS_EUR_FBREF_201617_202223['Squad']==Equipo_Similar)&(GRANDES_5_LIGAS_EUR_FBREF_201617_202223['Temporada']==Temporada_Similar)]
df_similar_team.set_index('Squad', inplace=True)
df_equipo_similar = pd.concat([df_Equipo, df_similar_team.iloc[:,2:]])
```

A continuación se dibuja el gráfico de radar:

- Se crea un gráfico de radar utilizando la biblioteca Matplotlib.
- Se dibujan las áreas sombreadas correspondientes a los valores del equipo seleccionado y del equipo más similar.
- Se etiquetan los ejes con el nombre de las variables.
- Se añade una leyenda para indicar qué representan las áreas sombreadas.

```
# Definir los datos
labels = np.array(Variables)
team_stats = np.array(df_equipo_similar[Variables].iloc[0]) # Valores equipo de estudio para cada categoría
similar_team_stats = np.array(df_equipo_similar[Variables].iloc[1]) # Valores equipo de estudio para cada categoría

# Número de variables
num_vars = len(labels)

# Ángulos para cada eje
angles = np.linspace(0, 2 * np.pi, num_vars, endpoint=False).tolist()

# El gráfico es un círculo, entonces cerramos el ciclo
team_stats = np.concatenate((team_stats,[team_stats[0]]))
similar_team_stats = np.concatenate((similar_team_stats,[similar_team_stats[0]]))
angles += angles[:1]

# Crear la figura y el eje polar
fig2, ax = plt.subplots(figsize=(12, 12), subplot_kw=dict(polar=True))

# Dibujar el radar para los valores del equipo de estudio
ax.fill(angles, team_stats, color='yellow', alpha=0.25, label=Equipo)
# Dibujar el radar para los valores del equipo campeón mas similar
ax.fill(angles, similar_team_stats, color='green', alpha=0.25, label=Equipo_Similar)

# Establecer las etiquetas de las variables en el eje polar
ax.set_xticks(angles[:-1])
ax.set_xticklabels(labels)

# Añadir leyenda
plt.legend(loc='upper right', bbox_to_anchor=(1.25, 1))
# Para que la leyenda entre en los márgenes de la imagen al descargarla
plt.tight_layout()
```

Por último, se añade un botón a la app para descargar el gráfico:

- Se genera un botón para que el usuario pueda descargar el gráfico como una imagen JPG.

```
# Ponemos la opción de guardar el gráfico como imagen
buffer2 = io.BytesIO()
fig2.savefig(buffer2, format='jpg')
buffer2.seek(0)

# Mostrar el gráfico
columna2.pyplot(fig2)
columna2.markdown('El equipo mas similar es el {} de la temporada {}'.format(Equipo_Similar, Temporada_Similar))

# Boton de descargar
st.download_button(label="Descargar gráfico 2 como imagen",
                    data=buffer2.getvalue(),
                    file_name='Grafico_2.jpg',
                    mime="image/jpeg")
```

4.4. Definición de los widgets de la app

Se crean unos botones desplegables para poder seleccionar el equipo de la temporada actual, las variables que se quieren estudiar y el tipo de distancia que medirá a que equipo campeón es más similar.

También se añade un botón para ejecutar el código una vez la selección ha sido realizada. Este es el botón 'Generar gráficos'.

```
# 4. Widgets de la app

# Título de la barra lateral de la app
st.sidebar.header("SELECTOR")

# Crear widgets para seleccionar los parámetros
equipo_seleccionado = st.sidebar.selectbox('Selecciona un equipo:', lista_Grandes_5_Ligas_EUR_FBREF_202324_I28)
variables_seleccionadas = st.sidebar.selectbox('Selecciona las variables:', Conjunto_Variables_dict.keys())
distancia_seleccionada = st.sidebar.selectbox('Selecciona la distancia:', distancias)

