Análisis de estadísticas de NBA

Máster data analytics semipresencial

Bryan Eduardo De Sousa Velásquez

Juan García Bericua Goyanes

Nuria Heras Louzao

Victor Maestro Calvo

Alberto Sánchez Delgado

2020

INDICE

# INTRODUCCIÓN

# ANTECEDENTES

# HIPÓTESIS

# EXTRACCIÓN

4.1 INTRODUCCIÓN

4.2 METODOLOGÍA

4.3 EXTRACTORES

# MÉTRICAS DE RENDIMIENTO

5.1 AWS

5.2 WP

5.3 EVALUACIÓN DE ESTAS

# OTRAS VARIABLES (Alberto)

# MODELOS

7.1 EXPLORACIÓN, LIMPIEZA Y CONSTRUCCIÓN DE VARIABLES

7.2 SELECCIÓN DE VARIABLES

7.3 SELECCIÓN DE MODELOS

7.4 CONSTRUCCIÓN DE MODELOS

7.5 EVALUACIÓN

# CONCLUSIONES

# REFERENCIAS BIBLIOGRÁFICAS / WEBGRAFÍA

*“La evaluación del jugador de baloncesto en los partidos es un proceso que despierta mucha atención de los entrenadores e investigadores” (Sampaio, 2002).*

# INTRODUCCIÓN

En un mundo en el que hasta ahora hemos basado nuestras decisiones empresariales y personales en experiencias y estadísticas de pequeños conjuntos de información, hemos pasado a un mundo en el que disponemos de gran cantidad de información.

El deporte evoluciona a pasos agigantados, y por supuesto, de la mano de la constante transformación digital en la que nos encontramos. Cada vez hay un volumen mayor de información cuantificable y accesible, y cada vez tiene mayor peso en la toma de decisiones ¿a quién debo fichar?, ¿qué debo corregir en mi estrategia?

La NBA, principal liga de baloncesto del mundo, es, además de una competición deportiva, un negocio que mueve grandes cantidades de dinero. En este paradigma, las decisiones tácticas, estratégicas o relativas a la composición de la plantilla deben ser estudiadas en profundidad. El análisis de grandes cantidades de datos para convertirlos en información relevante resulta por tanto muy adecuado para esta situación. En los últimos años la estrategia de los equipos ha cambiado enormemente debido a los avances en el campo de la Analítica de Datos.

Resultan remarcables, por ejemplo, los éxitos de Steven Kerr al frente de los Golden State Warriors, y su refrán, "*Strength in Numbers*". Ese equipo hacía patente la transformación de la NBA, siendo especialmente importante la figura de Sammy Gelfand, Head of analytics del equipo. La estadística demostraba que había que evolucionar hacia un equipo que tirase más triples en los partidos y trabajase a la par para mejorar el rango de estos y sus porcentajes. Steven Kerr se pone por primera vez al frente de un equipo profesional en 2014 y consigue el campeonato en 2015, 2017 y 2018, gracias a una planificación basada en los datos.

Vemos un cambio en la ratio triples - tiros de campo que se acentúa a partir de 2016. Como mencionamos en la introducción, el primer campeonato de Golden State Warrios (2015) provoca un cambio de tendencia en la NBA en ese sentido. Por ese motivo nos parece adecuado tomar esta temporada como punto de partida de nuestro estudio.

En este estudio queremos aplicar los procesos de analítica de datos a la NBA. Necesitamos por tanto obtener la información que consideremos relevante y transformarla adecuadamente, añadiendo nuevas variables que nos permitan, por ejemplo, evaluar de una forma objetiva el rendimiento de los jugadores, y más allá de eso, su influencia en el resultado del partido.

El objetivo principal de nuestro estudio es la predicción del resultado de los partidos de la NBA. Queremos obtener un conjunto de datos suficientemente completo y aplicar técnicas de aprendizaje automático aplicado para predecir el resultado de los futuros encuentros. Además, podremos analizar la significancia estadística de algunas de las variables en el estudio, esto es, si realmente aportan o no en la predicción. En definitiva, crearemos un modelo estadístico a partir de los datos que obtengamos, es decir, una aproximación matemática a la realidad que construimos a través datos. Este proceso de aprendizaje automático será en nuestro caso supervisado, ya que etiquetaremos la variable objetivo en nuestro conjunto de datos, es decir, el resultado del partido (victoria / derrota).

# ANTECEDENTES

Todos los entrenadores e investigadores han buscado la forma de conseguir sacarle el máximo rendimiento a las estadísticas que se recogen durante el tiempo del partido, tanto de sus propios jugadores como del resto de equipos de la NBA. Dentro de los avances obtenidos en el desarrollo y aplicación de nuevos métodos para el análisis cuantitativo de datos deportivos, sobre todo utilizando técnicas estadísticas. Esa acumulación incalculable de datos requiere de conclusiones que nos permitan analizar el estado de nuestro equipo, valorar el impacto de nuestros jugadores dentro del colectivo, prever futuras líneas de actuación y hasta cambiar puntualmente el estilo de juego en función de las estadísticas propias y del equipo rival

Desde la implementación hace más de un lustro de la medición, utilización y comprensión de los datos de estadísticas convencionales como, puntos, rebotes, asistencias nos ha permitido ir conociendo cómo ha evolucionado el juego.

La incorporación de tecnología para recabar datos se ha ido realizado paulatinamente, aunque el mayor avance se produce en el año 2013, con la instalación de cámaras de registro de movimientos, para proporcionar a los entrenadores, jugadores y aficionados la información de todos los partidos.

Los principales estudios se centran en el ritmo de juego (se centra en el uso de las faltas, las posesiones, y las jugadas desarrolladas durante el tiempo de partido), los porcentajes de tiro (desarrollo de un algoritmo para conocer le valor medio de los tiros en el campo, contando con un valor extra para el lanzamiento de línea de tres) y el intento de conocer mediante el análisis de datos cual es el mejor ataque y la mejor defensa.

Siendo estos los temas principales, se han desarrollado varios estudios acerca de las pérdidas de balón y varios estudios sobre las votaciones para los jugadores en el draft y la evolución del juego de triples.

Entre los estudios que destacan por su notoriedad se encuentran, el estudio de Ibáñez y Col y Sampaio y Janeira ambos del 2003 en el que sus conclusiones es que los equipos que anotan más tiros de 2 puntos y recogen más balones debajo del arco obtienen un índice mayor de partidos ganados.

La tesis de Sampaio y Leite, recoge los datos del Eurobasket d 2005, en el que concluyen que las estadísticas más diferenciadora se situaba en la línea de tres, los puntos fallados y las asistencias. Con este estudio se planteaba que la situación ofensiva se concluía con un tiro cercano, con elevadas probabilidades de acierto, para minimizar los riesgos del tiro exterior y escogiendo las mejores opciones de lanzamiento.

Un estudio de Hierro (2002), tras analizar la liga ACB y la NBA, concluye que los equipos con mejores porcentajes de tiros de 2 y de 3, mayor número de faltas recibidas, tapones y asistencias, obtenían más victorias; no considerando el número de pérdidas de balón y los tiros libres como aspectos definitivos.

No todos los estudios se centran en las estadísticas, ya que algunos toman de referencia la edad de los jugadores en la canga, (García, Ibañez y Feu (2007)), así como encontrar la posición más influyente en los partidos. Este estudio (Sampaio et Al, 2009), determinó que los jugadores más importantes eran los bases titulares y los pívots suplentes, reflejándose su aportación en los lanzamientos de 3 puntos, los rebotes y los lanzamientos de 2 puntos.

Otra de las líneas de investigación que hemos encontrado en el análisis de la literatura científica, es el estudio de las victorias y las derrotas en función de la condición del equipo como local o visitante. En la mayoría de los casos, los equipos que juegan en casa, tienen mayor ventaja (Courneya y Carron, 1992; Sánchez et al., 2001; Smith, 2005), debido a factores como viajes realizados, el conocimiento de la cancha de juego, la presión percibida por los árbitros, la familiaridad con el estadio, el público y determinadas variables psicológicas

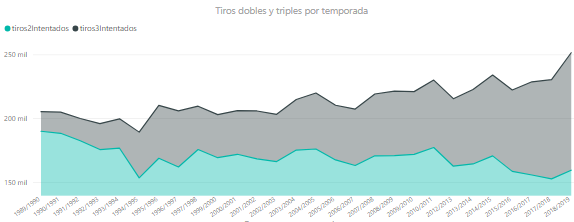
Diversos estudios cuentan con BBDD obtenidas de páginas de datos gratuitas, pero que en nuestro caso, hemos querido sacar nuestras propias variables, para así enriquecer este estudio. El objetico es la creación de un modelo que contenga como base la información más completa de estadísticas de los jugadores a nivel individual y como equipo.

**Cómo ha evolucionado el juego del baloncesto.**

El juego que se desarrolla actualmente en la NBA no es el mismo juego que se realizaba cuando se comenzaron a recoger datos. El análisis de datos de juego ha creado un antes y un después en el mundo del baloncesto. Tanto así que ha dado lugar a cambios importantes en la forma de jugar y las estrategias de los equipos.

La variación en la forma que se están obteniendo los datos y del año en el que comenzó a registrarse información está repercutiendo en el estilo de juego desarrollado por los equipos y en la forma de afrontar los partidos, buscando las estrategias más eficientes para conseguir el mayor número de puntos.

Con un simple vistazo a los datos de tiros por temporada a lo largo de los últimos 30 años, podemos ver que el volumen ha experimentado un gran aumento.



Esto se debe a que cada vez el juego es más rápido y por lo tanto da lugar a un mayor número de jugadas por partido. Y con ello, la oportunidad de anotar una mayor cantidad de tantos en cada encuentro.

Pero, ¿cómo podemos conocer que el juego es más rápido? ¿Se producen más intentos de tiro ahora que hace 20 años?

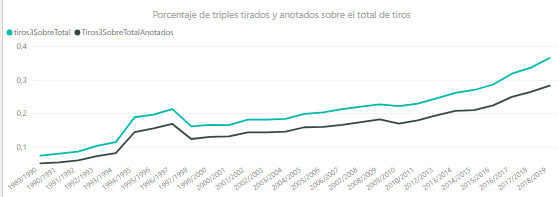
En el siguiente gráfico, comenzamos a analizar los puntos por temporada desde 1989/1990 hasta la temporada 2018/2019. La evolución se produce de forma lineal a lo largo del tiempo, teniendo un margen de crecimiento muy parecido en todos ambas métricas.



Sin embargo, el hecho de tirar más veces a canasta no es lo único que ha favorecido el aumento de puntos por temporada. Ha sido otro de los cambios tácticos importantes que ha dado la analítica de datos lo que más ha influido en este aumento.

Se trata de la importancia que se está dando a los tiros triples sobre los de 2 puntos, lo cual está invirtiendo el porcentaje de estos sobre el total de tiros, tanto en el caso de tiros intentados como en los anotados.

Como podemos ver en el siguiente gráfico, mientras en la temporada de 1989/1990, los triples sólo suponían un 7% sobre el total de tiros realizados y un 5% sobre los anotados, en la temporada de 2018/2019 estos porcentajes suponen un 37% y 28% respectivamente. Sobretodo se observa un aumento mucho más acusado en las últimas 5 temporadas, donde el incremento es de 11 y 7 puntos porcentuales respectivamente.



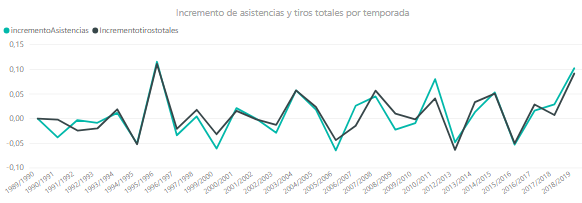
La respuesta a por qué se está llevando a cabo esta sustitución de tiros de dos puntos por tiros triples, se encuentra en la esperanza matemática que supone cada uno de ellos.

Como es evidente, el porcentaje de efectividad es superior en tiros de 2 puntos que en tiros de 3 puntos, -además este dato no ha experimentado grandes variaciones en el tiempo a pesar de que se esté tendiendo a una mayor práctica de tiros triples sobre dobles-, siendo de una media del 50% y del 35% respectivamente. Sin embargo, el hecho de anotar un triple, supone en el marcador un 50% más de puntos que un tiro doble.

Además, también se trata de una cuestión táctica. Al realizar el ataque desde fuera como es el caso de los triples, más tienen que abrirse los rivales. De esta manera resultaría más fácil atacar el aro, y por parte de la defensa, más difícil el apoyo entre los jugadores, ya que tienen que cubrir más terreno.

Una idea que podría surgir del hecho de que se esté tendiendo a un juego más basado en triples y se estén llevando a cabo jugadas más rápidas, es que el tipo de juego esté tendiendo a un modelo más individualista y menos colaborativo. Sin embargo, podemos descartar esa idea con un vistazo rápido a la evolución de los datos de tiros y asistencias en conjunto.

Como podemos observar en el siguiente gráfico, el incremento o disminución porcentual de cada temporada respecto a la anterior sigue una evolución muy similar entre ambas medidas, por lo que no podemos evidenciar un cambio en el juego hacia una tendencia más o menos individualista. Los datos de tiros totales y asistencias evolucionan en la misma dirección y en proporciones similares.



# HIPÓTESIS

Con la presentación de este trabajo final de Data analytics nuestro objetivo es la creación de un modelo predictivo que nos ayude a pronosticar el resultado de los partidos en la NBA, antes de que estos sucedan en función de las características de los equipos que compiten.

Sabemos que durante el desarrollo de un partido intervienen una gran cantidad de variables que puedan hacer cambiar el resultado final de un partido. Lesiones, acierto de tiros, faltas cometidas son algunas de las variables más importante que pueden llegar a afectar a los partidos.

