FACULTAD DE ESTUDIOS ESTADÍSTICOS

MÁSTER EN MINERÍA DE DATOS E INTELIGENCIA DE NEGOCIOS

Curso 2023/2024

Trabajo de Fin de Máster

Predicción de situaciones de arbitraje en el mercado de apuestas deportivas 1X2

***Alumno*: Arturo Sancho Spottorno**

***Tutor*: Javier Castro**

Septiembre de 2024

UNIVERSIDAD COMPLUTENSE
MADRID

Índice

Índice de gráficos

Índice de imágenes

Índice de tablas

Índice de anexos

Introducción

En el mundo de las apuestas deportivas se tienen en cuenta numerosos tipos de deportes y de clases de apuestas. Una de las modalidades más populares de estas últimas, es el "resultado a tiempo completo" en el fútbol, también conocido como 1X2, en la cual el jugador intenta predecir si gana el equipo local, el visitante o si habrá un empate. Las casas de apuestas ofrecen ratios a los distintos resultados que marcan cuánto dinero por euro invertido se llevará el jugador si acierta. En este mercado, en el que influyen muchos factores, se puede dar, de manera singular, una situación de que se denomina "de arbitraje". Ésta se considera una anomalía en la cual una combinación de apuestas a los 3 resultados gana dinero independientemente de quién gane el partido. Por ejemplo, si los ratios a local, empate y visitante son 3.2, 3.3 y 3.5, respectivamente, una apuesta de 1€ a cada resultado obtendrá como beneficio neto 20 céntimos si gana el equipo local, 30 céntimos si es un empate y 50 céntimos si gana el equipo visitante. Aunque el arbitraje se puede dar en prácticamente todos las clases de apuestas este trabajo se centra, por su popularidad, la cantidad y disponibilidad de los datos, exclusivamente en este tipo de apuesta, es decir las apuestas de resultado a tiempo completo (1X2).

Dentro de una misma casa de apuestas (CA), es casi imposible que se presente una oportunidad de arbitraje ya que la propia casa tiene una situación privilegiada para "apostar contra sí misma" , y así, corregir esta anomalía. Por esto es más fructífero buscar el arbitraje en combinaciones de ratios entre distintas casas (Franck, E., Nuesch, S., & Verbeek, E, 2009). Las cuotas que se ofrecen dependen principalmente de cómo hayan apostado sus clientes. Desde el punto de vista de la CA son sus clientes quienes apuestan entre ellos, con la casa como mediadora, y los ratios dependen de la relación entre el dinero total apostado a cada resultado, menos una comisión del bote total que se lleva la casa. Las diferencias entre los ratios de las casas de apuestas es muy habitual, de hecho, existen numerosas páginas web de comparadores de apuestas que observan el mercado en busca de los mejores ratios, entre una selección de corredores de apuestas para cada partido. Estas diferencias se deben, entre otras cosas, a sesgos de la clientela principal de la casa de apuestas. Por ejemplo, una casa con publico mayoritariamente catalán no tendrá los mismos ratios que una con un público madrileño, para un partido entre el Real Madrid y el FC Barcelona.

Las apuestas arbitradas no están intrínsecamente perseguidas por las casas de apuestas ya que éstas no pierden beneficios, es más, obtienen más ingresos porque se llevan una comisión de todo el dinero apostado. Sin embargo, sí que tienen una posición privilegiada para realizar este tipo de "inversión segura" y así alterar sus cuotas, pero al hacerlo se exponen a un riesgo si se alejan mucho de los ratios que su mercado de clientes genera. Además, no todas las casas tienen los recursos, el tamaño o la motivación para monitorizar el mercado y por lo tanto ignoran a los "árbitros" y se conforman con ganar un porcentaje de su dinero apostado.

Aun así, estas oportunidades son bastante raras y efímeras ya que hay una pequeña industria dedicada a detectarlas y aprovecharlas, desde fanáticos de las apuestas a las propias casas y proveedores o comparadores de ratios online. En esta industria se han

desarrollado numerosas técnicas para realizar el arbitraje, como la monitorización exhaustiva del mercado, en la cual se mira un grandísimo volumen de partidos y cuotas probando combinaciones entre ellas. Este método suele darse por las propias CA que disponen de esta información por un coste muy bajo. También existen estrategias que incorporan un pequeño nivel de riesgo como el arbitraje Inter temporal, que plantea estimar máximos de las cuotas en el mercado para casar apuestas en distintos puntos temporales (Torres–Cabrera, G. P. S., & González, C. Q,2018). La literatura en este campo es muy escasa y suele estar desactualizada ya que la gente que se dedica a esto profesionalmente no tiene ningún interés en hacer pública su metodología para evitar poder dar una ventaja a su competencia.

Una característica que comparten la mayoría de las metodologías es que requieren datos de muchos partidos con una latencia muy alta, ya que sólo se aprovecha del arbitraje el primero en detectarlo. Esto hace que el propio hecho de mirar el mercado tenga un coste elevado, pues no es igual de caro seguir 100.000 partidos actualizado la información cada segundo que mirar solo 1.000 partidos y refrescar los ratios cada 5 minutos. Estos costes socaban los beneficios obtenidos y puede llevar a muchos árbitros a determinar que no les sale rentable realizar dicho arbitraje.

Este trabajo de fin de máster (TFM) propone un modelo capaz de predecir en qué partidos se dará una situación de arbitraje en la siguiente hora, con información de latencia horaria de las cuotas, y agregados de la información de las 6 horas anteriores. De esta forma se puede monitorizar lentamente muchos partidos y casas, ya que la información con latencia horaria es muy barata. Durante este proceso, una vez que se prediga que pueda haber una situación de arbitraje, se realizará una monitorización intensiva de ese partido durante la hora siguiente.

Con esta metodología no se asume ningún riesgo de perder dinero en una apuesta, como sucede por ejemplo al hacerse arbitraje en el tiempo, y se reduce mucho el coste de mirar el mercado haciendo más asequible el arbitraje. Se pretende introducir la calidad de los modelos predictivos como una ventaja competitiva alternativa, y más eficiente, al modelo actual de monitorización extensiva e intensiva del mercado.

Objetivos

El objetivo de este trabajo de fin de master es estudiar y comprender el mercado de las apuestas deportivas con el fin de entender su funcionamiento y su evolución en las horas anteriores a los partidos. Con esta información se pretende poder realizar predicciones sobre su estado, y ayudar a posibles interesados en introducirse en el mercado del arbitraje

Como objetivos secundarios también se estudiará la presencia del arbitraje mediante combinaciones de ratios de casa de apuestas y la evolución de la proporción de partidos con arbitraje a lo largo del período de observación. Además, se verá cómo se reparten todas las situaciones de arbitraje a lo largo del tiempo con el fin de determinar que no hay un punto de corte óptimo a partir del cual se deba empezar a predecir. Con ello se concluirá la viabilidad de un modelo general capaz de predecir en todo el intervalo temporal, como se propone el objetivo principal.

Se pretende realizar un estudio descriptivo que evalúe el impacto de valores atípicos en las situaciones de arbitraje, proponer un sistema de muestreo y selección de grupos de validación cruzada de forma aleatorias basados en la selección de partidos, y no de filas que mantenga la integridad de los conjuntos de datos en el modelado validación y testeo. También se busca la eliminación o reducción de la multicolinealidad presente en las variables agregadas mediante la selección de variables para modelos con base lineal.

Se intentará obtener un algoritmo predictivo que sea general, de forma que pueda predecir a partir de cualquier hora, con un período inicial de observación. Comparar la eficacia de distintas arquitecturas de modelos de *machine learning* para adaptarse a la forma de nuestros datos. Esto incluye la proposición de variables que agregan la información del mercado de apuestas y evaluación de su eficacia mediante la selección de variables por distintos algoritmos.

La metodología de detección de arbitraje propuesta en este trabajo también tiene la intención de servir como modelo tanto a casas de apuestas (que pretenden evitar encontrarse en una situación de arbitraje con sus competidores o aprovecharse de ellas) como a inversores autónomos (que quieran introducirse en el mercado de la detección y aprovechamiento del arbitraje) de una forma poco costosa y asequible en términos de computación. además de ofrecer pautas para la realización de la metodología propuesta en caso de disponer de una mayor capacidad de computación.

Como ultimo objetivo secundario, se estudiará el efecto de las variables dependientes sobre la variable objetivo tratando de mejorar la explicabilidad del modelo final seleccionado en caso de que sea de caja negra.

Metodología empleada

Para este trabajo se seguirá la metodología SEMMA (Muestreo, Exploración, Manipulación, Modelado, y Valoración). No será necesario realizar la sección de muestreo puesto que los datos ya son una muestra realizada por Kaggle. Se realizará un análisis exploratorio preliminar para ver la densidad de información de la base de datos en bruto lo que lleva a una manipulación de los datos, en la cual la información en la base de datos original se agregará en distintas variables que se usaran en la exploración y más adelante en el modelado. Posteriormente, para cada modelo predictivo se hará una validación mediante un conjunto de datos de validación no usado en el modelado, y se seleccionará el que mejor prediga el arbitraje.

La variable objetivo "hay_arbitra" se creará a partir de los datos originales siguiendo la formula del arbitraje con los ratios más altos para cada resultado, *home*, *draw*, *away (H, D, A)* entre las casas para cada momento temporal.

$$Arbitrage\ si:\ 1 > \frac{1}{ratio\ H} + \frac{1}{ratio\ D} + \frac{1}{ratio\ A}$$

Una vez obtenida la base datos intermedia se realiza un análisis preliminar que evaluará las hipótesis básicas del modelo, como la ausencia de punto de corte optimo o la prevalencia de proporción de suceso de nuestra variable objetivo. Después se volverán a modificar los datos agregando la información para realizar el análisis exploratorio en el cual estudian y tratan los atípicos y los datos faltantes y se evaluará la correlación entre variables. Como último paso antes de la modalización se realiza una división de los datos en los conjuntos *Train, Validate*, *Test* con un 20%,40%,40% de los datos respectivamente. No se sigue una división de los datos más convencional por el gran tamaño de la base de datos resultante y la limitada capacidad de computación de la que se dispone.

Una vez obtenida la base datos final se realiza una división de los datos en los conjuntos *Train, Test y Valídate,* con un 20%, 40%, 40% de los datos respectivamente. Debido a que disponemos de una base de datos muy grande, y la limitada capacidad de computación de la que se dispone, no se podrá hacer una partición *train-test* habitual. Por el teorema central del límite se considera que con el número de filas que ocupan aproximadamente el 20% se tendrá una muestra suficientemente grande para poderse hacer una idea de la eficacia de los modelos manteniendo los tiempos de ejecución razonables, aunque esto solo abarca un fracción de los datos estos parecen suficientes para tener resultados representativos o por lo menos una idea informada de si esta línea metodológica puede dar resultados. En un trabajo real, en el que se disponga de más poder computación, sí sería recomendable utilizar una partición habitual de los datos de 70%,20%,10%.

La selección de individuos para estos conjuntos de datos será aleatoria, estratificada por las variables de componente temporal, para garantizar una representatividad y homogeneidad de la muestra en los 3 conjuntos.

En un trabajo real la variable a maximizar sería el beneficio. Mientras que conseguir ratios con una hora de *delay* (retraso) tiene un coste casi despreciable, mirar un partido segundo a segundo durante una hora, en la que se espera arbitraje, sí tiene un coste más elevado (C) y en los partidos en los que se aproveche el arbitraje se tendrá un retorno esperado de (R). Se debería encontrar una probabilidad mínima de que el partido presente una situación de arbitraje, modificando los pesos de las filas con arbitraje o moviendo el punto de corte del modelo predictivo de forma que se maximizase la siguiente fórmula:

$$Beneficio = R * \text{verdaderos positivos} - \text{C} * (\text{verdaderos positivos} + \text{falsos positivos})$$

Esta fórmula implica que no todos los tipos de errores son iguales. Por ejemplo, si quien pretende hacer el arbitraje fuese la propia empresa que vende los ratios a tiempo real se espera un C muchísimo menor que R, los falsos negativos serán mucho peores que los falsos positivos ya que se perderá una cantidad de dinero potencial alta; mientras que si el que realiza el arbitraje es un autónomo o una empresa independiente pequeña y tiene un C más alto, o un R más bajo por falta de fondos para invertir, entonces este querrá un modelo con muchos menos falsos positivos, independientemente del dinero potencial que dejen de ganar. En ambos casos se prepararían modelos con la métrica a optimizar de *accuracy* modificado, es decir, una combinación con pesos propios para la sensibilidad y la especificidad. Este trabajo plantea una metodología general que pueda servir en ambos casos y por lo tanto desconoce qué situación es la correcta. Frente a esta incertidumbre se puede considerar dar un peso idéntico a todos las ponderaciones (o puntos de corte) y tomar como métrica de calidad el área bajo la curva ROC. Esto, aunque daría una buena idea de la eficacia de los modelos en todos los puntos de corte, no sería representativo de un caso real ya que en cualquier caso se va a determinar un punto de corte concreto o unos pesos *a priori*. Por este motivo se usará el *accuracy* sin ponderar como métrica de bondad de ajuste (punto de corte de 0.5) como ejemplo genérico, ya que cualquier otro punto de corte seria puramente especulativo.

Como paso previo a la modelización, es posible que las variables que se utilicen presenten multicolinealidad. Para mitigar o eliminar el daño que puede causar esto a los determinados modelos, se va a realizar una selección de variables para la regresión logística mediante los siguientes algoritmos:

- *Stepwise* maximizando el AIC (Criterio de la información de Akaike).
- *Stepwise* maximizando el AIC repetido mediante CV (Validación Cruzada).
- *Stepwise* maximizando el BIC (Criterio de la información Bayesiano).
- *Stepwise* maximizando el BIC repetido mediante CV.
- *Boruta* (Guerrero, J. A, s/f).
- MXM (Tsagris, M., & Tsamardinos, I, 2018).
- Ses (Hyndman, R. J., Koehler, A., Ord, J. K., & Snyder, R. D, 2008).

Las variables seleccionadas por los distintos algoritmos se compran mediante validación cruzada de una regresión logística. Se utilizarán las variables que produzcan los mejores resultados para los modelos que por su arquitectura no lidian bien con la multicolinealidad, como son la regresión logística, Redes Neuronales o el *Suport Vector Machine*.

El siguiente paso es la modalización en la cual probaremos 8 algoritmos predictivos para clasificación binaria. Estos algoritmos son los siguientes:

Regresión Logística: La regresión logista es un método de predicción basado en la regresión lineal general (Chen, D.-G., & Chen, J. K.,2021). Este algoritmo es para la clasificación binaria. Como la mayoría de las modelos lineales, obtiene una fórmula concreta y determinada. Utiliza el método de máxima verosimilitud para estimar unos parámetros (B_i) que determinan el impacto de las variables independientes en la predicción. La característica particular de esta regresión es que toma valores entre 0 y 1 indicando la probabilidad de que se dé el evento en la variable respuesta utilizando la siguiente fórmula:

$$P(y = 1|x_1, x_2 \dots x_n) = \frac{1}{1 + e^{B_0 + \sum B_i * x_i}}$$

Este modelo es muy sencillo y fácil de interpretar además de no sufrir sobreajuste, si se realiza un cribado de variables poco útiles. Sin embargo, su capacidad predictiva es limitada ya que le es difícil detectar relaciones no lineales entre las variables o interacciones de estas. Se usará este modelo en parte para tener una base con la que comprar modelos más complejos de machine learning y en parte para hacernos una idea de la magnitud y dirección del impacto de las variables independientes sobre la dependiente.

Redes Neuronales: Las Redes Neuronales son un algoritmo predictivo compuesto por capas de nodos que se relacionan entre ellas mediante funciones lineales muy complejas que se llaman de "caja negra". Esta estructura se va entrenando mediante iteraciones en las cuales se modifican las funciones de cada nodo para mejorar el poder predictivo de la red. Esto permite que esta se ajustarse a la forma de los datos con los que entrenan y así hacer predicciones (Helias, M., & Dahmen, D, 2020).

La arquitectura de estos modelos depende de una serie de hiperparámetros. El número de capas ocultas que ajustan la complejidad y se ven limitadas tanto por la cantidad de datos que se disponga, como por la capacidad de computación, ya que la complejidad de los cálculos escala muy rápidamente con la cantidad de capas. El número de nodos de la red determina la flexibilidad de esta, y permite que se ajuste mejor a los datos. Aunque este número de nodos también se ve limitado por la cantidad de datos que se disponga y por el riesgo al sobreajuste. El número de iteraciones marca cuantas veces vamos a repetir el modelo provocando ligeros cambios para que mejore. Y el *decay* indica cuanto evoluciona la red en cada iteración.

Este modelo tiene muchas ventajas frente a modelos de regresión tradicionales, pues detectan con mucha precisión relaciones no lineales entre las variables del modelo y la variable respuesta. Además, son muy generales, es decir, se adaptan bien a todo tipo de datos y son especialmente útiles con datos complejos en los que puede haber un

componente temporal o muchas variables categóricas. Aunque, como dato negativo de este modelo, en problemas de naturaleza lineal no ofrece muchas ventajas en cuanto a capacidad predictiva frente a la logística. Así como presenta muy baja explicabilidad.

Diagrama de Árbol: es una técnica de predicción basada en la creación de reglas, llamadas nodos, que separan los datos en grupos más homogéneos, respecto de la variable objetivo, que el conjunto de datos input del nodo, y hojas, los grupos finales con una gran homogeneidad y que darán la predicción final (Kingsford, C., & Salzberg, S. L, 2008). Un diagrama de árbol de clasificación binaria utiliza una métrica, siendo la más común el índice de Gini, para calcular la proporción de respuestas positivas frente a negativas hay en los datos. Posteriormente busca, entre los niveles de las variables categóricas y puntos de corte de variables numéricas, la regla que divida los datos en dos grupos lo más homogéneos posibles. De esta forma una de las hojas hijas del nodo tendrá una mayor proporción de respuestas positivas que la hoja padre, y otra tendrá una menor. Esta operación se va repitiendo, separando mediante reglas los datos, hasta que nos quedan unos grupos muy pequeños y homogéneos. Finalmente se clasificará para todos los individuos en hojas donde la proporción de respuesta positiva sea muy alta como unos y los que tengan una proporción de respuesta positiva baja como 0. Considerando la probabilidad de pertenecer a la categoría suceso de una observación la proporción de sucesos en la hoja en la que se encuentre. Como ejemplo de diagrama de árbol, se muestra la siguiente figura (imagen 1).

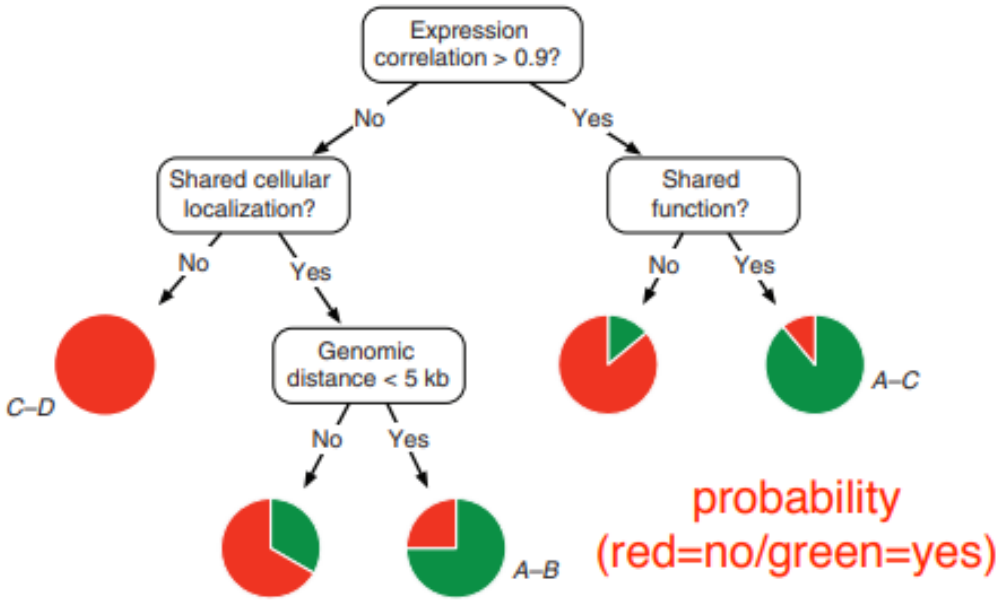


Imagen 1. Árbol de decisión como ejemplo hipotético de (Kingsford, C., & Salzberg, S. L, 2008).

Esta metodología tiene muchas ventajas que ninguna otra arquitectura que no derive de esta presenta. Es muy robusta frente a la multicolinealidad de las variables, pues, aunque se introdujesen variables muy correlacionadas o incluso duplicadas, el modelo no perdería poder predictivo ya que solo usaría una de ellas, una vez. También es muy robusto frente a valores atípicos. Ya que, al considerar puntos de corte, si un individuo tiene un valor atípico, este seguirá siendo solo un valor y su peso sobre que punto de corte es mínimo y no afectara de forma proporcional al tamaño de su atípico por lo que no dañara el modelo. Otra gran ventaja de este modelo es su explicabilidad, pues estos

diagramas, sobre todo si son pequeños, son muy interpretables y se puede visualizar el efecto de cada variable fácilmente.

Por otro lado, presenta algunos inconvenientes. Los árboles son poco estables, en el sentido de que dependen mucho de los datos que se utilicen, y alteraciones muy pequeñas de esos datos, o incluso submuestreo de estos, dan árboles totalmente distintos, aunque con resultados parecidos. También tiene una capacidad predictiva limitada, dando resultados muchas veces peores que una regresión. Otro defecto que tienen estos modelos es que las variables que segregan peor los datos tienen menos oportunidades de aportar información ya que se suelen ver opacadas por variables mejores que pueden repetirse.

Para diseñar un diagrama de árboles hay varios hiperparámetros que ajustar. El tamaño mínimo de hoja es fundamental para evitar un sobreajuste, ya que, si este es demasiado pequeño, el árbol se ajustará demasiado a los datos de entrenamiento y cada observación tendrá prácticamente su propia hoja, pero si este es demasiado grande, el modelo perderá poder predictivo. También se puede ajustar el Parámetro de Complejidad (CP). Este hyperparámetro elimina los nodos que no aumenten el poder predictivo del modelo general (el R^2) en al menos el valor del CP. A esto se le llama podar el árbol.

***Random Forest*:** estos modelos son derivados de los árboles de decisión y minimizan el principal problema que presentan estos, que es la gran variabilidad del modelo frente a alteraciones de los datos de entrenamiento. Para esto un modelo *Random Forest* (RF) realiza múltiples diagramas de árbol utilizando técnicas de remuestreo en los datos de entrenamiento. El modelo final consistirá en la media de todos los diferentes arboles creados (Genuer, R., & Poggi, J.-M, 2020). Además, los modelos Random Forest no consideran todas las variables para todos los nodos, sino que se puede determinar el hiperparámetro *mtry* que limita cuántas variables, seleccionadas aleatoriamente de las disponibles, se considera candidata para crear un nodo. De esta forma se extrae información de las variables con menos poder predictivo. Los modelos *Bagging* son un caso particular de los *Random Forest* en el que sí se consideran candidatas a todas las variables para todos los nodos con remuestreo. Esto puede dar buenos resultados si todas las variables son más o menos igual de importantes, aunque suelen dar peores resultados que RF.

Los resultados de estos modelos también dependen de otros hiperparámetros, como son el número de árboles, aunque este no se optimiza para balancear poder predictivo con sobreajuste. Dado que los RF son un conjunto de árboles teóricamente parecidos, ocurre que, a partir de cierto número de árboles, la métrica de validación a optimizar converge y, por lo tanto, no es necesario seguir calculando más árboles. También se debe optimizar el número de variables candidatas (mtry). Finalmente, se debe optimizar el tamaño mínimo de hoja que usaran todos los árboles. Este parámetro puede optimizarse o heredar del óptimo de los modelos de árbol.

Las ventajas de estos agregados de árboles son, que se arreglan con éxito los principales problemas de los árboles manteniendo las ventas de robustez frente a atípicos y multicolinealidad, y que aumenta significativamente el poder predictivo. Sin embargo,

el modelo pierde mucha aplicabilidad ya que no es factible mirar uno a uno los cientos de árboles que componen un *Random Forest*.

***Gradiente Boosting*:** se considera la modalidad de Gradient Boosting para clasificación binaria (Bühlmann, P., & Hothorn, T, 2007). Este modelo también se basa en los árboles de decisión, pero sigue una filosofía distinta. En este caso en vez de crear un bosque de árboles como el *Random Forest* o el *Bagging*, primero crea un modelo logit básico basándose únicamente en la proporción global de eventos. Una vez se obtiene esa función se calcula el error cometido también llamado el residuo del gradiente. Este error se intenta predecir mediante árboles de decisión usando todas las variables independientes, tomando como variable dependiente el error cometido. Se incorporan las predicciones del error al modelo original, lo que crea un nuevo error reducido. Se vuelve al paso de predicción del error mediante árboles y se van repitiendo hasta que el error desaparece o se realiza e*arly stopping* limitando el número de iteraciones.

Estos modelos tienen la ventaja de tener un gran poder predictivo, aunque a costa de una reducida explicabilidad, y, además, mantiene muchas de las ventajas que dan los árboles de decisión, como la robustez frente a atípicos, variables irrelevantes o multicolinealidad. Por último, también es capaz de corregir la falta de estabilidad del modelo predictivo.

Aunque este modelo también tiene desventajas, como puede ser la tendencia a sobreajustar si no se controla, parando el algoritmo antes de que elimine por completo el error del gradiente. También se puede limitar el sobreajuste con el parámetro de regularización (*srinkage*), que controla el efecto de los nuevos modelos. A media que se aumenta el parámetro el modelo converge más rápidamente y llega antes a error 0 en entrenamiento, pero pierde precisión. Otra limitación en cuanto a su universalidad se da en casos donde los datos no presentan mucha complejidad, el modelo tiende a ser peor que otros más sencillos como la regresión logística.

Extreme Gradient Boosting: es una variación del Gradient boosting que modifica cómo se construyen los árboles de decisión y cómo lidia con el sobreajuste mediante la reducción del error usando la regularización con los parámetros *gamma* y *lambda* (Anghel, A., Papandreou, N., Parnell, T., De Palma, A., & Pozidis, H,2018). Esta técnica trata de reducir el sobreajuste penalizando árboles con más hojas. También se diferencia de Gradient boosting en cuanto a la generalización, ya que se puedan usar más funciones objetivo.

