Grado en Estadística

TRABAJO FIN DE GRADO

Modelos de predicción en pruebas de motociclismo

Tamara Royán González

Rafael Pino Mejías
Estadística e Investigación Operativa

Sevilla, Junio de 2020

Índice general

Agradecimientos . . . III
Resumen . . . IV
Abstract . . . V
Índice de Figuras . . . VI
Índice de Cuadros . . . VII

1. Descripción del problema y los datos **1**
1.1. Depuración . . . 2
1.1.1. Variables . . . 3
1.2. Partición . . . 3

2. Selección y transformación de variables **4**
2.1. Versión 1 . . . 5
2.2. Versión 2 . . . 5
2.3. Versión 3 . . . 5
2.4. Versión 4 . . . 6

3. Regresión Lineal Múltiple **7**
3.0.1. Estimación de mínimos cuadrados . . . 8
3.1. Modelo . . . 8

4. Árboles **11**
4.0.1. Ventajas . . . 11
4.0.2. Desventajas . . . 12
4.1. Árboles de predicción . . . 12
4.1.1. Recursive Binary Splitting . . . 13
4.1.2. Pruning: poda del árbol . . . 13
4.2. Modelo . . . 15

5. Random Forest **17**
5.0.1. Construcción de un modelo Random Forest . . . 17
5.0.2. Ventajas . . . 18
5.0.3. Desventajas . . . 18
5.1. Modelo . . . 18

6. Boosting **19**
6.0.1. Diferencias entre bagging y boosting . . . 20
6.0.2. Comparación Random Forest y Boosting . . . 20
6.1. Modelo . . . 21

7. Deep Learning **22**
7.0.1. Redes neuronales . . . 22
7.1. Modelo . . . 23

8. Análisis de resultados **24**
8.1. Comparativa . . . 26
8.1.1. Bondad de ajuste . . . 26
8.1.1.1. Gráficos . . . 29
8.1.2. Importancia de variables . . . 41
8.1.2.1. Gráficos . . . 45
8.1.3. Conclusión final . . . 49

A. Código en R **50**
A.1. Tratamiento de datos . . . 50
A.2. Conjuntos test y entrenamiento . . . 53
A.3. Selección y transformación de variables . . . 53
A.3.1. Versión 1 . . . 53
A.3.2. Versión 2 . . . 53
A.3.3. Versión 3 . . . 54
A.3.4. Versión 4 . . . 54
A.4. Función bondad de ajuste . . . 54
A.5. Modelo regresión . . . 55
A.5.1. Versión 1 . . . 55
A.5.2. Versión 2 . . . 56
A.5.3. Versión 4 . . . 56
A.6. Modelo árboles . . . 57
A.6.1. Versión 1 . . . 57
A.6.2. Versión 2 . . . 58
A.6.3. Versión 4 . . . 58
A.7. Modelo Random Forest . . . 59
A.7.1. Codificación variables categóricas . . . 59
A.7.2. Versión 1 . . . 60
A.7.3. Versión 2 . . . 60
A.7.4. Versión 4 . . . 61
A.8. Modelo Boosting . . . 62
A.8.1. Codificación variables categóricas . . . 62
A.8.2. Configuración de parámetros . . . 62
A.8.3. Versión 1 . . . 65
A.9. Modelo Deep Learning . . . 66
A.9.1. Codificación variables categóricas . . . 66
A.9.2. Versión 1 . . . 67
A.9.3. Versión 2 . . . 68

B. Gráficas **70**
B.1. Boosting . . . 70

Bibliografía **75**

Agradecimientos

Me gustaría agradecer y dedicar estas páginas especialmente a mis padres, Inma y Pepe, por el apoyo y el amor incondicional durante todos estos años; a mi pareja, Ángel, por su paciencia; y a mi tutor, Rafael, por su ayuda durante el desarrollo de este trabajo.

Por supuesto, no me olvido de todos aquellos docentes y profesionales que se cruzaron en mi camino e hicieron de mí lo que soy hoy: muchas gracias.

Resumen

En las siguientes páginas el lector podrá ver diferentes métodos para la predicción de resultados en pruebas de motociclismo, partiendo de datos históricos de las carreras disputadas en las diferentes categorías desde la temporada de 2005 hasta la de 2018.

Primero se estudiarán modelos de regresión, seguirá el estudio con árboles y Random Forest (Bagging) y se completará con Deep Learning (redes neuronales) y Boosting.

Para poder construir dichos modelos, antes he tenido que depurar los datos y he podido darme cuenta de los verdaderos obstáculos y lo tedioso que resulta lograr un conjunto de datos ordenado y fácilmente accesible. Con la ayuda de Excel y R, que es el lenguaje de programación con el que he trabajado, he conseguido estos objetivos.

También es importante comentar que para cada modelo se contarán con tres versiones diferentes del mismo, que habrán sido determinadas previamente seleccionando y transformando las variables del estudio.

Finalmente se recogerán las conclusiones obtenidas y se expondrán todos los resultados conjuntamente para hacer un análisis global.

En los Anexos se recogen todo el código usado en R para la elaboración del trabajo, tanto depuración de base de datos como implementación de cada uno de los métodos anteriormente mencionados, y algunas gráficas aclaratorias.

Por tanto, el trabajo podría dividirse en seis fases fundamentales:

- Recolección, limpieza y tratamiento de datos.
- Selección y transformación de variables.
- Construcción de los modelos y entrenamiento de los mismos.
- Obtención de predicciones.
- Evaluación de los modelos con las predicciones obtenidas.
- Ranking con la importancia de las variables de cada modelo.

Abstract

In the following pages, you will be able to see different methods to predict motorcycle competition results, based on historical data of the races in the different categories from 2005 to 2018.

First of all, regression models will be studied, it will continue with the study using trees and Random Forest (Bagging) and it will be completed with Deep Learning (neural networks) and Boosting.

In order to construct such models, I have to data debugging and it's when I realize the real obstacles and how tedious it is to achieve an easily accessible data base. With the help of Excel and R, which is the programming language I have worked with, I have achieved these goals.

It's also important knowing that for each model you will see three different versions of it, which would have been determined previously by selecting and transforming our variables.

Finally the conclusions obtained will be collected and all the results will be presented together to make a global analysis.

The Annexes include all the code used in R for the elaboration of the study, both database debugging and implementation of each of the above mentioned methods, and some explicative graphs.

Therefore, the study could be split into six fundamental phases:

- Collection, cleaning and data processing.
- Selection and transformation of our variables.
- Model construction and training.
- Obtaining predictions.
- Evaluation of the models with the predictions obtained.
- Ranking with the variable importance in each model.

Índice de figuras

3.1. Análisis Regresión Versión 1 9
3.2. Análisis Regresión Versión 2 9
3.3. Análisis Regresión Versión 4 10

4.1. Árbol podado Versión 1 15
4.2. Árbol podado Versión 4 16

7.1. Ejemplo estructura red neuronal 23

8.1. Bondad de Ajuste Entrenamiento 26
8.2. Bondad de Ajuste Entrenamiento - Zoom 27
8.3. Bondad de Ajuste Test 28
8.4. Gráficos bondad de ajuste Regresión-Versión 1 29
8.5. Gráficos bondad de ajuste Regresión-Versión 2 30
8.6. Gráficos bondad de ajuste Regresión-Versión 4 31
8.7. Gráficos bondad de ajuste Árboles-Versión 1 32
8.8. Gráficos bondad de ajuste Árboles-Versión 2 33
8.9. Gráficos bondad de ajuste Árboles-Versión 4 34
8.10. Gráficos bondad de ajuste Random Forest-Versión 1 35
8.11. Gráficos bondad de ajuste Random Forest-Versión 2 36
8.12. Gráficos bondad de ajuste Random Forest-Versión 4 37
8.13. Gráficos bondad de ajuste Deep Learning-Versión 1 38
8.14. Gráficos bondad de ajuste Deep Learning-Versión 2 39
8.15. Gráficos bondad de ajuste Boosting 40
8.16. Gráficos Ranking Regresión 45
8.17. Gráficos Ranking Árboles 46
8.18. Gráficos Ranking Random Forest 47
8.19. Gráficos Ranking Deep Learning 48
8.20. Gráfico Ranking Boosting 48

B.1. Tune Max Tree Depth 70
B.2. Tune Minimum Sum of Instance Weight 71
B.3. Tune Subsample Ratio of Columns 71
B.4. Tune Minimum Loss Reduction 72
B.5. Tune Shrinkage 72

Índice de cuadros

1.1. Variables 3

6.1. Hiperparámetros Boosting 21

8.1. Predicciones 25
8.2. Bondad de Ajuste Entrenamiento 26
8.3. Bondad de Ajuste Test 27
8.4. Ranking Versión 1 41
8.5. Ranking Deep Learning-Versión 1 41
8.6. Ranking Versión 2 42
8.7. Ranking Deep Learning-Versión 2 43
8.8. Ranking Versión 4 43
8.9. Ranking Boosting 44

Capítulo 1

Descripción del problema y los datos

En los últimos años han aumentado de manera más que considerable las plataformas de apuestas deportivas. Lo que comenzó orientado exclusivamente al mundo del fútbol, se acabó extendiendo al resto de deportes. Tanto es así que en el motociclismo se analizan, ya no sólo los pilotos vencedores de las diferentes categorías, sino puntos tales como el tiempo de la vuelta rápida en carrera, la ventaja que conseguirá el vencedor y un largo etcétera.

Es por ello que consideré interesante aplicar técnicas estadísticas para poder predecir ciertos resultados, o tener una idea de cómo va a ir el campeonato tomando datos de temporadas anteriores.

