UNIVERSIDAD TORCUATO DI TELLA

TP - Machine Learning

MiM + Analytics - 2022

Evaluando estrategias de inversión en carreras de caballos

Alumnos: Juan Spadaro, Nicolas Viñolo y Victoria Sosa

Profesor: Gabriel Martos Venturini

Julio 2022

1. Introducción

1.1. Planteo del problema

El objetivo del trabajo es maximizar la rentabilidad esperada para un apostador de carreras de caballos a partir de la utilización de los datos provistos. Para esto, no solo debemos modelizar la probabilidad para cada caballo de ganar la carrera/entrar en el podio, sino que también tendremos que diseñar una estrategia de inversión buscando la mayor ganancia posible por dólar invertido.

1.2. Estructura del trabajo

En la sección dos pre-procesamos los datos. Buscando entender mejor la información con la que estamos trabajando y posibles inconsistencias, hicimos un primer análisis exploratorio. Luego, realizamos data cleansing para corregir o eliminar datos erróneos o faltantes. Posteriormente, estudiamos si era necesario o no hacer un rebalanceo de datos. Por último, pero no menos importante, realizamos ingeniería de atributos.

A lo largo de la tercera sección del trabajo, recorrimos distintos modelos estudiados en la materia. Aquí tuvimos en cuenta los fundamentos teóricos, y analizamos las ventajas y desventajas para cada uno de ellos. Luego, recopilamos los resultados de cada modelo y realizamos comparativas entre los mismos, buscando elegir el modelo óptimo con el menor fb score (métrica creada por nosotros que pondera más precision que recall, con un beta de 0,05) sobre los datos de testeo. El criterio para seleccionar esta métrica será explicado en detalle en la sección 1.5.

Habiendo definido el modelo a utilizar para predecir la probabilidad de ganar la carrera para cada caballo, en la sección cuatro planteamos posibles estrategias de inversión buscando maximizar la ganancia esperada.

Finalmente, resumimos las principales conclusiones que obtuvimos en el trabajo. Además, comentamos posibles extensiones y limitaciones de las soluciones propuestas.

1.3. Decisión sobre variable a predecir

Antes de comenzar a resolver el trabajo, hicimos una revisión de literatura y artículos de internet respecto a la utilización de algoritmos de machine learning para predicciones de carreras de caballos. Uno de los debates se refiere a cómo clasificar a los caballos.

Dentro de las clasificaciones más comunes encontramos dos: ganadores- perdedores y “caballos que entran al podio”-“caballos que no entran al podio”. En nuestro problema, si el caballo gana la carrera, recibe lo correspondiente a “win_odds” y sí entra en el podio recibe el monto de “place_odds”.

También se puede llevar a un problema de clasificación multiclase: primeros, segundos, terceros y perdedores.

Dado que el consenso gira en torno a clasificar como ganadores o perdedores, este fue el criterio adoptado en nuestro trabajo (Won=1 si es ganador, Won =0 si es perdedor).

1.4. Criterio de decisión

Para poder tomar una decisión respecto a que modelo y estrategia de inversión elegir, decidimos separar el dataset en tres: un set de entrenamiento (un 62Primero, planteamos cada uno de los modelos estimando los parámetros con los datos del training set. Para optimizar los hiperparámetros fuimos probando diferentes combinaciones de los mismos y luego realizamos predicciones sobre el set de validación. Luego, seleccionamos aquellas combinaciones de hiperparámetros que dieron un mayor fb score (f score que pondera más precision que recall, con un beta de un 0.05) para cada modelo.

Posteriormente, optimizamos el umbral utilizado para decidir si apostar o no en cada caballo a partir de realizar predicciones sobre el test de validación con los hiperparámetros seleccionados en el ítem anterior también buscando maximizar el fb score.

Luego de optimizar tanto hiperparámetros como el umbral, decidimos elegir el modelo que diera un mayor fb score sobre los datos de validación. En este caso fue XGBoost.

Al momento de definir la mejor decisión de inversión, calculamos la ganancia esperada para distintas posibles estrategias, en base a las probabilidades calculadas por el modelo mencionado en el párrafo anterior. La optimización de umbrales se realizó utilizando las predicciones sobre el test de validación buscando maximizar el retorno esperado por dólar invertido. Por último, la ganancia esperada para cada estrategia se probó sobre el set de testeo con el fin de simular un escenario similar al de producción.

1.5. Métricas y elección del F Beta score

Una de las discusiones que tuvimos durante el transcurso del trabajo refirió a cuál de las métricas tendríamos que utilizar para optimizar tanto hiperparámetros como umbral de clasificación.

Analizamos utilizar accuracy (que mide el porcentaje de aciertos que tiene el modelo), precision (cuántos de los que clasifique como 1 efectivamente lo son), recall (cuantos de los que son 1 pudimos clasificar como 1), F1 score (combina precision y recall en partes iguales) y la curva de ROC (nos dice qué tan bien puede distinguir el modelo entre dos categorías). Accuracy y recall pierden potencia frente a clases desbalanceadas, como es el caso de nuestro trabajo (ver sección 2.2.).

Dada la naturaleza del problema que estamos analizando (a qué caballos apostar) consideramos que precision es una métrica altamente relevante. Esto es porque queremos acertar la apuesta ya que sino perderemos dinero. Por otro lado, también nos interesa tener en cuenta la accuracy ya que por lo que leímos sobre el tema, si únicamente se maximiza precision es muy probable que se termine haciendo un overfitting de los datos.

Entonces, pensamos en crear una nueva métrica: F beta score con un beta de 0.05. Aquí se pondera de mayor manera precision, pero se sigue teniendo en cuenta recall. Esta es la métrica que intentaremos maximizar a lo largo del trabajo cuando analicemos modelos. Utilizamos la libreria metrics para crear el fbscore.

Esta métrica fue elegida por nosotros en base al problema. Dependiendo de qué estemos analizando, el beta podría cambiar. Inclusive, podríamos haber utilizado otro beta (aunque siempre dado el problema tendría que ser menor a 1 para que pondere más precision que recall).

2. Descripción general del pre-procesamiento de datos

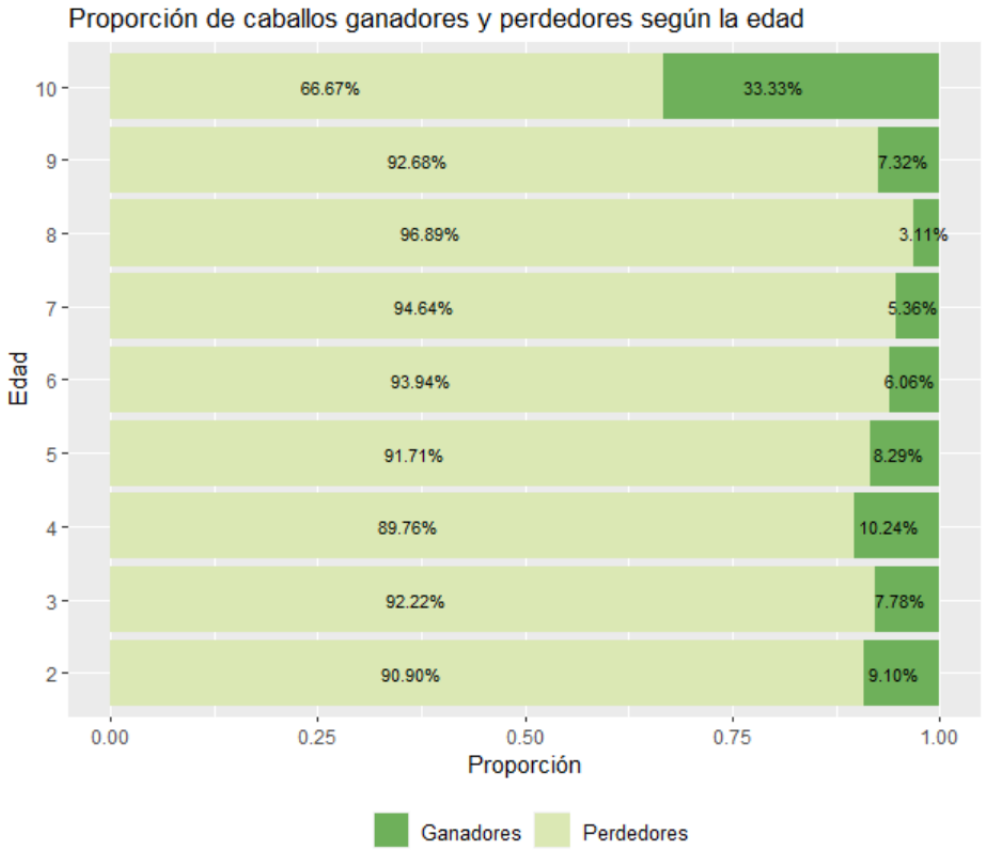
2.1. Exploración de datos y data cleansing

2.1.1. Comentarios generales

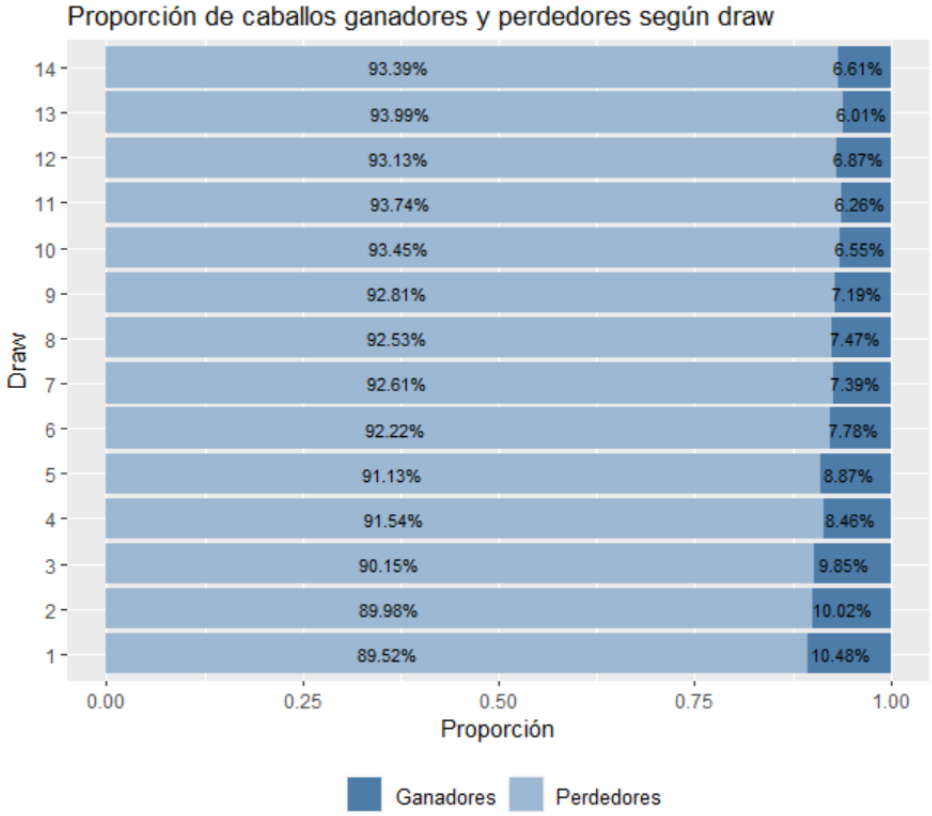
Para llevar a cabo este trabajo sobre carreras de caballos contamos con dos fuentes de información, por un lado el file "races" que contiene información particular sobre las carreras y por otro lado "runs" que describe las características de cada caballo que compite en las carreras dadas en "races". Ambos files tienen un dato en común, que es "race_id" el número único de identificación de cada carrera. A partir de la función dada, unimos ambos datasets para obtener una única base de datos, la cual cuenta con 79.447 observaciones y 73 variables.