Este algoritmo tiene las ventajas frente al Gradient boosting además de otras formas de evitar el sobreajuste, permitiendo la creación de modelos teóricamente más finos. También mantiene todas las características positivas del Gradient boosting y de los árboles, aunque es un modelo más complicado de implementar ya que hay que controlar la regularización y todos sus hiperparámetro.

Dentro de los hiperparámetros a controlar, además del parámetro *lambda* y *gamma*, que serían análogos al *srinkage* de las Redes Neuronales y al CP de los árboles respectivamente, se puede tunear la profundidad de los árboles, la cantidad que se realizan, o el *min child weight*, que asegura un mínimo de instancias en un nodo para que se realice una nueva partición.

***Suport vector machine*:** el modelo predictivo *Suport Vector Machine* (SVM) trata de encontrar una separación lineal mediante un hiperplano que sea capaz de segregar perfectamente las observaciones de clase =0 de las de clase =1 y que, además, sea el plano que maximiza la distancia entre las 2 categorías (Mammone, A., Turchi, M., & Cristianini, N, 2009). No suele darse el caso de que exista un hiperplano que resulte en una separación perfecta, por lo que el algoritmo permite que haya errores lo que a su vez genera un residuo. Esto incorpora el hiperparámetro C de regularización que indica cuanto se penalizan estos errores.

Además de que no suele haber una separación perfecta, esta no suele ser lineal para problemas complejos. Para solucionar este inconveniente se podría trabajar en un espacio con dimensionalidad más alta donde una separación lineal si funcione. Una forma de aumentar la dimensionalidad a partir de las dimensiones originales, sin que se haga computacionalmente inviable los cálculos (ya que escalan geométricamente), es mediante una función Kernel, también conocido como el Kernel *trick* (Schölkopf, B, 2000). Hay varios tipos de Kernel, siendo los más comunes y los que se van a probar: el lineal y el polinomial de grado 2.

Para tunear la arquitectura de este algoritmo se utiliza el hiperparámetro C que penaliza el error, y los hiperparámetros característicos de cada función Kernel, como puede ser la gamma (*scale*) que influye en la complejidad del modelo y puede llevar a un sobreajuste si es muy alta.

Este algoritmo tiene algunas ventajas frente a los anteriores, pues es capaz de mejorar a la regresión lineal cuando los datos sí son lineales, además de presentar mucha flexibilidad, aportada por los diferentes tipos de Kernel, aunque también tiene limitaciones como el no lidiar directamente por la multicolinealidad o requerir mucho tiempo de computación para desarrollarse

Métodos de ensamblado: cuando varios algoritmos tienen una calidad parecida al predecir los datos, se pueden utilizar los métodos de ensamblado. Estos se basan en combinar diferentes algoritmos para mejorar la predicción (Fernández, R, 2021). Esto reduce la varianza y el sesgo que tiene cada modelo individualmente y aumenta el poder predictivo final.

Hay diversas forma de realizar el ensamblado en clasificación. Se puede hacer una media de la probabilidad de que haya suceso de todos los modelos, se puede seleccionar la categoría que indiquen la mayoría de las modelos o se puede hacer una ponderación de los resultados.

La única desventaja de realizar modelos de ensamblado es su dificultad añadida al tener que ajustar numerosos modelos, y que se reduce aún más la explicabilidad y se pasa de un modelo de caja negra a muchos modelos de caja negra, y para una clasificación final es prácticamente imposible decir cómo se ha llegado a ese resultado. Además, aunque se puede hacer una estimación, no hay ninguna forma de saber con certeza a *priori* que combinaciones de modelos o forma de ensamblado tendrán más éxito, por lo que se deben probar todas ellas. Por otro lado, para comprobar la robustez de este ensamblado si se debe realizar validación cruzada en el conjunto de datos test.

Durante la modelización los distintos algoritmos serán creado a partir d ellos datos del conjunto *train* y comparados entre sí por la variable de bondad de ajuste *accuracy* sin ponderar, con el conjunto de datos *validate*. Este método castiga más el sobreajuste que la validación cruzada al usarse en la validación observaciones no utilizadas por el modelo en su creación. Además, tiene la ventaja de ser significativamente más rápido en cuanto a realizar la validación y el entrenamiento, esto es muy relevante en este trabajo, pues se lidia con un conjunto de datos muy grande.

Finalmente se realizará una elección de modelos de entre los elegidos como óptimos de cada algoritmo. Para esta elección final sí se utilizará validación cruzada con el conjunto de datos *validate*. De esta forma se podrá tener una idea de la robustez de cada modelo frente a los cambios en los datos *input* y se evitará cualquier sesgo tanto de sobreajuste a los datos train como de hiperparametrización centrada en los datos *validate*.

Para esa validación cruzada se ha creado una función a medida en R que separa los individuos por su identificador de partido creando un fichero en el que asigna a cada identificador uno de los grupos de la validación cruzada. Después une la base de datos con la que se trabaja con este fichero mediante el identificador, asegurándose que todas las horas de un partido se encuentren dentro del mismo grupo.

Metodología XAI: dentro de las conclusiones se utilizará este conjunto de algoritmos para poder hacerse una idea de la influencia que tienen las variables independientes sobre la variable objetivo en los modelos de caja negra (*Chapter 58 introduction to XAI (explainable AI) in R*, s/f). Se utilizarán las siguiente técnicas:

- *Permutaion feature importance*: Esta técnica baraja los valores reales de una variable y evalúa cuanta calidad pierde el modelo. De esta forma se puede obtener una idea de la importancia de una variable en relación con el resto.
- *PDP, ICE plot*: En este gráfico se ve el impacto marginal que tiene cada variable dependiente sobre la independiente, dibujando cuanto cambia la variable objetivo en respuesta a la modificación de cada variable independiente. Esto se aprecia en el gráfico *Individual Conditional Expectation* (ICE). La media de todos estos gráficos crea el *gráfico Partial Dependence Plot*(PDP) que representa la relación la variable dependiente y la independiente, resultante de agregar las evoluciones de predicciones individuales.

En cuanto al *software,* será empleado en la modelización principalmente la librería de caret del lenguaje R (Kuhn, M, 2019, marzo 27). A excepción de los modelos de árbol y Random Forest, para los cuales se usarán las librerías rpart y randomForest respectivamente. Para reducir la carga computacional del trabajo, se realizará en paralelo entre el portal de R estudio y 7 máquinas virtuales en Google colab de tipo CPU high-ram. Se utilizarán otras librerías de apoyo con diferentes funciones. Estas son:

- visualización de datos: ggplot2, rpart.plot, corrplot.
- Estructuración y lectura: dplyr, tidyir.
- Tratamiento de fechas: hms. Lubridate.
- Selección de variables: Boruta, MXM.

Preprocesamiento de la base de datos

Los datos que utiliza este trabajo provienen de un reto de *Kaggle*. Estos datos recogen información sobre 31.074 partidos de fútbol en 1005 ligas alrededor del mundo entre el 9/1/2015 hasta el 22/11/2016, con las cuotas de apuestas tipo resultado a tiempo completo de 32 casas de apuestas, durante 72 horas antes de que empezasen los partidos.

Cada fila corresponde a un partido concreto identificado mediante su ID. Cada columna tiene una cuota para ese partido. La información de la casa, la hora y a que resultado se refiere esta codificada dentro del nombre de las propias columnas. Por ejemplo, la columna "home_b1_1" indica el ratio para el resultado de que gane el equipo local en la casa de apuestas b1 en la hora 1 (teniendo en cuenta que la hora 0 es cuando se empieza a recoger datos y la hora 72 la última observación antes de que empiece el partido).

Esta base de datos original tiene 6.917 columnas por 31.074 filas, sin embargo, muchas de las celdas están vacías ya que no todas las CA participaron en todos los partidos, y no todas ofrecían cuotas 72 horas antes. Además, no se puede realizar un análisis descriptivo en condiciones si la mayor parte de la información categórica no se encuentra en las variables si no en el nombre de esta.

Por todos estos motivos realizamos una primera transformación o manipulación como indica la metodología SEMMA. Se crea una base de datos intermedia llamada os_desglosado, en la cual tenemos una columna para match_id, 32 para cada una de las casas b1 a b32 con los ratios, una columna para la hora a la que pertenecen los ratios de las casas y otra columna llamada bet_class que indica a que resultado se apuesta las cuotas de la fila (home, away o draw). También se crea nuestra variable objetivo intermedia hay_arbitra que toma el valor 0 si no hay arbitraje y 1 si lo hay en esa hora. Esta base de datos se crea para evaluar la prevalencia del arbitraje entre casas en el mercado y para determinar si se debe tomar un punto de corte a partir del cual empezar a predecir y si el nivel de arbitraje es suficiente para que merezca la pena predecirlo.

Lo primero que se estudia es la evolución de la proporción de arbitraje y de partidos sin cuotas. Como se ve en el gráfico 1 a medida que nos vamos acercando al momento del evento disminuye el número de partidos sin cuotas hasta llegar a 0 y la cantidad de arbitraje absoluta aumenta.

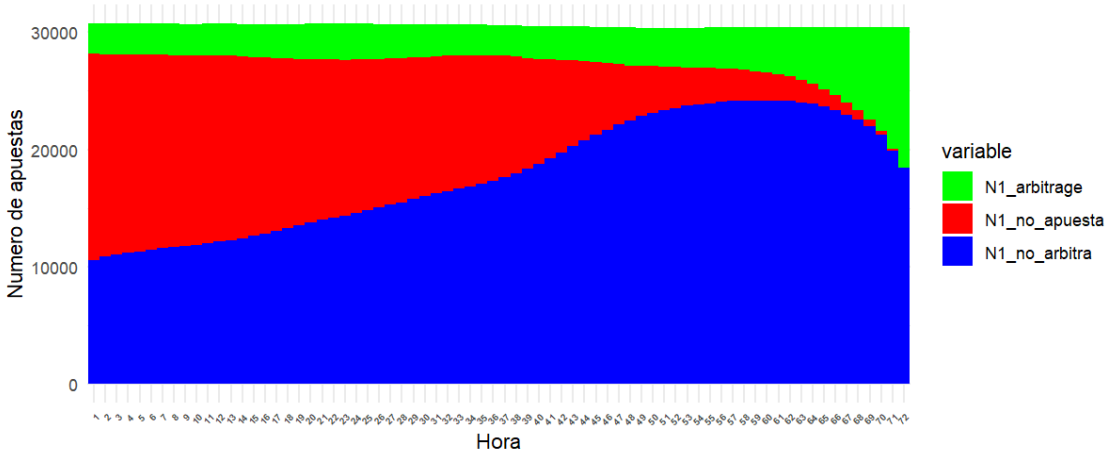


Gráfico 1: evolución de los partidos con cuotas, sin cuotas, y arbitraje en el tiempo.

En este gráfico se pueden distinguir 3 etapas del mercado: una inicial con muchos partidos sin apuestas, una etapa intermedia en la cual el número de arbitrajes se mantiene más o menos estable a medida que se van sacando cuotas apara los partidos que no tenían, y una etapa final en la cual el interés por el partido aumenta rápidamente y se puede asumir que el mercado evoluciona rápidamente, ya que muchas CA añaden apuestas a los partidos y se van creando más situaciones de arbitraje. Esto plantea la cuestión de si hay un momento óptimo en el que empezar a predecir el arbitraje y si se debería dividir las 72 horas en una sección de entrenamiento y otra de predicción.

Como muestra el gráfico 2 la proporción de partidos en los que hay arbitraje no es constante y fluctúa entre un 12% y un 35%, distinguiéndose muy bien las tres etapas del mercado. Al inicio cuando los ratios son frescos y las casa los sacan ,y el mercado de clientes todavía no ha tenido mucha influencia, se ve que hay una proporción más elevada de arbitraje. Una vez madura la apuesta se llega a un valle en el cual hay poco arbitraje proporcional pues se ha llegado a un equilibrio en el mercado y no hay un flujo de nuevas apuestas por parte de los clientes suficientemente alto como para provocar un aumento en el arbitraje, mientras que sí aumentan las casa que ofrecen apuestas. Finalmente se distingue la tercera etapa, justo antes del partido, en la que muchas apuestas nuevas mueven las cuotas previamente estables, creando arbitrajes, y obteniéndose la mayor proporción de estos en el intervalo temporal.

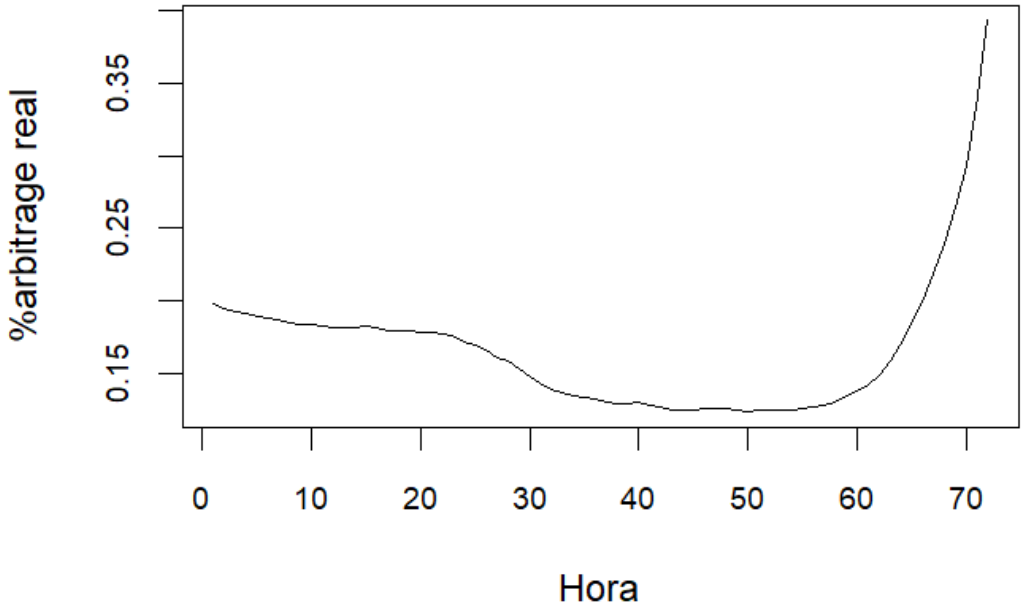


Gráfico 2: evolución de la proporción de partidos con arbitraje en el tiempo.

El gráfico 2 parece indicar que se debería empezar a predecir a partir de la hora 60. Sin embargo, si miramos cómo evoluciona el porcentaje de todas las situación de arbitraje que sucederán en la ventana temporal, reflejado en el gráfico 3, es prácticamente una evolución constante con un pequeño pico al final. Esto se debe a que la disminución en proporción de arbitraje en el periodo intermedio se debe a que se incorporan más partidos con ratios al mercado que la cantidad de situaciones de arbitraje que aparecen. Por esto mismo, si se empezase a predecir a partir de la hora 60, se perderían un 80% de las oportunidades de arbitraje del mercado. Con esto podemos concluir que no hay un punto de corte claro en el que empezar a predecir y, por lo tanto, es correcto crear un modelo predictivo general válido para cualquier hora.

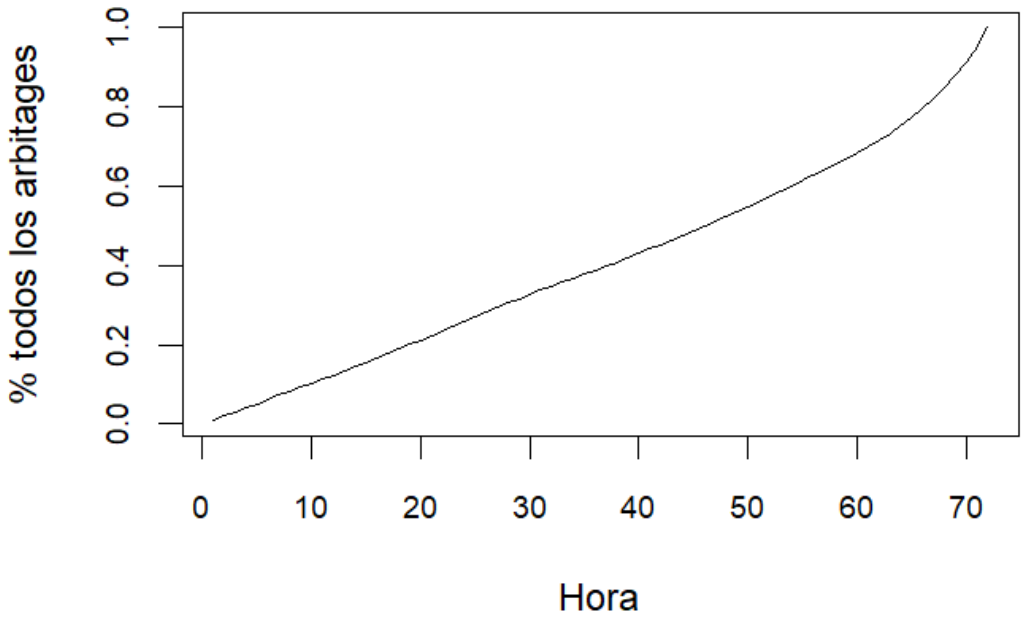


Gráfico 3: porcentaje de arbitrajes totales por hora.

El siguiente paso es transformar la base de datos intermedia en la base de datos final. Utilizamos distintas funciones que calculan agregados de la información para un *lag* temporal determinado. En este paso se va a agregar información de las cuotas de las 32 casa de apuestas. Se crean las variables por triplicado, una para cada tipo de resultado de forma que, por ejemplo, la función que calcula los ratios medios creara 3 columnas llamadas Mean_away, Mean_home y Mean_draw, Con los ratios medios para *away*, *home* y *draw* respectivamente.

Todas las variables nuevas se calculan sobre un periodo temporal que será idéntico para todas ellas. Es decir, se calculan sobre una hora concreta (que se registra en la variable hora) utilizando, además, un número determinado de horas previas. Ese espacio de tiempo lo denominaremos PT (Periodo temporal). Y el número de horas anteriores lo determina el lag. No hay una forma clara de determinar el *lag* óptimo a priori. En un trabajo real para elegir el *lag* óptimo se debe considerar distintas opciones, como podrían ser 6, 12, 18 y 24 horas. Y realizar toda la metodología para cada opción, para, posteriormente, elegir la que mejor prediga. Visto que este TFM no tiene como objetivo ver que ventana temporal es la óptima y, por limitaciones de tiempo y poder de computación, usaremos solamente un lag de 6.

Otra nomenclatura que se utilizará para la diferencia entre el ratio mínimo y máximo será el rango. Este se suele calcular sobre dos conjuntos: intra-casa, es decir, la diferencia entre el ratio mínimo y máximo en una casa concreta a lo largo del tiempo. O extra-casa, comprando el mínimo y máximo entre casas, en una hora concreta o a lo largo del PT.

Además, se modifica la variable objetivo, ya que pasa de ser “si hay o no arbitraje en esta hora” a “si lo habrá en la siguiente”.

Las variables agregadas que se calculan son las siguientes:

- **Mean_**: media de los ratios de las casa de apuestas en el periodo.
- **Max_**: El ratio máximo entre casa de apuestas en el periodo.
- **Min_**: El ratio mínimo entre casa de apuestas el periodo**.**
- **Mean_ .in**: La media del rango intra-casa. Es decir, la media del rango de todas las casa en el PT.
- **Max_.in**: El máximo del rango intra-casa.
- **Min_.in**: El mínimo del del rango intra-casa**.**
- **Max_.ex**: El máximo rango extra-casa. Es decir, la diferencia entre el ratio máximo y el mínimo de todas las casa en el PT.
- **Min_.ex**: El mínimo rango extra-casa.
- **H_mean_.ex**: La media del rango extra-casa dentro de cada hora (en este caso se calcula el rango de la hora 1,2…n y se hace la media de estos).
- **H_max_.ex**: El máximo rango extra-casa dentro de cada hora.
- **H_min_.ex**: El mínimo rango extra-casa dentro de cada hora.
- **Horario**: variable categórica que toma valores entre mañana, tarde, noche, que agrega la hora a la que se realizara el partido, siendo mañana, tarde y noche los periodos de tiempo 00:01-08:00, 08:01-16:00, 16:01-24:00 respectivamente.
- **Mes**: variable categórica en que mes se realiza el partido.
- **Hora**: 72 menos el número de horas que faltan para el inicio del partido.
- **Match_time:** la hora a la que será el partido.
- **Hay_arbitra**: la variable objetivo-binaria que indica si en la siguiente hora se dará una situación de arbitraje.

La base de datos final sobre la que trabajaremos tiene 38 variables y 2.019.810 filas.

Exploración y Depuración de la base de datos

Una vez se obtiene la base de datos final, y siguiendo con las pautas marcadas por la metodología SEMMA, se realizará un análisis exploratorio.

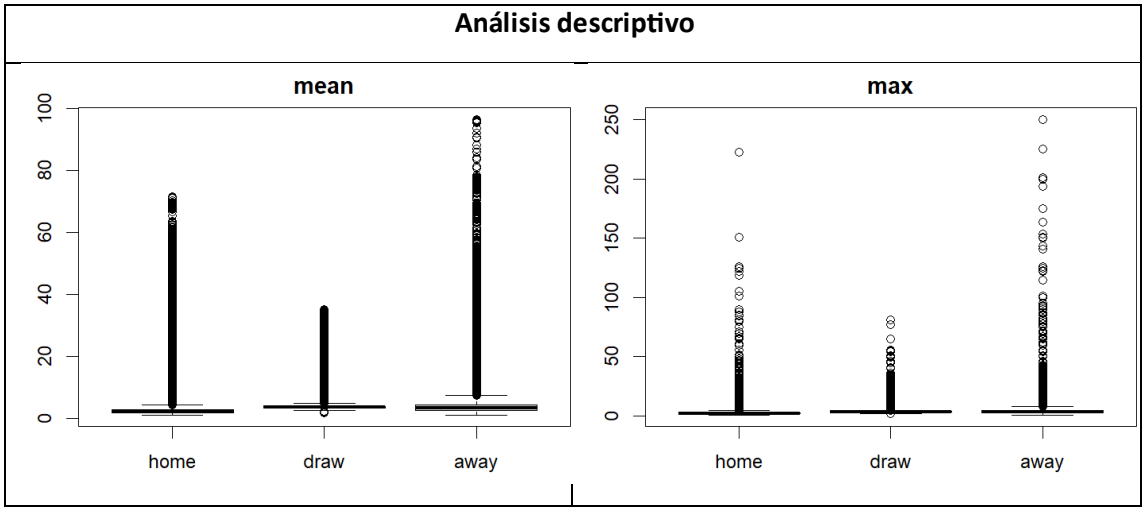
Lo primero que se detecta es la gran cantidad de datos faltantes (NA). Como se veía en el gráfico 1 hay una proporción importante de partidos en los cuales ninguna casa ofrece cuotas para unas determinadas horas. En estos casos se tendrán filas con prácticamente todas las columnas NA. Estos datos faltantes no se deben a un error, sino que indican

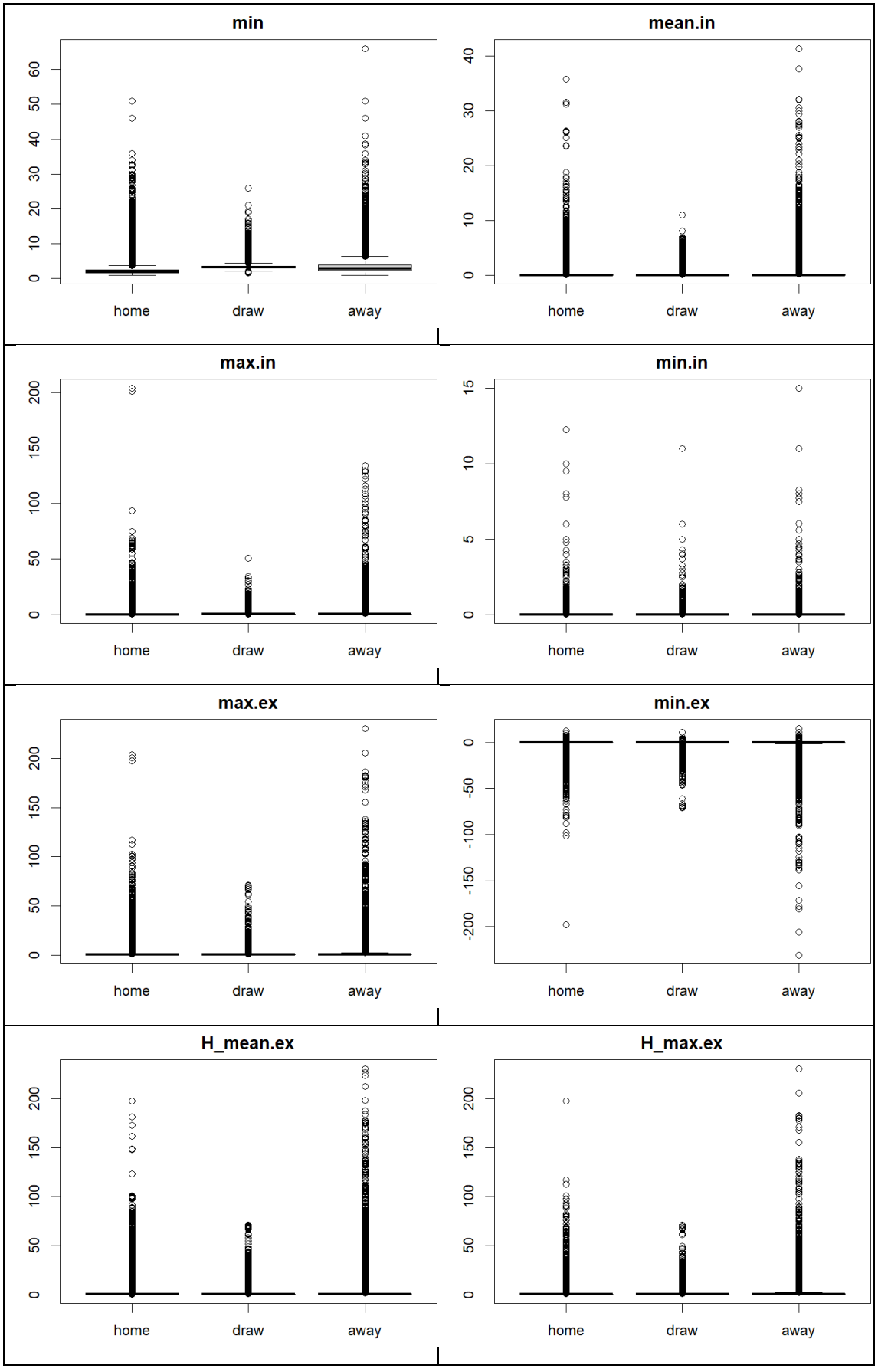
que a esa hora a todos los efectos prácticos del mercado no hay partido, y, por lo tanto, no tendría sentido imputar los datos faltantes.