Para ello se hacía imprescindible una completa base de datos donde se recogieran muchas variables a estudiar, para poder hacer un estudio de lo más completo.

El primer obstáculo con el que me encuentro es el hecho de que los pilotos no siempre permanecen en la misma fábrica, equipo ni categoría. Además, cada año los equipos implementan cambios que en algunos casos funcionan de manera positiva consiguiendo mejorar los resultados y otras veces todo lo contrario. Tener en cuenta todos estos detalles resultará difícil, pero iremos viendo cómo ir solucionando o, en su defecto, paliando sus efectos en el estudio.

Cuando busqué en la red acerca de estudios parecidos no encontré muchos artículos en este ámbito de los deportes de motor, pero pude basarme en una base de datos[1] publicada junto a un artículo[2] para elaborar la mía propia. La información de los enlaces a los mismos está incluida en la *Bibliografía*.

Decidí que trabajaría con los datos desde el 2005 al 2018, por lo que acabé obteniendo una base de datos con 32 variables y 20989 observaciones.

[1] https://github.com/Vishwacorp/motogp_regression

[2] https://medium.com/@Vishwacorp/predicting-motogp-race-finish-times-using-linear-regression-6dc3d1e1c2a4

1.1. Depuración

El primer paso fue analizar todas las variables con las que contaba y determinar cuáles serían útiles en el estudio y aquellas que no aportaban información de interés, tales como el nombre completo del circuito o el código de identificación del mismo, la fecha en la que se realizó la prueba, los puntos obtenidos por cada piloto[3] y los tiempos que separaban a un piloto de otro, porque mi objetivo era estudiar el tiempo total a la llegada a meta.

También eliminé la velocidad media de la categoría MotoGP y las distancias recorridas por cada categoría en cada circuito, más adelante calcularé éstas últimas para obtener resultados más fiables y exactos.

Tampoco trabajaré con la posición de los pilotos (también podría ordenar los tiempos y obtenerla fácilmente) y las vueltas completadas por cada uno.

Para identificar al piloto uní su dorsal y su nombre en una nueva variable.

Para que no hubiera mala interpretación de los datos de temporadas más lejanas en el tiempo, consideré que la categoría de 125cc se tomaría como Moto 3 y la de 250cc como Moto2, aunque realmente las actuales motos no tengan esta equivalencia.

- *MotoGP: se trata de la "categoría reina" del campeonato, pues en ella compiten las motos de mayor cilindrada. Desde mediados de los 70 hasta el año 2002, la categoría permitía una cilindrada de 500cc sin tener en cuenta si el motor era de dos o cuatro tiempos.Las nuevas reglas permitieron a los fabricantes elegir entre motos de dos tiempos (hasta 500c) y motos de cuatro tiempos (hasta 990cc).*
- *Moto2: es el escalafón intermedio del Mundial y se caracteriza por la utilización de motos de 600cc, con motor de cuatro tiempos. Estos motores entraron en liza para la temporada 2010, fecha en la que desaparecieron de esta categoría las motocicletas de 250cc con motor de dos tiempos.*
- *Moto3: Al igual que en las categorías superiores, los motores de dos tiempos dejaron paso a los de cuatro. A partir de 2012 las motos de 125cc han sido sustituidas por motos de cuatro tiempos monocilíndricas de hasta 250cc.*

Para que no hubiera duplicidades en las fábricas ni los nombres de equipo, depuré los espacios al principio y al final de dichos datos. Y procedí de manera análoga para las nacionalidades.

En el caso de los tiempos de llegada, los convertí a milisegundos para que pudiera estudiarlos de manera continua.

Como había tres tipos de carreras: RAC1, RAC2 y RAC3, según si se había disputado de una vez o se había dividido en dos partes, denoté la nueva variable Tipo para codificarlo.

Finalmente traduje las variables de mi base de datos y los niveles de alguna variable.

Seguí depurando mi base de datos y continué por eliminar los niveles de Nacionalidades y Fábricas que no eran representativos.

Al analizarla observé que había muchos datos vacíos de pilotos que se habían caído y no tenían tiempos finales, luego opté por quedarme sólo con los casos completos y reduje a 20212 observaciones.

[3]En este **enlace** se puede ver cómo se realiza el reparto de los puntos una vez tengamos a los pilotos ordenados, lo cual siempre puedo hacer teniendo los tiempos finales.

1.1.1. Variables

Cuadro 1.1: Variables

Variables	Estructura datos	Definición
Fecha	Entero	Año del evento
GranPremio	Factor con 25 niveles	Nombre del Gran Premio disputado
Categoria	Factor con 3 niveles	Categoría (MotoGP, Moto2, Moto3) a la que pertenecía el piloto
Tipo	Factor con 3 niveles	Carrera única: 1; Carrera dividida: 2 primera parte, 3 segunda
Condiciones_Meteo	Factor con 3 niveles	Clima de la carrera: Seco, Mojado, Mixto
Temp_Pista	Numérico	Temperatura del asfalto, en grados celsius (C)
Temp_Aire	Numérico	Temperatura del aire, en grados celsius (C)
Humedad	Numérico	Humedad: entre 0 (0 %) y 1 (100 %)
Piloto	Factor con 1088 niveles	Indica el dorsal y el nombre del piloto
Nacionalidad	Factor con 43 niveles	Nacionalidad del piloto
Equipo	Factor con 833 niveles	Equipo al que pertenecía el piloto en la carrera
Fabrica	Factor con 85 niveles	Marca de la moto que llevaba el piloto en la carrera
Tiempo	Numérico	Tiempo de llegada a línea de meta de cada piloto
Long_pista	Entero	Longitud de la pista en metros
Vueltas	Entero	Número de vueltas a completar en cada circuito y categoría
Distancia_comp	Entero	Distancia completa a recorrer en cada circuito y categoría
Curvas_izq	Entero	Número de curvas a izquierda
Curvas_dcha	Entero	Número de curvas a derecha
Anchura	Entero	Anchura de la pista en metros
MaxRecta	Entero	Longitud de la recta más larga de la pista en metros

1.2. Partición

Para aplicar los diferentes modelos he realizado una partición en mi base de datos, considerando los conjuntos entrenamiento, con el 75 % de los datos, y test, con el 25 % restante.

Reuniendo finalmente en el conjunto entrenamiento 15172 datos de mis 20 variables y en el conjunto test 5040 datos de las mismas 20 variables.

Capítulo 2

Selección y transformación de variables

Este paso será común para todos los modelos, puesto que los resultados aquí obtenidos serán aplicables posteriormente.

Usaré el método paso a paso (*stepwise*) para seleccionar los predictores de mi estudio. Este método emplea criterios matemáticos para decidir qué predictores contribuyen significativamente al modelo y en qué orden se introducen. Dentro de este método se diferencian tres estrategias:

- Dirección **forward**: El modelo inicial no contiene ningún predictor. A partir de este se generan todos los posibles modelos introduciendo una sola variable de entre las disponibles. Aquella variable que mejore en mayor medida el modelo se selecciona. A continuación se intenta incrementar el modelo probando a introducir una a una las variables restantes. Si introduciendo alguna de ellas mejora, también se selecciona. En el caso de que varias lo hagan, se selecciona la que incremente en mayor medida la capacidad del modelo. Este proceso se repite hasta llegar al punto en el que ninguna de las variables que quedan por incorporar mejore el modelo.
- Dirección **backward**: El modelo se inicia con todas las variables disponibles incluidas como predictores. Se prueba a eliminar una a una cada variable, si se mejora el modelo, queda excluida. Este método permite evaluar cada variable en presencia de las otras.
- Doble o **mixto**: Se trata de una combinación de la selección *forward* y *backward*. Se inicia igual que el forward pero tras cada nueva incorporación se realiza un test de extracción de predictores no útiles como en el backward. Presenta la ventaja de que si a medida que se añaden predictores, alguno de los ya presentes deja de contribuir al modelo, se elimina.

El que se ha usado en este estudio es esta última variante: **método paso a paso mixto**.

El método paso a paso requiere de algún criterio matemático para determinar si el modelo mejora o empeora con cada incorporación o extracción. Existen varios parámetros empleados, de entre los que destacan el C_p, AIC, BIC y $R^2 - ajustado$, cada uno de ellos con ventajas e inconvenientes. El método *Akaike (AIC)* tiende a ser más restrictivo e

introducir menos predictores que el $R^2 - ajustado$. Para un mismo conjunto de datos, no todos los métodos tienen porqué concluir en un mismo modelo.

En el caso de variables categóricas, si al menos uno de sus niveles es significativo, se considera que la variable lo es. Cabe mencionar que, si una variable se excluye del modelo como predictor, significa que no aporta información adicional al modelo, pero sí puede estar relacionada con la variable respuesta.

Procederé al estudio de 4 versiones diferentes de selección y transformación de las variables de mi estudio:

2.1. Versión 1

En esta ocasión no realizaré ninguna transformación. He obtenido las variables con las que construiré mis modelos para poder predecir el Tiempo, con un AIC de 403861.4:

- *Fecha, GranPremio, Categoria, Tipo, Condiciones_Meteo, Temp_Pista, Temp_Aire, Fabrica, Distancia_comp, Long_pista y MaxRecta.*

2.2. Versión 2

Ahora transformaré variables predictoras y objetivo en un mismo paso. Busco transformaciones para las variables cuantitativas además de la objetivo: Tiempo, Fecha, Temp_Pista, Temp_Aire, Humedad, Long_pista, Distancia_comp, Curvas_izq, Curvas_dcha, Anchura y MaxRecta. Y obtengo lo siguiente:

- *$Tiempo^3$, Fecha, sqrt(Temp_Pista), log(Temp_Aire), Humedad, 1/Long_pista, $Distancia\_comp^4$, log(Curvas_izq), Curvas_dcha, log(Anchura), log(MaxRecta), Vueltas, GranPremio, Categoria, Tipo, Condiciones_Meteo, Piloto, Nacionalidad, Equipo y Fabrica.*

Tendré que deshacer el cambio elevando a 1/3, pues mi variable objetivo, Tiempo, ha sido transformada previamente.

