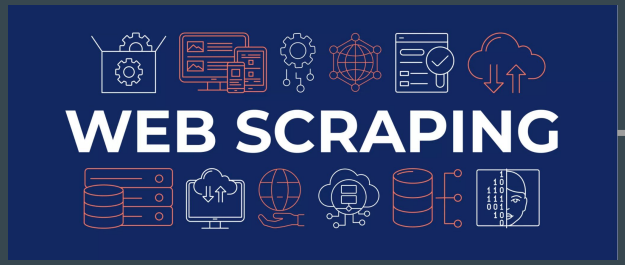
TRABAJO ATD

Alejandro Navarro del Castillo
Carlos Barrado Gutierrez
Aleixandre Tarrasó Sorní

ÍNDICE

Introducción



STATHEAD
FBREF

Premier League

BUNDESLIGA

SERIE A

LALIGA

CSV

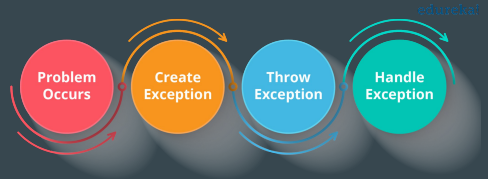
Web Scraping: Librerías y datos


```python
ligas = ["La Liga", "Serie A", "Bundesliga", "Premier League",
         "Ligue 1"]  # Ligas a considerar

años = list(range(2023, 2017, -1))  # Temporadas a considerar

tipos_tablas = [
    "standard", "keeper", "defense", "passing", "possession", "shooting"
]  # Tipos de tablas a considerar

tipos_tablas_cod = {tipo_t: i for i, tipo_t in enumerate(tipos_tablas)} # Tipos de tablas con una id asociada
tipo_dato = ["for"]  # Tipos de datos a considerar (for (local) o/y against (visitante))
```

Web Scraping: Gestión para guardar correctamente los archivos

```python
def crear_ruta(carpeta_es, carpeta_en, nombre_archivo):
  ruta_carpeta_es = os.path.join(os.path.expanduser("~"),
                                 carpeta_es)  #Ruta si tienes el SO en español
  ruta_carpeta_en = os.path.join(os.path.expanduser("~"),
                                 carpeta_en)  #Ruta si tienes el SO en inglés

  if os.path.exists(ruta_carpeta_es):
    ruta_carpeta = ruta_carpeta_es
  elif os.path.exists(ruta_carpeta_en):
    ruta_carpeta = ruta_carpeta_en
  else:
    print("No se encontró ninguna carpeta de descargas.")

  ruta_archivo = os.path.join(ruta_carpeta, nombre_archivo)

  return ruta_archivo


carpeta = ["Descargas", "Downloads"] # Carpeta donde se van a guardar los archivos

archivos = [("ruta_archivo_total", "trabajoatd_bd_en.xlsx"),
            ("ruta_archivo_liga_año", "trabajoatd_la_en.xlsx"),
            ("ruta_archivo_csv", "bdtotal_en.csv")] # [(nombre_ruta, nombre_archivo), ...]

rutas = {} # Diccionario para almacenar las rutas de los archivos
for ruta, nom_arch in archivos:
  rutas[ruta] = crear_ruta(carpeta[0], carpeta[1], nom_arch)
```

Web Scraping: Funciones clave




```python
def extraer_datos(sopa, id_tabla):
  try:
    tabla = sopa.find("table", id=id_tabla)
    if tabla is not None:
      df_tabla = pd.read_html(str(tabla))[0]
      return df_tabla
    else:
      print(f"No se encontró ninguna tabla con el ID '{id_tabla}'.")
      return None
  except pd.errors.EmptyDataError:
    print(f"Error: La tabla con ID '{id_tabla}' está vacía.")
    return None
  except Exception as e:
    print(
        f"Error al intentar extraer datos de la tabla con ID '{id_tabla}': {e}"
    )
    return None
```

```python
def eliminar_col(df, kw):  # Función más eficiente gracias a la vectorización
  cols_eliminar = df.filter(like=kw).columns
  try:
    df.drop(columns=cols_eliminar, inplace=True)
  except KeyError as e:
    print(f"No se encontraron columnas en el DataFrame que contengan {kw}")
  except Exception as e:
    print(f"Se produjo un error inesperado: {e}")
```

```python
def ordenar_df(df, keyword):
  try:
    # Intentar ordenar el DataFrame por la columna especificada
    df_ordenado = df.sort_values(by=keyword).reset_index(drop=True)
    return df_ordenado
  except Exception as e:
    # Manejar el error si la columna especificada no existe
    if isinstance(e, KeyError):
      print(f'Error: La columna "{keyword}" no existe en el DataFrame.')
      return df
    else:
      print(f"Se produjo un error inesperado tratando de ordenar el DF.")
      return df

def renombrar_col(index, cod_t):
  try:
    nuevo_index = [
        f"{cod_t}-{nivel2}-90" if "90" in nivel1 else f"{cod_t}-{nivel2}"
        if "Unnamed" in nivel1 else f"{cod_t}-{nivel1}:{nivel2}"
        for nivel1, nivel2 in index
    ]
    index = index.droplevel(1)
    return pd.Index(nuevo_index, name=index.name)
  except Exception as e:
    print(f"Error en la función renombrar_col: {e}")
```