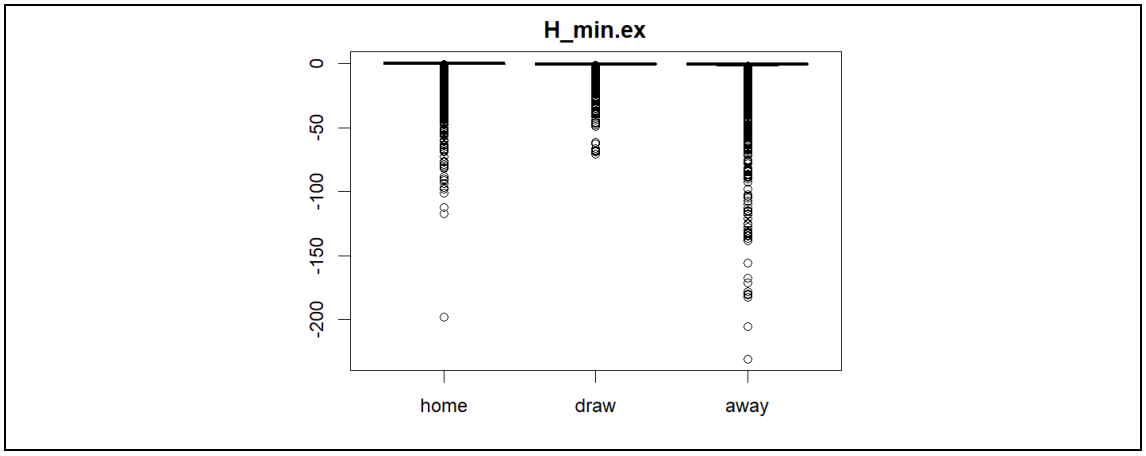
El tratamiento que realizaremos será eliminar todas las filas en las que en las 3 primeras columnas (la media de las cuotas de los 3 resultados) tengan algún NA. Si no hay ratios de ninguna casa se detectara en alguna en las 3 primeras columnas. Pero también se dan algunos casos raros en los que sí tenemos 1 cuota para uno o dos de los resultados. Esto se debe a alguna casa que no publica las 3 cuotas a la vez. En estos casos tampoco se puede determinar el arbitraje, y no tiene sentido imputar ese dato faltante por ser un caso tan extremo en el que para un partido solo una casa publica una cuota para un solo resultado. Es razonable eliminar estas filas ya que se trata de una situación causada por un error de publicación de cuota de la casa.

Respecto al análisis exploratorio visual se realizan los histogramas (anexo 1) donde resalta, a simple vista, la cantidad de atípicos que hay. Estos se pueden visualizar bien en los diagramas de cajas y bigotes para las variables numéricas que se pueden encontrar en la tabla 1.

Tabla 1: gráficos de cajas y bigotes para las variables numéricas de la base de datos final.




min
0
10
20
30
40
50
60
home
draw
away
mean.in
0
10
20
30
40
home
draw
away
max.in
0
50
100
150
200
home
draw
away
min.in
0
5
10
15
home
draw
away
max.ex
0
50
100
150
200
home
draw
away
min.ex
-200
-150
-100
-50
0
home
draw
away
H_mean.ex
0
50
100
150
200
home
draw
away
H_max.ex
0
50
100
150
200
home
draw
away



Los atípicos presentes en los datos provienen de situaciones raras del mercado, y no a errores en la transformación (consideramos atípicos aquellos valores fuera del intervalo media +-1.5*rango intercuartílico). Visto que se intenta predecir una anomalía del mercado es razonable esperar que haya una correlación entre los atípicos y la variable respuesta. Mediante el uso de la función de detección de *outliers* hecha a medid , como se ve en la imagen 2 , que crea una columna con un 1 si en la fila hay algún valor atípico y 0 si no lo hay. Esta sospecha se confirma tal y como se muestra en la tabla 2, ya que para todas las situaciones de arbitraje hay al menos una de sus variables que presenta un atípico.

```r
detectar_outliers <- function(df) {
  outliers <- rep(0, nrow(df))  # Vector de ceros para almacenar resultados

  for (col in names(df)) {
    if (is.numeric(df[[col]])) {      # Verificar si la columna es numérica
      q1 <- quantile(df[[col]], 0.25) # cuantil superior
      q3 <- quantile(df[[col]], 0.75) # cuantil inferior
      iqr <- q3 - q1                  # rango intercuantilico
      lower_bound <- q1 - 1.5 * iqr
      upper_bound <- q3 + 1.5 * iqr

      outliers <- outliers | (df[[col]] < lower_bound | df[[col]] > upper_bound) #deteccion de outliers
    }
  }

  return(outliers)
}

hay_atipico<-detectar_outliers(datos_sna6)
hay_atipico<-as.integer(hay_atipico)
hay_arbitra_df<-(datos_sna6$hay_arbitra)


table(hay_atipico,hay_arbitra_df)
```

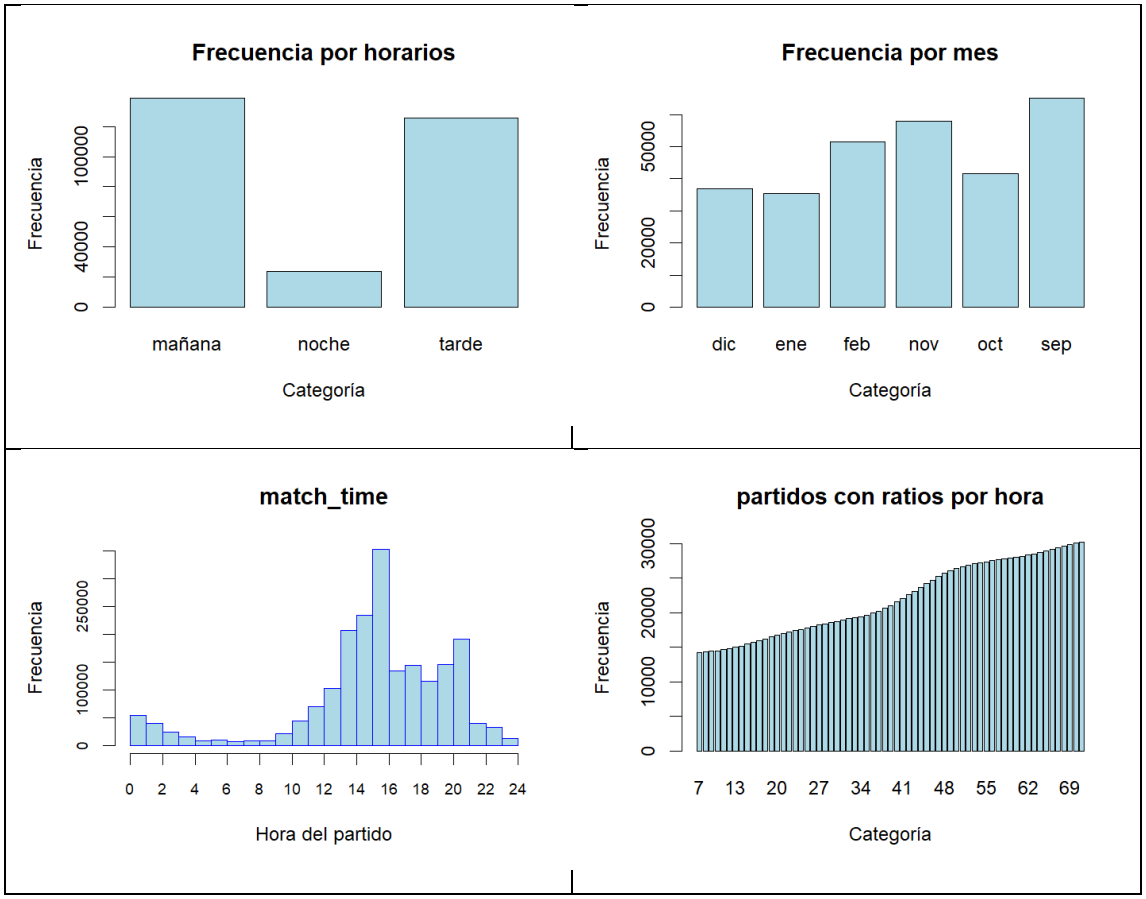
Imagen 2: Función a medida de detección de outliers por filas

Tabla 2: matriz de correlación entre filas con atípicos y filas con arbitraje.

	Hay arbitraje		
Hay atípico	**0**	**1**	
0	717246	0	50,06%
1	476068	239351	49,94%
	83.29%	16.71%	100%

Esta alta presencia de atípicos hace imposible plantearse la normalidad de los datos. También se realizan unas representaciones graficas de las variables categóricas que se muestran en el tabla 3. Tal y como se esperaba en los gráficos de la tabla 3 se aprecia que hay pocos partidos por la noche, y que la cantidad de partidos para los que hay cuotas aumenta se forma estable a medida que se acerca la hora 72.

Tabla 3: Diagrama de barras para las variables categóricas de la base de datos final.



En cuanto a la correlación de las de las variables como se puede observar en el gráfico 4, cómo existen correlaciones fuertes tanto positivas como negativas entre nuestras variables.

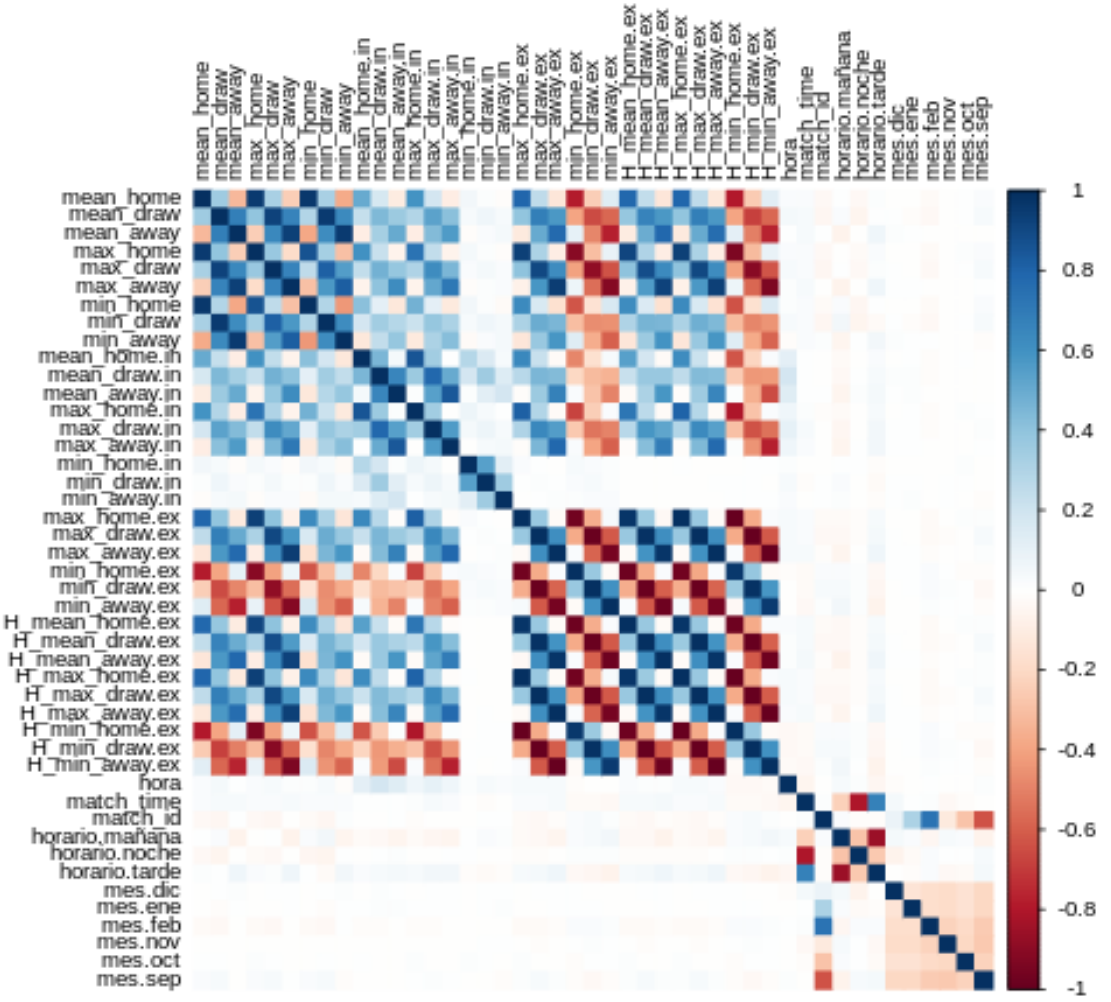


Gráfico 4: *heatmap* de la matriz de correlaciones de las variables numéricas.

Como cabría esperar por la forma en la que se han creado las variables, se encuentran grupos de 3 columnas (las que hacen referencia a un tipo de agregado para cada distinto resultado del partido), que se relacionan de forma parecida con el resto de las variables, pero que, sorprendentemente, no tienen siempre correlaciones muy fuertes entre ellas.

Estos resultados confirman que nuestras variables presentan multicolinealidad y que por lo tanto hace falta realizar una selección de variables previa a la modelización mediante algoritmos basados en componentes lineales, como la regresión logística, las Redes Neuronales o SVM.

Finalmente, de forma agregada nuestra base de datos tiene un 16.71% de casos de arbitraje como se ve en el gráfico 5. No es suficientemente bajo a priori como para que haya necesidad de añadir un peso artificial a los casos de arbitraje positivos, pero si para tener en cuenta que el modelo nulo tiene un accuracy del 83.29% que se deberá mejorar.

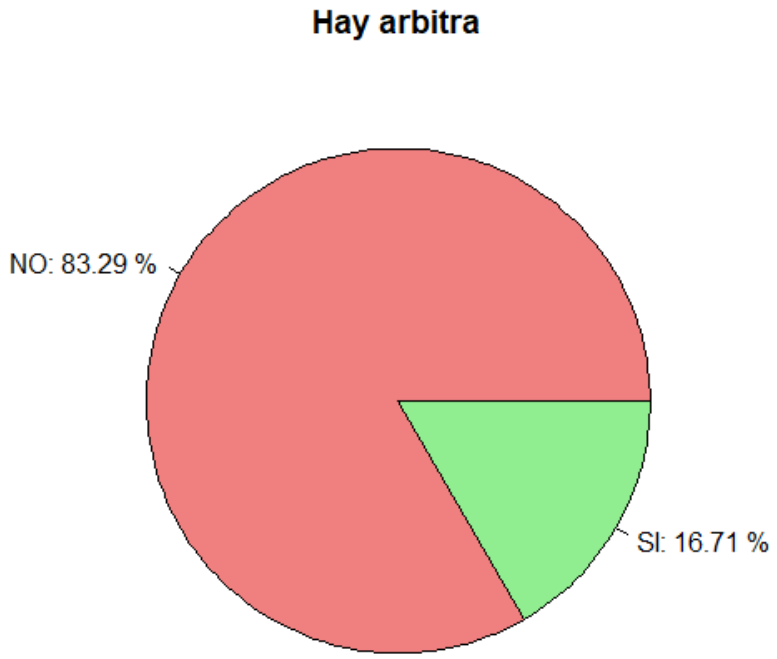


Gráfico 5: grafico de tartas para la proporción de arbitraje global.

A continuación, y siguiendo en la sección de modificación de la metodología SEMMA, se realiza una estandarización regular de las variables numéricas restándoles su media dividiéndolas por su varianza como se muestra en la siguiente formula.

$$Z_i = \frac{X_i - \bar{X}_i}{\sigma_i} \qquad con\ i = 1,2,3 \dots 35$$

Siendo X_i la variable numérica i estima (incluyendo además las variable hora y match time), Z_i la variable i estima estandarizada y σ_i, $\bar{X}$ la desviación típica y la media de la variable iesima respectivamente. Esto mejora el funcionamiento de determinados algoritmos, como pueden ser las redes neuronales, además de facilitar la comparación entre efectos de variables sobre modelos ya que todas las variables tienen media 0.

Como último paso antes de la modelización se realiza la división de la base de datos en los grupos *train*, *validate* y *test*, mediante muestreo estratificado proporcional de los partidos con un primer estrato en el horario y un segundo estrato en el mes del partido. Como se describe en la metodología se realizará una división de los datos en los grupos *train*, *validate* y *test* de un 20%, 40% y 40% de los datos respectivamente. Estos grupos acaban teniendo 6.214, 12.430 y 12.430 partidos que corresponden a 288.195, 572.258 y 572.212 filas no nulas respectivamente.

Este proceso se ha realizado mediante el uso de una función a medida realizada en R que se puede observar en la imagen 3.

```r
# crea muestra estratificada para horario y para mes
muestra_estrat<-function(datos_f,proporcion_m=0.2){
library(dplyr)
    set.seed(11101)

  datos_sna6<-datos_f

  length(unique(datos_sna6$match_id))

  df_uniques <- datos_sna6 %>%
    distinct(match_id, mes, horario)
  nrow(df_uniques)

  sum(is.na(df_uniques))

  prop_mes<-table(df_uniques$mes)/nrow(df_uniques)
  prop_horario<-table(df_uniques$horario)/nrow(df_uniques)

  df_uniques_d <- df_uniques %>%
    mutate(estrato = interaction(mes, horario))

  proporciones<-as.data.frame(table(df_uniques_d$estrato))
  proporciones$Freq<-proporciones$Freq/sum(table(df_uniques_d$estrato))


  tamaño_total_muestra <- round(proporcion_m*nrow(df_uniques))

  # Calcula el tamaño de muestra para cada estrato proporcional a su tamaño
  proporciones <- proporciones %>%
    mutate(tamaño_muestra =round(Freq * tamaño_total_muestra))

  # Realiza el muestreo estratificado proporcional
  set.seed(11101)
  trainIndex <- createDataPartition(df_uniques_d$estrato, p = tamaño_total_muestra/nrow(df_uniques_d), list = FALSE)
  muestra_estratificada <- df_uniques_d[trainIndex, ]
  table(muestra_estratificada$estrato)
  proporciones

  datos_sna6_train <- datos_sna6 %>%
    filter(match_id %in% muestra_estratificada$match_id)

  datos_sna6_test <- datos_sna6 %>%
    filter(!(match_id %in% muestra_estratificada$match_id))

  return(list(train=datos_sna6_train, test=datos_sna6_test))

}
```

Imagen 3: función a medida para crear los grupos *train*, *validate* y *test* estratificados y manteniendo las observaciones de cada partido en un mismo grupo

Selección de variables

Como se describe en la metodología, para evitar que la multicolinealidad dañe a los modelos que, por su arquitectura, no tienen formas intrínsecas de lidiar con ella, se realiza una selección de variables para la regresión logística. Los resultados en cuanto a la tasa de error se muestran en el gráfico 6.

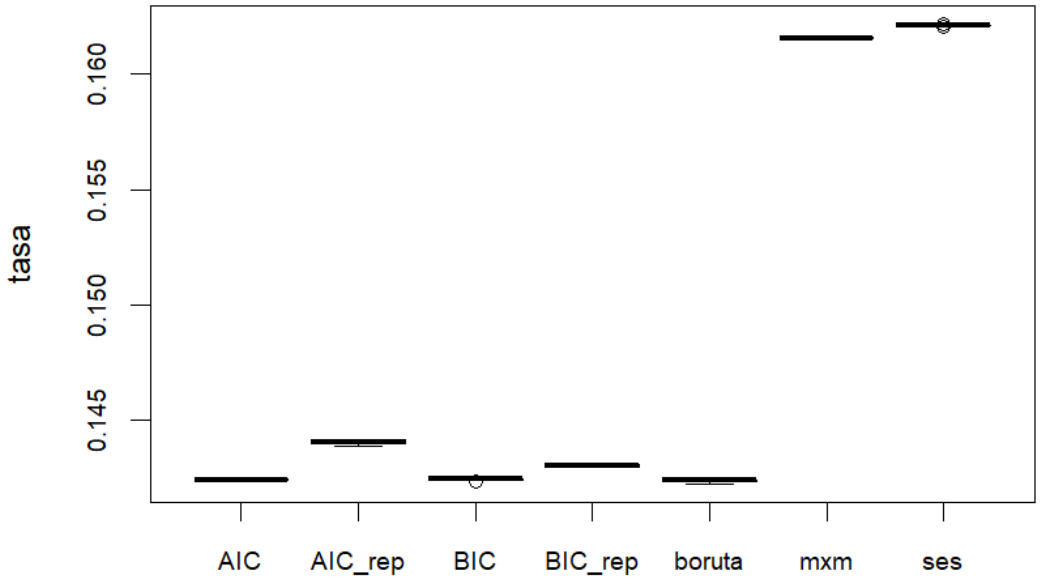


Gráfico 6: comparación de *boxplot* entre la tasa de fallo de las regresiones logísticas creadas a partir de variables seleccionadas por algoritmos de selección de variables.

De los 7 algoritmos se prueban, hay 5 claramente mejores que el resto. Aumentando esas 5 selecciones en el gráfico 7, se visualiza que valdría quedarse con las variables seleccionadas por AIC o por BIC. Nos quedaremos con las seleccionadas por AIC ya que su tasa media es marginalmente mejor.

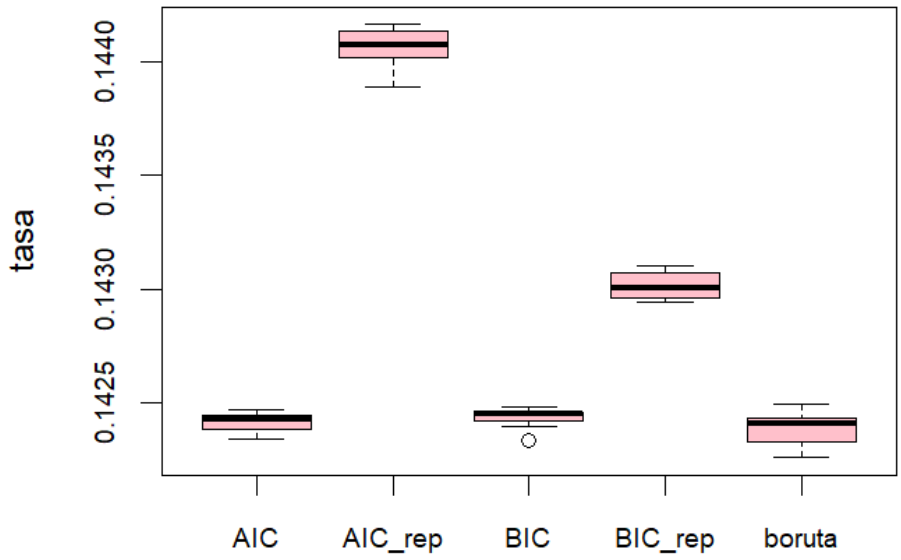


Gráfico 7: sección aumentada del grafico 6 con los 5 mejores algoritmos.

No se realiza una combinación de las variables más frecuentes en los modelos ya que esto no garantiza que se elimine la multicolinealidad. Se puede visualizar por qué con el siguiente ejemplo: si se tuviesen 6 variables y la 5 y la 6 estuviesen repetidas la mitad de los modelos seleccionar a la 5 y la otra mitad la 6. Si se cogiese un punto de corte de aparición del 60%, no usaría ninguna, por lo que perderíamos información relevante. Si por el contrario se usasen las 6 no se eliminaría multicolinealidad.

Regresión logística

Utilizaremos las variables seleccionadas anteriormente para construir una regresión logística. Este modelo, aunque no particularmente potente, nos permite hacernos una idea sobre la influencia de las variables en la probabilidad del arbitraje. En el gráfico 8 se ve reflejado la magnitud y el signo del impacto de estas variables en términos relativos.

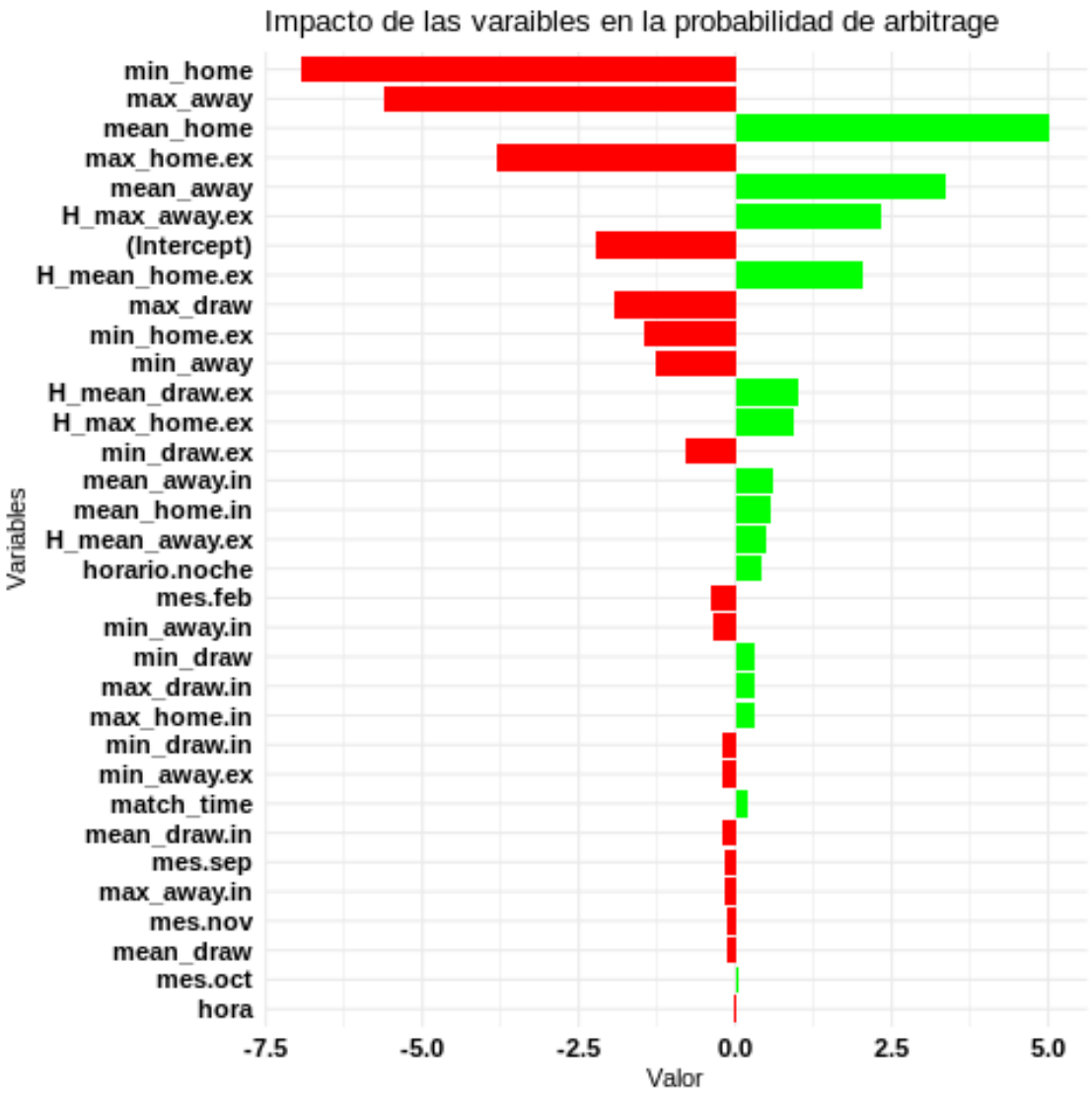


Gráfico 8: visualización de la magnitud y dirección del impacto de las variables independientes en la dependiente la regresión logística.

