Tipología y Ciclo de Vida de Datos

Web Scraping La Liga 19/20

Práctica 1

Alfonso Jiménez Hernández y Jorge Martín de la Calle

12-4-2020

Contexto

En este proyecto se ha decidido investigar sobre los datos que se generan a través del fútbol concretamente de la Liga Española de Fútbol profesional. Actualmente, con la aparición del Big Data se ha empezado a recopilar información para ser más competitivo, comprar jugadores más barato y explotar su rendimiento a un coste menor. Unido a esto encontramos un ejemplo visual que se observa en la película Moneyball. Esta película trata de un gerente general de un equipo de beisbol que con ayuda de un economista usa las estadísticas para fichar a jugadores para lograr un equipo competitivo.

Viendo cómo se encuentra el panorama actual y hacia dónde van los equipos de futbol en el ámbito de los datos, se quiere observar basándonos en estadísticas recogidas durante este año si se pueden sacar alguna conclusión interesante.

Para este trabajo se ha elegido la página https://www.sofascore.com/es/ que recoge resultados de futbol de todo el mundo, además de clasificaciones y estadísticas. Recoge la información de una manera clara y precisa.

Título para el dataset

El título para el dataset es **Stats_Players_LaLiga_19_20** ya que con estas tres palabras describimos lo suficiente para saber los registros que contiene y además de la brevedad del título.

Descripción del dataset

Tal y como se expresa en el nombre del dataset, está basado en la recopilación de las estadísticas más importantes que se tienen acerca de todos los futbolistas de la liga española de futbol del año actual 19/20. En el dataset se representa cada futbolista como registro único. Las unidades de las diferentes características se comentan en el apartado inferior según el caso.

Los datos, al ser extraídos de un proceso de web Scraping no han pasado un proceso de preprocesado o limpieza y puede existir valores que no se correspondan a sus atributos. También pueden venir valores no informados con el valor None, ‘?’ o vacío.

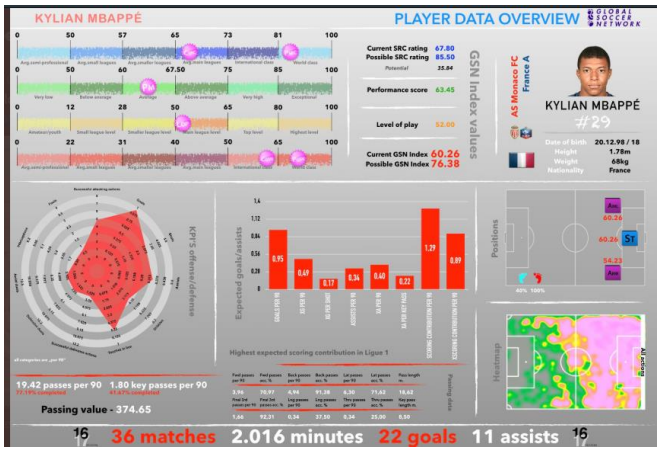
Representación gráfica

Comentar, que la imagen identificativa se muestra en portada.

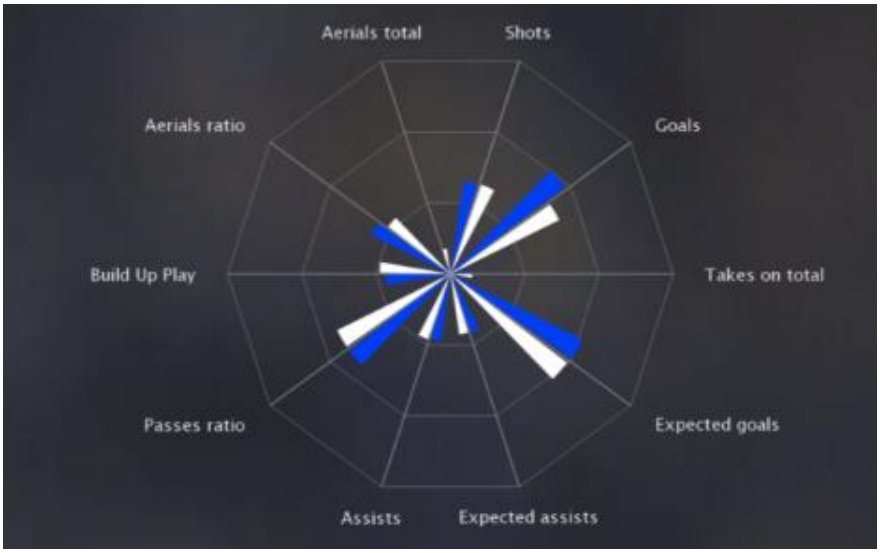
El esquema que identifica a nuestro dataset y su utilidad se compone de los siguientes pasos.

