

Proyecto Data Science From Scratch

Predicción del Campeón de la Copa del Mundo 2022

En este proyecto intentaremos predecir al campeón de fútbol de la Copa del Mundo 2022 a partir de los **datos históricos** que recolectaremos, limpiaremos y organizaremos para realizar, posteriormente, la predicción del campeón de mundo 2022. Para ello utilizaremos la [Distribución de Poisson](#)

Fases de un Proyecto de Data Science

 Fases de un Proyecto de Data Science

Parte 1

Extracción de los grupos 2022

```
In [1]: import pandas as pd
```

```
In [2]: # Leo todas las tablas - Tablas vacías de Mundial 2022
all_tables = pd.read_html('https://web.archive.org/web/20221115040351/https:
```

```
In [3]: # Leo las tablas que me interesan (Grupo A hasta Grupo H), las limpio y las
tables = []
for i in range(12, 62, 7):
    table = all_tables[i]

    # corrijo las columnas
    table.pop('Qualification')
    table.rename(columns={'Teamvte': 'Team'}, inplace=True)

    tables.append(table)
```

```
In [4]: # Creo un diccionario emparejando el 'grupo' con su respectiva 'tabla'
groups = {}
for i in range(0, len(tables)):
    k = chr(65 + i)
    v = tables[i]

    groups[f'Group {k}'] = v
```

```
In [5]: import pickle
```

```
In [6]: # Guardo el diccionario de grupos serializado en disco
with open('files/groups.dat', 'wb') as out:
    pickle.dump(groups, out)
```

Parte 2

Recolección de datos (todos los mundiales del 1930 al 2018)

```
In [7]: import pandas as pd
import requests
from bs4 import BeautifulSoup
```

```
In [8]: def get_matches(url, year):
    """Obtiene todos los partidos jugados en un año determinado.

    Parameters
    -----
    year : str or int
        Año del mundial.

    Returns
    -----
    pandas.core.frame.DataFrame: Dataframe con el resultado de todos los equ

    """

    response = requests.get(url)
    content = response.text

    # parser lxml
    soup = BeautifulSoup(content, 'lxml')

    # todos los partidos (encuentros)
    matches = soup.find_all('div', class_='footballbox')

    home = []
    score = []
    away = []
    for match in matches:
        home.append(match.find('th', class_='fhome').get_text().replace('\xa
        score.append(match.find('th', class_='fscore').get_text().replace('
        away.append(match.find('th', class_='faway').get_text().replace('\xa

    df = pd.DataFrame({
        'Home': home,
        'Score': score,
        'Away': away
    })
```

```
df['Year'] = year

return df
```

```
In [9]: # Obtengo los dataframes de todos los mundiales (todos los años)
matches_list = []
for year in range(1930, 2019, 4):
    # elimino los años que no se jugaron mundiales
    if year == 1942 or year == 1946:
        continue

    # URL de la cual voy a extraer los datos de los partidos
    url = f'https://en.wikipedia.org/wiki/{year}_FIFA_World_Cup'

    matches_list.append(get_matches(url, year))
```

```
In [10]: # unifico todos los mundiales en un único dataframe
df_fifa = pd.concat(matches_list, ignore_index=True)

# los exporto a un archivo csv
df_fifa.to_csv('files/fifa_worldcup_historical_data.csv', index=False)
```

Extracción del Fixture 2022

```
In [11]: # URL de la cual voy a extraer los datos de los partidos
url = 'https://web.archive.org/web/20221115040351/https://en.wikipedia.org/w

# año del mundial sin datos (2022)
year = 2022

# extraigo los partidos (encuentros) que se disputarán en 2022
df_fixture = get_matches(url, year)

# lo exporto a un archivo csv
df_fixture.to_csv('files/fifa_worldcup_fixture_2022.csv', index=False)
```

Parte 3

Obtención de los datos faltantes con la librería *Selenium*

```
In [12]: import pandas as pd
from time import sleep

from selenium import webdriver
from selenium.webdriver.firefox.service import Service
```

```
In [13]: def extract_matches(driver, year):
    """Extrae los partidos "faltantes" de la Copa del Mundo en un año determ
```

```

Args:
    driver (selenium webdriver): webdriver de selenium instanciado con e
    year (str or int): año del mundial

Returns:
    pandas.core.frame.dataframe: dataframe con los partidos (equipos y r
    """

URL = f'https://en.wikipedia.org/wiki/{year}_FIFA_World_Cup'

driver.get(URL)

matches = driver.find_elements('xpath', '//tr[@style="font-size:90%"]')
# matches = driver.find_elements('xpath', '//tr[@itemprop="name"]')

home = []
score = []
away = []
for match in matches:
    home.append(match.find_element('xpath', './td[1]').text.replace('\xa
    score.append(match.find_element('xpath', './td[2]').text.replace(' (
    away.append(match.find_element('xpath', './td[3]').text.replace('\xa

df = pd.DataFrame({
    'Home': home,
    'Score': score,
    'Away': away
})

df['Year'] = year

return df