La variable que buscaremos predecir para luego realizar estrategias de inversión que nos permitan maximizar la rentabilidad se llama "won" y toma valor 1 para aquellos caballos que ganaron la carrera y 0 en el resto de los casos.

Durante la exploración de los datos nos resultó llamativo que el 77 % de los caballos tienen 3 años de vida, luego de investigar un poco al respecto pudimos comprender que se debe a que es la edad más recurrente en carreras de caballos y es considerada por algunos expertos como óptima, en los gráficos que se encuentran a continuación quisimos comprender un poco si había diferencias significantes también en la proporción de ganadores/perdedores para cada edad y draw[1]:



[1]Post position number of the horse in this race.



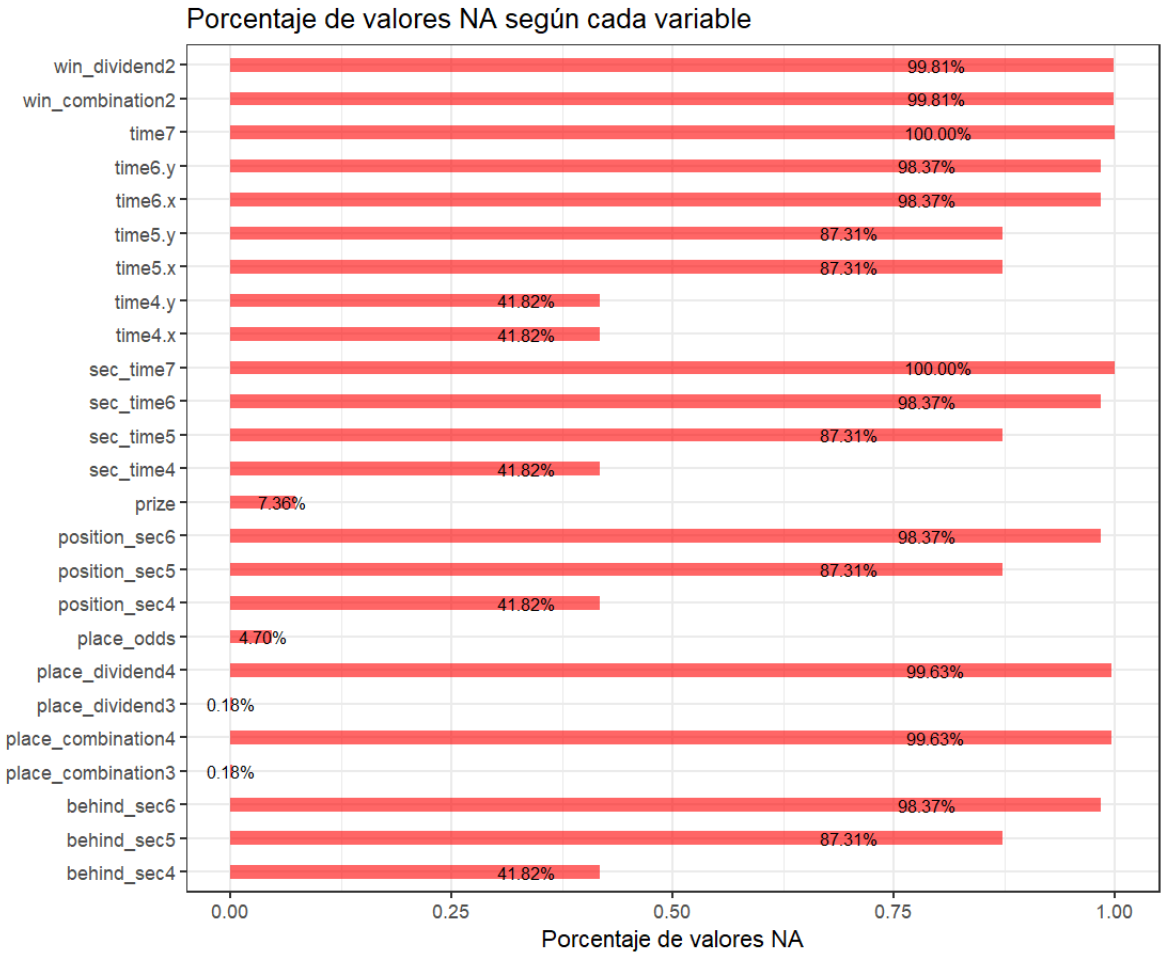
Con una simple inspección gráfica, uno podría argumentar que no hay grandes diferencias entre caballos ganadores y perdedores en las distintas edades, excepto para los caballos de 10 años, pero creemos que se debe a las pocas observaciones existentes para esa edad. Similares comentarios aplican para draw.

Siguiendo con el análisis, también puede observarse la predominancia de caballos de origen Australiano y Neozelandés, ya que entre ambas nacionalidades componen más del 70 % del total de los caballos. Respecto al resto de las variables categóricas no observamos diferencias significativas como para considerar que las clases están mal distribuidas.

Para poder dividir los datos entre los que se usarían para entrenar los modelos y los de validación, decidimos crear 3 conjuntos: "train", "test" y "valid". Para esto creamos una nueva variable "train_val_test" que le asigna "test" a las observaciones con fecha posterior al 05/01/2004, "valid" a las anteriores a esa fecha pero posteriores al 05/08/2002 y "train" al resto de las observaciones. Se decidió realizar una división de los sets con el criterio temporal mencionado y no de forma aleatoria, ya que consideramos la existencia de temporalidad y no independencia en las observaciones. Si bien esto nos permite identificar a qué conjunto pertenece cada observación, aún no realizamos la división y los siguientes pasos se llevaron a cabo sobre todo el set.

2.1.2. Data Cleansing

Al explorar el dataset, notamos la existencia de valores faltantes. Para identificarlos de forma más simple y clara creamos el siguiente gráfico que muestra la proporción de valores faltantes por feature que cuenten con esta característica:



Para las variables "place_odds", "place_dividend3" y "place_combination3" decidimos eliminar las observaciones con datos faltantes debido a que no representaban una gran cantidad de datos y contamos con suficiente información aún sin ellos.

En el caso de la variable "prize", al tratarse de una variable numérica, que representa los premios por carreras nos pareció óptimo imputar la media a los valores faltantes. Esto lo hicimos únicamente sobre los datos de entrenamiento, a partir de la media de los datos del mismo set para evitar cometer data leakage. Por este motivo, es que en el código se realizó más adelante, cuando realizamos la división de los conjuntos de sets.

Respecto al resto de las variables con presencia de missing values que no fueron mencionadas anteriormente, las consideramos variables spoilers, ya que contienen información que no tendremos disponible al momento de producción. Es por ese motivo que decidimos no incluirlas en nuestro dataset, además de que contienen un porcentaje de NAs muy alto.

En el análisis de los datos no se encontraron inconsistencias en la información, pero si outliers en las variables "weight_ratio", "distance" y "prize". Los outliers representan valores atípicos respecto al resto de la muestra, y fueron identificados a través de los diagramas de cajas y bigotes realizados con boxplots. Siguiendo este criterio, eliminamos todas las observaciones con valores atípicos en las 3 variables mencionadas, debido a que no eran un gran porcentaje de los datos y podrían generar distorsiones en nuestro modelos.

2.2. Tratamiento de clases desbalanceadas

Decidimos no realizar ninguna técnica de balanceo de clases, ya que por la naturaleza de la variable "won" siempre tendrá una proporción menor de caballos ganadores, porque de todos los caballos que compiten en una carrera el ganador es sólo uno. Ahora bien, tuvimos en cuenta el desbalanceo de clases para analizar nuestras métricas, como ya se ha nombrado anteriormente.

2.3. Feature Engineering

En esta instancia del trabajo se realizan modificaciones, nuevas variables y todo tipo de ingeniería sobre los datos que permitan tener una mejor formulación de los atributos buscando destacar aspectos importantes y que nos aporten una mayor información. En todo momento buscamos evitar realizar data leakage, es decir que en el conjunto de entrenamiento no se filtre información que no se espera que esté disponible al momento de usar el modelo para predecir. Está técnica suele estar asociada con una buena performance, pero sólo para entrenamiento y no así en producción.

Utilizamos variables numéricas para crear otras variables relativas y ratios. A partir de la columna "date" creamos nuevas variables buscando capturar comportamientos estacionales.

En el análisis de los datos observamos la presencia de muchos atributos que podrían ser de tipo factor. Algunas de las variables contaban con muchos valores que podrían considerarse como categorías, por lo que para evitar caer en una cantidad excesiva de niveles decidimos agruparlas en una menor cantidad de categorías, como se realizó por ejemplo con las variables "horse_country" que la convertimos en "horse_geography" y "horse_ratings" que pasó a ser "ratings_group". Para la variable "horse_gear" utilizamos la técnica de binarización que consiste en asignar un 1 a las observaciones que cumplen con cierta condición para ese feature, y 0 en caso contrario. Esto nos permitió convertir a dicho atributo en uno de sólo dos niveles, los cuales indican si el caballo tiene o no equipo ya que de otra forma hubiese tomado demasiados valores. Y por último, decidimos convertir algunas variables a tipo "factor" conservando los niveles que ya tenían.

De esta forma llegamos al dataset definitivo, y procedimos a eliminar ciertas variables, como por ejemplo las relacionadas al tiempo o posición de un caballo en una carrera, por considerar que tenían información que no tendríamos al momento de realizar las predicciones (las consideramos variables spoilers), o simplemente eran irrelevantes para el problema a resolver. Pasamos luego a la división del dataset en los subconjuntos de entrenamiento, test y validación que fueron asignados al comienzo del trabajo. Una vez identificado el "train_set" o conjunto de entrenamiento, procedimos a realizar cierta ingeniería de atributos sólo sobre ese conjunto que habíamos dejado para después con el fin de no cometer data leakage.

En primer lugar, imputamos la media a la variable "prize" para los valores faltantes. Asignamos un ID único a cada caballo, jockey y entrenador y para cada uno de ellos creamos un nuevo atributo que tomara valor 1 o 0 según si se lo considera experimentado o no en función de la cantidad de carreras en las que participó. Para este nuevo atributo se decidió tomar sólo la información obtenida a partir del set de entrenamiento y aplicarla a los id correspondientes en el set de validación y testeo. Esto nos permite basarnos en información histórica (solo entrenamiento) de cada uno para asignarle su expertise correspondiente, pero sin realizar data leakage.