1º Existe un partido de fútbol donde se recogen todas las estadísticas por jugador

2º Estas estadísticas sufren procesamiento, para visualizarlas de una manera correcta al ojo humano



3º Posteriormente mediante análisis predictivo basado en el rendimiento obtienen jugadores de un valor de mercado menor, pero con rendimiento similar.



4º Jugadores similares a Kylian Mbappe que serían menos costosos y con un rendimiento similar.

Contenido del dataset

Los datos fueron extraídos utilizando Python sobre diferentes páginas que se encuentran en Sofascore, primero se extrajo de la página de cada equipo las diferentes páginas individuales de los futbolistas de manera automática por equipo. De la página individual del futbolista obtenemos la información de los atributos dos al nueve. Posteriormente, como hemos extraído el id del futbolista de la página del equipo al que pertenece, recorremos el JSON individual que la página tiene por futbolista para recoger las estadísticas de manera automática. (La versión extensa del web scraping se encuentra en el siguiente apartado)

Los datos que se recopilan de la temporada actual, 19/20, con los equipos que disputan la competición. Por tanto, los dato serían validos hasta septiembre de 2020 cuando se inicie la próxima temporada. Los datos se han recolectado en el parón de partidos que ha habido en la Liga a consecuencia del coronavirus. En caso de reanudarse, se actualizaría automáticamente.

En el dataset se presentan las estadísticas más importantes de los futbolistas de primera división de la liga de futbol cuyos atributos son:

1. **Name:** Nombre del futbolista analizado (Primary Key)
2. **Team:** Equipo del futbolista analizado
3. **Nac.:** Nacionalidad del futbolista
4. **Age:** Edad del futbolista cuando se realiza el Web Scraping
5. **Height:** Estatura del futbolista
6. **Preffoot:** Pie dominante del futbolista, este valor puede ser Left, Right o Both
7. **Position:** Posición que ocupa en el terreno de juego, se define con una letra, G, D, M o F.
8. **Number:** Número que lleva el futbolista en su camiseta
9. **ValuePlayer:** Valor del futbolista actualmente, puede estar expresado en Millones o Miles de euros,
10. **matchesTotal:** Numero de partidos disputados por el futbolista hasta la fecha
11. **matchesStarting:** Partidos comenzados de titular
12. **minutesPerGame:** Minutos por partido
13. **goals:** Goles
14. **goalsFrequency: Frecuencia** de goles
15. **goalsAverage** Goles por partido
16. **totalShotsPerGame**: Tiros a puerta por partido
17. **bigChanceMissed:** Ocasiones claras falladas
18. **assists:** Asistencias, pase que acaba en gol
19. **touches:** Número de veces de toca el balón
20. **bigChanceCreated:** Gran ocasiones de gol creadas

21. **keyPasses:** Pases clave
22. **accuratePassesPerGame:** Pases completados por partido
23. **successfulPassesOwnHalf:** Pases completados en campo propio
24. **successfulPassesOppositionHalf**: Pases completados en campo contrario
25. **successfulLongPasses:** Pases largo completados
26. **accurateChippedPasses: Centros acertados**
27. **successfulCrossesAndCorners:** Saques de esquina con éxito, que tienen remate
28. **interceptionsPerGame:** Intercepciones por partido
29. **tacklesPerGame:** Entradas por partido
30. **possessionWonFinalThird:** Posesión ganada
31. **challengesLostPerGame:** Veces regateado por partido
32. **totalClearancesPerGame:** Despejes por partido
33. **errorLeadToAShot:** Error que conlleva a disparo
34. **errorLeadToaGoal:** Error que conlleva a tiro
35. **penaltiesConceded:** Penalti cometido
36. **successfulDribblesPerGame:** Duelos totales ganados
37. **duelsWonPerGame:** Duelos ganados por partido
38. **groundDuelsWonPerGame:** Duelos ganados por el suelo por partido
39. **aerialDuelsWonPerGame:** Duelos aéreos ganados por partido
40. **possessionLost:** Posesiones perdidas
41. **fouls:** Faltas realizadas
42. **wasFouled:** Faltas recibidas
43. **offsides:** Fueras de juego
44. **yellowCards:** Tarjetas Amarillas totales
45. **yellowRedCards:** Segunda tarjeta en el partido que conlleva a expulsión
46. **redCards:** Roja directa

Extracción del dataset

En esta sección se profundiza sobre la técnica realizada de Web Scraping.