# Llamar a la función con los parámetros seleccionados
if st.sidebar.button('Generar gráficos'):
    pintar_graf_radar[equipo_seleccionado, Conjunto_Variables_dict[variables_seleccionadas], distancia_seleccionada]
```

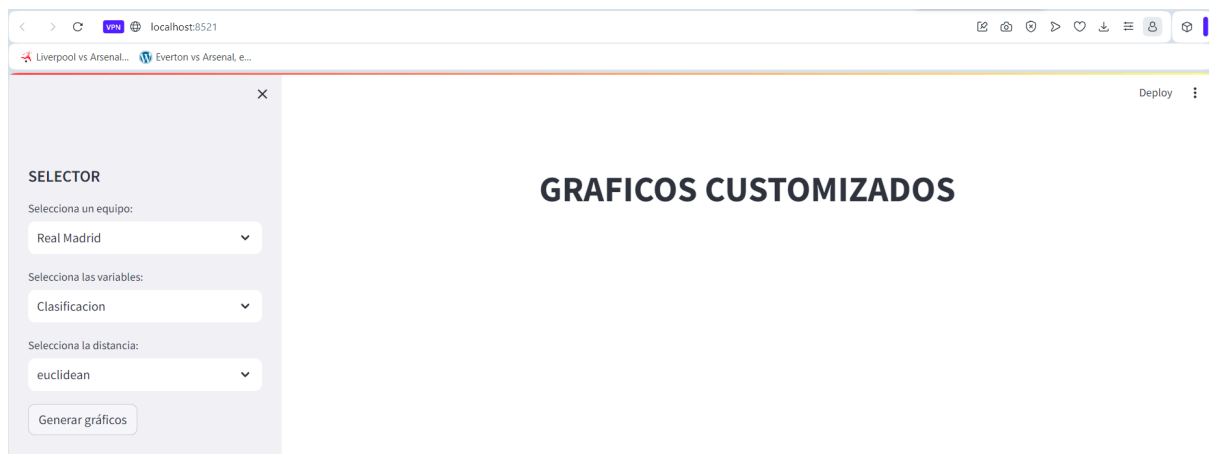
4.5. Resultado

Para ejecutar el código y obtener la aplicación web se debe aplicar en la terminal de la consola de python (en este caso de Virtual Studio Code) el siguiente comando:

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

PS C:\Users\carlo> streamlit run "C:\Users\carlo\Documents\Master Scouting\Master\Modulo 11. TFM\Crear_Graficas.py"
```

Automáticamente se abrirá la siguiente página web local en el navegador predeterminado:



5. Puesta en producción

El último paso implica la disponibilización de la aplicación para el público en general, utilizando los servidores de Streamlit. Esto se logra mediante los siguientes pasos:

Se cargan en GitHub los conjuntos de datos necesarios para el funcionamiento de la aplicación, nombrados como:

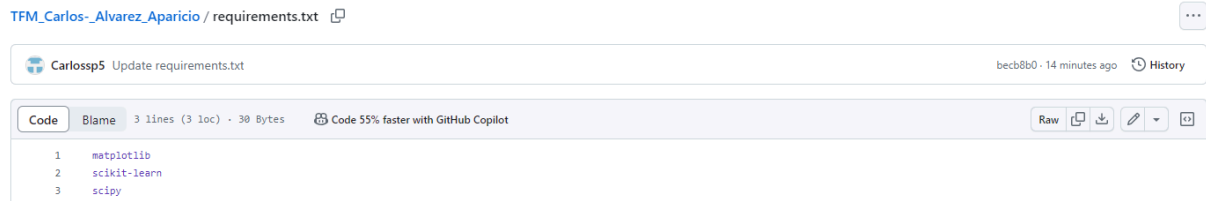
- GRANDES_5_LIGAS_EUR_FBREF_201617_202223
- GRANDES_5_LIGAS_EUR_FBREF_202324_J28
- CHAMP_GRANDES_5_LIGAS_EUR_FBREF_201617_202223