Cabe destacar, que esto sólo sirve para ID ya vigentes, por lo cuál debería hacerse dinámicamente para poder capturar nuevos ID. La variable caballo experimentado no la incorporamos porque en el set de validación y testeo la proporción era muy pequeña, ya que los caballos no compiten por tantos años y extrapolar información de entrenamiento a estos sets resultaba inconveniente. Para

más detalles, esto puede verse en el código en la línea 342.

2.4. Prevención de data leakage

Uno de nuestros principales esfuerzos estuvo en minimizar lo máximo posible el data leakage. Esto lo hicimos al sacar outliers (solo sacamos los outliers de entrenamiento, con información de entrenamiento) y al crear nuevas variables (como la de experiencia o no del corredor y entrenador).

Cuando elegimos hiperparámetros y umbral siempre lo hicimos sobre el set de validación para poder luego conocer sus predicciones en testeo.

3. Selección de modelos

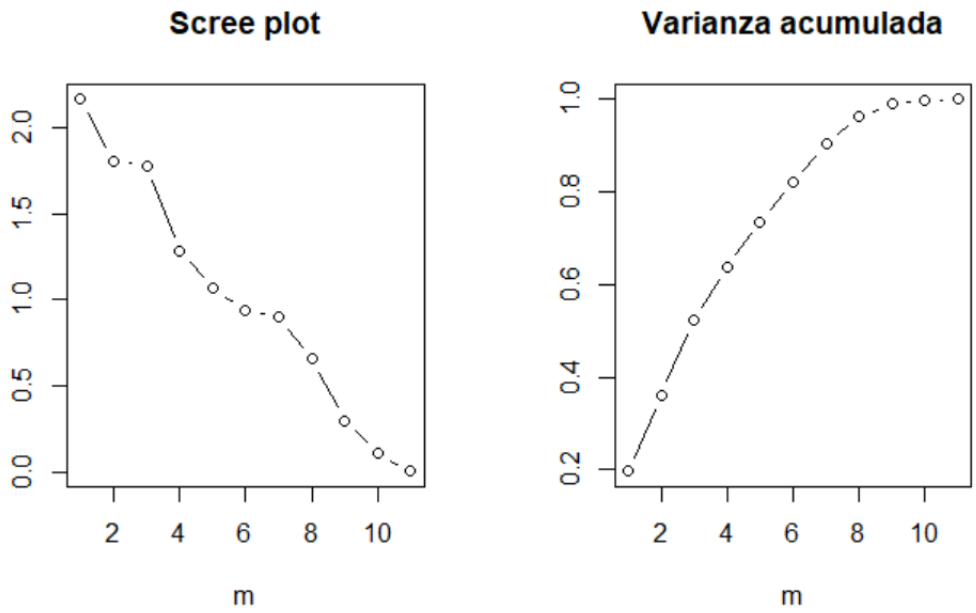
3.1. Modelos

Como mencionamos en la introducción, las decisiones a lo largo del trabajo fueron motivadas por la reducción de la métrica Fb score. A su vez, para los modelos probados, hemos realizado optimización de parámetros utilizando los datos de validación. Luego, reentrenamos los modelos con la nueva selección de hiperparámetros y umbral con los datos de entrenamiento + validación, e hicimos predicciones sobre el set de testeo.

Al momento de elegir hiperparámetros utilizamos la técnica de random grid que nos permite recorrer mejor las posibles combinaciones de hiperparámetros con un menor costo computacional. Esto se puede ver en la sección 8 del código en R.

3.1.1. Principal component analysis (PCA)

Decidimos llevar adelante un análisis de las componentes principales, técnica que nos permite visualizar los datos y comprender cuáles son los componentes que logran captar mayor parte de la variabilidad en nuestro modelo. Es importante destacar que esta técnica sólo puede aplicarse en variables numéricas y que a su vez lleva adelante una normalización de los datos para poder comprarlas todas en misma escala. Pudimos concluir que para capturar el 90 % de varianza de nuestro modelo sería óptimo incluir 8 componentes, como se ve en el gráfico a continuación.



Realizamos la regresión con componentes principales (PCR), y obtuvimos las predicciones correspondientes de nuestra variable "won" en testeo.

3.1.2. Árboles de decisión

El segundo análisis lo realizamos utilizando árboles, el cual definimos como nuestro modelo base. La construcción de los árboles de decisión estará guiada por la minimización del error en los datos de entrenamiento. El modelo consiste en particionar el espacio de atributos en regiones que estarán definidas por si se cumplen o no una serie de reglas, y en función a eso se asignará a una observación a una región u otra, creando así una estructura de árbol. El modelo particiona cada región de forma recursiva buscando encontrar la partición que produzca una mayor reducción en el error. Este modelo predice la probabilidad condicional de una clase, en función de la proporción de observaciones de la clase correspondiente que cayeron en la región de esa observación. Es decir, que para cada observación se va a predecir la probabilidad de que pertenezca o no a una clase determinada, en nuestro trabajo sería la probabilidad de ganar o no una carrera.

Uno de los problemas que surgió al intentar utilizar este modelo es que no se nos generaba el árbol. Pudimos entender luego que esto ocurre porque el parámetro de complejidad era demasiado alto, por lo que lo bajamos y pudimos fittear el árbol correctamente. Aquí optimizamos todos los hiperparámetros como ya mencionamos.

3.1.3. Regresión logística sin regularización

Decidimos probar el modelo de Regresión Logística sin regularización, este es un modelo probabilístico muy útil para clasificación binaria como es nuestro caso. Las estimaciones del modelo se realizan por máxima verosimilitud, es decir que los valores estimados para los parámetros desconocidos buscan dar un modelo que tenga una mayor probabilidad de reproducir los datos observados. Permite asumir formas funcionales no lineales, garantizando un output que está acotado entre 0 y 1. Es decir que el modelo arroja probabilidades, y para poder convertir esas predicciones probabilísticas en predicciones de clasificación es necesario elegir un umbral. Tuvimos que seleccionar un umbral para considerar a las observaciones de una clase u otra según el valor de

la probabilidad predicha. Entre sus desventajas se encuentra que es complicado obtener relaciones complejas entre variables. Algoritmos que puedan captar relaciones complejas es posible que sean superiores.

3.1.4. Regresión logística + LASSO

Para probar un algoritmo automático de selección de variables, decidimos llevar a cabo una regresión Lasso. Esta selecciona automáticamente qué variables utilizar al asignarle cero al coeficiente correspondiente a la variable que no va a utilizar (en contraposición a Ridge donde las variables menos relevantes asumen coeficientes cercanos a cero). Tuvimos que convertir las matrices a esparsas para poder realizar las estimaciones porque así lo pide la librería que usamos en R.

Al estimar el modelo, no obtuvimos una diferencia significativa respecto a la regresión logística sin regularización.

3.1.5. Random Forest

El quinto modelo probado pertenece a lo que se conoce cómo ensamble de modelos. Los que utilizan estas técnicas de ensamble, no entrenan con un único modelo sino que lo hacen con un conjunto de modelos obteniendo como resultado final una combinación de las predicciones de cada uno. El ensamblado permite obtener modelos con menor varianza que cada uno de ellos en solitario y de esta forma también logran alcanzar una mejor perfomance.

Dentro del ensamble de modelos, usamos Random Forest y se destaca principalmente por el doble remuestreo. Este modelo busca bajar la covarianza entre las predicciones de los modelos ensamblados, utilizando una técnica de remuestreo con reposición de la muestra donde no sólo alterna entre las filas a considerar (como lo hace Bagging), sino que también lo hace con las columnas. Esto le permite mejorar la capacidad predictiva del modelo, quitando correlación entre ellos y también lo hace más eficiente computacionalmente ya que favorece el entrenamiento del mismo. Random Forest va construyendo árboles con muestras de tipo bootstrap seleccionando de forma aleatoria una cantidad de features determinada como hiperparámetro, y así sucesivamente para cada árbol hasta cumplir con una profundidad máxima también indicada como hiperparámetro.

Como podemos ver en la sección 3.2. no obtuvimos una diferencia significativa frente a los modelos logísticos. Creemos que esto se debe a que la métrica que seleccionamos para maximizar en el algoritmo es “ROC” (que no lo logra tomar correctamente y termina maximizando accuracy). Es posible que si en cambio maximizara fb score, los resultados puedan ser mejores pero no es una opción dentro de las métricas a optimizar.

3.1.6. Extreme gradient boosting (XG Boost)

El último modelo entrenado fue XGBoost y también utiliza como técnica el ensamble de modelos pero dentro de un grupo que se denomina Boosting.

La diferencia de boosting con Random Forest es que este último se enfoca en la reducción de la varianza mientras que boosting consiste en el ensamble de modelos muy simples, de forma tal que al entrenarlos en conjunto se logre reducir el sesgo de las estimaciones. En estos modelos

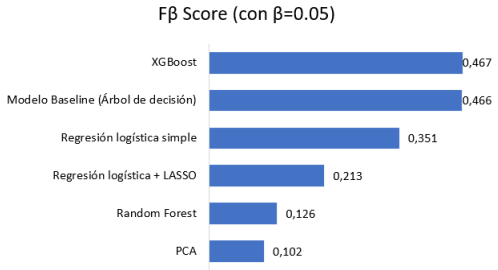
el ensamble es secuencial, y todos los modelos están comunicados entre si ya que cada modelo tratará de mejorar lo que el modelo de la secuencia anterior no haya logrado predecir bien.

Cada nuevo árbol trabaja sobre los residuos del modelo anterior, se construye de manera secuencial a partir de lo que construyó el anterior. En este caso, el remuestreo bootstrap es opcional ya que lo que va a cambiar modelo a modelo es el peso de las observaciones, otorgando un mayor peso a aquellas que el modelo anterior no pudo predecir correctamente. De esta forma, cada modelo irá teniendo importancias relativas diferentes que se tendrán en cuenta a la hora de combinar todos los modelos y obtener la predicción final. Es un modelo con una gran cantidad de hiperparámetros, lo cuál es importante controlar ya que podría cometerse overfitting.

3.2. Resultados y comparativas entre modelos

En la siguiente tabla exponemos los resultados obtenidos sobre las principales métricas mencionadas a lo largo del trabajo para cada modelo:

	PCA	Modelo Baseline (Árbol de decisión)	Regresión logística simple	Regresión logística + LASSO	Random Forest	XGBoost
Accuracy	0,397	0,92	0,919	0,92	0,507	0,92
Precision	0,102	0,48	0,367	0,286	0,125	0,489
Recall	0,843	0,038	0,019	0,002	0,868	0,024
F1 Score	0,182	0,07	0,036	0,004	0,219	0,046
Fβ Score (con β=0.05)	**0,102**	**0,466**	**0,351**	**0,213**	**0,126**	**0,467**
AUC	0,6005	0,517	0,508	0,5008	0,6715	0,5109



Como se puede ver, el modelo de mejor performance es XGBoost con un fb score de 0.467.