Mediante este trabajo buscamos saber la importancia de los jugadores en cada partido, por ello debemos de analizar su contribución de puntos durante la temporada y los partidos jugados hasta el momento.

Nuestro enfoque va a consistir en los siguientes pasos; recopilar las métricas, procesar los datos obtenidos y usar varios métodos de clasificación. Queremos plantar un modelo que esté basado en métricas que añadan valor y que no incluyan algunas de las variables que se han desarrollado anteriormente en otros trabajos por científicos de datos.

Para ello, realizaremos una sinterización de los datos, buscando aquellas variables que si afectan a un partido y que como veremos tienen una importancia dentro del modelo y durante un partido. Creemos que nuestra valoración de los equipos va a tener significancia estadística en la predicción de los resultados.

Además, para conocer la importancia de cada jugador se creará una variable para determinar la fatiga de los jugadores y conocer la influencia que tienen los desplazamientos a los partidos durante el año.

# EXTRACCIÓN

4.1 INTRODUCCIÓN

Tras una primera tentativa de consumir la información de la API oficial de la NBA (data.nba.net), nos decantamos por emplear scraping sobre diferentes fuentes debido principalmente a dos problemas que nos encontramos:

1. Falta de consistencia de la API: presenta un año de inicio relativamente alto (2017) y deja de estar disponible para los partidos jugados desde el 13 de marzo de 2019, esto nos plantea un gran problema porque, ya que el fichero de entrada final tenía únicamente con unos 2000 registros para entrenar al modelo.
2. Problemas para añadir variables al DataFrame: tendríamos que la información obtenida en la API enriquecer con otras fuentes de datos, ya que algunas de las variables que se pretenden incluir en el estudio no están recogidas, esto nos plantea otro problema: los códigos que utilizan para hacer referencia a los equipos y jugadores no son fácilmente *mapeables*, de hacerlo, sería un proceso manual y costoso.

Esta primera versión de la capa de extracción del proyecto sirvió como prueba de concepto, para delinear nuestros objetivos, pero finalmente resultó insuficiente.

El scraping en cambio nos resulta tremendamente flexible a la hora de realizar las uniones entre las diferentes fuentes de datos, y las fuentes que escogemos resultan ser mucho más consistentes en cuando a su disponibilidad temporal. Solo requerimos de transformaciones menores para adaptar el nombre los jugadores de ESPN al formato de basketball-reference, para adaptar la fecha de basketball-reference al formato de shrpsports y la creación de dos mapas (diccionarios) que unifiquen el nombre del equipo (i.e. Chicago Bulls), el nombre abreviado (CHI) y el nombre de la ciudad (Chicago).

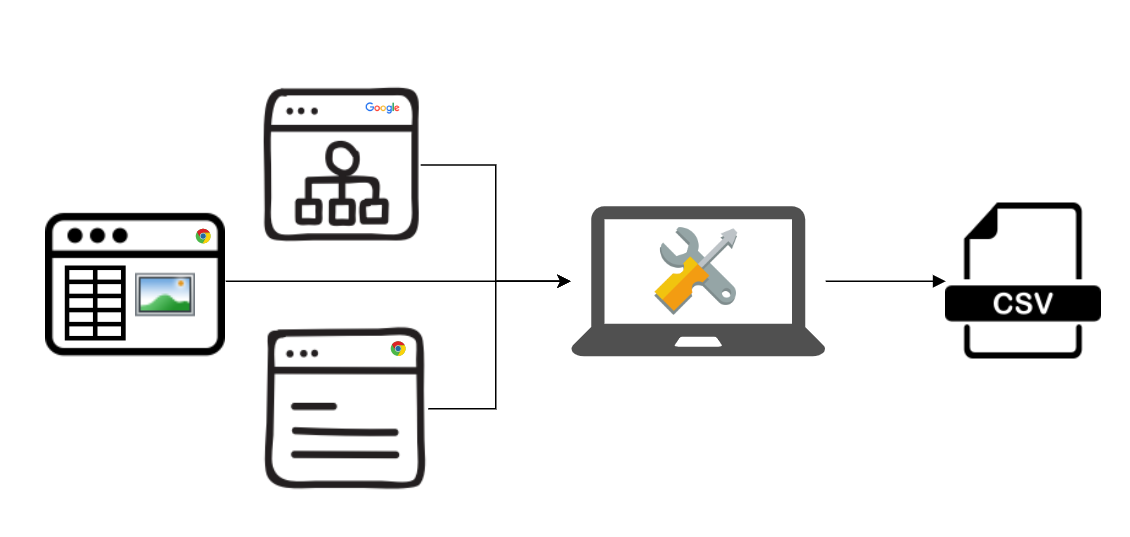
En resumen, habría sido un trabajo enormemente más sencillo consumir un json estructurado, con una única fuente de datos, y preocuparnos únicamente del número de peticiones por hora para evitar ser bloqueados, pero la cantidad de información sería insuficiente. Además, la significancia del proyecto sería menor, ya que existe una gran cantidad de estudios, como hemos comentado, que se limitan a estudiar los Boxscores y clasificaciones que la API suministra. Empleando este método podemos aportar nuevas variables al estudio que no se encuentran recogidas en ésta.

Con nuestro método conseguimos finalmente obtener 14000 registros (de 2010 en adelante) en el formato de entrada del modelo, frente a los 2000 de la primera versión, si bien decidimos posteriormente limitar ámbito temporal de nuestro estudio, debido a los cambios que se producen en el baloncesto en los últimos años, que hemos comentado anteriormente.

4.2 METODOLOGÍA

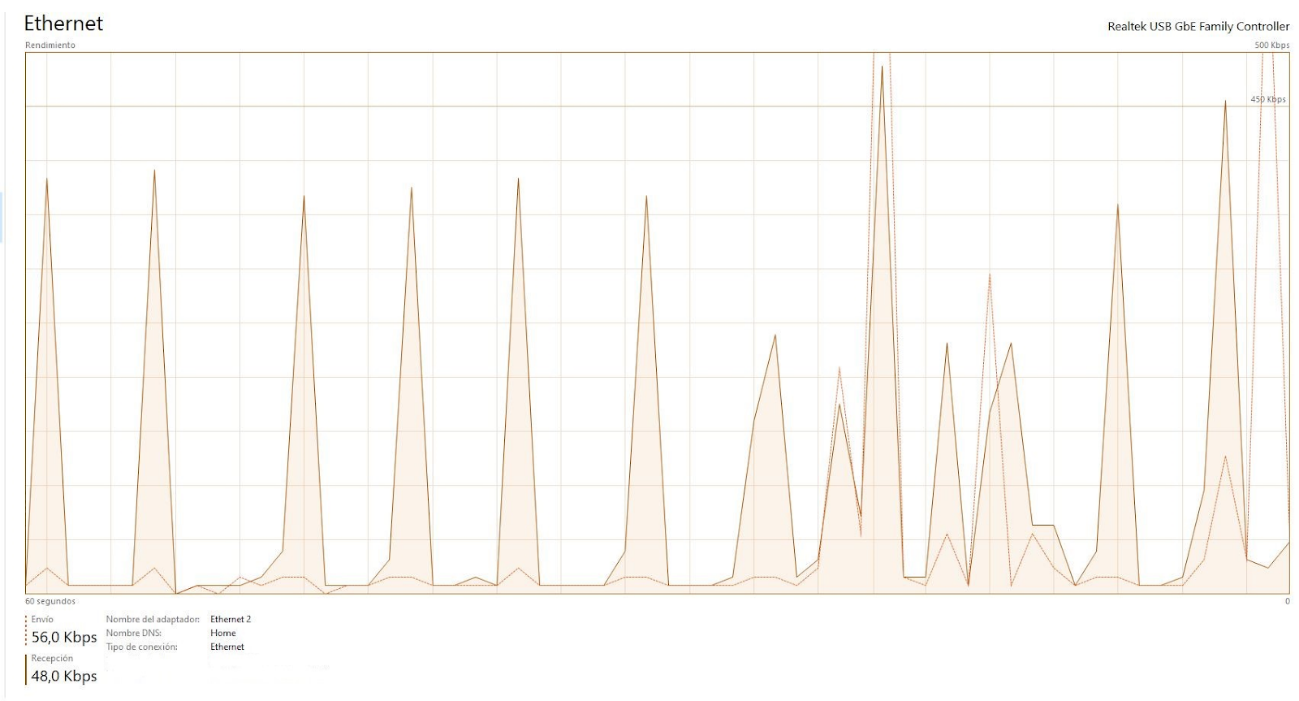
La extracción de los datos se lleva a cabo realizando peticiones GET a los endpoints mediante Python empleando las librerías request, y Pandas para recopilar la información en DataFrames, principalmente.

El proceso se basa en descomponer mediante la función "Split" la respuesta del endpoint para recopilar la información útil y desechar el resto, también se llevan a cabo en estos procesos unas transformaciones mínimas en los datos y uniones de DataFrames cuando es necesario, para dar lugar a ficheros csv que contengan la información que necesitamos para el análisis. Para dividir correctamente la respuesta de la request y generar el DataFrame hay que analizar el texto plano que llega y observar los patrones que repiten, de este modo conseguimos transformar la información no estructurada, o semiestructurada, a información estructurada con la que podemos realizar nuestro análisis.



Este proceso también nos permitirá acceder a una mayor cantidad de información que si empleásemos un DataFrame de los suministrados en portales como Kaggle, también nos permitirá incorporar más variables a nuestro estudio y obtener otra información que utilizaremos como referencia o control a algunas de las medidas de nuestro análisis.

La fuente de datos principal será www.basketball-reference.com. También, como ya hemos ido comentando, consumimos información de data.nba.net, de www.shrpsports.com y www.geodatos.net. Realizaremos las llamadas de forma que no sature a los receptores de las mismas, es decir, de manera secuencial (sin yield y sin emplear la librería scrapy), como podemos ver en la imagen. Como contrapartida nos encontraremos con procesos muy costosos en el tiempo.



Obtendremos mediante los extractores información relativa a:

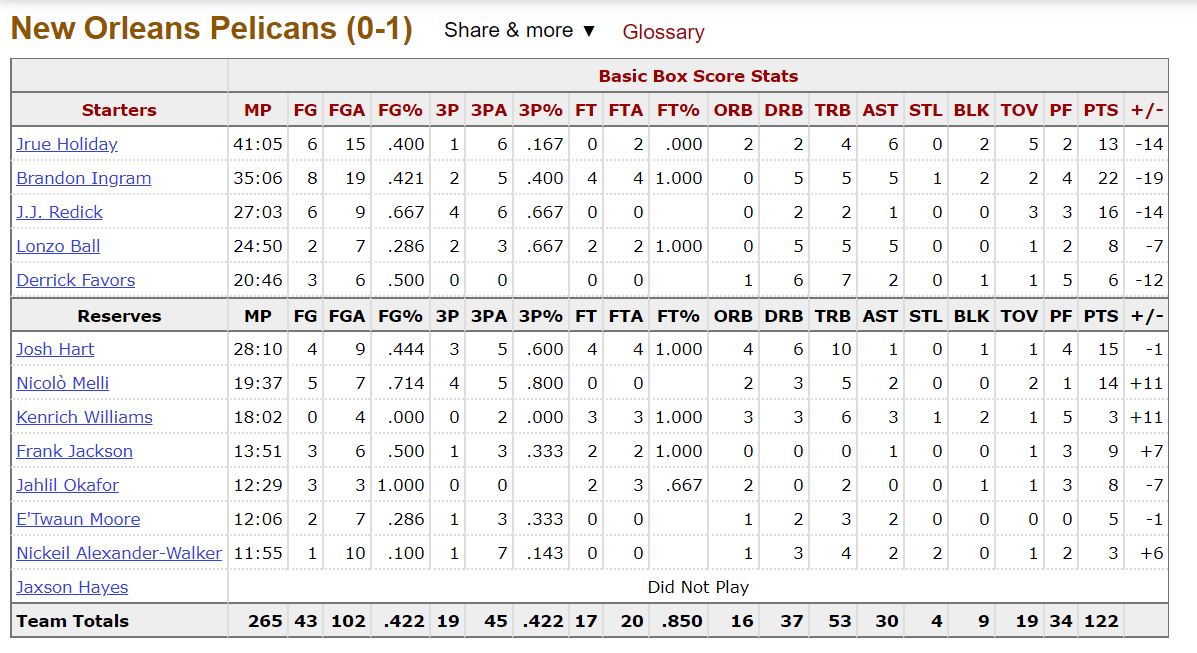
* Convocatoria de los partidos
* Clasificación previa al encuentro
* Estadísticas de los partidos (Boxscore)
* Nominaciones a los premios individuales
* Sueldos
* Distancias entre las ciudades que tienen equipo en la NBA
* Estadísticas al descanso
* Evolución del juego (estadísticas acumuladas de todos los equipos, por temporada)

4.3 EXTRACTORES

1. **StatsJugadoresV2.py**: Extrae la información de cada jugador y sus estadísticas en cada partido desde el 2010 a 2020. Para ello se realiza una llamada por equipo y temporada para obtener el calendario del equipo y de manera recursiva realiza llamadas al detalle de cada partido del calendario para obtener las siguientes variables.