Antes de realizar cualquier scraping es importante realizar una revisión al archivo robots.txt

```
User-agent: *
Disallow: /ajax/
Disallow: /search/
Disallow: /standings/
Disallow: /tournament/
Disallow: /betting-tips-today/
Disallow: /hr/ajax/
Disallow: /hr/pretraga/
Disallow: /hr/tablica/
Disallow: /hr/turnir/
Disallow: /hr/savjeti-za-kladjenje-danas/
Disallow: /it/ajax/
Disallow: /it/cerca/
Disallow: /it/classifiche/
Disallow: /it/torneo/
Disallow: /it/consigli-per-le-scommesse-oggi/
Disallow: /es/ajax/
Disallow: /es/buscar/
Disallow: /es/clasificacion/
Disallow: /es/torneo/
Disallow: /es/consejos-apuestas-hoy/
Disallow: /pt/ajax/
```

Este es el robots.txt donde sé que cualquier agente usuario está permitido, posteriormente vemos que tiene un directorio divido por país donde se observa que URL concretas no se pueden acceder, rastrear o indexar.

Para el scraping hemos utilizado las siguientes librerías:

```
1  import urllib.request
2  from bs4 import BeautifulSoup
3  import time
4  import pandas as pd
5  import requests
6  import json
```

urllib.request: Sirve para abrir las URL

BeautifulSoup: es la librería por excelencia de scraping, es la encargada para analizar los HTML de las páginas.

Time: Para crear los diferentes retardos en durante las diferentes peticiones para evitar bloqueos

Request: para usar el protocolo HTTP

Json: para tratar con este tipo ficheros

El código se ha divido en dos partes, una parte main.py y otro Scraper.py.

Para el **main.py** hemos seguido una serie de pasos:

1. Definir las cabeceras del dataset que hemos descrito anteriormente. Definir los diferentes equipos que componen la primera división. Para extraer los enlaces de primera división se podrían hacer más complejo el scraper y buscar estos

enlaces. En este caso hemos definido los diferentes equipos con sus respectivos links.

```
headers = ['Name','Team','Nac.','Age','Height','Preffoot','Position','Number','ValuePlayer','matchesTotal', 'matchesStarting',
           'minutesPerGame', 'goals', 'goalsFrequency', 'goalsAverage', 'totalShotsPerGame','bigChanceMissed','assists','touches',
           'bigChanceCreated', 'keyPasses','accuratePassesPerGame','successfulPassesOwnHalf','successfulPassesOppositionHalf','successfull
           'accurateChippedPasses','successfulCrossesAndCorners','interceptionsPerGame','tacklesPerGame','possessionWonFinalThird',
           'challengesLostPerGame','totalClearancesPerGame','errorLeadToAShot','errorLeadToaGoal','penaltiesConceded',
           'successfulDribblesPerGame','duelsWonPerGame','groundDuelsWonPerGame','aerialDuelsWonPerGame','possessionLost','fouls',
           'wasFouled','offsides','yellowCards','yellowRedCards','redCards']

team_links = {1:['Athletic Bilbao', 'https://www.sofascore.com/team/football/athletic-bilbao/2825'],
              2:['Atlético Madrid', 'https://www.sofascore.com/team/football/atletico-madrid/2836'],
              3:['Deportivo Alavés', 'https://www.sofascore.com/team/football/deportivo-alaves/2885'],
```

2. Posteriormente en el primer sraper extraemos de la página de cada equipo los diferentes links de los jugadores que servirán para luego :

```
for valor_dic_equipo in range(len(team_links)):
    print('Extrayendo IDs de los jugadores del ' + team_links[valor_dic_equipo+1][0] + '...')
    time.sleep(0.5)
    req = requests.get(team_links[valor_dic_equipo+1][1])
    html = BeautifulSoup(req.text, "html.parser")
```

3. Una vez realizada la extracción del id del jugador creamos el enlace para cada jugador y los añadimos a su nombre mediante un diccionario. Con estos enlaces y el nombre de los jugadores del equipo llamamos al fichero Scraper_player.py que se encargará de sacar la información de los diferentes jugadores. Esta función retorna una fila por jugador con sus atributos que se añaden a la lista que posteriormente constituira el dataframe final.

```
diccionario_id_jugadores = {}
for i in lista_id_jugadores:
    diccionario_id_jugadores[i[0]] = i[1]

#LLamar funcion de jugadores
rowPlayer=Scraper_player.playerScraper(diccionario_id_jugadores,headers)
rows.append(rowPlayer)
```

Para el **Scraper_player.py** hemos seguido una serie de pasos:

Una vez que invoca a la función *playerScraper* donde se te realizarán dos peticiones y se realizarán a dos Scraper, uno a la página oficial del jugador y otro para realizar el scraper de las estadísticas. La función se llama una vez por equipo y devolverá el número de registros igual al número de jugadores