La tabla ANOVA de la imagen 4 muestra, con más detalle, las magnitudes y la variabilidad de estas, lo que se observa con menos detalle en el gráfico precedente.

```
Coefficients:
                Estimate Std. Error  z value Pr(>|z|)
(Intercept)    -2.236289   0.012805 -174.642  < 2e-16 ***
H_mean_draw.ex  1.018111   0.080388   12.665  < 2e-16 ***
min_draw        0.316937   0.104879    3.022  0.00251 **
min_home.ex    -1.443340   0.052402  -27.544  < 2e-16 ***
min_home       -6.921175   0.094986  -72.865  < 2e-16 ***
mean_home       5.025553   0.106601   47.144  < 2e-16 ***
min_away       -1.283127   0.142895   -8.980  < 2e-16 ***
mean_away       3.358430   0.080965   41.480  < 2e-16 ***
mes.feb        -0.395322   0.018028  -21.928  < 2e-16 ***
match_time      0.193690   0.009621   20.132  < 2e-16 ***
max_draw.in     0.316800   0.020347   15.570  < 2e-16 ***
max_draw       -1.944242   0.140189  -13.869  < 2e-16 ***
max_away       -5.611643   0.296633  -18.918  < 2e-16 ***
mean_away.in    0.595354   0.024583   24.218  < 2e-16 ***
min_away.ex    -0.195804   0.065077   -3.009  0.00262 **
min_away.in    -0.357742   0.028822  -12.412  < 2e-16 ***
H_max_away.ex   2.319805   0.200146   11.591  < 2e-16 ***
mean_home.in    0.556510   0.022196   25.073  < 2e-16 ***
max_home.ex    -3.815637   0.177524  -21.494  < 2e-16 ***
H_mean_home.ex  2.027881   0.069419   29.212  < 2e-16 ***
min_draw.ex    -0.792243   0.060514  -13.092  < 2e-16 ***
max_home.in     0.295278   0.028125   10.499  < 2e-16 ***
horario.noche   0.405856   0.035265   11.509  < 2e-16 ***
mes.sep        -0.171476   0.016139  -10.625  < 2e-16 ***
mes.nov        -0.142388   0.016625   -8.564  < 2e-16 ***
mean_draw.in   -0.188932   0.026041   -7.255 4.01e-13 ***
mean_draw      -0.129722   0.088980   -1.458  0.14488
max_away.in    -0.160240   0.024286   -6.598 4.16e-11 ***
H_mean_away.ex  0.507819   0.090731    5.597 2.18e-08 ***
H_max_home.ex   0.948839   0.173148    5.480 4.25e-08 ***
mes.oct         0.057577   0.017741    3.245  0.00117 **
min_draw.in    -0.199786   0.047151   -4.237 2.26e-05 ***
hora           -0.008356   0.005772   -1.448  0.14770
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

Imagen 4: tabla ANOVA de la regresión logística.

Se aprecia la eficacia de la selección de variables, ya que solamente 2 variables tienen un p-valor superior al nivel de significación estándar de 0.05 en el test de significación. Estas dos variables se habrán mantenido en el modelo pues minimizan el criterio de la información de Akaike, y probablemente no parezcan significativas por su relación no lineal con la variable objetivo.

Respecto a la validación de este modelo vemos en la matriz de confusión de este modelo, que se muestra en la tabla 4, y los estadísticos de esta en la imagen 5, indican que el modelo apenas mejora en un 1% el modelo nulo.

Tabla 4: matriz de confusión de la regresión logística sobre los datos de conjunto *validate.*

	Validate		
Predicción	**No**	**Si**	
No	473036	83580	97,26%
Si	4152	11490	49,94%
	83.38%	16.61%	100%

```
                Accuracy : 0.8467
                  95% CI : (0.8458, 0.8476)
    No Information Rate : 0.8339
    P-Value [Acc > NIR] : < 2.2e-16

