UNIVERSIDAD AUTÓNOMA DE MADRID

ESCUELA POLITÉCNICA SUPERIOR

Máster en Big Data y Data Science: ciencia e ingeniería de datos

TRABAJO FIN DE MÁSTER

SISTEMA DE PREDICCIÓN DE RESULTADOS EN EVENTOS DEPORTIVOS

Guillermo Arrabal Martínez
Tutor: Iván Gonzalez

Junio 2020

Sistema de Predicción de Resultados en Eventos Deportivos

AUTOR: Guillermo Arrabal Martínez
TUTOR: Iván González

Escuela Politécnica Superior
Universidad Autónoma de Madrid
Junio de 2020

Resumen

Este Trabajo de Fin de Máster se centra en la obtención de pronósticos deportivos lo más certeros posibles y su posterior aplicación al mundo de las apuestas deportivas. El deporte, la tecnología y las apuestas están en pleno desarrollo, vamos a tratar de unirlos con el objetivo de analizar y predecir las probabilidades para los diferentes resultados deportivos. El deporte en el que voy a basar mi trabajo es el fútbol, en concreto el fútbol masculino tanto por afinidad como por facilidad para encontrar información.

Para ello vamos a tratar de analizar información histórica de resultados reales de partidos de fútbol desde la temporada 2008-2009 de las principales ligas de Europa.

También voy a incluir información de atributos de los jugadores que componen cada uno de los equipos, donde se detallan las cualidades técnicas y habilidades de cada uno de ellos. Estos datos los he extraído de EA Sports - FIFA, se trata de la empresa y uno de los videojuegos de fútbol más completos y prestigiosos del mercado.

En tercer lugar, voy a utilizar información de las principales casas de apuestas del mercado, donde podremos ver las cuotas a las que se pagan los tres posibles resultados de un partido; victoria, empate o derrota del equipo local.

El objetivo es encontrar un método de apuestas que nos permita "*ganar a la banca*" y conseguir sacar un rendimiento ya sea emocional, en el caso de hacer apuestas con moneda ficticia, o económico, en el caso de realizar apuestas con dinero real. Apostar o ganar una apuesta puntualmente puede ser muy sencillo, pero establecer un método sostenible y viable a largo plazo se plantea mucho más complicado. Hay un dicho que dice que siempre gana la banca, pues bien, vamos a tratar de desmentirlo aplicando las técnicas de Data Science que hemos aprendido a lo largo del Máster. Posteriormente plantearemos una arquitectura de forma teórica que podría soportar y desarrollar este trabajo en un entorno de producción.

Este modelo de trabajo podríamos aplicarlo en cualquier otro deporte colectivo con características similares, como podría ser el balonmano, voleibol, hockey, waterpolo, incluso el baloncesto. Para ello lógicamente deberíamos realizar importantes ajustes en el proceso, pero la esencia sería la misma.

Agradecimientos

Quiero agradecer en primer lugar a mi pareja Irene N.B, que me ha apoyado, me ha ayudado y me ha aguantado a lo largo de todo el Máster, con largos fines de semana trabajando en prácticas, estudiando y asistiendo a las clases. Su aportación a este proyecto y a la consecución del máster ha sido de vital importancia. También quiero agradecer a mis padres todo el apoyo que siempre me han brindado en el desarrollo de mi educación animándome a afrontar cualquier desafío, exigiéndome esfuerzo y facilitándome la vida durante mis años de formación académica.

Por otra parte, quiero agradecer a Iván González, que desde el primer momento no ha dudado en guiarme como tutor y me ha animado con mi idea, aportándome las pautas necesarias, el soporte y me ha marcado el camino para poder desarrollar mi trabajo con éxito. A mis compañeros de máster por esas conversaciones interminables tratando de resolver dudas, infinidad de pantallazos, líneas de código y maravillosos desayunos de sándwich mixto los sábados por la mañana.

Por último, quiero agradecer a todos los profesores que me han dado clase en estos dos largos años, por su dedicación, esfuerzo y todo lo que me han enseñado.

INDICE DE CONTENIDOS

INDICE DE FIGURAS

INDICE DE TABLAS

1 Introducción

1.1 Motivación

Esta memoria de TFM trata de juntar dos ámbitos que me generan un especial interés; el fútbol y las apuestas.

Mi idea inicial era poder desarrollar un proyecto de negocio. Desarrollar una idea, plantear un proyecto, desarrollarlo y valorar su posible ejecución y cabida en el mercado actual. Pero mi escasa experiencia real en el mundo del Big Data y el '*Internet of Things',* que es donde quería desarrollar la idea, tenía algunas lagunas por lo que he optado por descartar esta opción.

Por otro lado, tenía pensado lanzar algún proyecto en el entorno laboral que me permitiera optimizar los tiempos, aprovechando esfuerzos laborales en el desarrollo de mi proyecto, pero tampoco ha sido posible por la presión, la carga de trabajo diario y los cambios continuos que está experimentando la empresa actual donde presto mis servicios.

Por lo tanto, he tratado de encontrar una temática que me motive a nivel personal para poder hacer un buen trabajo y de esta forma poder disfrutar de las horas invertidas en el mismo. La base del proyecto se va a centrar en mi deporte favorito, el fútbol, el cual practico activamente desde que tengo uso de razón. He competido tanto en ligas sociales como federado en la Comunidad de Madrid, he sufrido graves lesiones, me he recuperado y sigo jugando siempre que puedo, sigue siendo un deporte que me apasiona incluso cuando lo veo por la televisión.
Las apuestas hoy en día van de la mano del fútbol, pueden hacer que un partido irrelevante por los equipos que se enfrentan o por la falta de interés que te genera se convierta en un partido frenético y lo vivas con entusiasmo llegando a celebrar incluso un saque de esquina. Además, las apuestas tienen el punto de poder compartir tus experiencias con los amigos, penas y glorias incluidas.

Después de toda la carga de trabajo que hemos tenido a lo largo de los dos años de máster, sumado a la situación excepcional que hemos vivido los últimos meses con la pandemia mundial provocada por el Covid-19, ha sido difícil encontrar las fuerzas para trabajar en mi TFM. Demasiado tiempo trabajando y estudiando en el mismo lugar, pero la elección del tema ha sido clave para encontrar la pequeña motivación que necesitaba.

1.2 Objetivos

El objetivo de este proyecto es el diseño, implementación y validación de un método de predicción de resultados en eventos deportivos y su posterior aplicación a las apuestas deportivas.

En concreto vamos a trabajar en la creación de modelos predictivos en el fútbol masculino basándonos en las grandes ligas europeas.
Una vez tengamos nuestro mejor modelo predictivo, lo aplicaremos a las cuotas de mercado propuestas por las casas de apuestas y de esta forma poder analizar si nos saliera rentable económicamente su aplicación. De esta forma podremos justificar o no el desarrollo del proyecto basándonos en su rentabilidad económica.

Este punto es uno de los grandes escollos en los proyectos de Big Data, justificar el retorno de la inversión y la viabilidad del proyecto.

1.3 Organización de la memoria

La memoria consta de los siguientes capítulos:

- **Problema inicial e ingesta de datos.**
  En este primer punto vamos a definir que retos queremos afrontar, cual es el problema al que queremos poner solución, que datos vamos a utilizar, como están estructurados, y como hemos realizado el proceso de ingesta de los datos.
- **Construcción de atributos predictivos o feature engineering.**
  En este punto, vamos a crear todas las variables necesarias para poder desarrollar los modelos predictivos, creando nuevas estadísticas y resumiendo los atributos de los jugadores.
- **Preprocesado y auditoría.**
  En este punto vamos a definir qué técnicas de preprocesado vamos a utilizar, los pasos en falso que hemos dado, reducción de dimensionalidad, transformación e integración de los datos.
- **Modelos probabilísticos y aprendizaje automático**
  En este punto vamos a probar diferentes modelos probabilísticos, optimización de parámetros, modelos con reducción de dimensionalidad, balanceo de clases. Finalmente vamos a comparar los modelos y nos quedaremos con el mejor.
- **Aplicación del modelo elegido en las casas de apuestas - Dashboard**
  En este punto aplicaremos el mejor modelo elegido y valoraremos económicamente si es viable su aplicación o si por el contrario no tiene sentido.
- **Conclusiones y trabajo futuro**
  En este punto expondremos todas las conclusiones que hayamos sacado a lo largo del desarrollo del proyecto y los potenciales próximos puntos de investigación a seguir para desarrollar el proyecto.
- **Bibliografía y Anexo**

2 Problema Inicial

El problema inicial que me he planteado para desarrollar este trabajo es el siguiente; ¿es posible utilizar todos los datos que tenemos disponibles para definir un modelo predictivo con el que apostar de forma sostenible a medio o largo plazo? Como hemos comentado en los objetivos, vamos a tratar de desarrollar e implementar una herramienta que pueda predecir los resultados de los eventos deportivos y validaremos su viabilidad económica.

2.1 Contexto e Historia de las Apuestas

Antes de entrar en materia vamos a contextualizar un poco la historia de las apuestas para dejar claro que existen hace muchos años y que no son un fenómeno de reciente aparición fruto de las nuevas tecnologías. Se han desarrollado a una velocidad de vértigo a lo largo de los últimos años y mueven anualmente grandísimas cantidades de dinero, solamente en España hay registradas cerca de 80 casas de apuestas u operadores con licencia [2].

Se trata de uno de los entretenimientos o pasatiempos más populares del mundo. Con el desarrollo de las nuevas tecnologías y la facilidad de acceso para todo el mundo con los *Smartphones*, los usuarios tienen la facilidad de apostar en cualquier momento a cualquier deporte deseado independientemente de donde se juegue o donde esté ubicado el usuario con una rapidez y una facilidad asombrosa.

El origen de las antiguas apuestas se remonta a la civilización griega, donde se celebraban las olimpiadas y los allí presentes elegían cual era a su entender el mejor deportista que iba a ganar una competición determinada y apostaban por el mismo [1]. Esta actividad pasó a manos de los romanos, que celebraban combates de gladiadores donde la gente apostaba quien sería el vencedor. El atletismo fue el primer deporte donde se realizaron apuestas. Ya en la civilización moderna, las carreras de caballos tomaron una relevancia especial en la sociedad británica donde se hicieron populares en la clase alta y en los aristócratas. Con el paso de los años han evolucionado tanto los deportes a los que apostamos como la forma en la que apostamos.

La primera casa de apuestas presencial que se estableció en España fue en 2008 y se llamaba Victoria, fruto del acuerdo entre dos gigantes de las apuestas, la multinacional española Codere y la británica William Hill [3]. El sector del juego supone el 0,9% del PIB en España y emplea cerca de 84.700 personas (cifras 2018), con una facturación de más de 9.400 millones de euros [4].

Actualmente el organismo que regula las apuestas en España es la Dirección *General de Ordenación del Juego*, donde se marcan las reglas que deben seguir las casas de apuestas y los derechos de los usuarios de sus servicios. Tienen competencia estatal los juegos de azar online, juegos de casino, póquer, máquinas de azar y concursos. Tienen competencia autonómica los juegos presenciales de casino, bingo, máquinas de azar, apuestas y loterías de ámbito autonómico.

2.2 Como funcionan las apuestas.

Las apuestas operan básicamente en dos canales, *físico y online*. Puedes acudir físicamente a una casa de apuestas y realizar una apuesta o puedes realizarla Online, registrándote en una de las casas de apuestas. El resultado es el mismo, lo único que cambia es la experiencia del usuario. Los usuarios pueden acudir a cualquiera de estos canales y apostar su dinero en diferentes deportes.
Las apuestas pueden ser *en vivo* (apostar a un evento que se está celebrando en el momento que se efectúa la apuesta) *o pueden realizarse previamente a la celebración del evento*. Lo que varía en este caso son las cuotas que ofrecen las casas de apuestas en función de los eventos y la información que tienen disponible.
En función de las casas de apuestas, las cuotas que ofrecen pueden ser *decimales*, como es el ejemplo de Europa y en las cuales vamos a basar este trabajo, o *fraccional*, utilizadas fundamentalmente en Reino Unido.
La forma en la que vamos a calcular **las ganancias** que nos puede generar una apuesta es la siguiente:

G = Ganancias
q = cuota $G = q * m$
m = montante apostado
Por ejemplo, si apostamos 10€ a una cuota de 1.5€: G = 1.5*10 = 15€
De esta forma obtenemos las ganancias, no el beneficio neto, para calcularlo tenemos que restarle al beneficio bruto la cantidad invertida en la apuesta.

B = beneficio $B = m * (q - 1)$
Por lo tanto, ahora podemos calcular **el beneficio** de nuestra apuesta: B = 10 * (1.5 – 1) = 5€
Las casas de apuestas lo que hacen es apostar contra el usuario, calculan sus propias probabilidades de ganar la apuesta y las ofrecen en forma de cuotas para que los usuarios consideren si la apuesta es atractiva para ellos. En función de la casa de apuestas, las cuotas varían, una casa de apuestas consolidada suele ofrecer cuotas más bajas que una casa de apuestas nueva o emergente, que quiere atraer a nuevos usuarios o clientes y ofrece recompensas más suculentas.
Las cuotas que ofrecen las casas de apuestas y que calculan con actualizaciones en tiempo real, estiman una probabilidad para cada uno de los acontecimientos que en ellas se ofrecen.
Todas las casas de apuestas ofrecen múltiples posibilidades para poder apostar en sus eventos:

- **Apuesta Sencilla**. Es la más común, elegimos un evento en un deporte concreto y apostamos directamente sobre el mismo. En este trabajo nos vamos a centrar en este tipo de apuesta y más en concreto en predecir el resultado final de un partido de fútbol.

- **Apuesta combinada.** En este tipo de apuesta combinamos varias apuestas sencillas, las cuotas se multiplican entre sí y las ganancias pueden ser exponencialmente superiores. Solo se cobran en el caso de acertar todos los eventos incluidos en la misma.

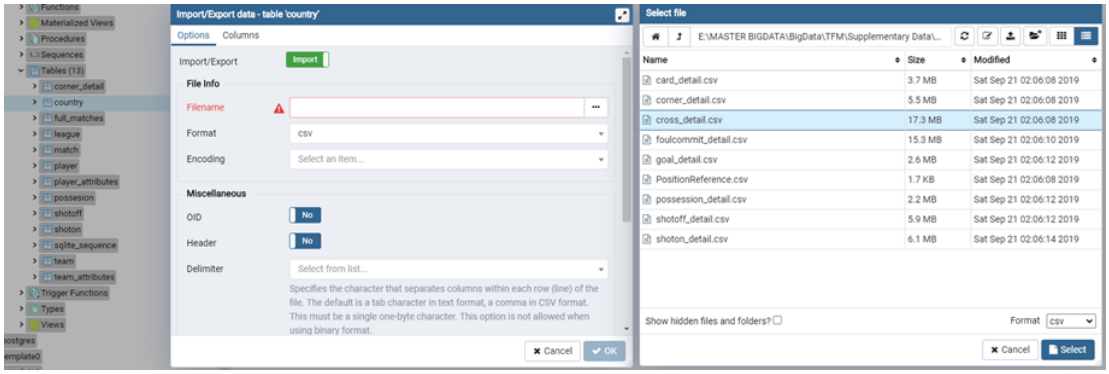
Por ejemplo, apostamos sobre tres partidos de fútbol diferentes con las siguientes cuotas y ganancias potenciales: G = 1.53*1.82*2,25 *m
La cuota final sería 6,26, si la cantidad apostada es de m = 10€: G = 6,26*10 = 62.6€

Las principales casas de apuestas que vamos a valorar en este trabajo son las siguientes:
Bet365, Bet & Win, Interwetten, William Hill, Ladbrokes, Pinnacle y VcBet. Vamos a trabajar la información de forma que sea posible elegir sobre qué casa de apuestas queremos apostar y podamos visualizar cuales serían nuestros resultados o beneficios.

2.3 Ingesta de datos

En este punto, vamos a ver como se ha realizado la ingesta de datos, algunas modificaciones en la información que hemos realizado y el origen de los datos que estamos utilizando.