1. Primero se definen los dos enlaces que tienen el mismo inicio para realizar el Scraper. Posteriormente se itera un bucle por cada jugador. Llamando a la primera url que será https://www.sofascore.com/es/jugador/ideljugador, en esta página se extraer la información general del futbolista edad, nacionalidad, posición en el campo etc. Según se produzca la extracción se va añadiendo atributos al registro del futbolista con row.append(). Como se puede comprobar tanto en el main como en esta función se añade un time.sleep para que la página no bloquee la extracción.

```
def playerScraper (diccionario_id_jugadores,headers):
    rows=[]
    iniciourl='https://www.sofascore.com/es/jugador/'
    iniciourlstds='https://www.sofascore.com/player/'



    # Para 1 equipo
    for key in diccionario_id_jugadores.keys():
        row=[]
        print('Extracting %s stats...' % key)
        row.append(key)
        url=iniciourl+key+'/'+diccionario_id_jugadores[key]
        # Wait for 0.55 seconds
        time.sleep(0.55)
        #Información del jugador
        req = requests.get(url)
        html = BeautifulSoup(req.text, "html.parser")
        equipo =html.find('h3', {'class': 'styles__TeamLink-sc-1ss54tr-7 hUZGuP'}).getText()
        row.append(equipo)
        for infoplayer in html.find_all(class_="styles__DetailBoxTitle-sc-1ss54tr-5 enIhhc"):
            #print(infoplayer.text)
            row.append(infoplayer.text)
```

Esta sería la parte superior de la página del futbolista

2. Una problemática que nos encontramos es que cuando queríamos extraer el código es que para algunos jugadores había datos faltantes y eso condicionada la extracción. Un jugador completo debería tener siete atributos de información inicial, pero algunos tenían menos atributos. Introdujimos algunas casuísticas, pero debido al número de jugadores que hay en cada plantilla existirán valores faltantes o nulos como dijimos anteriormente.

   Este sería un ejemplo de un futbolista con que no tiene los siete atributos de información inicial.

   .

3. Despues de revertir la problemática causada por los desinformación del jugador, procedemos a la extracción de las estadisticas del jugador. En un primer momento pensamos que con la misma petición a la primera pagina ya que se muestra en ella, pero inspeccionando elemento no la encontrabamos la clase de las estadisticas. Mediante inspección en la pestaña de Network encontramos el recurso JSON que correspondia a las estadisticas que corresponde al segundo URL definido en la función.

{"statistics":[{"uniqueTournamentId":8,"uniqueTournamentName":"LaLiga","seasons":[{"year":"19\/20","id":24127,"statistics":[{"statisticsItems":[{"matchesTotal":"Total played"},{"matchesStarting":"Started"},{"minutesPerGame":"Minutes per game"}],"matchesTotal":"27","matchesStarting":"25","minutesPerGame":"82","groupName":"Matches"},{"statisticsItems":[{"goals":"Goals"},{"goalsFrequency":"Scoring frequency"},{"goalsAverage":"Goals per game"},{"totalShotsPerGame":"Shots per game"{"bigChanceMissed":"Big chances missed"}],"goals":"6","goalsFrequency":"367 min","goalsAverage":"0.2","totalShotsPerGame":"1.1","bigChanceMissed":"6","more":{"statisticsItems":[{"goalConversion":"Goal conversion"},{"penalties":"Penalty goals"},{"penaltiesConversion":"Penalty conversion"},{"setPieceGoals":"Free kick goals"},{"setPiecesConversion":"Free kick conversion"},{"goalsInsideBox":"From inside the box"},{"goalsOutsideBox":"From outside the box"},{"headedGoals":"Headed goals"},{"leftFootGoals":"Left foot goals"},{"rightFootGoals":"Right foot goals"},{"penaltyWon":"Penalties won"}],"goalConversion":"26%","penalties":"1\/2","penaltiesConversion":"50%","setPieceGoals":"0\/0","setPiecesConversion":"0%","goalsInsideBox":"6\/33","goalsOutsideBox":"0\/2","headedGoals":"0","leftFootGoals":"1","rightFootGoals":"5","penalty"1","groupName":"Attacking"},"groupName":"Attacking"},{"statisticsItems":[{"assists":"Assists"},{"touches":"Touches"},{"bigChanceCreated":"Big chances created"},{"keyPasses":"Key passes"},{"accuratePassesPerGame":"Accurate per game"},{"successfulPassesOwnHalf":"Acc. own half"},{"successfulPassesOppositionHalf":"Acc. opposition half"},{"successfulLongPasses":"Acc. long balls"},{"accurateChippedPasses":"Acc. chipped passes"},{"successfulCrossesAndCorners":"Acc. crosses"}],"assists":"0","touches":"32.1","bigChanceCreated":"4","keyPasses":"0.8","accuratePassesPerGame":"13.1 (75%)","successfulPassesOwnHalf":"3.8 (85%)","successfulPassesOppositionHalf":"9.7 (67%)","successfulLongPasses":"0.4

