



universidad  
de león



## **GRADO EN INGENIERÍA INFORMÁTICA**

Sistemas de información de gestión y Bussiness  
Intelligence

# **SISTEMA DE PRECICCIÓN DEL RESULTADO DE PARTIDOS DE FÚTBOL**

**Autor:** Julio César Ruiz Calle

**Profesor:** Enrique López González

# ÍNDICE

<u>INTRODUCCIÓN</u>	Pág. 3
<u>OBJETIVOS DEL TRABAJO</u>	Pág. 4-5
<u>HERRAMIENTAS Y MÉTODOS DE TRABAJO</u>	Pág. 6-9
<u>DESARROLLO DEL TRABAJO</u>	Pág. 10-35
<u>ANÁLISIS CRÍTICO</u>	Pág. 36-37
<u>LÍNEAS DE FUTURO</u>	Pág. 38-39
<u>CONCLUSIÓN</u>	Pág. 40-41
<u>BIBLIOGRAFÍA</u>	Pág. 42-44

# INTRODUCCIÓN

El fútbol es un deporte en el cual los resultados de los partidos son difíciles de predecir. En este trabajo usaré Python para desarrollar un sistema de resultados de partidos de fútbol usando técnicas de aprendizaje automático.

La idea principal del trabajo es que el usuario elija una liga de las cinco que se le ofrecen (Liga alemana, liga española, liga francesa, liga inglesa o liga italiana) y posteriormente elija un partido de dicha liga para que se pueda predecir su resultado. Los datos del partido que el usuario tendrá que introducir serán nombre del equipo local, el nombre del equipo visitante, la cuota del equipo local, la cuota del empate y la cuota del equipo visitante. Tras pasárselos al algoritmo de *machine learning*, este le devolverá al usuario cuál es el resultado calculado, así como la probabilidad de que ocurra.

Pese a que este modelo de predicción utiliza cuotas de casas de apuestas como una de sus características para hacer los cálculos de aprendizaje automático, no es un trabajo realizado con el fin de acabar teniendo un entramado de apuestas deportivas, sino que ha sido realizado con el fin de explorar el mundo del *machine learning* y de la predicción de resultados futbolísticos.

# OBJETIVOS DEL TRABAJO

Por supuesto, el objetivo principal que me propuse de cara a este trabajo es terminar el proyecto de predicción de resultados de fútbol para que pueda ser presentado y entregado al profesor en los plazos estipulados, que dicho proyecto tenga un funcionamiento medianamente bueno y utilice las herramientas importantes que el profesor nos ha ido recomendando durante las distintas clases que hemos tenido. Principalmente estas herramientas son: base de datos de *Neo4j* para crear el grafo de conocimiento (*knowledge graph*) y *Cypher* que es su lenguaje propio, *Python* como lenguaje de programación más importante del proyecto y un algoritmo de inteligencia artificial que sirva para resolver nuestro problema (para mi caso he elegido la regresión logística ya que es un algoritmo que permite introducirle como input varias estadísticas y es capaz de arrojar en base a ellas un resultado final).

Otro objetivo muy importante es indagar en el tema de la predicción de resultados futbolísticos mediante técnicas de aprendizaje automático. Aunque para desarrollar mi propio sistema elegí la regresión logística, otro objetivo de este trabajo es el de conocer y explorar otras técnicas de aprendizaje automático además de la antes mencionada, y conocer las ventajas y desventajas que tienen unas con respecto a las otras (aunque no las ponga en funcionamiento en mi trabajo).

No es un objetivo principal el hecho de mejorar los sistemas de predicción que ya existen, aunque siempre esté ahí la posibilidad de poder intentarlo en un futuro. Esto es debido a que no dispongo ni de los recursos ni del tiempo necesarios para llevar a cabo esa tarea. Por otra parte, sí que es un objetivo primordial el hecho de buscar otros proyectos de predicción con el fin de explorarlos y conocerlos, así como conocer las posibilidades y los problemas que tienen para poder aprender de ellos y aplicar dichos conocimientos adquiridos a mi propio proyecto.

Tiene mucha importancia también el hecho de usar una base de datos de *Neo4j*. Ya que esto conlleva utilizar una herramienta que es nueva para mí, entonces podré aprender a utilizar un lenguaje nuevo (*Cypher*) y también podré aprender a utilizar una base de datos no relacional, que es algo importante que no se suele enseñar y puede serme útil de cara al futuro.

Es también un objetivo la utilización del lenguaje de programación *Python* para muchas tareas con respecto al proyecto. Esto es porque es un lenguaje con muchísimo potencial y que de cara al futuro me puede ser muy útil. Además de utilizar dicho lenguaje, también lo es aprender y utilizar librerías de *Python* que sirven para el campo de análisis de datos como por ejemplo *NumPy*, *Pandas*, *Matplot*, etc.

# HERRAMIENTAS Y MÉTODOS DE TRABAJO

En este apartado enumeraré las herramientas que he ido utilizando durante todo el proyecto, describiré de manera breve para qué se utiliza cada una de ellas y explicaré cómo y para qué las he utilizado yo en mi propio trabajo.

**Neo4J:** Se trata de una base de datos orientada a grafos. Su primera versión fue lanzada en el año 2010, que es hace relativamente poco tiempo. Tiene un lenguaje de consultas propio llamado *Cypher* que a diferencia de los lenguajes de consultas habituales es muy intuitivo y fácil de aprender. Se diferencia de las bases de datos relacionales más convencionales en que usa grafos para representar datos que se ilustran mediante círculos y las relaciones entre ellos que se ilustran mediante líneas. A dichos datos se le pueden aplicar tanto propiedades como etiquetas, con el fin de tener más información sobre los datos con los que estemos trabajando. Se trata de una herramienta con una gran escalabilidad ya que las posibilidades de añadir más nodos y relaciones a un grafo son enormemente grandes, lo cual la convierte en una herramienta muy útil para trabajar con cantidades muy grandes de datos (*big data*).

En mi caso he utilizado esta herramienta para leer los datos ‘en bruto’ de los archivos de tipo csv (que proviene de ‘*Comma-separated values*’ en español ‘Valores separados por comas’) mediante consultas en lenguaje *Cypher* y en base a ellos crear mi base de datos con el fin de tener almacenadas todas las estadísticas necesarias para posteriormente entrenar el modelo de inteligencia artificial (este proceso está descrito detalladamente en el apartado sobre ‘Desarrollo del trabajo’). También he utilizado dicha base de datos *Neo4J* para hacer consultas a mi propia base de datos con el fin de encontrar cuáles eran las estadísticas con mayor importancia en los encuentros para saber cuáles tenía que utilizar para entrenar el modelo de predicción y cuáles no.

**Python:** Es un lenguaje de programación versátil y que puede ser ejecutado en múltiples plataformas. Permite la fácil automatización de procesos y ejecución de tareas tanto en el lado del cliente como en el servidor. También es un lenguaje genial para trabajar con grandes cantidades de datos, debido a la alta calidad de librerías de recursos que ofrece.

En mi proyecto lo he utilizado para varias tareas. En primer lugar, para conectar la base de datos de *Neo4J* con el código de *Python* y así crear los sets de datos o *dataframes*. También he utilizado *Python* para eliminar del set de datos las columnas que no iba a utilizar para entrenar mi modelo de predicción. De la misma manera eliminé con un script de *Python* las filas de los sets de datos que tenían valores nulos ya que no servían para entrenar el modelo de predicción. Después de esto escribí en *Python* el código del algoritmo de regresión logística que entrena el modelo de predicción y al que se le puede meter un *input* para que haga la predicción de algún partido. Además de todo esto, también utilicé este lenguaje para crear el servidor de la aplicación web.

**NumPy:** Se trata de una librería de *Python*. Su nombre proviene de las palabras en inglés '*Numerical Python*'. Es una herramienta muy potente ya que ofrece potentes estructuras de datos, implementando matrices y matrices multidimensionales. La fuerza de esta herramienta viene dada básicamente porque garantiza cálculos a una velocidad enorme y de una manera muy eficiente.

En mi trabajo he utilizado esta herramienta para crear matrices de manera cómoda y rápida y pasárselas al algoritmo de regresión logística para entrenar el modelo.

**Pandas:** Es otra biblioteca de *Python*. Se utiliza para crear y manipular tablas numéricas y series temporales llamadas '*Dataframes*' y en base a ellos analizar y visualizar fácilmente los datos con los que se está trabajando. Se trata de una herramienta muy optimizada en lo que a rendimiento se refiere.

En mi proyecto he usado *Pandas* para crear varios '*Dataframes*' con las estadísticas de todos los partidos de cada una de las ligas correctamente ordenados para poder visualizarlos y observar sus características correctamente.

**Sklearn:** Se trata de otra librería de *Python* la cual tiene algoritmos de regresión, clasificación, *clustering* y reducción de dimensionalidad. Contiene una gran variedad de módulos y algoritmos enfocados al aprendizaje automático.

Para mi sistema de predicción he hecho uso de esta herramienta a la hora de crear el propio modelo de predicción con algoritmo de regresión logística, entrenarlo con el 70% de los datos que tenía y validarlo con el 30% restante. De esta librería también utilicé la función que permite conocer la probabilidad de que ocurran los distintos resultados. Por supuesto, también lo he utilizado a la hora de hacer la predicción con los datos sobre el partido que quiera introducir al usuario.

**Flask:** Es un *framework* codificado en lenguaje *Python* creado con el fin de facilitar el desarrollo de aplicaciones web con *Python*. Esta herramienta incluye un servidor web de desarrollo para que de manera sencilla se pueda ejecutar dicho servidor y probar la aplicación web de manera sencilla.

Para mi proyecto he usado *Flask* para ejecutar el servidor y que se quede escuchando por las peticiones que se hacen desde el lado del cliente, es decir, que recoja los datos del partido que el usuario introduce en la página web y que ejecute el algoritmo de predicción con dichos datos y devuelva el resultado y la probabilidad del mismo para que se le muestren al usuario de nuevo en la página web.

**Javascript:** Es un lenguaje de programación que permite controlar las interacciones que hace el usuario con una página web, como por ejemplo mostrar actualizaciones de contenido, interactuar con botones, animaciones gráficas...

Lo he utilizado en mi página web para que tras rellenar los datos del partido que el usuario elija y pulse el botón 'Hecho' se desencadene la acción de que dichos datos se envíen mediante una petición 'POST' al lado del servidor. Por otra parte, también he utilizado *Javascript* para que cuando el servidor envíe la respuesta esta sea tratada y se muestre el resultado que haya sido calculado en la página web para que el usuario lo pueda ver.

**VueJS:** Se trata de un *framework* de *Javascript*. Nos permite construir interfaces de usuarios en páginas web de una manera un poco más sencilla que si usásemos *Javascript* puro. Esta herramienta también admite que se construya lo que se llama 'Single Page Applications', las cuales trabajan reescribiendo datos en una misma



página siempre que se actualizan volviendo los refrescos de página mucho más rápidos y amenos para el usuario.

Yo he utilizado *VueJS* a la hora de construir la página web con el fin de hacer este proceso más sencillo y también para que mi aplicación web sea una '*Single Page Application*' que como ya he mencionado antes es mucho más presentable ya que permite que cuando la página se refresca prácticamente no se note nada.

**Vuetify:** Se trata de un plugin para la herramienta anterior (*VueJS*) que ofrece grandes posibilidades en cuanto a estética, ya que ofrece una enorme librería de componentes muy vistosos como por ejemplo botones, *sliders*, cajas de texto, *cards*...

En mi página web he utilizado *Vuetify* en varias ocasiones. Siempre con el fin de que 'se vea más bonita' la página web. Todo el texto, las cajas de selección de los equipos, los botones y las cajas donde se introducen las cuotas del partido están construidas con *Vuetify* para darle a la página web un toque más profesional.

# DESARROLLO DEL TRABAJO

Dentro de este apartado en primer lugar haré una descripción de cómo he ido desarrollando las distintas partes del trabajo hasta llegar a la versión final que es tener un sistema de predicción de resultados de fútbol en una página web completamente funcional. En segundo lugar, haré un pequeño escenario para que se vea el funcionamiento de la aplicación con algunos ejemplos reales.

