FOOTBALL ANALYSIS

1. [**BÚSQUEDA DATASETS**](#_bof0q7yrhc59) **2**
   1. [EXPLORANDO EL NATURE DATASET](#_mejhlzdi919g) **2**
   2. [EXPLORANDO FOOTBALL-DATA](#_owc8dj9q0ptk) **7**
2. [**CREACIÓN DATASETS**](#_3cjpjtu1g2yd) **9**
   1. [CREANDO NATURE DATASET](#_emoieb2nf6j2) **9**
   2. [AÑADIMOS CUOTAS AL DATASET](#_43b0yclf0ejr) **12**
   3. [TENDENCIAS ÚLTIMOS N PARTIDOS](#_dno42oex3vbe) **14**

## BÚSQUEDA DATASETS

Hasta ahora en el proyecto he hecho uso de dos Bases de Datos gratuitas encontradas en Internet:

* Dataset público usado para un estudio en la revista Nature[[1]](#footnote-0): contiene datos más específicos y concretos de cada partido, incluyendo estadísticas de los jugadores, eventos en el partido, etc.
* Football-Data[[2]](#footnote-1): una web que tiene publicados datos como estadísticas básicas sobre los partidos y especialmente las cuotas de las casas de apuestas de muchas ligas de todo el mundo.

En el punto en el que estoy actualmente, aún no he añadido características muy complejas del Dataset de Nature.

## EXPLORANDO EL NATURE DATASET

En este dataset tenemos incluidos una temporada (2017-18) de las 5 grandes ligas (española, inglesa, italiana, alemana y francesa) y además la Copa del Mundo de 2018 y la Eurocopa de 2016. Sin embargo nos centraremos solamente en las competiciones ligueras ya que son el mismo tipo de competición.

|  |
| --- |
| **Figura 1:** tabla del contenido del Dataset Nature |

En total haremos uso de 1826 partidos ligueras, que contienen más de 3 millones de eventos y mas de 3000 jugadores.

El dataset dispone de varios elementos que se almacenan en formato JSON.

Voy a mostrar cada elemento, explicando algunos atributos importantes o confusos que uso y mostrando algún ejemplo de como se almacenan los datos:

* **Competición**
* wyId es el identificador propio de este dataset.
* type indica si la competición es de clubes o de selecciones.

|  |
| --- |
| **Figura 2:** ejemplo archivo JSON de competición |

* **Partidos (Matches)**
* teamsData contiene muchos subcampos que describen la información de cada equipo que juega el partido, por ejemplo si tiene formación, los goles marcados en el partido, si juega de local o visitante.

Los equipos se identifican por el identificador de la DB.

* En el subcampo de Formation (si hasFormation = 1) tenemos tanto los jugadores titulares, como los sustitutos y el resto banquillo.

|  |
| --- |
| **Figura 3:** ejemplo archivo JSON de un partido |

|  |
| --- |
| **Figura 4:** ejemplo archivo JSON del campo formación de un equipo en un partido |

* **Equipos** (Teams)

|  |
| --- |
| **Figura 5:** ejemplo archivo JSON del equipo Juventus |

* **Evento**
* id identificador de cada evento
* eventId el identificador del tipo de evento
* eventName nombre del tipo de evento
* subEventId y subEventName
* eventSec tiempo de la parte en el cual ocurre el evento
* matchPeriod parte del partido en el que ocurre (1H/2H,E1,P…)

|  |
| --- |
| **Figura 6:** tipos y subtipos de eventos |

|  |
| --- |
| **Figura 7:** JSON evento |

* **Jugador** (Player) (Figura 6)
* role la posición, el rol que juega en el campo

|  |
| --- |
| **Figura 8:** ejemplo archivo JSON del jugador Lionel Messi |

## 

## EXPLORANDO FOOTBALL-DATA

Cada temporada de cada liga tiene su propio archivo CSV que siguen el mismo patrón de nombre: PaísDivision\_Temporada, siendo País las siglas del país, p.e. SP para España; Division si se trata de 1ª seria 1; y Temporada p.e. 1516 para la 2015-16; el nombre del fichero de este ejemplo sería SP1\_1516.csv.