Ahora hago la selección de variables y obtengo las variables con las que construir mis modelos para poder predecir el Tiempo, con un AIC de 1309721:

- *Fecha, sqrt(Temp_Pista), log(Temp_Aire), Distancia_comp, log(Curvas_izq), log(MaxRecta), Vueltas, GranPremio, Categoria, Tipo, Condiciones_Meteo, Fabrica y Nacionalidad.*

2.3. Versión 3

Ahora primero transformo las variables predictoras y luego haré lo mismo para la variable objetivo. Busco transformaciones para las variables cuantitativas: Fecha, Temp_Pista, Temp_Aire, Humedad, Long_pista, Distancia_comp, Curvas_izq, Curvas_dcha, Anchura y MaxRecta y obtengo el siguiente resultado:

- *Fecha, sqrt(Temp_Pista), log(Temp_Aire), Humedad, 1/Long_pista, $Distancia\_comp^4$, log(Curvas_izq), Curvas_dcha, log(Anchura), log(MaxRecta), Vueltas, GranPremio, Categoria, Tipo, Condiciones_Meteo, Piloto, Nacionalidad, Equipo y Fabrica.*

Una vez tengo estas transformaciones, repito el proceso para la variable objetivo y vuelvo a obtener que su transformación es la potencia de orden 3. Es decir, que obtengo exactamente el mismo resultado que en la Versión 2, por lo que no voy a repetir la selección de variables ya que el resultado sería el mismo de nuevo. En adelante sólo trabajaré con la Versión 2, simplificando así el estudio.

2.4. Versión 4

En esta ocasión transformaré sólo la variable objetivo, pero con el método Boxcox[1]. Es decir, tomando las transformaciones de las variables predictoras de la versión 2 (o bien de la versión 3, puesto que eran las mismas), se obtiene que la transformación en este caso para la variable objetivo es la potencia de orden 2, por lo que para deshacer el cambio en las predicciones tendré que aplicar la raíz cuadrada.

Realizamos ahora la selección de variables y obtengo, con un AIC de 858562.8:

- *Fecha, sqrt(Temp_Pista), log(Temp_Aire), Distancia_comp, log(Curvas_izq), log(MaxRecta), Vueltas, GranPremio, Categoria, Tipo, Condiciones_Meteo, Fabrica y Nacionalidad.*

[1]Las transformaciones de Box y Cox son una familia de transformaciones potenciales usadas en estadística para corregir sesgos en la distribución de errores, para corregir varianzas desiguales (para diferentes valores de la variable predictora) y principalmente para corregir la no linealidad en la relación (mejorar correlación entre las variables).

Capítulo 3

Regresión Lineal Múltiple

Uno de los principales objetivos de la Estadística es estudiar relaciones entre variables que describen situaciones reales. Para conseguirlo, es útil buscar una ecuación matemática o función que relacione dichas variables en estudio, es decir, construir un modelo estadístico que describa la situación real.

Sea (Y, $X_1, ..., X_p$) un vector aleatorio (p+1)-dimensional.

La función de $X_1, \ldots, X_p$ más cercana a Y en el sentido de mínimos cuadrados,

$$min_f E[(Y - f(X_1, ..., X_p))^2],$$

viene dada por

$$f(x_1, ..., x_p) = E(Y|X_1 = x_1, ..., X_p = x_p),$$

la curva regresión de Y sobre $X_1, ..., X_p$.

La regresión de Y sobre $X_1, \ldots, X_p$ es lineal si

$$E(Y|X_1 = x_1, ..., X_p = x_p) = \beta_0 + \beta_1 x_1 + ... + \beta_p x_p.$$

Incluso si la regresión no es lineal, podemos estar interesados en determinar

$$min_{f: f(x_1,...,x_p)=\beta_0+\beta_1 x_1+...+\beta_p x_p} E[(Y - f(X_1, ..., X_p))^2],$$

el hiperplano de mínimos cuadrados o hiperplano de regresión.

En ambos casos podemos escribir $Y = \beta_0 + \beta_1 X_1 + ... + \beta_p X_p + \epsilon$, con $E(\epsilon) = 0$. Supondremos que $var(\epsilon) = \sigma^2$ independientemente del valor del vector aleatorio $(X_1, ..., X_p)$. Esta hipótesis se denomina *homocedasticidad*.

3.0.1. Estimación de mínimos cuadrados

Sea $(y_1, x_{11}, ..., x_{1p}), ..., (y_n, x_{n1}, ..., x_{np})$ una m.a.s (i.i.d) de $(Y_1, X_1, ..., X_p)$,

$$y_i = \beta_0 + \beta_1 x_{i1} + ... + \beta_p x_{ip} + \epsilon_i, \quad 1 \leq i \leq n,$$

con $\epsilon_1, ..., \epsilon_n$ i.i.d., se tiene que

$$E(\epsilon_i) = 0, \quad 1 \leq i \leq n \quad y$$

$$var(\epsilon_i) = \sigma^2, \quad 1 \leq i \leq n \quad (homocedasticidad).$$

En particular,

$$cov(\epsilon_i, \epsilon_j) = 0, \quad \forall i \neq j.$$

Matricialmente

$$Y = X\beta + \epsilon,$$

donde $Y^t = (y_1, ..., y_n)$, $\epsilon^t = (\epsilon_1, ..., \epsilon_n)$, $\beta^t = (\beta_1, ..., \beta_p)$, $X = (x_1^t, x_2^t, ..., x_n^t)^t$ y $x_i^t = (1, x_{i1}, ..., x_{ip})$, $1 \leq i \leq n$.

El método de mínimos cuadrados estima $\beta_0, \beta_1, ...\beta_p$ mediante $\hat{\beta}_0, \hat{\beta}_1, ..., \hat{\beta}_p$ que minimizan la suma de los cuadrados de las diferencias entre los valores observados, y_i, y las estimaciones de la esperanza (condicional) de y_i,

$$\hat{y}_i = \hat{\beta}_0 + \hat{\beta}_1 x_{i1} + ... + \hat{\beta}_p x_{ip},$$

$$(\hat{\beta}_0, \hat{\beta}_1, ..., \hat{\beta}_p) = argmin_{\beta_0, \beta_1, ...\beta_p} \sum_{i=1}^{n} (y_i - \beta_0 - \beta_1 x_{i1} - ... - \beta_p x_{ip})^2.$$

Se tiene

$$\sum_{i=1}^{n} (y_i - \beta_0 - \beta_1 x_{i1} - ... - \beta_p x_{ip})^2 = (Y - X\beta)^t (Y - X\beta) = Y^t Y - 2Y^t X\beta + \beta^t X^t X\beta.$$

Derivando e igualando a cero, se obtiene el sistema de ecuaciones normales (SEN), $X^t X\beta = X^t Y$.

Si X es de rango total ($n \geq p + 1$ y ninguna columna es combinación lineal del resto), entonces X es invertible, y se obtiene la solución $\hat{\beta} = (X^t X)^{-1} X^t Y$.

A $\hat{y}_i = \hat{\beta}_0 + \hat{\beta}_1 x_{i1} + ... + \hat{\beta}_p x_{ip}$ se le denomina predicción o valor ajustado de y_i, $1 \leq i \leq n$.

A los valores $e_i = y_i - \hat{y}_i$, $1 \leq i \leq n$ se les denomina residuos.

3.1. Modelo

Para nuestro estudio en particular usaremos los mismos modelos que hemos obtenido en la fase de *Selección y transformación de variables.*

Aunque más adelante realizaremos el *Análisis de resultados*, podemos hacer algunos comentarios acerca de los modelos de regresión que hemos construido:

Figura 3.1: Análisis Regresión Versión 1

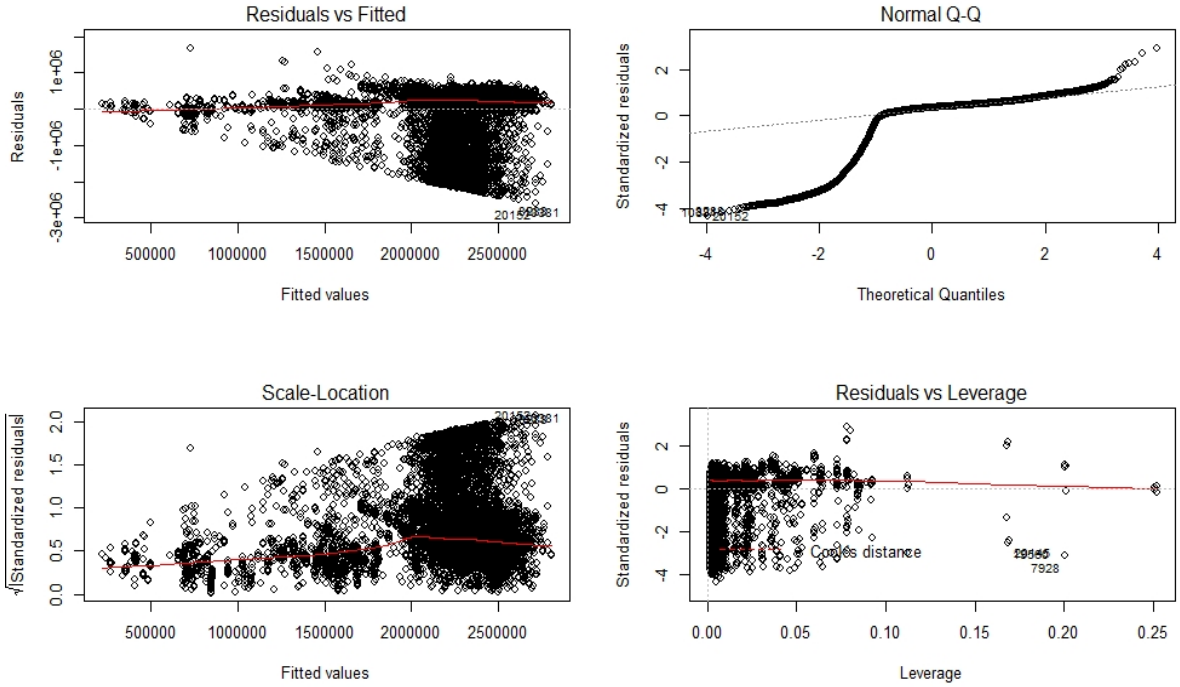


Figura 3.2: Análisis Regresión Versión 2

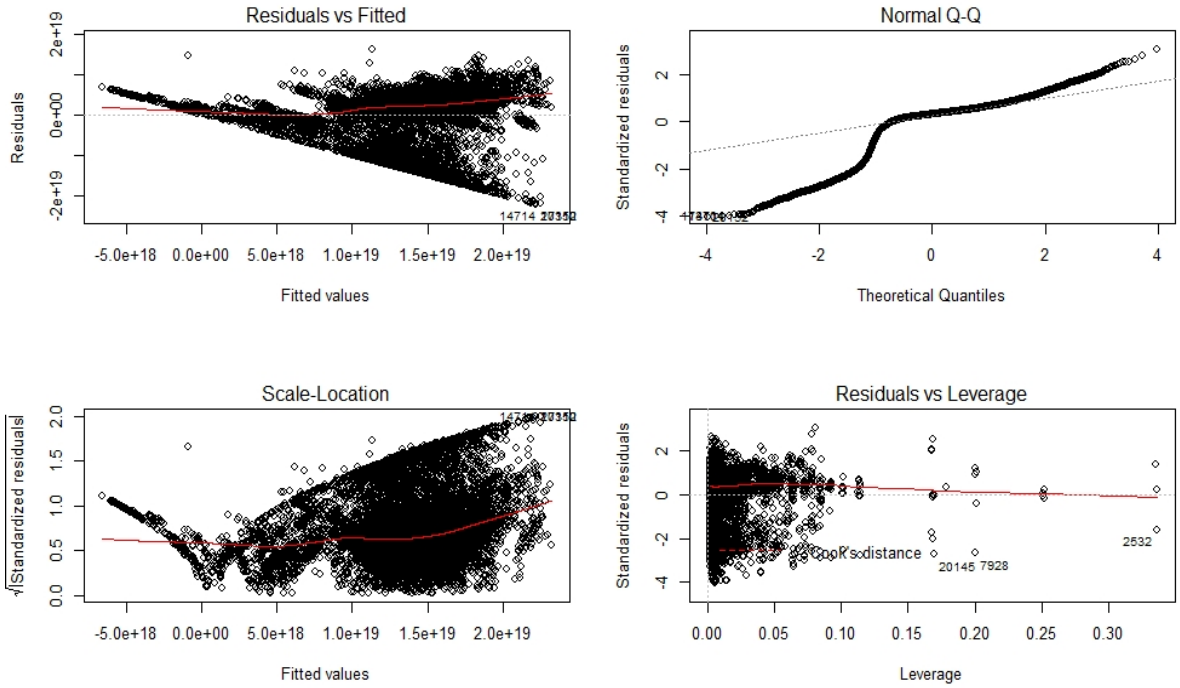
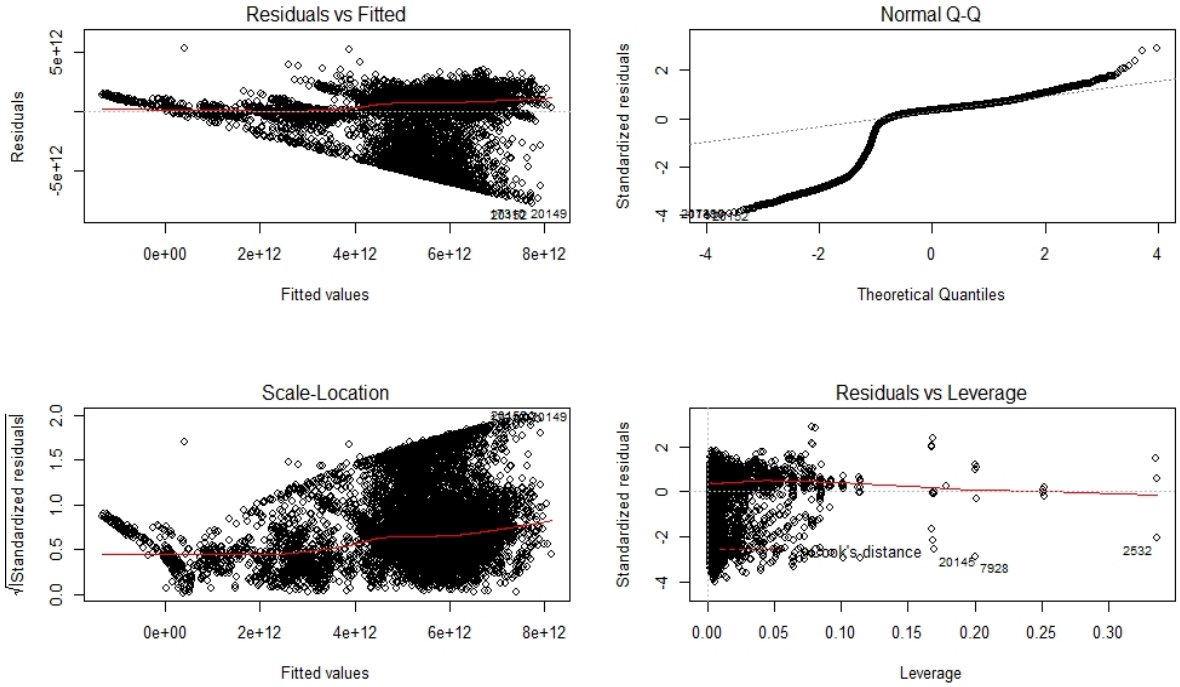


Figura 3.3: Análisis Regresión Versión 4



En todos los casos podemos intuir que la hipótesis de homocedasticidad no se cumple, ya que en el gráfico superior a la izquierda los puntos se distribuyen en forma de embudo. Ello sumado al hecho de que la línea roja del gráfico *Scale-Location* (gráfico inferior izquierdo) no es del todo horizontal nos indica que los modelos obtenidos no son del todo satisfactorios ni fiables.

Capítulo 4

Árboles

Los métodos predictivos como la regresión lineal o polinómica generan modelos globales en los que una única ecuación se aplica a todo el espacio muestral. Cuando el estudio implica múltiples predictores, que interaccionan entre ellos de forma compleja y no lineal, es muy difícil encontrar un modelo global que sea capaz de reflejar la relación entre las variables. Existen métodos de ajuste no lineal (*step functions, spline, etc*) que combinan múltiples funciones y que realizan ajustes locales, sin embargo, suelen ser difíciles de interpretar. Los métodos estadísticos basados en árboles engloban a un conjunto de técnicas supervisadas no paramétricas que consiguen segmentar el espacio de los predictores en regiones simples, dentro de las cuales es más sencillo manejar las interacciones.

Los árboles de decisión se pueden aplicar tanto a los problemas de regresión, como de clasificación, aunque en este caso consideremos el problema de regresión. Un árbol de regresión consiste en una serie de reglas de "reparto" o división del espacio definido por las variables predictoras.

4.0.1. Ventajas

- Los árboles son fáciles de interpretar aun cuando las relaciones entre predictores son complejas. Su estructura se asemejan a la forma intuitiva en que clasificamos y predecimos las personas, además, no se requieren conocimientos estadísticos para comprenderlos.
- Los modelos basados en un solo árbol se pueden representar gráficamente aun cuando el número de predictores es mayor de 3.
- Los árboles pueden manejar tanto predictores cuantitativos como cualitativos sin tener que crear variables dummy.
- Al tratarse de métodos no paramétricos, no es necesario que se cumpla ningún tipo de distribución específica.
- Por lo general, requieren mucha menos limpieza y pre procesado de los datos en comparación a otros métodos de aprendizaje estadístico.
- No se ven muy influenciados por outliers.
- Si para alguna observación, el valor de un predictor no está disponible, a pesar de no poder llegar a ningún nodo terminal, se puede conseguir una predicción empleando todas las observaciones que pertenecen al último nodo alcanzado. La precisión de la predicción se verá reducida pero al menos podrá obtenerse.

- Son muy útiles en la exploración de datos, permiten identificar de forma rápida y eficiente las variables más importantes.
- Son capaces de seleccionar predictores de forma automática.