4. Etapa prescriptiva

4.1. Motivación

Ya con los resultados de los modelos probados, procedemos a plantear las distintas alternativas de inversión buscando maximizar la rentabilidad del inversor. Aquí, reflexionamos respecto a qué es lo que estamos buscando para poder llevar a producción (es decir, fuera de este entorno que conocemos) un sistema de decisión de apuestas que permita obtener la mayor rentabilidad por dólar invertido.

Si bien obtenemos un mayor retorno por dólar invertido sí mantenemos un umbral alto y apostamos a pocos caballos, esta estrategia podría ser riesgosa en contextos de producción. Esto puede suceder si nuestro modelo no predice tan bien frente a datos nuevos y hacemos apuestas de montos grandes sobre nuevos datos. Entonces, tenemos un trade-off entre riesgo y rentabilidad. A continuación, veremos distintas estrategias a partir de la predicción con XGBoost con hiperpárametros optimizados.

4.2. Primera estrategia usando el mejor modelo (y apostando poco)

Lo primero que pensamos fue llevar a cabo una estrategia donde a partir de las estimaciones del modelo XGBoost apostemos solo a aquellos caballos que el modelo clasifica como ganadores, con

el umbral que optimizamos en la sección 3.1.6. Apostando únicamente a 38 caballos sobre 11.263 competidores obtiene una rentabilidad de un 66.84 % por dólar.

Ahora bien, el umbral fue seleccionado intentando maximizar el fb score calculado por nosotros. ¿Qué pasaría sí ajustamos el umbral maximizando retorno por dólar?

4.3. Segunda estrategia optimizando umbral

Para optimizar el umbral, utilizamos el set de validación y luego re entrenamos con validación + entrenamiento para hacer nuestra prueba final sobre testeo. Con el umbral optimizado, apostamos únicamente a 4 caballos sobre 11263 competidores y obtuvimos una rentabilidad de 97.5 %.

Estas dos estrategias pueden ser riesgosas ya que para obtener mucho dinero hay que apostar montos grandes en pocos caballos.

4.4. Tercera estrategia apostando doble y simple

Buscando diversificar el riesgo, pensamos en la posibilidad de apostar el doble a aquellos caballos que superan cierto umbral, y simple a aquellos que superan un umbral menos exigente. De esta manera, podríamos apostar a más caballos y lograr mitigar el riesgo. Apostando a 291 caballos (287 simple y 4 doble) obtenemos una rentabilidad de 69.28 %. La rentabilidad es menor pero el riesgo también.

4.5. Cuarta estrategia apostando triple, doble y simple

Siguiendo la misma lógica del ítem anterior, definimos tres umbrales (uno muy exigente, uno menos exigente y otro poco exigente) para decidir sí apostar triple, doble o simple a cada caballo. Apostando a 5096 caballos (4734 simple, 291 doble y 4 triple) obtenemos una rentabilidad del 65.099 %. Tanto la rentabilidad como el riesgo son aún menores que en las otras dos.

4.6. Aclaración respecto a estrategias

Una aclaración muy importante es que al momento de llevar a cabo las estrategias tres y cuatro no optimizamos los posibles umbrales tal que busquen maximizar la rentabilidad esperada. Frente a la elección de otros umbrales tanto la cantidad de apuestas como rentabilidad y riesgo se verían afectadas.

Por otro lado, sí hubiéramos armado el problema de clasificación como caballo perteneciente al podio o no, también podría haber cambiado la estrategia. Inclusive, se podría haber establecido apostar en todas las carreras al caballo con mayor probabilidad de ganar. La cantidad de posibles estrategias son muchas y no es posible abarcarlas en este trabajo.

5. Reflexiones finales

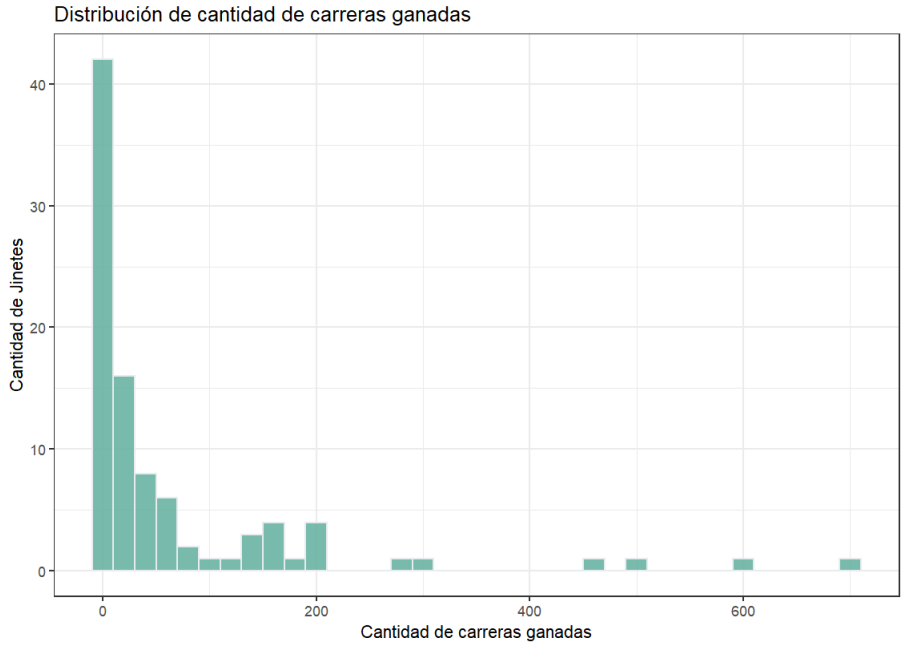
A lo largo del trabajo hemos analizado los datos, explorado distintos modelos e ideado distintas estrategias con el objetivo de poder llevar a cabo un sistema de apuestas de caballos que sea redituable para el apostador. Para este problema en específico, encontramos que XG Boost es el mejor modelo para poder hacer predicciones de caballos ganadores.

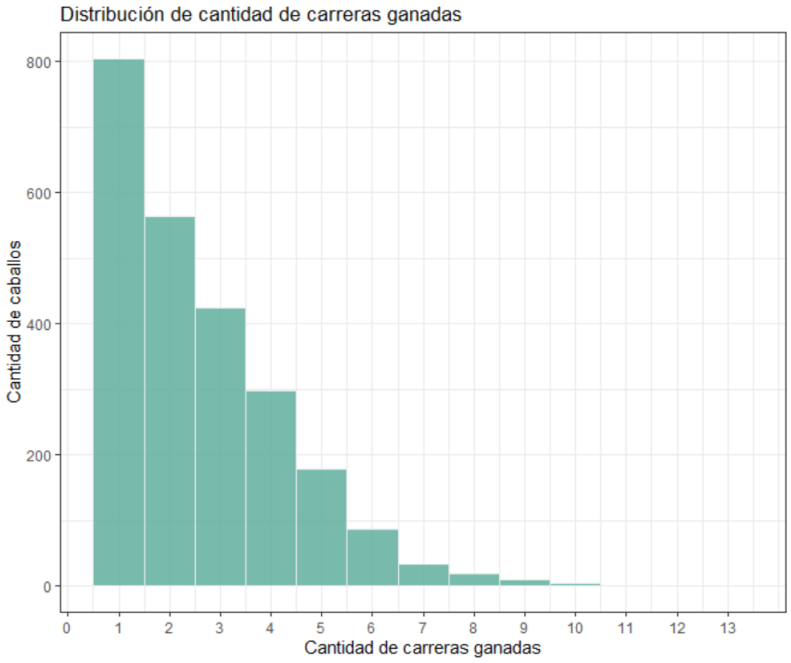
Los modelos siempre fueron analizados buscando maximizar el Fb score. Esto fue un criterio adoptado por nosotros, pero fácilmente el beta elegido podría cambiar como se mencionó en la sección 1.5.

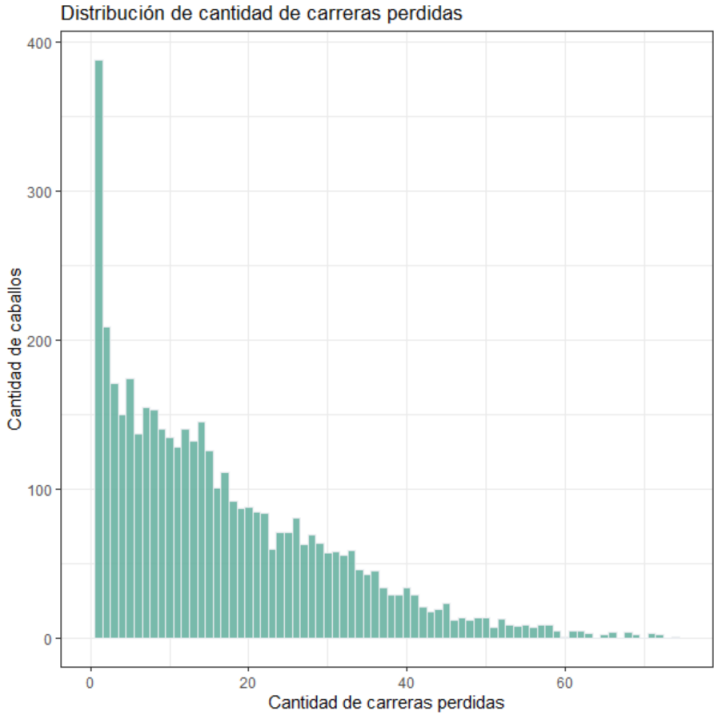
Al definir las estrategias en la última sección del trabajo, logramos idearlas dependiendo del perfil de apostador. La estrategia 4.3 puede ser tomada por un apostador dispuesto a tomar un riesgo muy alto, la estrategia 4.4 para un apostador con riesgo moderado y la estrategia 4.6 para un apostador dispuesto a tomar menos riesgo. Creemos que frente a este problema no es posible dar una solución única, dependerá de quién sea quien tome el modelo y en qué lapso de tiempo decidirá llevar a cabo las apuestas.