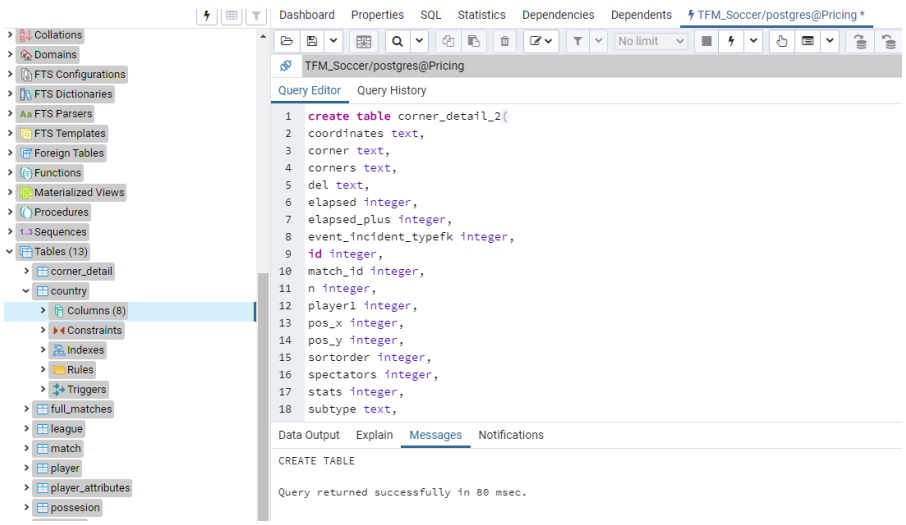
Podemos encontrar datos sobre fútbol con facilidad, pero habitualmente están desestructurados y esparcidos por diferentes webs con lo que no ha sido sencillo encontrar una fuente de datos para comenzar a trabajar. Finalmente he encontrado una web de origen británico, que me ha servido como fuente de información de datos históricos. Football-Data Uk [7]. Para la obtención de los atributos de los jugadores hemos utilizado la web Sofifa [8], la cual tiene valoraciones de los principales atributos de los jugadores y equipos con varias actualizaciones por temporada propiedad de EA Sports Fifa Games.
Navegando en diversas páginas y a través de un perfil de GitHub [9] donde ya se ha trabajado la información y se han investigado distintas fuentes, he podido descargar una batería de ficheros "csv" a partir de los cuales voy a construir una base de datos SQL.
Voy a utilizar Postgres SQL como base de datos, ya que estoy bastante familiarizado con su uso y cumplía los dos requisitos principales para avanzar en el proyecto; facilidad a la hora de importar ficheros para la creación de tablas y facilidad de integración con Python, que va a ser mi principal herramienta de analítica de la información.



1.Figura 2.3 Requisito facilidad ingesta.

2.3.1 Carga de Datos

El volumen de los datos (apenas son 600MB de información) y la infraestructura con la que cuento (un ordenador portátil), me han llevado a desarrollar la parte técnica en un entorno de desarrollo muy sencillo donde voy a efectuar la ingesta de datos en una base de datos Postgres SQL desde ficheros 'csv', posteriormente analizaré la información en Python y finalmente voy a crear un Dashboard en Microsoft Power Bi. En el caso de que el proyecto lo justifique y como una línea de desarrollo de este trabajo en el futuro, voy a plantear de forma teórica una arquitectura mucho más completa y con mayor capacidad de almacenamiento y cómputo más acorde a un entorno Big Data (Punto 7.2.1).
La carga de datos la realizamos desde la propia interfaz de PostgreSQL PgAdmin donde podemos interactuar de forma visual con nuestros datos. Para conocer la información contenida en cada uno de los ficheros que voy a utilizar consultar el Anexo A.
Se trata de información histórica de partidos de fútbol, estadísticas de cada uno de ellos, estadísticas de las cualidades de los jugadores y atributos cualitativos de los equipos y países. Diseñamos los scripts correspondientes para crear las tablas en nuestra base de datos Postgres, los ejecutamos en el Query editor y con ayuda del intérprete de carga de ficheros convertimos los csv en tablas de la base de datos (Anexo B, Scripts de Ingesta).



2.Figura 2.3 Creación de tablas en la base de datos.

Este proceso requiere de un análisis minucioso de los tipos de datos contenidos en cada columna de los ficheros csv, adecuando los scripts de creación de tablas para evitar errores cando importamos los datos.

2.3.2 Join de obtención de tabla maestra

Una vez tenemos todas las tablas cargadas en la base de datos, vamos a crear una nueva tabla maestra de partidos que será la base de nuestra parte analítica.
En esta tabla vamos a hacer diversos joins entre la tabla original de partidos y las tablas de corners, posesión, tarjetas, disparos a puerta y disparos fuera de la portería. La información no está estructurada al mismo nivel, en nuestra tabla de partidos tenemos una línea por partido con todos los detalles, en cambio en las tablas complementarias de disparos a puerta o tarjetas, tenemos multitud de registros para cada partido con una línea por disparo a puerta con los detalles del jugador o el minuto en el que se ha efectuado. Lo mismo ocurre con la tabla de posesión. Este join ha requerido de un trabajo previo en el que he realizado varios GroupBy para cada una de las tablas adecuándolas para tener una sola línea por partido.

- Tabla de Corners, agrupamos contando el número de corners por partido.
- Tabla de Posesión, tenemos registros de posesión 4 veces por tiempo, un total de 8 veces por partido. Agrupamos con el minuto más alto para cada partido, de esta forma obtendremos la posesión al final del encuentro.
- Tabla de disparos a puerta y disparos fuera, también agrupamos por número de disparos por partido.

Finalmente realizamos una consulta que englobe estas tres tablas agrupadas y haga un join sobre la tabla de partidos, con la salida de la query creamos la nueva tabla llamada “full_matches”.

```
--match shots
select*from(
--MATCH JOINED WITH CORNERS
select*from(
--TABLE A = MATCHES
        select*from match as a
left join
--TABLE B = CORNERS
            (select match_id, count(id) as corners from corner_detail group by match_id) as b
on a.id = b.match_id) as x
left join
--TABLE c = POSSESION
                    (select match_id,awaypos,homepos, max(elapsed) minuto from possesion group by match_id) as c
on x.id = c.match_id) as z
left join
--TABLE Y = SHOTSONYOFF
                        (select v.match_id, shotson, shotsoff from (select match_id, count(id) as shotson from shoton group by match_id) as v
                        left join (select match_id, count(id) as shotsoff from shotoff group by match_id) as w
                        on v.match_id = w.match_id) as y
on z.id = y.match_id
```

3.Figura 2.3.2.A Creación de la tabla maestra de partidos.

Confirmamos que se mantiene intacto el número de partidos inicial, 25.979 partidos, y que hemos incrementado el número de columnas de información. Luego veremos cómo hay algunos campeonatos en los que esta información no está muy completa.

2.3.3 Integración con nuestra herramienta de análisis

Una vez tenemos nuestra base de datos preparada y completa, vamos a importar las librerías psycopg2 y pandas.io.sql, que nos permiten ejecutar consultas o querys SQL sobre nuestra base de datos PostgreSQL de una forma muy sencilla y convertir las salidas de las consultas en Dataframes de Python con los que poder desarrollar toda la parte analítica.

```
In [4]: import psycopg2
        import pandas.io.sql as psql

        connection = psycopg2.connect(user = "postgres", password = "guille12", host = "127.0.0.1", port = "5432",
                                      database = "TFM_Soccer")
        df_original = psql.read_sql("select*from full_matches", connection)
        df_original.shape
        df_original.head(3)

Out[4]: (25979, 124)
```

Out[4]:

	id	country_id	league_id	season	stage	date	match_api_id	home_team_api_id	away_team_api_id	home_team_goal	...	bsa	match_id	corners	m
0	1	1	1	2008/2009	1	2008-08-17 00:00:00	492473	9987	9993	1	...	4,2	None	None	
1	2	1	1	2008/2009	1	2008-08-16 00:00:00	492474	10000	9994	0	...	3,6	None	None	
2	3	1	1	2008/2009	1	2008-08-16 00:00:00	492475	9984	8635	0	...	2,75	None	None	

3 rows × 124 columns

4.Figura 2.3.3 Ejemplo integración Postgres & Python.

La integración es sencilla y directa debido al volumen de datos con el que contamos. Si llegado el caso el volumen de los datos es superior no podremos permitirnos el lujo de cargar una consulta SQL de un determinado volumen en un Dataframe, por lo que tendremos que pensar en utilizar alguna herramienta que nos permita distribuir tareas en un clúster y que sea muy veloz; el principal candidato en este caso sería Spark y usaríamos su API de Python (PySpark) por ejemplo.

2.3.4 Datos futuros

De cara al futuro, una opción muy buena sería ampliar el horizonte de nuestras predicciones para realizar apuestas en vivo, donde hay muy buenas oportunidades en cuanto a la valoración de las cuotas. Las casas de apuestas tienen potentes mecanismos de reajuste de las cuotas, pero si estamos atentos siempre hay buenas oportunidades. En un futuro y teniendo en cuenta mi intención de que las predicciones sean lo suficientemente buenas para justificarlo, podríamos ampliar el volumen de datos incluyendo información de eventos en tiempo real, información de jugadores lesionados, información relativa a noticias positivas o negativas que aumenten la presión sobre determinados futbolistas incluso podríamos incluir información meteorológica para saber cuál será el estado del terreno de juego. Dejamos la puerta abierta para cuando tengamos un volumen suficiente, por qué no, pensar en montar un clúster con Hadoop donde podamos almacenar nuestra información en su sistema de ficheros distribuido HDFS, que nos garantizaría un sistema escalable, con alta disponibilidad, facilidad de replicación de nuestros datos y tolerante a fallos.

3 Feature Engineering

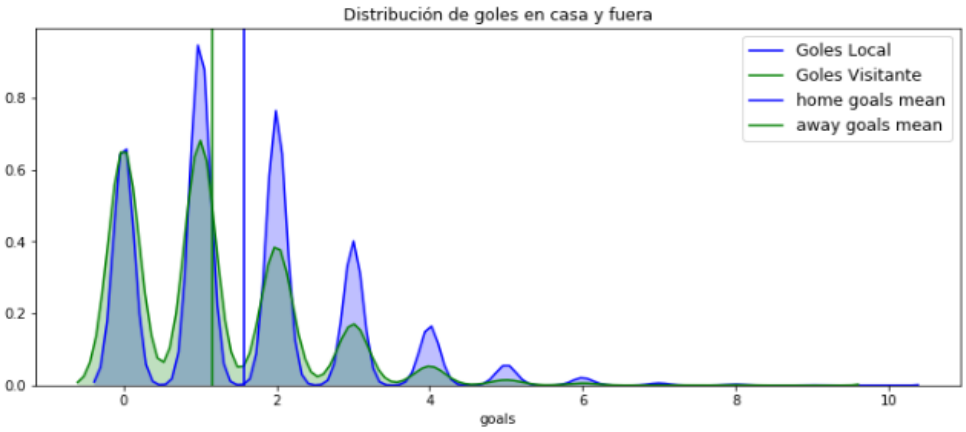
Antes de pasar a la fase de limpieza de nuestros datos vamos a crear nuevos atributos que van a ser de vital importancia a la hora de trabajar en nuestras predicciones. Tenemos mucha información que no es válida para realizar predicciones, o lo que llamaríamos "False Predictors" como por ejemplo los goles marcados por el equipo local, goles del equipo visitante, posesión de ambos equipos a lo largo del partido… Pero si podemos crear nuevos atributos que nos digan cuantos goles ha marcado un equipo en los últimos partidos o cuanta posesión ha tenido un equipo en los últimos partidos.

3.1 Atributos de Partidos

La primera variable que vamos a crear es nuestra variable "**target**", que va a definir el resultado del equipo local con tres etiquetas:

- 1: Victoria del equipo local
- 0: Empate
- -1: Derrota del equipo local

En segundo lugar, vamos a crear una serie de variables que nos van a ayudar a saber si un equipo está en racha o si tiene mucha facilidad para marcar goles. Van a ser las siguientes, todas ellas detalladas tanto si los equipos juegan en casa como si juegan a domicilio:
Número de goles anotados y recibidos por el equipo en los últimos *n* partidos, diferencia de goles en los últimos *n* partidos, posesión que ha tenido el equipo tanto de local como de visitante en los últimos *n* partidos, partidos ganados como local y como visitante en los últimos *n* encuentros, últimos 4 enfrentamientos directos entre los dos equipos analizados.
Una vez realizado todo el análisis he descubierto que el modelo tenía problemas a la hora de acertar los empates después de revisar en detalle la matriz de confusión, por lo que he añadido el número de partidos empatados en las últimas *n* jornadas.



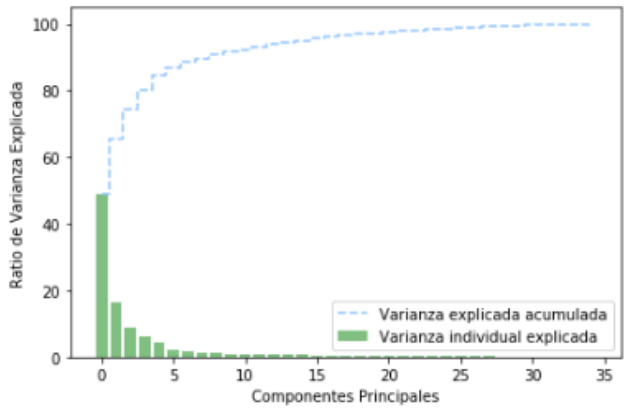
5.Figura 3.1.A Distribución de goles para equipos local y visitante.

3.2 Atributos de Jugadores

En este punto voy a tratar de decidir qué variables de los jugadores debo utilizar, he realizado un proceso de limpieza previo que comentaré en el siguiente punto, y después he aplicado un método de reducción de la dimensionalidad para tratar de detectar cuales son los atributos más relevantes. Voy a intentar hacer PCA en los atributos de los jugadores, tenemos 22 jugadores con 35 atributos cada uno, lo que hace un total de 770 atributos por partido, es

mucha información y para facilitar la computación de nuestros algoritmos de predicción vamos a tratar de reducir la dimensionalidad de los datos.
Primero estandarizamos nuestras estadísticas o valoraciones de los jugadores. Vamos a normalizar los datos para que sus valores estén comprendidos entre 0 y 1. Es una buena praxis para evitar que las distintas características de nuestro dataset estén expresadas en escalas diferentes, este no es el mejor ejemplo ya que todos los jugadores están expresados en una escala del 1 al 100 pero aun así vamos a aplicarlo. PCA asume que los datos con los que trabaja tienen una distribución gaussiana o normal, por lo que debemos ajustar nuestros datos para que la media sea igual a 0 y su varianza igual a 1. En Scikit-Learn tenemos la funcionalidad Standard Scaler que nos ayudara en esta tarea. Los autovectores son las direcciones en las cuales la varianza de nuestros datos es superior. La varianza en una variable aleatoria es una medida de dispersión que define la esperanza del cuadrado de la desviación de dicha variable respecto a su media. Estas direcciones o autovectores representan la información más significativa de nuestros datos, por este motivo los llamamos componentes principales (Principal Component Analysis). Los autovalores representan cuanto mide la varianza sobre esos autovectores. Por lo que para encontrar las componentes principales, calcularemos primero la matriz de covarianzas que nos dará la medida de dispersión entre las variables.

Para reducir la dimensionalidad del dataset, debemos descartar los autovectores cuyos autovalores son más bajos, son los que menos información aportan al conjunto total. Ordenamos el conjunto por parejas de autovectores y autovalores para ver si podemos responder a la pregunta de cuantas componentes debemos utilizar. La clave es ver si podemos expresar la esencia del dataset con el menor número de componentes principales, esto lo realizaremos visualizando la varianza explicada:



6.Figura 3.2 Varianza explicada atributos jugadores.

Para alcanzar una representatividad adecuada deberíamos considerar un mínimo de un 75-80% de la varianza explicada, lo cual nos supondría un mínimo de 3 componentes. Esto nos supone 66 atributos, siguen siendo muchos por lo que finalmente he optado por utilizar solamente el “overall_rating” de los 22 jugadores que participan en cada partido. A decir verdad, no puedo incluir más atributos, computacionalmente me da error de memoria en mi portátil al realizar el cruce de información que explico a continuación.
Creamos un Dataframe con los atributos de los 22 jugadores que participan en cada partido. Esta tarea es bastante costosa, su ejecución lleva unos 10-12 minutos ya que tiene que iterar en los 25K partidos, sobre sus 22 columnas y cruzarlo con la tabla de jugadores que contiene más de 180K registros. El punto clave es que hemos cruzado los atributos del jugador respecto a la fecha más cercana inferior o igual a la fecha del partido, lo cual añade

complejidad adicional. Tenemos incluso varias valoraciones para cada jugador a lo largo de una temporada, no podemos asignar una valoración de un jugador realizada en 2012 para un partido de 2014. El output es un Dataframe de 8.500 partidos, para el resto no tengo el detalle de los jugadores que han participado en el mismo.

4 Preprocesamiento y Auditoria

Esta parte es fundamental en nuestro análisis, y en el descubrimiento de información que tiene como objetivo la mejora de la calidad de nuestros datos y nos permite desarrollar predicciones más robustas. Limpieza de datos, integración, transformación y reducción de la dimensionalidad suelen ser sus procesos principales.