Es importante destacar que las características, es decir las columnas, de cada país varían un poco ya que las casas de apuestas cambian de un país a otro. Sin embargo nosotros vamos a ocuparnos de momento en una casa de apuestas que está presente en todas las ligas: BET365.

Un ejemplo de dataset de Football-Data es el de la siguiente figura:

|  |
| --- |
| **Figura 9:** ejemplo Football-Data liga española 2017-18 |

Realmente hay muchas más características en el dataset pero aquí muestro sobretodo las que he usado y me parecen más importante.

Tenemos características fundamentales de los partidos como **FTHG,** **FTAG** o **FTR** que nos indican los goles anotados por el Local, Visitante y quien ha ganado respectivamente, si el local (H), el visitante (A) o ha acabado empate (D). También tenemos los goles al descanso (**HTHG** y **HTAG**).

A parte tenemos también otras características muy interesantes, como las faltas cometidas por los equipos (**HF** y **AF**), los disparos (**HS** y **AS),** disparos a puerta, al palo, corners, fueras de juego, tarjetas,etc.

Estas características aparecen en todas las ligas y temporadas que he empleado hasta ahora. Para ver con más detalle la definición de cada característica recomiendo leer el archivo *notes.txt[[3]](#footnote-2)* que está disponible para descargar en la web del dataset.

Las siguientes características que aparecen son las de las cuotas de las Casas de Apuestas, y como he dicho anteriormente estas casas pueden ir cambiando dependiendo el país, aunque algunas estan presentes en todas o la mayoria de ligas y temporadas. Ese es el caso de BET365, y por ello he utilizado sus cuotas (**B365H** y **B365A**).

Es importante remarcar que no solo tenemos disponibles cuotas simples, sino que disponemos de diferentes tipos de cuotas, como la del Handicap Asiático, cuotas a goles marcados, etc.

## **CREACIÓN DATASETS**

Los datasets que creemos serán, por lo general, fusiones de diversos datasets.

## CREANDO NATURE DATASET

Hasta ahora, hemos usado el Dataset de Nature como base para futuros Datasets más grandes y completos. Pero para ello había de tratar y fusionar los distintos JSON para poder manejar los datos en un DataFrame de Pandas.

Por ello, la primera base de datos que he implementado ha sido para unir y organizar los JSON que almacenan los partidos dependiendo de la liga.

Este ha sido el proceso:

Importamos las librerías que vamos a utilizar en la creación de los datasets.

Estas librerías las usaremos en la creación de todos los datasets.

| import os import pandas as pd import numpy as np import json import difflib as dl |
| --- |

Importamos los JSONs de los partidos y los almacenamos en un diccionario donde las claves son las competiciones.

| *# loading the match data* matches={} nations = ['Italy','England','Germany','France','Spain'] for nation in nations:  with open('./matches/matches\_%s.json' %nation) as json\_data:  matches[nation] = json.load(json\_data) |
| --- |

Concatenamos todos los partidos en un array plano. El resultado será un array de JSONs, donde cada JSON es un partido. Fijarse que solo seleccionamos los partidos de las Ligas.

| partidos = []  for pais in nations:  for partido in matches[pais]:  partidos.append(partido)   partidos = np.array(partidos,dtype=dict) |
| --- |

Para transformar ese array de JSONs a una tabla (dataFrame) donde podamos ver cada partido con sus atributos, transformamos cada JSON a Series (un tipo de objeto Pandas) y almacenamos los partidos (Series) en un array *match\_serie*.

Luego cogemos los atributos de un partido cualquiera (en formato Series) y creamos el DataFrame con los atributos del partido.

Al final, hemos seleccionado unos pocos atributos generales de cada partido con los que nos interesa quedarnos de momento, *raw\_db*.

| match\_serie = [pd.Series(p) for p in partidos] rawDB = pd.DataFrame(match\_serie, columns=(match\_serie[0]).index)  columnNamesDB = ['wyId','competitionId','seasonId','roundId','dateutc','winner']  *# wyId = ID del partido en el DB* rawDB = rawDB[columnNamesDB] |
| --- |

A continuación añadimos a nuestro *rawDB* los datos de cada equipo que disputa el partido.