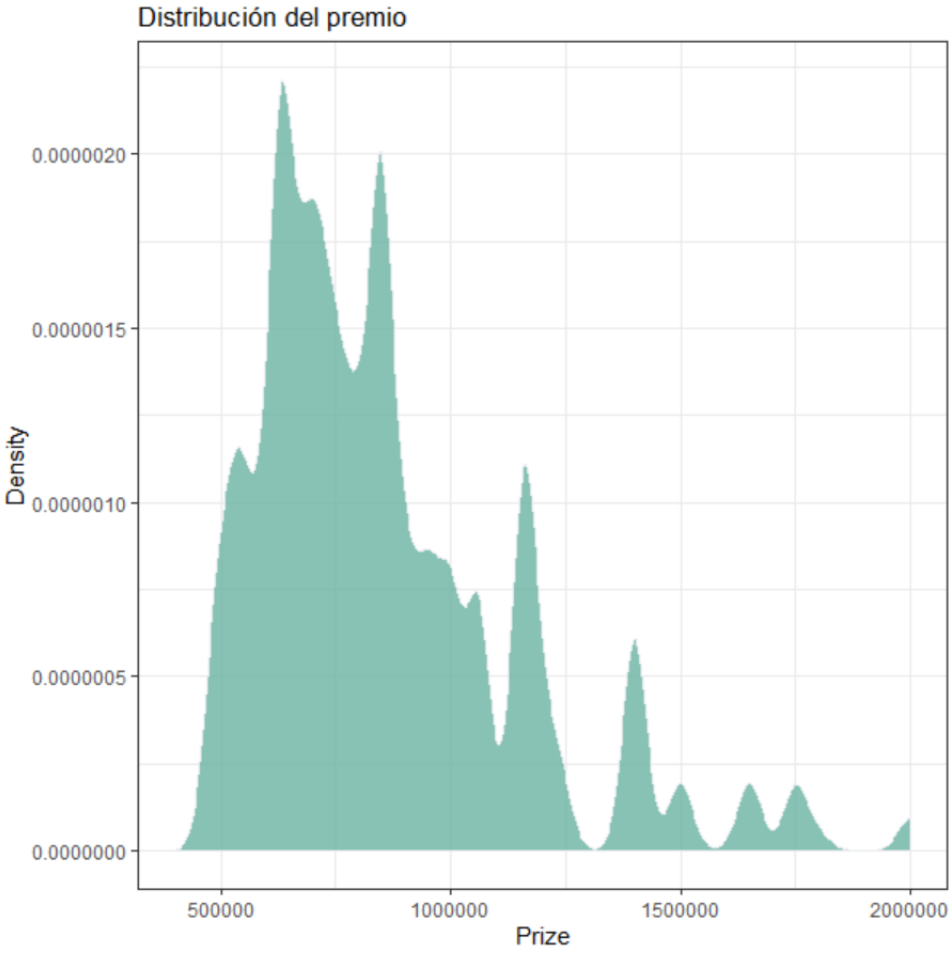
Como vimos en clase, es posible que las predicciones que realicemos no sean fiables en un periodo largo en el tiempo (como es el caso de las predicciones de la bolsa). Esto también debe ser tenido en cuenta al momento de tomar una decisión.

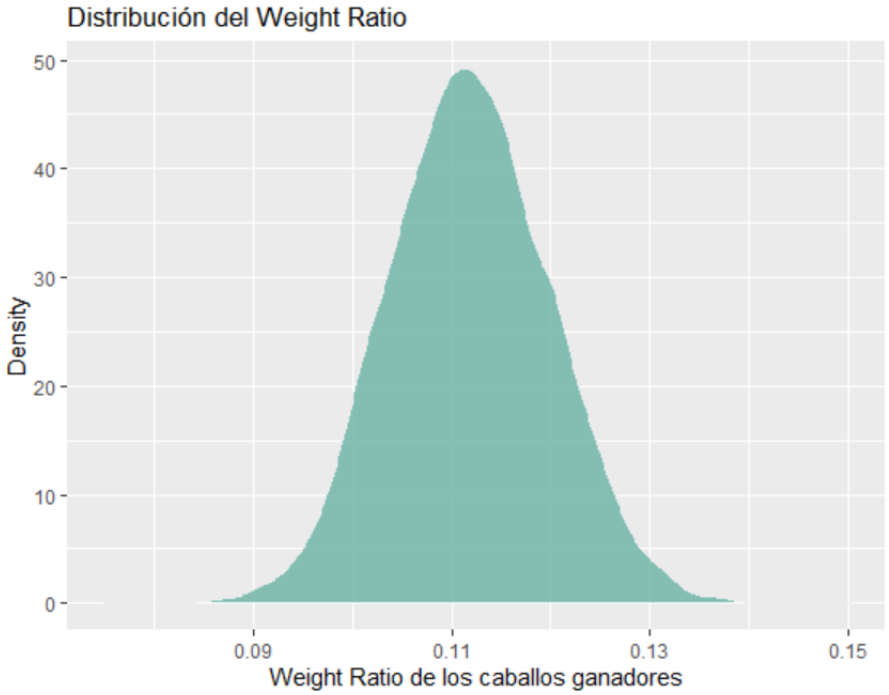
Anexo gráficos

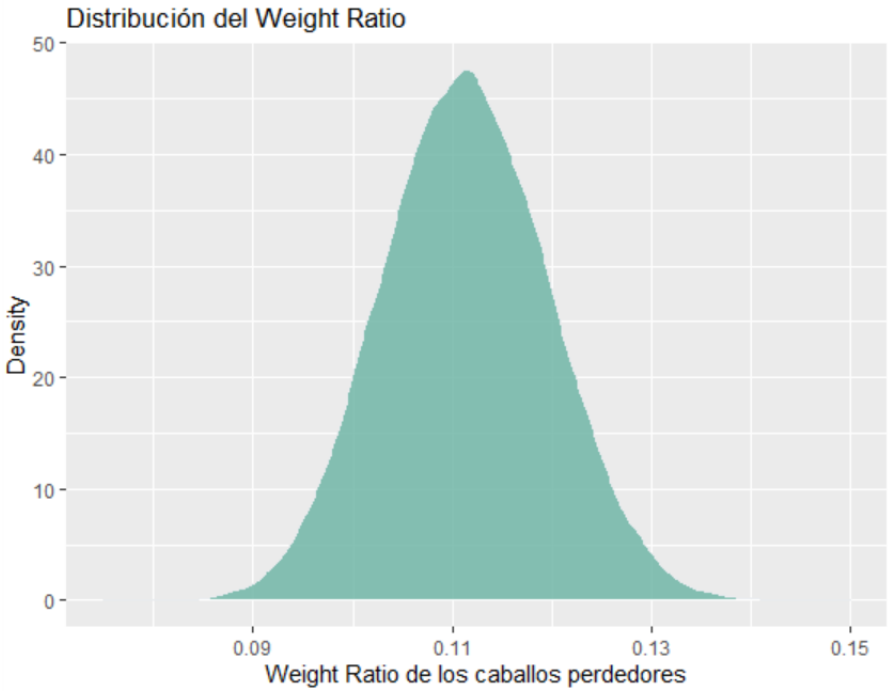


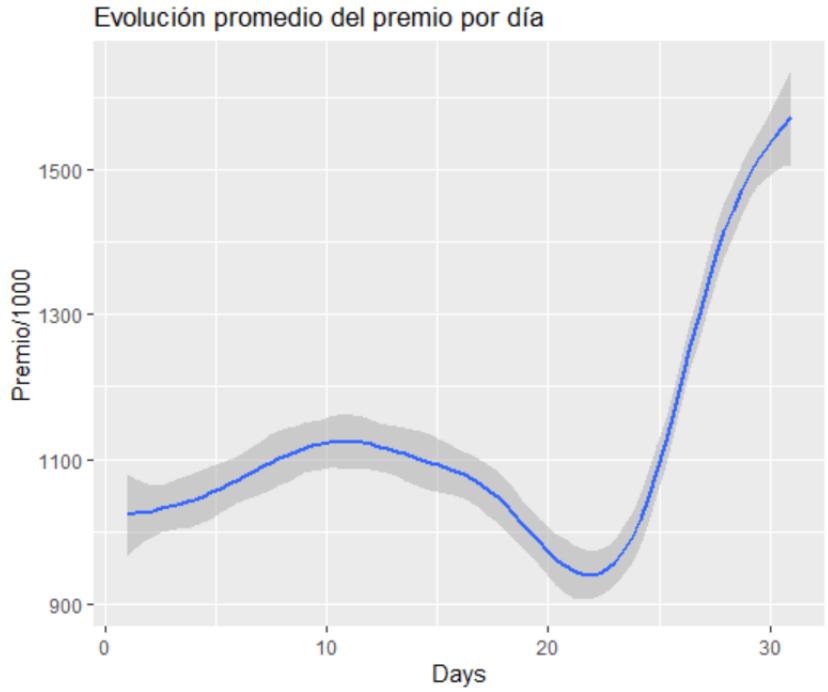

Distribución de cantidad de carreras ganadas
Cantidad de caballos
800
600
400
200
0
0
1
2
3
4
5
6
7
8
9
10
11
12
13
Cantidad de carreras ganadas

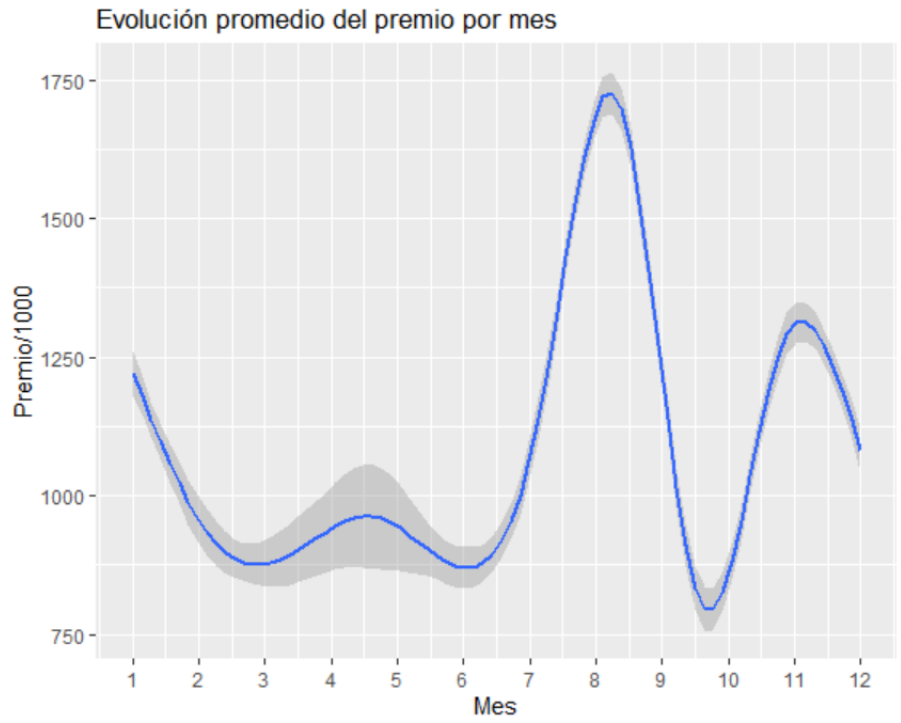

Distribución de cantidad de carreras perdidas
Cantidad de caballos
400
300
200
100
0
0
20
40
60
Cantidad de carreras perdidas