Se ajustan las rutas para que apunten a la carpeta correspondiente en GitHub donde están almacenados estos archivos.

```
# Cargo también el repositorio de GitHub donde subiré todo lo necesario para producir la app web
repo_path = "https://github.com/Carlossp5/TFM_Carlos-Alvarez-Aparicio.git"

# 1. Lectura de ficheros
# leo las tablas anteriormente guardadas
GRANDES_5_LIGAS_EUR_FBREF_201617_202223 = pd.read_csv("https://raw.githubusercontent.com/Carlossp5/TFM_Carlos-Alvarez-Aparicio/main/Datasets/GRANDES_5_LIGAS_EUR_FBREF_201617_202223_std.csv", sep=',', index_col=False)
GRANDES_5_LIGAS_EUR_FBREF_202324_J28 = pd.read_csv("https://raw.githubusercontent.com/Carlossp5/TFM_Carlos-Alvarez-Aparicio/main/Datasets/GRANDES_5_LIGAS_EUR_FBREF_202324_J28_std.csv", sep=',', index_col=False)
CHAMP_GRANDES_5_LIGAS_EUR_FBREF_201617_202223 = pd.read_csv("https://raw.githubusercontent.com/Carlossp5/TFM_Carlos-Alvarez-Aparicio/main/Datasets/CHAMP_GRANDES_5_LIGAS_EUR_FBREF_201617_202223_std.csv", sep=',', index_col=False)
```

Se crea un archivo llamado requirements.txt en la misma carpeta de GitHub donde se encuentra el código. Este archivo contiene la lista de bibliotecas que se utilizarán y que no están preinstaladas por defecto en Streamlit.