4.1 Integración de datos

En primer lugar, vamos a integrar en un único conjunto de datos toda la información que vamos a utilizar. En nuestro proyecto no disponemos de diversas fuentes de datos que vayan a requerir de un proceso ETL (Extract Transform Load), si en un futuro se amplía el número y la variedad de fuentes de datos desde las que vamos a trabajar será necesario definir estos procesos para la integración.
En nuestro caso, para la integración vamos a cruzar nuestra tabla maestra de datos de los partidos con los dos dataframes de nuevos atributos que hemos creado en el punto anterior de diseño de características. Realizaremos esta operación mediante varios join utilizando el atributo id único de cada partido. Nos traemos los atributos de los últimos 10 partidos y los atributos de los 22 jugadores.
También vamos a traernos el nombre del país mediante la tabla Country y el "country_id".
Hacemos lo mismo con el nombre de la liga, tabla League, campo "country_id".
Por último, nos traemos los nombres de los equipos que se enfrentan, esta información la traemos de la tabla "teams" mediante los identificadores tanto de equipo local como de equipo visitante.

4.2 Limpieza de datos

En esta fase del preprocesado vamos a tratar la eliminación de valores erróneos, también vamos a tratar de eliminar el ruido u outliers. Vamos a eliminar todos los datos que se encuentren fuera de sus intervalos de validez y trataremos de corregir posibles inconsistencias.
Consultamos todas las tablas que hemos creado en nuestra base de datos. Las tablas relevantes que vamos a utilizar son: "Player_Attributes" y "full_matches". Las tablas de jugadores, partidos, ligas y países solamente las utilizaremos para integrar información tal y como hemos detallado en el punto anterior. Las tablas de corners, posesión, disparos a puerta y disparos fuera ya las hemos integrado en la tabla de partidos llamada "full_matches".

	type	name	tbl_name	rootpage	sql
1	table	Player_Attributes	Player_Attributes	11	CREATE TABLE "Player_Attributes" (\n\t`id`\tIN...
2	table	Player	Player	14	CREATE TABLE `Player` (\n\t`id`\tINTEGER PRIMA...
3	table	Match	Match	18	CREATE TABLE `Match` (\n\t`id`\tINTEGER PRIMAR...
4	table	League	League	24	CREATE TABLE `League` (\n\t`id`\tINTEGER PRIMA...
5	table	Country	Country	26	CREATE TABLE `Country` (\n\t`id`\tINTEGER PRIM...
6	table	Team	Team	29	CREATE TABLE "Team" (\n\t`id`\tINTEGER PRIMARY...
7	table	Team_Attributes	Team_Attributes	2	CREATE TABLE `Team_Attributes` (\n\t`id`\tINTE...
8	table	corner_detail	corner_detail	305672	CREATE TABLE corner_detail(\ncoordinates text,...
9	table	possesion	possesion	310713	CREATE TABLE possesion(\nawaypos integer,\ncar...
10	table	shoton	shoton	312783	CREATE TABLE shoton(\nblocked text,\ncard_type...
11	table	shotoff	shotoff	318400	CREATE TABLE shotoff(\ncard_type text,\ncoordi...
12	table	full_matches	full_matches	323810	CREATE TABLE full_matches (\n id ...

1.Tabla 4.2.A Visualización de las tablas disponibles en la base de datos.

He comenzado analizando la tabla de atributos de jugadores "*player_attributes"*. Mediante la función describe de pandas, que nos facilita algunas estadísticas descriptivas; como el número de registros, la media, el máximo, el mínimo, desviación típica y los 3 cuartiles de cada columna del Dataframe o atributo.

	count	mean	std	min	25%	50%	75%	max
id	183978.0	91989.500000	53110.018250	1.0	45995.25	91989.5	137983.75	183978.0
player_fifa_api_id	183978.0	165671.524291	53851.094769	2.0	155798.00	183488.0	199848.00	234141.0
player_api_id	183978.0	135900.617324	136927.840510	2625.0	34763.00	77741.0	191080.00	750584.0
overall_rating	183142.0	68.600015	7.041139	33.0	64.00	69.0	73.00	94.0
potential	183142.0	73.460353	6.592271	39.0	69.00	74.0	78.00	97.0
crossing	183142.0	55.086883	17.242135	1.0	45.00	59.0	68.00	95.0
finishing	183142.0	49.921078	19.038705	1.0	34.00	53.0	65.00	97.0
heading_accuracy	183142.0	57.266023	16.488905	1.0	49.00	60.0	68.00	98.0
short_passing	183142.0	62.429672	14.194068	3.0	57.00	65.0	72.00	97.0
volleys	181265.0	49.468436	18.256618	1.0	35.00	52.0	64.00	93.0
dribbling	183142.0	59.175154	17.744688	1.0	52.00	64.0	72.00	97.0
curve	181265.0	52.965675	18.255788	2.0	41.00	56.0	67.00	94.0
free_kick_accuracy	183142.0	49.380950	17.831746	1.0	36.00	50.0	63.00	97.0
long_passing	183142.0	57.069880	14.394464	3.0	49.00	59.0	67.00	97.0

2.Tabla 4.2.B Describe de la tabla player_attributes.

Eliminamos todo registro con valores mínimos por debajo de cero o máximos por encima de 100. Apreciamos que hay registros con NA. Diseñamos una función que nos facilita entender los valores dependiendo de su tipología; numéricos (entero o decimal) o categóricos.

Data Frame 'Matches' contains following columns of float data

	Unique Values	qty	na
potential	[71.0, 66.0, 65.0, 76.0, 75.0, 77.0, 78.0, 79....	57	836
overall_rating	[67.0, 62.0, 61.0, 74.0, 73.0, 71.0, 70.0, 69....	62	836
reactions	[47.0, 46.0, 67.0, 71.0, 69.0, 70.0, 66.0, 62....	79	836
jumping	[58.0, 85.0, 84.0, 77.0, 73.0, 64.0, 48.0, 65....	80	2713
balance	[65.0, 90.0, 87.0, 62.0, 92.0, 84.0, 56.0, 41....	82	2713
agility	[59.0, 78.0, 79.0, 81.0, 77.0, 74.0, 85.0, 76....	82	2713
strength	[76.0, 56.0, 50.0, 49.0, 48.0, 43.0, 37.0, 38....	83	836
stamina	[54.0, 79.0, 80.0, 77.0, 76.0, 74.0, 73.0, 63....	85	836
sprint_speed	[64.0, 78.0, 82.0, 81.0, 72.0, 47.0, 46.0, 59....	86	836
acceleration	[60.0, 79.0, 80.0, 84.0, 85.0, 69.0, 43.0, 42....	87	836
gk_handling	[11.0, 10.0, 7.0, 6.0, 22.0, 21.0, 12.0, 5.0, ...	91	836
aggression	[71.0, 63.0, 62.0, 68.0, 67.0, 66.0, 64.0, 60....	92	836
gk_reflexes	[8.0, 7.0, 12.0, 11.0, 22.0, 13.0, 21.0, 10.0,...	93	836
curve	[45.0, 44.0, 70.0, 68.0, 67.0, 66.0, 65.0, 63....	93	2713
gk_diving	[6.0, 5.0, 14.0, 13.0, 16.0, 15.0, 8.0, 7.0, 1...	94	836
volleys	[44.0, 43.0, 40.0, 32.0, 29.0, 28.0, 30.0, 52....	94	2713

Data Frame 'Matches' contains following columns of object data

	Unique Values	qty	na
home_status	[D, W, L]	3	0
season	[2008/2009, 2010/2011, 2011/2012, 2012/2013, 2...	8	0
Country	[England, France, Germany, Italy, Netherlands,...	8	0
League	[England Premier League, France Ligue 1, Germa...	8	0
Home_team_name	[Manchester United, Arsenal, Sunderland, West ...	174	0
Away_team_name	[Newcastle United, West Ham United, Hull City,...	175	0
date	[2008-08-17 00:00:00, 2010-08-16 00:00:00, 201...	1181	0

3.Tabla 4.2.C Función para detectar los valores NA.

Conocedores de la naturaleza de nuestros datos, tenemos 836 registros a los cuales les falta el "*overall_rating*" (podemos verlo en la tabla superior), que es el equivalente a la valoración final del jugador, todos esos registros los eliminamos. El resto de los atributos con valores NA son completados con la media de sus columnas correspondientes, veremos si tiene

sentido una reducción dimensional más adelante, en ese caso volveremos a este punto para tratar con más cuidado estos valores NA. La tabla final contiene registros de 180.354 jugadores. Este proceso lo hemos realizado previamente a la creación de los atributos de los jugadores mencionada en el punto 3.2 para evitar tener valores en blanco, pero estoy manteniendo la estructura del documento por eso se incluye en este epígrafe.

A continuación, hemos analizado la tabla de partidos "*full_matches*", en la cual hemos integrado toda la información que vamos a necesitar en nuestros algoritmos creando un Dataframe.

- En primer lugar, he eliminado los atributos que no vamos a necesitar, quitamos todos los id de país, de liga, de los 22 jugadores que componen los equipos local y visitante. Solamente vamos a mantener el índice de partido.
- En segundo lugar, hemos tratado los *"missing values"* en los atributos de los jugadores. Hay partidos en los que nos falta la nota de alguno de los jugadores o bien porque no se ha registrado esa información o bien porque no tenemos ese jugador en la base de datos. Hemos trabajado esos huecos haciendo un GroupBy con la temporada, el equipo, la posición y completamos con la media. Para aquellos campos en los que aun así nos ha quedado algún valor *"NaN"*, lo completamos con la media de la liga y la temporada en esa posición.

En cuanto a los *"missing values"* de posesión, donde no tengamos información o la posesión sea inferior al 10%; la cual considero el mínimo que debe tener un equipo por lo tanto sería un outlier, lo hemos completado con una posesión neutral, 50% para cada equipo.
Una vez tenemos este Dataframe limpio y con las columnas que vamos a utilizar le pasamos la función describe + transpose de pandas, que nos da una buena visión para comprobar si tenemos outliers.
Todos los *"missing values"* que tenemos en los campos correspondientes a las casas de apuestas son completados con ceros, no disponemos información de las cuotas en ese portal para ese partido en concreto. De las casas de apuestas más relevantes; Bet365 y William Hill tenemos información para más del 99,9% de los partidos.
Como en el caso de la tabla de atributos de jugadores, utilizamos la función diseñada para entender que valores contiene cada columna y el número de valores NaN contenidos en cada una de ellas, como hemos podido ver en el ejemplo de la Tabla 4.2.C.
Solamente nos quedan valores nulos en las columnas de disparos, las cuales completamos con la media de esa temporada en esa liga.

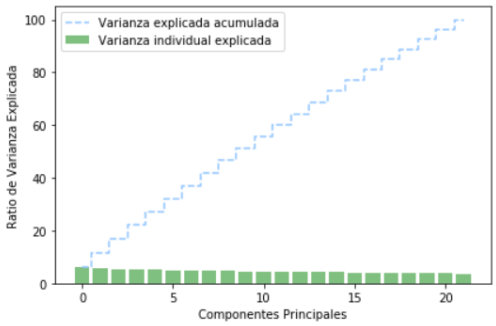
4.3 Transformación de datos - Estandarización

El siguiente paso que he llevado a cabo, ha sido la normalización de los datos, en concreto de las columnas que vamos a utilizar como variables predictivas. Toda la información de las cuotas de las casas de apuestas vamos a dejarla apartada hasta nuevo aviso.
Con la normalización buscamos ajustar nuestros valores que están medidos en diferentes escalas y los vamos a ajustar respecto a una escala común. Como he comentado en el apartado de atributos para los jugadores, he utilizado la funcionalidad Standard Scaler de la librería scikit-learn, que ajusta nuestros datos para que la media sea igual a 0 y su varianza igual a 1.
Fórmula de la estandarización, necesitamos calcular la media y la desviación de cada columna:

$$z = \frac{x - \mu}{\sigma} \qquad \mu = \frac{1}{N}\sum_{i=1}^{N}(xi) \qquad \sigma = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(xi - \mu)^2}$$

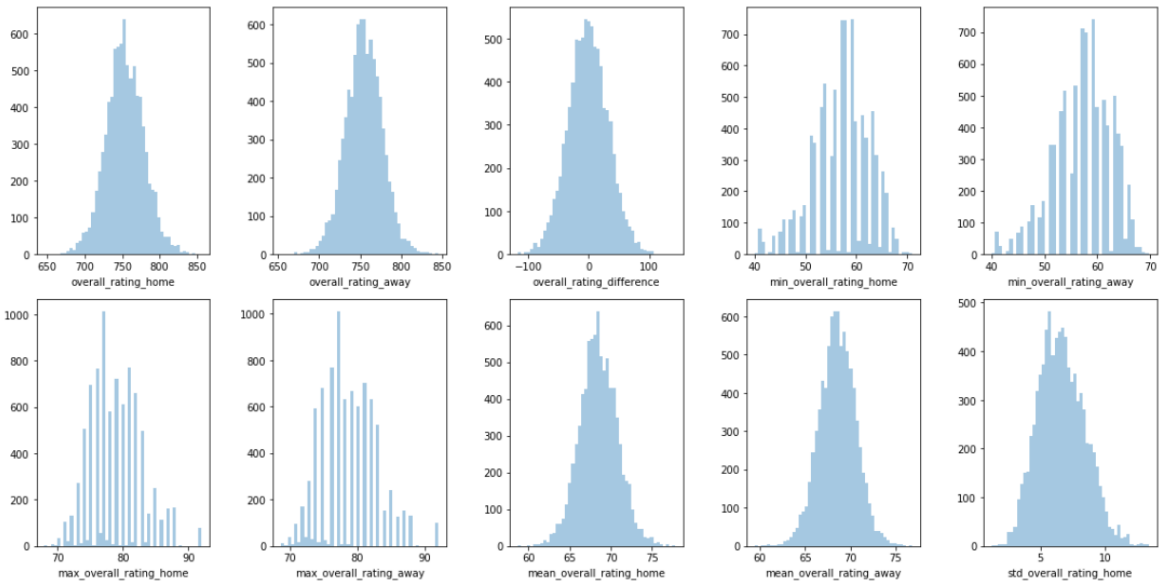
4.4 Reducción dimensionalidad

Para reducir la dimensionalidad, tenemos que elegir entre realizar una selección de un subconjunto de características o atributos originales, o la construcción de otros nuevos atributos que puedan resumir la información de los atributos originales (PCA, LDA).
En primer lugar, hemos intentado aplicar PCA solamente sobre los datos de los 22 jugadores, para reducir el número de predictores.
Hemos seguido el mismo proceso detallado en la creación de sus atributos (punto 3.2), pero los resultados como se puede ser en la figura inferior me han hecho cambiar de estrategia, ya que para llegar a un porcentaje de varianza explicada medianamente aceptable voy a necesitar muchos componentes, lo cual no reduce la dimensionalidad de los datos.



7.Figura 4.4.A Varianza explicada overall_rating 22 jugadores.

Después de descartar la creación de nuevas variables vía PCA, vamos a crear una serie de estadísticas que nos van a dar una visión resumida de las cualidades de los jugadores de cada equipo vista la inefectividad de PCA para estos atributos (este punto podría estar perfectamente dentro del apartado 3.2 Atributos de Jugadores).
Vamos a crear dos atributos que nos van a dar el valor medio del "overall_rating" de los equipos locales y visitantes; y otro atributo que va a representar la diferencia entre ambos.
También crearemos cuatro parejas (local y visitante) de atributos adicionales, el mínimo, el máximo, la media y la desviación típica de las calificaciones de cada equipo. Todas estas estadísticas nos dan una visión de la muestra de notas de cada equipo sin necesidad de incluir los 22 campos generados en nuestros algoritmos predictivos. Los normalizamos.