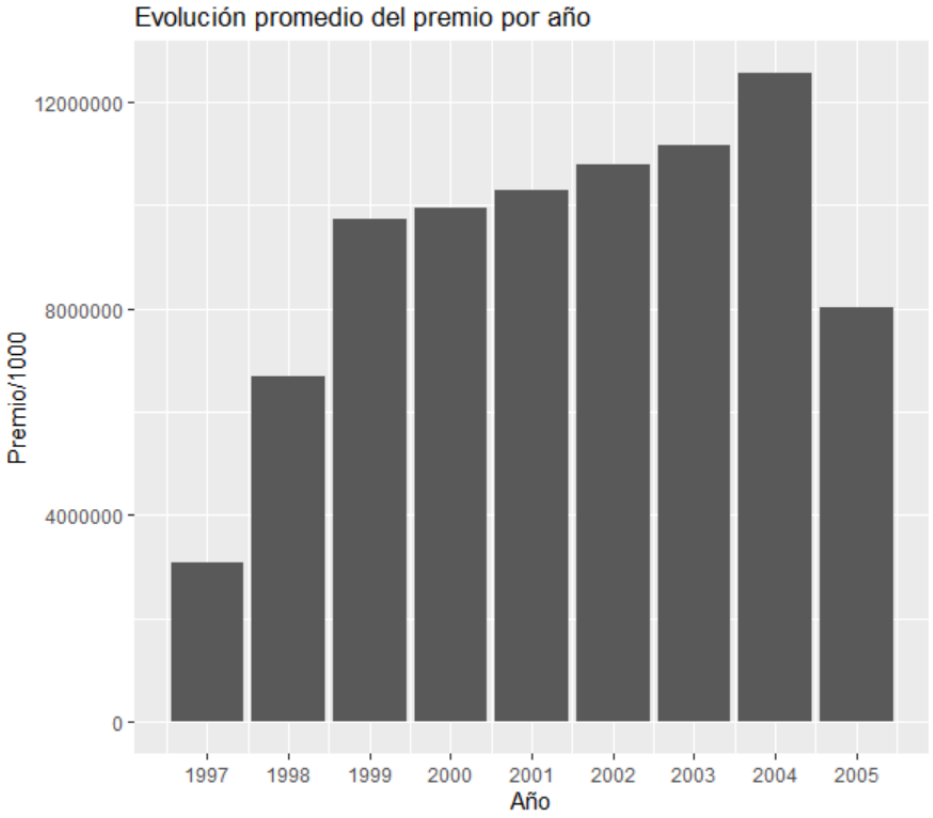

Distribución del premio
0.0000020
0.0000015
0.0000010
0.0000005
0.0000000
Density
500000
1000000
1500000
2000000
Prize

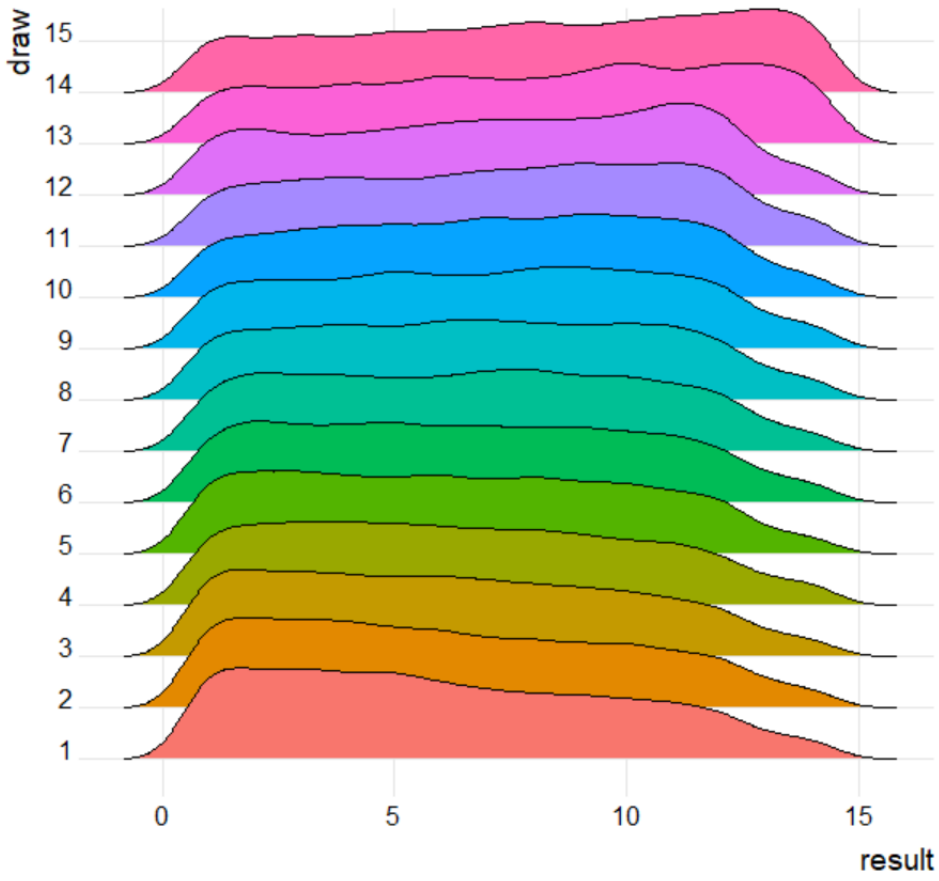

Distribución del Weight Ratio
50
40
30
20
10
0
Density
0.09
0.11
0.13
0.15
Weight Ratio de los caballos ganadores

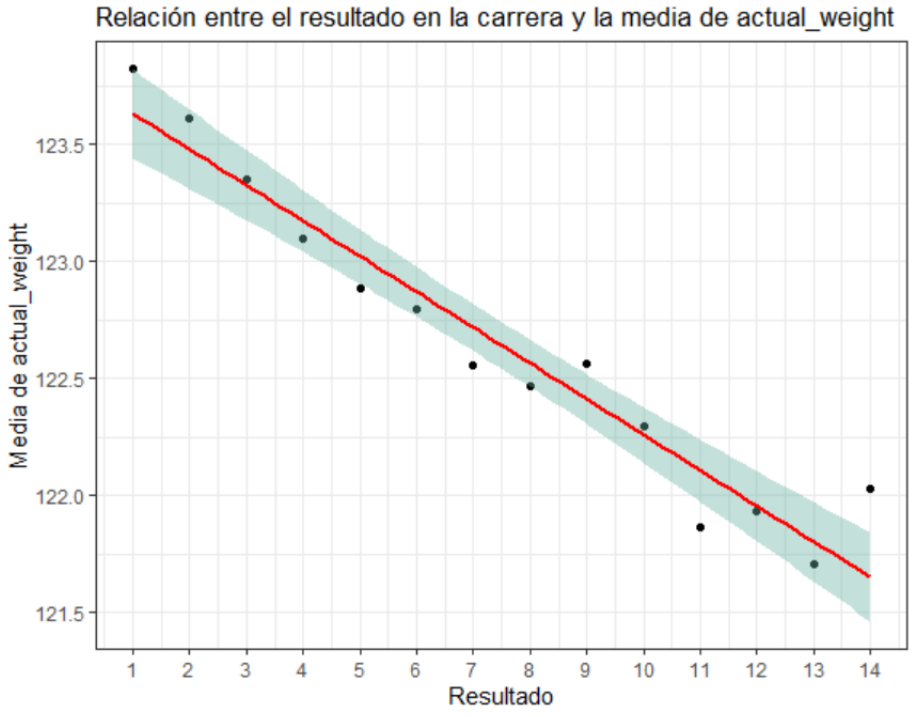

Distribución del Weight Ratio
Density
50
40
30
20
10
0
0.09
0.11
0.13
0.15
Weight Ratio de los caballos perdedores

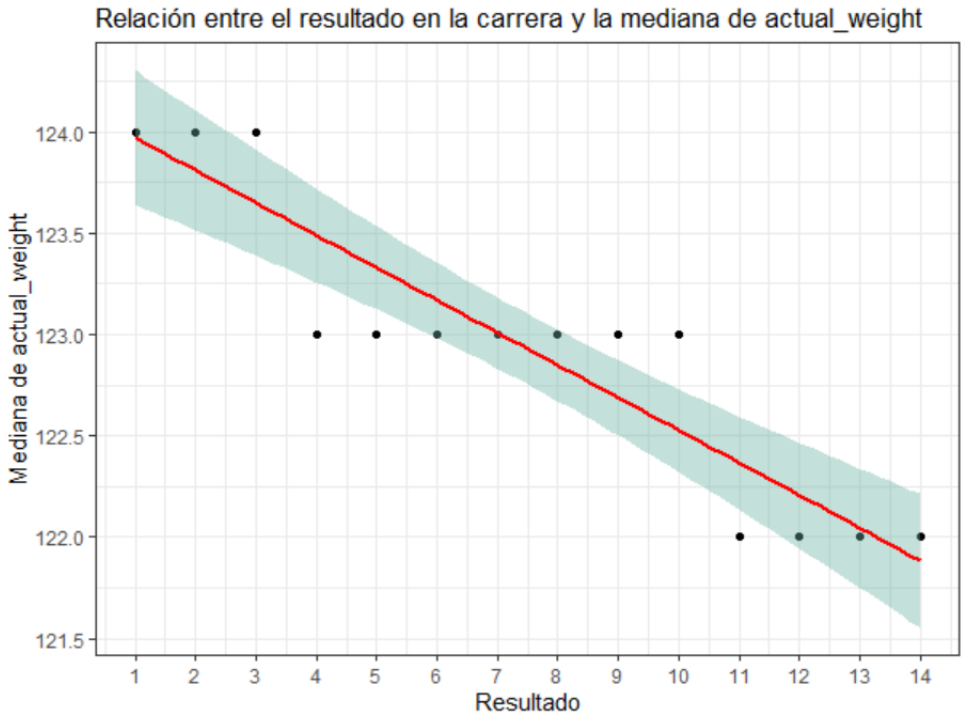

Evolución promedio del premio por día
Premio/1000
1500
1300
1100
900
0
10
20
30
Days

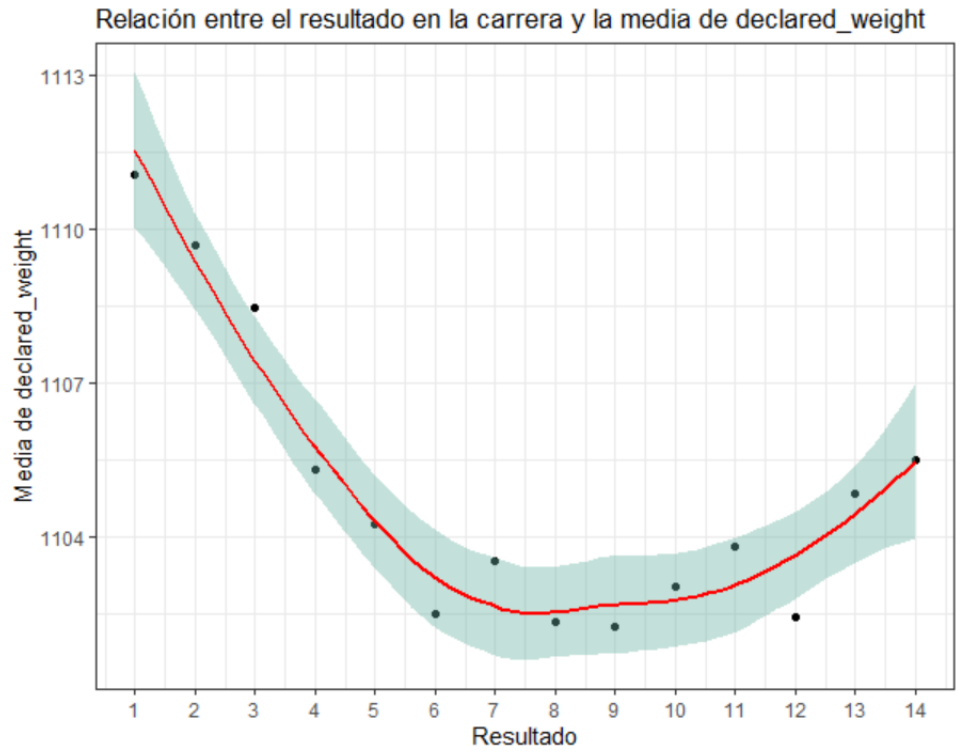

Evolución promedio del premio por mes
Premio/1000
1750
1500
1250
1000
750
1
2
3
4
5
6
7
8
9
10
11
12
Mes