4. Procedemos a sacar las estadisticas haciendo una petición a la URL correspondiente, realizando otro time.sleep(), utilizando la librería json para crear un el objeto de diccionario y despues iterar.

```
#print(row)
time.sleep(0.77)
# Sacar estadisticas por jugador
url=iniciourlstds+diccionario_id_jugadores[key]+'/statistics/json'
r = requests.get(url)
soup = BeautifulSoup(r.content, 'lxml')
json_object = json.loads(r.content)
```

Aquí nos enfrentamos a dos causiticas, que el jugador no tenga estadisticas, es decir que aparezca en plantilla pero no haya jugado ningun partido en ese caso el JSON estará vacio. En este caso añadimos el carácter '?' para todos los atributos a partir de la posición nueve, ya que los atributos anteriores pertenecen al primer scraper realizado.

```
if len(json_object['statistics']) == 0:
    for stats in headers[9:]:
        row.append('?')
```

En caso que tenga estadisticas, recorremos el arichvo JSON buscando las estadistica para añadir al jugador. Posteriormente retornamos los registros de los jugadores que se añadirán a los totales.

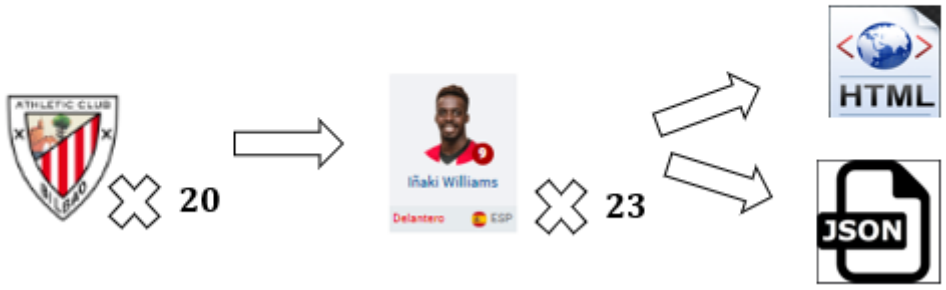
```
        else:
            for tournament in json_object['statistics'][0]['seasons'][0]['statistics']:
                for stats in tournament.keys():
                    if stats in headers:
                        #print (tournament[stats])
                        row.append(tournament[stats])


        rows.append(row)
    return rows
```

Por último en el fichero main.py despues de haber retornado los registros de los jugadores, se va creando el dataframe hasta realizar el último jugador del último equipo.

```
df = pd.DataFrame(columns=headers)
for i in range (len(rows)):
    df =df.append(pd.DataFrame(rows[i], columns=headers))

consolidated.append(df)
pd.concat(consolidated).to_csv(r'Stats_Players_LaLiga_19_20.csv', sep=',', encoding='utf-8-sig',index=False)
```

El resumen función sería:

De los 20 equipos se extraen mediante scraper los ID de los futbolistas, donde se realiza dos scraper por cada futbolista para extraer la información para crear el dataframe



Agradecimientos

Agradecer en todo momento al sitio web Sofascore que se encarga de tener los resultados del mundo del fútbol en directo de ligas de fútbol en todo el mundo, copas y torneos, además de resultados, estadísticas, clasificaciones, vídeos de los momentos destacado.

Inspiración

El interés que surge debido a los datos que se empiezan a recolectar del mundo del futbol son cada vez mayores, ya sea para sacar rendimiento de los jugadores o por diversión en juegos de simulación basados en estadísticas o puntuaciones como el Comunio[1] o biwenger [2].

Esto ha llevado en el ámbito del rendimiento de jugadores a buscar jugadores que tengan un comportamiento en el terreno de juego similar a otros jugadores con más valor de mercado, para así ser más competitivos e intentar competir de tú a tú contra rivales con mayor presupuesto.

Para esto, existen algunos artículos interesantes sobre big data en el mundo del futbol, donde empresas basándose en una serie de atributos del jugador realizan mediante análisis predictivo estas búsquedas.

El caso más conocido es el caso del Sevilla FC donde su director deportivo no busca futbolistas concretos sino busca jugadores según sus cualidades o características [3]. Dado este tipo de compras de jugadores, normalmente aparte de conseguir más competitivos logran en un futuro recibir un beneficio en caso de que el jugador sea vendido por un precio mayor al comprado. Como conclusión, el dato se usa para encontrar jugadores de edades tempranas que puedan llegar a ser grandes jugadores y para obtener beneficios en un futuro como cualquier empresa que dedique a invertir dinero.

Dadas estas premisas, cuando uno intenta buscar un datasets para investigación periodística o para elaborar modelos predictivos en función de su rendimiento y su valor de mercado, es bastante complicado, porque existen diferentes APIS de pago para conseguir estos atributos. Gracias a este dataset se podrá acceder a los datos de una manera gratuita y se podrá usar para elaborar desarrollar modelos predictivos o usarlos para ganar a tus amigos en competiciones de simulación.