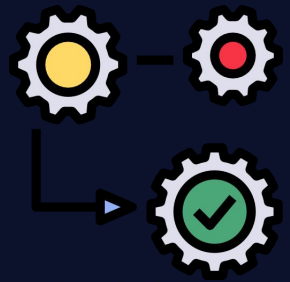
Web Scraping: Código Main (Tratamiento y extracción de datos)

```
df_final = pd.DataFrame()  # Crear dataframe común a todas las ligas y años
url_ini = "https://fbref.com/en/comps" # URL base

with pd.ExcelWriter(rutas["ruta_archivo_liga_año"],
                    engine='xlsxwriter') as writer:

  for liga in ligas:

    n_id_liga, n_comp = extraeridliga(liga) #Manejo de ids
    liga = liga.replace(" ", "-")  # Formato para URL

    for año in años:
      url = f"{url_ini}/{n_comp}/{año-1}-{año}/{año-1}-{año}-{liga}" #URL a scrapear
      resp = requests.get(url)
      sopa = BeautifulSoup(resp.text, "html.parser")

      df_combinado = pd.DataFrame()  # Crear dataframe común a una liga y un año
      ids_tablas_rs = [f"results{año-1}-{año}{n_id_liga}_overall"
                       ]  # Ids de las tablas de la Regular Season

      for id_rs in ids_tablas_rs:
        df_tabla = extraer_datos(sopa, id_rs)
        df_tabla = ordenar_df(df_tabla, "Squad")
        df_combinado = pd.concat([df_combinado, df_tabla], axis=1)
```

```
for tipo_t, cod_t in tipos_tablas_cod.items():
  id_t = f"stats_squads_{tipo_t}_{tipo_dato[0]}" # ID de la tabla
  df_tabla = extraer_datos(sopa, id_t)
  if df_tabla is not None: #Postprocesado
    df_tabla = ordenar_df(df_tabla, "Squad")
    df_tabla.columns = renombrar_col(df_tabla.columns, cod_t)
    eliminar_col(df_tabla, "Squad")
    eliminar_col(df_tabla, "Playing Time")
  df_combinado = pd.concat([df_combinado, df_tabla], axis=1)
```

Web Scraping: Código Main (Control de solicitudes y Guardado)

```
        df_combinado["League"] = liga  # Actualizar el valor de la columna Liga
        df_combinado["Season"] = f"{año-1}-{año}"  # Actualizar el valor de la columna Año

        df_final = pd.concat([df_final, df_combinado],ignore_index=True)

    time.sleep(1)  #Esperar para que no salte la alerta antiwebscraping

df_final.to_excel(rutas["ruta_archivo_total"], engine="xlsxwriter", sheet_name="Hoja1")  # Guardar el dataframe_total en un excel
df_final.to_csv(rutas["ruta_archivo_csv"])  #Guardar el dataframe_total en un csv
```



Web Scraping: Transformaciones

```python
def transformacion_pd(df, lista):
    for atr in lista:
        first_value = df.loc[:, atr].iloc[0]
        if isinstance(first_value, int):
            df[atr] = df[atr].astype(float)
        elif isinstance(first_value, str):
            try:
                df[atr] = df[atr].str.replace(',', '.').astype(float)
            except AttributeError:
                pass
    return df
```

Función para transformar una serie de atributos en tipo float

Problemas que han aparecido

Respecto al web scraping:

1. Cada url de la liga tiene una id arbitraria.
2. La página fbref tiene un método *antiwebscraping* .
3. En diferentes tablas existen mismas columnas que habrá que eliminar.