* MP: Minutos jugados
* FG: Tiros de campos anotados
* FGA: Tiros de campo intentados
* FG%: Porcentaje de tiros de campo
* 3P: Triples anotados
* 3PA: Triples intentados
* 3P%: Porcentaje de triples
* FT: Tiros libres anotados
* FTA: Tiros libres intentados
* FT%: Porcentaje de tiros libres
* ORB: Rebotes ofensivos
* DRB: Rebotes defensivos
* TRB: Rebotes totales
* AST: Asistencias
* STL: Recuperaciones
* BLK: Tapones
* TOV: Pérdidas
* PF: Faltas personales
* PTS: Puntos
* +/-: Más menos (marcador parcial con el jugador en pista)
* TS% -- True Shooting Percentage
* eFG% -- Effective Field Goal Percentage
* 3PAr -- 3-Point Attempt Rate
* FTr -- Free Throw Attempt Rate
* ORB% -- Offensive Rebound Percentage
* DRB% -- Defensive Rebound Percentage
* TRB% -- Total Rebound Percentage
* AST% -- Assist Percentage
* STL% -- Steal Percentage
* BLK% -- Block Percentage
* TOV% -- Turnover Percentage
* USG% -- Usage Percentage
* ORtg -- Offensive Rating
* DRtg -- Defensive Rating
* BPM -- Box Plus/Minus

En la imagen vemos un ejemplo de un Boxscore a scrapear, en la imagen solo se muestran algunas de las variables (puede verse completo en https://www.basketball-reference.com/boxscores/201910220TOR.html):



1. **StatsJugadoresV2\_h1.py**: Con este proceso buscamos documentar la información de cada jugador y sus estadísticas al descanso en cada partido desde 2010 a 2020. Para ello realizamos una llamada para obtener el calendario del equipo, que nos facilita la URL del Boxscore de cada partido. Esta información viene a nivel de jugador, debemos agrupar por equipo y unir las filas de las variables del equipo local a las del visitante. Recogemos la siguiente información:

* FG: Tiros de campos anotados
* FGA: Tiros de campo intentados
* FG%: Porcentaje de tiros de campo
* 3P: Triples anotados
* 3PA: Triples intentados
* 3P%: Porcentaje de triples
* FT: Tiros libres anotados
* FTA: Tiros libres intentados
* FT%: Porcentaje de tiros libres
* ORB: Rebotes ofensivos
* DRB: Rebotes defensivos
* TRB: Rebotes totales
* AST: Asistencias
* STL: Recuperaciones
* BLK: Tapones
* TOV: Pérdidas
* PF: Faltas personales
* PTS: Puntos

1. **ResultadosClasificacionv2.py:** Extrae la información relativa a la clasificación en todas las fechas posibles desde octubre de 2009 a abril de 2020. Los datos los obtenemos a través de la realización de una serie de llamadas a [www.shrpsports.com](http://www.shrpsports.com). Recogemos los siguientes campos:

* Fecha (mes y día)
* Año
* Temporada
* Equipo
* Posición de conferencia
* Victorias
* Derrotas
* Porcentaje victorias / derrotas
* Diferencia con el líder de conferencia
* Victorias como local
* Derrotas como Local
* Victorias intra-división
* Derrotas intra-división
* Victorias intra-conferencia
* Derrotas intra-conferencia
* Victorias inter-conferencia
* Derrotas inter-conferencia

También hemos realizado las mismas llamadas en el extractor StatsjugadoresV2.py para obtener los mismos datos que obtenemos en este extractor a nivel de equipo, para el local y el visitante.

Para adaptar las nomenclaturas, hemos realizado un diccionario con los nombres de cada equipo en cada fuente (shrsports y basketball-reference,com) para poder realizar la unión de estos datos en un único dataframe.

Es importante indicar en este punto que la unión se realiza de modo que en cada línea del Dataframe se recoja el resultado del partido, las estadísticas de los equipos en el mismo y la clasificación previa al encuentro.

1. **DNP.py:** Incluye la información de qué jugadores (nombre y equipo) no participaron en cada partido durante el campeonato. Para ello realizamos llamadas para obtener el calendario y para obtener el Boxscore de cada partido, recogiendo la información del jugador que no jugó y el ID del partido.
2. **Premios.py:** Recoge la información de los jugadores que han recibido votos para recibir el premio MVP (premio a mejor jugador del año) desde 2010 hasta 2020. Se registra el nombre del jugador, el número de votos en primera posición, los puntos obtenidos, los puntos máximos y el share de éstos.
3. **Distancias.py:** Mediante llamadas a [www.geodatos.net](http://www.geodatos.net/) obtenemos las distancias entre las ciudades de todos los equipos que componen la NBA. Realizamos las llamadas iterando sobre la misma lista de equipos (que se relacionan con la ciudad mediante un diccionario) dos veces, bajo la condición de que el equipo no sea el mismo. Obtenemos así la distancia de Equipo A al Equipo B en kilómetros. Nos servirá para incluir alguna variable que recoja la distancia viajada por el equipo visitante en el modelo.
4. **Evolucion.py:** Recoge las estadísticas totales de la NBA en cada temporada para realizar un análisis sobre la evolución en el baloncesto NBA desde 1990 hasta 2020. La fuente de estos datos es [www.basketball-reference.com](http://www.basketball-reference.com/). Este análisis, como hemos visto anteriormente, nos ayuda a establecer un límite temporal a la información que incluimos en el modelo.

* FG: Tiros de campos anotados
* FGA: Tiros de campo intentados
* FG%: Porcentaje de tiros de campo
* 3P: Triples anotados
* 3PA: Triples intentados
* 3P%: Porcentaje de triples
* FT: Tiros libres anotados
* FTA: Tiros libres intentados
* FT%: Porcentaje de tiros libres
* ORB: Rebotes ofensivos
* DRB: Rebotes defensivos
* TRB: Rebotes totales
* AST: Asistencias
* STL: Recuperaciones
* BLK: Tapones
* TOV: Pérdidas
* PF: Faltas personales
* PTS: Puntos

1. **Sueldos.py:** Este extractor obtiene la información de los sueldos de los jugadores de la NBA de 2010 en delante de www.espn.com/nba/salaries/. Exporta la información a nivel de jugador (nombre del jugador, el equipo, la temporada y el sueldo en dólares). La información en la web viene paginada con un número de páginas variable de un año a otro, de modo que el extractor recoge la información realizando llamadas que incrementan en uno el número de página hasta que se produce una excepción, momento en el que pasa al siguiente año.

Generación del fichero de entrada:

Una vez completado todo el proceso de extracción de datos, tenemos que generar un fichero que sirva de input al modelo predictivo recogiendo toda la información. El fichero final contendrá la siguiente la información.

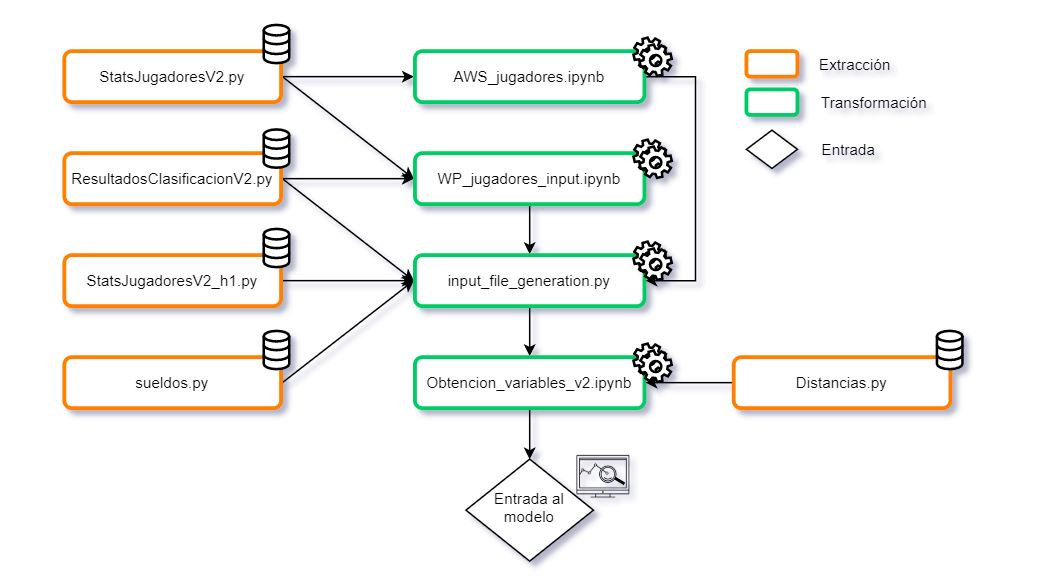
|  |  |
| --- | --- |
| Información | extractor |
| Clasificación antes del partido (posición, rachas, etc) | ResultadosClasificacionV2.py |
| Boxscore al descanso | StatsJugadoresV2\_h1.py |
| Suma de los sueldos del equipo | Sueldos.py |
| AWS | StatsjugadoresV2.py |
| WP | StatsjugadoresV2.py y ResultadosClasificacionV2.py |
| Divisiones y conferencias | - |
| Boxscore medio agrupado antes del encuentro | ResultadosClasificacionV2.py |

Además, durante este proceso, hemos eliminado toda la información duplicada o innecesaria, así como llevado a cabo la adaptación de toda la información al formato local /visitante.

Además de los campos obtenidos directamente en el proceso de scraping se realizan transformaciones para obtener los siguientes:

* Ultimos10Victorias: las victorias conseguidas en los últimos diez partidos \* jugados.
* Ultimos10Derrotas: las derrotas conseguidas en los últimos diez partidos jugados.
* Sueldo: La suma del sueldo del equipo en la temporada.
* División: división a la que pertenece el equipo.
* Conferencia: conferencia a la que pertenece el equipo.
* AWS\_MEDIO\_AGRUPADO: La medida AWS será explicada más adelante, una vez calculada para cada jugador, hacemos un rolling mean y un shift, para obtener la media hasta antes del partido del jugador en la temporada, posteriormente se agrupan por equipo y se realiza el join, obteniendo una valoración conjunta de los jugadores que participan en el encuentro por equipo.
* WP\_MEDIO\_AGRUPADO: Se trata de una medida análoga al AWS.

El flujo de la extracción y transformación de datos para generar un fichero de entrada es el siguiente:



# MÉTRICAS DE RENDIMIENTO

5.1 AWS

Con el objetivo de enriquecer nuestro modelo de predicción de resultados, con nuevas variables que puedan aportar información sobre la calidad de los equipos que se van a enfrentar en el mismo, se incluye a las ya existentes, una medida de valoración de los jugadores que van a participar en el encuentro, tanto en la posición de local como en la de visitante.

En la literatura existe una cantidad ingente de estudios al respecto, en la que profesionales de distintas disciplinas académicas (economistas, periodistas, ingenieros, psicólogos…) contribuyen a la creación de numerosas formas de valorar de manera más objetiva el desempeño de cada jugador.

Por lo que, teniendo en cuenta las características de nuestro modelo y los datos de los que disponemos, tras una revisión de varios sistemas de valoración -tanto los que se utilizan de manera oficial en las ligas profesionales, como aquellos empleados por webs y analistas especializados en este ámbito-, hemos seleccionado los dos que mejor podrían encajar con nuestro estudio y mayor respaldo o consistencia teórica muestran respecto a las críticas observadas en la literatura.

El tipo de datos del que se basan las medidas de valoración de jugadores, podría diferenciarse en dos principalmente. Información recogida de las estadísticas de cada partido, que se resumen en el “box-score”, o información más detallada basada en cada jugada que se produce en un encuentro, detallada en el “play by play”.

Teniendo en cuenta el objetivo de nuestro modelo de predicción, y las variables de las que disponemos para su estimación, la variable que queremos generar deberá estar basada en datos a nivel de encuentro, y por lo tanto nos serviremos de la información recogida en el  “box-score” para su cálculo.

Por otro lado, ya que el objetivo de esta medida de valoración será calcular el desempeño de cada uno de los jugadores, independientemente de la posición de estos, esta deberá basarse en una evaluación combinada, es decir, teniendo en cuenta cuestiones del juego tanto ofensivas como defensivas.

La primera de las medidas que se han seleccionado sería el Alternate Win Score (AWS) creada por Daniel Lewin y Dan Rosenbaum como alternativa al Win Score de David Berri, con pequeñas modificaciones en los pesos atribuidos a algunas de las métricas.

AWS= Pts + BR +BP + 0,5 (As + TF – FPC) + 0,7(RO – CCI) + 0,3(RD – CCC) – 0,35C1I – 0,15C1C

La otra medida seleccionada sería el Wins Produced (WP) detallada en el libro Wages of Wins and Stumbling on Wins por David Berri, Martin Schmidt y Stacey Brook. Se trata de un  índice que relaciona las victorias, y la eficiencia ofensiva y defensiva de los equipos. Se construye a través de un modelo de regresión previo, que determina el peso de cada una de las variables, en función a su influencia sobre la victoria de los partidos.

Esta medida se construye en varios pasos. Primeramente, y una vez calculados los pesos por los que se va a ponderar cada una de las variables, se obtiene una primera medida a nivel de jugador con las variables del box-score, tanto del jugador en concreto como del equipo oponente. Posteriormente se realiza un ajuste a la medida, añadiendo variables a nivel de equipo. Y por último, un ajuste extra con variables calculadas a nivel de liga.

Teniendo en cuenta la necesidad concreta de este estudio, y el tipo de modelo que queremos estimar, la medida que nosotros vamos a calcular siguiendo este sistema, no contemplará el último ajuste a nivel de liga, ya que las variables de nuestro modelo se van  construyendo de acuerdo a los partidos que han tenido lugar desde el comienzo de la temporada hasta el partido en cuestión.

Algunas de las premisas que nos han llevado a seleccionar estos sistemas de valoración para la creación de una nueva variable para nuestro modelo, es que aunque está basado en ponderaciones lineales como otras técnicas que hemos valorado, Win Score, Alternate Win Score o Wins Produced, premia a los jugadores a partir de un cierto rango de

eficiencia. En este sentido, se han realizado críticas hacia los sistemas de valoración Eficiencia y  PER, por premiar la “ineficiencia” de los jugadores, ya que con unos porcentajes de acierto bajos pueden obtener valores altos en esos índices.