                   Kappa : 0.1685

 Mcnemar's Test P-Value : < 2.2e-16

            Sensitivity : 0.9913
            Specificity : 0.1209
         Pos Pred Value : 0.8498
         Neg Pred Value : 0.7346
             Prevalence : 0.8339
         Detection Rate : 0.8266
   Detection Prevalence : 0.9727
      Balanced Accuracy : 0.5561
```

Imagen 5: estadísticos de bondad de ajuste matriz de confusión de la regresión logística sobre los datos de conjunto *validate.*

La utilidad principal de este modelo es aportar una idea sobre la dirección y magnitud del impacto de las variables independientes sobre la dependiente, y también se usará como base, junto con el modelo nulo, con la que comparar otros modelos más avanzados de *machine learning*, para ver si la pérdida en explicabilidad viene de la mano de una mejora significativa en la predicción.

Redes neuronales

El primer algoritmo avanzado que se va a probar es el modelo de redes neuronales, para el que vamos a tunear los hiperparámetros mencionados en la metodología: el número de iteraciones, para realizar *early stopping* y evitar así un posible sobreajuste; el decay, que controla cuanto se modifica la red neuronal con cada iteración; y el número de nodos, que controla el tamaño y complejidad de la red neuronal. Se mantendrán en su valor por defecto de *true el lineout* para que aplique una función lineal en la capa de salida, y de *false en Bagging*, para que no sortee observaciones en la creación del modelo. Los valores del *grill* a tunear son los siguiente:

- *Decay*: 0.001, 0.01, 0.1
- Nº de neuronas: 5, 10, 15, 20, 25
- Nº de iteraciones: 20, 50, 100, 200, 300, 500, 700, 1000

En el gráfico 9 se muestra como las distintas combinaciones de *decay* y tamaño reaccionan con el número de iteraciones. Los modelos con más nodos sufren un gran sobreajuste a partir de las 200 iteraciones, siendo este el punto en el que maximizan su curva de aprendizaje. Para los modelos más pequeños estas curvas son más suaves. En el caso del modelo con 10 nodos es robusto hasta las 500 iteraciones y el modelo de 5 nodos no llega a maximizar su curva de aprendizaje, aunque sí que reduce mucho su mejora, y por el progreso no se espere que supere al resto de modelos. Respecto al *decay* vemos, por lo general, que el mejor es 0.01, aunque no hay mucha diferencia entre los 3, en los modelos no sobre ajustados. Aunque se observa que los modelos con un *decay* más bajo tienden a obtener mejores resultados a medida que se incrementan las iteraciones ya que estos modelos requieren más tiempo para ajustarse a los datos lo que permite una convergencia más precisa.

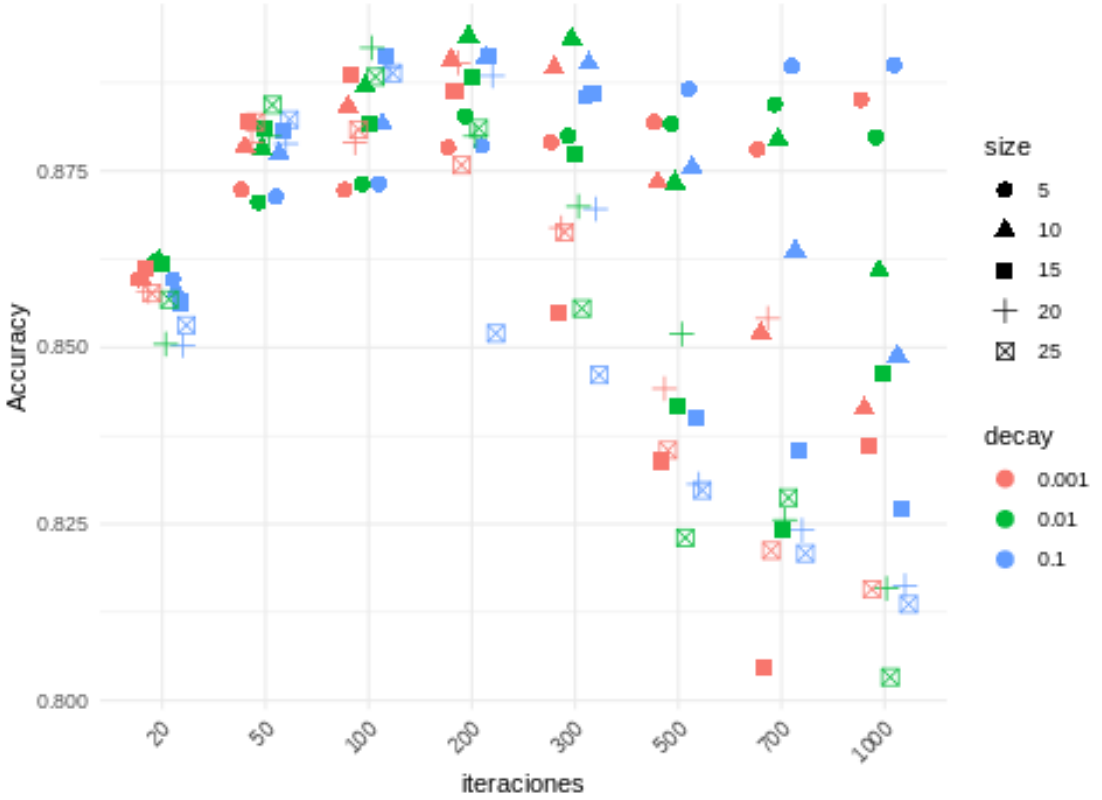


Gráfico 9: *grill* para el modelo de redes neuronales.

No hace falta probar más nodos ni iteraciones, ya que los modelos con más de 10 nodos alcanzan su máximo poder predictivo entre los 100 y 2000 nodos antes de sobreajustar. Y la curva de aprendizaje de los modelos de 5 nodos parece estabilizarse entre las 700 y las 1000 iteraciones.

Del grill realizado los mejores modelos obtenidos son los presentes en la tabla 5. Se determina como óptimo al modelo 1. Aunque los 5 mejores modelos son muy parecidos, los tres hiperparámetros del modelo 1 dan buenos resultados tanto en el punto óptimo, como para sus valores cercanos lo que aportara robustez al modelo. Se elegirá los hiperparámetros que den mejores resultados en general.

Tabla 5: mejores 5 modelos del grill de redes neuronales.

Modelo	Iter	size	decay	Accuracy	Sensitivity	Specificity
1	200	10	0.01	0.8941194	0.9734172	0.4960976
2	300	10	0.01	0.8937000	0.9693852	0.5138109
3	100	20	0.01	0.8924489	0.9600472	0.5531503
4	100	15	0.10	0.8914144	0.9628323	0.5329441
5	200	15	0.10	0.8912711	0.9518617	0.5871463

El modelo elegido de las Redes Neuronales tendrá 200 iteraciones decay de 0.01y 10 nodos

Arboles de decisión

En cuanto a los árboles de decisión se va a buscar optimizar los hiperparámetros descritos en la metodología, como son el tamaño mínimo de hoja o mb (*minbuket*) que va a ir desde un tamaño muy bajo de 5 que se espera que sobreajuste a un tamaño de 10000 3.4% de los datos, y el parámetro de complejidad (CP) que como indica la teoría, ayuda a simplificar o podar el árbol para eliminar nodos redundantes. Si se ven indicios de que aún se pueden obtener modelos mejores aumentando el tamaño mínimo de hoja, se continuaran probando mb más grandes. Los valores que se testearan en el grill son:

- CP: 0, 0.01, 0.05, 0.1, 0.5
- Mb: 5, 10, 30,50 ,70, 100, 150, 200, 300, 500, 1.000, 1.500, 2.000, 2.500, 3.500, 4.000, 4.500, 5.000, 5.500, 6.000, 7.000, 8.000, 9.000, 10.000

Como muestra el gráfico 10 el CP nos impide ver bien el efecto del mb, ya que los árboles con muchas ramas por tener un mb muy pequeño son podados hasta dar un árbol, el cual se repite para todos los valores de mb cercanos hasta que este crece lo suficiente para pasar al siguiente árbol, que también se repetirá por ser un óptimo local. Esto se aprecia en los gráficos escalonados de los árboles con CP>0. Por lo tanto, para

determinar el tamaño de hoja óptimo que se usará en el resto de los modelos basados en árboles, hay que mirar los puntos con CP=0. Se aprecia como el *accuracy* crece hasta un mb de 300 y a partir de ahí disminuye. También se ve un escalonamiento final debido a que el árbol con un mb de 3500 tiene hojas mucho más grades que el mínimo requerido, lo que provoca el mismo suceso de árboles repetidos por grupos de mb que teníamos antes.

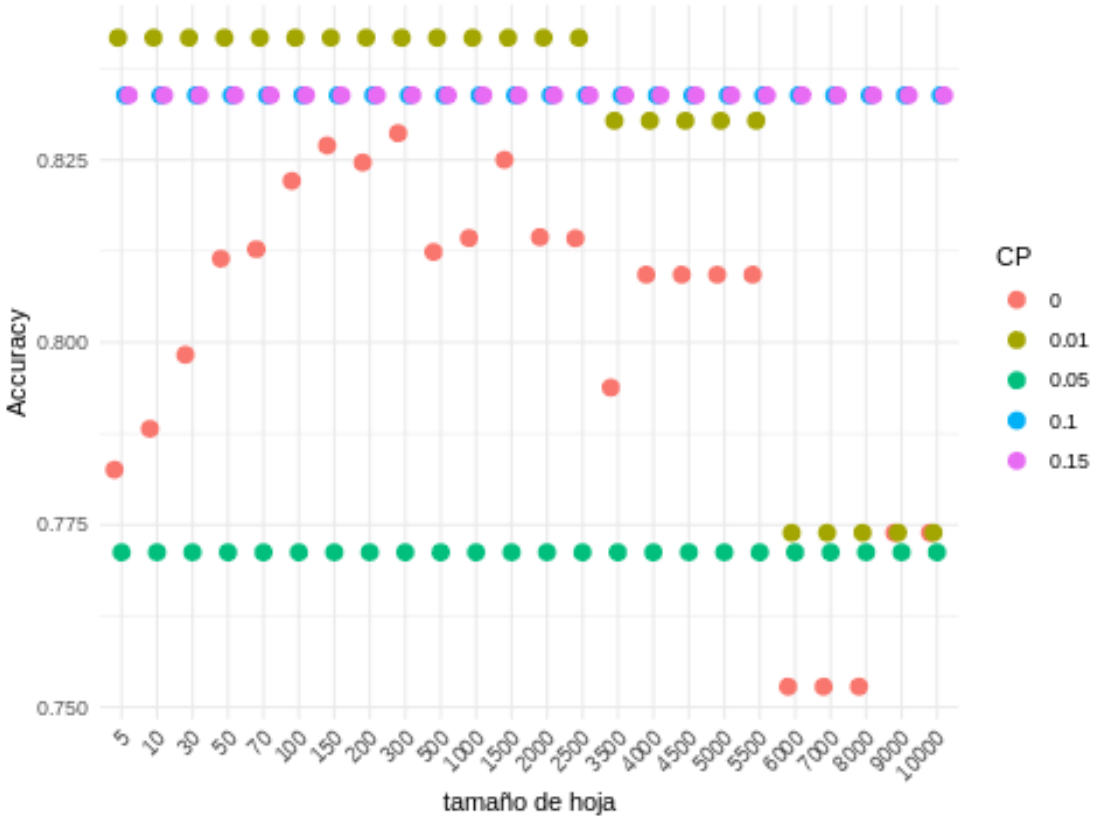


Gráfico 10: *grill* para el modelo de árboles de decisión.

Para el diagrama de árbol se usará el que posea un *accuracy* más alto con el máximo de mb, que, en teoría, es el que menos sobreajusta. Se tomará un cp de 0.01 y un mb de 2500. Los 5 mejores modelos en este caso son todos el mismo árbol por lo que no tiene sentido compararlos. Pero si se puede ver el diagrama del árbol seleccionado en el grafico 11, el cual nos dará una idea de las variables con más capacidad de segregar los partidos con y sin arbitraje.

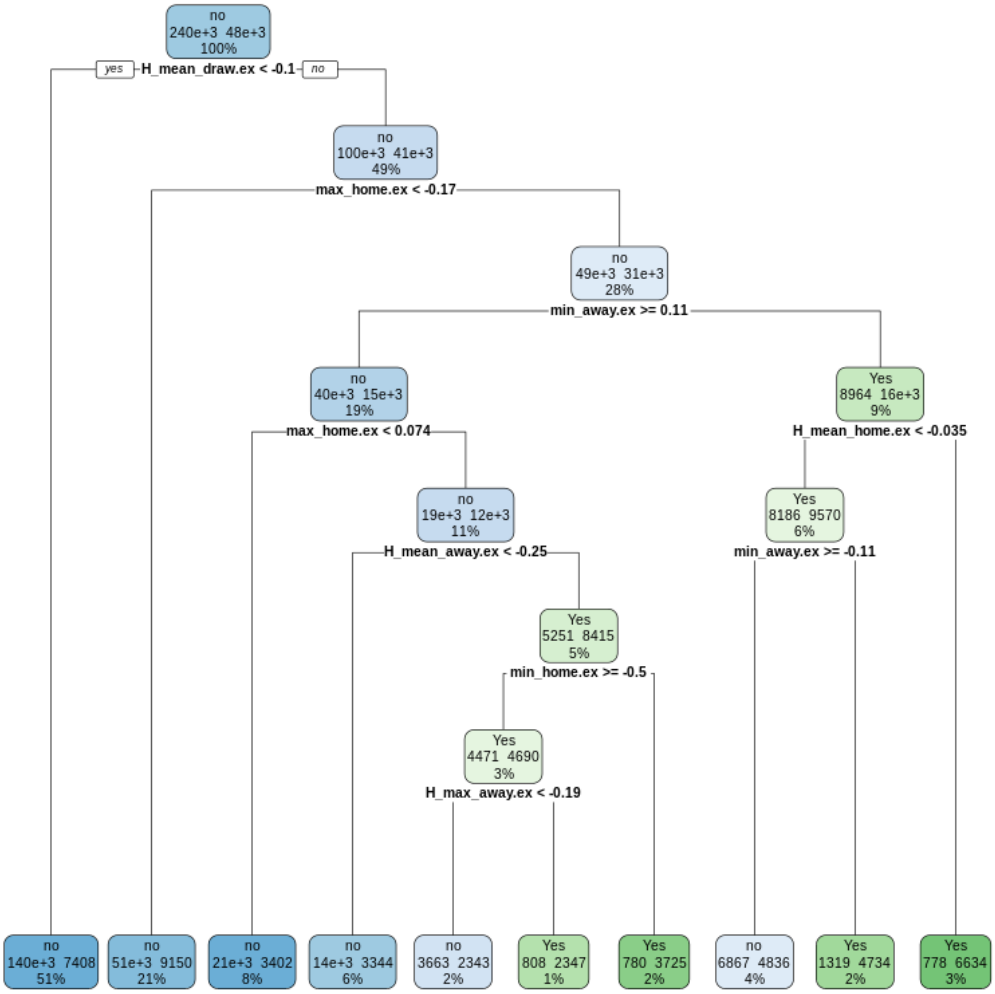


Gráfico 11: arbole de decisión seleccionado.

Vemos en el árbol que los partidos con una media del rango extra-casa para el resultado empate menor que -0.1 (recordamos que los valores están estandarizados y por eso puede dar negativo), no tiene prácticamente ninguna posibilidad de tener arbitraje, mientras que una combinación de valores altos en el máximo ratio al equipo local, un mínimo alto al equipo visitante, y a media extra-casa alta de los ratios al equipo local, dan la máxima probabilidad de que se de arbitraje.

Era de esperar que para determinar el arbitraje se necesitasen unos valores altos de medidas de ratios para los 3 posibles resultados.

Random forest

Para los modelos *Random Forest* se hereda el tamaño mínimo de hoja óptimo de los modelos de árbol. Este tamaño de hoja deberá tomarse considerando CP=0, ya que este hiperparámetro no se ajusta en los modelos *Random Forest* y por lo tanto toma un valor por defecto de 0. También se tuneará el número de árboles que compondrán nuestro bosque. Se espera encontrar un número de árboles a partir del cual el *accuracy* del bosque se estabilice. Al contrario que en redes neuronales, un excesivo número de árboles no sobreajusta, solo es un "desperdicio" de tiempo de computación y reduce la velocidad de predicción. También se probarán valores para el número de variables candidatas a cada nodo (*mtry*), incluido el máximo (44). Los valores que se van a probar en el *grill* son:

- Mtry: 44, 40, 30, 20, 10, 5
- Número de árboles: 50, 100, 300, 500, 700, 1000, 1200

Los Random forest convergen gráficamente en el grill del gráfico 12, tras una evolución inicial todos los modelos se estabilizan a partir de las 500 observaciones ya que se observa una progresión sin pendiente alrededor de una recta horizontal, aunque con una variación alrededor de esta. Si miramos como el error del conjunto de datos *train* del gráfico 13 se ve claramente como converge entre los 300 y 500 árboles, donde la diferencia es de milésimas en la tasa de error y se ve mucha menos variabilidad entre modelos tras la convergencia. También se observa que tasa de error es extremadamente baja, así que es posible que El modelo Random forest este sobreajustando a los datos de entrenamiento. En el *grill* con los datos de *validate* (gráfico 12) consideraremos las variaciones entre modelos pasados los 500 fruto del azar y no son significativas.

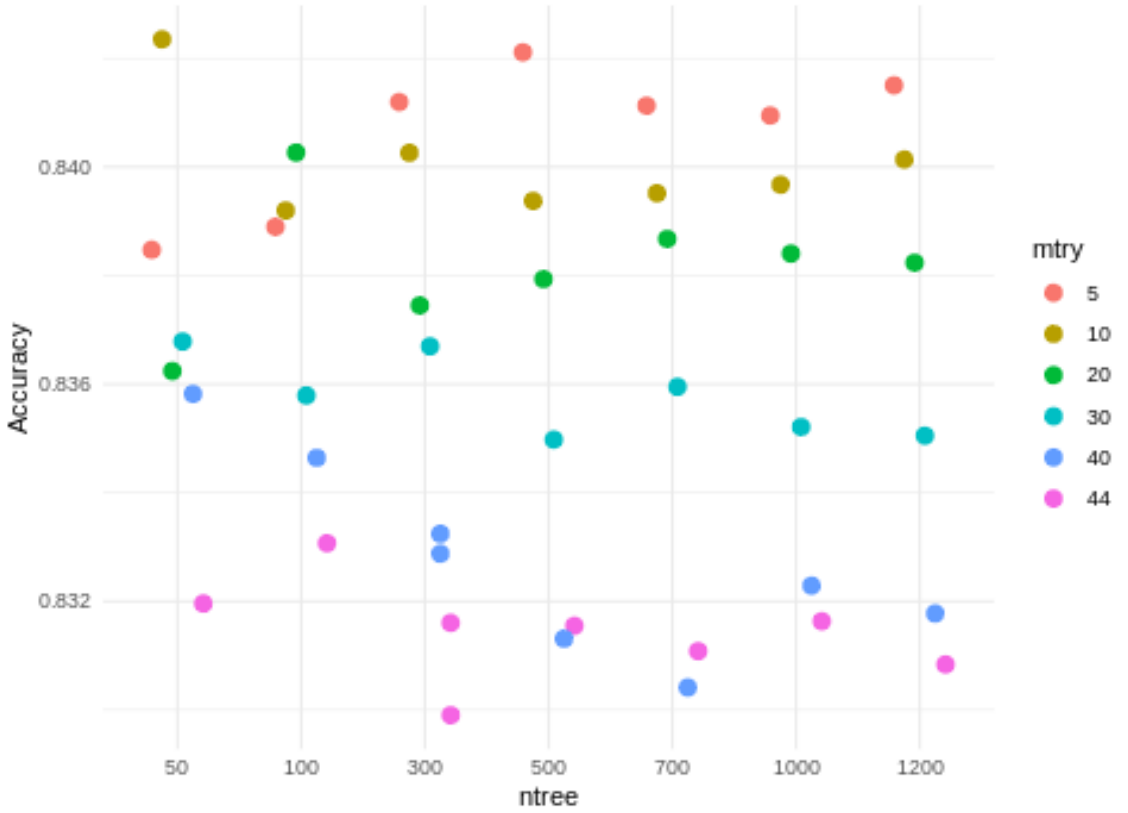


Gráfico 12: *grill* del modelos *Random Forest*.

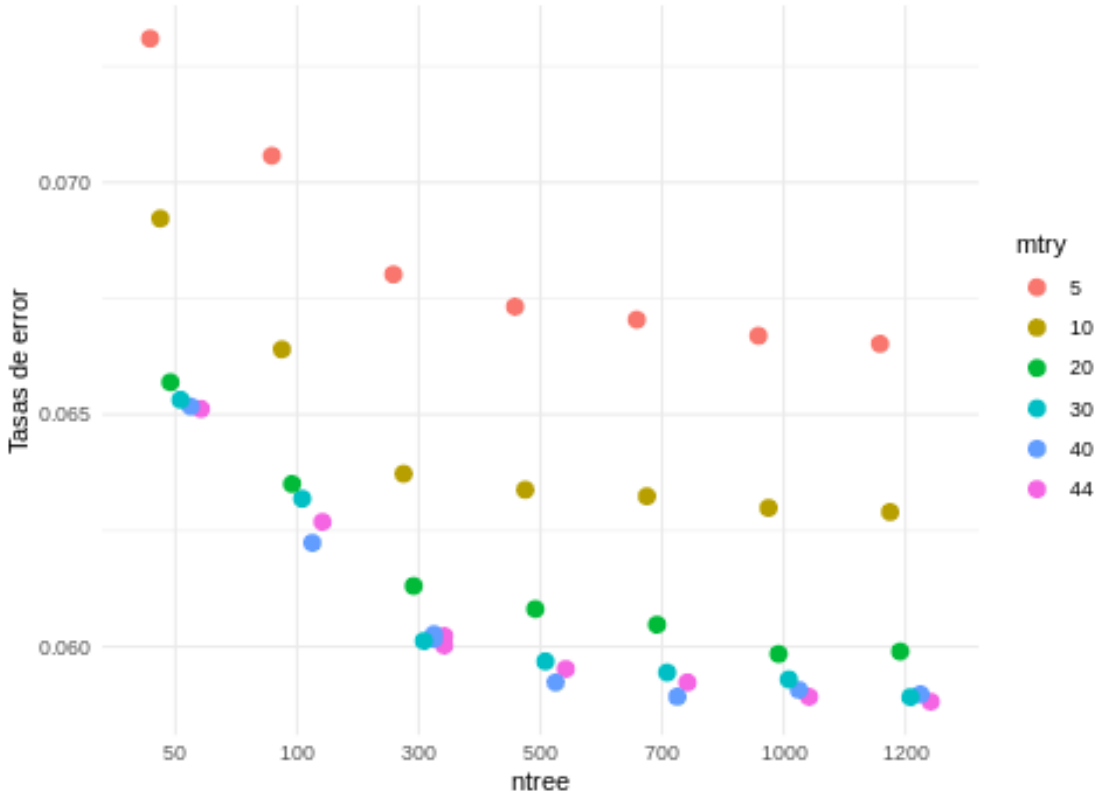


Gráfico 13: *grill* de la tasa de fallos del modelo *Random Forest* en los datos de entrenamiento.

Respecto a *mtry* óptimo no se pueden sacar conclusiones de los datos de entrenamiento del gráfico 13, ya que cuantas más variables mejor ajusta a datos que ya conoce, y, por lo tanto, se debe mirar al *grill* del gráfico 12, en el cual se ve que cuando se estabilizan los modelos los que tienen un *mtry* de 5, son mejores al resto.

Los 5 mejores modelos *Random Forest* según su *accuracy* son los que se muestran en la tabla 6. Nos quedaremos con el segundo mejor modelo. Ya que el primero tiene muy pocos árboles y es accuracy tan alto se da antes de estabilizarse.

Tabla 6: mejores 5 modelos del grill de *Random Forest.*

Modelo	ntree	mtry	nodesize	Accuracy	Sensitivity	Specificity	train_err
5	50	10	300	0.8423491	0.8606105	0.7506890	0.06921977
26	500	5	300	0.8421097	0.8614320	0.7451246	0.06732254
44	1200	5	300	0.8415016	0.8600007	0.7486484	0.06652125
20	300	5	300	0.8411940	0.8596214	0.7487010	0.06801722
32	700	5	300	0.8411259	0.8595983	0.7484064	0.06704280

El modelo *Random Forest* seleccionado tendrá un *mtry* de 5, 500 árboles y 300 de tamaño mínimo de hoja.

GBM

Para el *grill* de *Gradient Boosting machine* no se debe heredar el número de árboles del modelo RF puesto que este algoritmo, en vez de hacer combinaciones entre los resultados de los modelos, los utiliza para predecir el error del conjunto de modelos anteriores, en este caso un número excesivo de árboles sí lleva al sobreajuste. Aun así, el RF puede dar una idea de la magnitud de árboles adecuado para el grill. Se probará el valor por defecto de 100 árboles, así como 300 y 500. Además de distintos valores del parámetro de regularización (*Srinkage*).

Se realizará inicialmente un *grill* con pocos valores en a lo largo de un rango relativamente extenso, para luego aumentar la sección en la que parece encontrarse el modelo con *accuracy* máximo para cada modelo con distinto número de árboles. Al igual que el *Random Forest* se heredará el tamaño mínimo de hoja óptimo sin CP de 300 de los modelos de árbol. Los hiperparámetros que se probarán en el *grill* serán:

- Srinkage: 0.001, 0.01, 0.03, 0.05, 0.1, 0.2, 0.3
- Numero de árbol: 100, 300, 500

El gráfico 14 muestra que el modelo con 100 árboles supera tanto en su óptimo como en casi todos los puntos que no bajan al 83,3% de *accuracy* (modelo nulo) a los otros dos modelos, a excepción de un punto en el 0.001. Aun así, hay indicios de que hay un máximo local entre el 0.01 y el 0.001 para los modelos con 500 y 300 árboles, teniendo este el valor más alto del *grill* inicial. Mientras que el *srinkage* óptimo del modelo con 300 parece estar entre el 0.3 y el 0.01. Esto entra dentro de los resultados esperados ya que los modelos con más arboles pueden evolucionar más lentamente, pues tienen más intentos, mientras que los modelos con menos árboles mejoran con saltos más grandes. Se procederá a extender el *grill* de cada modelo en el rango donde se sospecha que hay máximos locales. No se extenderá el grill para valores más altos de 0.3 pues no hay ningún indicio de que se mejore en ese rango de valores.

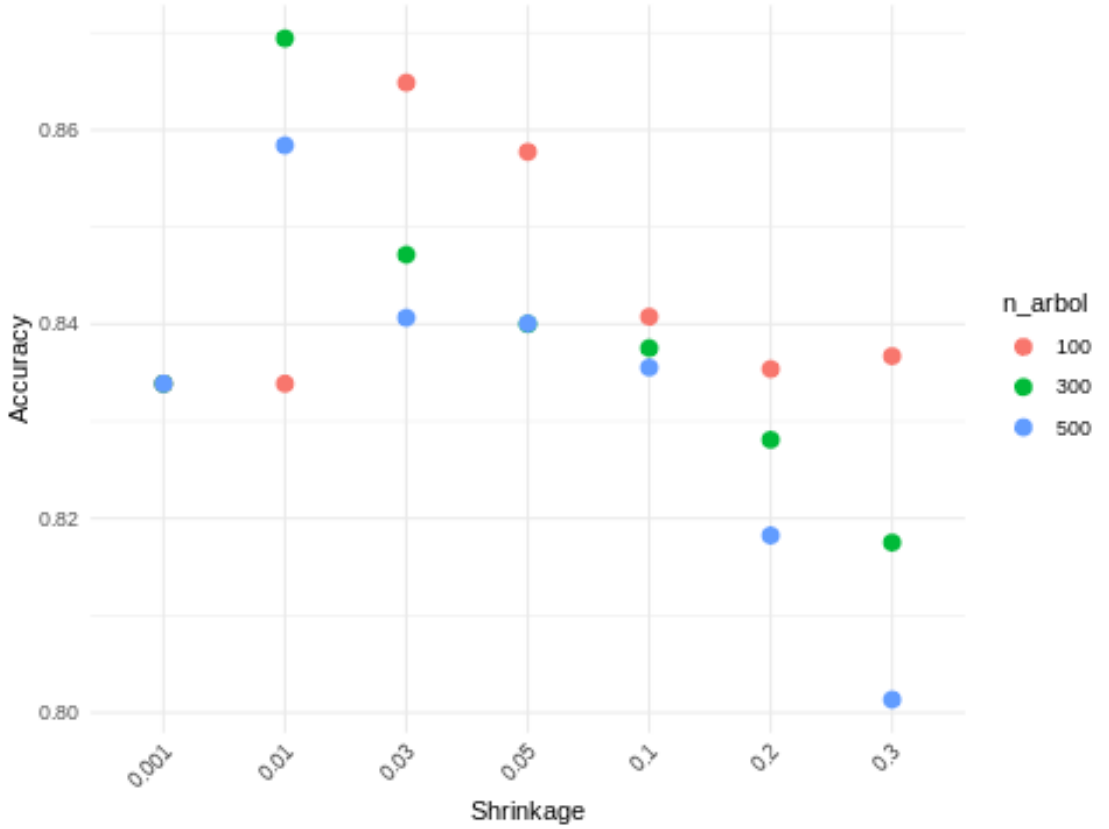


Gráfico 14: *grill* inicial para el modelo de *Gradient Boosting Machine.*

El gráfico 15 muestran el *grill* de valores extendidos, con los valores de C como factores. Y se aprecia que el *srinkage* óptimo de *gbm* para 100 árboles es de 0.023, siendo este más alto que el *accuracy* máximo de los modelos con 500 árboles y 300 respectivamente.

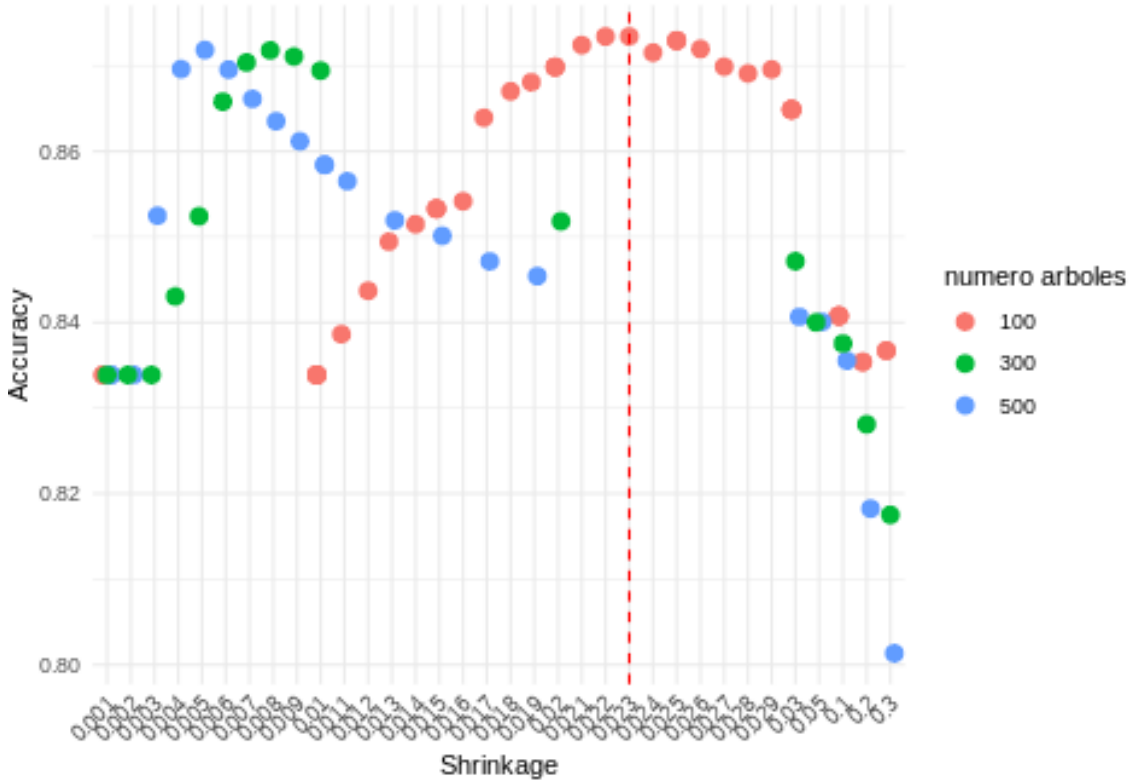


Gráfico 15: *grill* extendido para el modelo de *Gradient Boosting Machine.*

Los 5 mejores modelos se pueden ver en la tabla 7. Aunque la diferencia entre los modelos óptimos con distinto número de árboles es muy pequeña, menos de 3 centésimas, se tomará como modelo óptimo el que presenta mayor *accuracy*. Además, tiene la ventaja de que, al tener menos árboles, tardará menos tiempo de computación en dar la predicción. También es más robusto al no caer su *accuracy* tan drásticamente cuando se aleja ligeramente de su *srinkage* óptimo.

Tabla 7: mejores 5 modelos del grill de *Gradient Boosting Machine*

Modelo	**srinkage**	**Accuracy**	**Sensitivity**	**Specificity**	**n_arbol**
66	0.023	0.8734260	0.9804773	0.3360997	100
65	0.022	0.8734172	0.9849493	0.3136005	100
23	0.025	0.8729419	0.9714368	0.3785632	100
13	0.005	0.8718445	0.9702004	0.3781635	500
101	0.008	0.8717956	0.9730924	0.3633533	300

El modelos seleccionado de *GBM* es el 66 con un *srinkage* de 0.023, 100 árboles y un tamaño mínimo de hoja de 300.

XGB

Para el *grill* del algoritmo *Extreme Gradient Boosting Machine* (XGB) se mantendrá constante el parámetro de complejidad *gamma* en 0, ya que planeamos controlar el sobreajuste mediante el *min child weight.* La profundidad máxima de los arboles creados en 6, que es el valor que toma por defecto, y no se realizará *sub-sampling*, ni sorteo de variables, es decir, no se crearán los árboles del XGB aplicando las metodologías del *Bagging* o *Random Forest.* Sí se tunearán los siguientes parámetros:

- Peso de los hijos: 5, 10, 20, 50, 1000, 10000
- Eta: 0.001, 0.01, 0.03, 0.1, 0.2
- Numero de árboles: 100, 200, 300, 500, 700

En el gráfico 16 se ve que el *min child weight* apenas tiene efecto comparado con el *eta* o el número de rondas. Es decir, de los dos parámetros que controlan el sobre ajuste solo está afectando el *eta*.

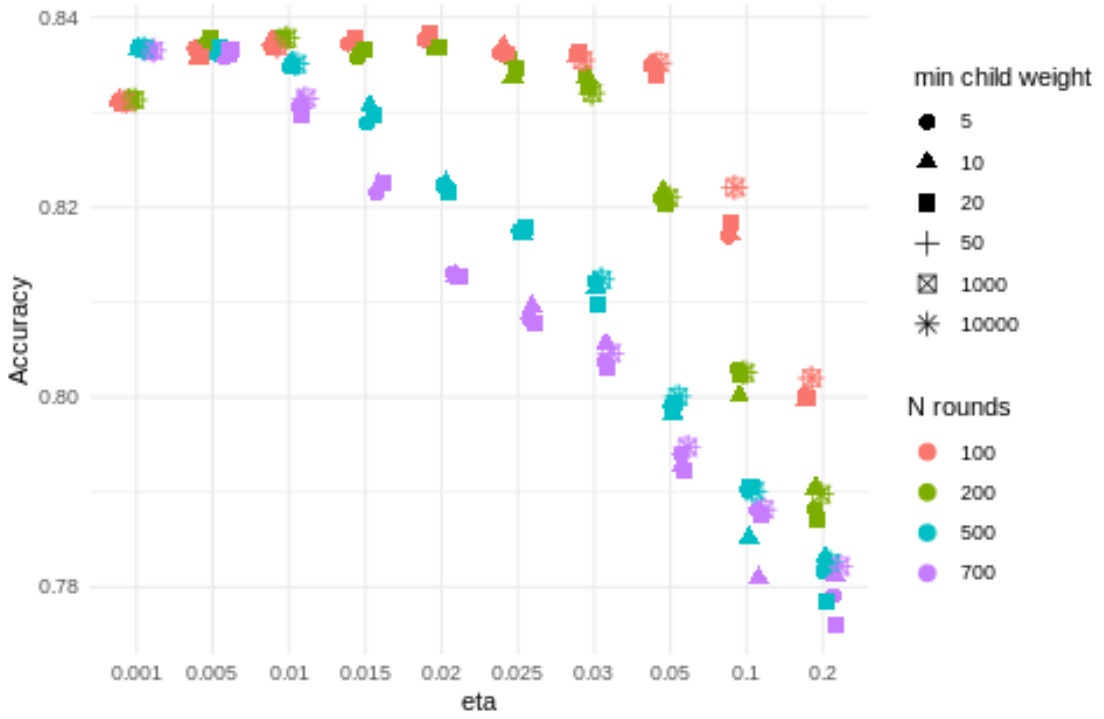


Gráfico 16: *grill* para el modelo *Extreme Gradient Boosting.*

Los mejores modelos resultantes son, de nuevo, muy parecidos entre sí, y, al contrario que en casos anteriores, es similar tanto en la sensibilidad como la especificidad. Estos se pueden ver en la tabla 8.

Se seleccionará como modelo optimo el modelo 187 ya que para esa combinación de número de árboles y *mind child weight* se obtienen modelos muy buenos, aunque el *eta* varie, es decir, es robusto al sobreajuste.

Tabla 8: mejores 5 modelos del grill de *Extreme Gradient Boosting*

Modelo	min_child_weight	eta	nrounds	Accuracy	Sensitivity	Specificity
187	20	0.020	100	0.8382338	0.8585798	0.7361102
183	20	0.015	100	0.8378039	0.8591939	0.7304407
19	5	0.010	200	0.8377602	0.8577165	0.7375934
92	5	0.010	200	0.8377602	0.8577165	0.7375934
67	20	0.010	200	0.8377148	0.8576620	0.7375934

El modelo seleccionado de XGB tendrá 20 de min child weight, un eta de 0.02 y 100 arboles

SVM

Para este algoritmo se probarán los Kernels polinómicos de grado 2 y lineales. Aunque en un trabajo real se deberían probar polinómicos de grado superior, que, por limitaciones en la capacidad de computación, no se han podido hacer.

Se comenzará tuneando el SVM lineal, del cual no se esperan muy buenos resultados, ya que los modelos anteriores han dado a entender que existen relaciones no lineales entre las variables. Se probarán valores para su único hiperparámetro, la penalización del error de clasificación C. Los valores del *grill* serán:

- C: 0.001, 0.005, 0.01, 0.05, 0.1, 0.2, 0.5, 1, 2

Se ve en el gráfico 17 como el C óptimo está en 0.01, aunque la diferencia entre todos los modelos está por debajo del 0.5% de *accuracy*.

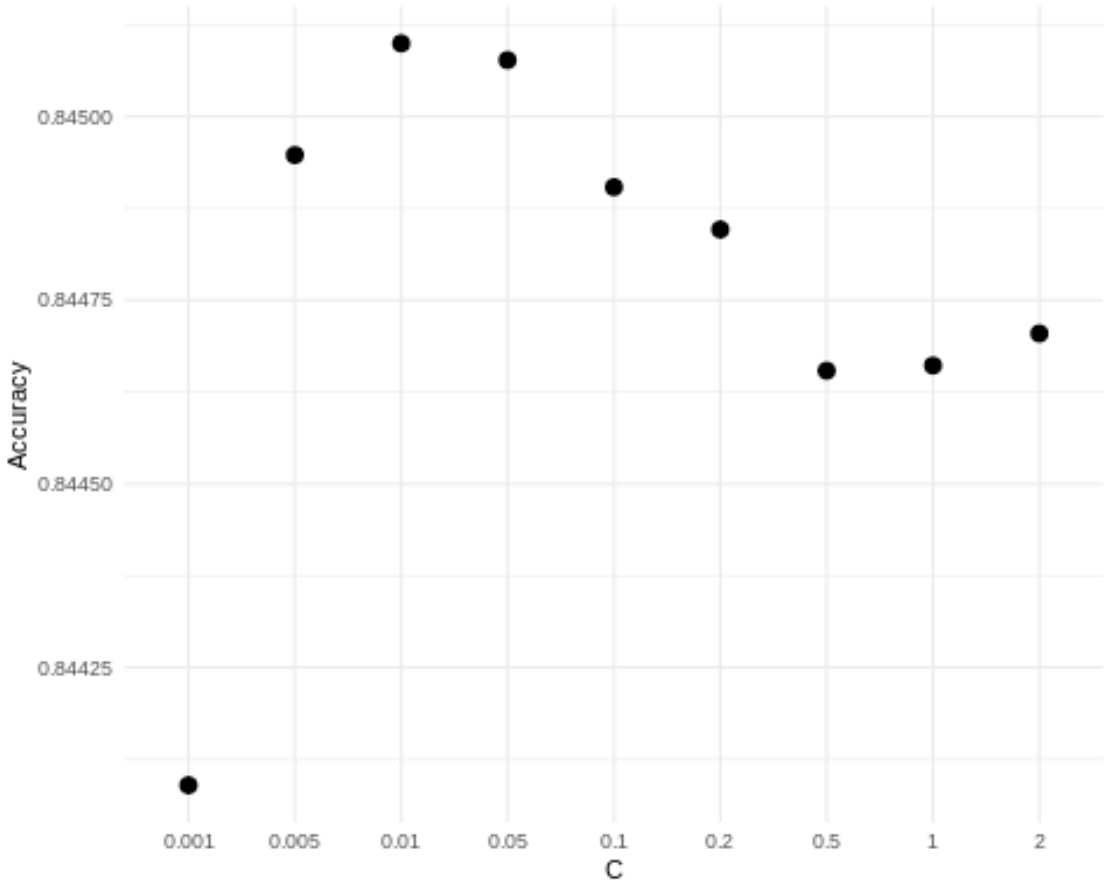


Gráfico 17: *grill* para el modelo *Suport Vector Machine* lineal.

Se podría considerar todos los modelos iguales y no particularmente buenos ya que apenas mejoran el modelo nulo en poco más de un 1.%, como se ve en la tabla 9 con los mejores modelos.

Tabla 9: mejores 5 modelos del grill de *Suport Vector Machine* lineal.

Modelo	**C**	**Accuracy**	**Sensitivity**	**Specificity**
1	0.010	0.8450996	0.9906159	0.1147050
2	0.050	0.8450769	0.9906829	0.1142316
8	0.005	0.8449476	0.9907898	0.1129168
3	0.100	0.8449039	0.9907982	0.1126118
4	0.200	0.8448462	0.9908485	0.1120122

Los resultados del SVM lineal confirman que las relaciones internas de los datos son demasiado complejas para tratarlas de forma lineal y por lo tanto se debe usar un Kernel polinómico.

En el *grill* del SVM polinomial de grado 2, se tunearán más hiperparámetros, como la penalización de errores C, al igual que el SVM lineal, y el *gamma* (*scale*) que regula la influencia de cada punto en la superficie controlando la complejidad del modelo. En un trabajo real también se podría tunear el grado del polinomio, pero por limitaciones de tiempo de computación, no se ha realizado. Los valores para probar en el *grill* son los siguientes:

- *Scale*: 0.005, 0.01, 0.05, 0.1, 0.5
- C: 0.001, 0.01, 0.05, 0.2, 0.5, 1, 2

Los resultados mostrados en el gráfico 18 indican que se ajusta mejor a los datos una superficie más regular para valores altos de penalización del error, puesto que de lo contrario el modelo sobreajusta mucho. Para modelos con el error menos penalizado se obtienen resultados mejores en general, pues aumenta la robustez sin comprometer precisión, ya que para escalas más altas se sobreajusta mucho.

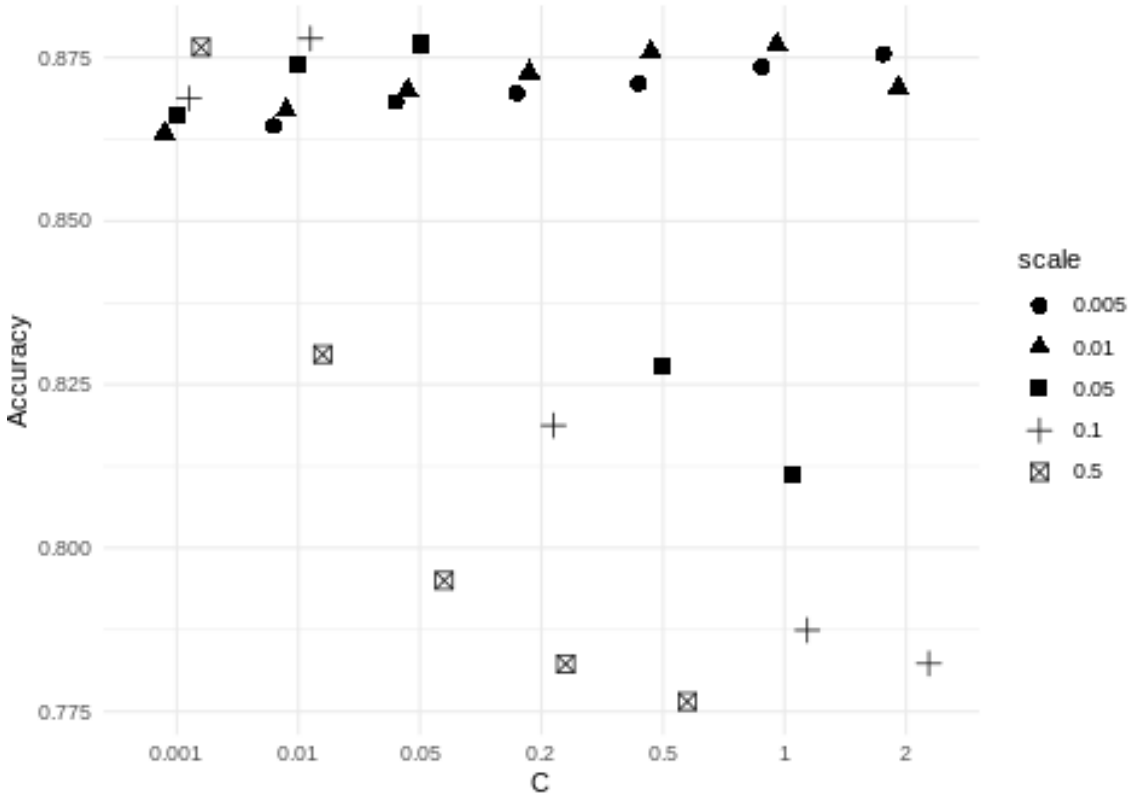


Gráfico 18: *grill* para el modelo *Suport Vector Machine* polinomial.

De los 5 mejores modelos, reflejado en la tabla 10, seleccionaremos el tercero. Puesto que entre estos 5 apenas hay diferencias en el accuracy mientras que si varia bastante

las especificidad. Mirando el gráfico 16 se ve que la escala 0.01 es menos dependiente del parámetro C y por lo tanto más robusta al sobreajuste.

Se tomará como mejor modelo de *SVM* el número 27 de la tabla 10. Aunque se podría argumentar seleccionar el modelo número 9, puesto que su C es más robusta, dando mejores resultados en general, a costa de que su escala sobreajuste mucho para todo C ligeramente mayor a su óptimo. Sin embargo, al no saber si es más probable que se modifique en los datos globales, la C o la escala, se selecciona el modelo 27 pues se asume menos riesgo.

Tabla 10: mejores 5 modelos del grill de *Suport Vector Machine* polinomial.

modelo	**C**	**scale**	**degree**	**Accuracy**	**Sensitivity**	**Specificity**
9	0.010	0.10	2	0.8778977	0.9635133	0.4481645
13	0.050	0.05	2	0.8770607	0.9583518	0.4690333
27	1.000	0.01	2	0.8770135	0.9644480	0.4381508
5	0.001	0.50	2	0.8765766	0.9535068	0.4904386
22	0.500	0.01	2	0.8758515	0.9760116	0.3731145

El modelo seleccionado de SVM tendrá un Kernel polinómico de grado 2, un parámetro de penalización del error C de 1 y una escala de 0.01

Ensamblado

Para el ensamblado se van a comparar los distintos modelos seleccionados durante la modelización anterior mediante validación cruzada de los datos *validate.* De esta forma se pone a prueba la robustez de los modelos frente a cambios en los datos de entrada y se puede tener una medida visual de la relación varianza/sesgo que tienen dichos modelos.

Al igual que la clasificación en grupos de entrenamiento, validación y testeo los grupos de validación cruzada se van a hacer por partidos y no por filas. Para esto se ha creado una función a medida como se ve en la imagen 6 que asigna a cada identificador único de partido un grupo mediante asignación sistemática y posteriormente agrupa todas las filas con ese identificador en su grupo de validación cruzada.