Para añadirlos hay que tener en cuenta que los datos de cada equipo se encuentran como un objeto JSON dentro del JSON del partido, es decir dentro del atributo *teamsData*.

Dentro del atributo *teamsData* encontramos dos JSON, uno para cada equipo. El índice de cada JSON es el Id del equipo

| *# Cogemos cada equipo (JSON) de cada partido*  teamsData = [list(m['teamsData'].values()) for m in match\_serie]  *# partido x equipo*  *# Creamos las listas donde irán los equipos Locales y Visitantes* homeT = [] awayT = []  for m in teamsData:  if m[0]['side'] == 'home':  homeT.append(pd.Series(m[0]))  awayT.append(pd.Series(m[1]))  else:  homeT.append(pd.Series(m[1]))  awayT.append(pd.Series(m[0]))    *# debemos cambiar el nombre de las columnas para que no se llamen igual las Home que las Away* colH = [str(x) + '\_home' for x in homeT[0].index] colA = [str(x) + '\_away' for x in awayT[0].index]  *# Creamos dos DataFrames, cada uno tendrá los atributos de cada equipos H y A* hData = pd.DataFrame(homeT, columns=homeT[0].index) aData = pd.DataFrame(awayT, columns=awayT[0].index) hData.columns = colH aData.columns = colA  assert len(hData) == len(aData) == len(rawDB)  *# Concatenamos estos dos DataFrames que acabamos de crear al que ya habiamos creado*  rawDB = pd.concat([rawDB,hData,aData],axis=1) |
| --- |

En este momento en *rawDB* tendremos los siguientes atributos:

|  |
| --- |
| **Figura 10:** todas las columnas de rawDB |

Todos los atributos que tenemos ahora son Integers, Strings, Dates, Booleanos salvo *formation\_home/away* que contiene un JSON con la formacion del equipo, como hemos visto en [el apartado que explicaba el dataset de Nature](#_mejhlzdi919g).

Por último nos quedamos solamente con varios atributos, para crear los siguientes Datasets. Realizamos también varias comprobaciones.

| rawDB = rawDB[[  'wyId', 'competitionId', 'seasonId', 'roundId', 'dateutc', 'winner',  'teamId\_home','score\_home','scoreHT\_home','teamId\_away', 'score\_away',  'scoreHT\_away'  ]]  *# comprobamos que estan todos los partidos y que no hay ninguno repetido* assert len(rawDB) == 1826 == len(set(rawDB['wyId'])) |
| --- |

## AÑADIMOS CUOTAS AL DATASET

Después de transformar nuestro Dataset de JSONs de Nature a Pandas DataFrames, le añadimos las cuotas de [*Football-Data*](#_owc8dj9q0ptk).

Vamos a añadir las cuotas a nuestro Dataset porque tener como base las predicciones de los partidos que hacen las grandes Casas de Apuestas nos puede servir como modelo base con el que comparar futuros modelos.

Para ello importamos los CSV de la temporada 2017-18 de las 5 ligas, concatenaremos los CSV en un mismo DataFrame y nos quedaremos solamente con varios atributos:

| ['Div','Date','HomeTeam','AwayTeam','FTHG', 'FTAG', 'FTR','B365H', 'B365D', 'B365A'] |
| --- |

Ahora, el gran problema es identificar los partidos en ambas bases de datos. Es decir, tenemos que encontrar la fórmula para identificar cada partido en ambas BD.

Para ello he usado la libreria *difflib*, que com su metodo *get\_close\_matches(String,list)* que devuelve en una lista los elementos de la *list* que más se parecen al *String*.

Para encontrar la solución al problema he ido por pasos:

Antes que nada, guardamos en 2 listas los nombres de los equipos en cada BD.

El primer paso es buscar que equipos tienen el mismo nombre en ambas BD, y guardamos esos equipos en una lista llamada *iguales*, y los demás en *desiguales*. Hay que puntualizar que guardamos en esas listas los nombres de Nature, los de Football-Data los usamos para comparar.