Licencia

La licencia elegida es **Released Under BY_NC-SA 4.0 License** (https://creativecommons.org/licenses/by-nc-sa/4.0/). En virtud de los datos extraidos a la pagina Sofascore no podrán ser usados para obtener ningún beneficio, tienen prohibido scrapear las páginas correspondientes a apuestas. Con esta licencia logramos o conseguimos que las modificaciones del dataset lleven consigo estas condiciones.

Código

El código se encuentra en el link: https://github.com/ajimenezhdez/Practica-1-Sraping/tree/master/src

A continuación, pegamos el código:

Fichero main.py

```
import
urllib.r
equest
            from bs4 import BeautifulSoup
            import time
            import pandas as pd
            import requests
            import json
            import Scraper_player

            rows = []
            rowPlayer = []
            consolidated = []
            headers =
            ['Name','Team','Nac.','Age','Height','Preffoot','Position','Number','Va
            luePlayer','matchesTotal', 'matchesStarting',
                       'minutesPerGame', 'goals', 'goalsFrequency', 'goalsAverage',
            'totalShotsPerGame','bigChanceMissed','assists','touches',
                       'bigChanceCreated',
            'keyPasses','accuratePassesPerGame','successfulPassesOwnHalf','successf
            ulPassesOppositionHalf','successfulLongPasses',

            'accurateChippedPasses','successfulCrossesAndCorners','interceptionsPer
            Game','tacklesPerGame','possessionWonFinalThird',

            'challengesLostPerGame','totalClearancesPerGame','errorLeadToAShot','er
            rorLeadToaGoal','penaltiesConceded',
```

```
'successfulDribblesPerGame','duelsWonPerGame','groundDuelsWonPerGame','
aerialDuelsWonPerGame','possessionLost','fouls',

'wasFouled','offsides','yellowCards','yellowRedCards','redCards']

team_links = {1:['Athletic Bilbao',
'https://www.sofascore.com/team/football/athletic-bilbao/2825'],
              2:['Atlético Madrid',
'https://www.sofascore.com/team/football/atletico-madrid/2836'],
              3:['Deportivo Alavés',
'https://www.sofascore.com/team/football/deportivo-alaves/2885'],
              4:['Leganés',
'https://www.sofascore.com/team/football/leganes/2845'],
              5:['Getafe',
'https://www.sofascore.com/team/football/getafe/2859'],
              6:['Osasuna',
'https://www.sofascore.com/team/football/osasuna/2820'],
              7:['Valencia',
'https://www.sofascore.com/team/football/valencia/2828'],
              8:['Villarreal',
'https://www.sofascore.com/team/football/villarreal/2819'],
              9:['Granada',
'https://www.sofascore.com/team/football/granada/33779'],
              10:['Barcelona',
'https://www.sofascore.com/team/football/barcelona/2817'],
              11:['Sevilla',
'https://www.sofascore.com/team/football/sevilla/2833'],
              12:['Real Valladolid',
'https://www.sofascore.com/team/football/real-valladolid/2831'],
              13:['Celta Vigo',
'https://www.sofascore.com/team/football/celta-vigo/2821'],
              14:['Espanyol',
'https://www.sofascore.com/team/football/espanyol/2814'],
              15:['Real Betis',
'https://www.sofascore.com/team/football/real-betis/2816'],
              16:['Real Madrid',
'https://www.sofascore.com/team/football/real-madrid/2829'],
              17:['Real Sociedad',
'https://www.sofascore.com/team/football/real-sociedad/2824'],
              18:['Eibar',
'https://www.sofascore.com/team/football/eibar/2839'],
              19:['Levante',
'https://www.sofascore.com/team/football/levante/2849'],
              20:['Mallorca',
'https://www.sofascore.com/team/football/rcd-mallorca/2826']}
```

```
for valor_dic_equipo in range(len(team_links)):
    print('Extrayendo IDs de los jugadores del ' +
team_links[valor_dic_equipo+1][0] + '...')
    time.sleep(0.5)
    req = requests.get(team_links[valor_dic_equipo+1][1])
    html = BeautifulSoup(req.text, "html.parser")

    lista_enlaces_jugadores = []
    ini_enlace_jugadores = '/player'

    for tag_jugador in html.find_all(class_="squad__player squad-player
u-tC js-show-player-modal ff-medium"):
        lista_enlaces_jugadores.append(tag_jugador.get('href'))

    lista_id_jugadores = []
    for i in lista_enlaces_jugadores:

lista_id_jugadores.append(i[(len(ini_enlace_jugadores)+1):].split('/'))

    diccionario_id_jugadores = {}
    for i in lista_id_jugadores:
        diccionario_id_jugadores[i[0]] = i[1]

    #LLamar funcion de jugadores

rowPlayer=Scraper_player.playerScraper(diccionario_id_jugadores,headers
)
    rows.append(rowPlayer)

df = pd.DataFrame(columns=headers)
for i in range (len(rows)):
    df =df.append(pd.DataFrame(rows[i], columns=headers))

consolidated.append(df)
pd.concat(consolidated).to_csv(r'Stats_Players_LaLiga_19_20.csv',
sep=',', encoding='utf-8-sig',index=False)
```