```r
cv_grupos<-function(datos_f,grupos=5){
  set.seed(11101)

  df_uniques <- datos_f %>%  # vector con el numero de identificadores unicos
    distinct(match_id)
  nrow(df_uniques)


  grcv<-rep(c(1:grupos), ceiling(nrow(df_uniques)/grupos)) # seleccion sistematica de los grupos

  if(length(grcv)>nrow(df_uniques)){ # comporvacion de que cada individuo tiene un grupo
    grcv<-grcv[1:nrow(df_uniques)]
  }
  df_uniques<-cbind(df_uniques,grcv)

  datos_f<-left_join(datos_f,df_uniques, by= "match_id" ) # juntar grupo con los datos

  return(datos_f)

}
```

Imagen 6: función para la creación de grupos de validación cruzada por partidos.

Los diagramas de caja de la validación cruzada por partidos, que se observan en el gráfico 19, muestran 2 grupos de modelos. Los que apenas mejoran el modelo nulo y flotan alrededor de un *accuracy* del 84,5%, como son la regresión logística, diagrama de árboles, Random forest, XGB; y los que si consiguen mejorar significativamente al modelo nulo y por lo tanto predecir con más éxito el arbitraje con un *accuracy* alrededor del 88.0% (*GBM, redes neuronales, Suport Vector Machine*).

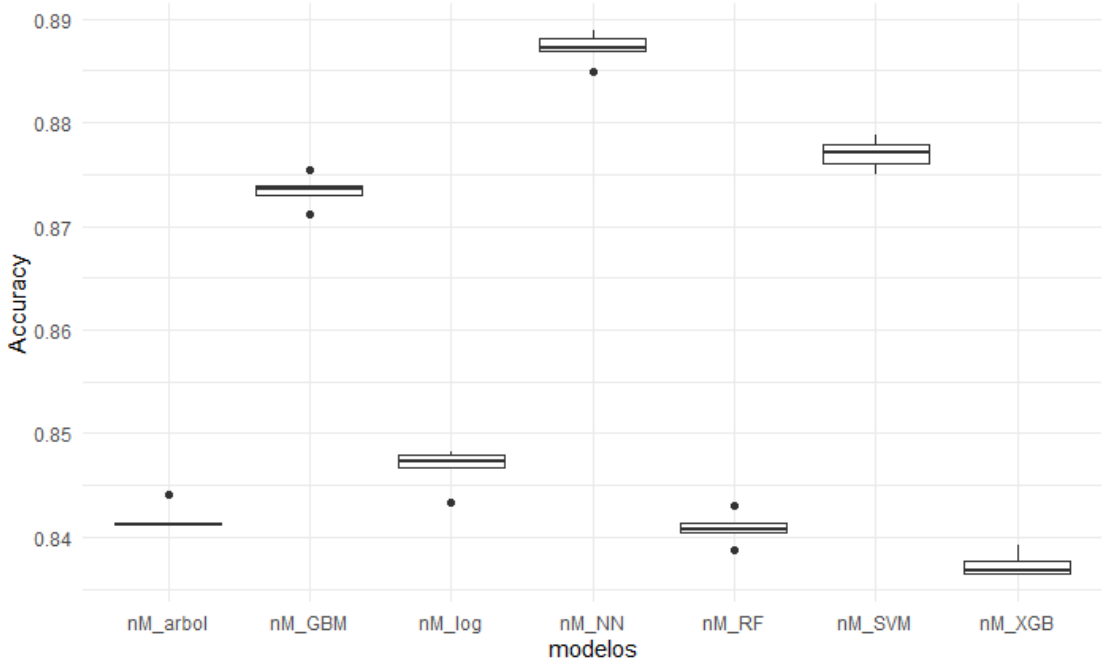


Gráfico 19: comparación de precisión y varianza de los modelos individuales seleccionados durante al modelización.

Pese a que hay claramente modelos mejores que otros en el ensamblado no hay una forma consistente de saber si los modelos perores ayudaran. Esto puede darse si en las predicciones en las que el modelo más potente dude más el modelo "peor" sea capaz de discriminar bien, sin perjudicar en el resto de las predicciones.

Por esto se prueban todas las combinaciones posibles de los modelos individuales seleccionados anteriormente. . Se muestra el resultado de probar los modelos en VC de datos validate. Los ensamblados por media en el gráfico 20, y el resultado de ensamblado por mediana en el gráfico 21.

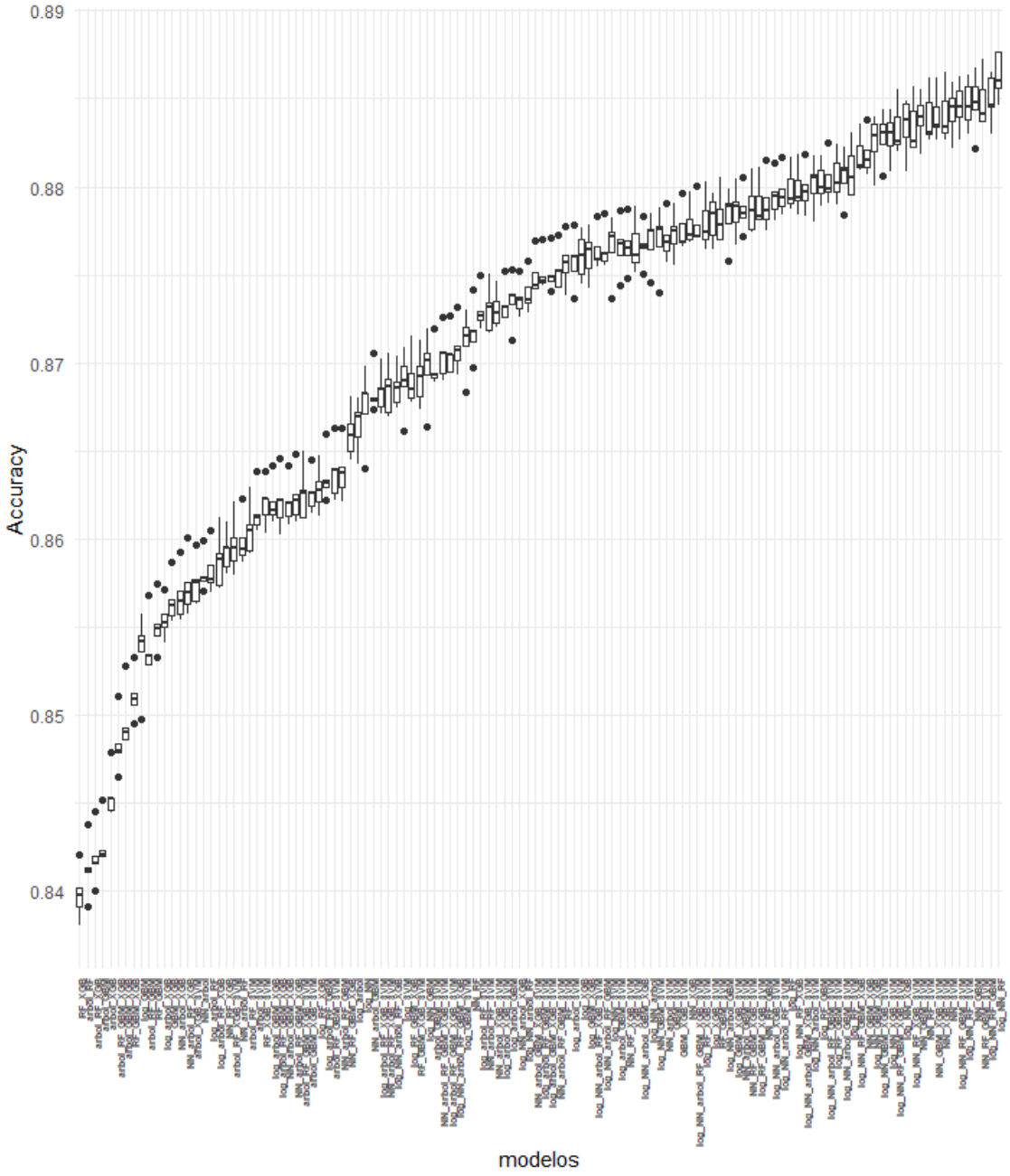


Gráfico 20: comparación de precisión y varianza entre todos los modelos de ensamblado por media posibles.

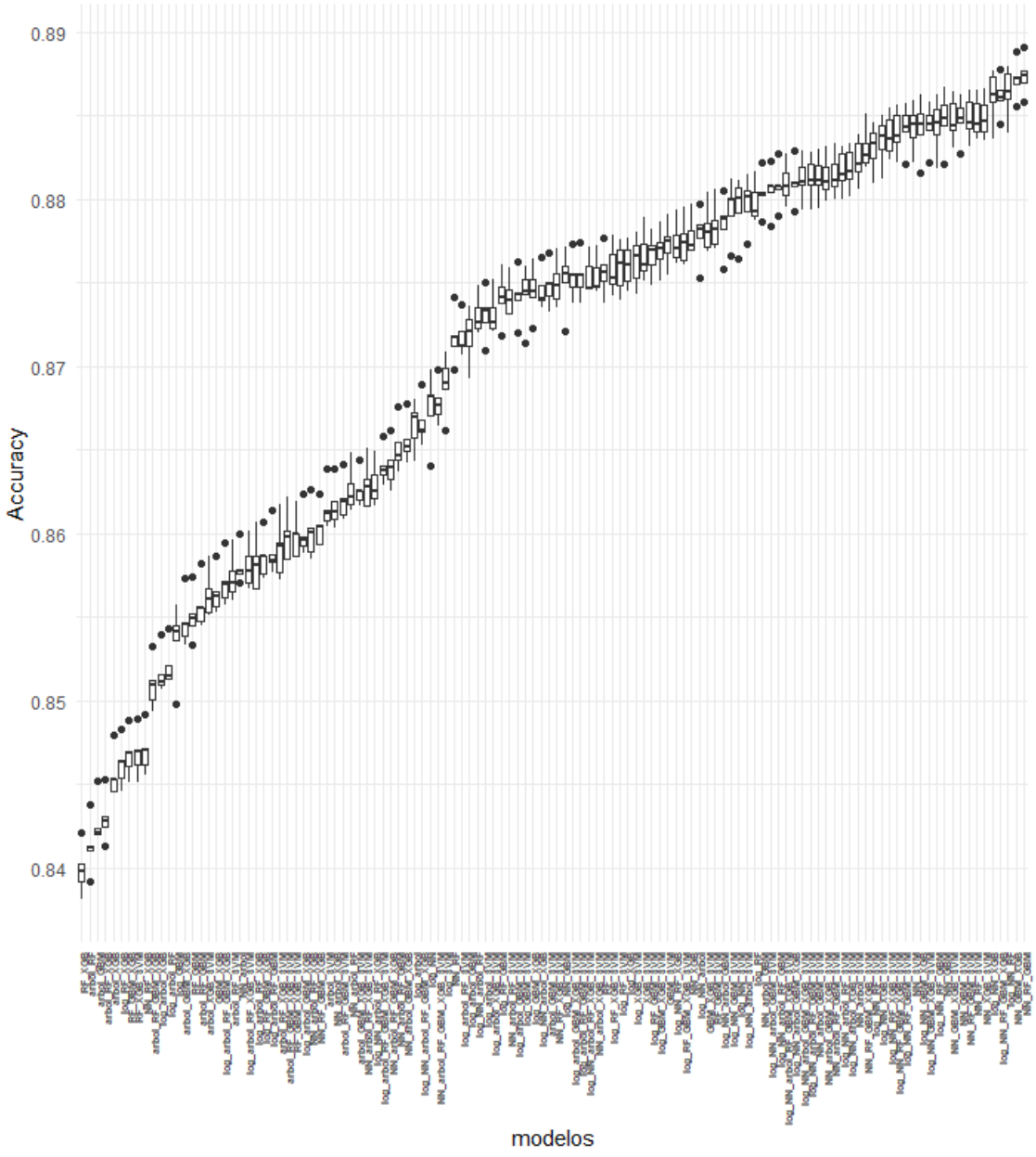


Gráfico 21: : comparación de precisión y varianza entre todos los modelos de ensamblado por mediana posibles.

Finalmente se comparan los 5 mejores modelos de ensamblado creados usando la media y la mediana, observándose en el gráfico 22, que los mejores 5 modelos de cada tipología se crean a partir de las mismas combinaciones de modelos individuales. De estos vemos que hay 3 en los que el ensamblado por media es mejor que el ensamblado por mediana. Así que para cada ensamblado se coge el tipo que mejor resultado de para compararlos con los modelos individuales.

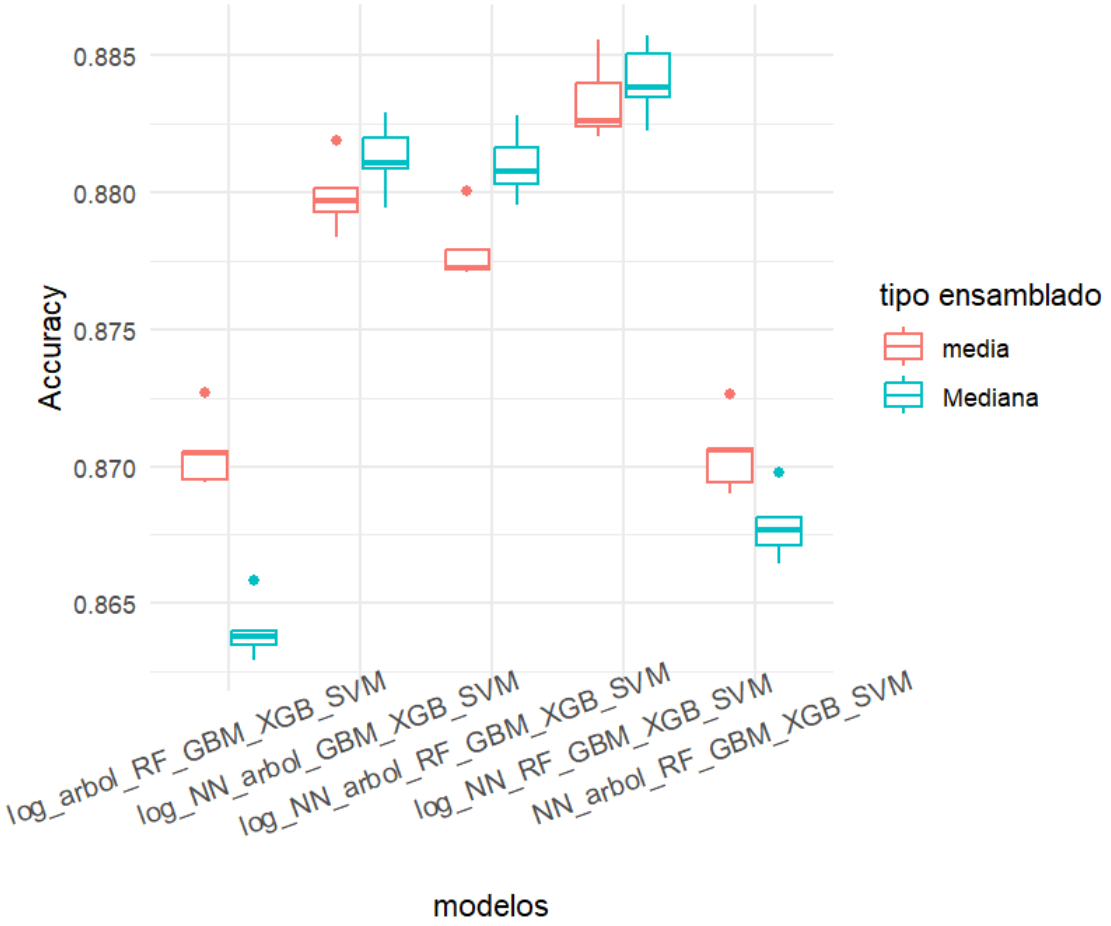


Gráfico 22: comparación de precisión y varianza entre los 5 mejores modelos de ensamblado por media y por mediana.

Los 5 mejores modelos de ensamblado se pueden ver en la tabla 11, con los estadísticos de bondad de ajuste en la predicción de todo el conjunto *validate*.

Tabla 11: mejores 5 modelos de ensamblado.

modelo	**método**	**Accuracy**	**Sensitivity**	**Specificity**
log_NN_RF_GBM_XGB_SVM	Mediana	0.88443136	0.9643144	0.48050378
log_NN_RF_GBM_XGB_SVM	Media	0.8832974	0.9554878	0.5209764
log_NN_arbol_GBM_XGB_SVM	Mediana	0.88123754	0.9700707	0.43536846
log_NN_arbol_RF_GBM_XGB_SVM	Mediana	0.88058246	0.9599064	0.48564512
log_NN_arbol_GBM_XGB_SVM	Media	0.8798567	0.9648990	0.4530217

Se seleccionará, como el mejor modelo de ensamblado, el que combina las probabilidades de predicción de la regresión logística, las redes neuronales, *Gradient Boosting machine*, *Extreme Gradient Boosting*, y *Suport Sector Machine* mediante y utiliza la mediana de sus valores.

Selección final de algoritmo

Para el *grill* comparativo final, representado en el grafico 23, se utilizará el conjunto de datos test para simular datos nuevos que no se han usado ni en la creación de modelos, castigando así a los modelos que sobreajustan, ni en la selección de los algoritmos, puesto que se han seleccionado los algoritmos principalmente por su eficacia al predecir el conjunto de datos *validate*. Podría ocurrir que los hiperparámetros tuneados en algún modelo no fuesen consistentes para otros conjuntos de datos, aunque con el gran volumen de datos esto es poco probable.

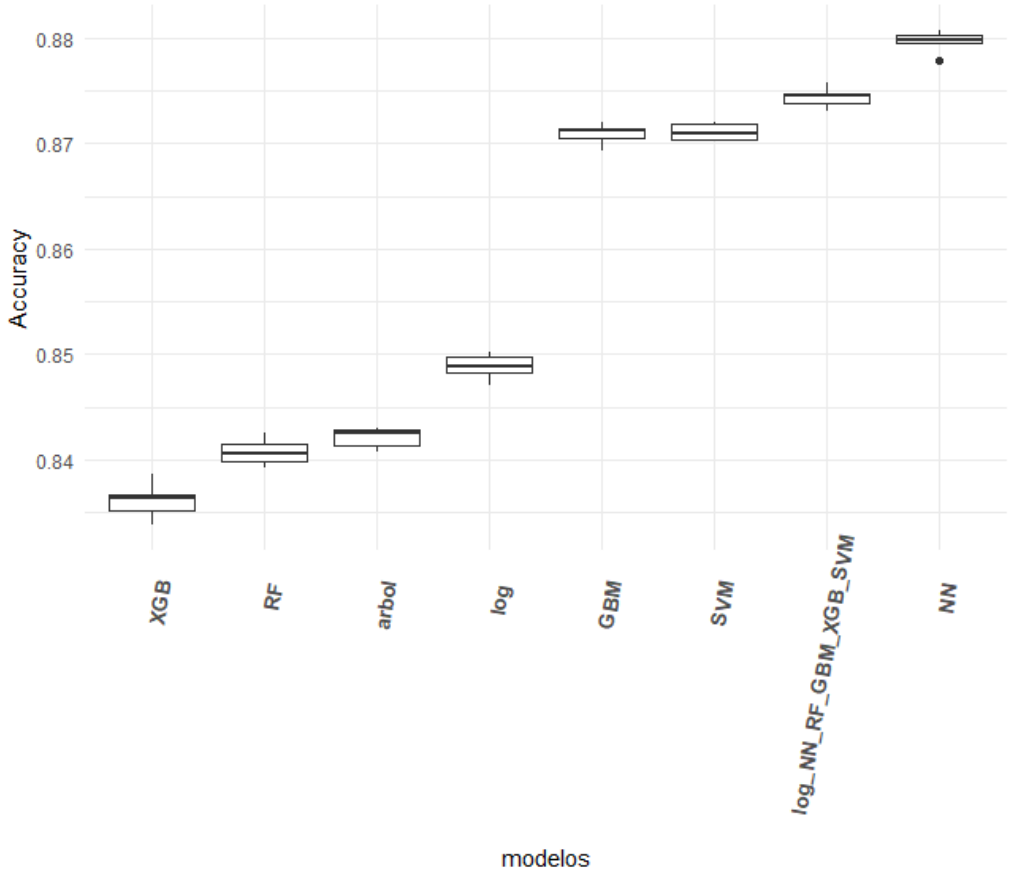


Gráfico 23: comparación de precisión y varianza entre los algoritmos seleccionados en la etapa de modelización sobre los datos *test.*

En la comparación de modelos mediante CV de los datos *test* se observa una reducción en el *accuracy* de todos los modelos entre un 0.5% y un 1%. Se deduce que el mejor modelo de todos los evaluados es de Redes neuronales. Como se muestra en la tabla 12.

Tabla 12: Selección del mejor modelo final

Modelo	Accuracy	Sensitivity	Specificity
Redes Neuronales	0.8796303	0.9525215	0.5196742
log_NN_RF_GBM_XGB_SVM	0.8743822	0.9435437	0.5328311
GBM	0.8708730	0.9811813	0.3261283
SVM	0.8710880	0.9496109	0.4833055
logística	0.8488060	0.9895473	0.1537779
Arboles	0.8420970	0.9194588	0.4600597
RF	0.8407741	0.8742601	0.6754098