Respecto al análisis:

1. Las ligas no presentan el mismo número de partidos por temporada.

Desarrollo del trabajo

Además de realizar el minado de datos, también hemos realizado un análisis de ellos y hemos extraído diferentes conclusiones que explicaremos a continuación:

1. **Competitividad en las ligas**
2. **Porteros principales con mejores porcentajes**

Competitividad en las ligas

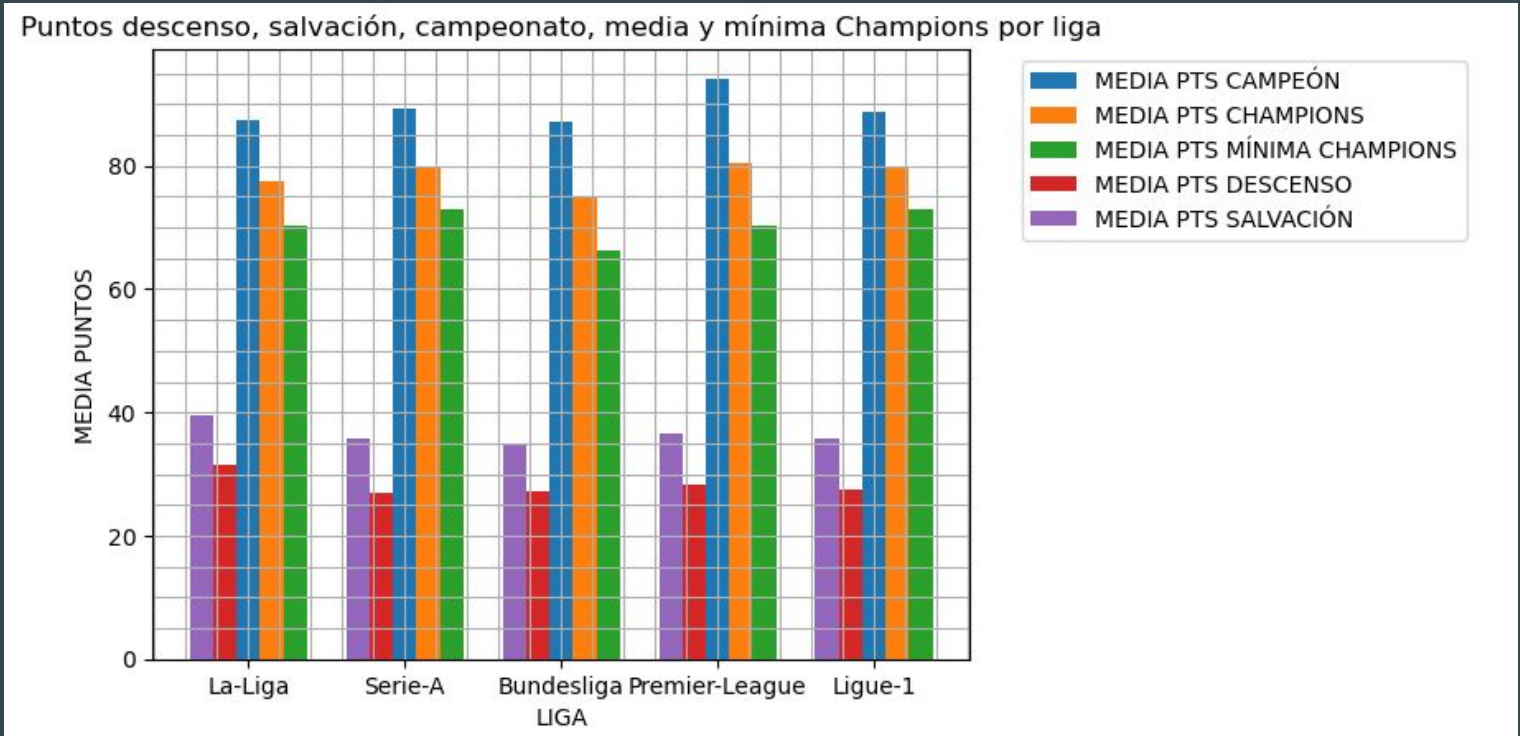


Gráfico de Barras para las Medias de Puntos de cada liga para diferentes variables