Estos sistemas  utilizan  un  criterio  basado  en  la conversión de la producción del jugador a  las victorias del equipo. Wins Produced  y Win Score, ambas creadas por David Berri, tienen una base econométrica. Sin embargo, la segunda de estas, recibió críticas sobre el planteamiento de la función de producción del jugador, dando con la creación del Alternate Win Score por parte de Lewin y Rosenbaum, quienes teniendo en cuenta que no todos los lanzamientos fallados son pérdidas de posesión, modifican la ponderación del valor de esos lanzamientos, y también de los rebotes ofensivos y defensivos, de manera que se diferencie entre estas dos opciones de posesión.

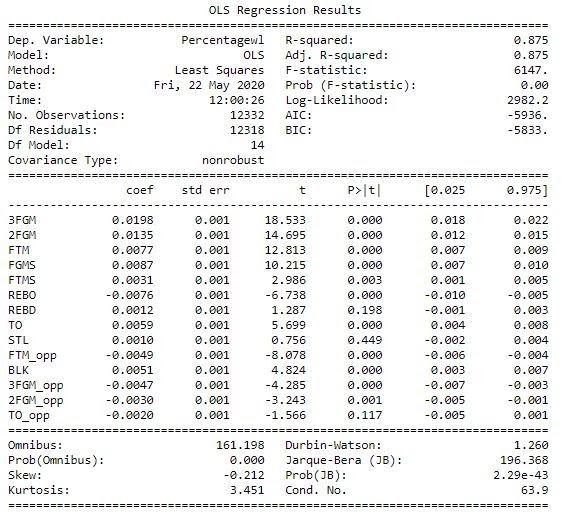
Para la creación de nuestra medida de valoración conjunta de los jugadores, se han calculado dos medidas de acuerdo a los dos sistemas comentados anteriormente. Siguiendo con la idea de valorar cómo están desarrollando la temporada los jugadores, la medida de valoración consistirá en la media -para cada uno de los jugadores- del desempeño que han realizado en cada encuentro en el que han participado. Posteriormente, para cada encuentro en cuestión, esta medida constará de la suma, de esas medias obtenidas para cada uno de los jugadores que va a participar en él.

Por lo tanto, el primer paso sería valorar el desempeño de cada uno de los jugadores en cada encuentro en el que han participado. En el caso de la variable AWS, esta valoración se obtendría aplicando la fórmula creada por lo autores y comentada anteriormente, con las estadísticas generadas por cada uno de los jugadores en cada partido.

De esta manera, teniendo el resumen al final del partido, de las variables que intervienen en el cálculo, podemos obtener una valoración del desempeño de cada jugador en el mismo.

5.2 WP

En el caso del sistema de valoración WP, primero debemos hacer una regresión de las variables del box-score contra el porcentaje de victorias del equipo. Para ello disponemos de un dataframe donde tenemos los resultados de todas estas medidas para cada uno de los encuentros de los últimos años. De esta manera, obtendremos la influencia de cada una de esas variables sobre la victoria.



Como podemos observar en el summary del modelo de regresión, las variables que tienen una mayor influencia positiva sobre el porcentaje de victorias, son aquellas que corresponden con la anotación de puntos, tanto dobles como triples.

De igual manera, las que tienen una influencia mayor sobre estas victorias, pero en una dirección opuesta, son aquellas que corresponden a estas mismas cuestiones pero para el equipo oponente.

Tiene sentido que los puntos anotados contribuyan a ganar el encuentro, mientras que encajar puntos del oponente signifique una mayor probabilidad de perder el encuentro. Siguiendo con este último punto, la victoria no consiste solo en un tema ofensivo si no también defensivo; es decir, intentar que el oponente anote la menor cantidad de puntos posible. Entre las variables defensivas, encontramos que los bloqueos al contrincante, es la medida que mayor influencia muestra sobre las victorias.

Sin embargo, los robos y los rebotes defensivos no parecen tener significatividad estadística individual en esta regresión.

Posteriormente, aplicaremos la fórmula del WP, ponderando cada una de las variables que la componen por la “influencia” obtenida en dicha regresión.

Aunque algunas de las variables no son significativas individualmente, todos los parámetros del modelo son conjuntamente significativos a cualquier nivel de confianza, por lo que incluiremos todas ellas en el cálculo del WP, siguiendo con el sistema creado por los autores.

Al igual que en el caso del AWS, esta medida se calcula para cada jugador en cada uno de los encuentros en los que ha participado, para poder valorar su desempeño en cada uno de ellos.

Para poder introducir estas dos variables en el modelo como medidas de valoración de la calidad de juego que están desempeñando los jugadores en cada momento de la temporada, y poder medir con ello el nivel de los equipos que se enfrentan en cada encuentro, -como se comentó anteriormente- una vez obtenido el valor de ambas para cada uno de los jugadores en cada encuentro en el que participó, se realiza una media con los partidos jugados desde el comienzo de la temporada hasta el partido en cuestión, y se suman esos valores obtenidos, de todos los jugadores que participan en el encuentro.

Al tratarse de dos medidas de valoración de jugadores basadas en las métricas del box-score, pero con distintas ponderaciones para cada una de ellas, podemos hacer una comparación visual de los resultados que arroja cada una de ellas antes de introducirlas entre las potenciales variables del modelo.

Las siguientes tablas muestran cuál sería el Top15 de jugadores en cada temporada según cada una de estas medidas. Para calcularlo, se ha aplicado la fórmula correspondiente a cada una de ellas, con la cifra total de la temporada para las variables que influyen en ellas.





5.3 EVALUACIÓN DE ESTAS

# OTRAS VARIABLES

Con el objetivo de enriquecer el modelo realizaremos transformaciones sobre los resultados de los extractores para obtener nuevas variables.

Para calcular estas nuevas variables se utilizarán tres orígenes:

* Input.csv que es el resultado de todas las extracciones y transformaciones anteriores.
* Distancias.csv que es el resultado del extractor Distancias.py
* stats\_jugadores\_27\_04\_20.csv que es el resultado del extractor StatsJugadoresV2.py

Estos archivos se guardan mediante el comando read\_csv en los siguientes dataframes:

* Input.csv 🡪 stats\_equipos
* Distancias.csv 🡪 distancias
* stats\_jugadores\_27\_04\_20.csv 🡪 stats\_jugadores

Teniendo en cuenta que en el input.csv el máximo nivel de granularidad viene dado por el id del partido, se eliminan todos los posibles duplicados.

En un primer lugar, se realiza una transformación de tipos en algunas de las variables que se detallan a continuación:

1. Se transforman las siguientes variables de float a int: con ello se pretende poder operar de forma correcta con los valores y evitar posibles fallos ya que las variables que se citan posteriormente no tiene ningún sentido que sean decimales.
   * Season
   * Puntos
   * Faltas
   * Perdidas
   * Robos
   * Asistencias
   * Rebotes total
   * Rebotes ofensivos
   * Tiros libres intentado
   * Tiros libres anotados
   * Tiros de tres intentados
   * Tiros de tres anotados
   * Tiros intentados
   * Team Points
   * Opponent Points
   * Tiros anotado
   * Rebotes defensivo
   * Tapones
2. Transformación variable Minutes Played: esta variable viene con el formato mm:ss siendo mm los minutos disputados por un jugador en un determinado partido y ss los segundos disputados por un jugador en un determinado partido. Para poder realizar operaciones con esta variable se transforma a un formato mm.ss de tipo float.
3. Transformación variable ID Partido: esta variable viene con información no relevante debido al proceso de extracción que genera ruido. Esta variable tiene el siguiente formato:

“/boxcores/fecha\_partido0abreviaturaequipoquejuegalocal.html”

Siendo:

* + fecha\_partido la fecha en la que se juega el partido en formato yyyymmdd
  + abreviaturaequipoquejuegalocal la abreviatura en tres letras del equipo que juega como local.

Con ello se realizan transformaciones para solo obtener el ID Partido con el formato:

“fecha\_partido0abreviaturaequipoquejuegalocal”

Una vez realizadas estas transformaciones se realiza un proceso de transformación de variables:

1. Obtención día partido: se calcula la columna number\_date para obtener la fecha en el que se juega el partido. Esta variable es importante porque permitirá realizar de forma sencilla diversos cálculos. Para extraer la fecha se realizan los siguientes pasos:
   1. Eliminación letras mediante el comando replace sustituyendo las letras por “” es decir, por nada. De forma análoga se elimina el carácter “.” y “/”.

* 1. Al ser las primeras ocho cifras las indicativas de la fecha se realiza un slice para extraerlas y se guardan en la columna “number\_date”.

1. Cálculo día anterior, dos días anteriores, tres días anteriores y previous\_10\_days: se calcula en cada registro que día ha sido el anterior, dos días anteriores y tres días anteriores. Esto se guardará en las variables previous\_date, two\_days\_ago y three\_days\_ago y permitirá realizar transformaciones para obtener el resto de variables que utilizará el modelo.
   1. Para calcular estas variables se utiliza timedelta de la librería datetime. A partir de la variable number\_date, anteriormente calculada, se le resta, mediante timedelta, uno, dos, tres o diez días según la variable que se desee calcular.
   2. Una vez ya tenemos las variables calculadas se transforman a formato datetime.
2. Cálculo variable prórroga: se desea calcular si un partido ha tenido prórroga o no. Para ello, se realizan los siguientes pasos:
   1. Se crea un dataframe llamado stats\_jugadores\_ordered con solo tres columnas con el ID\_Partido y las variables visitor\_prorrogue y local\_prorrogue. Este dataframe por lo tanto no tendrá como granularidad el trío de columnas jugador-equipo-id\_partido sino que será el id\_partido. Este dataframe, además, será mucho más manejable ya que su número de columnas es menor que al original.
   2. En la columna ID\_Partido del dataframe stats\_jugadores\_ordered se añade todos los ID\_Partido únicos es decir, se añaden los id\_partido sin repeticiones ya que debido a la naturaleza del daframe stats\_jugadores estos estaban repetidos.
   3. Se suma todos los minutos de cada jugador por partido y equipo en el dataframe stats\_jugadores. De esta forma, se obtendrán los minutos que ha durado el partido.
   4. Se recorre el dataframe stats\_jugadores\_ordered y se crea un dataframe en el que se guardan todos los registros del dataframe stats\_jugadores que coindicen con el ID\_Partido correspondiente del dataframe stats\_jugadores\_ordered.
   5. Si el índice mínimo del borrador es mayor que 50.0 significa que en ese partido ha habido prórroga por lo que la variable visitor\_prorrogue y local\_prorrogue se pone a 1. Esto es así ya que un partido dura 48 minutos y la prórroga 5 minutos lo que suman 53 min, por lo que el umbral para decidir si hay prórroga o no sería mayor que 48 min aunque se utilizará 50 min por si ha habido algún ligero descuadre en la extracción de los datos.
   6. Este dataframe mediante un merge se fusiona con el dataframe stats\_equipos.
3. Creación de variables a calcular: a partir de las variables anteriores se realizarán diferentes transformaciones que permitirán obtener variables que servirán para enriquecer el modelo. Para abreviar la explicación solo se detallarán las variables para el local, aunque para el visitante serán análogas. Estas variables son:

* local\_dif\_between\_previous\_game: servirá para calcular la diferencia de días respecto al último partido. Se calcula esta variable para evaluar si la diferencia de días respecto al partido anterior afecta al resultado del próximo partido.
* local\_played\_previous\_date: servirá para calcular si el local ha jugado el día anterior. Se calcula esta variable porque pensamos que el local juegue el día anterior puede tener influencia sobre el partido ya que puede estar más cansado.
* local\_played\_two\_days\_ago: servirá para calcular si el local ha jugado dos días antes. Se calcula esta variable porque pensamos que el que el local juegue dos días antes puede tener influencia sobre el partido ya que puede estar más cansado. Se diferencia del día anterior porque pensamos que la diferencia de significancia puede ser notable.
* local\_played\_three\_days\_ago: servirá para calcular si el local ha jugado tres días antes. Se calcula esta variable porque pensamos que el local juegue tres días antes puede tener influencia sobre el partido ya que puede estar más cansado. Se diferencia del día anterior y dos días anteriores porque pensamos que la diferencia de significancia puede ser notable.
* local\_played\_prorrogue\_previous\_date: servirá para calcular si el local ha jugado el día anterior prórroga. Se calcula esta variable porque pensamos que el que el local juegue el día anterior prórroga puede tener influencia sobre el partido ya que puede estar más cansado.
* local\_played\_prorrogue\_two\_days\_ago: servirá para calcular si el local ha jugado el dos días antes prórroga. Se calcula esta variable porque pensamos que el que el local juegue dos días antes prórroga puede tener influencia sobre el partido ya que puede estar más cansado. Se diferencia del día anterior porque pensamos que la diferencia de significancia puede ser notable.
* local\_played\_prorrogue\_three\_days\_ago: servirá para calcular si el local ha jugado tres días antes prórroga. Se calcula esta variable porque pensamos que el local juegue tres días antes prórroga puede tener influencia sobre el partido ya que puede estar más cansado. Se diferencia del día anterior y dos días anteriores porque pensamos que la diferencia de significancia puede ser notable.
* local\_played\_local: servirá para calcular si el local jugó como local su último partido. Se piensa que jugar varios partidos como local seguidos puede tener influencia en el resultado del partido actual.
* local\_played\_visitor: servirá para calcular si el local jugó como visitante su último partido. Se piensa que jugar varios partidos como local seguidos puede tener influencia en el resultado del partido actual.
* place\_local\_played: servirá para calcular donde jugó el local el último partido. Se calcula esta variable porque nos ayudará a calcular la distancia que ha recorrido el equipo entre partidos.
* avg\_distance\_place\_local\_played servirá para calcular la distancia que ha recorrido el local respecto al último partido jugado por él entre los días que han pasado desde su último partido.