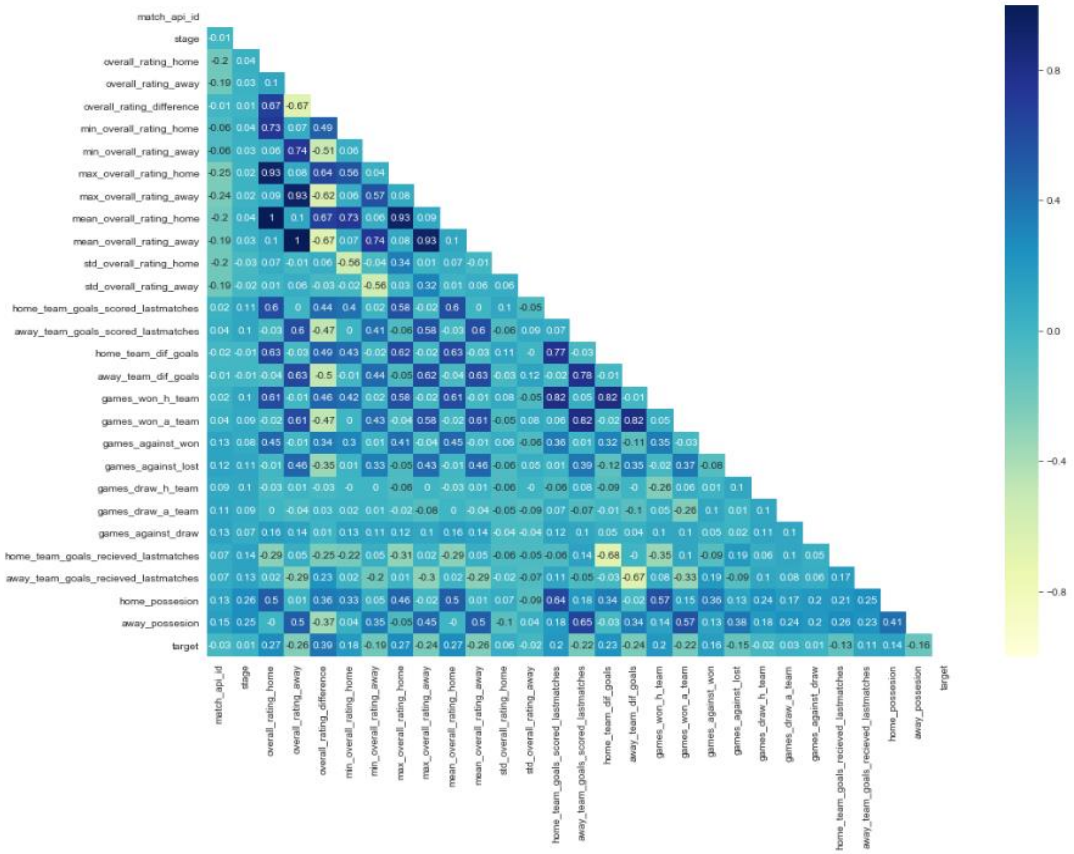


8.Figura 4.4.B Distribución nuevos atributos.

He llevado a cabo dos estrategias para dirigir en enfoque de mis predicciones, en la primera no voy a hacer reducción de la dimensionalidad, voy a tratar de averiguar cuál de las ligas es más predecible en función del rendimiento de los algoritmos. En la segunda estrategia, voy a aplicar PCA a los datos reduciendo su dimensionalidad y voy a comparar el rendimiento de los algoritmos en ambas estrategias.

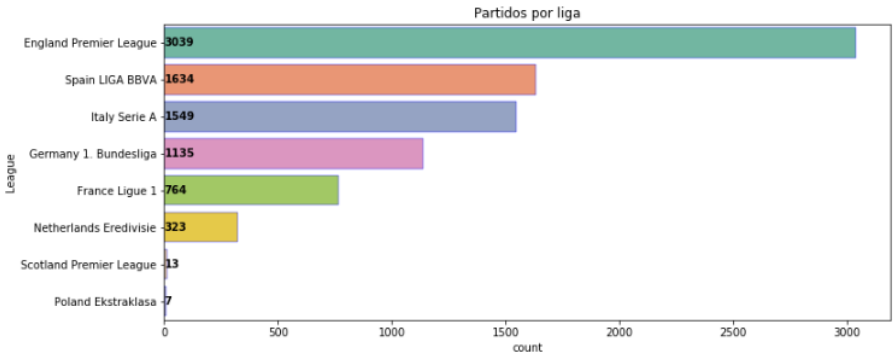
4.5 Visualización de datos limpios

En este punto vamos a ver algunas imágenes sobre cómo han quedado los datos una vez hemos completado la fase de preprocesamiento.
El primer gráfico mostrado a continuación, nos muestra la correlación entre las variables predictoras y la variable target. Apreciamos que las variables que provienen del rating de los jugadores están bastante correlacionadas entre si (parte superior), lo cual tiene todo el sentido. También apreciamos cierta correlación entre variables con atributos de los partidos y variables con atributos de los jugadores, lo cual es buena noticia y nos da esperanzas para conseguir lograr buenas predicciones.



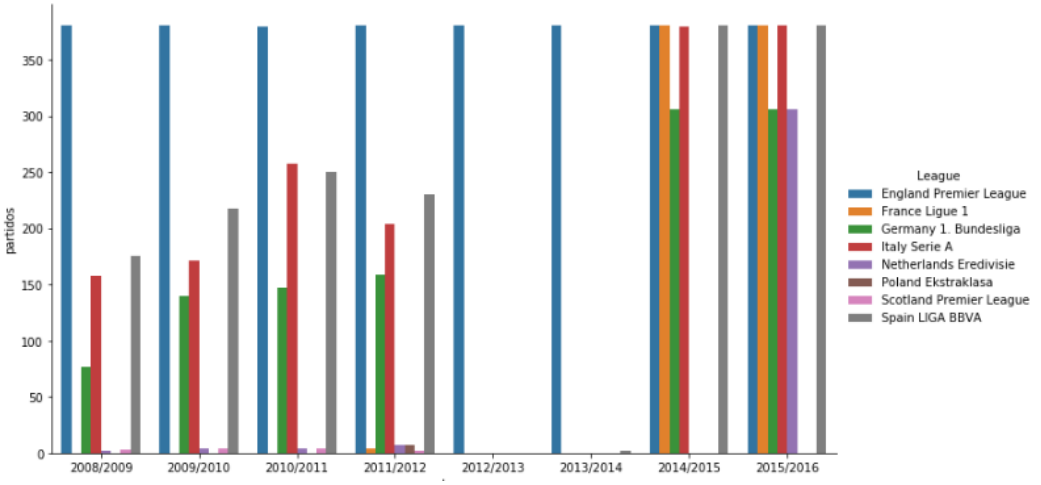
9.Figura 4.5.A Correlación entre variables predictivas.

En la siguiente figura, vamos a ver cuántos partidos por liga tenemos que cumplan todos los pasos que hemos ido dando. Automáticamente vamos a descartar las ligas escocesa y polaca por falta de información. Los cuatro candidatos por volumen de información son la Premier League, la Liga, la Serie A y la Bundesliga. Las ligas francesa y holandesa las trataremos también para ver cómo se comportan los algoritmos con menores volúmenes de información.

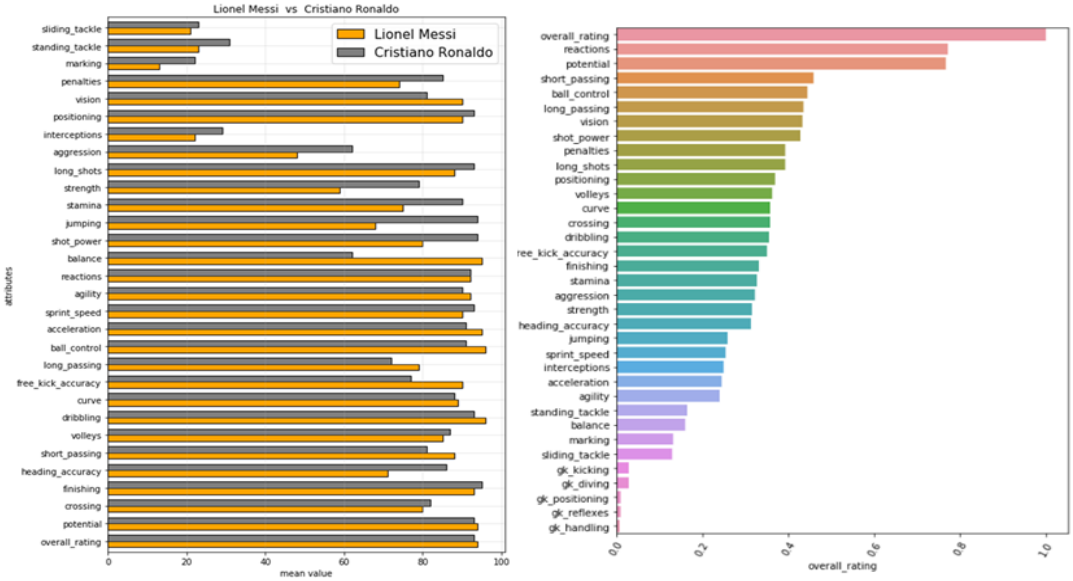


10.Figura 4.5.B Partidos por liga.

En la siguiente figura vamos a ver el mismo detalle, pero con el Split por temporadas. En el caso de la liga española, contamos con 20 equipos, lo cual hace 10 partidos por jornada, por 38 jornadas, una temporada completa son 380 partidos. Lo mismo pasa con la liga inglesa, la italiana y la francesa, una temporada completa son 380 partidos.
En cambio, en la Bundesliga y en la Eredivisie son 18 equipos, por lo tanto 9 partidos por 34 jornadas hacen un total de 306 partidos.



11.Figura 4.5.C Partidos por temporada y por liga.



12.Figura 4.5.D Comparativa Messi-C.Ronaldo y correlación Overall rating vs resto atributos.

En estas dos figuras mostradas vamos a ver, en la parte izquierda la comparativa de atributos de los dos mejores jugadores de la última temporada disponible, y en la parte derecha la correlación entre la variable overall_rating contra el resto de los atributos de cada jugador.

5 Desarrollo de modelos predictivos

En este apartado vamos a comentar los modelos predictivos que hemos utilizado. En nuestro caso y dada la casuística de nuestro problema el objetivo es obtener una etiqueta con el resultado del partido, son problemas de clasificación. El aprendizaje va a ser supervisado ya que vamos a definir de una forma clara cuál es nuestra variable objetivo o target.

5.1 Modelos de clasificación utilizados

Hemos probado cada uno de los modelos detallados a continuación en cada uno de los países, de esta forma seremos capaces de encontrar cual es el país más predecible, para entrar en detalle de sus algoritmos, optimizar sus parámetros y buscar un modelo que nos dé ciertas garantías.

5.1.1 Regresión Logística multinomial

A pesar de su nombre, no es un algoritmo para problemas de regresión sino de clasificación. Si tenemos más de dos clases, como es nuestro caso, debemos usar regresión logística multinomial y utilizaremos la entropía cruzada como función de error. Algunas de sus ventajas son que es simple, eficaz y fácil de computar. Sus resultados se interpretan con facilidad. Su funcionamiento es mejor cuando se usan atributos relacionados con la variable a predecir y se quitan los que no tienen relación.
Algunas de sus desventajas son que no puede resolver problemas que no son lineales, tiene dependencia de las características y no permite identificar cuáles son las más relevantes. Este punto nos hace pensar que no va a ser uno de los algoritmos con mejor rendimiento. Las clases de la variable objetivo deben ser separables linealmente.[10]

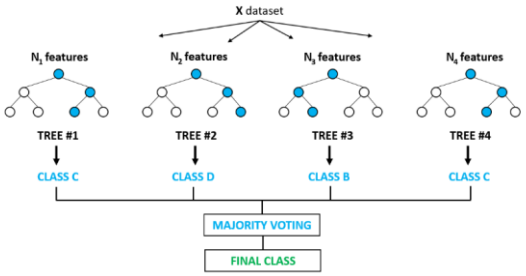
5.1.2 Random Forest

Está considerado como un algoritmo muy preciso y robusto, está compuesto por un gran número de árboles de decisión.
Algunas de sus ventajas son las siguientes; no sufre over-fitting ya que toma la media de todas las predicciones eliminando el sesgo, podemos utilizarlo en problemas de regresión y clasificación y nos permite sacar la importancia relativa de cada una de las características.

An example of overfitting, underfitting and a model that's "just right!"

13.Figura 5.1.2 Explicación visual over-fitting [12].

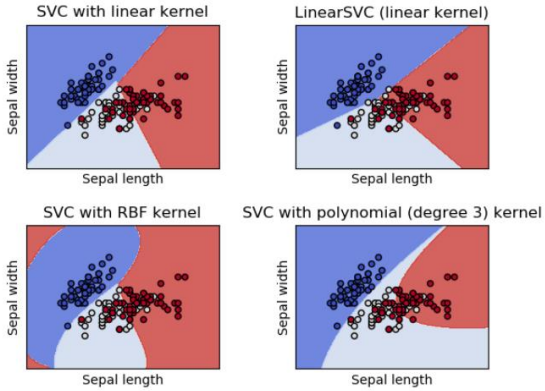
Como desventajas, es un algoritmo lento ya que consta de múltiples árboles de decisión, y es complicado de interpretar comparado con un árbol donde podemos entender la decisión siguiendo la trayectoria del árbol.



14.Figura 5.1.2 Representación de Random Forest [10].

5.1.3 Support Vector Machines

Los SVM o máquinas de vectores soporte son métodos de aprendizaje supervisado que sirven para problemas de clasificación, regresión o detección de outliers. Buscan el hiperplano dentro de las N dimensiones que permite clasificar los distintos puntos. Algunas ventajas que tienen son su efectividad en espacios dimensionales amplios, eficiencia en el uso de memoria, versatilidad ya que podemos especificar diferentes funciones del kernel. Algunas desventajas son en el caso de tener un número de características superior al número de muestras utilizado, es complicado evitar el over-fitting o sobreajuste. En problemas de clasificación multi clase transforma los resultados de una función de decisión uno contra uno a uno contra resto.



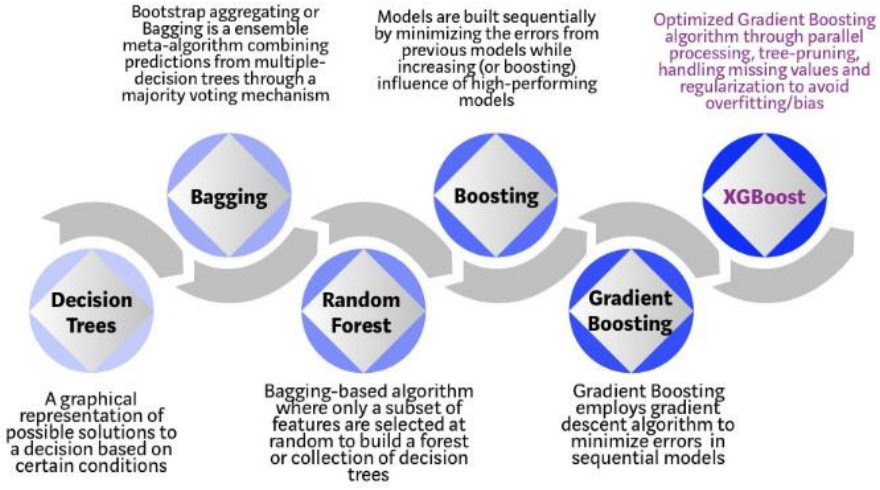
15.Figura 5.1.3 Representación de SVM [13].

5.1.4 Adaptative Boosting Clasificator

El boosting es un enfoque basado en la idea de crear reglas de predicción precisas combinando muchas reglas relativamente imprecisas o débiles.
ADA es la abreviatura de adaptative boosting. Funciona eligiendo un algoritmo base que suelen ser los árboles de decisión y trata de mejorarlo a base de iteraciones. Asigna pesos iguales a todos los datos de entrenamiento y va aumentando los pesos sobre los datos en los que la clasificación es incorrecta. Realiza *n* iteraciones y su resultado es la suma ponderada de los *n* algoritmos base. Uno de los inconvenientes que tienen las técnicas de Boosting como este algoritmo es el sobreajuste.

5.1.5 Extreme Gradient Boosting Clasificator

Es un algoritmo que ha tenido muy buenos resultados en competiciones de Machine Learning con datos estructurados. Se trata de una implementación de árboles de decisión con gradient boosting diseñado para maximizar el rendimiento y velocidad de ejecución. Toma solamente valores numéricos como entrada.
El Gradient Boosting implica tres elementos; una función de pérdida a optimizar, en el caso de problemas de clasificación hablaremos de entropía cruzada, un algoritmo de aprendizaje débil (Árboles de decisión) para hacer las predicciones, por último, un modelo aditivo para añadir los algoritmos de aprendizaje débiles y poder minimizar la función de pérdida [14].



16.Figura 5.1.5 Evolución de los algoritmos desde los árboles de decisión [15].