Competitividad en las ligas

Gráfico de Barras de las desviaciones para cada temporada de LA LIGA

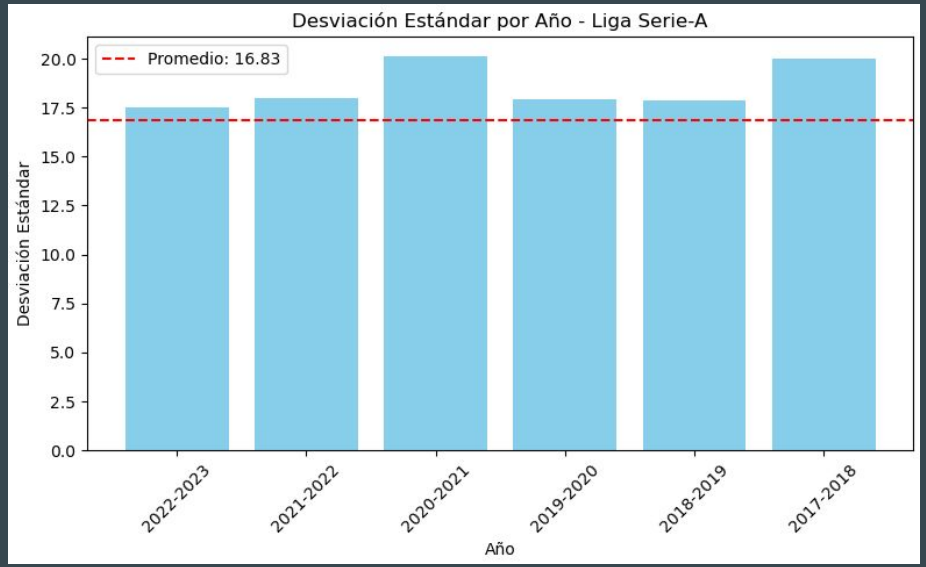


Gráfico de Barras de las desviaciones para cada temporada de la SERIE A

Mejores Temporadas de Porteros

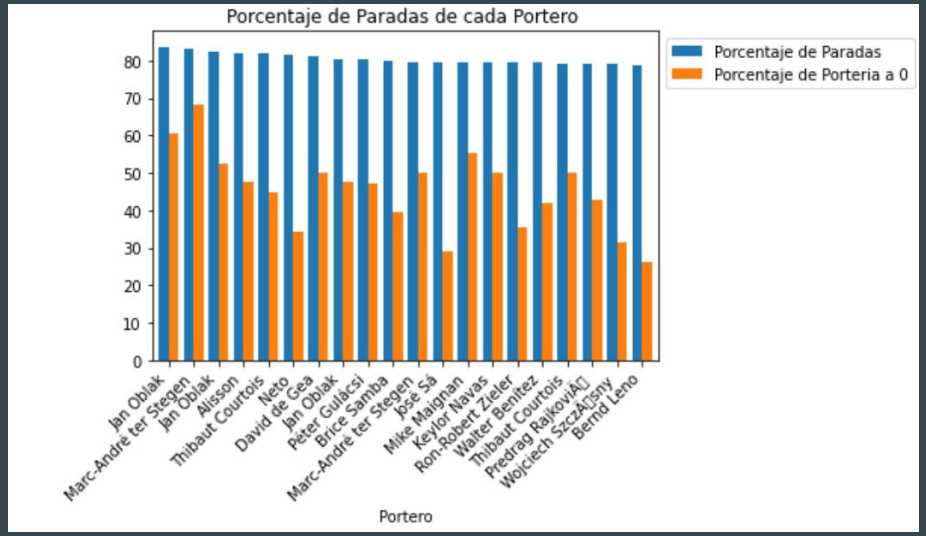


Gráfico de Barras de Porcentaje de Paradas y de Portería a 0

Goalkeeper	Squad	Season	1-Performance:GA90	1-Performance:Save%	1-Performance:CS%	1-Penalty Kicks:PKatt	1-Penalty Kicks:Save%
Jan Oblak	Atlético Madrid	2017-2018	0.58	83.7	60.5	4.0	75
Marc-André ter Stegen	Barcelona	2022-2023	0.53	83.3	68.4	3.0	0
Jan Oblak	Atlético Madrid	2018-2019	0.76	82.4	52.6	5.0	0
Alisson	Roma	2017-2018	0.74	81.9	47.4	5.0	40
Thibaut Courtois	Real Madrid	2020-2021	0.74	81.9	44.7	8.0	12,5
Neto	Valencia	2018-2019	0.92	81.6	34.2	8.0	12,5
David de Gea	Manchester Utd	2017-2018	0.74	81.1	50.0	1.0	0
Jan Oblak	Atlético Madrid	2020-2021	0.66	80.3	47.4	4.0	66,7
Péter Gulácsi	RB Leipzig	2018-2019	0.85	80.2	47.1	6.0	0
Brice Samba	Lens	2022-2023	0.76	79.8	39.5	3.0	0
Marc-André ter Stegen	Barcelona	2017-2018	0.76	79.7	50.0	2.0	0
José Sá	Wolves	2021-2022	1.13	79.6	28.9	10.0	0

Estadísticas de los Porteros ordenada por elmayor porcentaje de paradas de cada temporada

Conclusiones