1. Cálculos variables: ahora se tiene los datos para calcular las variables se recorre el dataframe stats\_equipos. Para realizar el cálculo de estas variables se realizan los siguientes pasos:
   1. Primero, se calculan los siguientes dataframes sobre los que se aplicarán diferentes cálculos:
   * Jugo\_dia\_anterior: se guardan todos los registros que coinciden con la variable previous\_date del registro en curso. De forma análoga se crean dos dataframes que son jugo\_dos\_dias\_antes y jugo\_tres\_dias\_antes pero con las variables two\_days\_ago y three\_days\_ago
   * Se calculan los dataframes visitor\_jugo\_dia\_anterior y local\_jugo\_dia\_anterior. Estos dataframes se calculan cogiendo el dataframe jugo\_dia\_anterior y viendo si coincide el visitante o el local están en la variable local\_team o visitor\_team con el local\_team o el visitor\_team del registro del dataframe stats\_equipo. De forma análoga se hace para los dataframes visitor\_jugo\_dos\_dias\_antes, local\_jugo\_dos\_dias\_antes, visitor\_jugo\_tres\_dias\_antes y local\_jugo\_tres\_dias\_antes.
   1. Una vez calculados estos dataframes se realizan las siguientes comprobaciones:
   * Si local\_jugo\_dia\_anterior no está vacío la variable local\_played\_previous\_date se pone a uno. Si, además, ese registro de local\_jugo\_dia\_anterior tiene local\_prorrogue a uno la variable local\_played\_prorrogue\_previous\_date se pone a uno.
   * Si visitor\_jugo\_dia\_anterior no está vacío la variable visitor\_played\_previous\_date se pone a uno. Si, además, ese registro de visitor\_jugo\_dia\_anterior tiene visitor\_prorrogue a uno la variable visitor\_played\_prorrogue\_previous\_date se pone a uno.
   * Para las variables local\_played\_two\_days\_ago, local\_played\_prorrogue\_two\_days\_ago, visitor\_played\_two\_days\_ago, visitor\_played\_prorrogue\_two\_days\_ago, visitor\_played\_three\_days\_ago, visitor\_played\_prorrogue\_three\_days\_ago local\_played\_three\_days\_ago y local\_played\_prorrogue\_three\_days\_ago, se trabaja de forma parecida al punto anterior pero con los dataframes visitor\_jugo\_dos\_dias\_antes, local\_jugo\_dos\_dias\_antes, visitor\_jugo\_tres\_dias\_antes y local\_jugo\_tres\_dias\_antes.
   1. Posteriormente, para calcular el resto de las variables se realizan los siguientes pasos:
   * Se calcula un dataframe diez\_dias\_previos con todos los registros de stats\_equipos que están entre la fecha anterior del partido y diez días antes. Así, acotamos en una determinada línea temporal el análisis y eliminamos casos extremos como principio de temporada, vacaciones o posibles parones donde han ocurrido acontecimientos hace más de diez días pero que consideramos que no influyen en el resultado.
   * Si el dataframe diez\_dias\_previos está vacío sus variables mantienen el valor por defecto. Si no está vacío se crean cuatro dataframes que son local\_jugo\_local, local\_jugo\_visitante, visitante\_jugo\_local y visitante\_jugo\_visitante. Estos dataframes se calculan a partir del dataframe diez\_dias\_previos comprobando si el local jugó como local o visitante y comprobando si el visitante jugó su último partido como local o visitante.
   * Si local\_jugo\_local o local\_jugo\_visitante no están vacíos quiere decir que el local jugó los diez anteriores.
     1. Si el registro con mayor fecha pertenece a local\_jugo\_local quiere decir que place\_local\_played es el campo del local, que avg\_distance\_between\_previous\_date es cero y además se resta el number\_date del registro actual menos la fecha del último partido que jugó como local para saber la diferencia de días con su último partido jugado.
        1. Si, además, local\_played\_local es mayor que -1 se suma uno a la variable local\_played\_local del registro actual de stats\_equipo y se pone visitor\_played\_local a cero.
        2. Si no es mayor que -1 local\_played\_local es uno y local\_played\_visitor cero.
     2. Si el registro con mayor fecha pertenece a local\_jugo\_visitante quiere decir que place\_local\_played es el campo del local de su útlimo partido. Además se resta el number\_date del registro actual menos la fecha del último partido que jugó como visitante para saber la diferencia de días con su último partido jugado.
        1. Si, además, local\_played\_visitor es mayor que -1 se suma uno a la variable local\_played\_visitor del registro actual de stats\_equipo y se pone local\_played\_local a cero.
        2. Si no es mayor que -1 local\_played\_visitor es uno y local\_played\_visitor cero.
     3. Si local\_jugo\_local no está vacío pero si está vacío local\_jugo\_visitante se hace un desarrollo parecido al anterior pero sin comparar si el local jugó como local o visitante su último partido ya que jugó como local.
     4. Si local\_jugo\_visitante no está vacío pero si está vacío local\_jugo\_local se hace el mismo desarrollo que en el punto iii. Pero sin comparar si el local jugó como visitante o local su último partido ya que jugó como visitante.
     5. Para las variables place\_visitor\_played, visitor\_dif\_between\_previous\_game, visitor\_played\_local y visitor\_played\_visitor se hace el mismo desarrollo que en los puntos iii, iv y v pero con los dataframes visitante\_jugo\_local y visitante\_jugo\_visitante.
   1. Una vez realizado el bucle anterior nos queda calcular la variable avg\_distance\_place\_visitor\_played y la variable avg\_distance\_place\_local\_played cuando el local no jugó su último partido como local. Para ello, se realizan los siguientes pasos:
   * Se cambia los nombres del dataframe distancias por Equipo A = local\_team y Equipo B = visitor\_team y se realiza un merge entre stats\_equipos y distancias por la clave local\_team y place\_local\_played. Con ello, ya sabemos la distancia que recorrió el equipo local respecto a su último encuentro que jugó en el caso de que lo hiciera como visitante.
   * Se cambia los nombres del dataframe distancias por Equipo A = visitor\_team y Equipo B = local\_team y se realiza un merge entre stats\_equipos y distancias por la clave visitor\_team y place\_visitor\_played. Con ello, ya sabemos la distancia que recorrió el visitante respecto a su último partido.
   * Se calculan las variables avg\_distance\_place\_visitor\_place dividiendo la distancia recorrida respecto a su último partido entre visitor\_dif\_between\_previous\_date y avg\_distance\_place\_local\_place dividiendo la distancia recorrida respecto a su último partido entres local\_dif\_between\_previous\_date

Por último, se eliminan las columnas previous\_10\_days, previous\_date, two\_days\_ago, three\_days\_ago, visitor\_prorrogue y local\_prorrogue y se guarda el dataframe en un archivo csv llamado input\_extendido.

Además, se planteó realizar un modelo que midiera la contribución de cada estadística de cada jugador al global del equipo pero finalmente se decidió no realizar por las siguientes razones:

* Se optaron por utilizar otras medidas ya predefinidas que permiten medir el desempeño global de un jugador y poder compararlos entre ellos.
* Demasiadas columnas para el análisis.
* Simplemente se iba a ver la influencia de cada estadística sobre el resultado, pero no se iba a poder clasificar que jugadores son mejores que otros.

# MODELOS

El objetivo de este apartado es tomar el output de todas las transformaciones realizadas y explicadas anteriormente y obtener un modelo que sea capaz de predecir qué equipo va a ganar un partido. Para ello, se toma como input el csv input\_extendido que es el resultado de todas las transformaciones realizadas y explicadas anteriormente.

**FALTA UNA BUENA INTRODUCCION**

7.1 EXPLORACIÓN, LIMPIEZA Y CONSTRUCCIÓN DE VARIABLES

En esta sección se realiza la preparación de los datos, limpieza del dataframe por valores nulos, renombrado de variables, cálculo de nuevas variables, descarte de variables que no forman parte de análisis, exploración gráfica y otros procesos. Haciendo uso de las librerías de Pandas y Numpy de Python.

En primer lugar, se importa el csv input extendido en un dataframe. Este dataframe inicial contempla 6220 filas y 180 columnas (variables). Después de un proceso de limpieza por valores nulos derivados del proceso de extracción y de cálculos de divisiones de denominadores iguales a 0, se ha obtenido finalmente 5809 (son menos hay que calcularlo si se considera necesario) filas para trabajar.

Seguidamente se pasa al cálculo de variables compuestas donde destaca, entre alguno de los cálculos, la creación de la variable “target” (objeto de estudio de valores binarios) a partir de la diferencia de los puntos marcados por el equipo local y el equipo visitante, de tal forma que, valores igual a 1 para indicar cuando el equipo local gana y 0 para el equipo visitante cuando pierde.

Entre las 180 variables que se han importado del .csv resaltan 2 variables categóricas que contienen la información de la división y la conferencia a la que pertenece el equipo local y el equipo visitante. Estas variables categóricas deben ser transformadas en variables dicotómicas (dummy´s). Esta transformación se ha realizado mediante el enfoque de One Hot encoding con funciones de la librería de Pandas. Dado que existen dos conferencias y seis divisiones en la NBA, han resultado 14 variables dicotómicas que sustituyen en cuanto a información a las dos variables originales.

Una vez identificadas, calculadas, transformadas, y organizadas todas las variables en diversas listas (variables continuas y dicotómicas) se decide filtrar las variables que a priori no aportan ningún valor al modelo, muchas derivan del proceso de extracción de datos y otras que, si bien tienen mucha información asociada, se han descartado por otras variables que en información ya las sustituye.

Básicamente un proceso de “Garbage in, garbage out” que nos permite trabajar específicamente con las variables de interés para el modelo.

Este proceso de selección se divide entre variables previas al partido y variables del partido se la siguiente forma:

(se puede hacer una especie de cuadro o algo más visual)

7.2 SELECCIÓN DE VARIABLES

En esta apartado se realiza la selección definitiva de variables para el modelo desde distintos enfoques que dependen del tipo de variable. Inicialmente se han separado las variables en diferentes listas; continuas y dicotómicas, así como, la separación del conjunto de entrenamiento y test para la evaluación del modelo final.

Este proceso de separación de los datos para el proceso de selección de variables se realiza para evitar extraer información del total conjunto de datos, ya que, posteriormente una parte de estos datos se deben usar para evaluar el modelo. Comúnmente en aprendizaje automático a este fenómeno se le conoce como “fuga de datos” (data leakege).

Haciendo uso de la librería de sklearn de Python se ha desarrollado todo el proceso de aprendizaje automático. Sobre la función train\_test\_split se han separado los datos en un conjunto de entrenamiento (70% equivalente a 4.066 partidos) y otro para test del modelo (30% equivalente a 1.743 partidos), recordando que el rango de temporadas seleccionadas para el estudio es 2016-2020.

Es importante resaltar que, durante la fase de análisis exploratorio (Parte I) se realizaron matrices de correlación sobre las variables continuas previas y durante el partido (por separado) sobre “target” para evaluar el posible grado de relación lineal que tienen las variables entre sí y la variable objetivo.

En este análisis de correlación no se han descartado variables porque en este estudio no se pretende realizar un análisis de causalidad entre variables explicativas y la variable objetivo. Este proceso de selección de variables tiene sentido en esos casos porque se pretende cuantificar la atribución de una variable respecto a otra y de esta forma el nivel de importancia.

En el caso de modelajes predictivos es un error excluir variables por presencia de altos niveles de correlación entre variables explicativas (excepto perfecta colinealidad entre independientes) porque se contribuye al deterioro de la precisión del modelo, esto se debe porque las variables en conjunto se complementan y comprenden más información que cada una de estas por separado.

Del mismo modo, el descarte de variables por escasez de correlación es debatible, depende en muchos casos el modelo a utilizar y de la relación establecida a priori entre la variable y la objetivo. El coeficiente de correlación de Pearson, por ejemplo, es una medida de la relación "lineal" entre dos variables, en este contexto, se puede decir que no necesariamente las variables tienen que estar relacionadas linealmente.

Asimismo, en el apartado anterior (Parte I) se han descartado variables que pretenden aportar la misma información en el modelo. En cualquier caso, de requerirse para modelos futuros, se ha dejado en el notebook funciones\_modelo.ipynb la función construida para Factor Inflación de la Varianza (VIF) que permite de forma automática, descartar variables explicativas dependiendo del nivel de correlación establecido.

A continuación, separamos el proceso de selección de variables como se ha comentado en el inicio de este apartado:

1. Selección de Variables Dicotómicas:

Después del proceso de transformación de variables categóricas comentado en la primera sección y sumado a las variables dicotómicas generadas en previos procesos del estudio, se han obtenido en total 26 variables dummy´s que se reparten en partes iguales entre el equipo local y visitante.

|  |  |
| --- | --- |
| played\_previous\_date | Conf\_Este |
| played\_two\_days\_ago | Div\_Atlantic |
| played\_three\_days\_ago | Div\_Central |
| played\_prorrogue\_previous\_date | Div\_Northwest |
| played\_prorrogue\_two\_days\_ago | Div\_Pacific |
| played\_prorrogue\_three\_days\_ago | Div\_Southeast |
|  | Div\_Southwest |

Estas variables se han calculado para enriquecer la complejidad del modelo, sin embargo, como es posible sospechar, muchas de estas pueden que no sean relevantes para predecir el resultado de un juego, y su inclusión en el modelo solo contribuyen a la generación de “ruido” (perturbaciones sobre las predicciones).

Por lo tanto, es necesario conocer cuáles de estas variables son las que más información aportan o afectan el comportamiento de la variable objetivo. Entre las distintas técnicas y conceptos estadísticos que existen se ha decidido utilizar la prueba de independencia de chi-cuadrado y la medida de información mutua.

La prueba de independencia de ji-cuadrado (chi-cuadrado) para variables categóricas contrasta la hipótesis nula de que las variables dentro de la prueba son independientes a través de la distribución que presenta una frente a la otra mediante tablas de contingencia.