Para la primera parte, he decidido dividir el desarrollo de este proyecto en varias sub partes:

1. Obtención de los datos
2. Limpieza de los datos
3. Creación de la base de datos de grafos
4. Creación del algoritmo de aprendizaje automático
5. Creación de la página web

## 1.- Obtención de los datos

Es evidente que para realizar mi sistema de predicción de resultados de partidos de fútbol necesitaba datos, pero estos datos tenían que cumplir varias condiciones. Primeramente, necesitaba que los datos estuviesen divididos para cada partido, es decir, que cada partido fuese único en dicho set de datos. Además de esto, necesitaba que en el set de datos apareciesen las cuotas que había tenido dicho partido porque es una parte importante en el sistema de predicción ya que quería que este las tuviese en cuenta. Por supuesto, necesitaba también que la cantidad de encuentros fuese grande, ya que cuantos más tuviese para entrenar el modelo este sería más preciso. Por esta misma razón también estuve buscando un set de datos que recogiese un gran número de estadísticas sobre el partido, ya que así el entrenamiento del modelo sería aún más preciso. Por último, como quería hacer un sistema para al menos las principales ligas de Europa, necesitaba datos sobre varias ligas. Y además de todo esto necesitaba que los datos fuesen de descarga libre.

Tras buscar en varias páginas web como *Kaggle* y algunas similares al final encontré <https://www.football-data.co.uk/>. En los archivos csv de esta página cada partido ocupa una fila, tienen las cuotas de varias casas de apuestas, pude descargar un gran número de partidos (hasta 2010), recogen un gran número de estadísticas (tiros, tiros a puerta, faltas, tarjetas amarillas, tarjetas rojas, saques de esquina...) y siempre diferenciando

entre local y visitante. También aparecen muchísimas ligas para consultar (aunque a mí solo me interesaban las cinco principales ligas europeas). Y la descarga de los archivos de esta página web es gratuita.

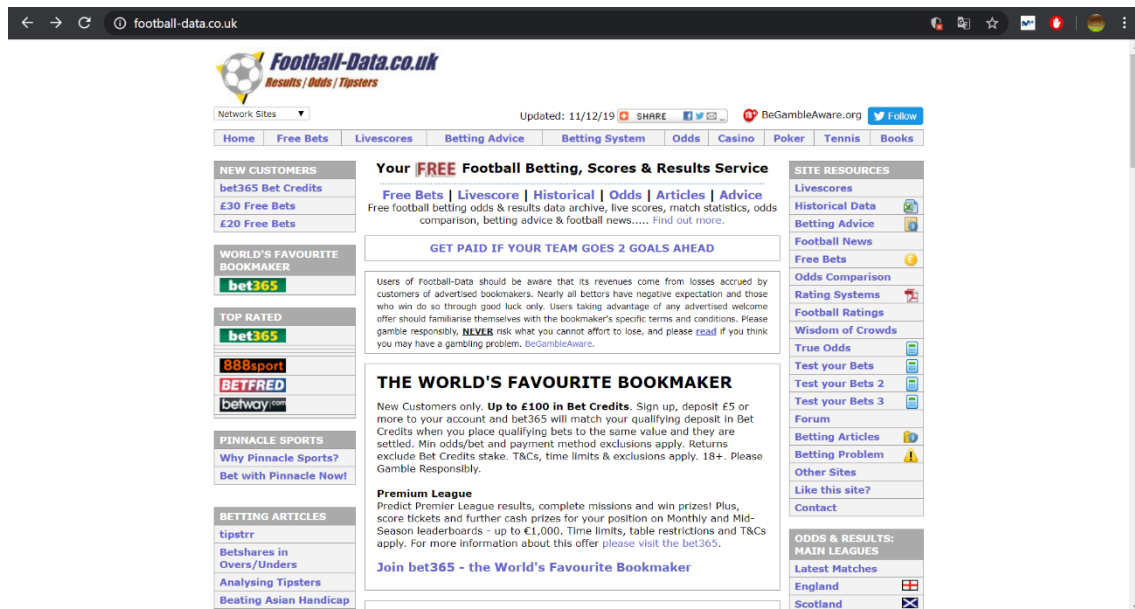


Figura 1

Por tanto, se cumplían la gran mayoría de las condiciones que había propuesto para encontrar un set de datos y poder aplicarle el algoritmo de aprendizaje automático. En la figura 2 se puede observar uno de estos archivos de tipo csv, en este caso de la liga alemana.

The image shows a screenshot of an Excel spreadsheet titled 'D1-1011.csv - Excel'. The spreadsheet contains football match data. The columns are labeled with abbreviations: Div, Date, HomeTeam, AwayTeam, FTHG, FTAG, FTR, HTHG, HTAG, HTR, HS, AS, HST, AST, HF, AF, HC, AC, HY, AY, HR, AR, B365H, B365D, B365A, BWH, BWD, BWA, GBH, GBD, GBA, JWH, JWD, JWA, LBH, LBD, LBA, SBH, SBD, SBA, W. The data rows show match details for various teams like Bayern Munich, FC Köln, Freiburg, etc., along with numerous numerical values representing different statistics.

Figura 2

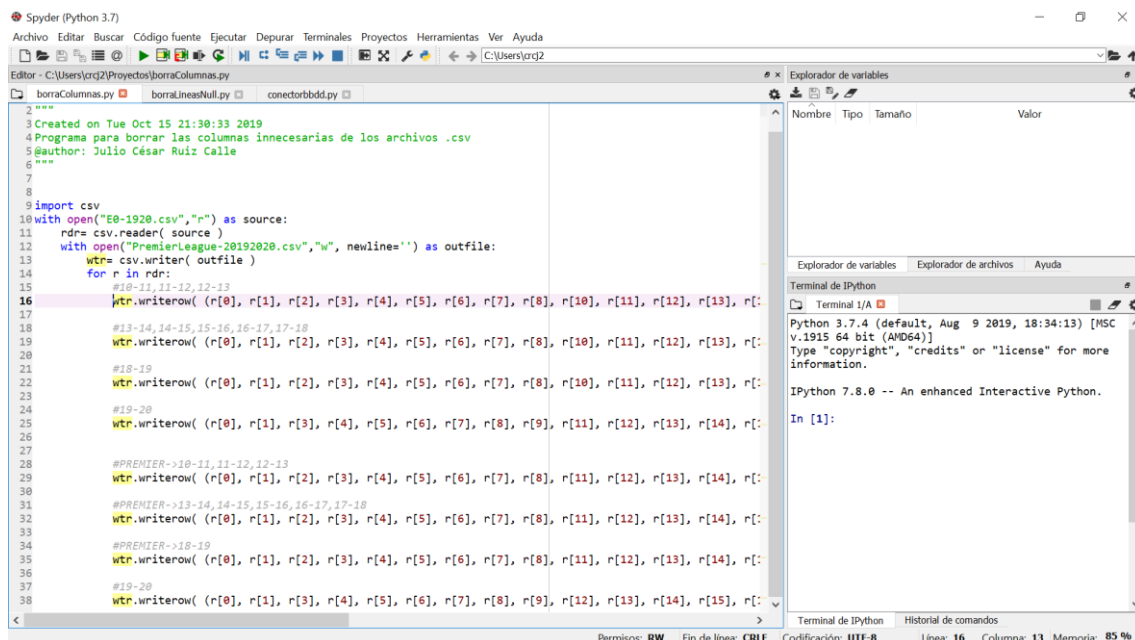
Debido a que los nombres de cada una de las columnas de los archivos son siglas, la página ofrece una guía para poder identificar qué significa cada una:

The image shows a screenshot of a webpage from football-data.co.uk. The page contains a 'Key to results data' section and a 'Match Statistics (where available)' section. The 'Key to results data' section lists abbreviations for league division, match date, time of match, home team, away team, full time home team goals, full time away team goals, full time result, half time home team goals, half time away team goals, and half time result. The 'Match Statistics (where available)' section lists abbreviations for attendance, crowd attendance, referee, home team shots, away team shots, home team shots on target, away team shots on target, home team hit woodwork, away team hit woodwork, home team corners, away team corners, home team fouls committed, away team fouls committed, home team free kicks conceded, away team free kicks conceded, home team offside, away team offside, home team yellow cards, away team yellow cards, home team red cards, away team red cards, home team bookings points, and away team bookings points.

Figura 3

## 2. Limpieza de los datos

Tras encontrar unos archivos de tipo csv con los datos que estaba buscando la siguiente tarea que tenía pendiente era la de limpiar dichos datos, es decir, eliminar todas las filas que tuviesen algún valor vacío o 'null' (ya que no servían para entrenar el modelo de predicción y además crearían conflicto) y eliminar las columnas que no me interesaba tener almacenadas en el futuro en la base de datos de grafos (ya que no las iba a utilizar para ninguna tarea). Para cumplir estas dos tareas escribí dos scripts de *Python* bastante sencillos: el primero es *borraColumnas.py*, que se puede ver en la figura 4. Lo que hace cuando es ejecutado es muy sencillo, simplemente abre el archivo csv de procedencia y sobrescribe en dicho archivo los mismos datos que haya, pero sólo de las columnas que yo le he introducido al programa, de esa manera tras ejecutar dicho script sobre todos los archivos de tipo csv conseguí tener los mismos archivos, pero sólo con las columnas que a mi realmente me interesaban. El segundo script que escribí se llama *borraLineasNull.py*, se le puede echar un vistazo en la figura 5. Este script lo que hace es que cuando se ejecuta importa la librería '*Pandas*', lee el archivo .csv de procedencia y sobrescribe en dicho archivo todos los datos que había, pero eliminando las filas a las que le faltase algún valor. Por tanto, tras ejecutar dicho script sobre todos los archivos de datos conseguí tener todos los datos sin valores vacíos. Es por eso que después de ejecutar ambos scripts contra todos los archivos de datos había cumplido mi objetivo de limpieza de datos.



```
2 """
3 Created on Tue Oct 15 21:30:33 2019
4 Programa para borrar las columnas innecesarias de los archivos .csv
5 @author: Julio César Ruiz Calle
6 """
7
8
9 import csv
10 with open("E0-1920.csv","r") as source:
11     rdr= csv.reader( source )
12     with open("PremierLeague-20192020.csv","w", newline='') as outfile:
13         wtr= csv.writer( outfile )
14         for r in rdr:
15             #10-11,11-12,12-13
16             wtr.writerow( (r[0], r[1], r[2], r[3], r[4], r[5], r[6], r[7], r[8], r[10], r[11], r[12], r[13], r[14], r[15]) )
17
18             #13-14,14-15,15-16,16-17,17-18
19             wtr.writerow( (r[0], r[1], r[2], r[3], r[4], r[5], r[6], r[7], r[8], r[10], r[11], r[12], r[13], r[14], r[15]) )
20
21             #18-19
22             wtr.writerow( (r[0], r[1], r[2], r[3], r[4], r[5], r[6], r[7], r[8], r[10], r[11], r[12], r[13], r[14], r[15]) )
23
24             #19-20
25             wtr.writerow( (r[0], r[1], r[3], r[4], r[5], r[6], r[7], r[8], r[9], r[11], r[12], r[13], r[14], r[15]) )
26
27
28             #PREMIER->10-11,11-12,12-13
29             wtr.writerow( (r[0], r[1], r[2], r[3], r[4], r[5], r[6], r[7], r[8], r[11], r[12], r[13], r[14], r[15]) )
30
31             #PREMIER->13-14,14-15,15-16,16-17,17-18
32             wtr.writerow( (r[0], r[1], r[2], r[3], r[4], r[5], r[6], r[7], r[8], r[11], r[12], r[13], r[14], r[15]) )
33
34             #PREMIER->18-19
35             wtr.writerow( (r[0], r[1], r[2], r[3], r[4], r[5], r[6], r[7], r[8], r[11], r[12], r[13], r[14], r[15]) )
36
37             #19-20
38             wtr.writerow( (r[0], r[1], r[3], r[4], r[5], r[6], r[7], r[8], r[9], r[12], r[13], r[14], r[15], r[16]) )
```