4.0.2. Desventajas

- La capacidad predictiva de los modelos de regresión y clasificación basados en un único árbol es bastante inferior a la conseguida con otros modelos debido a su tendencia al overfitting. Sin embargo, existen técnicas más complejas que, haciendo uso de la combinación de múltiples árboles (bagging, Random Forest, boosting), consiguen mejorar en gran medida este problema.
- Cuando tratan con variables continuas, pierden parte de su información al categorizarlas en el momento de la división de los nodos. Por esta razón, suelen ser modelos que consiguen mejores resultados en clasificación que en regresión.
- La creación de las ramificaciones de los árboles se consigue mediante el algoritmo de *recursive binary splitting*. Este algoritmo identifica y evalúa las posibles divisiones de cada predictor acorde a una determinada medida (*RSS, Gini, entropía...*). Los predictores continuos o predictores cualitativos con muchos niveles tienen mayor probabilidad de contener, solo por azar, algún punto de corte óptimo, por lo que suelen verse favorecidos en la creación de los árboles.

4.1. Árboles de predicción

El proceso de construcción de un árbol de predicción (regresión o clasificación) se divide en dos etapas:

- División sucesiva del espacio de los predictores generando regiones no solapantes (nodos terminales) $R_1, R_2, R_3, ..., R_j$. Aunque, desde el punto de vista teórico las regiones podrían tener cualquier forma, si se limitan a regiones rectangulares (de múltiples dimensiones), se simplifica en gran medida el proceso de construcción y se facilita la interpretación.
- Predicción de la variable respuesta en cada región.

A pesar de la sencillez con la que se puede resumir el proceso de construcción de un árbol, es necesario establecer una metodología que permita crear las regiones $R_1, R_2, R_3, ..., R_j$, o lo que es equivalente, decidir donde se introducen las divisiones: en qué predictores y en qué valores de los mismos. Es en este punto donde se diferencian los algoritmos de árboles de regresión y clasificación.

En el caso de los árboles de regresión, que es el que nos compete, el criterio más frecuentemente empleado para identificar las divisiones es el Residual Sum of Squares (RSS). El objetivo es encontrar las J regiones $(R_1, ..., R_j)$ que minimizan el Residual Sum of Squares (RSS) total: $RSS = \sum_{j=1}^{J} \sum_{i \in R_j} (y_i - \hat{y}_{R_j})^2$, donde $\hat{y}_{R_j}$ es la media de la variable respuesta en la región R_j. Una descripción menos técnica equivale a decir que se busca una distribución de regiones tal que, el sumatorio de las desviaciones al cuadrado entre las observaciones y la media de la región a la que pertenecen sea lo menor posible.

Desafortunadamente, no es posible considerar todas las posibles particiones del espacio de los predictores. Por esta razón, se recurre a lo que se conoce como recursive binary splitting (división binaria recursiva). Esta solución sigue la misma idea que la selección de

predictores stepwise (backward o fordward) en regresión lineal múltiple, no evalúa todas las posibles regiones pero, alcanza un buen balance computación-resultado.

4.1.1. Recursive Binary Splitting

El objetivo de este método es encontrar en cada iteración el predictor X_j y el punto de corte (umbral) s tal que, si se distribuyen las observaciones en las regiones $\{X|X_j < s\}$ y $\{X|X_j \geq s\}$, se consigue la mayor reducción posible en el RSS. El algoritmo seguido es:

1. El proceso se inicia en lo más alto del árbol, donde todas las observaciones pertenecen a la misma región.
2. Se identifican todos los posibles puntos de corte (umbrales) s para cada uno de los predictores $(X_1, X_2, ..., X_p)$. En el caso de predictores cualitativos, los posibles puntos de corte son cada uno de sus niveles. Para predictores continuos, se ordenan de menor a mayor sus valores, el punto intermedio entre cada par de valores se emplea como punto de corte.
3. Se calcula el RSS total que se consigue con cada posible división identificada en el paso 2, $\sum_{i:x_i \in R_1(j,s)}(y_i - \hat{y}_{R_1})^2 + \sum_{i:x_i \in R_2(j,s)}(y_i - \hat{y}_{R_2})^2$, donde el primer término es el RSS de la región 1 y el segundo término es el RSS de la región 2, siendo cada una de las regiones el resultado de separar las observaciones acorde al predictor j y valor s.
4. Se selecciona el predictor X_j y el punto de corte S que resulta en el menor RSS total, es decir, que da lugar a las divisiones más homogéneas posibles. Si existen dos o más divisiones que consiguen la misma mejora, la elección entre ellas es aleatoria.
5. Se repiten de forma iterativa los pasos 1 a 4 para cada una de las regiones que se han creado en la iteración anterior hasta que se alcanza alguna norma de parada. Algunas de las más empleadas son: que ninguna región contenga un mínimo de n observaciones, que el árbol tenga un máximo de nodos terminales o que la incorporación del nodo reduzca el error en al menos un porcentaje mínimo.

Esta metodología conlleva dos hechos:

- Que cada división óptima se identifica acorde al impacto que tiene en ese momento. No se tiene en cuenta si es la división que dará lugar a mejores árboles en futuras divisiones.
- En cada división se evalúa un único predictor haciendo preguntas binarias (sí-no), lo que genera dos nuevas ramas del árbol por división. A pesar de que es posible evaluar divisiones más complejas, hacer una pregunta sobre múltiples variables a la vez es equivalente a hacer múltiples preguntas sobre variables individuales.

4.1.2. Pruning: poda del árbol

El proceso de división binaria recursiva puede conseguir buenas predicciones con los datos de entrenamiento, ya que reduce el RSS de entrenamiento, lo que implica un sobreajuste a los datos (derivado de la facilidad de ramificación y posible complejidad del árbol resultante), reduciendo la capacidad predictiva para nuevos datos.

Una posible alternativa supone construir un árbol hasta el punto en el que la reducción del RSS debido a cada división supere un determinado límite (alto). Con esta estrategia obtendremos árboles más pequeños, con menos ramificaciones, con lo que conseguimos

reducir la varianza y mejorar la interpretabilidad a expensas de una pequeña reducción del sesgo. Concretamente, la estrategia consistirá en construir un árbol grande T_0 y luego podarlo para obtener un sub-árbol que consiga el menor test error, obtenido mediante validación cruzada o validación simple. Sin embargo, suele resultar muy costoso obtener el error de validación para cada posible sub-árbol debido al gran número de posibles sub-árboles, por lo que se opta por seleccionar un grupo pequeño de sub-árboles a considerar. Una manera de conseguir esto es mediante el *cost complexity pruning*, al que corresponde el siguiente algoritmo:

1. Aplicar el método de división binaria recursiva para obtener un árbol grande (T_0) con los datos de entrenamiento, finalizando el proceso con el cumplimiento de una determinada norma de parada.

2. Aplicar cost complexity pruning a T_0 para obtener una secuencia de mejores sub-árboles, en función del parámetro de penalización o tuning parameter α, que controla el equilibrio entre la complejidad del árbol y su ajuste a los datos de entrenamiento. Conforme α aumenta, más ramas se podan del árbol. A cada valor de α le corresponde un sub-árbol $T \subset T_0$ tal que se minimice

$$\sum_{m=1}^{|T|} \sum_{x_i \in R_m} (y_i - \hat{y}_{R_m})^2 + \alpha |T|,$$

   donde $|T|$ es el número de nodos terminales u hojas del árbol T, y R_m es el rectángulo correspondiente al nodo terminal m. Siendo $\alpha = 0$, T será igual a T_0.

3. Usar k-fold cross validation para escoger α. Esto consiste en dividir las observaciones de entrenamiento en K grupos de manera que para cada $k = 1, ..., K$:

   - Repetir pasos 1 y 2 en los $k - 1$ grupos.
   - Evaluar el error cuadrático medio (MSE) con el grupo k_i de observaciones, en función de α (conveniente graficarlo en función de $|T|$).

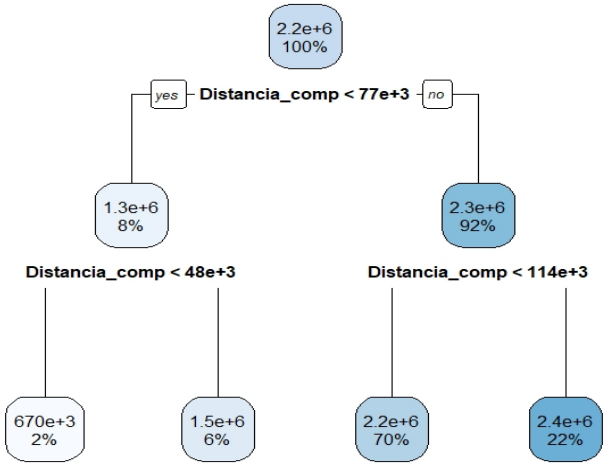
   Promediar los resultados para cada valor de α, escogiendo el valor que minimice el test error.

