CAPTURA MASIVA DE INFORMACIÓN ESTRUCTURADA

Antes de realizar la captura masiva de información se tuvo que elaborar un catálogo de datos con fuentes fiables donde poder conseguirla. Tras las recomendaciones de los tutores y una búsqueda exhaustiva se obtuvieron las siguientes fuentes:

|  |
| --- |
| * <https://fbref.com/es/> |
| * <https://www.transfermarkt.es/> |
| * <https://es.whoscored.com/> |
| * <https://www.footballdatabase.eu/> |
| * <https://one-versus-one.com/es/> |
| * <https://www.livescore.com/en/> |
|  |

A continuación, se analizaron las distintas fuentes contemplando lo que podían ofrecer cada una, y en una primera extracción se decidió utilizar la fuente <https://fbref.com/es/>

Esta fuente es la que más estadísticas a nivel técnico-táctico ofrece contando con la posibilidad de generar reportes de scouting y de crear widgets personalizados con el tipo de estadísticas que deseen obtener.

En un enfoque más técnico, para realizar el scraping de datos de FBREF, se crearon widgets\* personalizados(\*urls dentro de la propia página que dan acceso a determinados datos en función de los filtros que se apliquen) para obtener todos las datos generadas por un jugador perteneciente a las 5 ligas Europeas de Fútbol más importantes: La Liga(España) , Serie A(Italia), Ligue 1(Francia), Bundesliga(Alemania).

Ejemplos de widget que se ha creado en FBREF:

* 'https://widgets.sports-reference.com/wg.fcgi?css=1&site=fb&url=%2Fen%2Fcomps%2F{league\_name\_short}%2F{season}%2Fshooting%2Fplayers%2F{season}-{league\_name\_long}&div=div\_stats\_shooting'

Con este widget se accede al código html en forma de tabla de FBREF donde se proporcionan las estadísticas de disparos en función de la temporada y liga que se desee. En mi caso, realizo una extracción por temporada para no saturar el server y tener más controlada la ejecución en Collab y cuanto a las ligas se seleccionaron las 5 más importantes de Europa ya mencionadas anteriormente.

Teniendo esto la extracción es muy sencilla, basta con utilizar la función read\_html de pandas pasándole como input el link del widget, y generar un Dataframe con la salida.

En este ejemplo de widget se obtenían estadísticas de disparo, pero también se han obtenido otro tipo de estadísticas como por ejemplo de posesión, regate, defensa, tiempo de juego o incluso estadísticas de porteros los cuales pueden generar otros datos que los jugadores de campo no generan como por ejemplo: % de paradas, precisión en saque de puerta o goles encajados.

Finalmente se realiza una concatenación de todos los dataframes obtenidos por cada widget y se obtiene como salida un dataframe final con una gran cantidad de variables, que se exporta en csv y está preparado para el posterior preproceso y transformación.

ANALISÍS DE SENTIMIENTOS DE TWEETS MEDIANTE CLASIFICACIÓN BINARIA CON RED NEURONAL CONVOLUCIONAL

1. Primero obtengo un data set de tweet con dos variables:
   1. Tweet: el propio texto del tweet
   2. Predicción: cuyo valor será 1 si el tweet expresa un sentimiento negativo o 0 si si es positivo.
2. Realizo una limpieza de los tweets: paso el tweet entero a minúsculas, elimino urls, números, caracteres especiales, todo ello con expresiones regualares o funciones de Python tipo replace, Split, etc
3. Aplico lematización a las palabras con Spacy. Esto se basa en obtener la raíz de una palabra para tener un tweet formado por palabras claves, lo cual será más fácil de analizar semánticamente.
4. Convierto los tweets a Word vectors de Spacy, esto son representaciones en forma de vector de las palabras. La distancia de vectores indica la similitud entre palabras, esto por ejemplo nos permitiría saber si dos frases totalmente distintas sintácticamente pueden estar expresando algo similar. Además este formato facilitará el entrenamiento del modelo.
5. Se preparan los datos para la red neuronal convolucional:
   1. Hay que aumentar la dimensión de los vectores para poder utilizar redes convolucionales, uso reshape.
   2. La variable de predicción tiene que ser categórica, uso to\_categorical de keras.
   3. Divido los datos entrenamiento y test.
6. Preparo el modelo:
   1. Creo un modelo secuencial
   2. Utilizo el optmizador del gradiente descendente SGD, para controlar el error con momentum nesterov de 0.9 .
   3. Añado dos capas Conv1d de 128 y 64 nodos, y tres capas dense de 128, 64 y 2 nodos, todo con dropout de 0.25
   4. Utilizo un input\_shape de 300, y una tasa de aprendizaje del 0.01
   5. Utilizo callbacks para mejorar el ajuste del modelo como: ReduceLROnPlateau, ModelCheckPoint y EarlyStoping.
   6. Entreno el modelo durante 200 iteraciones
7. Con el modelo ya entrenado compruebo como ha sido su rendimiento comparando los la variable de predicción de los datos de entrenamiento con la que ha predicho mi modelo consiguiendo una buena puntuación y un buen ajuste.

Todo esto lo muestro en unos gráficos y en la matriz de confusión donde se pueden ver cuántos positivos y negativos ha acertado y cuantos ha fallado.

1. Por último hago una predicción en los datos de test para ver como se comporta el modelo en datos que no conoce y obtengo buenos resultados.