Figura 4

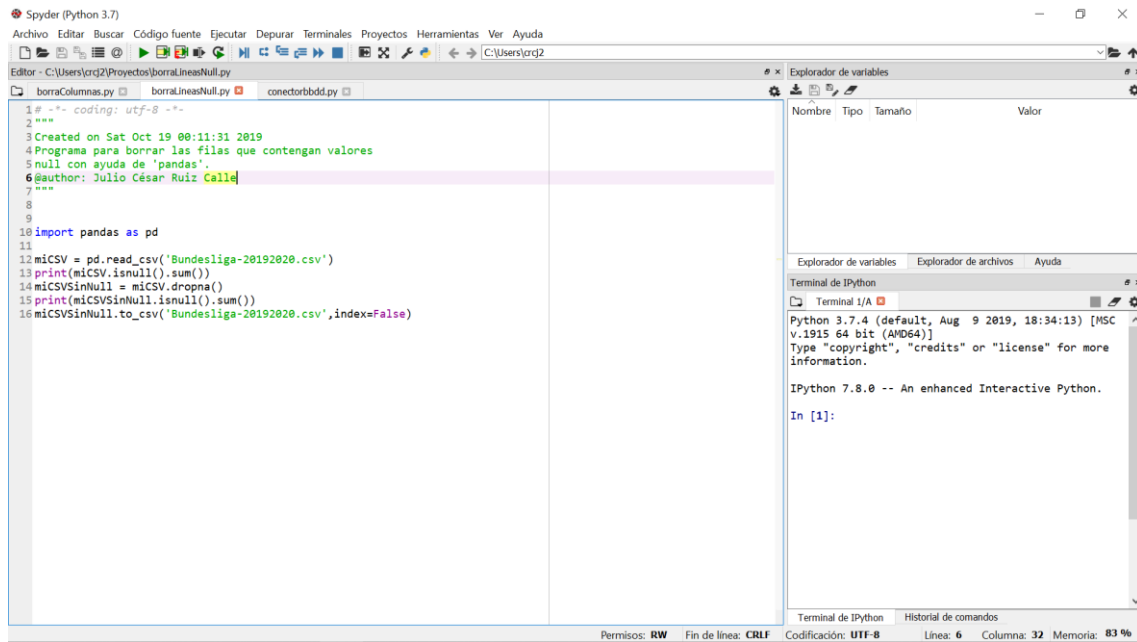


Figura 5

Tras aplicarle la limpieza a los sets de datos estos se ven mucho más pulcros ya que tienen menos columnas y menos filas. En la figura 6 podemos ver un archivo csv tras haberle aplicado los scripts para quitarle las filas y las columnas innecesarias con los scripts de *Python* borraColumnas.py y borraLineasNull.py. Si lo comparamos con el que aparece en la figura 2, que está sin limpiar, se puede apreciar la diferencia.

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
1	Div	Date	HomeTeam	AwayTeam	FTG	FTAG	FTR	HTG	HTAG	HS	AS	HST	AST	HF	AF
2	D1,20/08/10	Bayern Munich	Wolfsburg	2,1	H	1,0	17,11	5,6	25,9	3,1	3,0	0,1	57,4	0,5	1,5
3	D1,21/08/10	FC Koln	Kaiserslautern	1,3	A	1,0	10,17	4,6	10,25	5,6	1,2	1,0	2,1	3,4	4,2
4	D1,21/08/10	Freiburg	St Pauli	1,3	A	0,0	9,17	3,7	12,13	3,6	0,0	0,0	2,2	3,3	3,3
5	D1,21/08/10	Hamburg	Schalke 04	2,1	H	0,0	18,13	6,4	11,17	7,4	2,0	0,1	2,38	3,4	2,88
6	D1,21/08/10	Hannover	Ein Frankfurt	2,1	H	1,1	13,17	7,3	9,14	2,6	0,0	2,6	3,25	2,7	2,6
7	D1,21/08/10	Hoffenheim	Werder Bremen	4,1	H	4,1	10,10	7,2	24,16	5,5	1,2	0,0	2,88	3,4	2,38
8	D1,21/08/10	Freiburg	Nurnberg	1,1	D	1,1	12,10	5,5	18,6	5,1	1,0	0,1	9,1	3,4	4,0
9	D1,22/08/10	Dortmund	Leverkusen	0,2	A	0,2	14,15	6,4	20,13	8,1	0,0	0,2	38	3,4	2,88
10	D1,22/08/10	Mainz	Stuttgart	2,0	H	1,0	17,14	6,5	16,13	5,4	1,0	0,3	1,3	4,2	2,5
11	D1,27/08/10	Kaiserslautern	Bayern Munich	2,0	H	2,0	11,20	2,3	24,13	7,7	0,1	1,0	7,0	4,33	1,44
12	D1,28/08/10	Ein Frankfurt	Hamburg	1,3	A	1,0	12,14	7,4	12,12	4,6	1,0	0,3	1,3	5,2	2,3
13	D1,28/08/10	Nurnberg	Freiburg	1,2	A	1,1	17,5	4,3	12,28	4,2	1,0	0,1	9,1	3,4	4,0
14	D1,28/08/10	Schalke 04	Hannover	1,2	A	0,1	15,11	5,4	10,22	15,1	2,2	0,0	1,33	4,75	10,0
15	D1,28/08/10	St Pauli	Hoffenheim	0,1	A	0,0	12,8	4,3	16,16	5,9	2,0	0,3	2,4	2,2	3,05
16	D1,28/08/10	Werder Bremen	FC Koln	4,2	H	2,1	17,11	9,5	14,18	6,7	1,2	0,0	1,4	4,75	7,0
17	D1,28/08/10	Wolfsburg	Mainz	3,4	A	3,1	8,13	4,4	11,19	5,5	3,3	0,0	1,57	4,0	5,1
18	D1,29/08/10	Leverkusen	M'gladbach	3,6	A	1,3	12,16	6,10	12,11	4,8	1,0	0,1	36	4,75	8,0
19	D1,29/08/10	Stuttgart	Dortmund	1,3	A	0,3	11,16	5,9	16,18	7,3	0,2	0,2	2,3	3,25	2,5
20	D1,10/09/10	Hoffenheim	Schalke 04	2,0	H	1,0	23,14	8,5	18,16	3,6	1,0	0,2	4,3	3,2	8,2
21	D1,11/09/10	Bayern Munich	Werder Bremen	0,0	D	0,0	14,13	7,3	14,13	11,4	0,0	0,0	1,57	4,0	5,1
22	D1,11/09/10	Dortmund	Wolfsburg	2,0	H	0,0	20,10	1,18	23,4	0,2	3,0	0,2	1,3	5,3	2,5
23	D1,11/09/10	Freiburg	Stuttgart	2,1	H	0,1	19,18	5,8	20,17	6,7	1,1	0,0	3,4	3,2	1,3

Figura 6



### 3. Construcción de la base de datos de grafos

En este punto ya tenía los archivos de datos correctamente ‘limpiados’ en formato csv. Cada archivo correspondía a una temporada de una liga, como había decidido entrenar el modelo con diez temporadas y hacerlo para cinco ligas (liga alemana, liga española, liga italiana, liga inglesa y liga francesa) entonces tenía cincuenta archivos csv con los que construir una base de datos de grafos. En primer lugar, dediqué bastante tiempo a diseñar una especie de esquema sobre cómo sería mi base de datos, es decir, cuáles serían los nodos, las relaciones y las propiedades. Cuando ya tenía cómo sería la estructura de la base de datos de grafos, tuve que empezar a escribir las sentencias de *Cypher* necesarias para ir creando dichos nodos, relaciones y propiedades. Tras introducir todas las sentencias *Cypher* necesarias conseguí terminar mi base de datos de grafos, la cual tiene la estructura que se muestra en la figura 7.

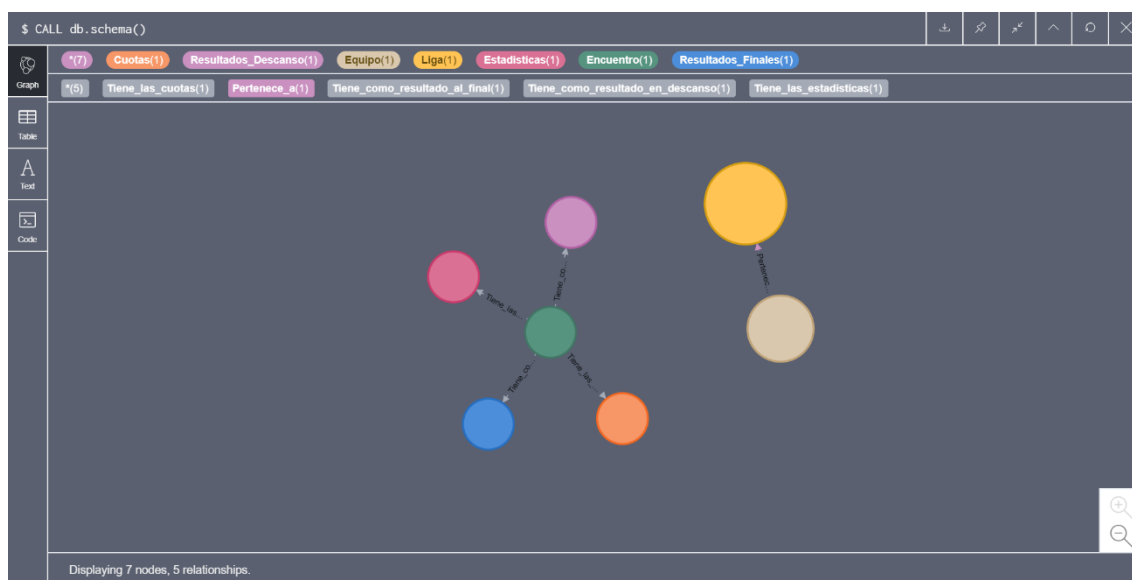


Figura 7

Como podemos observar, mi base de datos tiene dos partes:

1. La primera parte, que se ve al lado derecho son todos los equipos almacenados. Todos los equipos que hay tienen una relación llamada 'Pertenece\_a' que apunta hacia una de las cinco ligas posibles.
2. En la segunda parte, que se puede ver al lado izquierdo contiene la información acerca de todos los encuentros. El nodo central, que es el verde, es el propio encuentro y tiene cuatro relaciones.

- 2.1 La primera de ellas es 'Tiene\_como\_resultado\_en\_descanso' que apunta hacia el nodo de color morado 'Resultados\_Descanso', en el que se guarda el resultado al descanso del encuentro en cuestión: los goles que ha marcado el equipo local, los goles del equipo visitante y el resultado como tal (0=va ganando el equipo local, 1=el partido está empatado, 2=va ganando el equipo visitante)
- 2.2 La segunda de ellas es 'Tiene\_como\_resultado\_al\_final' que apunta hacia el nodo de color azul llamado 'Resultados\_Finales'. En este nodo, al igual que en el anterior, se guardan los resultados, pero en este caso al final del encuentro en cuestión: los goles que ha marcado el equipo local al final, los goles que ha marcado el equipo visitante al final y el resultado final (0=ganó el equipo local, 1=el partido acabó en empate, 2=ganó el equipo visitante)
- 2.3 La tercera relación es 'Tiene\_las\_estadisticas' que apunta hacia el nodo de color rosa llamado 'Estadisticas'. En este nodo se guardan todas las estadísticas del partido en cuestión: las faltas cometidas por el equipo local, las faltas cometidas por el equipo visitante, las tarjetas amarillas recibidas por el equipo local, las tarjetas amarillas recibidas por el equipo visitante, las tarjetas rojas recibidas por el equipo local, las tarjetas rojas recibidas por el equipo visitante, los saques de esquina lanzados por el equipo local, los saques de esquina lanzados por el equipo visitante, los tiros que hizo el equipo local, los tiros que hizo el equipo visitante, los tiros a puerta que hizo el equipo local y los tiros a puerta que hizo el equipo visitante.
- 2.4 La cuarta, que es la última de ellas es 'Tiene\_las\_cuotas' que apunta hacia el nodo de color naranja llamado 'Cuotas'. Como es evidente, en este nodo se almacenan las cuotas de las distintas casas de apuestas para cada uno de los partidos que hay almacenados en la base de datos: la cuota del equipo local, la cuota del empate y la cuota del equipo visitante. Están almacenadas las cuotas de cuatro casas de apuestas: Bet365, Bwin, William Hill e Interwetten.

Como se puede apreciar en la figura 8, mi base de datos en total tiene 96365 nodos con 67237 relaciones.