A screenshot of a GitHub web interface showing the 'requirements.txt' file in the 'TFM_Carlos-Alvarez-Aparicio' repository. The file content is: 1 matplotlib, 2 scikit-learn, 3 scipy. The interface includes a header with the repository name, a commit hash 'becb8b0' from 14 minutes ago, and a 'History' link. Below the file name, it shows '3 lines (3 loc) · 30 Bytes' and a note 'Code 55% faster with GitHub Copilot'. On the right, there are icons for 'Raw', 'Download', 'Edit', and 'Add'.

Se copia la URL del código que genera la aplicación en GitHub, y Streamlit genera automáticamente la interfaz web en sus servidores.

La ruta al GitHub:

https://github.com/Carlossp5/TFM_Carlos-Alvarez-Aparicio/tree/main

6. Ejemplos de uso

6.1 Ejemplo de uso 1. Director Deportivo del Girona

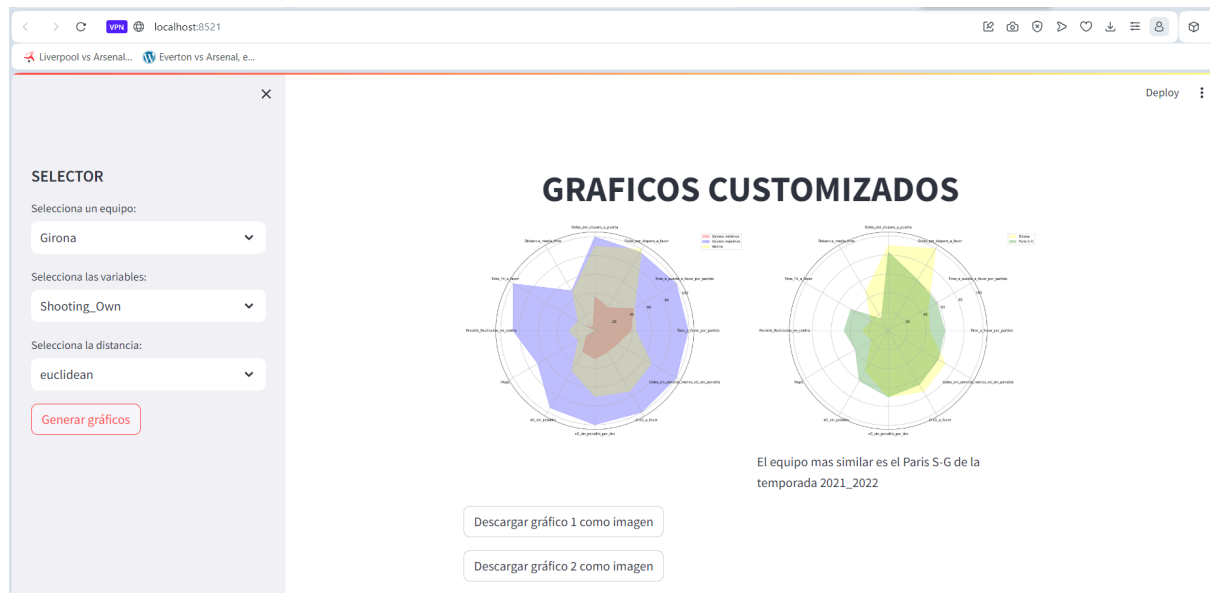
En el presente epígrafe se va a adjuntar un ejemplo de uso en el hay que imaginar que somos el director deportivo del Girona y queremos ver cómo podemos mejorar la finalización del equipo mediante fichajes.