4. Seleccionar el sub-árbol del paso 2 correspondiente al valor escogido de α por validación cruzada.

4.2. Modelo

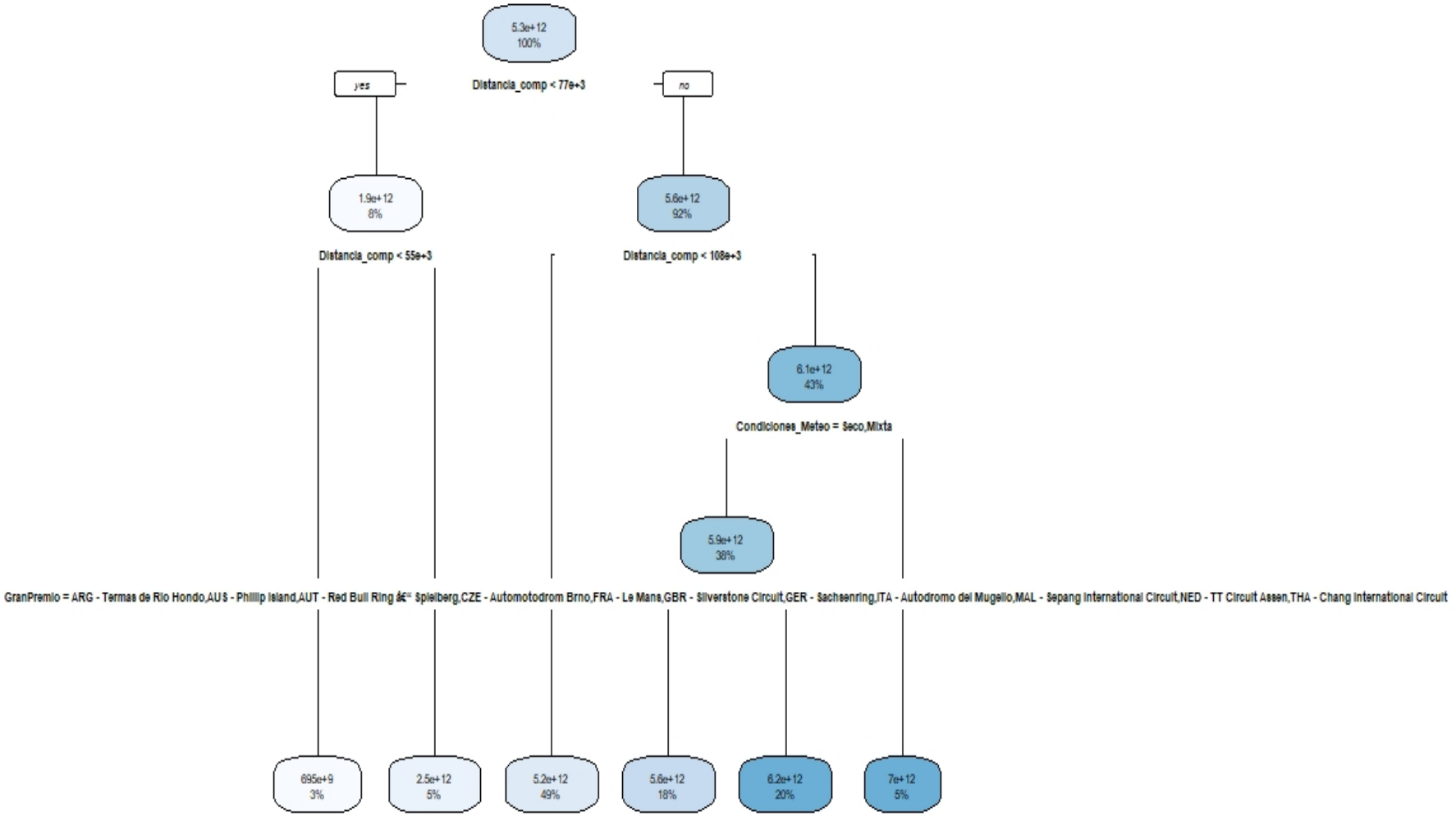
Después de construir los árboles con las variables seleccionadas y transformadas previamente, procedí a la poda de los mismos para su simplificación.

Figura 4.1: Árbol podado Versión 1



Tanto en este caso como en la *Figura 4.2* podemos intuir que la variable *Distancia_comp* va a tener mucha importancia a la hora de montar nuestros árboles, hecho que comentaremos en *Importancia de variables*.

Figura 4.2: Árbol podado Versión 4



Capítulo 5

Random Forest

Los métodos tipo ensamblador están formados por un grupo de modelos predictivos que permiten alcanzar una mejor precisión y estabilidad del modelo. Estos proveen una mejora significativa a los modelos de árboles de decisión.

Así como todos los modelos, un árbol de decisión también sufre de los problemas de sesgo y varianza. Es decir, *cuánto en promedio son los valores predichos diferentes de los valores reales* (**sesgo**) y *cuan diferentes serán las predicciones de un modelo en un mismo punto si muestras diferentes se tomaran de la misma población* (**varianza**).

Al construir un árbol pequeño se obtendrá un modelo con baja varianza y alto sesgo. Normalmente, al incrementar la complejidad del modelo, se verá una reducción en el error de predicción debido a un sesgo más bajo en el modelo. En un punto el modelo será muy complejo y se producirá un sobre-ajuste (*overfitting*) del modelo el cual empezará a sufrir de varianza alta.

El modelo óptimo debería mantener un balance entre estos dos tipos de errores. A esto se le conoce como "trade-off" (equilibrio) entre errores de sesgo y varianza. El uso de ensambladores es una forma de aplicar este "trade-off".

- **¿Qué es el proceso de bagging y cómo funciona?**

En lugar de ajustar un único árbol, se ajustan muchos de ellos en paralelo formando un "bosque". En cada nueva predicción, todos los árboles que forman el "bosque" participan aportando su predicción. Como valor final, se toma la media de todas las predicciones (variables continuas) o la clase más frecuente (variables cualitativas). Uno de los métodos de bagging más conocidos es Random Forest.

En resumen: crear múltiples subconjuntos de datos, construir múltiples modelos y combinar dichos modelos.

5.0.1. Construcción de un modelo Random Forest

- Dado que el número de casos en el conjunto de entrenamiento es N. Una muestra de esos N casos se toma aleatoriamente pero con reemplazo. Esta muestra será el conjunto de entrenamiento para construir el árbol i.
- Si existen M varibles de entrada, un número $m < M$ se especifica tal que para cada nodo, m variables se seleccionan aleatoriamente de M. La mejor división de estos m

atributos es usado para ramificar el árbol. El valor m se mantiene constante durante la generación de todo el bosque.
- Cada árbol crece hasta su máxima extensión posible y no hay proceso de poda.
- Nuevas instancias se predicen a partir de la agregación de las predicciones de los x árboles (en nuestro caso, al ser regresión usaremos el promedio).

5.0.2. Ventajas

- Existen muy pocas suposiciones y por lo tanto la preparación de los datos es mínima.
- Puede manejar hasta miles de variables de entrada e identificar las más significativas.
- Método de reducción de dimensionalidad.
- Una de las salidas del modelo es la importancia de variables.
- Incorpora métodos efectivos para estimar valores perdidos.
- Es posible usarlo como método no supervisado (*clustering*) y detección de outliers.

5.0.3. Desventajas

- Pérdida de interpretación.
- Bueno para clasificación, no tanto para regresión. Las predicciones no son de naturaleza continua.
- En regresión no puede predecir más allá del rango de valores del conjunto de entrenamiento.
- Poco control en lo que hace el modelo (modelo caja negra[1] para modeladores estadísticos).

5.1. Modelo

En este caso, tuve que hacer un tratamiento previo de los conjuntos de entrenamiento y test para poder construir el modelo correctamente:

- ***Recodificar las variables categóricas***: esencialmente, cada categoría la sustituye por su frecuencia absoluta y así se obtienen variables numéricas. De esta manera el estudio se hace de manera continua.

[1]De una caja negra nos interesará su forma de interactuar con el medio que le rodea entendiendo qué es lo que hace, pero sin dar importancia a cómo lo hace.

Capítulo 6

Boosting

Es otra estrategia de ensemble que se puede emplear con un amplio grupo de métodos de *statistical learning*, entre ellos los árboles de predicción. La idea detrás de boosting es ajustar, de forma secuencial, múltiples *weak learners* (modelos sencillos que predicen solo ligeramente mejor que lo esperado por azar). Cada nuevo modelo emplea información del modelo anterior para aprender de sus errores, mejorando iteración a iteración. En el caso de los árboles de predicción, un weak learners se consigue utilizando árboles con una o pocas ramificaciones. A diferencia del método de bagging, el boosting no hace uso de muestreo repetido (*bootstrapping*), por lo que cada árbol construido depende en gran medida de los árboles previos.

Al igual que en bagging, es posible identificar la influencia que tiene cada predictor sobre un modelo boosting. En cada división de los árboles, se registra el descenso conseguido en la medida empleada como criterio de división (*índice Gini, entropía o MSE*). Para cada uno de los predictores, se calcula el descenso medio conseguido en el total de weak learners que forman el ensemble.

Los algoritmos de boosting se caracterizan por tener una cantidad considerable de hiperparámetros, cuyo valor óptimo se identifica mediante validación cruzada. Tres de los más comunes son:

- El número de weak learners o número de iteraciones: A diferencia del bagging y Random Forest, el boosting puede sufrir *overfitting* si este valor es excesivamente alto. Para evitarlo se emplea un término de regularización conocido como *learning rate*.
- *Learning rate* (λ): Controla el ritmo al que aprenden los modelos. Suelen recomendarse valores de 0.01 o 0.001, aunque la elección correcta puede variar dependiendo del problema. Cuanto menor sea λ, más árboles se necesitan para alcanzar buenos resultados pero menor es el riesgo de overfitting.
- Si los weak learners son árboles, el número de divisiones d de cada árbol. Suelen emplearse valores pequeños, entre 1 y 10.