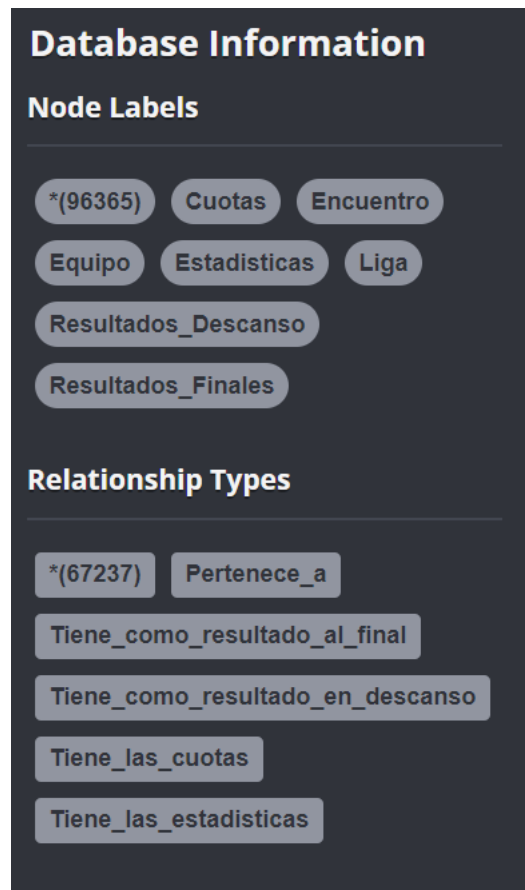


Figura 8

## 4. Creación del algoritmo de aprendizaje automático

Con la base de datos ya creada podía hacer consultas con lenguaje *Cypher* para tratar de averiguar qué estadísticas eran más significativas y tenían mayor peso para saber si incluirlas o no en el modelo de predicción. Tras meditar bastante sobre este tema decidí que lo mejor sería utilizar las siguientes estadísticas: nombre del equipo local, nombre del equipo visitante, goles del equipo local al descanso, goles del equipo visitante al descanso, resultado del partido al descanso, goles del equipo local al final, goles del equipo visitante al final, resultado del partido al final, tiros del equipo local, tiros del equipo visitante, tiros a puerta del equipo local, tiros a puerta del equipo visitante, faltas cometidas por el equipo local, faltas cometidas por el equipo visitante, tarjetas amarillas equipo local, tarjetas amarillas equipo visitante, tarjetas rojas equipo local, tarjetas rojas equipo visitante, cuota de victoria del equipo local, cuota de empate y cuota de victoria del equipo visitante.

Como el algoritmo de aprendizaje automático iba a estar escrito en *Python* necesitaba trasladar los sets de datos desde mi base de datos de grafos a una estructura de datos de *Python* para poder trabajar con ellos. Para realizar esta tarea tuve que investigar un

poco sobre cómo conectar la base de datos de grafos de *Neo4J* a un script de *Python*. No fue una tarea muy complicada ya que encontré varios ejemplos en la web sobre cómo llevarla a cabo. Simplemente había que usar la librería de *Neo4J* de *Python* y hacer una consulta con lenguaje *Cypher*, después de esto, dicha consulta devuelve los datos que se hayan seleccionado en un *dataframe* de '*Pandas*' tras ejecutar la consulta de la base de datos de *Neo4J*. En la figura 9 está una captura de dicha función.

```

1 #-*- coding: utf-8 -*-
2 """
3 Created on Mon Oct 28 17:20:07 2019
4
5 @author: crcj2
6 """
7
8 from neo4j import GraphDatabase
9 import pandas as pd
10 from warnings import simplefilter
11
12 simplefilter(action='ignore', category=FutureWarning)
13 uri = "bolt://localhost:7687"
14 driver = GraphDatabase.driver(uri, auth= ("neo4j", "pass"))
15
16 def crear_dataframe_encuentros(tx, liga):
17     Neo4j_Query = tx.run("MATCH (n:Encuentro { liga: {liga} })--(rf:Resultados_Finales) "
18     "MATCH (n:Encuentro { liga: {liga} })--(rd:Resultados_Descanso) "
19     "MATCH (n:Encuentro { liga: {liga} })--(e:Estadisticas) "
20     "MATCH (n:Encuentro { liga: {liga} })--(c:Cuentas) "
21     "RETURN n.equipo_local AS Equipo_Local, n.equipo_visitante AS Equipo_Visitante, rd.goles_descanso_local AS "
22     "+ \"Goles_Local_Descanso, rd.goles_descanso_visitante AS Goles_Visitante_Descanso, rd.resultado_al_descanso AS "
23     "+ \"Resultado_al_descanso, rf.goles_finales_local AS Goles_Local_Final, rf.goles_finales_visitante AS "
24     "+ \"Goles_Visitante_Final, rf.resultado_final AS Resultado_Final, n.fecha AS Fecha, e.tiros_local AS Tiros_Local,"
25     "+ \"e.tiros_visitante AS Tiros_Visitante, e.tiros_a_puerta_local AS Tiros_A_Puerta_Local, "
26     "+ \"e.tiros_a_puerta_visitante AS Tiros_A_Puerta_Visitante, e.faltas_local AS Faltas_Local, e.faltas_visitante "
27     "+ \"AS Faltas_Visitante, e.corners_local AS Corners_Local, e.corners_visitante AS Corners_Visitante, "
28     "+ \"e.tarjetas_amarillas_local AS Amarillas_Local, e.tarjetas_rojas_local AS Rojas_Local, "
29     "+ \"e.tarjetas_amarillas_visitante AS Amarillas_Visitante, e.tarjetas_rojas_visitante AS Rojas_Visitante, "
30     "+ \"c.Bet365_local AS Bet365Local, c.Bet365_empate AS Bet365Empate, c.Bet365_visitante AS Bet365Visitante, "
31     "+ \"c.bwin_local AS bwinLocal, c.bwin_empate AS bwinEmpate, c.bwin_visitante AS bwinVisitante, "
32     "+ \"c.Interwetten_local AS InterwettenLocal, c.Interwetten_empate AS InterwettenEmpate, c.Interwetten_visitante "
33     "+ \"AS InterwettenVisitante, c.WilliamHillLocal AS WilliamHillLocal, c.WilliamHill_empate AS WilliamHillEmpate, "
34     "+ \"c.WilliamHill_visitante AS WilliamHillVisitante ", liga=liga)
35     df = pd.DataFrame(Neo4j_Query, columns = ["Equipo_Local", "Equipo_Visitante", "Goles_Descanso_Local", "Goles_Descanso_Visitante", "Res
36     pd.set_option('display.max_columns', 33)
37     return df.drop_duplicates()

```

Figura 9

Tras tener el set de datos ya preparados al 100% la tarea que tenía pendiente era pensar qué algoritmo de aprendizaje automático o *machine learning* iba a utilizar para llevar a cabo mi sistema de predicción de resultados. Es por esto que tuve que leer e informarme sobre varios algoritmos posibles, por ejemplo: regresión logística, regresión lineal, SVM (que significa 'Máquinas de Soporte Vectorial'), K-Means, clasificador de Naive Bayes, una red neuronal...

Tras informarme sobre cómo funcionaba cada uno y ver un poco por encima cómo sería su implementación en *Python* al final me decidí por la regresión logística. Las razones principales fueron las siguientes:

1. Se trata de un algoritmo para resolver problemas de clasificación. En mi caso mi problema a resolver es clasificar cada encuentro en tres posibles resultados (0=victoria del equipo local, 1=empate, 2=victoria del equipo visitante).
2. En regresión logística hay que diferenciar entre variable dependiente, que es la que se desea predecir (en mi caso el resultado final) y variables independientes,

que son las distintas características con las que se entrena el modelo (en mi caso todas las estadísticas previamente mencionadas)

3. La regresión logística es un algoritmo relativamente simple y muy eficiente, tanto en entrenamiento como en ejecución.
4. Ofrece resultados altamente interpretables.

Ahora que ya tenía decidido el algoritmo de *machine learning* que iba a utilizar lo que me tocaba era observar y probar ejemplos sobre cómo implementar dicho algoritmo para mi caso particular. Para ello, busqué en la web varios modelos de clasificación con regresión logística y los probé en mi ordenador para observar su funcionamiento. Quiero destacar de esta parte uno de ellos, ya que gracias a su clara explicación me dejó claros la mayoría de los conceptos y aunque era bastante distinto a mi proyecto me facilitó mucho el hecho de desarrollar mi propio modelo. Este ejemplo se puede encontrar en el enlace <https://www.aprendemachinelearning.com/regresion-logistica-con-python-paso-a-paso/>. En este modelo se clasifica el sistema operativo de distintos usuarios (0=Windows, 1=Macintosh, 2=Linux) en función de cuatro características: duración de la estancia en la web, páginas leídas, acciones realizadas y valor de dichas acciones. Se trata de un ejemplo que probablemente no tenga mucha utilidad a la hora de usarlo, pero me aclaró bastante las ideas sobre como implementar la regresión logística a mi set de datos.

Después de observar varios ejemplos de modelos de predicción utilizando una regresión logística comencé a escribir en mío en mi propio script de Python. Decidí hacerlo en una función a la que se le pasan siete parámetros: el *dataframe* inicial de la liga en cuestión, el *dataframe* de entrenamiento (al que le elimino las columnas que el modelo no puede leer, como por ejemplo los nombres de los equipos y la fecha de los encuentros), el nombre del equipo local, el nombre del equipo visitante, la cuota de victoria del equipo local, la cuota de empate y la cuota de victoria del equipo visitante. Del *dataframe* de entrenamiento al final también eliminé las cuotas de todas las casas de apuestas menos bet365, es decir, eliminé Bwin, Interwetten y William Hill con el fin de simplificar el proyecto.

Para la regresión logística utilicé la librería de *Python sklearn*, por tanto, lo primero que tenía que hacer era importar dichas librerías para poder trabajar con ellas:

```
from sklearn import linear_model
from sklearn import model_selection
```

```

from sklearn.metrics import classification_report
from sklearn.metrics import confusion_matrix
from sklearn.metrics import accuracy_score

```

Con estas instrucciones creé el modelo de predicción. Como se puede observar, es aquí donde le indico que la columna que debe tratar de predecir es la del resultado final. (0=Victoria del equipo local, 1=Empate, 2=Victoria del equipo visitante)

```

X = np.array(df_entrenar.drop(['Resultado_Final'],1))
y = np.array(df_entrenar['Resultado_Final'])
X.shape
model = linear_model.LogisticRegression()
model.fit(X,y)

```

El siguiente paso tras la creación del modelo era validarlo. Para ello lo que hice fue dividir mi set de datos: el setenta por ciento lo utilicé para entrenar el modelo y el treinta por ciento restante lo utilicé para validarlo. Como en mi set de datos tengo varios años almacenados lo que hice para que el entrenamiento fuese lo más uniforme posible es desordenar todos los partidos del set de manera aleatoria. Las instrucciones que escribí fueron las siguientes:

```

predictions = model.predict(X)
model.score(X,y)
validation_size = 0.30
seed = 7
X_train, X_validation, Y_train, Y_validation =
model_selection.train_test_split(X, y, test_size=validation_size,
random_state=seed)

```

Con el fin de conocer los resultados utilicé las siguientes instrucciones:

```

name='Logistic Regression'
kfold = model_selection.KFold(n_splits=10, random_state=seed)
cv_results = model_selection.cross_val_score(model, X_train, Y_train,
cv=kfold, scoring='accuracy')
msg = "%s: %f (%f)" % (name, cv_results.mean(), cv_results.std())
print(msg)

```

El output que me ofrecen es: Logistic Regression: 0.996315 (0.004275)

Este output o salida significa que al probar el modelo con el treinta por ciento del total de los datos que no se utilizó para entrenarlo obtengo un 0.99 por ciento de precisión. Es un valor altísimo, pero tiene una explicación, lo que este valor nos está diciendo es que si quisiera predecir un partido de fútbol y tuviese todos estos datos: nombre del equipo local, nombre del equipo visitante, goles del equipo local al descanso, goles del equipo visitante al descanso, resultado del partido al descanso, goles del equipo local al final, goles del equipo visitante al final, resultado del partido al final, tiros del equipo local, tiros del equipo visitante, tiros a puerta del equipo local, tiros a puerta del equipo visitante, faltas cometidas por el equipo local, faltas cometidas por el equipo visitante, tarjetas amarillas equipo local, tarjetas amarillas equipo visitante, tarjetas rojas equipo local, tarjetas rojas equipo visitante, cuota de victoria del equipo local, cuota de empate y cuota de victoria del equipo visitante pues el modelo nos arrojaría un resultado preciso al 99%. ¿Cuál es el problema de esto? Que es evidente que antes de empezar el partido es imposible tener todos estos datos, ya que los únicos que podríamos conseguir serían: nombre del equipo local, nombre del equipo visitante, cuota para el equipo local, cuota para el empate y cuota para el equipo visitante. Es por eso que a la hora de predecir un partido real ese porcentaje de acierto bajará mucho.