```

```

In [14]: service = Service(executable_path='driver/geckodriver')
driver = webdriver.Firefox(service=service)

df_list = []
for year in range(1930, 2019, 4):
    # en estos años no se jugaron mundiales
    if year == 1942 or year == 1946:
        continue

    df = extract_matches(driver, year)
    df_list.append(df)

    sleep(1)

driver.close()

df_fifa = pd.concat(df_list, ignore_index=True)
df_fifa.to_csv('files/fifa_worldcup_missing_data.csv', index=False)

```

Parte 4

Limpieza de los datos

```
In [15]: import pandas as pd
```

```
In [16]: historical_data = pd.read_csv('files/fifa_worldcup_historical_data.csv')
missing_data = pd.read_csv('files/fifa_worldcup_missing_data.csv')
fixture = pd.read_csv('files/fifa_worldcup_fixture_2022.csv')
```

Limpiando el fixture

```
In [17]: # Esto en mi caso no es necesario porque ya lo limpié al momento de extraer
# Sin embargo lo dejo documentado como se hace en el curso porque son métodos
fixture['Home'] = fixture['Home'].str.strip()
fixture['Away'] = fixture['Away'].str.strip()
```

Limpiando *missing data*

```
In [18]: # Esto está mal en el curso (es incorrecto), y en mi caso no es
# necesario, sin embargo dejo documentado lo que se hizo.

# para saber si hay datos nulos (no son los NaN como se dice)
missing_data[missing_data['Home'].isnull()]
# para eliminarlos
missing_data.dropna(inplace=True)
```

Agrupando y limpiando toda la data (historical y missing)

```
In [19]: # agrupo los dataframes con datos de los partidos
all_data = pd.concat([historical_data, missing_data], ignore_index=True)

# Esto en mi caso no es necesario porque ya lo limpié al momento de extraer
# Sin embargo lo dejo documentado como se hace en el curso porque son métodos
all_data['Home'] = all_data['Home'].str.strip()
all_data['Away'] = all_data['Away'].str.strip()

# elimino valores nulos
all_data.dropna(inplace=True)

# elimino valores duplicados
all_data.drop_duplicates(inplace=True)

# ordeno todos los datos por los valores de la columna 'Year' (de menor a mayor)
all_data.sort_values(by='Year', inplace=True)
```

```
In [20]: # Hay un partido que no se jugó
# busco el index (sé cual es porque se indica en el curso)
index_to_drop = all_data[all_data['Home'].str.contains('Sweden')] &
```

```
all_data['Away'].str.contains('Austria')).index

# elimino el partido que no se jugó (por el índice)
all_data.drop(index_to_drop, inplace=True)
```

```
In [21]: # ahora debo terminar de limpiar la columna 'Score'
# usamos Regex para encontrar las filas que contienen datos no deseados
#
# Nota: el circunflejo se usa para indicar NOT y los corchetes indican REGEX
#
all_data[all_data['Score'].str.contains('[^\d-]')] # el guión lo copio del a
```

```
Out[21]:
```

	Home	Score	Away	Year
531	France	1-0 (a.e.t./g.g.)	Paraguay	1998
600	South Korea	2-1 (a.e.t./g.g.)	Italy	2002
595	Sweden	1-2 (a.e.t./g.g.)	Senegal	2002
604	Senegal	0-1 (a.e.t./g.g.)	Turkey	2002

```
In [22]: # como se ve arriba, hay 4 campos del score que contienen caracteres adicionales
# así que los elimino haciendo un reemplazo por cadena vacía
all_data['Score'] = all_data['Score'].str.replace('[^\d-]', '', regex=True)
```