Para ello establecemos la siguiente configuración en la app:



The image shows a web application interface titled "SELECTOR". It contains three dropdown menus for configuration: "Selecciona un equipo:" with "Girona" selected, "Selecciona las variables:" with "Shooting_Own" selected, and "Selecciona la distancia:" with "euclidean" selected. Below these menus is a button labeled "Generar gráficos".

Se pincha en el botón Generar gráficos y se obtiene la siguiente visualización:



Pinchando en los botones de Descarga se puede obtener estos gráficos en formato jpg para llevarlos a nuestros informes de forma sencilla:

Gráfico 1:

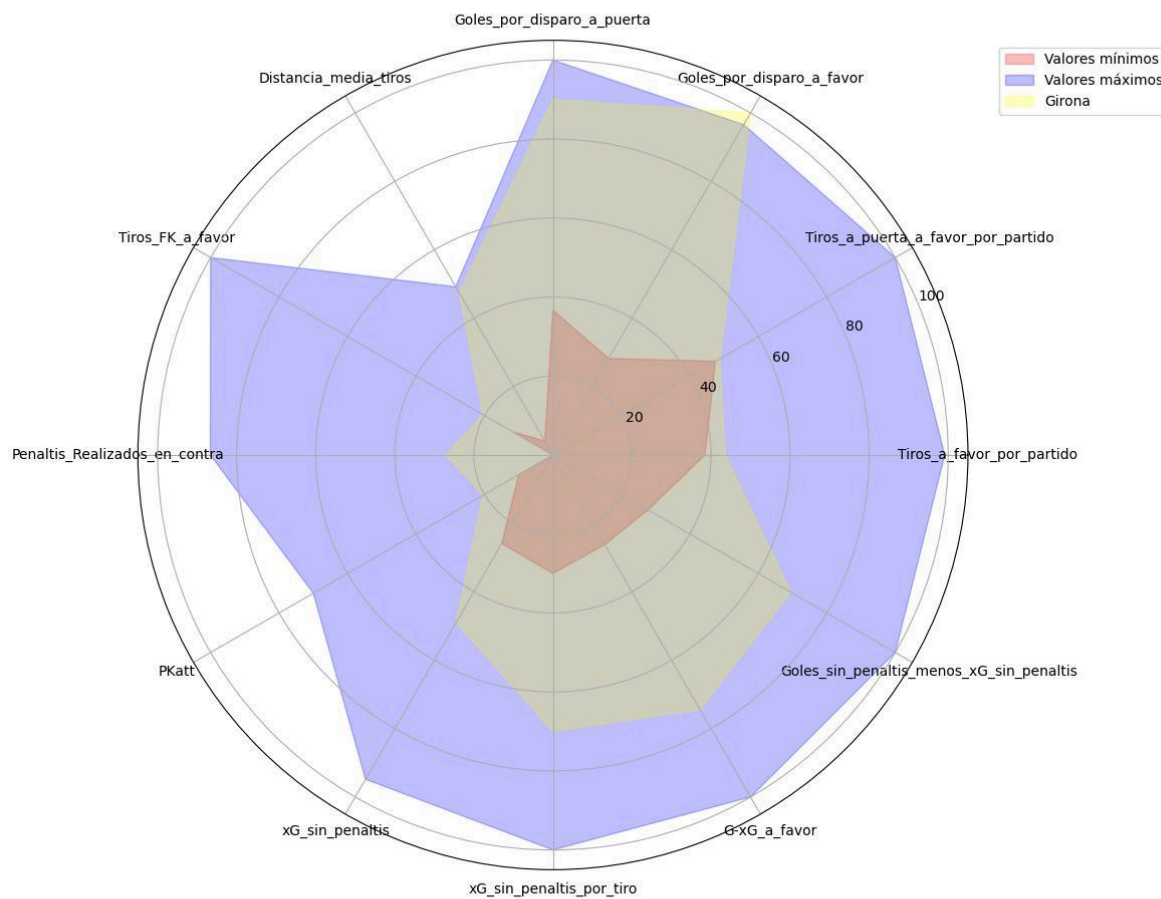


Imagen del gráfico del Girona en las variables relacionadas con el tiro, comparado con los valores máximos y mínimos que han sido jamás alcanzados por un equipo campeón de alguna de las 5 grandes ligas europeas entre las temporadas 2017-2018 y 2022-2023.

En este caso como Directores Deportivos del Girona podemos observar que el equipo en esta temporada tiene valores de campeón en el ámbito de los tiros a portería. Siendo incluso mejor que el mejor registro jamás obtenido por un equipo campeón en 'Goles por disparo a favor'.

Gráfico 2:

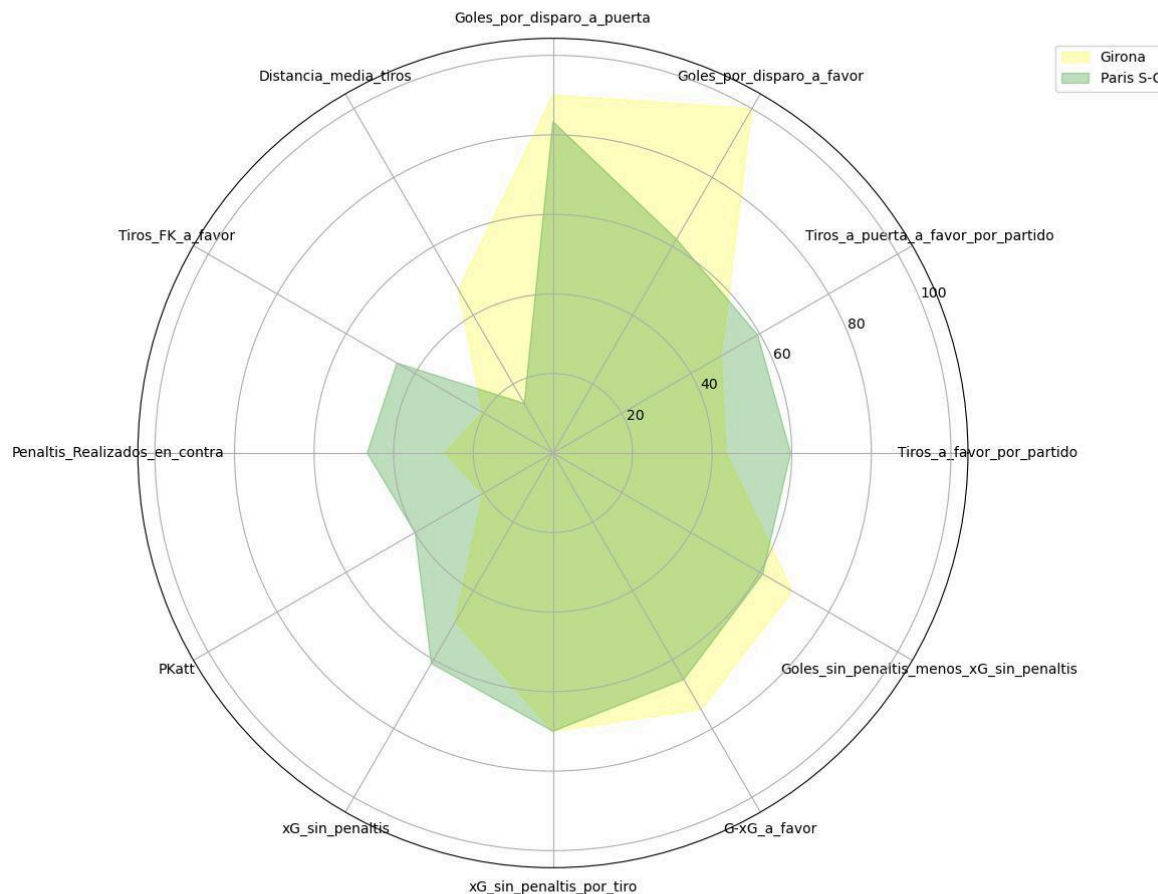


Imagen del gráfico del Girona en las variables relacionadas con el tiro, comparado con el equipo con el que mayor similitud tiene según la distancia euclídea en dichas variables.

En este caso el Girona de la temporada 2023-2024 tiene similitud en cuanto a las variables de disparos con el Paris Saint Germain de la temporada 2021-2022.

Como Directores Deportivos del Girona observamos que el PSG realizaba más golpeos a puerta que el Girona, por lo que podemos interpretar principalmente que si sacrificamos algo de eficacia (dónde el Girona destaca en Goles por Disparo) a cambio de generar más disparos (para parecernos más al PSG de la temporada 2021-2022) podríamos tener una mejora en el rendimiento colectivo por lo que buscaremos jugadores que estén disparando a puerta más que los que tenemos en el equipo aunque su eficacia en goles por disparo no sea tan elevada como la nuestra.

6.2 Ejemplo de uso 2. Analista del Arsenal.

En el presente epígrafe se va a adjuntar un ejemplo de uso en el hay que imaginar que somos el analista del Arsenal y queremos ver cómo podemos mejorar la finalización del equipo mediante entrenamiento.

Para ello establecemos la siguiente configuración en la app:



The image shows a mobile application interface titled "SELECTOR". It features three dropdown menus for configuration. The first dropdown, labeled "Selecciona un equipo:", has "Arsenal" selected. The second dropdown, labeled "Selecciona las variables:", has "Possession_Own" selected. The third dropdown, labeled "Selecciona la distancia:", has "euclidean" selected. At the bottom of the form is a button labeled "Generar gráficos". A red line is visible at the top of the screen, and a close button (X) is in the top right corner of the modal.

Gráfico 1:

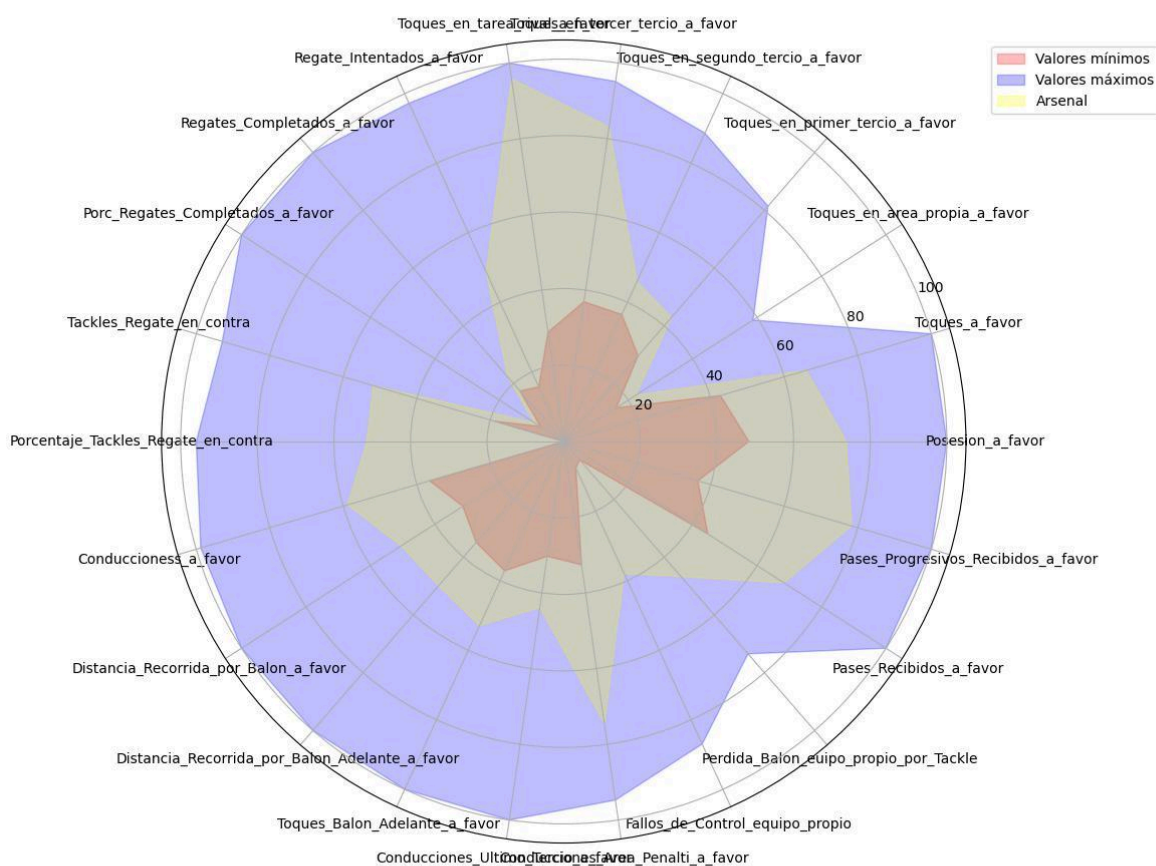


Imagen del gráfico del Arsenal en las variables relacionadas con posesión, comparado con los valores máximos y mínimos que han sido jamás alcanzados por un equipo campeón de alguna de las 5 grandes ligas europeas entre las temporadas 2017-2018 y 2022-2023.

En este caso como Analistas del Arsenal podemos observar que el equipo en esta temporada tiene valores de campeón en el ámbito de posesión en todas las variables medidas.

Gráfico 2:

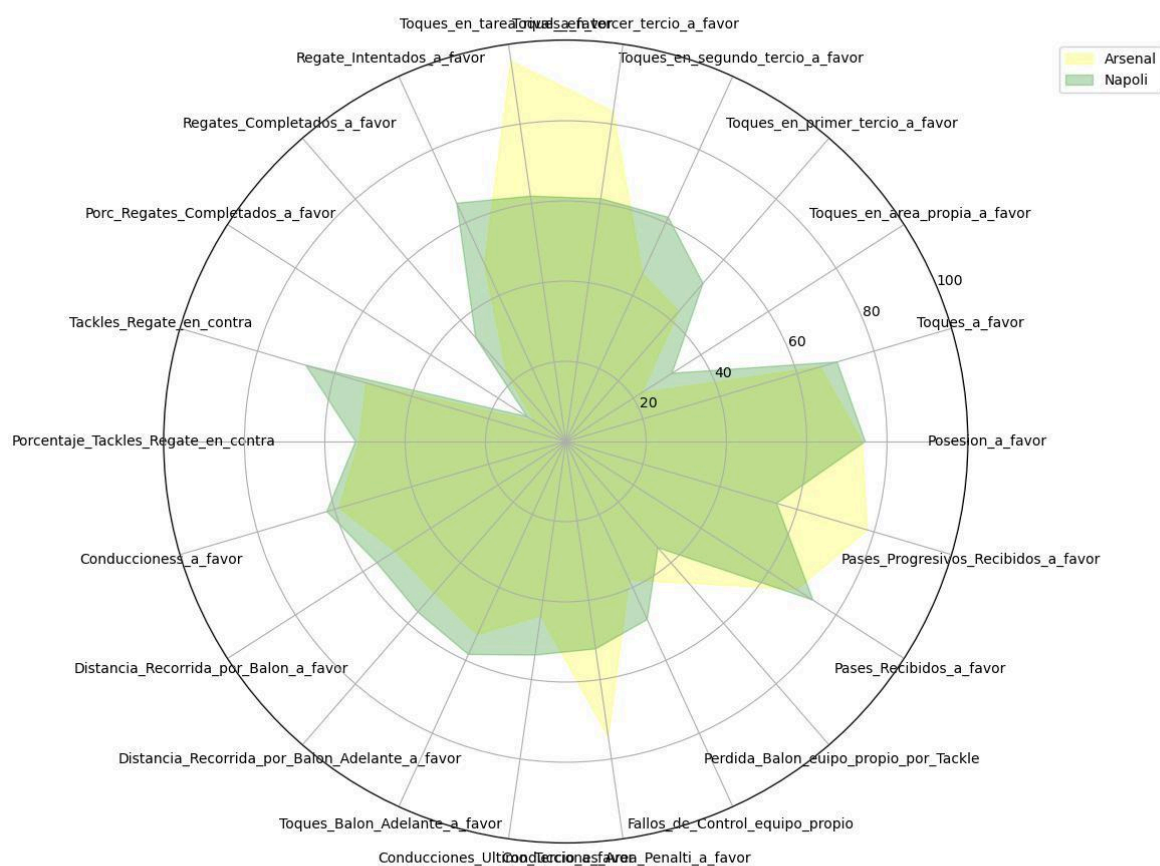


Imagen del gráfico del Arsenal en las variables relacionadas la defensa, comparado con el equipo con el que mayor similitud tiene según la distancia euclídea en dichas variables.

En este caso el Arsenal de la temporada 2023-2024 tiene similitud en cuanto a las variables defensivas con el Nápoles de la temporada 2022-2023.

Como Analistas del Arsenal observamos que el Nápoles daba menos toques en el área contraria y más en el resto de zonas. Sin embargo, vemos mejores estadísticas en regates realizados. Por esto mismo podemos recomendar al entrenador que debemos mejorar las situaciones de encarar así como entrenar la resolución de esta tarea específica del juego. A cambio de dar menos toques en el último tercio.

7. Conclusión

A través de este Proyecto de Fin de Máster se ha desarrollado una aplicación de acceso público que ofrece gráficos de radar comparativos. Estos gráficos contrastan el desempeño del equipo seleccionado durante la temporada 2023-2024 con los valores máximos y mínimos de los equipos campeones de las cinco principales ligas europeas desde la temporada 2017-2018 hasta la temporada 2022-2023. Además, la aplicación determina cuál de los campeones anteriores se asemeja más al equipo bajo estudio y proporciona una comparación detallada. Estos análisis se basan en métricas seleccionadas, utilizando la clasificación de FBREF.

Estos gráficos, que son fácilmente descargables, pueden ser utilizados para la elaboración de informes, la toma de decisiones por parte de la Dirección Deportiva, y para identificar áreas de mejora colectiva dentro del Cuerpo Técnico.

8. Anexos

8.1 Anexo I. Link de acceso a la Aplicación

<https://tfmcarlos-alvarez-aparicio.streamlit.app/>