Para predecir un partido de fútbol que se elija mi idea fue que el usuario metiese como datos el nombre del equipo local, el nombre del equipo visitante, la cuota del equipo local, la cuota del empate y la cuota del equipo visitante. Como ya mencioné anteriormente esto es un problema, ya que el modelo además de esos datos necesita todas las demás estadísticas del partido con las que se le ha entrenado. Para resolver este problema se me ocurrió que la manera más sencilla e inteligente de solventarlo era introducirle las medias de cada una de esas estadísticas para el equipo en cuestión. Por ejemplo, si un usuario quisiese predecir un partido en el que el equipo local fuese el Sevilla, el equipo visitante fuese el Real Madrid, la cuota del Sevilla fuese 2.43, la cuota de empate 3.21 y la cuota del Real Madrid fuese 2.23 las demás estadísticas serían calculadas de la siguiente manera:

1. **Goles del equipo local al descanso:** Hago una media de los goles hasta el descanso que haya metido el Sevilla jugando como equipo local.
2. **Goles del equipo visitante al descanso:** Hago una media de los goles hasta el descanso que haya metido el Real Madrid jugando como equipo visitante.
3. **Resultado al descanso:** Hago una media de los resultados que han obtenido al descanso ambos equipos. Este resultado siempre oscilará entre 0 y 3, cuanto más se acerque al 0 significará que el equipo local suele tener mejores resultados al descanso, si se acerca al 1 querrá decir que ambos equipos suelen

ir empatados al descanso y cuanto más se acerque al 2 significará que el equipo visitante suele ir ganando al descanso.

4. **Goles del equipo local al final:** Hago una media de los goles al final del partido que haya metido el Sevilla jugando como equipo local.
5. **Goles del equipo visitante al final:** Hago una media de los goles al final del partido que haya metido el Real Madrid jugando como equipo local.
6. **Resultado final del partido:** Hago una media de los resultados que han obtenido al final del partido ambos equipos. Este resultado siempre oscilará entre 0 y 3, cuanto más se acerque al 0 significará que el equipo local suele tener mejores resultados al final del partido, si se acerca al 1 querrá decir que ambos equipos suelen acabar empatando al final del partido y cuanto más se acerque al 2 significará que el equipo visitante suele ganar los partidos al final del partido.
7. **Tiros del equipo local:** Hago una media de los tiros que ha hecho el Sevilla en todos los partidos que ha jugado.
8. **Tiros del equipo visitante:** Hago una media de los tiros que ha hecho el Real Madrid en todos los partidos que ha jugado.
9. **Tiros a puerta del equipo local:** Hago una media de los tiros a puerta que ha hecho el Sevilla en todos los partidos que ha jugado.
10. **Tiros a puerta del equipo visitante:** Hago una media de los tiros a puerta que ha hecho el Real Madrid en todos los partidos que ha jugado.
11. **Faltas cometidas por el equipo local:** Hago una media de las faltas que ha cometido el Sevilla en todos los partidos que ha jugado.
12. **Faltas cometidas por el equipo visitante:** Hago una media de las faltas que ha cometido el Real Madrid en todos los partidos que ha jugado.
13. **Tarjetas amarillas del equipo local:** Hago una media de las tarjetas amarillas que ha recibido el Sevilla a en todos sus partidos.
14. **Tarjetas amarillas del equipo visitante:** Hago una media de las tarjetas amarillas que ha recibido el Real Madrid a en todos sus partidos.
15. **Tarjetas rojas del equipo local:** Hago una media de las tarjetas rojas que ha recibido el Sevilla a en todos sus partidos

**16. Tarjetas rojas del equipo visitante:** Hago una media de las tarjetas rojas que ha recibido el Real Madrid a en todos sus partidos.

Evidentemente este ejemplo práctico en el que he puesto como equipo local al Sevilla y como equipo visitante al Real Madrid sirve para cualquier par de equipos. Las instrucciones en *Python* para realizar estas medias y hacer en base a ellas la predicción del resultado del partido en cuestión son las siguientes:

```
medias_Local = df_inicial.groupby('Equipo_Local')
medias_Visitante = df_inicial.groupby('Equipo_Visitante')

medias_Local = medias_Local.mean()
medias_Visitante = medias_Visitante.mean()

media_Resultado_Descanso =
(medias_Local['Resultado_Al_Descanso'][equipo_local] +
medias_Visitante['Resultado_Al_Descanso'][equipo_visitante]) / 2

X_new =
pd.DataFrame({'Goles_Descanso_Local': [float(medias_Local['Goles_Descanso_Local']
[equipo_local])],
'Goles_Descanso_Visitante': [float(medias_Visitante['Goles_Descanso_Visitante']
[equipo_visitante])], "Resultado_Al_Descanso": [media_Resultado_Descanso],
"Goles_Final_Local": [float(medias_Local['Goles_Final_Local'][equipo_local])],
"Goles_Final_Visitante": [float(medias_Visitante['Goles_Final_Visitante'][equipo_visitante])],
"Tiros_Local": [float(medias_Local['Tiros_Local'][equipo_local])],
"Tiros_Visitante": [float(medias_Visitante['Tiros_Visitante'][equipo_visitante])],
"Tiros_Puerta_Local": [float(medias_Local['Tiros_Puerta_Local'][equipo_local])],
"Tiros_Puerta_Visitante": [float(medias_Visitante['Tiros_Puerta_Visitante'][equipo_visitante])],
"Faltas_Local": [float(medias_Local['Faltas_Local'][equipo_local])],
"Faltas_Visitante": [float(medias_Visitante['Faltas_Visitante'][equipo_visitante])],
"Corners_Local": [float(medias_Local['Corners_Local'][equipo_local])],
"Corners_Visitante": [float(medias_Visitante['Corners_Visitante'][equipo_visitante])],
"Amarillas_Local": [float(medias_Local['Amarillas_Local'][equipo_local])],
"Rojas_Local": [float(medias_Local['Rojas_Local'][equipo_local])],
"Amarillas_Visitante": [float(medias_Visitante['Amarillas_Visitante'][equipo_visitante])],
"Rojas_Visitante": [float(medias_Visitante['Rojas_Visitante'][equipo_visitante])],
'Bet365_Local': [cuota_local], 'Bet365_Empate': [cuota_empate],
'Bet365_Visitante': [cuota_visitante]})

resultado = model.predict(X_new)
```

En el hipotético caso de que el usuario introdujese este partido de ejemplo para predecir al modelo tendría que hacerlo de esta manera:

```
entrenar_modelo_predecir(LaLiga_df , df, 'Sevilla', 'Real Madrid', 2.43, 3.21, 2.23)
```

Obteniendo como *output*: [2]

En el caso de que le introduzcamos el encuentro de ejemplo al modelo, este predice que el ganador sería el Real Madrid, ya que arroja como resultado el número dos (la victoria del equipo visitante).

Con el fin de completar algo más mi proyecto investigué si había alguna manera de obtener las probabilidades que el modelo de predicción calcula para cada uno de los resultados posibles. Tras indagar un poco descubrí que existía una función de sklearn llamada `predict_proba()` que permitía dicha tarea. El código de esta función sería muy sencillo:

```
probabilidad = model.predict_proba(X_new)
print(probabilidad[0][0])
print(probabilidad[0][1])
print(probabilidad[0][2])
```

Obteniendo como output: 0.0031382651060814608 0.45794675832619564  
0.5389149765677229

De esta salida podemos deducir que el modelo predice que el equipo local (en este caso el Sevilla) tiene muy poca probabilidad de ganar, que la probabilidad del empate es de casi el 46% y que la probabilidad de que gane el equipo visitante (en este caso el Real Madrid) es de casi el 54%.

Como esta forma de introducir los datos de un partido y visualizar el resultado y la probabilidad es un poco tediosa y muy poco vistosa decidí construir una aplicación web para complementar este modelo de predicción.

## 5. Creación de la página web

Como he mencionado previamente, la creación de una página web para que el usuario pueda interactuar con ella me pareció una manera adecuada de continuar con el trabajo para que al final me quedase un proyecto de una calidad decente.

Como siempre, antes de ponerse a programar se debe hacer un diseño de al menos la estructura del producto final que el programador desea ofrecer. En este caso, mi idea inicial era que la página principal simplemente fuese una ventana en la que se le pidiese al usuario que eligiese una liga de las cinco con las que este modelo trabaja (Bundesliga, La Liga Santander, Ligue One, Serie A y Premier League). Cada una de esas ligas tendría una página secundaria donde se mostrarían los equipos que cada liga tendría disponibles y unos componentes de texto donde se podría introducir las cuotas



para el partido elegido. De esa manera cuando se pulsase un botón de 'OK' se mostraría el resultado final calculado para ese partido. Dicho esquema inicial que dibujé está reflejado a continuación en la figura 10.

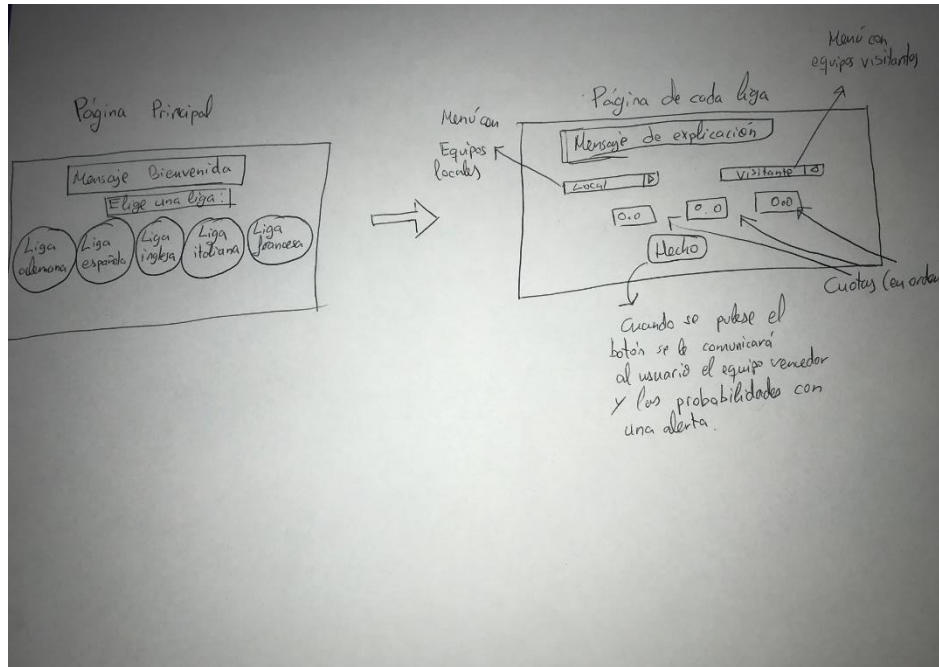


Figura 10

Ahora que ya tenía un esquema inicial hecho ya podía empezar a programar para crear el *frontend*. Como utilicé *Vuejs* con *Vue Cli* integrado lo que hice fue trabajar con vistas. Cada vista o archivo de extensión *.vue* es una ventana, así que lo que tenía que hacer es la ventana principal (*Home.vue*), y una ventana para cada liga (*Bundesliga.vue*, *La\_liga.vue*, *Ligue\_one.vue*, *Premier\_league.vue* y *Serie\_a.vue*). La organización de dichos elementos la podemos ver a continuación en la figura 11:

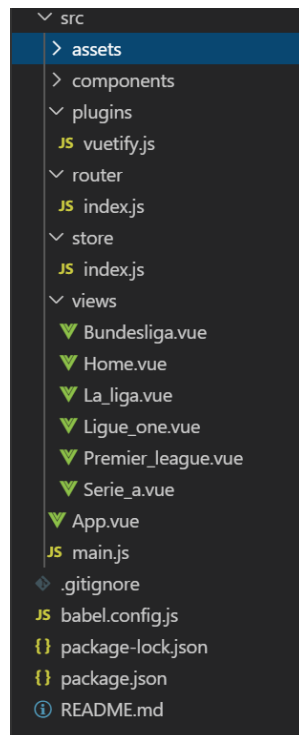


Figura 11

Los demás archivos que aparecen en la figura 11 son simplemente archivos de configuración. En la página principal simplemente hay un mensaje de bienvenida al usuario en el que se le pide que elija una liga de las que aparecen en la página. Dependiendo de la liga en la que haga *click* se le enviará a una ventana o a otra. Podemos observar una captura de pantalla de la página principal en la figura 12.