Acondicionando los datos para poder procesarlos

```
In [23]: # Separo los datos numéricos de la columna 'Score' en dos nuevas columnas y
all_data[['Home Goals', 'Away Goals']] = all_data['Score'].str.split('[^\d-]')

# elimino la columna 'Score' porque ya no la necesito
all_data.drop('Score', axis=1, inplace=True)

# cambio los datos de los 'Goals' a tipo integer
all_data = all_data.astype({
    'Home Goals': int,
    'Away Goals': int,
    'Year': int # no es necesario, pero ya que está lo pongo
})

# renombro las columnas para más claridad
all_data.rename(columns={'Home': 'Home Team', 'Away': 'Away Team'}, inplace=
```

```
In [24]: # aquí veo como se cambiaron los tipos del df
all_data.dtypes
```

```
Out[24]: Home Team      object
Away Team      object
Year           int64
Home Goals     int64
Away Goals     int64
dtype: object
```

```
In [25]: # agrego una última columna con los goles totales del partido (sólo por gusto
# es interesante observar el hecho de que no podría hacerlo si no hubiera
# cambiado los tipos de datos
all_data['Total Goals'] = all_data['Home Goals'] + all_data['Away Goals']

In [26]: # reorganizo las columnas enviando al final del df a 'Year'
all_data = all_data.reindex(columns=['Home Team', 'Away Team', 'Home Goals',
```

Comprobación de los datos

```
In [27]: # Compruebo que la cantidad de datos sea correcto (cantidad de partidos por
print('Year      ', 'Matches Played')
print('-----', '-----')

for year in range(1930, 2019, 4):
    if year == 1942 or year == 1946:
        continue

    matches = all_data[all_data['Year'] == year]
    matches_played = len(matches)

    print(year, ' ' * 8, matches_played)
```

Year	Matches Played
----	-----
1930	18
1934	17
1938	18
1950	22
1954	26
1958	35
1962	32
1966	32
1970	32
1974	38
1978	38
1982	52
1986	52
1990	52
1994	52
1998	64
2002	64
2006	64
2010	64
2014	64
2018	64

Guardo los dataframes limpios en el disco
(formato csv)

```
In [28]: all_data.to_csv('files/clean_fifa_worldcup_historical_data.csv', index=False)
         fixture.to_csv('files/clean_fifa_worldcup_fixture_2022.csv', index=False)
```

Parte 5

Creación del modelo (usando la distribución de Poisson)

```
In [29]: import pandas as pd
         import pickle
         from scipy.stats import poisson
```

```
In [30]: # primero recuperamos los datos limpios guardados a disco
         historical_data = pd.read_csv('files/clean_fifa_worldcup_historical_data.csv')
         fixture = pd.read_csv('files/clean_fifa_worldcup_fixture_2022.csv')

         with open('files/groups.dat', 'rb') as file:
             tables = pickle.load(file)
```

```
In [31]: tables['Group A']
```

```
Out[31]:
```

	Pos	Team	Pld	W	D	L	GF	GA	GD	Pts
0	1	Qatar (H)	0	0	0	0	0	0	0	0
1	2	Ecuador	0	0	0	0	0	0	0	0
2	3	Senegal	0	0	0	0	0	0	0	0
3	4	Netherlands	0	0	0	0	0	0	0	0

Calcular el Team Strength (poderío del equipo)

```
In [32]: # ahora crearemos un df unificado conteniendo 3 columnas
         # 1. 'Equipo' 2. 'Goles Anotados' 3. 'Goles Recibidos'
         #
         # primero separamos y renombramos las columnas en 2 dataframes
         home_data = historical_data[['Home Team', 'Home Goals', 'Away Goals']]
         away_data = historical_data[['Away Team', 'Home Goals', 'Away Goals']]

         home_data = home_data.rename(columns={'Home Team': 'Team', 'Home Goals': 'Score'})
         away_data = away_data.rename(columns={'Away Team': 'Team', 'Away Goals': 'Score'})
```

```
In [33]: # concateno los dfs y agrupo por 'Team'
         data = pd.concat([home_data, away_data], ignore_index=True)

         # calculo el promedio de goles de c/u
         team_strength = data.groupby('Team').mean()
```


In [34]: team_strength




Out[34]:

	Scored Goals	Conceded Goals
Team		
Algeria	1.000000	1.461538
Angola	0.333333	0.666667
Argentina	1.691358	1.148148
Australia	0.812500	1.937500
Austria	1.482759	1.620690
...
Uruguay	1.553571	1.321429
Wales	0.800000	0.800000
West Germany	2.112903	1.241935
Yugoslavia	1.666667	1.272727
Zaire	0.000000	4.666667

85 rows × 2 columns

Usando Poisson

Cosideraciones sobre la distribución de poisson

 Descripción de la distribución de Poisson  Condiciones para usar Poisson
 Fórmula de Poisson