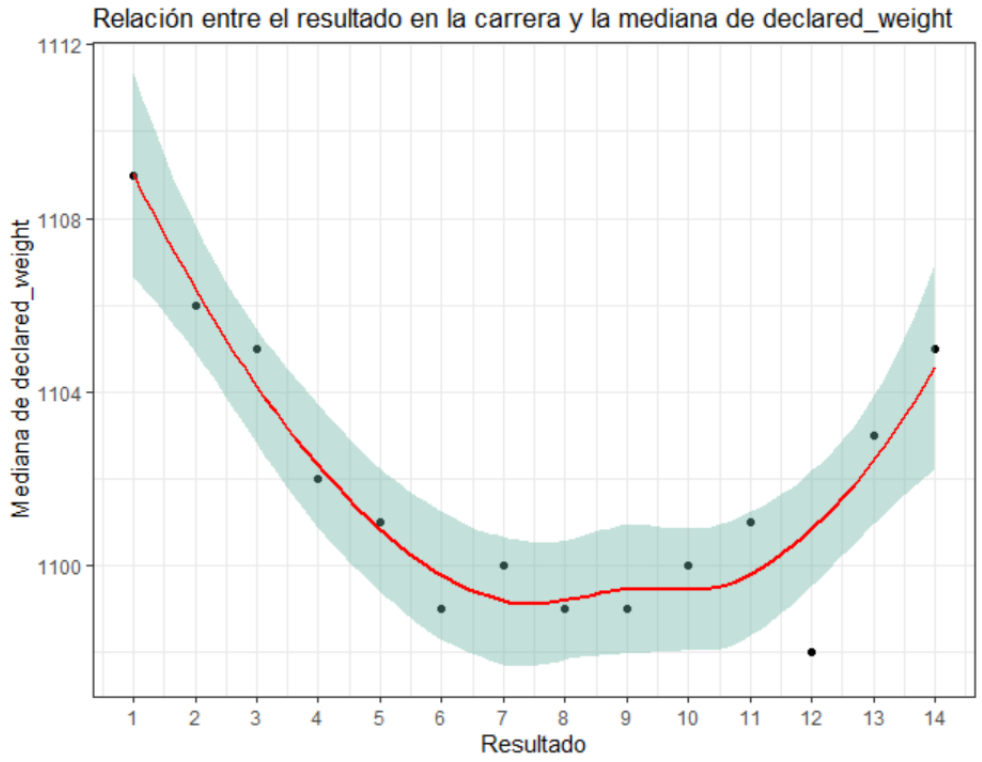

Evolución promedio del premio por año
Premio/1000
12000000
8000000
4000000
0
1997
1998
1999
2000
2001
2002
2003
2004
2005
Año

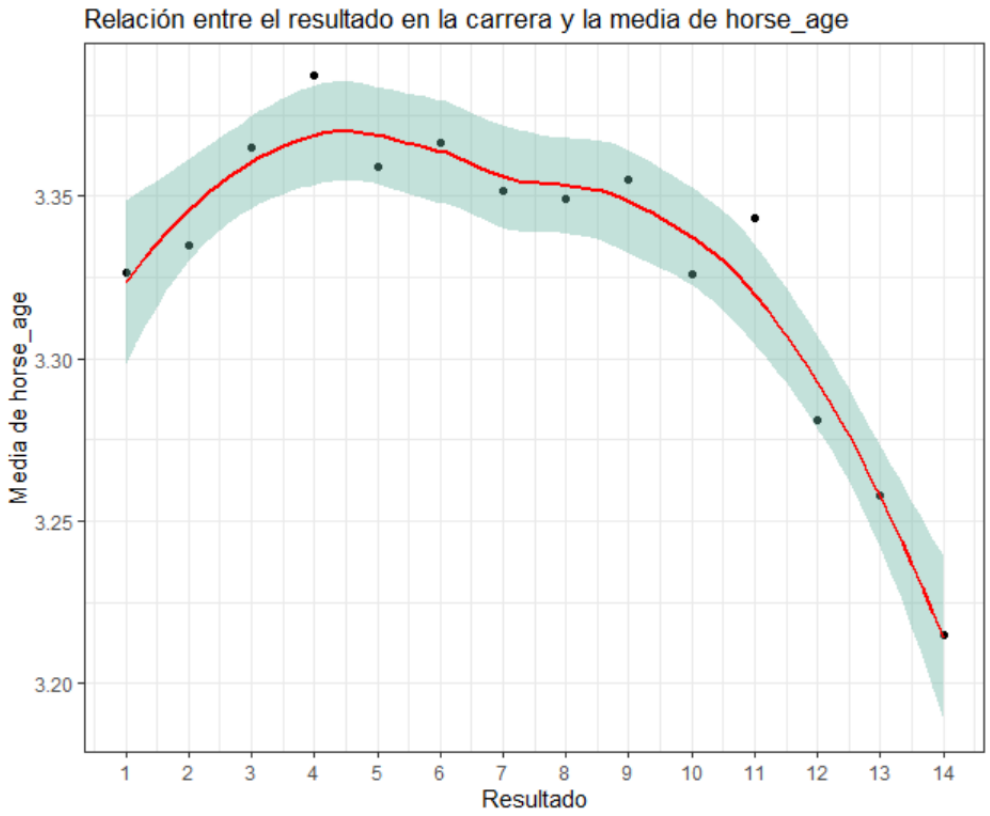

draw
15
14
13
12
11
10
9
8
7
6
5
4
3
2
1
0
5
10
15
result

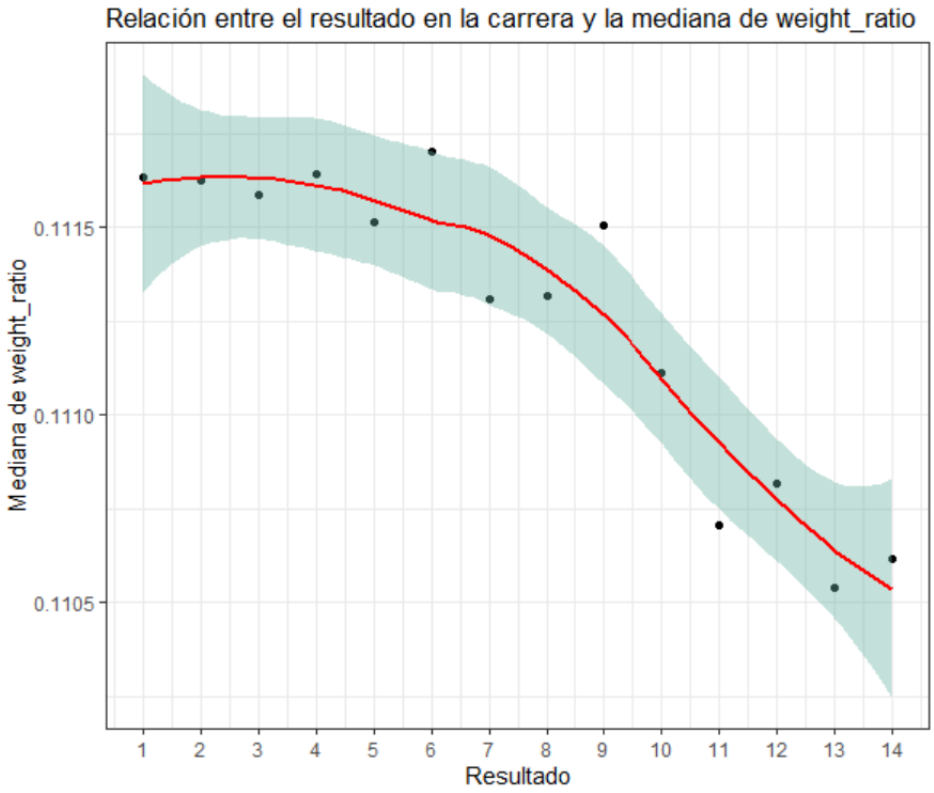

Relación entre el resultado en la carrera y la media de actual_weight
Media de actual_weight
123.5
123.0
122.5
122.0
121.5
1
2
3
4
5
6
7
8
9
10
11
12
13
14
Resultado

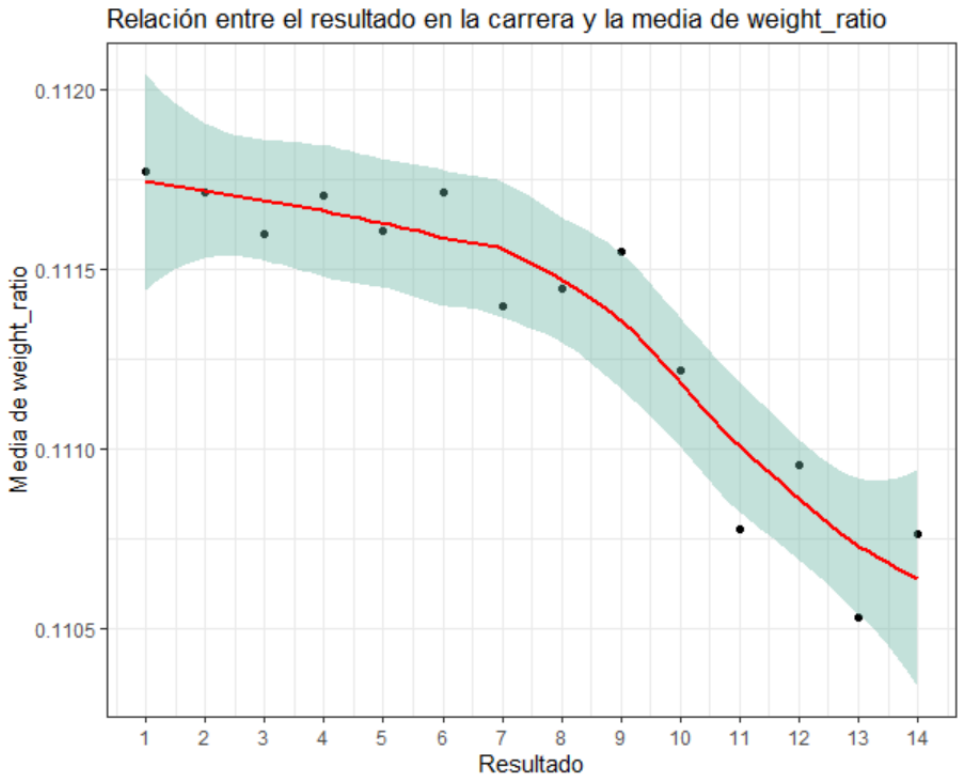

Relación entre el resultado en la carrera y la mediana de actual_weight
Mediana de actual_weight
124.0
123.5
123.0
122.5
122.0
121.5
1
2
3
4
5
6
7
8
9
10
11
12
13
14
Resultado