Figura 12

Cuando el usuario hace *click* en la imagen de alguna de las ligas se le envía a la ventana de la liga en cuestión, de la que podemos observar una captura de imagen en la figura 13. En esta ventana aparece un mensaje en el que se le pide al usuario que elija los equipos y las cuotas. Los equipos los elige de los dos menús desplegables y las cuotas las introduce en las distintas cajas de texto. Cuando el usuario hace *click* en enviar se le envían dichos datos al servidor de *Python* (del que hablaré a continuación) y este devuelve el resultado que calcula con el algoritmo de regresión logística para que dicho resultado se le muestre al usuario en la página web.

La imagen muestra una interfaz web superpuesta sobre una fotografía de un campo de fútbol con una portería. En la parte superior, un mensaje en un recuadro oscuro indica: "Elige los equipos e introduce las cuotas del partido.". Debajo, hay dos menús desplegables etiquetados "Equipo Local" y "Equipo Visitante". En la fila inferior, hay tres campos de entrada de texto etiquetados "Cuota equipo local", "Cuota empate" y "Cuota equipo visitante", cada uno con el número "1" ingresado. A la derecha de estos campos hay un botón blanco con el texto "ENVIAR" en negro.

Figura 13

Para ejecutar el frontend utilizo el comando 'vue ui', el cual me abre una página web en la que puedo consultar el estado y los problemas que puede tener mi página web. El puerto en el que escucha la aplicación es el 8000. Se podría decir que esta página web de ayuda es una especie de taller en el que se pueden consultar varios aspectos acerca de mi propia página web. El aspecto que tiene es el siguiente:

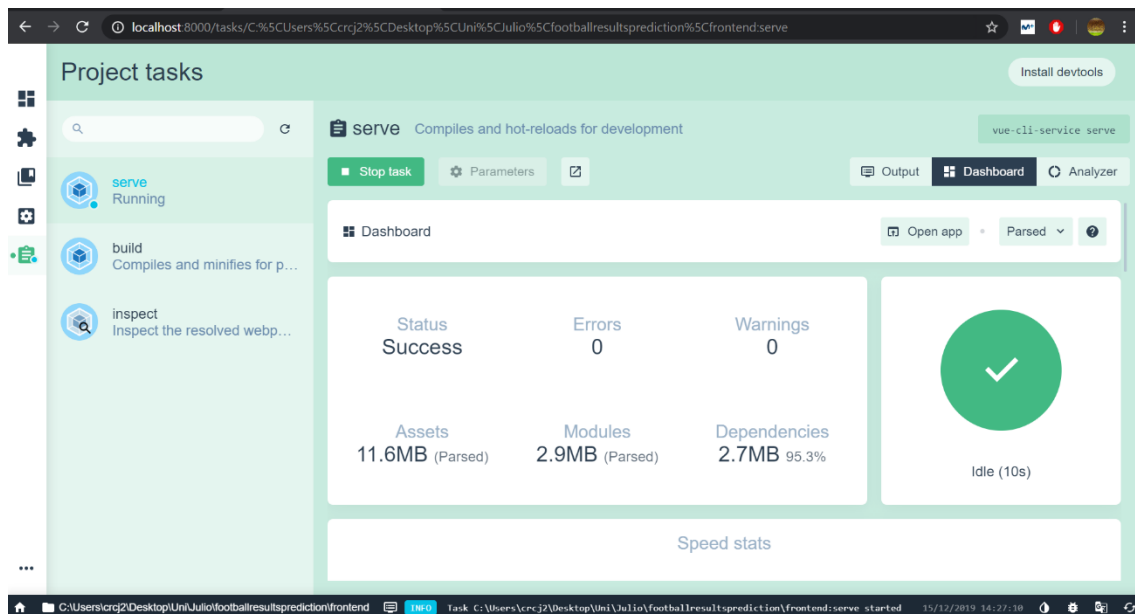
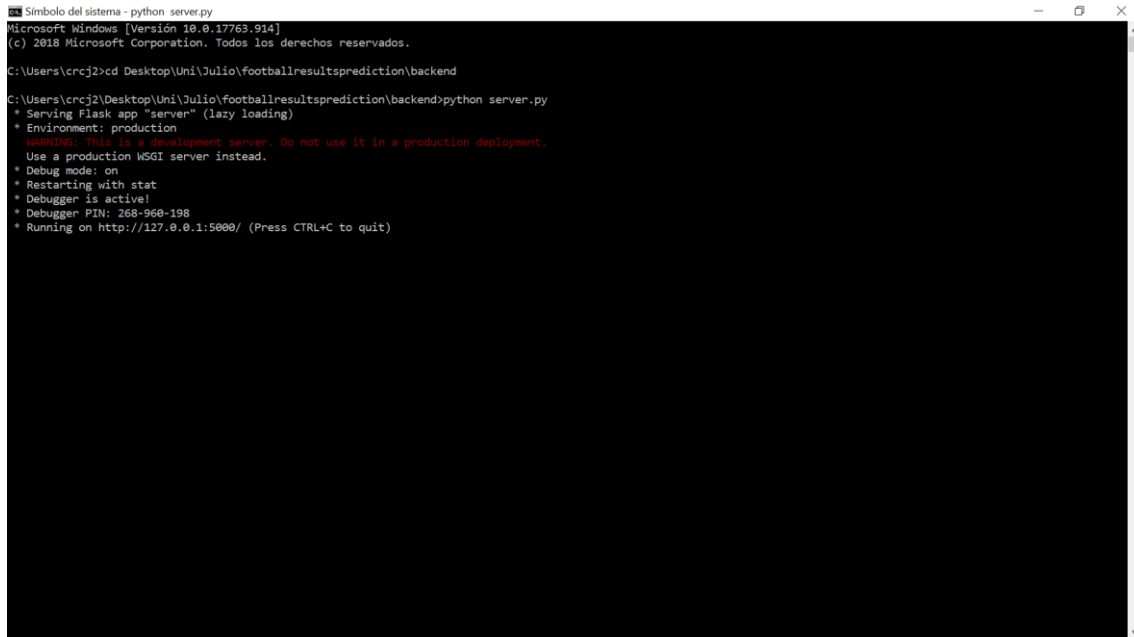


Figura 14

El servidor de Python, que es el que trabaja como *backend*, es un script de Python que usa la herramienta *Flask* y se dedica a escuchar las peticiones que se le envían desde el *frontend*, es decir, desde la página web. En dicho script también están las funciones que detallé anteriormente en las que se crean los *dataframes* a partir de la base de datos de grafos y se entrena el modelo. La ejecución del servidor se hace con el comando `'python server.py'`. Cuando el servidor se está ejecutando en el puerto 5000 y por tanto comienza a esperar a que el *frontend* se comuniquen con él se puede observar el siguiente mensaje en la terminal:




```
Símbolo del sistema - python server.py
Microsoft Windows [Versión 10.0.17763.914]
(c) 2018 Microsoft Corporation. Todos los derechos reservados.

C:\Users\crcj2>cd Desktop\Unl\Julio\footballresultsprediction\backend

C:\Users\crcj2\Desktop\Unl\Julio\footballresultsprediction\backend>python server.py
* Serving Flask app "server" (lazy loading)
* Environment: production
  WARNING: This is a development server. Do not use it in a production deployment.
  Use a production WSGI server instead.
* Debug mode: on
* Restarting with stat
* Debugger is active!
* Debugger PIN: 268-960-198
* Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)
```

Figura 15

Es cuando le llega una petición al *backend* cuando este se encarga de hacer una predicción y devuelve tanto el resultado final del encuentro con los datos que ha recibido como la probabilidad de que ocurra. Para que quede el ejemplo clarificado dejo una captura de pantalla del archivo `server.py` que es el *backend* en la figura 16. En ese fragmento de código se puede observar lo que hace el servidor cuando se le hace una petición desde la ventana de la liga alemana, aunque funcionaría igual para todas las demás ligas.



```
app = Flask(__name__)
cors = CORS(app, resources={r"/api/*": {"origins": "*"}})

simplefilter(action='ignore', category=FutureWarning)
uri = "bolt://localhost:7687"
driver = GraphDatabase.driver(uri, auth= ("neo4j", "pass"))

@app.route('/api/v1.0/prediccionbundesliga', methods=['GET','POST'])
def bundesliga():
    data = request.get_json()
    array = []
    for key, value in data.items():
        array.append(value)

    equipo_local = array[0]
    equipo_visitante = array[1]
    cuota_local = array[2]
    cuota_empate = array[3]
    cuota_visitante = array[4]

    nombre_equipo_local = getNombreBundesliga(int(equipo_local))
    nombre_equipo_visitante = getNombreBundesliga(int(equipo_visitante))

    with driver.session() as session:
        Bundesliga_df = session.read_transaction(crear_dataframe_encuentros, "Bundesliga")
        session.close()

    df = Bundesliga_df.copy()
    resultado = entrenar_modelo_predecir(Bundesliga_df , df, nombre_equipo_local, nombre_equipo_visitante, cuota_local, cuota_empate, cuota_visitante)
    return jsonify(resultado_final=int(resultado[0]), probabilidad_local=resultado[1], probabilidad_empate=resultado[2], probabilidad_visitante=resultado[3])
```

Figura 16

Me surgió un problema a la hora de acceder a la base de datos con los datos de la página web. Este problema consistía en que en la página web los nombres están almacenados sin tildes y sin 'ñ', además de esto en algunos casos los nombres de los equipos no son del todo descriptivos. Por ejemplo, a la Real Sociedad está almacenada simplemente como 'Sociedad', el Atlético de Madrid está almacenado como 'Ath Madrid' o el Espanyol está almacenado como 'Espanol'. No hubiese sido una buena idea colocar estos nombres en el menú de elección de equipos, pero lo que tampoco podía hacer era simplemente poner de manera correcta los nombres en la interfaz porque esto arrojaría un error. La solución que pensé fue hacer una función con Python por cada liga a la que le envío el nombre incorrecto y dependiendo de este nombre retorna el nombre correcto. En la figura 17 podemos ver un ejemplo de esa función, en este caso es la de la liga española, aunque como ya he dicho hay una por cada liga.

```
145 def getNombreLiga(num):
146     switcher={
147         1:'Barcelona',
148         2:'Real Madrid',
149         3:'Ath Madrid',
150         4:'Sevilla',
151         5:'Valencia',
152         6:'Ath Bilbao',
153         7:'Sociedad',
154         8:'Espanol',
155         9:'Betis',
156         10:'Malaga',
157         11:'Getafe',
158         12:'Celta',
159         13:'Alaves',
160     }
161     return switcher.get(num,"Invalid team")
162
```

Figura 17

Ahora que ya he explicado cómo es el funcionamiento de la aplicación web que he construido podemos observar lo fácil que es introducirle los datos al modelo ahora en comparación con lo tedioso que era introducirlos sin que existiese una aplicación web. Veamos cómo se mostraría un resultado en la página web, para ello he introducido los datos que ya he utilizado previamente, es decir, el hipotético partido del Sevilla contra el Real Madrid. El resultado de introducir estos datos en la página web se pueden observar a continuación en la figura 18:

Elige los equipos e introduce las cuotas del partido.

Equipo Local  
Sevilla FC

Equipo Visitante

Cuota equipo local  
2.43

El ganador del encuentro será el Real Madrid.

Dicho resultado ha sido calculado para el partido Sevilla FC vs Real Madrid que tenía como cuotas 2.43 para el Sevilla FC, 3.21 para el empate y 2.23 para el Real Madrid. La probabilidad de que gane el Sevilla FC es el 0.3138265106081461%. La probabilidad del empate es el 45.79467583261956%. La probabilidad de que gane el Real Madrid es el 53.891497656772295%.

ENVIAR

VALE

Figura 18

Como ya he mencionado antes, pienso que de esta manera el proyecto queda mucho más vistoso y completo, además de que es mucho más fácil introducirle al modelo los datos para que haga la predicción.

Implementé en el modelo otra ventana a la que se le lleva al usuario en el caso de que elija el mismo equipo como local que como visitante. Por ejemplo, si el usuario elige la liga española, como equipo local selecciona al Betis y como equipo visitante selecciona también al Betis, se le muestra un diálogo en el que se le dice que ha cometido un error al elegir los equipos. Si pulsa 'OK' saldrá de dicho diálogo y podrá volver a hacer la selección. Se puede observar dicho escenario en la figura 19.



Elige los equipos e introduce las cuotas del partido.

Equipo Local: Betis

Equipo Visitante: Betis

Cuota equipo local: 2.25

No se ha podido realizar la predicción

Revisa los datos que has introducido, el equipo local no puede ser el mismo que el equipo visitante.