Function predict_points

```
In [35]: def predict_points(home, away):  
    if home in team_strength.index and away in team_strength.index:  
        # scored_goals * conceded_goals  
        home_lamb = team_strength.at[home, 'Scored Goals'] * team_strength.a  
        away_lamb = team_strength.at[away, 'Scored Goals'] * team_strength.a  
  
        prob_home, prob_away, prob_draw = 0, 0, 0  
        for home_goals in range(0, 11):  
            for away_goals in range(0,11):  
                probability = poisson.pmf(home_goals, home_lamb) * poisson.p  
  
                if home_goals > away_goals:
```

```

        prob_home += probability
    elif home_goals < away_goals:
        prob_away += probability
    else:
        prob_draw += probability

    home_points = 3 * prob_home + prob_draw
    away_points = 3 * prob_away + prob_draw

    return (home_points, away_points)

else:
    return (0, 0)

```

```
In [36]: predict_points('Argentina', 'Mexico')
```

```
Out[36]: (2.3129151525530505, 0.5378377125059863)
```

Prediciendo el Mundial 2022

Fase de grupo

```
In [37]: # divido el fixture en grupo, octavos, cuartos, etc
fixture_group_48 = fixture[:48].copy()
fixture_knockout = fixture[48:56].copy()
fixture_quarter = fixture[56:60].copy()
fixture_semi = fixture[60:62].copy()
fixture_final = fixture[62:].copy()
```

```
In [38]: # un ejemplo de la fragmentación del fixture (en este caso los cuartos de fi
fixture_quarter
```

```
Out[38]:
```

	Home	Score	Away	Year
56	Winners Match 53	Match 58	Winners Match 54	2022
57	Winners Match 49	Match 57	Winners Match 50	2022
58	Winners Match 55	Match 60	Winners Match 56	2022
59	Winners Match 51	Match 59	Winners Match 52	2022

```
In [39]: # interesante observar que esto me devuelve un ndarray (numpy)
# los nombres de la columna 'Team' del 'grupo A'
tables['Group A']['Team'].values
```

```
Out[39]: array(['Qatar (H)', 'Ecuador', 'Senegal', 'Netherlands'], dtype=object)
```

Esto es otro error en el curso que no fue tomado en cuenta (si bien no afectaría el resultado final, ya que Qatar no es tomado en cuenta en las predicciones por no tener data histórica)

```
In [40]: # de lo anterior observamos que hay discrepancias de nombres entre
# las tablas de Grupos y el Fixture. Específicamente con el nombre
# 'Qatar' vs 'Qatar (H)' (fixture vs grupos)
#
# por tanto corregimos el nombre en el fixture (forma fácil)
fixture_group_48.replace({'Qatar': 'Qatar (H)'}, inplace=True)

# por tanto corregimos el nombre en el fixture (forma complicada)
# for index, row in fixture_group_48.iterrows():
#     if row['Home'] == 'Qatar':
#         fixture_group_48.loc[index, 'Home'] = 'Qatar (H)'

#     if row['Away'] == 'Qatar':
#         fixture_group_48.loc[index, 'Away'] = 'Qatar (H)'

In [41]: # predigo todos los partidos de la fase de grupo y actualizo sus respectivas
for group in tables:
    group_table = tables[group]['Team'].values
    group_matches = fixture_group_48[fixture_group_48['Home'].isin(group_table)]

    for index, row in group_matches.iterrows():
        home, away = row['Home'], row['Away']

        home_points, away_points = predict_points(home, away)

        tables[group]['Pts'] = tables[group]['Pts'].astype(float)
        tables[group].loc[tables[group]['Team'] == home, 'Pts'] += home_points
        tables[group].loc[tables[group]['Team'] == away, 'Pts'] += away_points

    tables[group] = tables[group].sort_values('Pts', ascending=False).reset_index()
    tables[group] = tables[group][['Team', 'Pts']]

    tables[group] = tables[group].round(0)
    tables[group]['Pts'] = tables[group]['Pts'].astype(int)

In [42]: # así quedarían las tablas de puntos (diccionario)
tables['Group A']
```

```
Out[42]:
```