XGB	0.8361237	0.8719125	0.6593912

Explicabilidad del modelo

Pese a que el modelo final de Redes Neuronales es considerado de caja negra existen técnicas, descritas en la metodología, para poder interpretar el efecto de las variables predictoras en la predicción final.

Empezaremos con el grafico *Permutaion feature importance* ,descrito en la metodología, que se observa en el grafico 24. Se distinguen 5 variables con mucha relevancia. El máximo, mínimo y la media de los ratios para el partido visitante. El rango máximo del equipo visitante dentro de cada hora y el mínimo del equipo visitante.

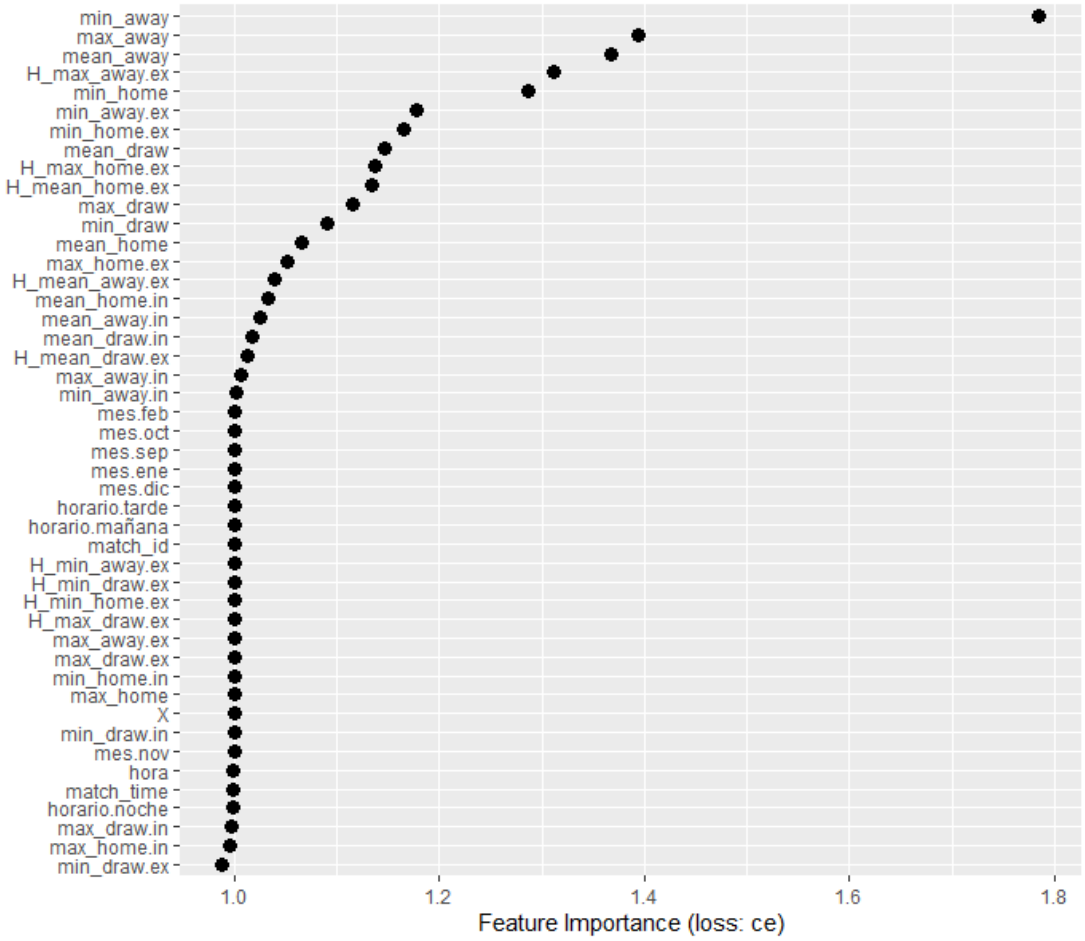


Gráfico 24: *Permutaion feature importance* sobre el modelo óptimo de Redes Neuronales

Es coherente que las 3 variables más relevantes hagan referencia al mínimo, máximo y media de un solo tipo de resultado ya que hay una fuerte correlación inversa entre las cuotas a *home*, *draw* y *away*. También se realiza un estudio más detallado del impacto de las 5 variable cuyas permutaciones más afectan a la probabilidad de que haya arbitraje. Este se puede ver en el gráfico 25, 26, 27, 28, 29. A medida que aumenta la media de la cuota mínima estandarizada aumenta rápidamente la probabilidad de arbitraje siguiendo una crecimiento más o menos logarítmico.

Respecto de las variables máximo y media de las cuotas al equipo visitante se ve lo contrario. Esto podría deberse q que si los equipos están muy igualados se dan situación con ratios para los resultados no favoritos tan bajos que no se da arbitraje.

Este efecto se ve en la variable mínimo de los ratios a home. A medida que crece dentro de un rango, aumenta la probabilidad de arbitraje, hasta cierto punto, a partir del cual lo que hace es bajar la probabilidad.

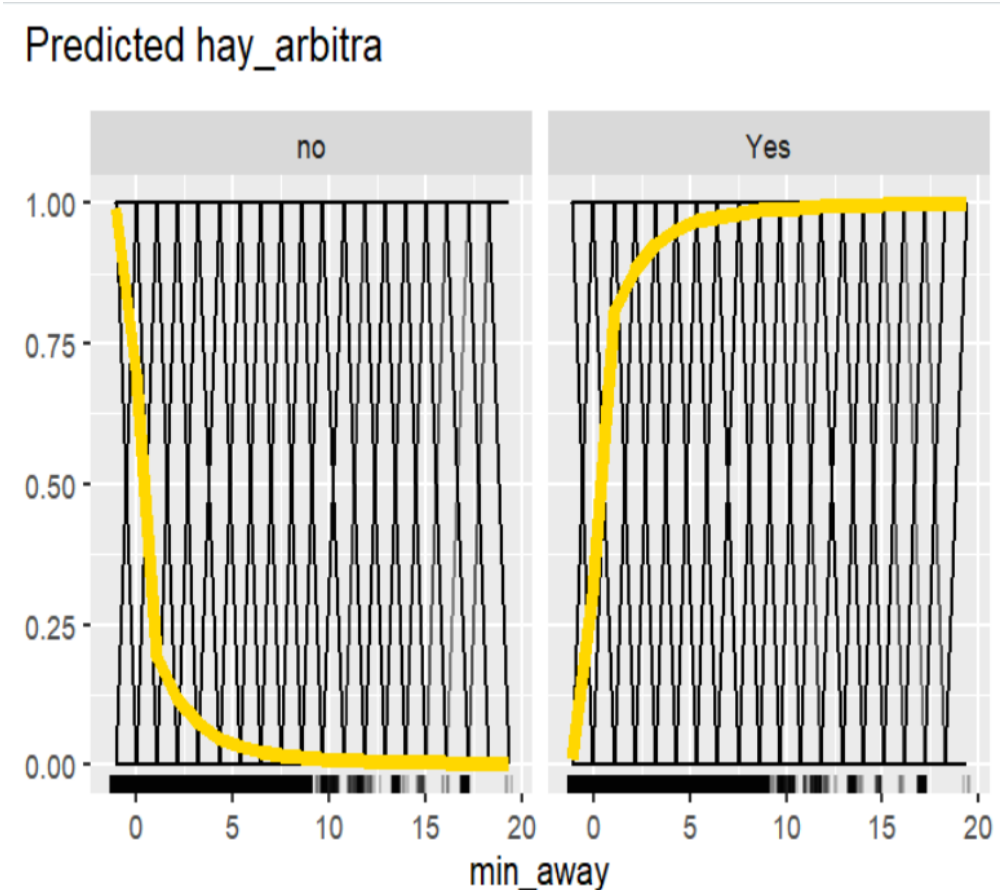


Gráfico 25: diagrama *PDP, ICE* del mínimo de las cuotas del equipo visitante entre las diferentes casas de apuestas (CA) en el periodo temporal (PT).

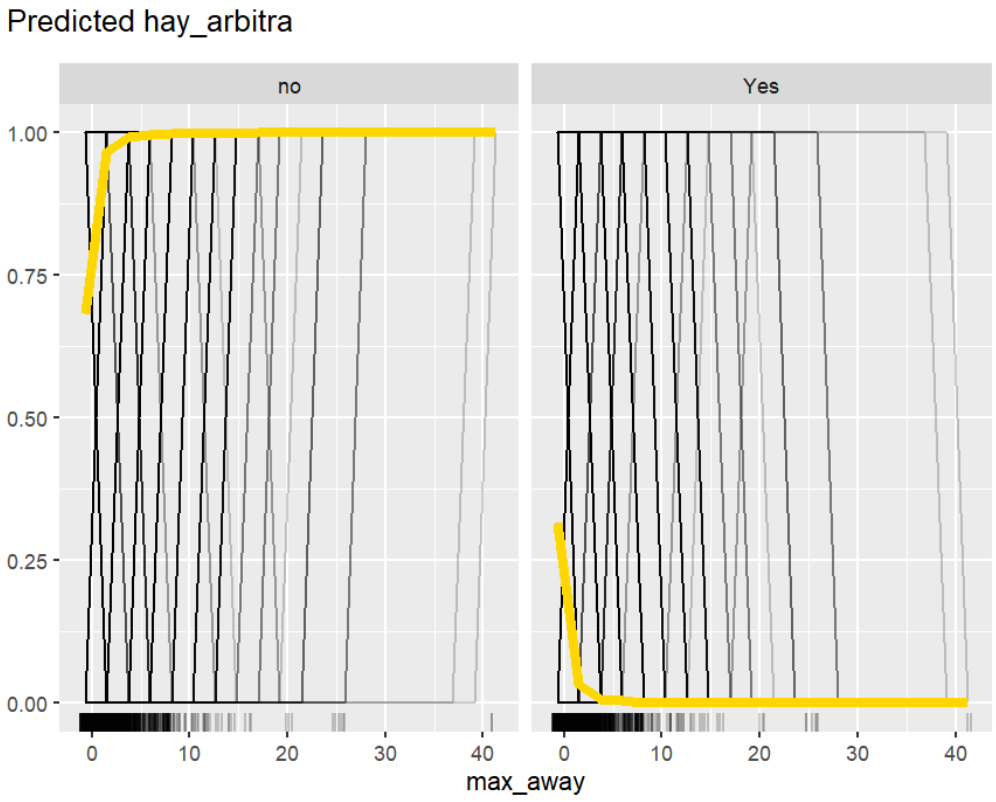


Gráfico 26: diagrama *PDP, ICE* del máximo de las cuotas del equipo visitante entre las CA en el PT.

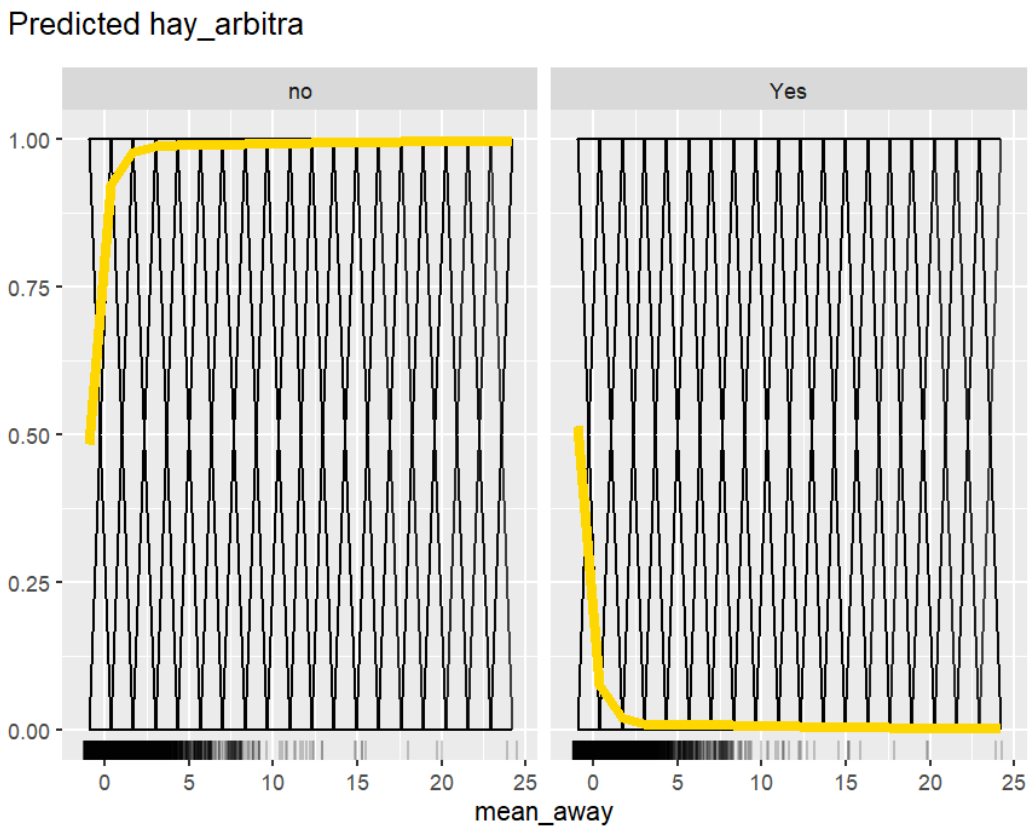


Gráfico 27: diagrama *PDP, ICE* de la media de las cuotas del equipo visitante entre las CA en el PT.

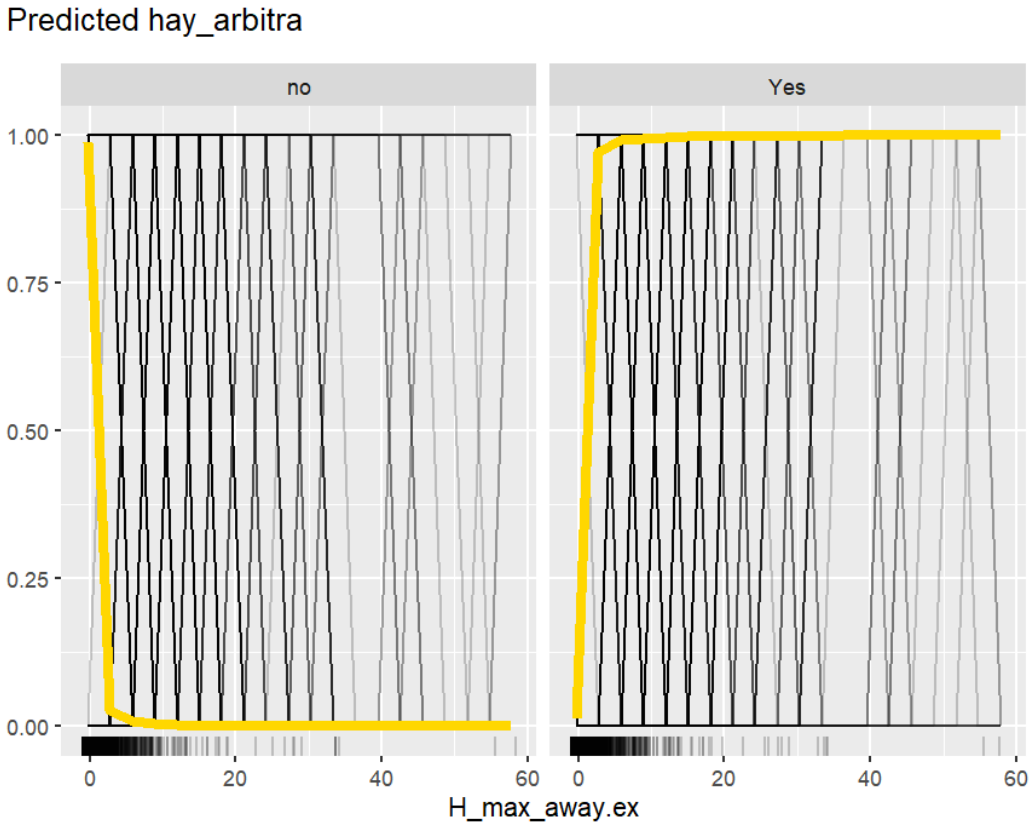


Gráfico 28: diagrama *PDP, ICE* del máximo de los rangos entre las CA de cada hora.

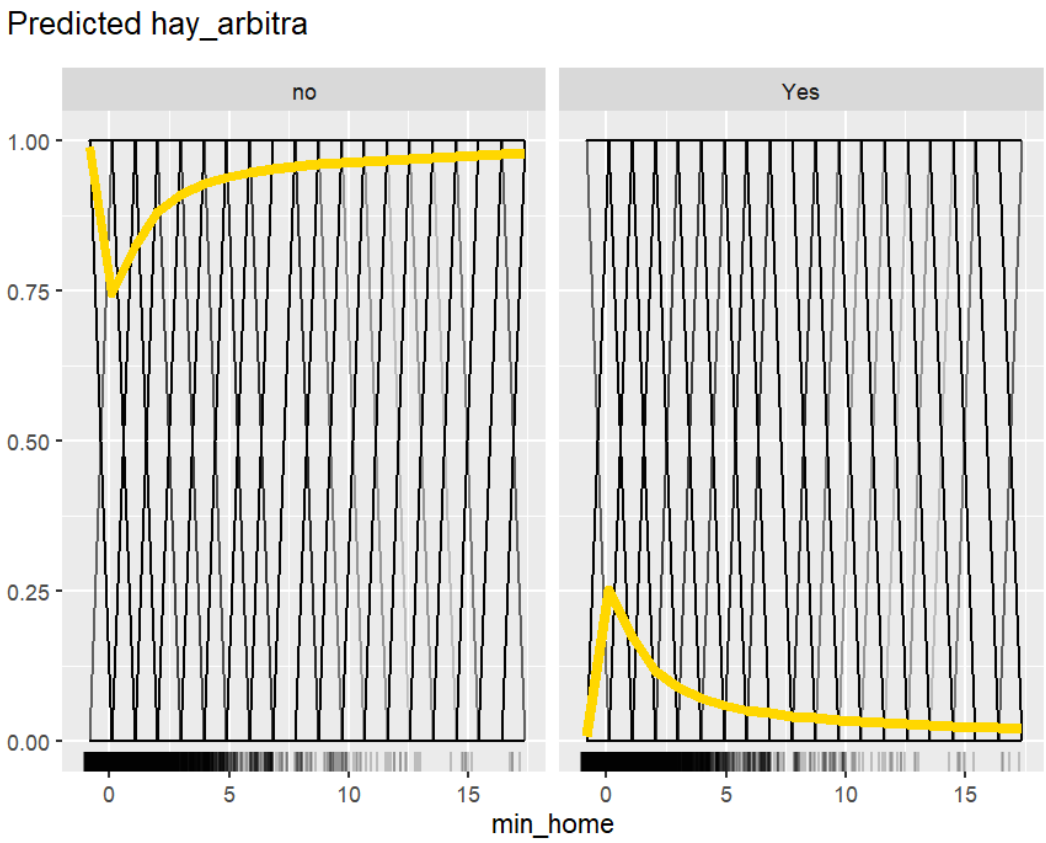


Gráfico 29: diagrama *PDP, ICE* del máximo de las cuotas del equipo local entre las CA en el PT.

Conclusiones

En este TFM se ha conseguido crear un modelo capaz de predecir con un 88% de precisión si se dará una situación de arbitraje durante la siguiente hora mediante la arquitectura de Redes Neuronales en cualquier hora dentro de las 67 antes del partido.

Se ha concluido que la presencia de arbitraje es relativamente frecuente en el mercado como combinación de apuestas entre casas de apuestas siendo una 16.71% en media, aunque varía dependiendo de la etapa del mercado en la que nos encontremos.

Se ha podido observar las 3 etapas de evolución del mercado, siendo las etapas: la inicial, la de estabilización y de fuerte crecimiento, con alrededor de un 20%, 12% y 25%-35% de partidos con arbitraje, así como la distribución de las situaciones de arbitraje a lo largo del tiempo, siendo esta lineal exceptuando los últimos momentos en los crece más rápidamente. Este estudio ha reflejado que, aunque se podría centrar un modelo en alguna de las 3 etapas, esto limitaría el porcentaje de arbitrajes totales en el periodo temporal que se podrían detectar, perdiéndose un 80% de los arbitrajes si solo se predice en la etapa de fuerte crecimiento. y, por lo tanto, es eficiente desarrolla run modelo general que funcione en todo el intervalo temporal.

Se ha realizado, mediante la creación de funciones a medida en R, un muestreo que mantiene la integridad y representatividad de los datos globales en los grupos *train*, *validate* y *test*.

Respecto de las variables independientes propuestas se ha determinado la utilidad de los datos atípicos por su relación con la presencia de arbitraje. Y se ha conseguido detectar la presencia de multicolinealidad en las variables predictoras y su eliminación mediante la comparación de algoritmos de selección de variables.

Se ha concluido que la mejor arquitectura para obtener el modelo del objetivo principal son las redes neuronales con 200 iteraciones, un decay de 0.01 y 10 neuronas. Tras conseguir tunear y comparar distintas metodologías de *machine learning.*

Se ha conseguido aportar explicabilidad del modelo midiendo el impacto que tienen las variables independientes sobre la dependiente, así como la forma de ese impacto para la variable más influyentes.