5.1.6 K- Nearest Neighbors

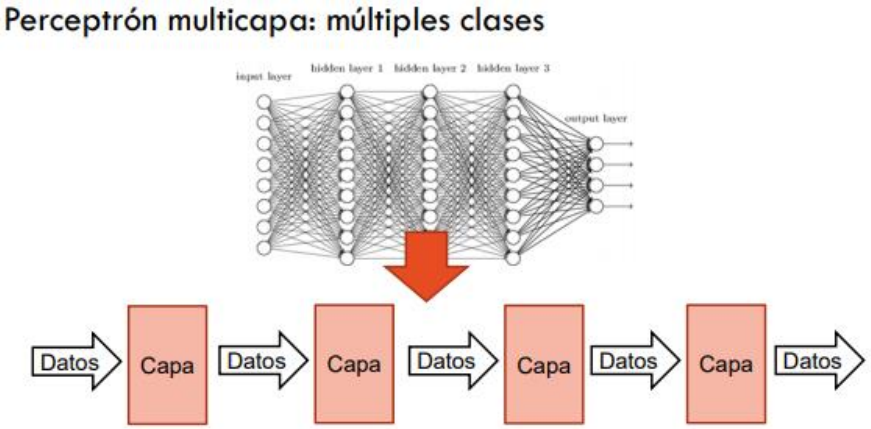
También llamado KNN es uno de los algoritmos más populares para el reconocimiento de patrones y correlaciones. Es supervisado y vale para problemas de clasificación o regresión. Clasifica los valores buscando los puntos de datos más similares que ha aprendido en la fase de entrenamiento. Es un algoritmo basado en instancia, por lo que no aprende un modelo, memoriza las instancias de entrenamiento y las utiliza como base de conocimiento en la predicción. Como ventajas, es sencillo de implementar y aprender, como desventajas, utiliza todo el dataset para entrenar cada punto por lo que requiere mucha capacidad de computación. Es recomendable en datasets pequeños, como el nuestro.

5.1.7 Gaussian Naive Bayes

Es un algoritmo muy utilizado en clasificación de datos. Los clasificadores de la familia Naive Bayes o ingenuo Bayes están basados en el teorema de Bayes. Hacen una suposición de independencia entre los predictores. Como ventajas es fácil de implementar y útil en grandes conjuntos de datos. Funciona bien en problemas multiclase, por ello lo hemos elegido como uno de los candidatos. Como inconveniente la asunción de que las variables son independientes. En nuestro caso y en la mayoría de las situaciones reales las variables no son independientes, por lo que tengo serias dudas de que este algoritmo nos vaya a dar buenos resultados.

5.1.8 Redes Neuronales

Como no podía ser de otra forma, no podíamos dejar de lado a las redes neuronales como otro de los candidatos que vamos a tener en cuenta. Pueden servirnos para resolver problemas de regresión y de clasificación, supervisados y no supervisados. En este caso en concreto de clasificación supervisada, utilizaremos perceptrón multicapa, se trata de una generalización del perceptrón simple, surge a raíz de las limitaciones del perceptrón simple a la hora de clasificar conjuntos linealmente separables. Es fácil de utilizar y de aplicar, pero también tiene algunas limitaciones en su proceso de aprendizaje a la hora de aprender problemas complejos con muchas variables. En cuanto a las redes neuronales convolucionales, están más enfocadas a la clasificación de imágenes trabajando con grandes matrices de datos.



17.Figura 5.1.8 Perceptrón multicapa con múltiples clases [16].

5.2 Consideraciones previas

Para una correcta predicción de los datos he tenido que hacer algunas consideraciones previas fundamentales para garantizar el funcionamiento adecuado de los algoritmos. Estas consideraciones son discutibles y pueden modificarse en un futuro.
Una vez realizado todo el proceso de predicción que voy a detallar a continuación, he decidido descartar los datos previos a la temporada 2011-2012 para tratar de mejorar los resultados. Revisando la figura 4.5.C y conociendo cómo evoluciona este deporte creo que tiene sentido considerar solamente las últimas temporadas de información.
Para realizar la división de los datos en conjunto de entrenamiento y conjunto de prueba, he considerado que vamos a tratar de predecir la segunda mitad de la última liga excluyendo los 4 últimos partidos en los que hay muchos equipos que ya no se juegan nada y los resultados pueden ser desconcertantes. Por la naturaleza de los datos, vamos a considerarlos prácticamente como una serie temporal, teniendo importancia el orden, no podemos dividirlos aleatoriamente ya que no queremos predecir algo que ya ha sucedido, queremos predecir los resultados de acontecimientos futuros.

5.3 Liga más predecible

Vamos a comparar el rendimiento de los algoritmos de clasificación utilizados para cada una de las ligas analizadas, una vez encontremos el país elegido profundizaremos sobre sus estimaciones tratando de mejorarlas.

5.3.1 Premier League

Vamos a hacer una primera iteración sin reducir la dimensionalidad de los datos, seleccionando los partidos correspondientes a la Premier League. Adjunto tabla donde podemos ver que algoritmos he utilizado, su precisión y la diagonal principal de su matriz de confusión. Los resultados son muy flojos, el azar nos daría un 33,3% de probabilidad de acertar y los modelos en este país apenas alcanzan un 48%. Es cierto que la liga inglesa es conocida por su competitividad y porque el resultado de los partidos es impredecible, pero los resultados no convencen. En el mejor de los casos hemos acertado 77 resultados de partidos sobre una muestra a predecir de 160.

Algoritmo_Utilizado	Accuracy	Aciertos_H.Lose	Aciertos_H.Draw	Aciertos_H.Victory	Hits
Neuronal Net	0.48125	22	0	55	77
Logistic Regression	0.46250	20	1	53	74
Random Forrest	0.45625	17	2	54	73
Support Vector Classification	0.45000	18	0	54	72
XGB Classifier	0.44375	15	1	55	71
AdaBoost Classifier	0.44375	16	0	55	71
Gaussian Model	0.41875	22	5	40	67
K Neighbors Model	0.40625	18	8	39	65

4.Tabla 5.3.1 Comparativa algoritmos Premier League.

Si decidimos apostar 100€ a todos los pronósticos con las cuotas de la casa de apuestas Bet365 obtendríamos un resultado de -1.981€, con un Yield o rentabilidad del -12,4%.

5.3.2 Serie A

En la liga italiana hemos mejorado nuestra precisión hasta alcanzar más de un 55% utilizando árboles de decisión. Es curioso como los algoritmos que mejor clasifican los empates son al final los que peor rendimiento tienen. También vemos claramente como clasifican mejor las victorias locales que las victorias a domicilio.
En el caso de Italia si decidimos apostar 100€ a todos los pronósticos en la casa de apuestas Bet&Win obtendríamos un resultado de 1.152€, con un Yield o rentabilidad del 7,2%.

Algoritmo_Utilizado	Accuracy	Aciertos_H.Lose	Aciertos_H.Draw	Aciertos_H.Victory	Hits
Random Forrest	0.55625	21	1	67	89
AdaBoost Classifier	0.55000	22	0	66	88
Logistic Regression	0.52500	22	2	60	84
Neuronal Net	0.51875	22	1	61	84
Support Vector Classification	0.51875	23	0	60	83
Gaussian Model	0.51875	24	5	54	83
XGB Classifier	0.50000	16	4	60	80
K Neighbors Model	0.46875	21	9	45	75

5.Tabla 5.3.2 Comparativa algoritmos Serie A.

5.3.3 Bundesliga

En la liga alemana vemos como empeoran los resultados, los SVM y la Regresión Logística son los algoritmos con mejor rendimiento, pero seguimos obteniendo 0 aciertos en los empates de los partidos. KNN hasta el momento es el peor algoritmo en los tres países

analizados, la no independencia de las variables predictoras parece estar haciendo que este algoritmo obtenga una tasa de acierto tan baja.
Si decidimos apostar a todos nuestros pronósticos con las cuotas de la casa de apuestas Interwetten, obtendríamos unas pérdidas de 14€ después de haber apostado 14.400€; un negocio redondo.

Algoritmo_Utilizado	Accuracy	Aciertos_H.Lose	Aciertos_H.Draw	Aciertos_H.Victory	Hits
Logistic Regression	0.527778	21	0	55	76
Support Vector Classification	0.527778	20	0	56	76
Random Forrest	0.500000	17	0	55	72
XGB Classifier	0.486111	17	1	52	70
AdaBoost Classifier	0.479167	14	0	55	69
Neuronal Net	0.472222	18	0	50	68
Gaussian Model	0.458333	19	13	34	66
K Neighbors Model	0.402778	23	4	31	58

6.Tabla 5.3.3 Comparativa algoritmos Bundesliga.

5.3.4 Ligue 1

En la liga francesa el algoritmo que mejor ha clasificado ha sido Random Forest y con bastantes aciertos en los empates (teniendo en cuenta que el resto no acierta ningún empate). Hasta el momento es la precisión más alta obtenida, con 84 aciertos sobre 150 partidos. Si decidimos apostar a todos nuestros pronósticos con el mejor algoritmo para este país con las cuotas de la casa de apuestas Interwetten, obtendríamos un beneficio de 447€, con un Yield del 2,98%.

Algoritmo_Utilizado	Accuracy	Aciertos_H.Lose	Aciertos_H.Draw	Aciertos_H.Victory	Hits
Random Forrest	0.560000	18	6	60	84
Neuronal Net	0.533333	24	0	55	79
Support Vector Classification	0.526667	21	0	58	79
Logistic Regression	0.520000	20	6	52	78
Gaussian Model	0.513333	27	7	43	77
XGB Classifier	0.493333	17	9	48	74
AdaBoost Classifier	0.493333	12	1	61	74
K Neighbors Model	0.380000	22	7	28	57

7.Tabla 5.3.4 Comparativa algoritmos Ligue1.

5.3.5 Eredivisie

En cuanto a la liga holandesa, es bastante curioso como el mejor modelo ha sido K-Nearest Neighbors, cuando en el resto de las ligas ha sido el peor. Sin lugar a duda y conociendo la naturaleza de este algoritmo, al tener solamente disponible información de la temporada que estamos intentando predecir (consultar Figura 4.5.C), el conjunto de entrenamiento es mucho más pequeño que en los países analizados hasta el momento y mientras que esto penaliza al resto de algoritmos, vemos como hace que KNN pueda clasificar mejor y acertar los resultados de la segunda mitad de la liga comparando con lo sucedido en la primera mitad de la liga.
Si confiáramos en todas las predicciones de este modelo KNN y lo aplicamos a las cuotas de la casa de apuestas Interwetten, obtendríamos un beneficio de 1.426€ con un Yield del 10,5%.

Algoritmo_Utilizado	Accuracy	Aciertos_H.Lose	Aciertos_H.Draw	Aciertos_H.Victory	Hits
K Neighbors Model	0.540741	26	9	38	73
Random Forrest	0.496296	18	1	48	67
XGB Classifier	0.496296	21	4	42	67
Gaussian Model	0.496296	22	7	38	67
AdaBoost Classifier	0.466667	13	6	44	63
Logistic Regression	0.429630	17	4	37	58
Support Vector Classification	0.429630	18	3	37	58
Neuronal Net	0.407407	2	0	53	55

8.Tabla 5.3.5 Comparativa algoritmos Eredivisie.

5.3.6 La Liga

En el caso de la liga española, obtenemos los mejores resultados hasta el momento: 57,6% de acierto. De nuevo los mejores algoritmos tienen muchísimos problemas para clasificar correctamente los empates, después de realizar múltiples pruebas y de haber creado atributos específicos para identificar equipos con tendencia a empatar no conseguimos obtener buenos resultados en la predicción de empates.

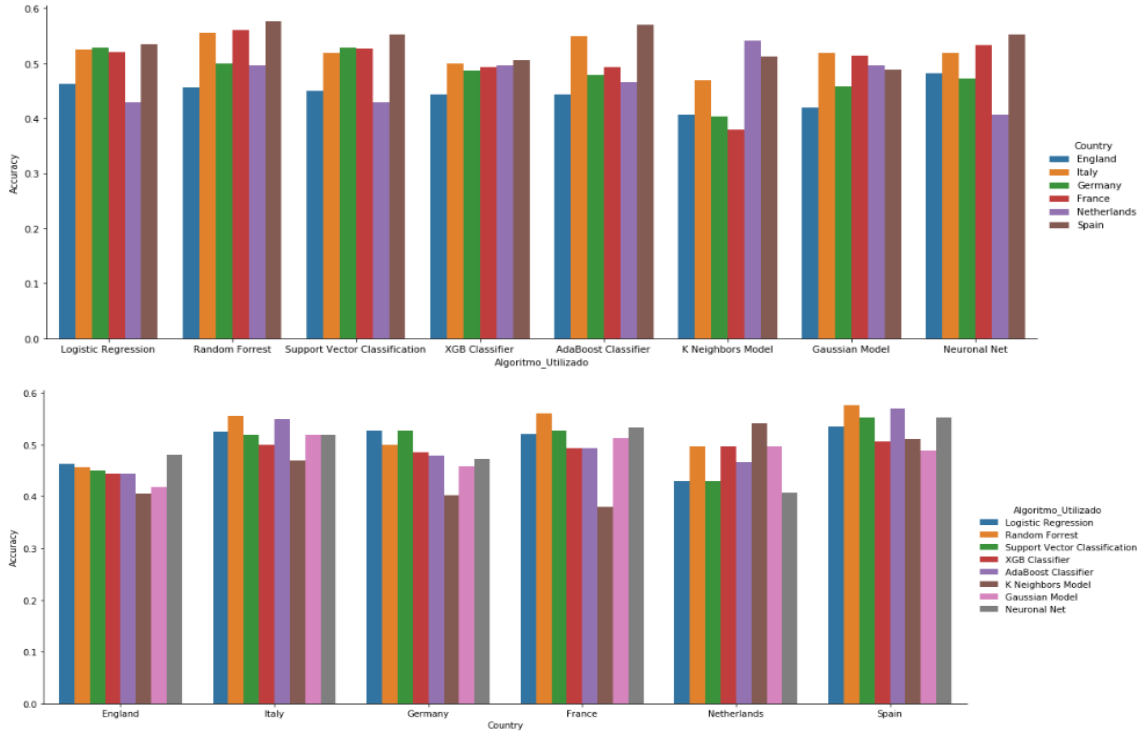
Algoritmo_Utilizado	Accuracy	Aciertos_H.Lose	Aciertos_H.Draw	Aciertos_H.Victory	Hits
Random Forrest	0.576471	21	0	77	98
AdaBoost Classifier	0.570588	21	0	76	97
Neuronal Net	0.552941	21	0	73	94
Support Vector Classification	0.552941	21	0	73	94
Logistic Regression	0.535294	22	2	67	91
K Neighbors Model	0.511765	25	10	52	87
XGB Classifier	0.505882	21	4	61	86
Gaussian Model	0.488235	20	17	46	83

9.Tabla 5.3.6 Comparativa algoritmos La Liga.

Si confiamos en todas las predicciones apostando 100€ a cada partido con las cuotas de Bet365 obtendríamos un beneficio de 24€ con un Yield del 0,14%.

5.3.7 Comparativa de ligas y algoritmos

Una vez hemos analizado todas las ligas vamos a visualizar la comparativa de la precisión de los diferentes modelos por país, para tratar de fundamentar nuestra liga más predecible con datos reales.
Este gráfico nos ha ayudado a tomar una decisión sobre cuál es el país que vamos a hacer un mayor foco y vamos a profundizar nuestros análisis. Pese a que económicamente no era el más rentable, vamos a centrarnos en el criterio de tasa de acierto o precisión. En un problema de clasificación multi etiqueta como el nuestro, mide sobre el conjunto de etiquetas que hemos predicho cuantas coinciden exactamente con los valores reales que tienen esas etiquetas.

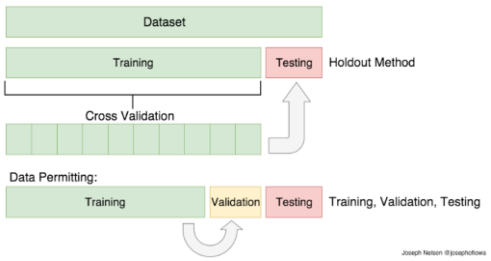


18.Figura 5.3.7 Comparativa de ligas por tipo de algoritmo.

Podemos ver como por encima de todos, España ha destacado consiguiendo mejores resultados en algunos de sus modelos, y ha obtenido buenos resultados comparado con el resto de las ligas.
En cuanto a los algoritmos, Random Forest es el que mejor se ha comportado si hacemos una media de los resultados obtenidos por cada algoritmo en el conjunto de los campeonatos analizados.

5.4 Optimización de parámetros

En este apartado, una vez elegida la liga en la que obtengo mayor número de aciertos con las predicciones, voy a tratar de buscar cuales son los parámetros específicos de cada modelo que mejor funcionan con el conjunto de datos seleccionado. En los primeros pasos hemos utilizado los atributos por defecto, sin embargo, es recomendable utilizar para cada combinación *algoritmo - conjunto de datos*, la combinación de atributos que mejor se ajusten a esos datos y que nos permita optimizar el rendimiento en la clasificación.
Para ello, disponemos en la librería scickit-learn, de dos herramientas que facilitan la búsqueda de estos parámetros óptimos. Cross Validation y Grid Search. La clave del funcionamiento de un modelo es saber cómo clasificará datos nuevos que no han sido analizados previamente.
Cross Validation es una técnica que se utiliza para poder evaluar los resultados de un análisis garantizando que son independientes de cómo se han particionado los datos de entrenamiento y test. Funciona dividiendo el conjunto de entrenamiento en *k* subconjuntos y entrenando *k-1* conjuntos para comprobar con el último conjunto no entrenado.