VALE

ENVIAR

Figura 19

He de decir que en teoría se le puede pasar al modelo cualquier equipo mientras dicho equipo exista en la base de datos. Pese a esto, en la página web decidí mostrar sólo los equipos más importantes de cada liga. Esto lo hice por una sencilla razón, si el usuario elige un equipo que ha disputado muy pocos partidos en primera división en las últimas diez temporadas va a haber muy pocos registros de dicho equipo en mi base de datos y por tanto el resultado que el modelo arrojaría no sería muy preciso. Los equipos que se pueden elegir de cada liga porque existen suficientes registros suyos en la página web son los que aparecen en la siguiente tabla:

Liga alemana	Liga española	Liga francesa	Liga italiana	Liga inglesa
Borussia Dortmund	Real Madrid	Olympique Lyon	Milan	Manchester United
Bayern Munich	FC Barcelona	Paris Saint Germain	Inter	Chelsea
Bayern Leverkusen	Atlético de Madrid	Saint-Etienne	Napoli	Manchester City
Borussia Monchengladbach	Sevilla	Olympique Marseille	Lazio	Arsenal
Schalke 04	Valencia	Lille	Roma	Tottenham
Wolfsburg	Athletic de Bilbao	Rennes	Juventus	Liverpool
Hoffenheim	Real Sociedad	Bordeaux	Fiorentina	Everton
Hannover 96	Espanyol	Nice	Torino	Southampton
Werder Bremen	Betis	Monaco	Udinese	Leicester
Frankfurt	Málaga	Montpellier	Parma	Newcastle
Hamburg	Getafe	Toulouse	Genoa	West Bromwich
Friburg	Celta	Lorient	Sampdoria	West Ham
FC Koln	Alavés	Nantes	Atalanta	Stoke City



Con el fin de dejar un poco más claras las fases de desarrollo del proyecto incluyo un pequeño esquema en el que aparecen junto a las herramientas utilizadas en cada una de las fases:

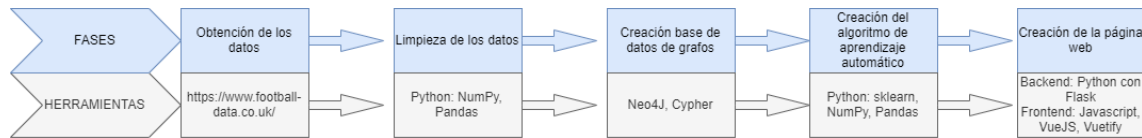


Figura 20

Tras explicar las distintas partes del proyecto, pasamos a la segunda mitad de este apartado, en la que le introduciré varios ejemplos reales para ver el funcionamiento del sistema de predicción y analizaré los resultados obtenidos. Para llevar a cabo esta tarea voy a utilizar encuentros que ya se hayan jugado pero que obviamente no se encuentran almacenados en la base de datos. Por ejemplo, voy a usar encuentros recientes que se hayan jugado en las últimas jornadas (a fecha 12/12/2019) para que sea fácil encontrar la información de dichos encuentros y comprobar la veracidad de los mismos. En la tabla que aparece a continuación se muestran el equipo local, el equipo visitante, las cuotas del encuentro, el resultado real, el resultado calculado por mi modelo y las probabilidades que le atribuyó a cada resultado.

Local	Visitante	Cuota local	Cuota empate	Cuota visitante	Resultado real	Resultado calculado	% local	% empate	% visitante
Alavés	Real Madrid	6.25	4.75	1.42	2	2	<0.1%	18.36%	81.64%
Atlético de Madrid	FC Barcelona	2.80	3.25	2.45	2	2	0.2%	32.06%	67.72%
Real Madrid	Espanyol	1.16	7.00	15.00	0	0	93.98%	6.02%	<0.1%
Betis	Athletic	2.25	3.20	3.20	0	1	34.01%	65.34%	0.64%
Bayern Munich	Bayern Leverkusen	1.25	6.00	9.50	2	1	1.85%	79.98%	18.15%
M'gladbach	Friburg	1.62	4.00	5.00	0	0	80.23%	19.76%	<0.1%
Wolfsburg	Werder Bremen	2.15	3.60	3.10	2	1	41.71%	57.74%	0.54%
Bayern Leverkusen	Schalke	1.95	3.60	3.60	0	0	88.09%	11.91%	<0.1%
Olympique Lyon	Lille	1.67	4.00	4.50	2	0	83.18%	16.82%	<0.1%
Saint-Etienne	Nice	2.15	3.39	3.50	0	0	73.12%	26.86%	<0.1%
Toulouse	Monaco	3.20	3.40	2.15	2	1	0.65%	73.73%	25.61%
Olympique Marseille	Bordeaux	1.83	3.50	4.20	0	0	70.16%	29.83%	<0.1%
Newcastle	Manchester City	15.00	7.25	1.18	1	2	<0.1%	10.07%	89.87%
Arsenal	Southampton	1.45	4.75	7.00	1	0	84.19%	15.81%	<0.1%
Manchester City	Manchester United	1.75	3.75	4.75	2	0	85.99%	14.00%	<0.1%
Everton	Chelsea	3.50	3.75	2.00	0	1	19.59%	78.86%	1.54%
Inter	Roma	1.87	3.60	3.90	1	1	42.37%	57.14%	0.48%
Udinese	Napoli	4.90	3.60	1.70	1	2	0.22%	45.82%	54.94%
Genoa	Torino	2.40	3.25	2.90	2	1	9.25%	86.90%	3.83%
Lazio	Juventus	2.80	3.25	2.45	0	1	31.77%	67.09%	1.13%

Como se puede apreciar en la tabla, el modelo acierta claramente el resultado de los partidos que a priori parecen fáciles, es decir, partidos en los que juega un equipo fuerte contra otro más humilde. Como por ejemplo el Alavés-Real Madrid, Real Madrid-Espanyol o M'gladbach-Friburg. Esto ocurre debido a que en los partidos que se le pasaron al modelo como entrenamiento hay un gran número de encuentros en los que los equipos fuertes ganan a equipos más humildes. Por otra parte, cuando se le pasan al modelo de predicción partidos relativamente igualados este suele arrojar como resultado final la victoria del equipo local (como por ejemplo el Bayern Leverkusen-Schalke, Olympique Marseille-Bordeaux o Manchester City-Manchester United) aunque muchas veces también arroja empate (como por ejemplo el Betis-Athletic, Inter-Roma o Genoa-Torino). En otro orden de cosas, también se puede ver que normalmente los resultados en los que 'se da la sorpresa', es decir, en los que el resultado final del partido está muy lejos del esperado el modelo suele fallar. Un par de ejemplos de estos resultados son el Bayern Leverkusen-Bayern Munich en el que el que salió victorioso el equipo local pero el modelo predijo que ganaría el equipo visitante (ya que en teoría debería ganar o al menos haber empatado ese partido). Otro ejemplo de la misma índole es el Lazio-Juventus. Sobre el papel la Juventus debería haber ganado o al menos haber empatado con el Lazio pero sin embargo el ganador fue el equipo local. Por tanto, el modelo falló este partido ya que predijo que el encuentro acabaría en empate.

En cuanto a las probabilidades que el sistema atribuye a cada uno de los resultados, cabe decir que suele ser muy tajante. Como podemos ver en la tabla en muchas ocasiones el resultado principal tiene casi siempre un valor muy alto y los resultados menos posibles tienen valores bajísimos, llegando en algunas ocasiones a ser menor del 0.1%. Es sólo en algunas ocasiones cuando podemos ver que se obtienen dos probabilidades similares, por ejemplo, cuando se enfrentan dos equipos que tienen un nivel muy parecido (como es el caso del Inter-Roma) se obtienen porcentajes similares (42% para la Roma y 57% para el empate).

En conclusión, el sistema predice partidos de una manera similar a una persona, normalmente acierta los partidos en el que alguno de los dos equipos es superior y también acierta bastantes empates. El problema viene cuando el resultado del partido no es el esperado, es ahí donde mi modelo de predicción falla y no pronostica nada bien dichos desenlaces.

# ANÁLISIS CRÍTICO

El problema principal de desarrollar un proyecto que consiga predecir resultados de partidos de fútbol es que se trata de un deporte en el cual es muy difícil acertar un resultado. Este obstáculo se produce porque en los partidos de fútbol el marcador final suele ser la gran mayoría de las veces un resultado demasiado ajustado. Aunque un equipo fuerte juegue contra uno bastante más débil, en el resultado final no se suele apreciar dicha diferencia, ya que perfectamente un partido de este tipo puede terminar con un resultado ajustado como 1-0 o 2-1. Es por eso que es complicado crear un modelo de predicción de resultados futbolísticos que posea una alta probabilidad de acierto (actualmente no existe ninguno). A donde quiero llegar es que, en mi caso, desde un principio yo ya sabía que iba a ser prácticamente imposible crear un sistema que acertase todos los partidos, y muy complicado crear un sistema que acertase la mayoría de los partidos. Aunque mi sistema no cumple ninguna de estas dos condiciones, sí que adivina unos cuantos resultados y es por eso que me doy por satisfecho ya que con eso he cumplido mi objetivo.

Considero que el número de equipos que se pueden elegir por cada liga (13) es también una debilidad. Si el usuario quiere usar mi modelo para predecir un partido se puede dar la situación de que alguno de los dos equipos o incluso ambos no se encuentren en la lista de equipos disponibles para elegir. Como ya detallé en apartados anteriores, preferí hacer esta lista de equipos un poco más pequeña con el fin de que el algoritmo de *machine learning* trabaje mejor ya que en esa lista de equipos he añadido todos los equipos que tienen un número elevado de partidos para que así el entrenamiento del modelo sea de mejor calidad.

Pienso que la página web que he desarrollado para este trabajo es una de las fortalezas del mismo, ya que un proyecto con estas cualidades en el que se necesita que un usuario introduzca unos datos determinados para poder pasárselos al algoritmo de predicción necesita alguna clase de interfaz con la que el usuario pueda interactuar. En mi opinión, si mi proyecto no tuviese integrada la página web en la que se introducen los datos, perdería mucha calidad porque tendrían que introducirse los datos a mano con los posibles problemas que esto puede producir. Además de todo esto, me considero satisfecho en cuanto a cómo ha quedado la página web. Pese a que se trata de una interfaz sencilla creo que me ha quedado muy vistosa y atractiva.

He llegado a la conclusión de que he hecho un buen trabajo creando la base de datos de grafos con la herramienta *Neo4J*. Pienso esto debido a que en algunas ocasiones tras crearla tuve que incluir algún pequeño cambio sin mucha importancia y a la hora

de hacerlo me di cuenta de que era muy sencillo aplicar dichos cambios. Esto es porque el diseño general de la estructura de la base de datos que realicé era bueno y permitía cambios posteriores de manera sencilla y sin tener quebraderos de cabeza sobre datos que estuviesen mal relacionados o algo similar. También ayuda el hecho de que *Neo4J* ofrece *Cypher*, un lenguaje de consulta sencillo y de gran versatilidad que permite hacer los cambios de los que he hablado de manera un poco más rápida que en bases de datos más convencionales como *MySQL* o *phpmyadmin*.

# LÍNEAS DE FUTURO

Como ya he mencionado, una de las razones que impedía que este trabajo fuese de mayor calidad era el hecho de que no disponía de mucho tiempo para trabajar sobre las distintas partes del mismo o mejorarlas. En este apartado, explicaré varias ideas que se me han ido ocurriendo durante el desarrollo del trabajo para mejorar el proyecto en el futuro, es decir, cuando pueda dedicarle más tiempo.