El siguiente paso es encontrar dentro de los *desiguales* los nombres semejantes en Football-Data. Para ello usamos el metodo de la libreria *difflib*

| encontrados = [] no\_encontrados = []  for des in desiguales:  find = dl.get\_close\_matches(des,fdTeams)  *# fdTeams = nombres de Football-Data*  if len(find)>0:  encontrados.append((des,find[0]))  else:  no\_encontrados.append(des) |
| --- |

Como resultado tenemos en *encontrados* los nombres que ha visto similares en ambos BD. Los guardo como pares *(nombre\_Nature, nombre\_FD)*.

Los restantes, *no\_encontrados*, deberemos cambiarlos a mano (en total 12 equipos de 98).

Finalmente unimos todos en un diccionario *equipos* de esta forma:

| equipos = {} *# { nature\_data : football\_data } }* |
| --- |

Ahora, para poder añadir correctamente las cuotas a cada partido en la BD de Nature, tenemos que tener un diccionario con el cual las claves sean los IDs de los equipos y los valores sean el nombre del equipo, debido a que en los partidos se identifica al equipo por el ID y no por el nombre. El diccionario se crea facilmente iterando sobre el JSON de [*teams*](#_mejhlzdi919g).

Un vez hecho esto, iteramos sobre cada partido de Nature y sobre cada partido de Football-Data, buscando la correspondencia de IDs y nombres, tanto para el local como para el visitante en cada partido.

El código de las iteraciones sería tal que así

| for h,a in zip(rawDB['teamId\_home'],rawDB['teamId\_away']): *# rawDB = DB de Nature*  for i in range(1826):  p = fdataDB.loc[i] *# fdataDB = DB de Football-Data*  *# nat\_ID\_Names = { ID : nombre\_equipo } -> equivalencia ID - nombreNAT*  *# equipos -> equivalencia nombreNAT -> nombreFD*  h\_name = equipos[nat\_ID\_Names[h]]  a\_name = equipos[nat\_ID\_Names[a]]  if p['HomeTeam']==h\_name and p['AwayTeam']==a\_name:  bet\_h\_nat.append(p['B365H'])  bet\_a\_nat.append(p['B365A'])  break |
| --- |

Esta es una de las partes más costosas computacionalmente de la creación de los datasets.

Por último exportamos el Dataset a CSV, con nombre *cuotaNatDB*.

## TENDENCIAS ÚLTIMOS N PARTIDOS

En la segunda versión del Dataset de Cuotas, he decidido crear una nueva característica que he pensado que puede aportar nueva información a nuestro modelo con el fin de mejorar su eficacia.

Esta nueva característica se basa en el rendimiento del equipo en los últimos *n* partidos que ha disputado. En concreto, se basa en los puntos que ha conseguido en esos últimos *n* partidos.

La formula seria la siguiente:

siendo .

Esto nos daría dos características, dos columnas independientes entre sí que serían **tendencia\_H** y **tendencia\_A**.

Probamos con N=5, y exportamos el Dataset a CSV, con nombre *cuotasDB\_tend5*.

En un futuro, podriamos expandir esta misma idea para los goles marcados, las faltas o la formación.

**Otra versión**

Otra manera de representar esta información en nuestro dataset de una manera más compacta sería reduciendo ambas columnas a una sola.

La idea seria restar una a la otra y dependiendo el signo del resultado un equipo llegaría al partido con mejor tendencia de resultados que su contrincante.

El nuevo atributo **dif\_tend** se calcula de la siguiente manera:

De esta manera si dif\_tend < 0, el local llegaria al partido con mejores resultados en los últimos N partidos que el visitante, y si dif\_tend > 0 lo contrario. En caso de dif\_trend = 0 significaria que ambos llegan con idénticos puntos en los últimos N partidos.

1. https://www.nature.com/articles/s41597-019-0247-7 [↑](#footnote-ref-0)
2. https://www.football-data.co.uk/data.php [↑](#footnote-ref-1)
3. https://www.football-data.co.uk/notes.txt [↑](#footnote-ref-2)