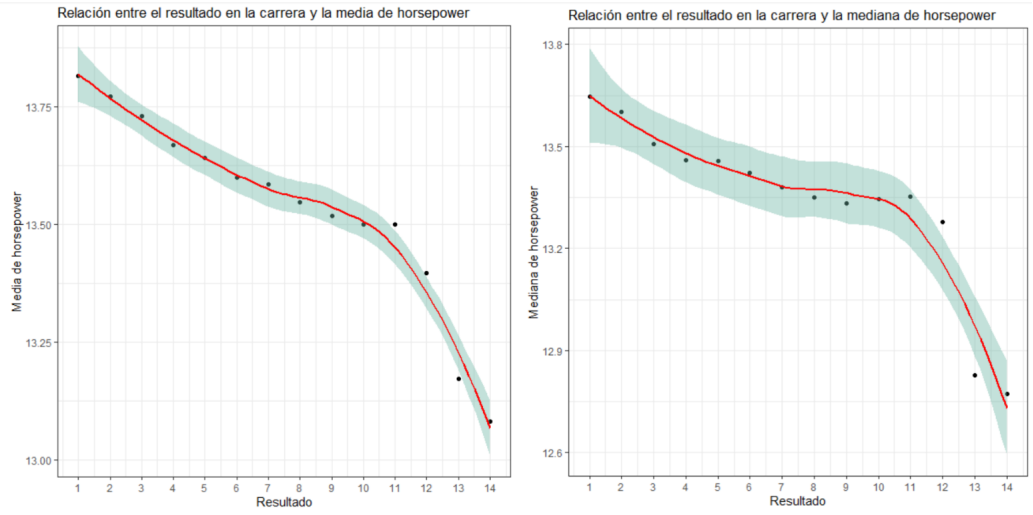

Relación entre el resultado en la carrera y la media de declared_weight
Media de declared_weight
1113
1110
1107
1104
1
2
3
4
5
6
7
8
9
10
11
12
13
14
Resultado

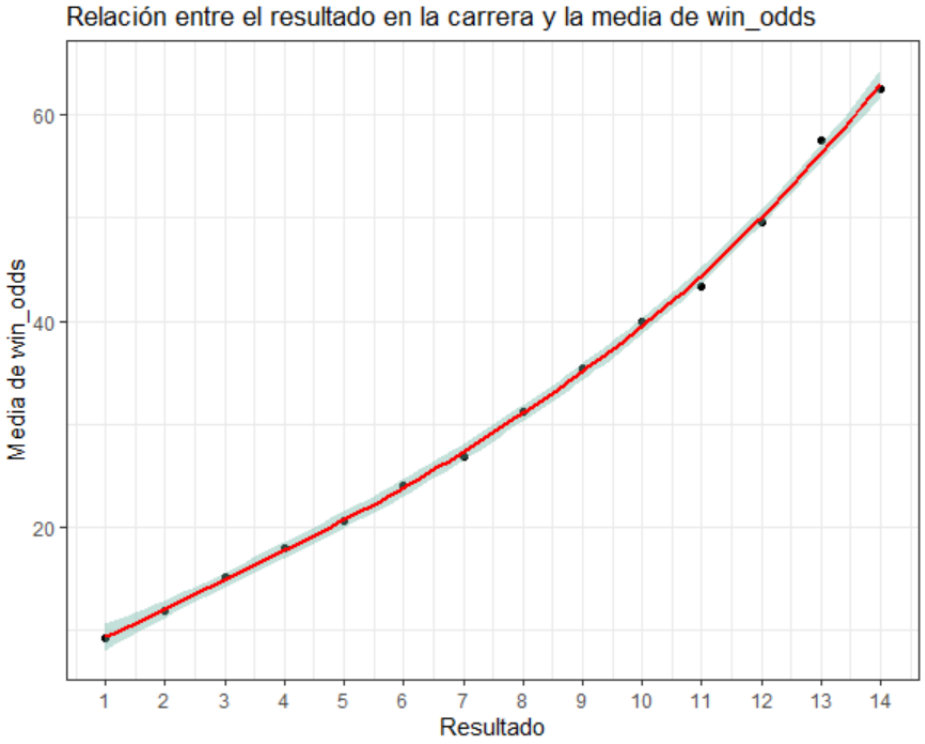

Relación entre el resultado en la carrera y la mediana de declared_weight
Mediana de declared_weight
1112
1108
1104
1100
1 2 3 4 5 6 7 8 9 10 11 12 13 14
Resultado

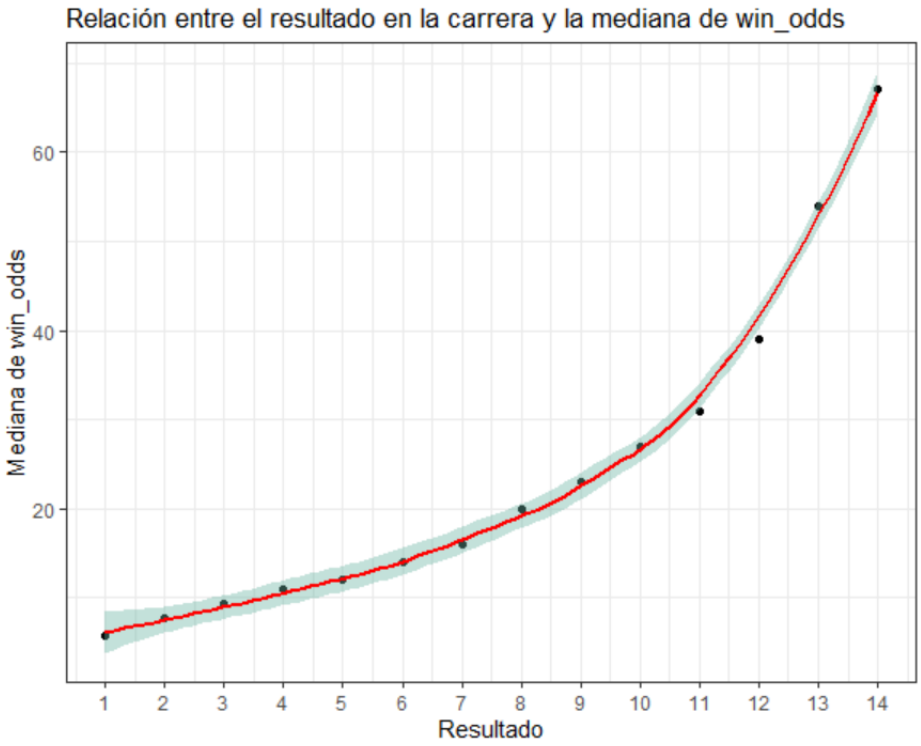

Relación entre el resultado en la carrera y la media de horse_age
Media de horse_age
3.35
3.30
3.25
3.20
1 2 3 4 5 6 7 8 9 10 11 12 13 14
Resultado

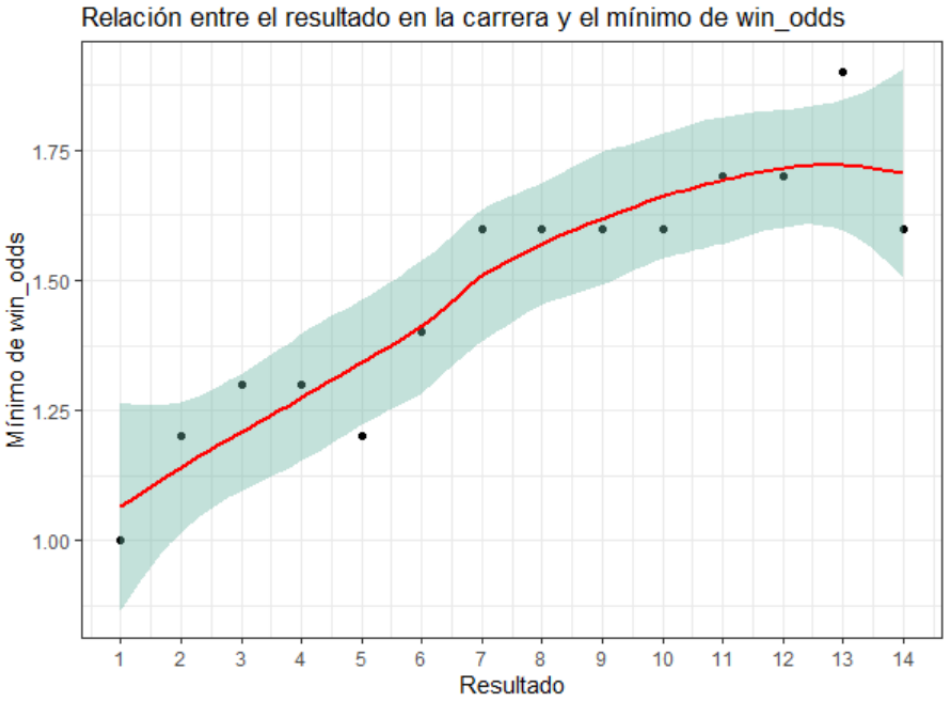

Relación entre el resultado en la carrera y la mediana de weight_ratio
Mediana de weight_ratio
0.1115
0.1110
0.1105
1
2
3
4
5
6
7
8
9
10
11
12
13
14
Resultado


Relación entre el resultado en la carrera y la media de weight_ratio
Media de weight_ratio
0.1120
0.1115
0.1110
0.1105
1
2
3
4
5
6
7
8
9
10
11
12
13
14
Resultado


Relación entre el resultado en la carrera y la media de horsepower
Media de horsepower
13.75
13.50
13.25
13.00
1 2 3 4 5 6 7 8 9 10 11 12 13 14
Resultado
Relación entre el resultado en la carrera y la mediana de horsepower
Mediana de horsepower
13.8
13.5
13.2
12.9
12.6
1 2 3 4 5 6 7 8 9 10 11 12 13 14
Resultado


Relación entre el resultado en la carrera y la media de win_odds
Media de win_odds
60
40
20
1 2 3 4 5 6 7 8 9 10 11 12 13 14
Resultado


Relación entre el resultado en la carrera y la mediana de win_odds
Mediana de win_odds
60
40
20
1
2
3
4
5
6
7
8
9
10
11
12
13
14
Resultado


Relación entre el resultado en la carrera y el mínimo de win_odds
Mínimo de win_odds
1.75
1.50
1.25
1.00
1
2
3
4
5
6
7
8
9
10
11
12
13
14
Resultado