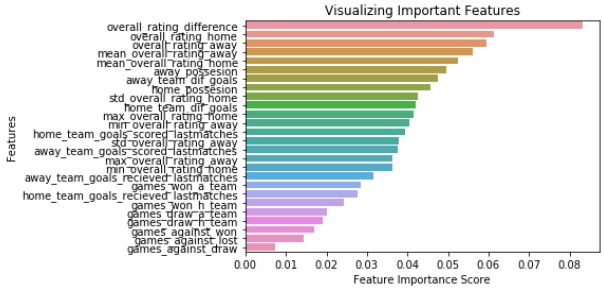


19.Figura 5.4.A Representación visual Cross Validation.

En concreto vamos a utilizar la técnica de validación cruzada "Shuffle Split", que funciona tomando muestras aleatorias a lo largo de cada iteración generando un conjunto de entrenamiento y otro de validación. El mismo valor puede elegirse varias veces en cada iteración por lo que solamente vamos a marcar una iteración evitando que estos se repitan.
Grid Search por su parte consiste en realizar una búsqueda exhaustiva de hiper – parámetros, que son los parámetros que no se aprenden dentro del algoritmo. Estos parámetros ayudan al algoritmo a realizar los cálculos necesarios en su fase de aprendizaje. Esta técnica busca exhaustivamente cual es el mejor valor para un parámetro realizando búsqueda y comparación en rejilla. Por defecto funciona con la métrica "accuracy_score" que es la que venimos utilizando.
Random Forest Grid Search: En este algoritmo hemos tratado de optimizar los siguientes hiper-parámetros:

- n_estimators, que son el número de árboles que componen nuestro bosque.
- max_depth, corresponde a la máxima profundidad que tendrá cada árbol.
- max_features, número de características a considerar cunado buscamos la mejor división.
- min_samples_leaf, mínimo número de muestra necesarias para cada hoja nodo.
- min_samples_split, mínimo número de muestras necesarias para hacer una división en un nodo interno.
- bootstrap, si se usan muestras de Bootstrap en la construcción de árboles. Si es falso usamos todo el dataset para construir cada árbol.

Hemos obtenido una precisión del 57% ejecutando la búsqueda de hiper-parámetros, si los aplicamos sobre el modelo Random Forest obtenemos una precisión de 56,47%.
También he tratado de "podar" este bosque aleatorio, quitando las cuatro variables predictoras menos importantes. En el siguiente gráfico podemos ver cuáles son las variables más importantes y las menos relevantes para este modelo.



20.Figura 5.4.B Random Forest Feature Importance Score.

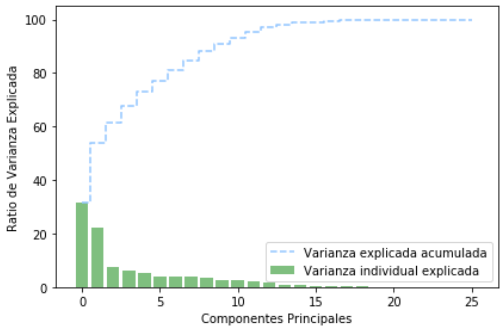
La precisión ha empeorado bajando hasta un 55,88%; por lo que hemos descartado esta aproximación.
Support Vector Machine Grid Search: Para este algoritmo he tratado de optimizar tres hiper-parámetros, el parámetro 'C' y el tipo de kernel son los más relevantes.

- Parámetro ‘C’, parámetro de regularización, la fuerza de la regularización es inversamente proporcional a su valor. Siempre debe ser positivo.
- Kernel, tipo de kernel utilizado en el algoritmo, hay varios tipos; ‘rbf’, ‘sigmoidal’, ‘lineal’…
- Parámetro ‘gamma’, coeficiente del kernel.

Después de ejecutar este modelo, hemos obtenido una precisión del 58,82% la más alta hasta el momento. Aplicamos estos hiper-parámetros sobre el clasificador con todo el conjunto de datos y obtenemos la misma precisión.
Aplicaremos esta técnica sobre más modelos, pero primero vamos a tratar de disminuir el número de variables predictoras.

5.5 Modelos con reducción de dimensionalidad

La dimensión de los datos en un factor muy importante que debemos considerar para almacenar y computar nuestros datos. Hay diversas técnicas que nos permiten extraer y proyectar hacia nuevos conjuntos de datos con un menor número de atributos, pero con métricas muy parecidas al conjunto original.
En nuestro caso vamos a utilizar (PCA) Principal Component Analysis. Como ya he detallado su funcionamiento en el punto 4.4 voy a ir al grano. Vamos a visualizar la varianza explicada de los 26 atributos que estamos utilizando en nuestros modelos.



21.Figura 5.5.A Varianza explicada Atributos.

Hay casos en los que reduciendo la dimensionalidad se pierde calidad de los datos, aunque el entrenamiento pueda ser más rápido. Después de realizar varias pruebas creando el nuevo conjunto de datos con entre 3 y 7 componentes, he optado por utilizar 4 componentes que es el que mejores resultados me ha dado cubriendo entre un 65-70% de la varianza explicada de los atributos. Voy a comentar los modelos probados.

Random Forest + PCA + Grid Search: Una vez encontramos los mejores hiper-parámetros para las nuevas 4 componentes obtenidas tras aplicar PCA sobre nuestro conjunto de datos inicial, el resultado del modelo tiene un 54,41% de precisión. Queda por debajo de las precisiones obtenidas antes de aplicar la reducción de dimensionalidad.

Extreme Gradient Boosting + PCA + Grid Search: Ahora probamos con este modelo tan famoso por sus excelentes resultados, paso a detallar los parámetros que he tratado de optimizar:

- Learning_rate, tasa de aprendizaje, afecta a como se actualizan los pesos en el descenso de gradiente.
- min_child_weight, define la suma mínima de los pesos de las observaciones en cada árbol hijo.

- Parámetro ‘gamma’, mínimo reducción necesaria de la función de pérdidas necesaria para hacer una nueva partición en un nodo de la hoja del árbol.
- subsample, submuestra de las instancias de entrenamiento, si su valor en 0.5 el modelo selecciona aleatoriamente la mitad de los datos de entrenamiento antes de hacer crecer a los árboles.
- colsample_bytree, familia de parámetros para coger submuestras de las columnas.
- max_depth, máxima profundidad de cada árbol.

El resultado de este modelo con esta serie de parámetros optimizados ha sido de un 57%, cercano a nuestro mejor modelo (SVM optimizado), pero no llega a superarlo.

Adaptative Boosting + PCA + Grid Search: También he probado este modelo tratando de optimizar estos dos parámetros. El resultado es un 55,88%; insuficiente.

- Learning_rate, tasa de aprendizaje, afecta a como se actualizan los pesos en el descenso de gradiente.
- n_estimators, que son el número de árboles que componen el bosque.

K-Nearest Neighbors + PCA + Grid Search: Vamos a probar este modelo a pesar de ser el peor en rendimiento en muchas de las ligas analizadas. Los resultados son muy parecidos al caso anterior, el rendimiento de este algoritmo ha mejorado con la reducción de la dimensionalidad dado que independiza sus variables. Optimizamos los siguientes parámetros:

- n_neighbors, número de vecinos utilizado.
- leaf_size, tamaño de la hoja, puede afectar a la velocidad de la construcción y a la memoria requerida para almacenar el árbol.
- Algorithm, algoritmo utilizado para computar los vecinos cercanos (BallTree, KDTree).

Support Vector Machine + PCA + Grid Search: Para este algoritmo he tratado de optimizar los tres mismos hiper-parámetros que antes de aplicar PCA. La precisión de este es del 56,47%.

La conclusión en este apartado es que estamos perdiendo calidad de los datos, ya que los resultados no mejoran con la reducción de la dimensionalidad.

5.6 Balancear clases

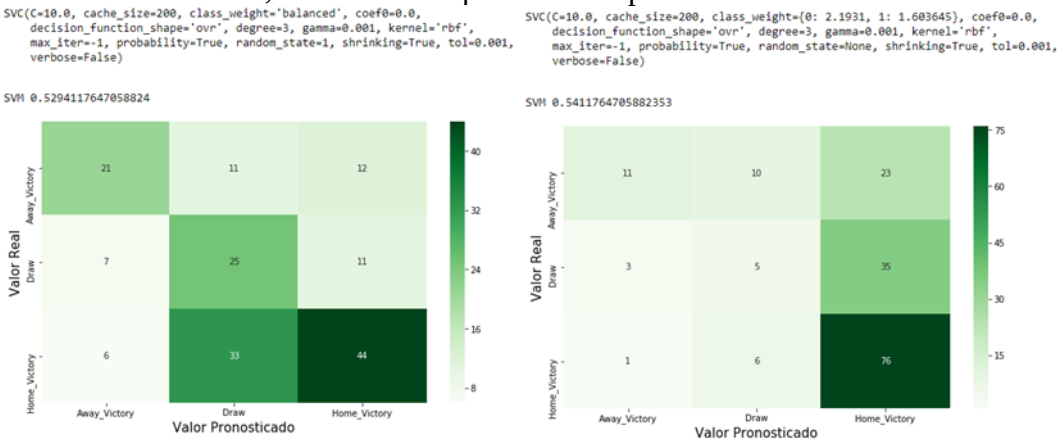
Una vez hemos terminado de realizar todas las pruebas, toca elegir el modelo más preciso posible. Es muy curioso ver que ningún modelo consigue acertar en los empates. Revisando las clases, se aprecia que las mismas no están balanceadas; 704 Victorias Locales, 321 Empates y 439 Victorias Visitantes. El desbalanceo tampoco es excesivamente grave, pero he probado a ajustar los mejores modelos a ver si mejoran los resultados balanceando la clase ‘0’, que es la que corresponde a los empates. Los modelos con los que hemos realizado pruebas son Random Forest, Extreme Gradient Boosting y SVM.

Algoritmo_Utilizado	Accuracy	Aciertos_H.Lose	Aciertos_H.Draw	Aciertos_H.Victory
SVC Grid Search	0.588235	20	0	80
Support Vector Machine Parametros Tuneados	0.588235	20	0	80
XGboost Grid Search + PCA	0.570588	19	0	78
AdaBoost Classifier	0.570588	21	0	76
Random Forest Grid Search	0.570588	22	0	75
Random Forrest	0.570588	22	0	75
Neuronal Net	0.564706	24	0	72
Support Vector Classification Tuneado + PCA	0.564706	19	0	77

10.Tabla 5.6.A Segunda comparativa algoritmos La Liga.

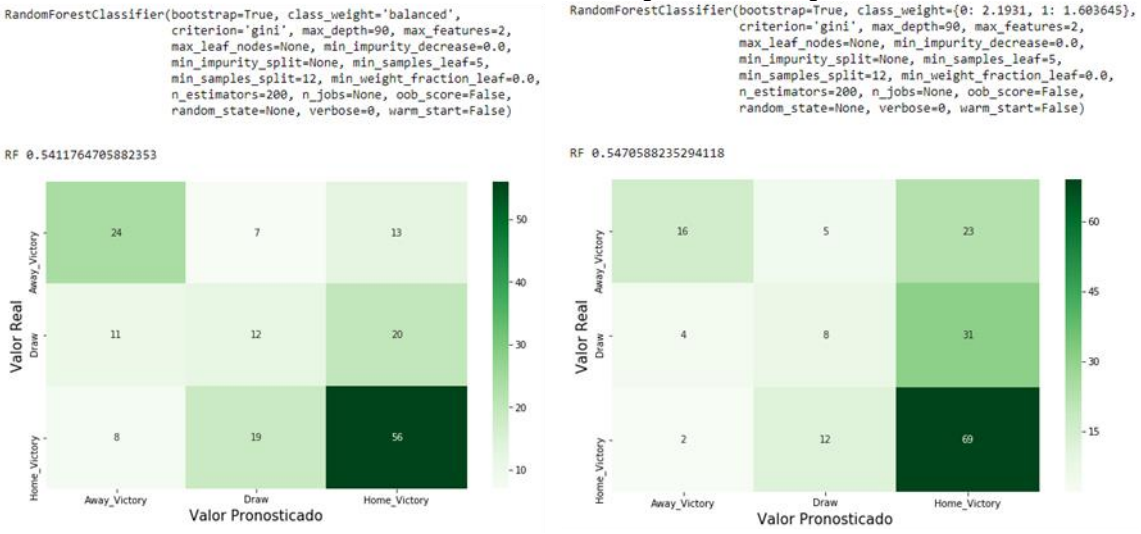
Para ello modificamos el parámetro class_weight dentro de los modelos, cargando un diccionario con cada clase y el peso que queremos asignar o marcando el valor del parámetro = ‘balanced’ donde automáticamente balancea la clase minoritaria.

- Resultados con **SVM**, utilizando los parámetros óptimos.



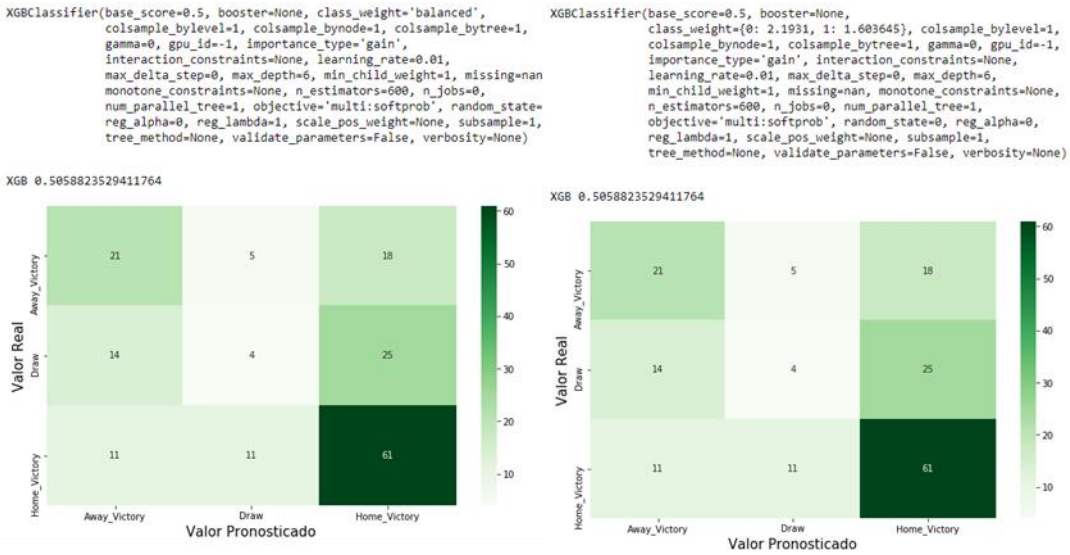
22.Figura 5.6.B Matriz de confusión SVM Balanceados.

- Resultados con **Random Forest**, utilizando los parámetros óptimos.



23.Figura 5.6.C Matriz de confusión Random Forest Balanceados.

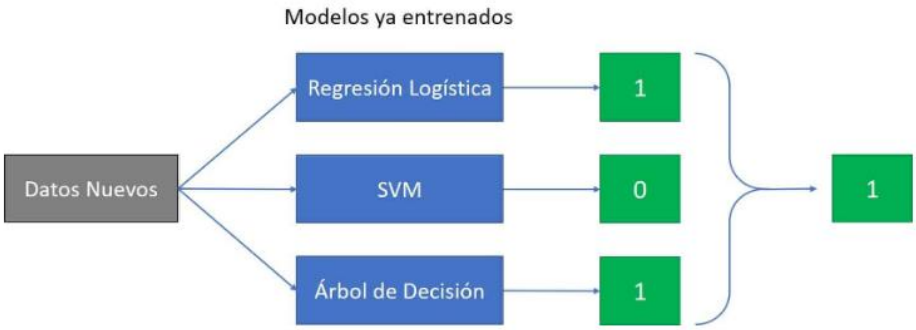
- Resultados con **Extreme Gradient Boosting,** utilizando los parámetros óptimos.



24.Figura 5.6.D Matriz de confusión Extreme Gradient Boosting.

5.7 Clasificador de Votos (Ensemble voting classifier)

Por último, antes de tomar una decisión definitiva, vamos a probar con un clasificador de votos. Consiste en un conjunto de modelos de machine learning en el que cada uno hace una predicción diferente, las predicciones de estos modelos se combinan para finalmente obtener una única predicción y eligen una clase de salida. Los errores de los modelos tienden a compensarse.
En este ejercicio voy a aplicar votación por mayoría.



25.Figura 5.7.A Modelo Votación por mayoría [19].