6.0.1. Diferencias entre bagging y boosting

- La estrategia seguida para reducir el error total: El error total de un modelo puede descomponerse como *sesgo+varianza+*ϵ. En bagging, se emplean modelos con muy poco sesgo pero mucha varianza, agregando muchos de estos modelos se consigue reducir la varianza sin apenas inflar el sesgo. En boosting, se emplean modelos con muy poca varianza pero mucho sesgo, ajustando secuencialmente muchos modelos se reduce el sesgo.
- La forma en que se crean los distintos modelos que forman el ensemble final. En el caso de bagging, cada modelo es distinto de los otros porque se entrenan con diferentes muestras obtenidas por bootstrapping a partir de la muestra original. También se le conoce como *parallel ensemble* porque cada modelo se ajusta independientemente de los otros. En boosting, los modelos se ajustan secuencialmente y la importancia (peso) de las observaciones varía en cada iteración, dando lugar a diferentes ajustes. También se les conoce como *sequential ensemble*.

La clave para que los métodos de ensemble consigan mejores resultados que cualquiera de sus modelos individuales es que, los modelos que los forman, sean lo más diversos posibles.

6.0.2. Comparación Random Forest y Boosting

Ambos métodos son muy potentes y la superioridad de uno u otro depende del problema al que se apliquen. Algunos aspectos a tener en cuenta son:

- Gracias al *out-of-bag error*, el método de Random Forest no necesita recurrir a validación cruzada para la optimización de sus hiperparámetros. Esto puede ser una ventaja muy importante cuando los requerimientos computacionales son limitantes.
- Random Forest tiene menos hiperparámetros, lo que hace más sencillo su correcta implementación.
- Si existe una proporción alta de predictores irrelevantes, Random Forest puede verse perjudicado, sobre todo a medida que se reduce el número de predictores (m) evaluados.
    - *Supóngase que, de 100 predictores, 90 de ellos no aportan información (son ruidos). Si el hiperparámetro m es igual a 3 (en cada división se evalúan 3 predictores aleatorios), es muy probable que los 3 predictores seleccionados sean de los que no aportan nada. Aun así, el algoritmo seleccionará el mejor de ellos, incorporándolo al modelo.*
    - Cuanto mayor sea el porcentaje de predictores no relevantes, mayor la frecuencia con la que esto ocurre, por lo que los árboles que forman el Random Forest contendrán predictores irrelevantes. Como consecuencia, se reduce su capacidad predictiva. En el caso de boosting, siempre se evalúan todos los predictores, por lo que los no relevantes se ignoran.
- En Random Rorest, cada modelo que forma el ensemble es independiente del resto, esto facilita mucho la paralelización del algoritmo.
- Random Forest no sufre problemas de overfitting por muchos árboles que se agreguen.

6.1. Modelo

En este caso,también tuve que hacer un tratamiento previo de los conjuntos de entrenamiento y test para poder construir el modelo correctamente:

- ***Recodificar las variables categóricas***: esencialmente, cada categoría la sustituye por su frecuencia absoluta y así se obtienen variables numéricas. De esta manera el estudio se hace de manera continua.
- ***Hacer una configuración de los hiperparámetros óptimos a usar***.

Los resultados que obtuve finalmente fueron:

Cuadro 6.1: Hiperparámetros Boosting

nrounds	max_depth	eta	gamma	colsample_bytree	min_child_weight	subsample
1400	3	0.01	1	0.8	3	0.75

- ***Trabajar con todas las variables***: no usamos las versiones previamente estudiadas con la selección y transformación de variables, sino que dejamos que el modelo trabaje con todas ellas.

Capítulo 7

Deep Learning

El término deep learning engloba a todo un conjunto de modelos basados en redes neuronales artificiales (*artificial neural networks*) que contienen múltiples capas intermedias (ocultas).

Estas redes se caracterizan por:

- Pueden tener una o múltiples capas intermedias, también conocidas como capas ocultas. Son las capas de neuronas que hay entre la capa de entrada y la de salida.
- La estructura es *full conected*, lo que significa que cada neurona está conectada con todas las neuronas de la capa siguiente.
- El peso (participación de cada neurona en la red) se aprende mediante el proceso *feed-forward*. A grandes rasgos, la estrategia de aprendizaje consiste en un proceso iterativo. En cada iteración (época), la red predice las observaciones de entrenamiento, se calcula el error de estas predicciones comparando los resultados frente al valor real y se reajustan los pesos de las neuronas con el objetivo de minimizar el error (*backpropagation* y descenso de gradiente).

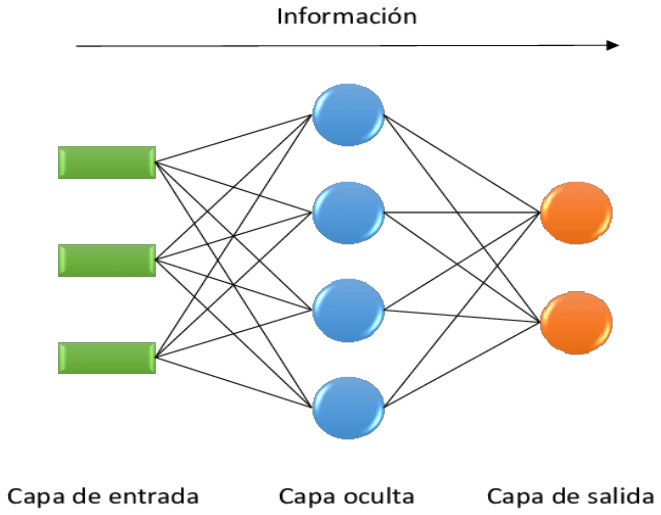
7.0.1. Redes neuronales

Conviene definir desde un principio qué son las Redes de Neuronas Artificiales, que no son más que un paradigma de computación basado en una representación simplificada del funcionamiento del sistema nervioso humano.

Un *perceptrón mononivel*, también llamado perceptrón simple o perceptrón, es una Red de Neuronas Artificiales alimentada hacia adelante formada por dos capas de nodos que verifican las dos condiciones:

- Los nodos de la primera capa no realizan ningún cálculo, limitándose a proporcionar los valores de entrada a la red.
- La segunda capa está formada por un solo nodo, que aplica una función de activación paso o signo.

Figura 7.1: Ejemplo estructura red neuronal



Un *perceptrón multinivel o multicapas*, es una Red de Neuronas Artificiales alimentada hacia adelante con tres o más capas de neuronas. Las capas situadas entre la primera y la última reciben el nombre de capas ocultas o capas intermedias.

Los modelos basados en redes neuronales tienen la característica de ser potencialmente capaces de aprender cualquier función. La velocidad con la que aprenden está influenciada principalmente (aunque hay muchos otros hiperparámetros) por el número de observaciones, el número de neuronas y el número de iteraciones de aprendizaje (comúnmente llamadas *epoch*).

7.1. Modelo

En este caso,también tuve que hacer un tratamiento previo de los conjuntos de entrenamiento y test para poder construir el modelo correctamente:

- ***Recodificar las variables categóricas***: esencialmente, cada categoría la sustituye por su frecuencia absoluta y así se obtienen variables numéricas. De esta manera el estudio se hace de manera continua.
- ***Variables sin transformar***: a la hora de trabajar con las distintas versiones planteadas, se hizo sólo con la selección de las variables, sin transformaciones. Por tanto, se redujo el estudio a dos versiones, puesto que la 2 y la 4 se diferenciaban en las transformaciones realizadas, pero las variables eran las mismas.
- ***Transformar variable Tiempo***: las observaciones del Tiempo las dividí entre el máximo (*3275570*), para simplificar un poco los datos con lo que trabajaba el algoritmo en cuestión. Este cambio lo deshice posteriormente a la hora de obtener las predicciones, así la interpretación seguirá siendo la correcta.

Capítulo 8

Análisis de resultados

Una vez he entrenado todos los modelos en cada una de las versiones, con las diferentes variables y transformaciones previamente seleccionadas, he obtenido las predicciones en los conjuntos entrenamiento y test (breve resumen en el cuadro 8.1).

Llegados a este punto, lo que voy a hacer es comparar la bondad de cada ajuste que he obtenido, y para ello he definido una *Función bondad de ajuste* a la que le doy el vector de observaciones y de predicciones, a partir de los cuales calculo:

- **Error Cuadrático Medio (MSE).**
  - Mide el promedio de los errores al cuadrado, es decir, la diferencia entre el estimador y lo que se estima.
  - Estas desviaciones se denominan *residuos* cuando los cálculos se realizan sobre la muestra de datos que se utilizó para la estimación y se denominan *errores* cuando se calculan fuera de la muestra.
  - Hemos dividido el resultado entre 1000000, ya que nuestros resultados están en milisegundos, y así se podría interpretar mejor la medida obtenida.
- **Raíz del Error Cuadrático Medio (RMSE).**
  - Se define como la raíz cuadrada del error cuadrático medio
  - Es siempre no negativo y un valor de 0 (casi nunca alcanzado en la práctica) indicaría un ajuste perfecto a los datos. En general, un RMSE más bajo es mejor que uno más alto.
  - Las comparaciones entre diferentes tipos de datos no serían válidas porque la medida depende de la escala de los números utilizados.
- **Coeficiente de Correlación (R^2).**
  - Se define como la proporción de la varianza total de la variable explicada por la regresión.
  - El resultado oscila entre 0 y 1. Cuanto más cerca de 1 se sitúe su valor, mayor será el ajuste del modelo a la variable que estamos intentando explicar. De forma inversa, cuanto más cerca de cero, menos ajustado estará el modelo y, por tanto, menos fiable será.
- **Porcentaje de Error Cuadrático Medio Absoluto (PEMA).**
  - Seguiremos trabajando con el error cuadrático medio, en este caso con su valor absoluto, y calcularemos el porcentaje de dicha medida.
  - Nos interesarán valores bajos, aunque hay que tener cuidado al ser una medida adimensional.