Fichero Scraper_player.py

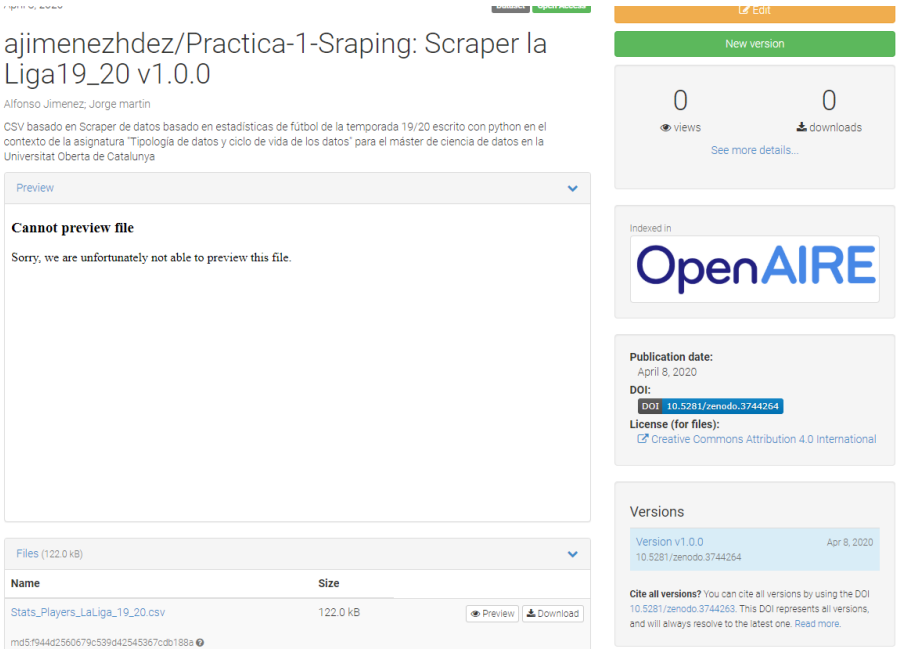
```
import
time
        import requests
        import json
        from bs4 import BeautifulSoup
```

```
def playerScraper (diccionario_id_jugadores,headers):
    rows=[]
    iniciourl='https://www.sofascore.com/es/jugador/'
    iniciourlstds='https://www.sofascore.com/player/'



    # Para 1 equipo
    for key in diccionario_id_jugadores.keys():
        row=[]
        print('Extracting %s stats...' % key)
        row.append(key)
        url=iniciourl+key+'/'+diccionario_id_jugadores[key]

        # Wait for 0.55 seconds
        time.sleep(0.55)

        #Información del jugador
        req = requests.get(url)
        html = BeautifulSoup(req.text, "html.parser")
        equipo =html.find('h3', {'class': 'styles__TeamLink-sc-1ss54tr-7
hUZGuP'}).getText()
        row.append(equipo)

        for infoplayer in html.find_all(class_="styles__DetailBoxTitle-
sc-1ss54tr-5 enIhhc"):
            if((len(row)==7 )& (len(infoplayer.text)>3)):
                row.append('None')
                row.append(infoplayer.text)

            else:
                row.append(infoplayer.text)

        #Poner none algun dato que no este, es decir de la posicion 0 a
la 8 todo tiene que estar informado.
        for i in range(9-len(row)):
            row.append('None')



        time.sleep(0.77)
        # Sacar estadisticas por jugador

url=iniciourlstds+diccionario_id_jugadores[key]+'/statistics/json'
        r = requests.get(url)
        soup = BeautifulSoup(r.content, 'lxml')
```

```
            json_object = json.loads(r.content)



            if len(json_object['statistics']) == 0:
                for stats in headers[9:]:
                    row.append('?')
            else:
                for tournament in
        json_object['statistics'][0]['seasons'][0]['statistics']:
                    for stats in tournament.keys():
                        if stats in headers:
                            #print (tournament[stats])
                            row.append(tournament[stats])


            rows.append(row)
        return rows
```