He probado cargando tres modelos, SVM, Random Forest y KNN. Los dos primeros utilizando los parámetros óptimos. A continuación, muestro los resultados, se han realizado pruebas con dos tipos de votaciones; 'Hard Voting' y 'Soft Voting'. Para medir el rendimiento también utilizo la precisión.
Hard Voting: La clase de salida pronosticada es la clase con la mayoría de los votos. Precisión 57,64%
Soft Voting: La clase de salida es la predicción basada en el promedio de la probabilidad dada a esa clase. El peso que he asignado a las clases en este caso es de 1 a la victoria visitante, 2 al empate y 0.5 a la victoria local. Precisión 58,23%
Comparado con todos los modelos que hemos probado los resultados son muy buenos, pero no los mejores.

5.8 Modelo elegido – Matriz de confusión

Una vez hemos terminado de realizar todas las pruebas, toca elegir el modelo más preciso posible. El mejor modelo ha sido el SVM con los mejores parámetros que hemos sacado buscando en malla mediante Grid Search. Ha alcanzado una precisión de 58,82%.

Algoritmo_Utilizado	Accuracy	Aciertos_H.Lose	Aciertos_H.Draw	Aciertos_H.Victory
SVC Grid Search	0.588235	20	0	80
Support Vector Machine Parametros Tuneados	0.588235	20	0	80
XGboost Grid Search + PCA	0.570588	19	0	78
AdaBoost Classifier	0.570588	21	0	76
Random Forest Grid Search	0.570588	22	0	75
Random Forrest	0.570588	22	0	75
Neuronal Net	0.564706	24	0	72
Support Vector Classification Tuneado + PCA	0.564706	19	0	77

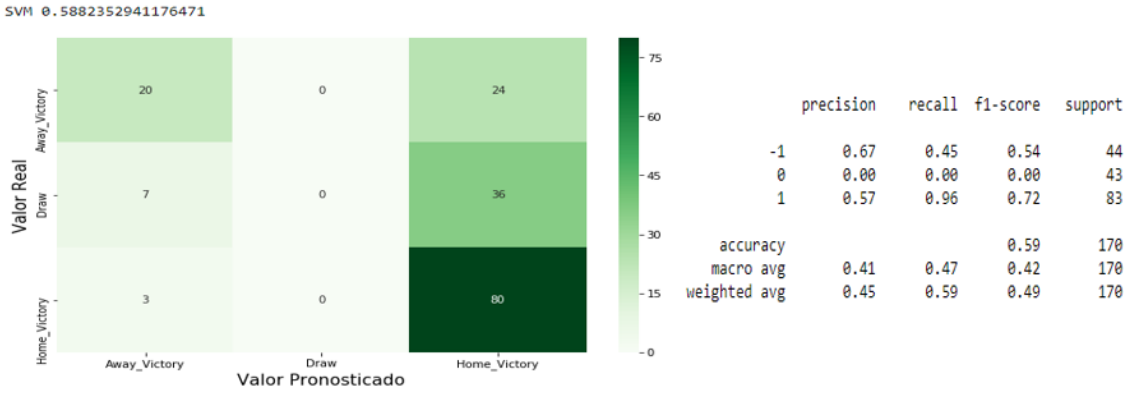
11.Tabla 5.8.A Tabla final comparativa algoritmos La Liga.

Claramente ha conseguido clasificar muy bien las victorias locales, pero muy mal las otras dos clases. Estas son las métricas que se muestran a continuación

$$\text{Precisión} = \frac{VP}{VP+FP} \qquad \text{Recall} = \frac{VP}{VP+FN} \qquad \text{Especificity} = \frac{VN}{VN+FP}$$

$$\text{Exactitud}(Accuracy) = \frac{VP+VN}{VP+FP+VN+FN} \qquad \text{f1}-score = \frac{2*Precisión*Recall}{Precisión + Recall}$$

La precisión en el caso de los empates es de 0, no ha conseguido clasificar ningún empate correctamente. En el caso de las victorias locales, la precisión es del 57% ya que ha cometido muchos errores clasificando empates y victorias visitantes como victorias locales.
En el caso de la sensibilidad o Recall, con las victorias locales solo ha fallado en tres partidos, por lo que tiene un 96%. En el caso de los empates 0 aciertos, y las victorias visitantes ha fallado en 24 partidos y ha acertado en 20 por lo que la métrica no es muy buena.
El f1-score combina precisión y sensibilidad, es muy útil en casos como el nuestro en el que la distribución de las clases es desigual, podemos ver como lo que más nos penaliza son los empates.



	precision	recall	f1-score	support
-1	0.67	0.45	0.54	44
0	0.00	0.00	0.00	43
1	0.57	0.96	0.72	83
accuracy			0.59	170
macro avg	0.41	0.47	0.42	170
weighted avg	0.45	0.59	0.49	170

26.Figura 5.8.B Matriz de confusión y métricas del mejor modelo.

5.9 Aplicación de las predicciones en las apuestas

Una vez he elegido cual va a ser el modelo candidato en el que voy a depositar mi confianza para apostar, solamente tengo que elegir en que casa de apuestas voy a hacer efectivas mis

predicciones. No obstante, voy a valorar 4 posibles estrategias a la hora de invertir, donde voy a tratar de minimizar el riesgo y maximizar la rentabilidad. La rentabilidad vamos a llamarla Yield o rendimiento en inglés. La confianza en una apuesta es conocida como 'stake', pero no vamos a utilizar este término. Estas son las 4 estrategias:

1. **Vamos a por todas**: Confiamos en todas nuestras predicciones.
2. **Estrategia A**: Confiamos en las predicciones con una probabilidad de clasificación correcta de la clase desprendida por el algoritmo (predict_proba) igual o superior al 66%.
3. **Estrategia B**: Confiamos en las predicciones con predict_proba superior al 74%
4. **Estrategia C**: Confiamos en las predicciones con predict_proba superior al 80%

Vamos a elegir la casa de apuestas Bet365 y visualizamos los resultados:

```
Confiamos en todas nuestras predicciones: € 695.0
Yield %  4.09 Total Apostado:  17000

Estrategia A, probabilidad superior al 66%: € 68.0
Yield %  2.72 Total Apostado:  2500

Estrategia B, probabilidad superior al 74%: € 38.0
Yield %  19.0 Total Apostado:  200

Estrategia C, probabilidad superior al 80%: € 0.0
Yield %  nan Total Apostado:  0
```

27.Figura 5.9.A Resultados aplicación de las apuestas.

En este caso, la estrategia B es muy interesante, podemos conseguir un Yield del 19% apostando 200€, es la opción que yo elegiría. En el caso de ir a por todas, obtenemos mayores ganancias, pero asumimos mucho más riesgo apostando 17.000€.

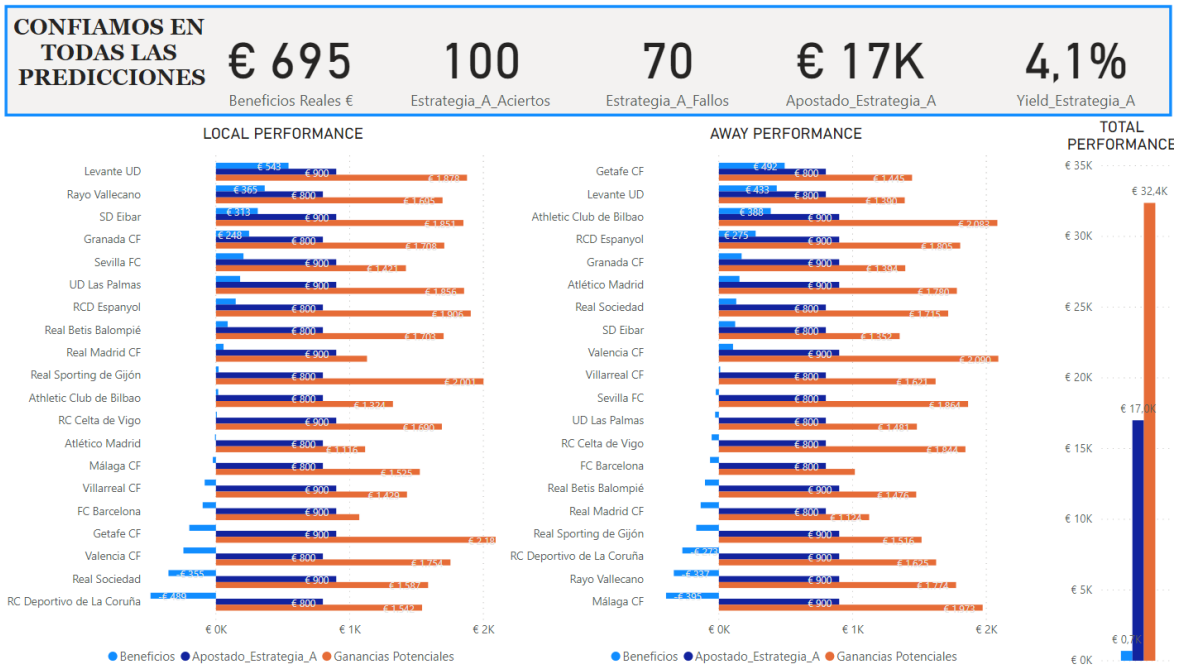
6 Dashboard

6.1 Dashboard de usuario final

Para poder visualizar los resultados en detalle, ver que equipos nos generan más beneficios, comparar estrategias y ver de un vistazo los principales KPI'S, se ha creado un Dashboard donde poder visualizarlo. Este Dashboard lo he implementado en Power Bi, se trata de una herramienta de visualización muy completa que es propiedad de Microsoft. La versión de escritorio es gratuita pero solo está disponible para sistemas operativos Windows, sin embargo, podemos trabajarla en la nube accediendo desde cualquier navegador y sistema operativo.

6.1.1 Confiamos en todas las predicciones

Podemos ver de un vistazo los principales KPI´S, con el beneficio que nos está generando cada uno de los equipos en sus actuaciones como local y visitante.



28.Figura 6.1.1.A Pantalla del Dashboard.

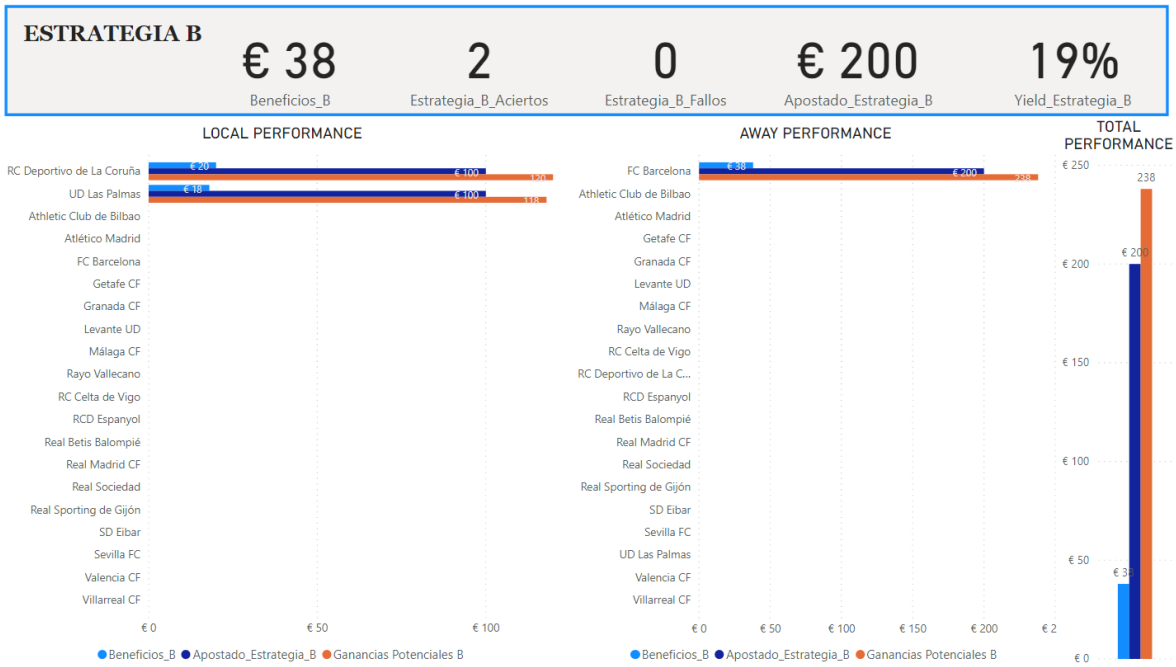
Para facilitar el entendimiento de los datos adjunto una tabla con el ejemplo del beneficio generado por el Barcelona. Se han acertado 7 de 9 partidos, pero al ser las cuotas para este equipo muy bajas se han obtenido pérdidas por valor de -91€.

Home_team_name	Away_team_name	target	Beneficio	Ganancias Posibles	Apostado
FC Barcelona	Athletic Club de Bilbao	1	15	115	100
FC Barcelona	Atlético Madrid	1	53	153	100
FC Barcelona	Getafe CF	1	4	104	100
FC Barcelona	Granada CF	1	2	102	100
FC Barcelona	RC Celta de Vigo	1	10	110	100
FC Barcelona	Real Madrid CF	*-1*	*-100*	162	100
FC Barcelona	Real Sporting de Gijón	1	5	105	100
FC Barcelona	Sevilla FC	1	20	120	100
FC Barcelona	Valencia CF	*-1*	*-100*	112	100
			-91	**1.083**	**900**

12.Tabla 6.1.1.B Ejemplo Beneficio Generado.

6.1.2 Estrategia B

En esta estrategia, hemos querido asegurar nuestras apuestas. Hemos seleccionado solamente las que nos daban una probabilidad de victoria de uno de los equipos igual o superior 80%. Tan solo hemos apostado a dos partidos y hemos acertado en ambos.



29.Figura 6.1.2 Pantalla del Dashboard.

6.1.3 Visualización de pronósticos

En la siguiente pantalla del Dashboard visualizaremos los partidos ordenados por fecha, con el detalle de cuál es el equipo local, cual es el equipo visitante, cual es el pronóstico que ha calculado el modelo y las probabilidades que estima el modelo para cada uno de los tres posibles resultados. Podríamos añadir cualquier dato adicional que consideremos relevante.

date	Home_team_name	Away_team_name	y_pred	probab_win	probab_draw	probab_lose
sábado, 9 de enero de 2016	FC Barcelona	Granada CF	1	0,698561748	0,230391409	0,071046843
sábado, 9 de enero de 2016	Getafe CF	Real Betis Balompié	1	0,588859328	0,204786402	0,20635427
sábado, 9 de enero de 2016	Levante UD	Rayo Vallecano	1	0,604479536	0,172179414	0,22334105
sábado, 9 de enero de 2016	Real Madrid CF	RC Deportivo de La Coruña	1	0,693099449	0,226097483	0,080803068
sábado, 9 de enero de 2016	Sevilla FC	Athletic Club de Bilbao	1	0,460805148	0,241870603	0,297324249
domingo, 10 de enero de 2016	RC Celta de Vigo	Atlético Madrid	-1	0,290651871	0,243854719	0,46549341
domingo, 10 de enero de 2016	Real Sociedad	Valencia CF	1	0,467553517	0,262677203	0,269769281
domingo, 10 de enero de 2016	SD Eibar	RCD Espanyol	1	0,505212955	0,219050373	0,275736672
domingo, 10 de enero de 2016	UD Las Palmas	Málaga CF	1	0,38059558	0,205021708	0,414382712
domingo, 10 de enero de 2016	Villarreal CF	Real Sporting de Gijón	1	0,621465109	0,227338606	0,151196285
sábado, 16 de enero de 2016	RC Celta de Vigo	Levante UD	1	0,589991512	0,252960284	0,157048204
sábado, 16 de enero de 2016	Real Sociedad	RC Deportivo de La Coruña	1	0,551499745	0,204036114	0,24446414
sábado, 16 de enero de 2016	Sevilla FC	Málaga CF	1	0,574497967	0,219228954	0,206273079
sábado, 16 de enero de 2016	Villarreal CF	Real Betis Balompié	1	0,622359481	0,198546362	0,179094157
domingo, 17 de enero de 2016	FC Barcelona	Athletic Club de Bilbao	1	0,702219714	0,19456079	0,103219496
domingo, 17 de enero de 2016	Getafe CF	RCD Espanyol	1	0,574976844	0,211247294	0,213775862
domingo, 17 de enero de 2016	Real Madrid CF	Real Sporting de Gijón	1	0,710874978	0,221893905	0,067231117
domingo, 17 de enero de 2016	UD Las Palmas	Atlético Madrid	-1	0,17602236	0,242854968	0,581122672
domingo, 17 de enero de 2016	Valencia CF	Rayo Vallecano	1	0,64668047	0,203753425	0,149566105
lunes, 18 de enero de 2016	SD Eibar	Granada CF	1	0,512418385	0,221816903	0,265764712
viernes, 22 de enero de 2016	Real Sporting de Gijón	Real Sociedad	1	0,391416527	0,169745204	0,438838269
sábado, 23 de enero de 2016	Granada CF	Getafe CF	1	0,485134035	0,25528448	0,259581485
sábado, 23 de enero de 2016	Málaga CF	FC Barcelona	-1	0,081956267	0,243811647	0,674232086
sábado, 23 de enero de 2016	Rayo Vallecano	RC Celta de Vigo	1	0,567216531	0,198021353	0,234762116
sábado, 23 de enero de 2016	RCD Espanyol	Villarreal CF	-1	0,283293207	0,265468737	0,451238056
domingo, 24 de enero de 2016	Athletic Club de Bilbao	SD Eibar	1	0,588767857	0,230408245	0,180823898
domingo, 24 de enero de 2016	Atlético Madrid	Sevilla FC	1	0,436964892	0,247572063	0,315463045
domingo, 24 de enero de 2016	RC Deportivo de La Coruña	Valencia CF	1	0,430549721	0,221434686	0,348015593
domingo, 24 de enero de 2016	Real Betis Balompié	Real Madrid CF	-1	0,056659085	0,261477175	0,681863739
lunes, 25 de enero de 2016	Levante UD	UD Las Palmas	1	0,512771452	0,186654364	0,300574184
sábado, 30 de enero de 2016	FC Barcelona	Atlético Madrid	1	0,667768709	0,184635873	0,147595418
sábado, 30 de enero de 2016	Getafe CF	Athletic Club de Bilbao	1	0,435517888	0,229010814	0,335471299
sábado, 30 de enero de 2016	Real Sociedad	Real Betis Balompié	1	0,632312299	0,188065848	0,179621853
sábado, 30 de enero de 2016	SD Eibar	Málaga CF	1	0,432842241	0,243051632	0,324106128

30.Figura 6.1.3 Pantalla pronósticos.