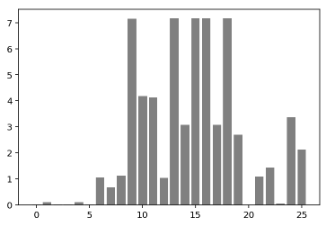
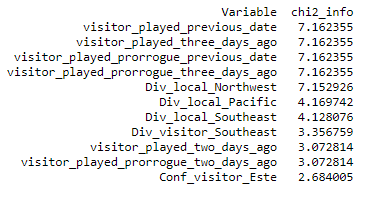
Mientras que, la información mutua (mutual information) es una cantidad que mide la relación entre dos variables aleatorias que se muestrean simultáneamente. En particular, mide cuánta información se comunica, en promedio, en una variable aleatoria sobre otra.

Un teorema importante de la teoría de la información dice que la información mutua entre dos variables es 0 si y solo si las dos variables son estadísticamente independientes.

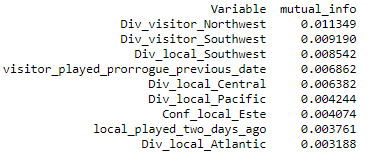
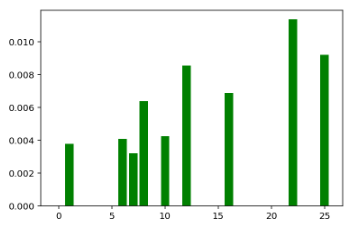
En efecto, el mismo resultado que evaluamos cuando se realiza la hipótesis nula de una prueba de chi-cuadrado y se espera rechazar cuando se realiza selección de variables categóricas, por ejemplo.

Más allá del criterio empírico del proceso estadístico, luego de evaluar los resultados de ambos enfoques, se decide utilizar las variables seleccionadas por el cálculo de la información mutua, a priori, parece que las variables seleccionadas por este método tienen mayor importancia en cuanto al criterio del deporte del baloncesto respecta.

Las variables dummy´s seleccionadas por chi-cuadrado son las siguientes:



Las variables dummy´s seleccionadas por información mutua son las siguientes:

1. Selección de Variables Continuas:

Derivado del proceso de filtrado de “Garbage in, garbage out” en el apartado de análisis exploratorio (Parte I) se obtuvieron 53 variables continuas. Al igual que el proceso de selección anterior, se utilizaron diversos métodos para descartar las variables no relevantes para predecir el resultado de un juego, entre las que destacan: Genetic Algorithms, Permutation Importance y Random Forest Feature Importance.

En entornos de aprendizaje automático podemos citar a Joaquín Amat Rodrigo para definir que los algoritmos genéticos son “métodos de optimización heurística que, entre otras aplicaciones, pueden emplearse para encontrar la combinación de variables que consigue maximizar la capacidad predictiva de un modelo”.

El primer intento de selección de variables se basó en este método, utilizando básicamente adaptaciones de códigos suministrados en una asignatura durante el Máster y aplicaciones técnicas encontradas en la web. La maximización de la capacidad predictiva del modelo se fundamentó en la medida del índice de impureza de Gini.

Según O’Relly Machine Learning Book etc etc (cita).“Dada una muestra, la impureza de Gini mide la probabilidad de una clasificación errónea si una etiqueta se elige aleatoriamente usando la distribución de probabilidad de la rama. El índice alcanza su mínimo (0.0) cuando todas las muestras de un nodo se clasifican en una sola categoría”.

El segundo proceso de selección de variables fue el Random Forest Feature Importance usando adaptaciones del código propuesto por la librería de sklearn en la web. Según la web este algoritmo es un “meta-estimador que ajusta un número de clasificadores de árbol de decisión en varias submuestras del conjunto de datos y utiliza el promedio para mejorar la precisión predictiva y controlar el sobreajuste”.

El criterio de selección de importancia de variables por defecto de la librería de sklearn es basado en la medida del índice de impureza de Gini (Gini impurity index).

La importancia de las variables basadas en impurezas se fundamenta en el criterio de “cuanto más alto el índice, más importante es la variable”. Y esta importancia se calcula como la “reducción total (normalizada) del índice de cada variable”.

Finalizado el proceso de selección anterior, se decide realizar un último proceso de selección llamado “Permutation Importance” derivado de las recomendaciones de la propia librería de sklearn por las carencias que tiene el algoritmo anterior sobre el criterio de selección basado en la medida de importancia de Gini; en la web indican que este proceso de selección puede ser “engañoso” para variables de alta cardinalidad (sobrestimación de su importancia), asimismo, demuestran cómo el algoritmo tiende a sobreestimar la importancia de variables que no tienen relación alguna con la variable objetivo.

La “importancia de permutación” de una variable se calcula de la siguiente manera:

Primero, se evalúa una métrica de línea base (baseline metric), definida por una puntuación (scoring) en un conjunto de datos (diferente a la línea base). A continuación, se permuta una columna de una variable del conjunto de validación y se evalúa nuevamente la métrica. La importancia de la permutación de cada variable se define como la diferencia entre la métrica de línea base y la métrica obtenida de permutar la columna de la variable.

Cada uno de los métodos mencionados anteriormente arrojaron resultados diferentes, sin embargo, a priori parece existir un patrón entre el número optimo de variables necesarias para maximizar la métrica con la que se estuviese evaluando cada método. Este hallazgo es importante, y es la razón por el cual se ha decidido contrastar diferentes metodologías para la selección de variables. Si bien cada método le dio más o menos importancia a algún grupo de variables en concreto, todas resultaron arrojar entre 22 y 27 variables esencialmente importantes para maximizar el desempeño del modelo.

Se decidió elegir finalmente las variables por el criterio de “importancia de permutación”; específicamente por mayor conocimiento del cálculo de la métrica de selección (comparado con el complejo proceso del proceso del Algoritmo Genérito) y porque este proceso es recomendado por la librería de sklearn por encima del proceso del Random Forest Feature Importance.

Las variables continuas más importantes seleccionadas por cada uno de los métodos seleccionados anteriormente son las siguientes:

|  |  |  |
| --- | --- | --- |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

\*Nota 1:

El proceso de selección de variables forma parte del proceso de optimización de parámetros durante la construcción y evaluación del modelo, en este contexto, no existen garantías de uso exclusivo de las variables seleccionadas en esta sección. La totalidad de estas variables se utilizan como base para la construcción del modelo, luego durante el proceso de optimización de parámetros y evaluación del modelo se determina si la inclusión (exclusión) de variables mejora o no el desempeño del modelo.

Una vez realizado el diagnostico de las curvas de aprendizaje de los modelos y de las métricas de evaluación, se encontró (derivado de muchas pruebas de ensayo y error) que la inclusión de la totalidad de las variables del conjunto de entrenamiento contribuiría a la mejora de la complejidad del modelo sin incidir en el detrimento de las métricas de evaluación.

Según el costo de oportunidad de “alto sesgo” y “baja varianza” (high bias and low variance) analizado en las curvas de aprendizaje, se prefirió incrementar la complejidad de los modelos (mayor varianza) y en efecto, reducir el sesgo.

Cuanto menos sesgado sea un modelo (magnitud del error), mayor será su capacidad para ajustar bien los datos (cuidando el riesgo del overfitting). Cuanto mayor es la capacidad del modelo para ajustarse bien a los datos, mayor es la varianza (medida de dispersión entre modelos respecto al modelo poblacional).

Finalmente, con esta decisión hemos optimizado dos modelos muy diferentes. El primero con un alto sesgo (baja varianza), y el segundo con alta varianza (bajo sesgo). Por lo que el primero tiene una mayor capacidad de generalización (replicar el comportamiento aprendido en el conjunto de entrenamiento en el conjunto de validación) y el segundo, un modelo más complejo que se ajusta al patrón observado en el conjunto de entrenamiento.

7.3 SELECCIÓN DE MODELOS

En esta sección se definen los algoritmos de aprendizaje automático que se utilizan para crear los modelos que mejor se ajustan a los datos extraídos, de esta forma tratar de encontrar el modelo con mayor capacidad predictiva sobre los resultados de los partidos de la NBA.

Sobre la división de los datos en el conjunto de entrenamiento y test realizado en la sección anterior se debe usar exclusivamente el conjunto de entrenamiento para preparar el algoritmo. Sin embargo, ¿es este conjunto de entrenamiento una muestra lo suficientemente representativa para que un modelo tenga la capacidad de aprender el patrón de datos que define el ganador de un partido? No lo sabemos, a priori se debe afirmar esta hipótesis de forma imperativa y en cualquier caso, de ser necesario, mejorar las condiciones de los datos de tal forma que lo que observe el algoritmo en el patrón de los datos sea lo más semejante a la realidad de los datos “poblacionales”.

Cuando el conjunto de datos no representa todas las clases de datos por igual, por ejemplo, el modelo podría ajustarse en exceso a la clase que está más representada en los datos que está observando (en nuestro caso como es de esperarse mayores juegos ganados por parte del local) y pasar por alto la existencia de la clase minoritaria (los juegos perdidos del local), lo que conlleva a mermar la capacidad de generalización del modelo sobre predicciones de datos no observados.

Se ha evidenciado presencia de datos desbalanceados desde el inicio del proyecto, sin embargo, esta ha disminuido a medida que hemos podido ingresar y enriquecer más datos de diferentes fuentes.

A inicios del proyecto se evidenciaban niveles de 70%-30% (sobre la variable target), y se ha podido reducir esta diferencia hasta un 60-40%.

Con esta distribución no parece tener problemas la construcción del modelo, en cierto sentido, el modelo debe tener una ligera preferencia (como lo es en la realidad) porque gane el local.

En cualquier caso, de no poderse obtener mayores datos: ¿cuál es la forma correcta de afrontar este problema?

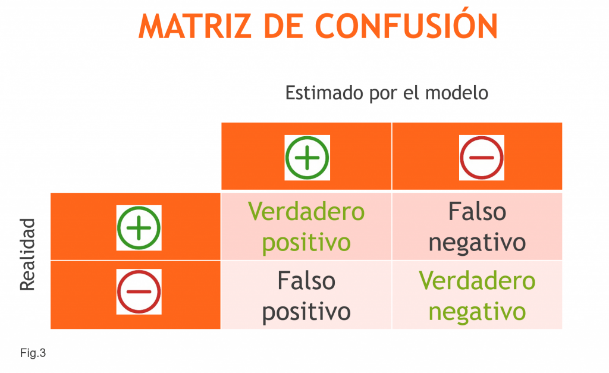
En teoría se debería realizar una operación de sobremuestreo al azar (Random Oversampling) ¿por qué? (colocar cita o viñeta que indique información del libro + links) + definición

Una vez identificado y contrastado que no existe presencia de datos desbalanceados. Nos preguntamos ¿Qué métricas se utilizan para evaluar el desempeño de un modelo? Y ¿Cómo se mide la capacidad de generalización del modelo?

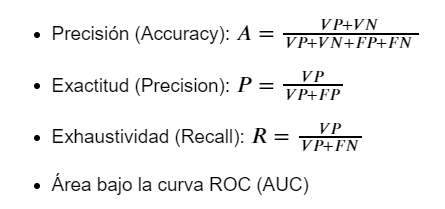
En modelos supervisados de clasificación (problemas de clasificación de clase binario), en la que los resultados se etiquetan positivos (P) o negativos (N), hay cuatro posibles resultados que se pueden obtener:

* Verdaderos Positivos (VP)
* Verdaderos Negativos (VN)
* Falsos Positivos (FP) o Error tipo I
* Falsos Negativos (FN) o Error de tipo II

Para facilitar el análisis, los resultados se pueden organizar en una matriz de confusión:



A partir de estos resultados se pueden construir diferentes métricas para evaluar la calidad de los modelos construidos, algunas de las más utilizadas son:



Cuadro 1

En scikit-learn todas estas métricas se encuentran en la librería metrics

Entre las métricas más utilizadas, usaremos la Precision (Accuracy) y Área Bajo la Curva ROC (AUC) para definir cual algoritmo de aprendizaje automático elegir entre los distintos que se pretenden examinar.

Este proceso de selección se explicará más adelante, por ahora centraremos la atención en explicar brevemente las dos métricas que utilizaremos para este proceso.

Empezando por la más sencilla, la precisión (Accuracy). A partir de ahora nos referimos a esta métrica con su nombre en inglés para evitar confusiones. Informalmente, el Accuracy es la fracción de predicciones que el modelo realizó correctamente, mientras que, formalmente se calcula como se muestra en el cuadro 1.

En aprendizaje automático es ampliamente conocido evaluar modelos a través del cálculo del área bajo la curva (AUC por sus siglas en inglés). AUC significa "área bajo la curva ROC". Esto significa que el AUC mide toda el área bidimensional por debajo de la curva ROC desde los puntos (0,0) hasta (1,1).

El área bajo la curva es una métrica de evaluación de modelos que considera todos los umbrales de clasificación posibles. Entendiéndose umbrales de clasificación como; criterio de valor escalar que se aplica a la predicción de un modelo para separar la clase positiva de la negativa (tal como hemos observado anteriormente en la matriz de confusión).

Matemáticamente la curva ROC es la tasa de verdaderos positivos frente a la tasa de falsos positivos en diferentes umbrales de clasificación.



Una de las bondades de utilizar el área bajo la curva ROC es que está estrechamente relacionada con el coeficiente de Gini que en muchos casos se utiliza como una medida alternativa. Si recordamos en el apartado anterior se ha comentado que el índice de Gini en la librería de sklearn es la métrica utilizada por defecto.

Comúnmente se conoce el índice de Gini como el doble del área entre la curva ROC y la diagonal (línea de 45 grados que pasa por el origen).



Finalizado esta breve explicación de las métricas a considerar para evaluar el modelo, se continua con el proceso de selección de estos.