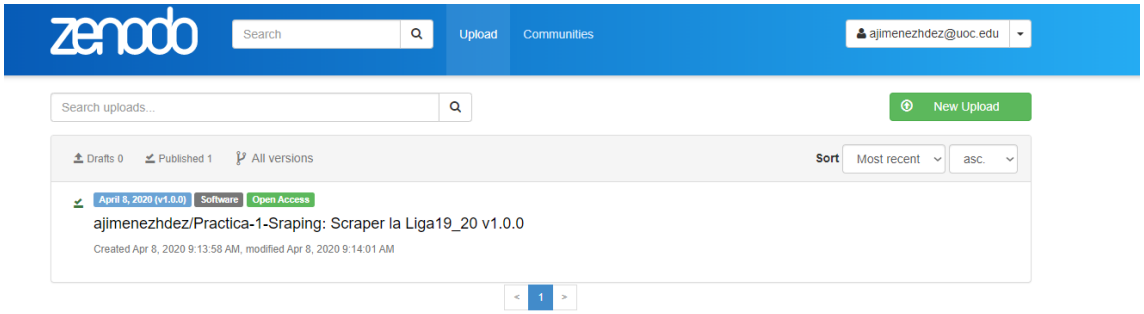
Dataset Zenodo

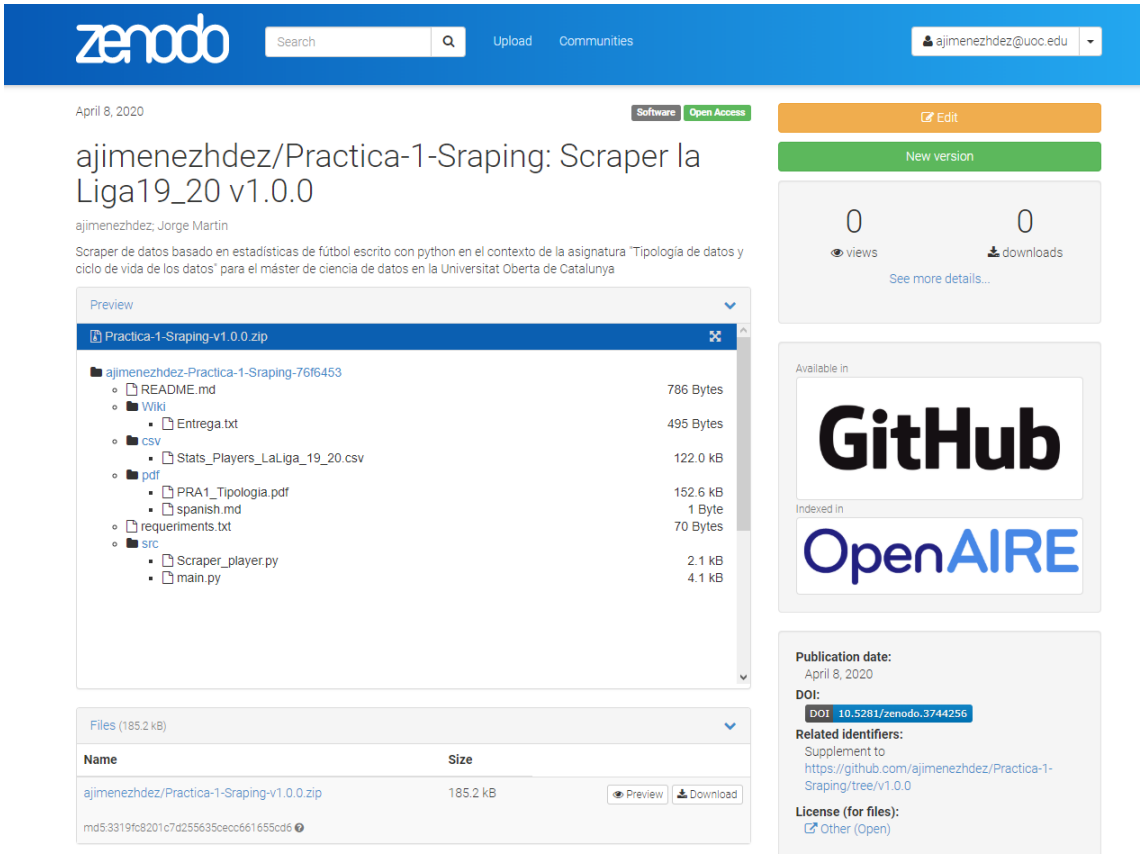
El dataset se encuentra en el enlace:
https://zenodo.org/record/3744264#.Xo2eSatR1hE



DOI Github

Enlace: https://zenodo.org/record/3744256#.Xo2etatR1hE





Bibliografía

[1] https://www.comunio.es

[2] https://www.biwenger.com

[3] https://www.elconfidencial.com/deportes/futbol/2019-10-17/entrevista-monchi-sevilla-big-data-443_2278023/

[4] Lawson, R. (2015). Web Scraping with Python

[5] Subirats, L., Calvo, M. (2018). Web Scraping. Editorial UOC

Firmas

CONTRIBUCIONES	FIRMA
INVESTIGACIÓN PREVIA	AJH, JMC
REDACCIÓN DE LAS RESPUESTAS	AJH, JMC
DESARROLLO CÓDIGO	AJH, JMC