1. **Alimentar la base de datos:** La base de datos de la que se nutre el sistema de predicción está actualizada hasta principios del mes de octubre del año 2019. Mi idea es seguir actualizándola progresivamente ya que a medida que vaya habiendo un número mayor de datos registrados en la base de datos más preciso se va a ir volviendo el modelo de predicción, es decir, subirá su porcentaje de acierto de partidos.
2. **Implementar la funcionalidad de que se pueda elegir la casa de apuestas de la que se introducen las cuotas:** Se trata de una funcionalidad que tenía pensado implementar desde un principio, pero pensé que para hacer el desarrollo un poco más sencillo era buena idea hacerlo simplemente para sólo una de ellas. En el futuro estaría bien que el usuario pudiese elegir la casa de apuestas.
3. **Añadir más equipos por liga:** Actualmente en la aplicación web solo se pueden seleccionar trece equipos por liga. Esto puede llegar a ser un problema ya que si el usuario desea predecir un partido en el que participe un equipo que no se encuentra en esta lista no podrá hacerlo. A medida que se vaya alimentando más y más la base de datos, con el tiempo habrá registros suficientes de equipos que actualmente tienen muy pocos. Cuando llegue ese momento podré comenzar a introducir equipos nuevos con el fin de tener los máximos posibles.
4. **Añadir más ligas:** Aunque es cierto que he añadido las ligas más importantes europeas, es probable que en el futuro podría llegar a existir un gran número de usuarios que quisiesen utilizar mi sistema de predicción con otras ligas menos importantes como pueden ser la segunda división española o la liga de Estados Unidos, que cada vez tienen más seguidores. De la misma manera también se me ha ocurrido añadir partidos internacionales, como por ejemplo de la UEFA Champions League o de la UEFA Europa League, así como partidos entre selecciones para predecir la Eurocopa o el mundial de fútbol.
5. **Abrir la puerta a otros deportes:** Este proyecto está enfocado solamente al fútbol, pero en el futuro, si me planteo crear una página web dedicada a la

predicción de resultados deportivos estaría bien crear sistemas de predicción similares al que tengo ya creado, pero sobre otros deportes que también estén relacionados con el mundo de las apuestas, por ejemplo, la NBA o la liga de béisbol americana. Incluso en este último caso pienso que sería más fácil crear el sistema de predicción ya que las estadísticas en este ámbito suelen dar más posibilidades de acción.

6. **Mejorar el algoritmo de *machine learning*:** Actualmente, mi proyecto utiliza un algoritmo de regresión logística muy simple. Aunque sea tan sencillo, funciona relativamente bien pero aún así veo que es posible mejorarlo revisando los pesos que le asigna la regresión logística a cada una de las estadísticas que se le pasan. Otro modo de mejorar la precisión del sistema de predicción sería combinar la regresión logística que ya tengo con otros algoritmos sobre *machine learning*.
7. **Añadir más características al modelo:** Sería de gran utilidad añadir varias características al modelo para que gane en precisión a la hora de predecir resultados. Le he echado un vistazo a varios de los modelos de predicción más utilizados por los usuarios en la web, como por ejemplo Fivethirtyone (<https://fivethirtyeight.com/methodology/how-our-club-soccer-predictions-work/>). En este sistema de predicción además de utilizar las estadísticas de los partidos utilizan estadísticas de jugadores, en base a unos rankings sacados de la página Transfermarkt (<https://www.transfermarkt.es/>). Si le añadiese a mi base de datos un set de datos similar a este y se lo pasase también al modelo para entrenarlo, las predicciones deberían de ser más precisas.

# CONCLUSIÓN

El fútbol es uno de los deportes que más seguidores mueve a día de hoy, es por eso que también es uno de los deportes que más dinero mueve alrededor del mundo. Es muy interesante ver cómo los equipos más poderosos tratan de tener en su plantilla los mejores jugadores y el mejor cuerpo técnico posible porque cada pequeño detalle es el que marca la diferencia.

Desde ya hace unos años cada vez más equipos de fútbol han fichado a analistas de datos en sus filas con el fin de que estos les orienten a la hora de tomar decisiones acerca de fichajes, tipos de entrenamiento, cómo va a jugar el rival, etc. El ejemplo más famoso en este ámbito fue la utilización de la aplicación '*Match Insights*' por la selección alemana para que les ayudase a ganar el mundial de fútbol del año 2014. Dicha aplicación les permitía observar el rendimiento de los jugadores y algunos puntos débiles del oponente, además de otras funcionalidades. A medida que vayan pasando los años, los analistas de datos tanto en el fútbol como en otros deportes irán ganando importancia debido al potencial que tiene analizar los datos.

Este potencial se puede observar claramente en mi trabajo. Sin ser yo una persona con mucha experiencia en ámbito del análisis de datos y mucho menos en el ámbito del *machine learning* he sido capaz de crear un sistema de predicción de resultados futbolísticos simple en relativamente poco tiempo con el que he conseguido predecir el resultado de varios partidos. A donde quiero llegar es a que, si yo he conseguido esto pese a mi falta de experiencia y mi falta de tiempo, si hay una gran inversión de dinero y varias personas con la experiencia y tiempo necesarios se ponen a desarrollar un proyecto de predicción similar al mío, dicho trabajo puede tener un poder y un valor realmente considerables. Y por supuesto, a medida que las tecnologías de este campo vayan avanzando las posibilidades irán creciendo también a un ritmo altísimo.

Por supuesto, no sólo el fútbol se puede medir. Un claro ejemplo de esto fueron los Athletics de Oakland. Se trata de un equipo de béisbol americano el cual en el año 2002 venía de una crisis deportiva y estaba obligado a formar una plantilla competitiva con un presupuesto muy inferior al de sus rivales. Para ello se les ocurrió una idea, actuar en base a estadística avanzada para fichar jugadores y añadirlos a sus filas. Este método fue bautizado con el nombre de '*Moneyball*' y gracias a que lo llevaron acabo de manera correcta consiguieron contra todo pronóstico veinte victorias seguidas. Esto marcó un antes y un después y forzó a los demás equipos a utilizar técnicas similares



con el fin de mejorar su nivel a la hora de contratar deportistas y cuerpo técnico, así como analizar sus jugadas. A partir de esta historia se produjo la famosa película que lleva el mismo nombre que el método de los Athletics de Oakland.

Los análisis de datos combinados a su vez con técnicas de aprendizaje automático llevados a cabo de una manera apropiada son una herramienta tremendamente potente y además con gran potencial y versatilidad. Cada vez vemos más ejemplos de equipos que no solamente en el fútbol, sino también en muchos otros deportes utilizan estas herramientas u otras similares con el fin de potenciar sus resultados. A medida que vayan pasando los años, los datos irán ganando más y más valor ya que con ellos se podrán llevar a cabo predicciones asombrosas que serán de un valor elevado.

# BIBLIOGRAFÍA

- <https://www.football-data.co.uk/> Página web que tiene muchos archivos de partidos de fútbol con sus correspondientes estadísticas y cuotas de la casa de apuestas de la que he descargado los conjuntos de datos necesarios para este trabajo.
- <https://fivethirtyeight.com/> Página de predicción de resultados deportivos que tomé como ejemplo para mi proyecto.
- <https://neo4j.com/graphacademy/> Academia online de Neo4J en la que aprendí a utilizar tanto su plataforma como su lenguaje de consulta Cypher.
- <https://www.kaggle.com/datasets> Página en la que visualicé varios conjuntos de datos.
- <https://arxiv.org/> Página en la que busqué inspiración buscando artículos sobre predicción de resultados de fútbol.
- <https://www.analyticslane.com/2019/03/25/como-eliminar-columnas-y-filas-en-un-dataframe-pandas/> Ejemplo sobre eliminación de filas y columnas de un set de datos utilizando la librería Pandas.
- <https://github.com/neo4j/neo4j-python-driver> Ejemplo de conexión de una base de datos de Neo4J con un script de Python.
- <https://neo4j.com/docs/cypher-manual/current/clauses/load-csv/> Ejemplos de lecturas de archivos csv para pasarlos a una base de datos de Neo4J
- <https://www.youtube.com/channel/UCWN3xxRkmTPmbKwht9FuE5A/videos> Canal de Youtube de Siraj Raval. En el que he visto algún vídeo sobre preparar datasets, inteligencia artificial y predicción de mercados.
- <https://www.augur.net/> Página en la que se pueden encontrar predicciones de mercados en varios ámbitos.
- <https://neo4j.com/docs/api/python-driver/current/> Ejemplo de consulta desde Python a una base de datos de Neo4J y obtención de los datos de dicha consulta.
- <https://github.com/neo4j-examples/movies-python-py2neo> Ejemplo de aplicación escrita en Python conectada a una base de datos de Neo4J.
- [https://e-archivo.uc3m.es/bitstream/handle/10016/27480/TFG\\_Sergio\\_Calderon\\_Perez-Lozao\\_2017.pdf?sequence=1&isAllowed=y](https://e-archivo.uc3m.es/bitstream/handle/10016/27480/TFG_Sergio_Calderon_Perez-Lozao_2017.pdf?sequence=1&isAllowed=y) Trabajo de fin de grado en el que me he inspirado, sobre predicción de resultados deportivos con técnicas de Machine Learning aplicado al fútbol.
- <https://www.arame.com.mx/blog/8-algoritmos-aprendizaje-automatico-data-mining-mas-usados-aramex-blog/> Artículo para entender cuáles son los distintos algoritmos de aprendizaje automático.
- <https://www.apd.es/algoritmos-del-machine-learning/> Otro artículo para ver que algoritmos de Machine Learning existen.
- <https://conceptosclaros.com/que-es-regresion-logistica/> Artículo para entender qué hace una regresión logística.

- <https://www.revistanefrologia.com/es-la-regresion-logistica-una-herramienta-articulo-X0211699500035664> Otro artículo que explica más acerca de la regresión logística.
- <https://www.aprendemachinelearning.com/regresion-logistica-con-python-paso-a-paso/> Ejemplo de implementación de un algoritmo de regresión logística en Python.
- [https://scikit-learn.org/stable/modules/generated/sklearn.linear\\_model.LogisticRegression.html](https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html) Documentación sklearn sobre regresión logística.
- <https://jaxenter.com/implement-switch-case-statement-python-138315.html> Ejemplo sobre cómo crear una sentencia switch-case en Python.
- <https://medium.com/uptake-tech/how-to-create-a-simple-frontend-api-and-model-with-python-vue-js-a51841c66f8a> Artículo de Medium sobre la creación de una aplicación web usando Vuejs y Flask.
- <https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.DataFrame.std.html> Documentación de la librería Pandas.
- <http://flask.palletsprojects.com/en/1.1.x/> Documentación del framework Flask.
- <https://es-vuejs.github.io/vuejs.org/v2/guide/> Documentación del framework Vuejs.
- <https://vuetifyjs.com/es-MX/getting-started/quick-start> Documentación del framework Vuetify.
- <https://codigofacilito.com/articulos/integracion-vue-flask> Ejemplo de integración de Vuejs con Flask.
- <https://www.youtube.com/watch?v=iwhuPxJ0dig> Ejemplo ilustrado de integración de Flask con Vuejs.
- <https://github.com/GoogleCloudPlatform/ipython-soccer-predictions> Ejemplo de un proyecyo de Google de predicción de resultados de fútbol.
- <https://www.youtube.com/watch?v=jiE3wsrVUQs&t=56s> Vídeo en el que Tim Ward explica cómo ha utilizado Neo4J y machine learning para crear un sistema de decisión.
- <https://www.apuestasdeportivas24.org/Prediccion/pronostico-lazioroma-juventustorino-seriaa-07-12-2019/> Para consultar las cuotas que tenían los partidos con el fin de hacer pruebas utilizando mi modelo de predicción para ver su funcionamiento con casos reales.
- <https://www.draw.io/> Plataforma online que he utilizado para crear algunos esquemas ilustrativos, por ejemplo, el de la figura 20.
- <https://blog.michelletorres.mx/listas-desplegables-en-los-formularios-html/> Ejemplos de utilización de menús desplegables. Son los que yo he utilizado para la selección de los equipos.
- <https://www.luisllamas.es/vuetify-estetica-material-design-para-tus-apps-en-vuejs/> Página en la que se explican las posibilidades que ofrece Vuetify.
- <https://www.wordreference.com/sinonimos/> Diccionario de sinónimos.
- <https://stats.stackexchange.com/questions/179977/scikit-predict-proba-output-interpretation> Ejemplo sobre la interpretación de la función predict\_proba(), que devuelve las probabilidades del modelo.
- <https://unsplash.com/> Página de la que se pueden descargar fotos sin derechos de autor, son las que he utilizado en la página web.
- <https://alligator.io/vuejs/rest-api-axios/> Ejemplos sobre como enviar y/o recibir datos desde el frontend al backend.

- <https://testdriven.io/blog/developing-a-single-page-app-with-flask-and-vuejs/> Ejemplo ilustrado sobre como utilizar Flask a la hora de desarrollar una Single Page Application.
- <https://htmlcolorcodes.com/es/> Página de la que he sacado códigos de colores en HTML para utilizarlos en mi página web.
- <https://www.youtube.com/watch?v=ImKHsIGDJhQ&list=PLPl81lqbj-4J-gfAERGDCdOQtVgRhSvIT&index=22> Vídeo en el que se explica cómo utilizar enlaces de tipo router para cambiar las ventanas de la página web.
- <https://codepen.io/> Página que he visitado para realizar pruebas y explorar soluciones a problemas que me surgieron a la hora de desarrollar el frontend.