Siguiendo la extensa literatura sobre los modelos de aprendizaje automático que plantean problemas de clasificación de clase binario para la predicción del ganador de un juego del baloncesto podemos darnos cuenta de que muchas de las conclusiones identifican que los modelos basados en estimaciones lineales son los que mejores resultados arrojan.

Por citar algunos, por ejemplo, "Prediction of NBA games based on Machine Learning Methods" [Torres 2013] en base a sus resultados utilizando regresiones lineales concluye que en comparación con otros estudios pilares del conocimiento en el área como lo es "NBA Oracle" [Beckler et. al. 2008] han podido lograr resultados muy satisfactorios y cercanos a estos, entorno a un accuracy del 60% y 70%, mientras que "NBA Oracle" [Beckler et. al. 2008] entre todos sus modelos complejos de aprendizaje automático el mejor resultado ha sido un 73% de accuracy.

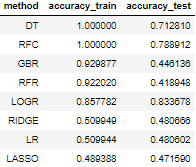
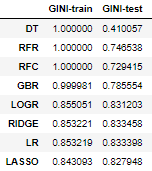
En "NBA Oracle" han podido experimentar con modelos de Logistic Regression, Support Vector Machines, Logistic Regression, y Artificial Neural Networks. Concluyendo brevemente al igual que [Torres 2013] que los modelos con mejores resultados en el baloncesto son los más sencillos, es decir, las regresiones lineales.

Finalmente, es importante comentar el estudio de “Nba Game Prediction Based On Historical Data And Injuries” [Dionny et. al. 2016] en el que destaca entre todos sus modelos de aprendizaje automático la regresión lineal con un accuracy del 68%.

Más allá de seguir los descubrimientos de la literatura de aprendizaje automático aplicado al entorno del deporte, este proyecto no pretende sesgar el enfoque al descubrimiento que han realizado otros, es por esto que desde una perspectiva neutra se han evaluado 8 algoritmos en su estructura de parámetros “más sencilla” con las métricas comentadas anteriormente para decidir al menos 2 para desarrollar con mayor profundidad.

Utilizando un conjunto de funciones, procesos y conocimientos adquiridos durante el Máster se ha desarrollado un proceso automático de selección de modelo basado principalmente en la comparación de métricas como el accuracy y el índice de Gini en un resumido dataframe (para mayor detalle revisar el notebook del modelo)

En resumen, los 8 modelos seleccionados con parámetros por defecto (según la sklearn) para la comparación de métricas son los siguientes:



Donde:

* DT: Decision Tree
* RFR: Random Forest Regression
* RFC: Random Forest Classifier
* GBR: Gradient Boosting Regression
* LOGR: Logistic Regression
* Ridge
* LR: Linear Regression
* Lasso

Tal como se ha comentado en el inicio de este apartado (Parte III), el objetivo es determinar el modelo de aprendizaje automático que mejor se ajusta a los datos extraídos.

En este contexto, solo se deben considerar los resultados en entrenamiento e ignorar así los resultados en test; este último nos sirve de referencia para determinar el porcentaje de caída de las métricas de evaluación o de manera informal determinar la capacidad de generalización del modelo (sabiendo que los parámetros de todos los modelos vienen por defecto y no se ha realizado una selección de variables importantes).

Sobre el dataframe de resultados de Gini; se observan muy buenos resultados en general, entre otras cosas, los modelos de árboles de decisión sobre ajustados (comportamiento usual en este tipo de algoritmos complejos) y los modelos lineales con mayor capacidad de generalización. (según el criterio del porcentaje de caída de las métricas en el conjunto de test).

Sobre el dataframe de resultados de Accuracy; destacan los modelos de árboles de decisión con elevados resultados en entrenamiento y elevada perdida de capacidad de generalización. De igual forma destaca el comportamiento del Logistic Regression entre los modelos lineales en cuanto a la capacidad de generalización.

Posterior al análisis, se decide elegir el Random Forest Classifier y el Logistic Regression para el desarrollo de los modelos, proceso que conlleva tanto la optimización de hiperparámetros (hyperparameter optimization) como la selección de variables óptimas para cada uno.

Si existen dudas sobre el proceso de selección de variables optimas leer Nota 1. Aun cuando en el apartado anterior (Parte II) se han determinado las variables más importantes, estas se han utilizado como variables “base” para el proceso de selección del modelo (Parte III).

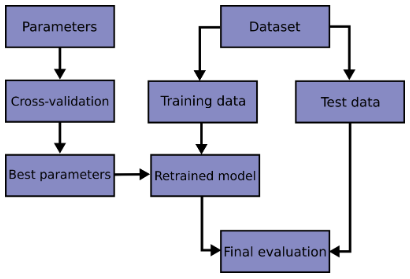
7.4 CONSTRUCCIÓN DE MODELOS

“Aprender los parámetros de una función de predicción y probarlo con los mismos datos es un error metodológico: un modelo que simplemente repita las etiquetas de las muestras que acaba de ver tendría una puntuación perfecta pero no podría predecir nada útil todavía en datos que no ha visto”.

Aun cuando se pruebe el modelo en datos que no ha visto (conjunto de test) y se observen buenos resultados (como se comentó en la sección anterior en el proceso de selección de modelos). ¿Son estos resultados lo suficientemente robustos? O ¿es posible que un componente aleatorio influya en los resultados? Dependiendo cómo dividamos los datos en estos dos conjuntos (entrenamiento y test) tendremos una estimación diferente del error de generalización. En unos casos pensaremos que el modelo generaliza mejor y en otros peor.

Para incrementar la robustez de los resultados (minimizando el componente aleatorio) y evitar el sobreajuste de los modelos existe la validación cruzada (cross-validation). Una práctica común de su uso es para determinar los parámetros óptimos de un modelo.

Aquí hay un diagrama de flujo de trabajo típico de validación cruzada en el entrenamiento de modelos (extraído directamente de la web de sklearn).



Los mejores parámetros pueden determinarse mediante diversas técnicas, en sklearn existen dos enfoques: RandomizedSearchCV y Grid Search.

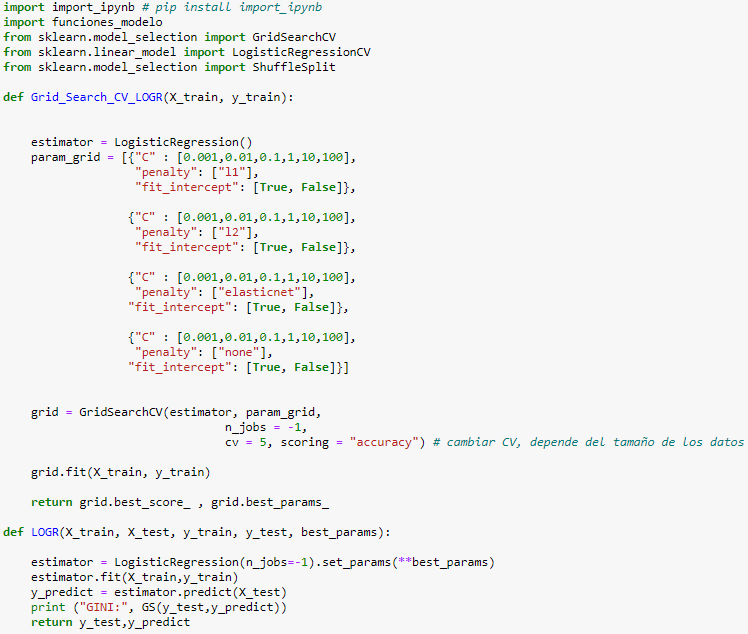
En este estudio se ha decidido usar la función de Grid Search aun cuando su costo computacional y tiempo es más elevado, su desempeño en la búsqueda es mejor; el algoritmo utiliza exhaustivamente todas las combinaciones posibles de los parámetros especificados.

Para el Random Forest Classifier el tiempo de ejecución del proceso fue aproximadamente 45 minutos, mientras que, el Logistic Regression fue un total de 5 segundos. Ambos en un equipo promedio con una memoria RAM de 8 gigabytes.

En resumen, se puede entender con la comparación de ambos tiempos de ejecución la magnitud de la diferencia de complejidad de ambos algoritmos, aun cuando en ambos procesos existen diferencias en parámetros y el número de iteraciones necesarias para encontrar el parámetro óptimo.

A continuación, se pueden observar las funciones creadas para ambos procesos. Posteriormente mediante loops los resultados fueron arrojados las veces que se fuese necesaria, para ambos casos se obtuvieron 5 resultados. (Revisar notebook para mayor detalle).





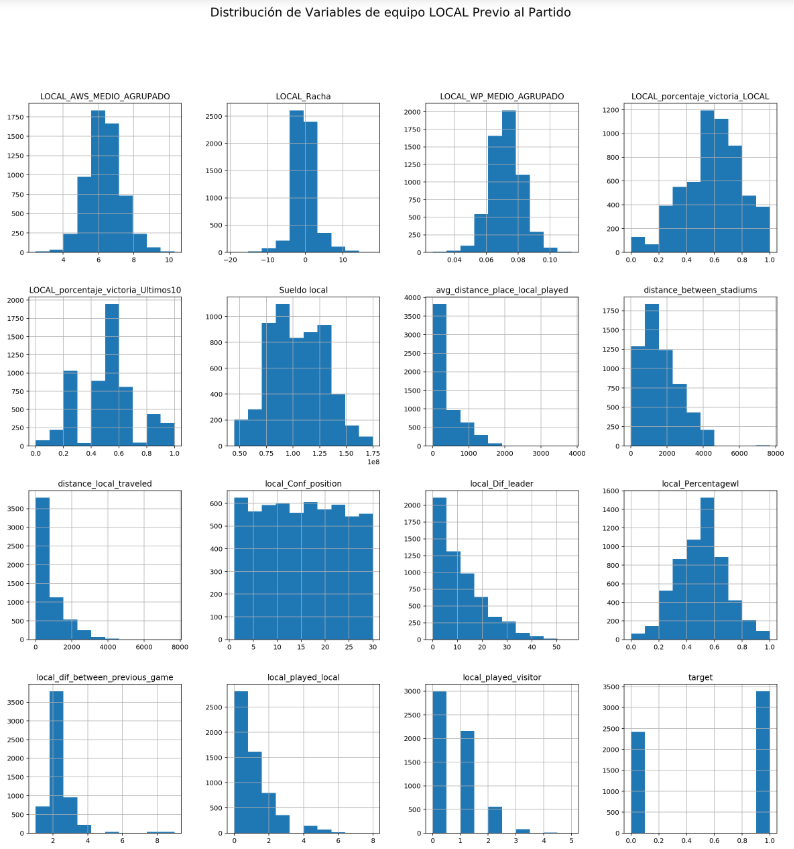
En las funciones anteriores no se presentaron la totalidad de parámetros para cada modelo, a medida que se introducen mayores parámetros incrementa el tiempo de ejecución del código, por lo que se han establecido solo los prioritarios para conocer los umbrales de los parámetros numéricos, por ejemplo, o al menos determinar un patrón de criterios que mayor peso tienen respecto a las métricas de evaluación.

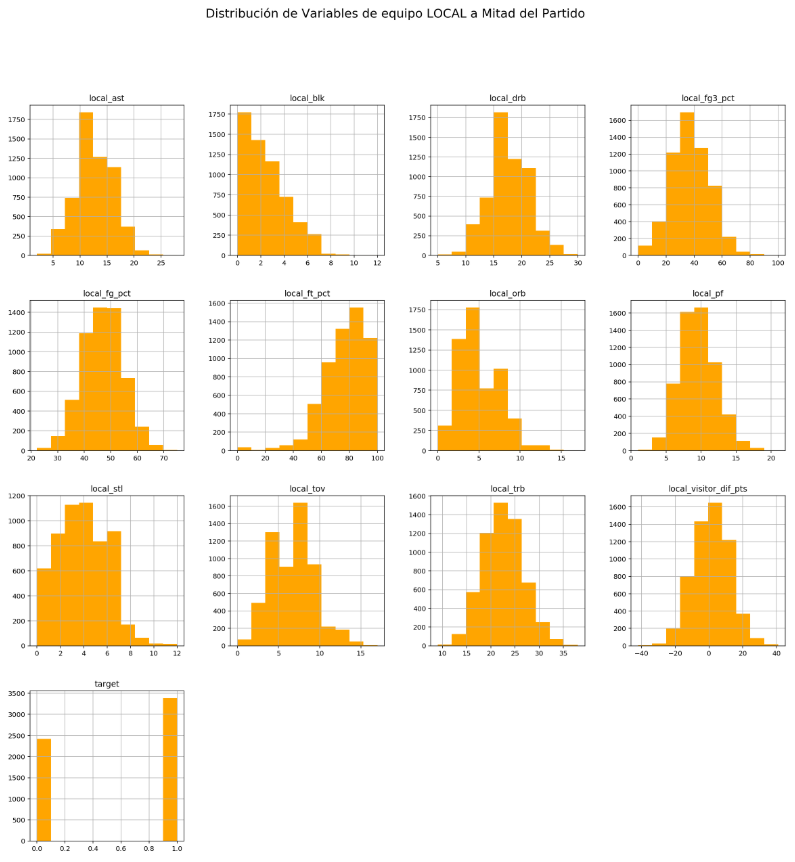
Asimismo, es importante destacar que el proceso de optimización de parámetros no finaliza con los procesos establecidos anteriormente, se debe recordar que estos parámetros solo garantizan resultados robustos en cuanto al conjunto de datos de entrenamiento, por lo que se deben probar en el conjunto de test. En muchas ocasiones la combinación de parámetros se debe ajustar mediante ensayo y error de forma manual en el conjunto de datos de test.