Este trabajo ha podido definir una metodología de predicción de arbitraje que puede ayudar a los participantes en detección de arbitraje a detectar situaciones antes que sus competidores y por un coste menor, independientemente de la saturación del mercado. Además, el modelo de redes neuronales resultante tiene la ventaja de producir predicciones muy raptadamente lo que puede dar una ventaja competitiva frente a algoritmos más lentos como podría ser *SVM.* De forma que se pueden obtener una predicción de arbitraje/no arbitraje para todos los partidos del mercado aumentando rápidamente, permitiendo poner el foco en partidos propensos a arbitrare antes que la competencia.

Bibliografía

Por orden de aparición

1. Franck, E., Nuesch, S., & Verbeek, E. (2009). *Inter-market arbitrage in sports betting* (No. 48). National Centre for Econometric Research.

2. Torres–Cabrera, G. P. S., & González, C. Q. (2018). Data mining y análisis matemático de las cuotas de las casas de apuestas deportivas online. *Rect@: Revista Electrónica de Comunicaciones y Trabajos de ASEPUMA*, *19*(2), 111-122.

3. Guerrero, J. A. (s/f). *El problema de la dimensionalidad*. Revistaindice.com. Recuperado el 26 de julio de 2024, de http://www.revistaindice.com/numero68/p22.pdf

4. Tsagris, M., & Tsamardinos, I. (2018). Feature selection with the R package MXM. *F1000Research*, *7*.

5. Hyndman, R. J., Koehler, A., Ord, J. K., & Snyder, R. D. (2008). *Forecasting with exponential smoothing: The state space approach* (2008a ed.). Springer.

6. Chen, D.-G., & Chen, J. K. (2021). *Statistical regression modeling with R : longitudinal and multi-level modeling*. Springer. https://doi.org/10.1007/978-3-030-67583-7

7. Helias, M., & Dahmen, D. (2020). *Statistical field theory for neural networks*. Springer. https://doi.org/10.1007/978-3-030-46444-8

8. Kingsford, C., & Salzberg, S. L. (2008). What are decision trees?. *Nature biotechnology*, *26*(9), 1011-1013.

9. Genuer, R., & Poggi, J.-M. (2020). *Random forests with R*. Springer. https://doi.org/10.1007/978-3-030-56485-8

10. Bühlmann, P., & Hothorn, T. (2007). Boosting algorithms: Regularization, prediction and model fitting.

11. Anghel, A., Papandreou, N., Parnell, T., De Palma, A., & Pozidis, H. (2018). Benchmarking and optimization of gradient boosting decision tree algorithms. *arXiv preprint arXiv:1809.04559*.

12. Mammone, A., Turchi, M., & Cristianini, N. (2009). Support vector machines. *Wiley Interdisciplinary Reviews: Computational Statistics*, *1*(3), 283-289.

13. Schölkopf, B. (2000). The kernel trick for distances. *Advances in neural information processing systems*, *13*.

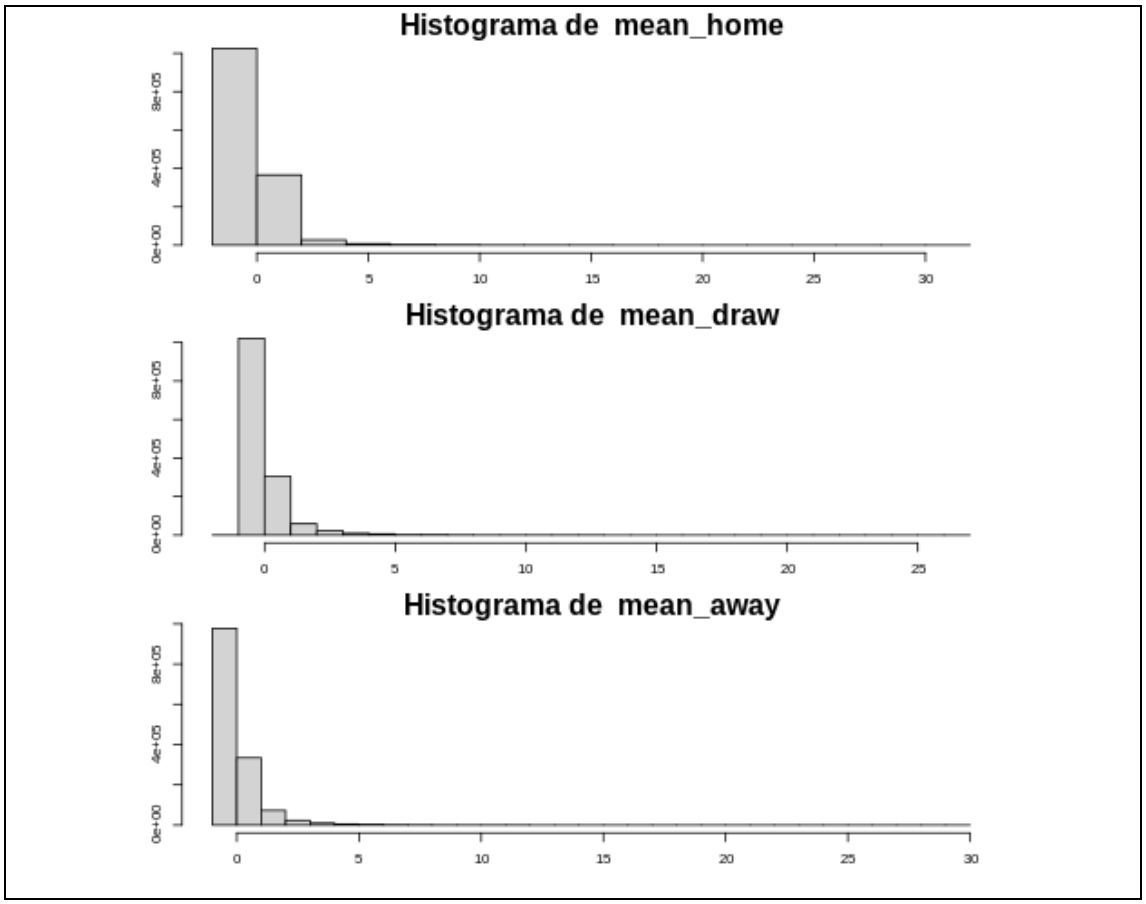
14. Fernández, R. (2021). Métodos de ensamblado en Machine Learning. *Eamo. Usc. Es*.

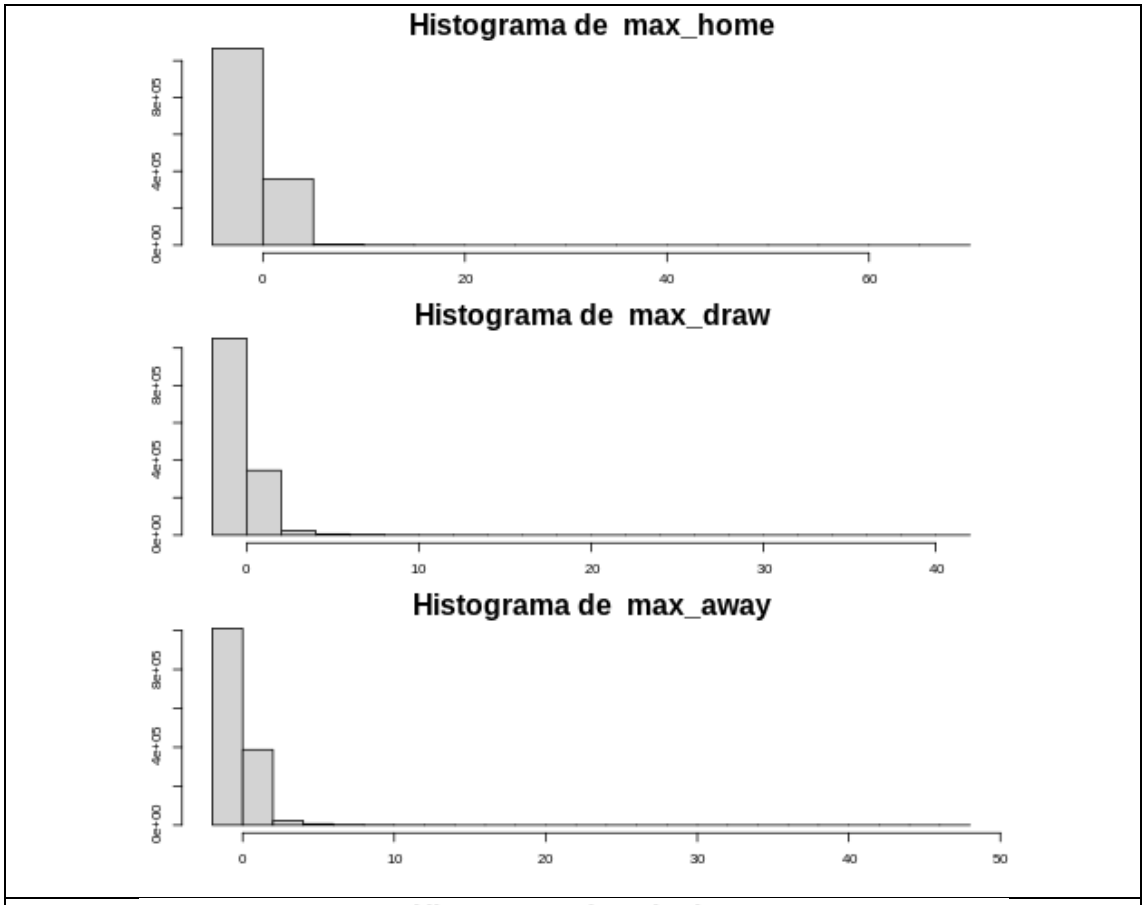
15. *Chapter 58 introduction to XAI (explainable AI) in R*. (s/f). Github.Io. Recuperado el 27 de julio de 2024, de https://jtr13.github.io/cc21fall2/introduction-to-xai-explainable-ai-in-r.html

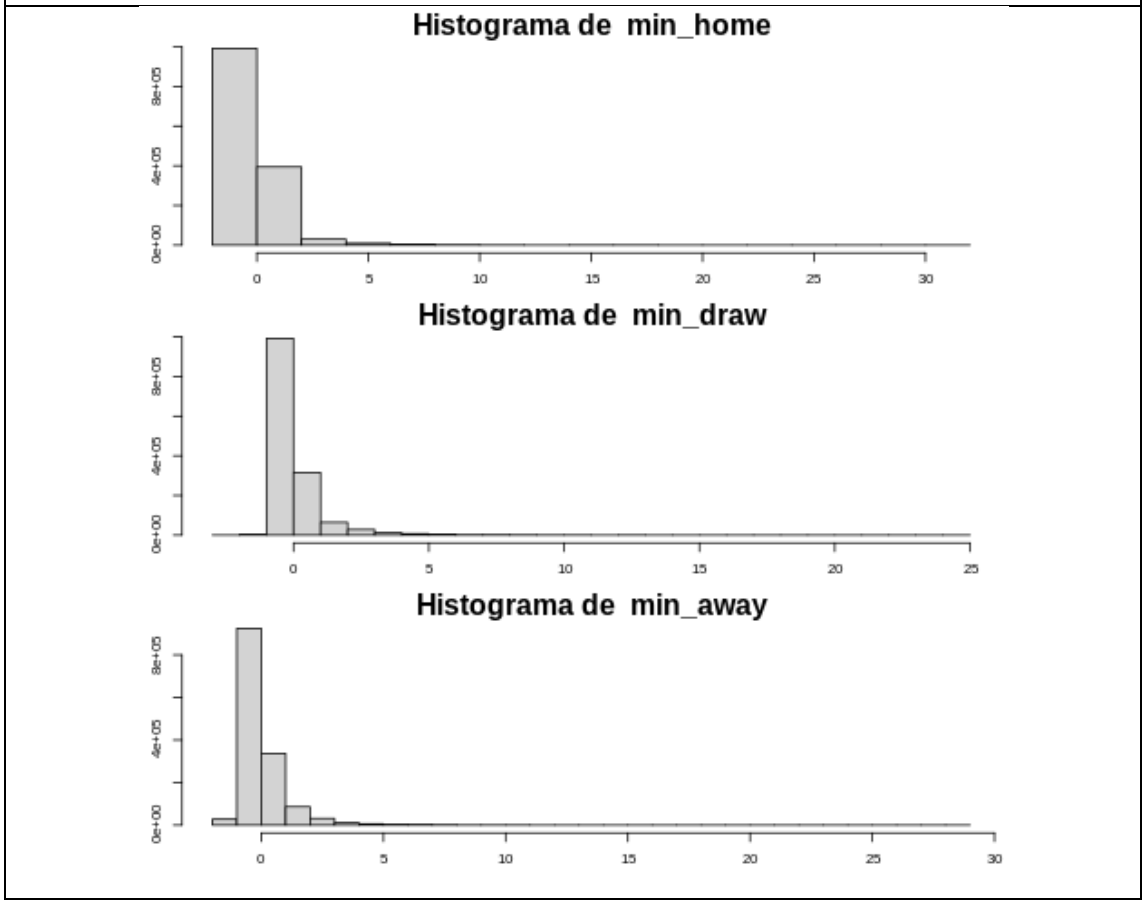
16. Kuhn, M. (2019, marzo 27). *The caret package*. Github.io. https://topepo.github.io/caret/index.html

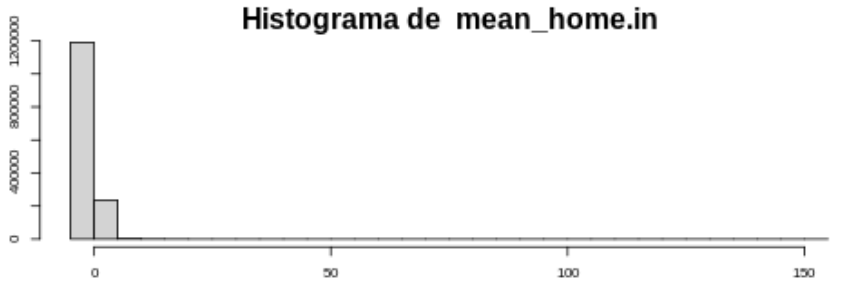
Anexo

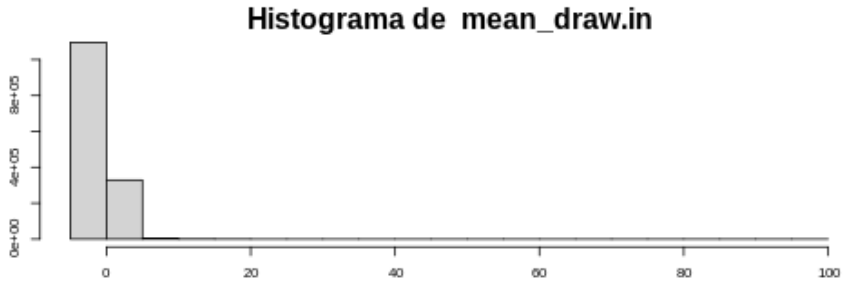
En este anexo se muestran los histogramas del análisis exploratorio sobre las variables numéricas de la base de datos final. En ellos se ve una masa grande de valores centrados en el 0 así como una cola de atípicos en una sola dirección.

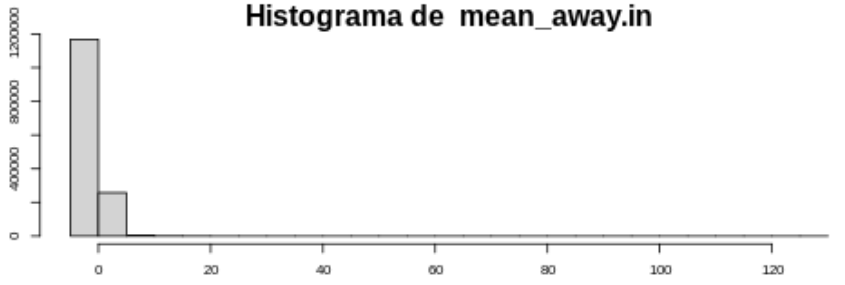


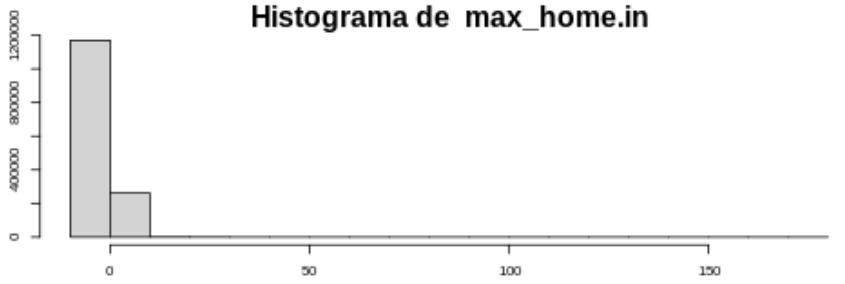

Histograma de max_home
0e+00
4e+05
8e+05
0
20
40
60
Histograma de max_draw
0e+00
4e+05
8e+05
0
10
20
30
40
Histograma de max_away
0e+00
4e+05
8e+05
0
10
20
30
40
50

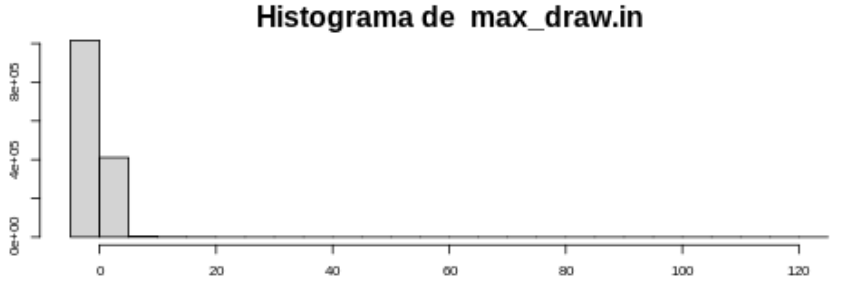

Histograma de min_home
0e+00
4e+05
8e+05
0
5
10
15
20
25
30
Histograma de min_draw
0e+00
4e+05
8e+05
0
5
10
15
20
25
Histograma de min_away
0e+00
4e+05
8e+05
0
5
10
15
20
25
30

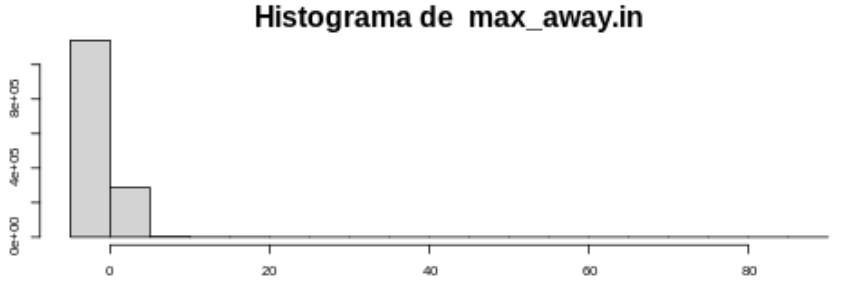

Histograma de mean_home.in
0
400000
800000
1200000
0
50
100
150

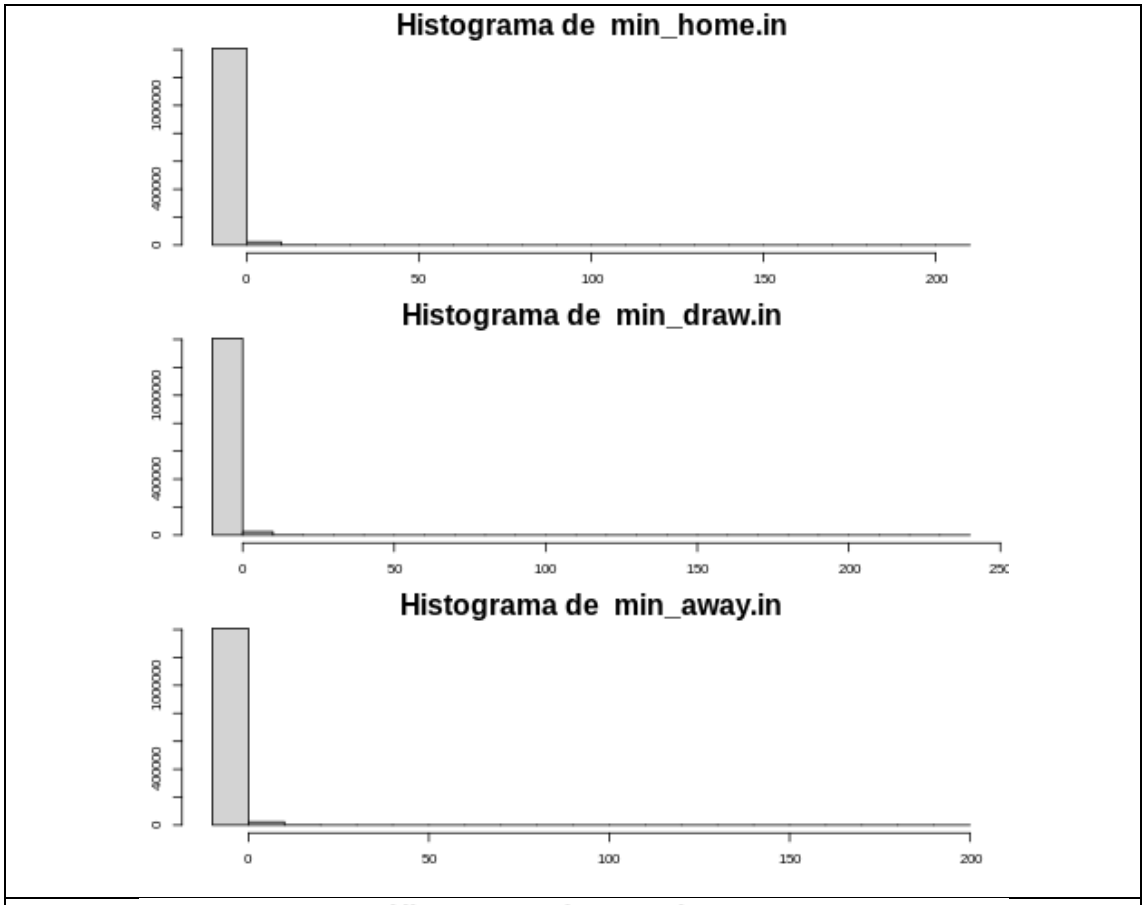

Histograma de mean_draw.in
0e+00
4e+05
8e+05
0
20
40
60
80
100

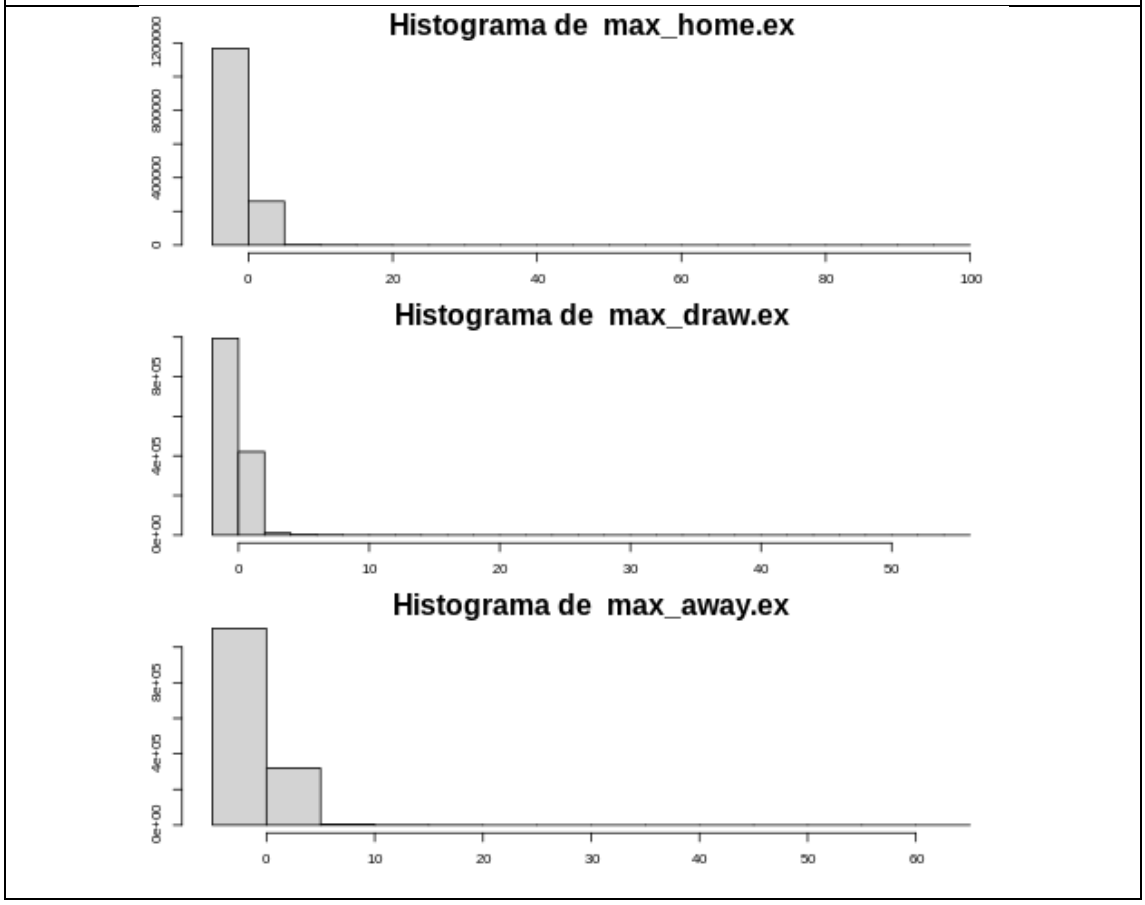

Histograma de mean_away.in
0
400000
800000
1200000
0
20
40
60
80
100
120

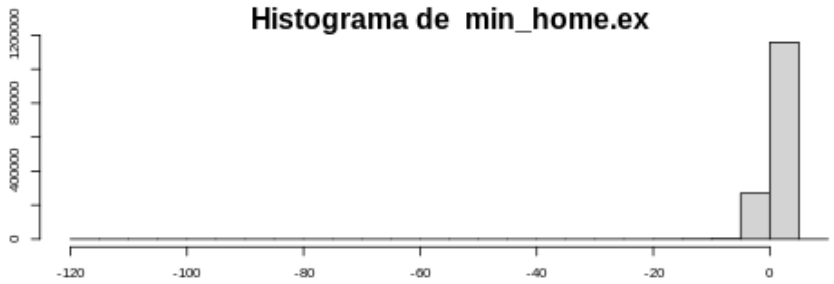

Histograma de max_home.in
0
400000
800000
1200000
0
50
100
150

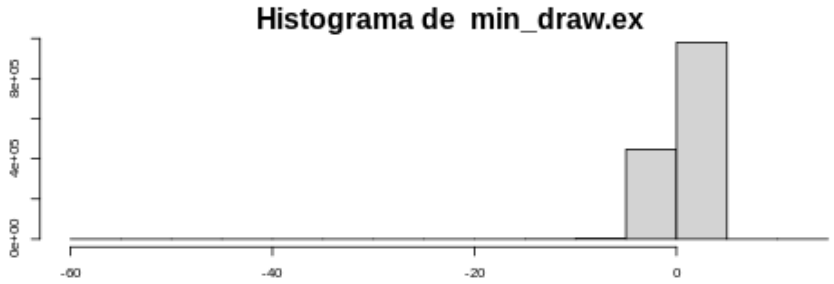

Histograma de max_draw.in
0e+00
4e+05
8e+05
0
20
40
60
80
100
120

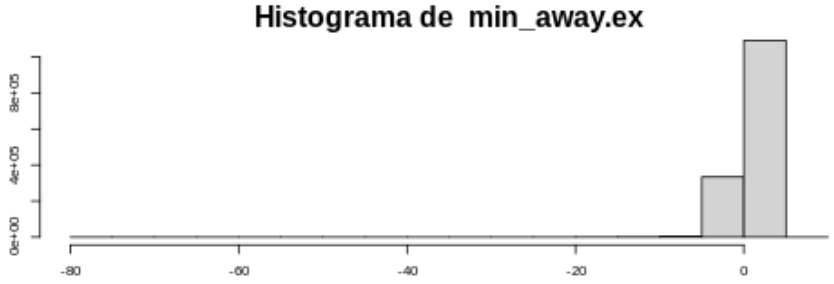

Histograma de max_away.in
0e+00
4e+05
8e+05
0
20
40
60
80

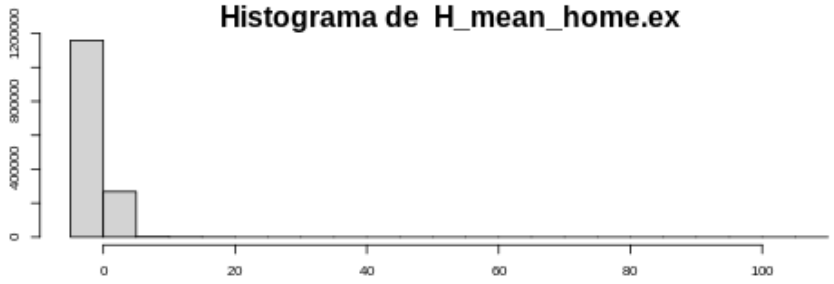

Histograma de min_home.in
Histograma de min_draw.in
Histograma de min_away.in

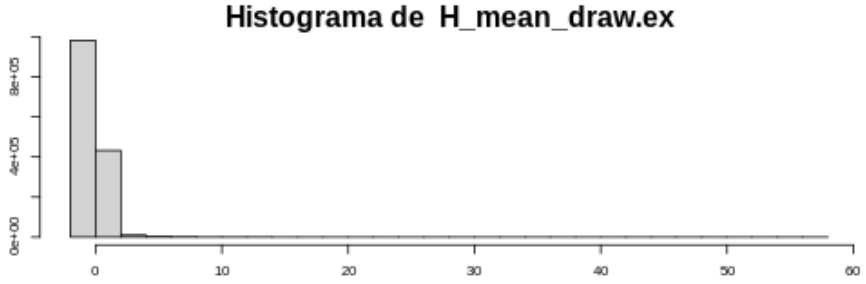

Histograma de max_home.ex
Histograma de max_draw.ex
Histograma de max_away.ex

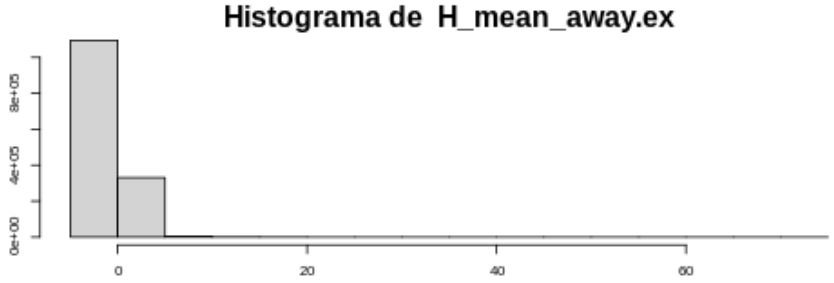

Histograma de min_home.ex
0
400000
800000
1200000
-120
-100
-80
-60
-40
-20
0

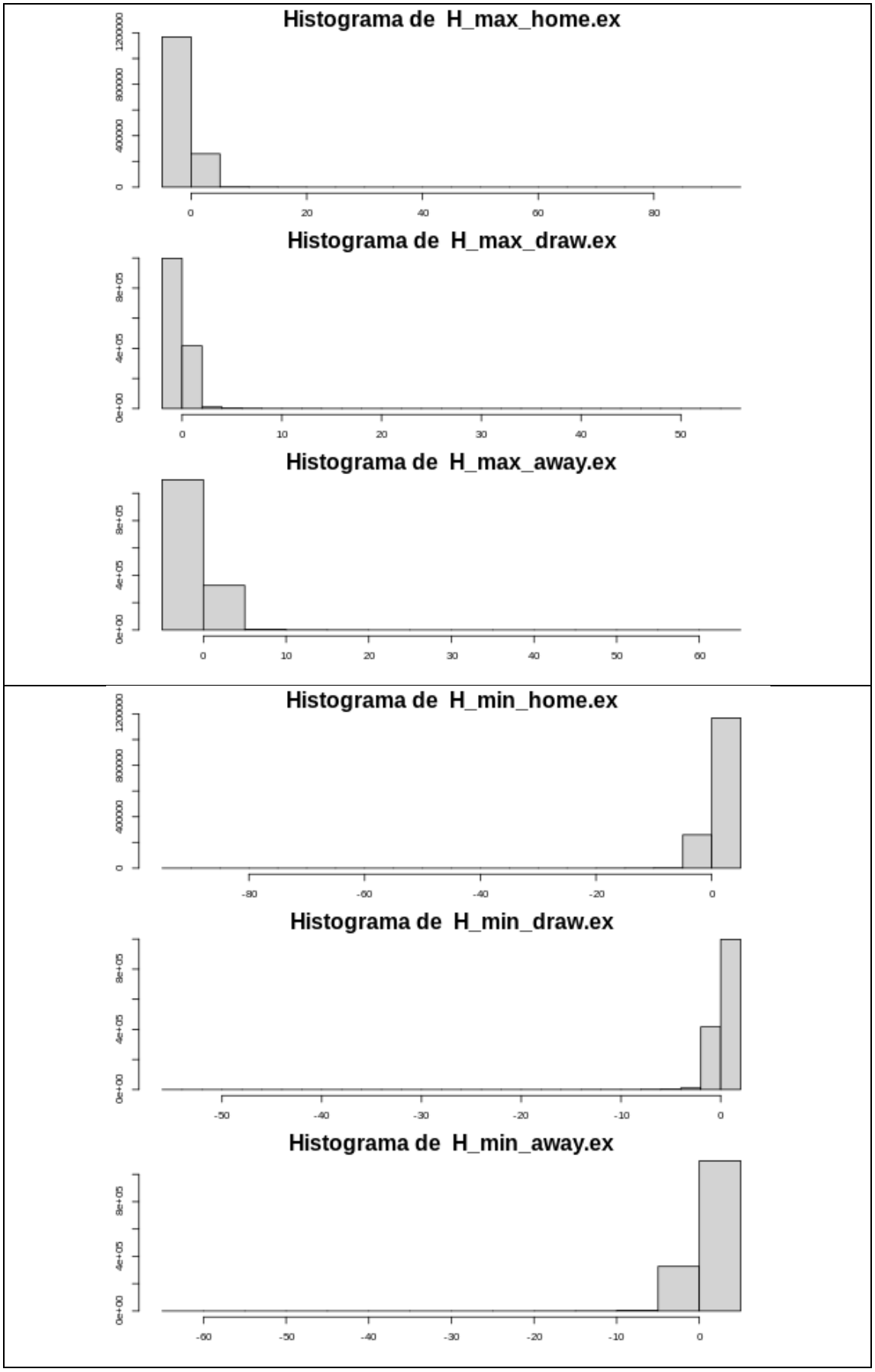

Histograma de min_draw.ex
0e+00
4e+05
8e+05
-60
-40
-20
0


Histograma de min_away.ex
0e+00
4e+05
8e+05
-80
-60
-40
-20
0


Histograma de H_mean_home.ex
0
400000
800000
1200000
0
20
40
60
80
100


Histograma de H_mean_draw.ex
0e+00
4e+05
8e+05
0
10
20
30
40
50
60


Histograma de H_mean_away.ex
0e+00
4e+05
8e+05
0
20
40
60



Anexo 1: histogramas de las variables numéricas de la base de datos final