	Team	Pts
0	Netherlands	4
1	Senegal	2
2	Ecuador	2
3	Qatar (H)	0

Octavos

```
In [43]: fixture_knockout
```

```
Out[43]:
```

	Home	Score	Away	Year
48	Winners Group A	Match 49	Runners-up Group B	2022
49	Winners Group C	Match 50	Runners-up Group D	2022
50	Winners Group D	Match 52	Runners-up Group C	2022
51	Winners Group B	Match 51	Runners-up Group A	2022
52	Winners Group E	Match 53	Runners-up Group F	2022
53	Winners Group G	Match 54	Runners-up Group H	2022
54	Winners Group F	Match 55	Runners-up Group E	2022
55	Winners Group H	Match 56	Runners-up Group G	2022

```
In [44]: tables['Group A']
```

```
Out[44]:
```

	Team	Pts
0	Netherlands	4
1	Senegal	2
2	Ecuador	2
3	Qatar (H)	0

```
In [45]: # actualizo el fixture de octavos de final
for group in tables:
    winner = tables[group].loc[0, 'Team']
    runners_up = tables[group].loc[1, 'Team']

    fixture_knockout.replace({
        f'Winners {group}': winner,
        f'Runners-up {group}': runners_up
    }, inplace=True)

fixture_knockout
```

Out[45]:

	Home	Score	Away	Year
48	Netherlands	Match 49	Wales	2022
49	Argentina	Match 50	Denmark	2022
50	France	Match 52	Poland	2022
51	England	Match 51	Senegal	2022
52	Germany	Match 53	Belgium	2022
53	Brazil	Match 54	Uruguay	2022
54	Croatia	Match 55	Spain	2022
55	Portugal	Match 56	Switzerland	2022

In [46]: *# función para obtener los ganadores de los octavos de final*

```
def get_winner(fix):  
    for index, row in fix.iterrows():  
        home, away = row['Home'], row['Away']  
  
        home_pts, away_pts = predict_points(home, away)  
  
        if home_pts > away_pts:  
            fix.loc[index, 'Winner'] = home  
        else:  
            fix.loc[index, 'Winner'] = away
```

In [47]: `get_winner(fixture_knockout)`
`fixture_knockout`

Out[47]:

	Home	Score	Away	Year	Winner
48	Netherlands	Match 49	Wales	2022	Netherlands
49	Argentina	Match 50	Denmark	2022	Argentina
50	France	Match 52	Poland	2022	France
51	England	Match 51	Senegal	2022	England
52	Germany	Match 53	Belgium	2022	Germany
53	Brazil	Match 54	Uruguay	2022	Brazil
54	Croatia	Match 55	Spain	2022	Spain
55	Portugal	Match 56	Switzerland	2022	Portugal

Cuartos de final

In [48]: `fixture_quarter`

Out[48]:

	Home	Score	Away	Year
56	Winners Match 53	Match 58	Winners Match 54	2022
57	Winners Match 49	Match 57	Winners Match 50	2022
58	Winners Match 55	Match 60	Winners Match 56	2022
59	Winners Match 51	Match 59	Winners Match 52	2022

```
In [49]: def resolve_fixture(complete_fixture, unresolved_fixture):
        for idx, row in complete_fixture.iterrows():
            match, winner = row['Score'], row['Winner']

            unresolved_fixture.replace({f'Winners {match}': winner}, inplace=True)
```

```
In [50]: resolve_fixture(fixture_knockout, fixture_quarter)
get_winner(fixture_quarter)

fixture_quarter
```

Out[50]:

	Home	Score	Away	Year	Winner
56	Germany	Match 58	Brazil	2022	Brazil
57	Netherlands	Match 57	Argentina	2022	Netherlands
58	Spain	Match 60	Portugal	2022	Portugal
59	England	Match 59	France	2022	France

```
In [51]: resolve_fixture(fixture_quarter, fixture_semi)
get_winner(fixture_semi)

fixture_semi
```

Out[51]:

	Home	Score	Away	Year	Winner
60	Netherlands	Match 61	Brazil	2022	Brazil
61	France	Match 62	Portugal	2022	France

```
In [52]: # obtengo el ganador (no resuelvo el segundo y tercer lugar)
resolve_fixture(fixture_semi, fixture_final)

# para no tener que armar una función solo para modificar una línea del df
# para los tercero y cuarto puesto, lo modifico manualmente
fixture_final.replace({
    'Losers Match 61': 'Netherlands',
    'Losers Match 62': 'Portugal'
}, inplace=True)

get_winner(fixture_final)
```

```
fixture_final
```

Out[52]:

	Home	Score	Away	Year	Winner
62	Netherlands	Match 63	Portugal	2022	Netherlands
63	Brazil	Match 64	France	2022	Brazil

Final positions Worldcup 2022 (predicción)

1. **Brazil**
2. **France**
3. **Netherlands**
4. **Portugal**