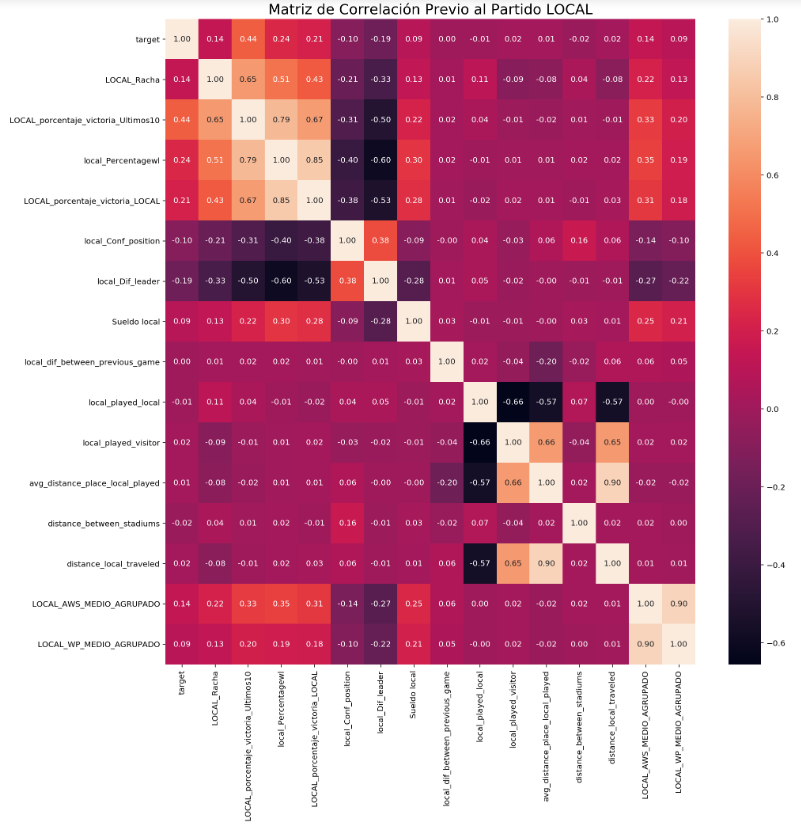
En el siguiente apartado se proporciona más información acerca de la función y de la importancia de cada parámetro, así como el ajuste final de cada uno para garantizar los resultados mas cercanos posibles a los encontrados durante la validación cruzada.

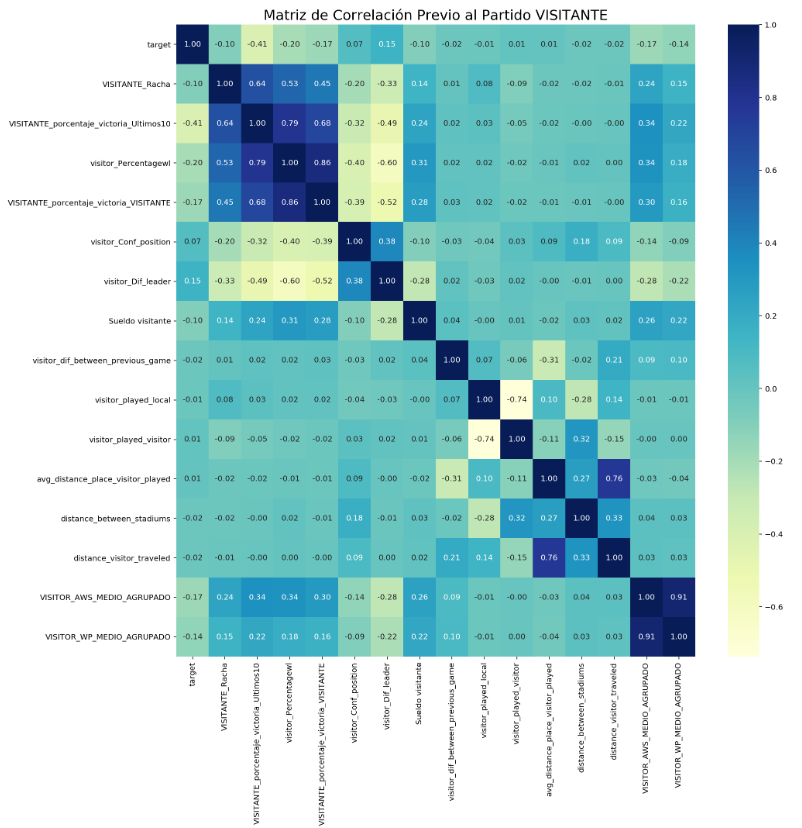
7.5 EVALUACIÓN

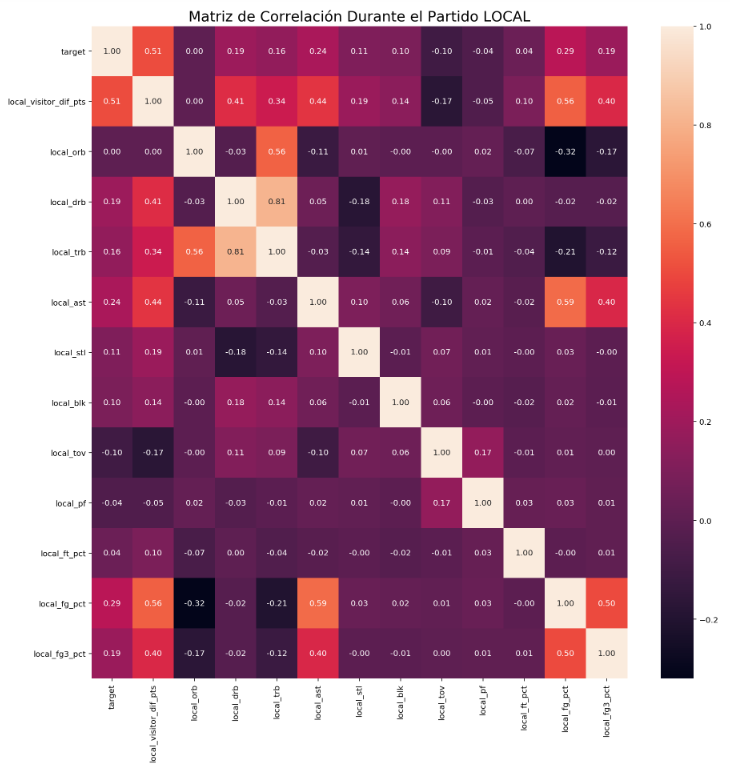
7.6 ANEXO

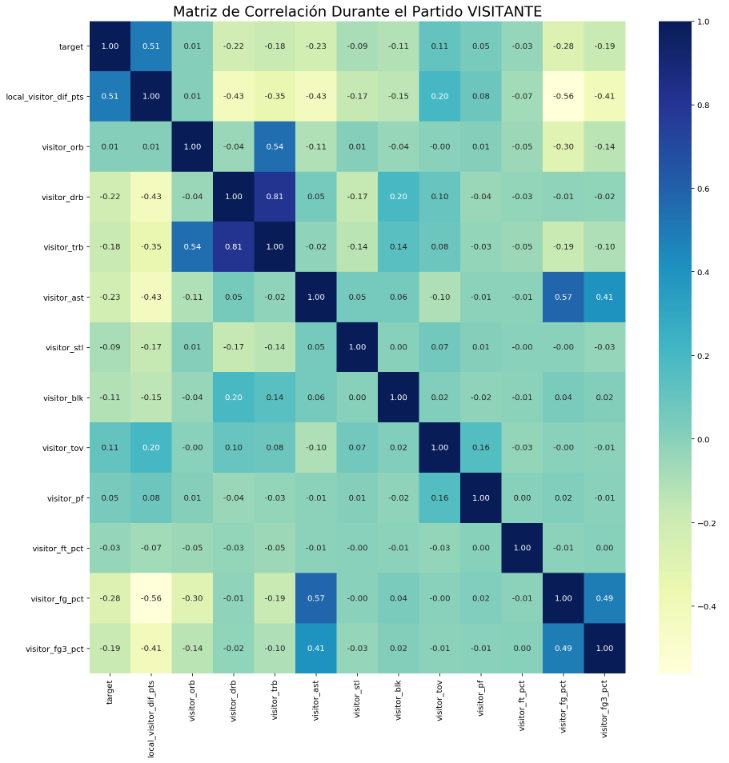


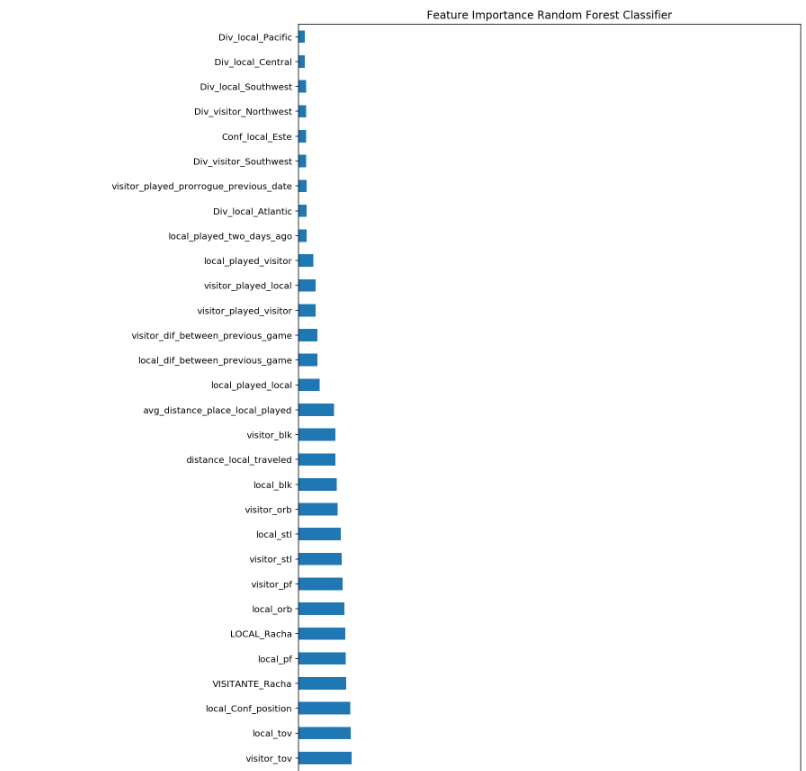
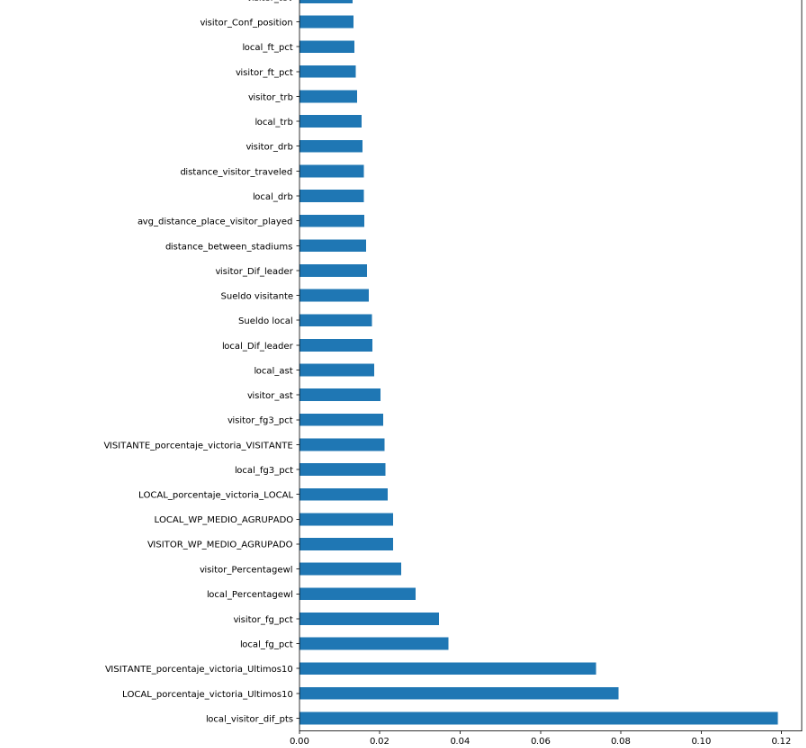


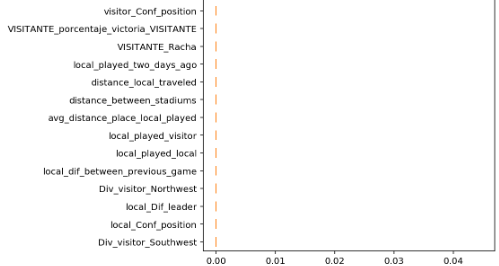








# CONCLUSIONES

# REFERENCIAS BIBLIOGRÁFICAS / WEBGRAFÍA

Renato Amorim Torres. *Prediction of NBA games based on Machine Learning Methods****.***(2013) <http://homepages.cae.wisc.edu/~ece539/fall13/project/AmorimTorres_rpt.pdf>.

Matthew Beckler, Hongfei Wang. NBA Oracle (<http://www.mbeckler.org/coursework/2008-2009/10701_report.pdf> )

SERRANO y BAKER, Basketball Analytics, 2013, Advanced Metrics, Lake St. Louis

García, J. Ibáñez, S. J., Feu, S. (2010). Estadísticas de juego que discriminan las selecciones participantes en los Campeonatos del Mundo de 2006 de Baloncesto, en función del nivel y género de los equipos. Revista Kronos, IX, 17, 57-63.

Sampaio, J., Fraga, F., Silva, T. (2004). La evolución de las estadísticas de los tiros libres en partidos de baloncesto de formación. III Congreso de la Asociación Española de Ciencias del Deporte. Valencia, 11-13 de marzo.

Sampaio, J., Leite, N. (2006). ¿Por qué ganaron o perdieron los partidos de baloncesto los equipos que participaron en el Eurobasket 2005? Revista Kronos, nº9, pp.67-73.

Sampaio, J., Lorenzo, A. Gómez Ruano, M. A., Matalarranha, J., Ibáñez, S. I., Ortega, E. (2009).