7 Conclusiones y trabajo futuro

7.1 Conclusiones

El objetivo de este trabajo era el diseño, implementación y validación de un método predictivo de resultados en eventos deportivos y su posterior aplicación en casas de apuestas deportivas.
El objetivo está cumplido, el problema es que después de todo el trabajo realizado, podemos concluir que 'no es tan fácil ganar a la banca'. A pesar de todo el análisis realizado con los datos disponibles en este trabajo, no he conseguido completar un modelo de análisis lo suficientemente maduro para poder concluir que podemos sacar rentabilidad regularmente y de forma sostenible apostando contra la 'banca'. Las casas de apuestas llevan muchos años trabajando en esto y desarrollando algoritmos para batir a los usuarios, por lo que generar beneficios apostando a la larga no es factible en términos de probabilidad. Pese a que acertamos más partidos de los que fallamos, si en los partidos que acertamos no tenemos una buena combinación de cuotas bajas y cuotas altas, los partidos que fallamos se comen los beneficios y es muy difícil ganar dinero.

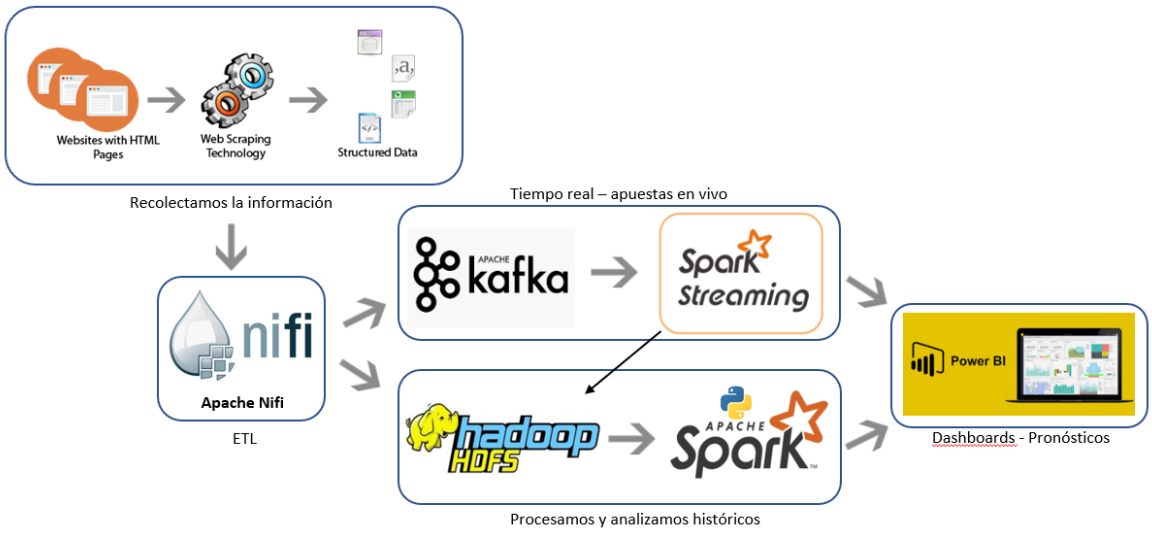
7.2 Trabajo futuro

Para poder sacar conclusiones más positivas y conseguir una mayor robustez en el análisis tengo que seguir trabajando.
Uno de los caminos sería añadir más información y crear nuevas variables que sean clave en las predicciones, aumentar el enfoque trabajando sobre nuevas apuestas como número de tarjetas a lo largo de un partido, número de corners, primer equipo en anotar goles, que equipos van a anotar goles…
Otra línea de desarrollo de este trabajo puede ser aplicar este modelo con apuestas del tipo 1X o 2X, en las cuales reducimos mucho la cuota, pero aumentamos las probabilidades de ganar, en todos aquellos partidos que no obtengan una probabilidad de acierto determinada les asignaríamos una cuota doble asegurando una tasa de acierto superior.
Tal y como he comentado en el punto 2.3.4 otra idea es empezar a trabajar con eventos en vivo, donde podemos analizar cuotas más suculentas y pueden aparecer más oportunidades. Para desarrollar este punto deberíamos desarrollar una arquitectura que nos permita realizar la ingesta, el procesado y las predicciones de resultados prácticamente en tiempo real.

7.2.1 Arquitectura ideal

Para poder convertir este entorno de producción en un entorno de desarrollo y comenzar a trabajar en las líneas que acabamos de comentar necesitaríamos una arquitectura más profesional, más completa y con mayor similitud a un entorno real de Big Data. La arquitectura debe definir la estructura estática de un sistema y su comportamiento dinámico. Voy a plantear un ejemplo que pueda cumplir con los requerimientos funcionales (capacidades necesarias por los usuarios de la solución para cumplir con el objetivo), los requerimientos no funcionales (expectativas que debe cumplir, cualidades y limitaciones) y los requerimientos futuros (procesamiento de información en tiempo real). La arquitectura propuesta va a emular a una de las más comunes en este tipo de proyectos; una Arquitectura Lambda [20]. Con un sistema tolerante a fallos, escalable linealmente y con capacidades de lectura y escritura con baja latencia. Tendrá dos capas, una de procesamiento en tiempo real que solo trabaja con nuevos datos, y otra de procesamiento en batch, que almacena los datos históricos.

- La captura de datos realizaríamos mediante técnicas de Web Scrapping, extrayendo la información del código HTML.
- La extracción, transformación y carga corre a cuenta de Apache Nifi, sirve para automatizar el movimiento de datos entre sistemas diversos. Es compatible con fuentes de datos diversas y permite hacer seguimiento de los datos en tiempo real.
- Para la capa de procesamiento en tiempo real, combinaremos Apache Kafka, que es una plataforma distribuida de transmisión de datos que nos permite almacenar, procesar flujos, suscribirse a ellos o publicarlos; con Spark Streaming, que nos ofrece procesamiento de datos en streaming, tolerancia a fallos, escalabilidad y alto rendimiento.
- Para la capa de procesamiento en batch o de históricos, vamos a utilizar el framework Hadoop, que nos proporciona capacidad para almacenar y procesar grandes cantidades de datos con su sistema de ficheros distribuido, es tolerante a fallos, flexible y escalable. Junto con Hadoop utilizaremos Spark, que no almacena datos en sí mismo, pero permite leer datos desde sistemas de almacenamiento distribuidos HDFS. Con Spark dispondremos de un motor de procesamiento distribuido diseñado para ser muy rápido trabajando en memoria y que permite dividir y paralelizar el trabajo. Utilizaríamos una de las API que nos proporciona, PySpark.
- Por último, como herramienta de visualización utilizaremos PowerBi, que nos permite diseñar cuadros de mando o Dashboard con facilidad, tiene compatibilidad con múltiples fuentes de datos, y podemos incluso realizar diseños para la aplicación móvil de este programa.



31.Figura 7.2.1 Arquitectura propuesta.

Palabras clave: Scikit-learn, Aprendizaje automático, Big Data, Python, Inteligencia Artificial, PySpark, Hadoop, Spark, PowerBi, Kafka, Apache Nifi.

Referencias

[1] Sports Betting: Past, Present and Future - Part 1 by Jeremy Martin.

[2] Operadores con licencia en España junio de 2020 https://www.ordenacionjuego.es/es/operadores/buscar

[3] El País, 3 abril 2008, Madrid Autoriza la primera casa de apuestas. https://elpais.com/diario/2008/04/03/madrid/1207221856_850215.html

[4] El periódico Octubre 2019, El sector del juego supone el 0,9% del PIB en España https://www.elperiodico.com/es/economia/20191023/el-sector-del-juego-supone-el-09-del-pib-y-84700-empleos-7696444

[5] Ley 13/2011, de 27 de mayo, de regulación del juego.

[6] Operadores con licencia en España,

[7] Servicio gratuito de resultados estadísticas y apuestas de fútbol: https://www.football-data.co.uk/data.php

[8] Web especializada en valoración de jugadores con las estadísticas del videojuego Fifa. https://sofifa.com/

[9] Repositorio a partir del cual he investigado y obtenido diversos ficheros para construir mi base de datos. https://github.com/hugomathien/football-data-collection

[10] Imagen sacada de la web Rpubs https://rpubs.com/Avalos42/randomforest

[11] An introduction to Logistic Regression Analysis and Reporting. Chao-Ying Joanne Peng

[12] Web Towards Data Science: https://towardsdatascience.com/train-test-split-and-cross-validation-in-python-80b61beca4b6

[13] Página oficial Scikit-Learn Support Vector Machines: https://scikit-learn.org/stable/modules/svm.html

[14] Artículo publicado por Raúl Lopez sobre el Boosting en Machine Learning. https://relopezbriega.github.io/blog/2017/06/10/boosting-en-machine-learning-con-python/

[15] Artículo publicado en la web Towards Data Science abril 2019. https://towardsdatascience.com/https-medium-com-vishalmorde-xgboost-algorithm-long-she-may-rein-edd9f99be63d

[16] Imagen sacada de los apuntes de la asignatura Indexación Búsqueda y Análisis, en concreto a la parte de Multimedia impartida por Juan Carlos San Miguel.

[17] I. Goodfellow, Y. Bengio, and A. Courville, "Deep Learning", MIT Press, 2016 Sec 6 y 9 (disponible gratuitamente en http://www.deeplearningbook.org/)

[18] Widaman, K. F. (1993). Common factor analysis versus principal component analysis: Differential bias in representing model parameters? Multivariate Behavioral Research, 28, 263-311

[19] Artículo sacado de IArtificial.net. https://iartificial.net/ensembles-voting-bagging-boosting-stacking/

[20] Evolución de las Arquitecturas, artículo escrito por Paradigma Digital. https://www.paradigmadigital.com/techbiz/de-lambda-a-kappa-evolucion-de-las-arquitecturas-big-data/

Anexo A

Vamos a describir el contenido de cada uno de los ficheros y sus columnas que van a confeccionar nuestra base de datos:

Fichero Matches:

- Season = Temporada (años en los que transcurre la misma).
- Stage = Jornada
- Date = Fecha del partido (dd/mm/yy)
- Match_api_id = identificador único de partidos
- Home_team_api_id = identificador único de equipo local
- Away_team_api_id = identificador único de equipo visitante
- Home_team_goal = goles marcados por el equipo local
- Away_team_goal = goles marcados por el equipo visitante
- Player_X = Jugador x, aparecerán 11 jugadores locales y 11 jugadores visitantes.
- Goal = goles marcados en el partido
- Shoton = disparos a puerta
- Shotoff = disparos fuera de la porteria
- Foulcommit = faltas cometidas
- Card = tarjetas
- Cross = cambios de juego
- Corner = número de saques de esquina
- Possesion = posesión de la pelota

*Estos últimos 4 indicadores no están bien completados, es por ello que creamos una nueva tabla que va a contener esta información correctamente completada.

Indicadores de apuestas:

- B365H = Bet365 cuota de apuesta a la victoria del equipo local
- B365D = Bet365 cuota de apuesta al empate
- B365A = Bet365 cuota de apuesta a la victoria del equipo visitante
- BWH = Bet&Win cuota de apuesta a la victoria del equipo local
- BWD = Bet&Win cuota de apuesta al empate
- BWA = Bet&Win cuota de apuesta a la victoria del equipo visitante
- GBH = Gamebookers cuota de apuesta a la victoria del equipo local
- GBD = Gamebookers cuota de apuesta al empate
- GBA = Gamebookers cuota de apuesta a la victoria del equipo visitante
- IWH = Interwetten cuota de apuesta a la victoria del equipo local
- IWD = Interwetten cuota de apuesta al empate
- IWA = Interwetten cuota de apuesta a la victoria del equipo visitante
- LBH = Ladbrokes cuota de apuesta a la victoria del equipo local
- LBD = Ladbrokes cuota de apuesta al empate
- LBA = Ladbrokes cuota de apuesta a la victoria del equipo visitante
- PSH = Pinnacle cuota de apuesta a la victoria del equipo local
- PSD = Pinnacle cuota de apuesta al empate
- PSA = Pinnacle cuota de apuesta a la victoria del equipo visitante
- WHH = William Hill cuota de apuesta a la victoria del equipo local
- WHD = William Hill cuota de apuesta al empate

- WHA = William Hill cuota de apuesta a la victoria del equipo visitante
- SJH = Stan James cuota de apuesta a la victoria del equipo local
- SJD = Stan James cuota de apuesta al empate
- SJA = Stan James cuota de apuesta a la victoria del equipo visitante
- VCH = VC Bet cuota de apuesta a la victoria del equipo local
- VCD = VC Bet cuota de apuesta al empate
- VCA = VC Bet cuota de apuesta a la victoria del equipo visitante
- GBH = Gamebookers cuota de apuesta a la victoria del equipo local
- GBD = Gamebookers cuota de apuesta al empate
- GBA = Gamebookers cuota de apuesta a la victoria del equipo visitante
- BSH = Blue Square cuota de apuesta a la victoria del equipo local
- BSD = Blue Square cuota de apuesta al empate
- BSA = Blue Square cuota de apuesta a la victoria del equipo visitante

Fichero Country:

- id = Id único de país.
- name = Nombre del país.

Fichero League:

- id = Id único de cada Liga.
- Country_id = Id único de país
- name = Nombre de la liga.

Fichero Players (jugadores):

- id = Id del registro.
- Player_api_id = Id único de cada jugador
- Player_name = Nombre de cada jugador
- Birthday = Cumpleaños de cada jugador
- Height = Altura de cada jugador
- Weight = peso de cada jugador.

Fichero Player attributes:

- id = Id del registro.
- Player_api_id = Id único de cada jugador
- date = Fecha en la que se hace la valoración de los atributos.
- overall_rating = valoración total del jugador
- potential = potencial de evolución del jugador
- Para el resto de los atributos podemos consultar su significado en el repositorio indicado en la bibliografía [8].

Fichero Team:

- id = Id del registro.
- Team_api_id = Id único del equipo
- Team_long_name = nombre completo del equipo
- Team Short Name = nombre corto del equipo

Información adicional incluida:

Los siguientes ficheros los he incluido en la base de datos, van a ayudar a crear la tabla maestra full_matches, combinando toda la información que esta contiene con estas tablas complementarias.

- Card_details = detalle de tarjetas por Partido.
- Corner_details = detalle de los corners por partido.
- Posesión_detail = detalle de la posesión de balón de ambos equipos por partido
- Shotoff_detail = detalle de los tiros fuera realizados por cada equipo.
- Shoton_detail = detalle de los tiros a puerta efectuados por cada uno de los equipos.

Anexo B

Scripts y código del proyecto:

Scripts para la creación de la base de datos: Se han realizado scripts de creación de todas las tablas que contiene la base de datos y otro mediante el cual hacemos diversos joins para confeccionar una tabla más completa de partidos con todo el detalle de estadísticas de corners, disparos a puerta, tarjetas, posesión.

También tenemos disponible el notebook utilizado en el trabajo y el Dashboard creado en Power Bi.

https://github.com/guillermoarrabalmartinez/football