Cuadro 8.1: Predicciones

Regresion.V1	Regresion.V2	Regresion.V4	Arboles.V1	Arboles.V2	Arboles.V4	RF.V1	RF.V2	RF.V4	DL.V1	DL.V2	Boost.V1
Entrenamiento											
2124298	2310842	2249810	2236453	2335516	2287988	2214726	2378267	2297057	2164249	2203069	2276432
2455180	2584771	2555072	2431954	2538835	2497993	2453220	2552087	2521784	2341978	2341462	2392249
2244462	2328287	2283439	2236453	2335516	2287988	2299567	2449764	2446632	2192333	2207161	2256535
2420942	2564254	2490376	2236453	2488777	2287988	1532713	1704289	1376786	2294053	2388086	1978471
2265257	2462090	2402341	2236453	2389012	2358320	2184796	2306482	2201791	2354699	2286082	2269057
2220889	2344937	2281469	2236453	2335516	2287988	2127765	2269492	2224647	2197630	2193002	2114148
2179437	2269937	2217198	2236453	2389012	2358320	2147372	2292965	2267249	2228934	2232834	2055756
2519833	2615503	2585347	2431954	2538835	2497993	2634641	2692722	2700067	2474374	2557299	2570432
1735022	1856608	1820166	2236453	1675676	2287988	2048960	2182545	2144951	1958719	1996401	1923317
1704280	1889976	1824117	2236453	2177918	2287988	1777955	1776265	1778267	1989740	1995750	1903704
2303672	2357916	2320158	2236453	2335516	2287988	2366627	2418971	2410597	2229520	2250891	2229338
2090339	2248937	2199891	2236453	2335516	2287988	2043273	2078457	2023882	2162823	2193407	2069548
2473508	2568409	2519262	2431954	2538835	2497993	2362149	2301305	2073998	2397549	2373725	2077250
2140001	2270898	2230924	2236453	2335516	2287988	2209883	2271608	2224771	2189519	2228571	2188484
2271699	2463780	2407255	2431954	2538835	2497993	2433345	2529661	2475947	2397728	2423361	2204267
Test											
2285316	2464839	2409673	2236453	2335516	2287988	2406203	2535466	2527109	2215161	2235147	1814731
2354295	2530225	2471113	2431954	2538835	2497993	2545968	2684883	2675088	2347245	2468075	2481371
2410224	2558735	2509103	2431954	2538835	2497993	2473718	2589002	2578977	2534325	2583005	2509725
2320671	2475382	2406481	2236453	2538835	2497993	2343773	2515008	2465776	2341378	2323979	2310515
2505505	2648710	2603337	2236453	2712371	2653531	1950143	2675818	2533200	2398240	2515264	2388242
2227160	2291936	2225072	2236453	2335516	2287988	2397157	2302678	2270066	2173558	2184853	2220480
2555591	2644086	2623426	2431954	2538835	2497993	2596287	2622933	2611790	2486663	2508662	2493020
2252080	2303099	2281636	2236453	2335516	2287988	2228939	2309149	2298711	2230197	2236039	2283420
2175262	2295885	2247831	2236453	2389012	2358320	2148020	2309361	2273311	2198920	2229939	2152046
2420904	2498489	2460847	2431954	2389012	2358320	2459487	2454021	2401455	2332648	2270199	2297059
2237124	2367549	2325270	2236453	2335516	2287988	2058071	2271431	2233734	2245254	2233297	2194858
2120681	2263930	2219664	2236453	2389012	2358320	1962412	2217256	2187232	2197209	2245642	2106636
2252515	2388842	2344550	2236453	2335516	2287988	2341845	2451303	2441695	2199419	2228714	2262617
2440208	2514930	2508416	2236453	2335516	2287988	2389954	2448036	2411499	2211920	2241549	2317555
2328720	2446748	2413857	2236453	2335516	2287988	2332580	2406580	2373326	2158547	2178053	2244840

[a] Nota: Los resultados se dan expresados en milisegundos.

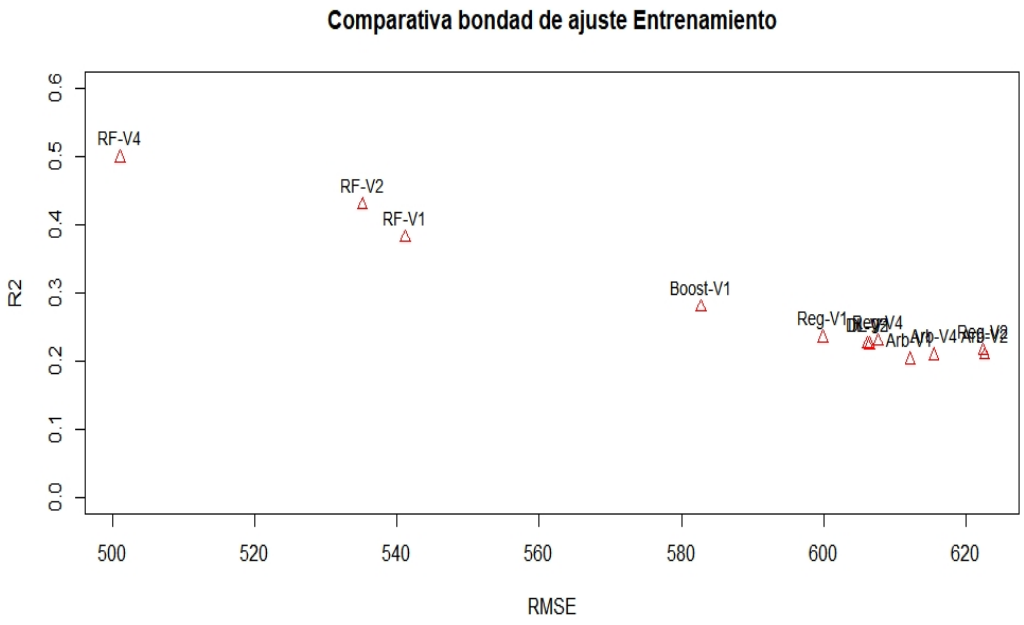
8.1. Comparativa

8.1.1. Bondad de ajuste

Cuadro 8.2: Bondad de Ajuste Entrenamiento

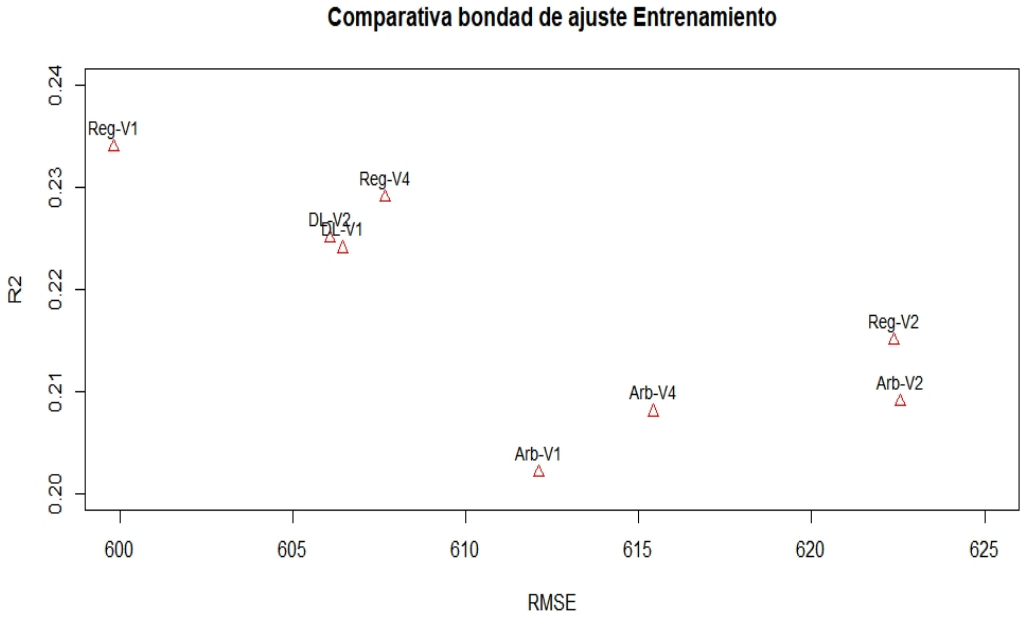
	MSE	RMSE	R2	PEMA
Regresion-V1	359793.2	599.828	0.234	61.237
Regresion-V2	387364.5	622.386	0.215	63.626
Regresion-V4	369249.4	607.659	0.229	62.283
Arboles-V1	374687.0	612.117	0.202	62.600
Arboles-V2	387581.2	622.560	0.209	63.803
Arboles-V4	378734.6	615.414	0.208	63.137
RF-V1	292834.8	541.142	0.381	53.296
RF-V2	286377.4	535.142	0.429	53.758
RF-V4	251089.8	501.089	0.498	50.335
DL-V1	367770.3	606.441	0.224	62.828
DL-V2	367324.8	606.073	0.225	62.887
Boosting-V1	339592.0	582.745	0.279	59.273

Figura 8.1: Bondad de Ajuste Entrenamiento



El modelo "ideal" sería aquel con menor MSE, RMSE y PEMA y el R^2 más cercano a uno. En este caso nos quedaríamos con los modelos situados en el cuadrante superior izquierdo de nuestra gráfica que en este caso coincide con las tres versiones estudiadas con el modelo Random Forest, siendo la Versión 4 la que mejor rendimiento muestra, aunque sigue sin ser un modelo óptimo ya que el R^2 sigue estando bastante lejos de 1.

Figura 8.2: Bondad de Ajuste Entrenamiento - Zoom



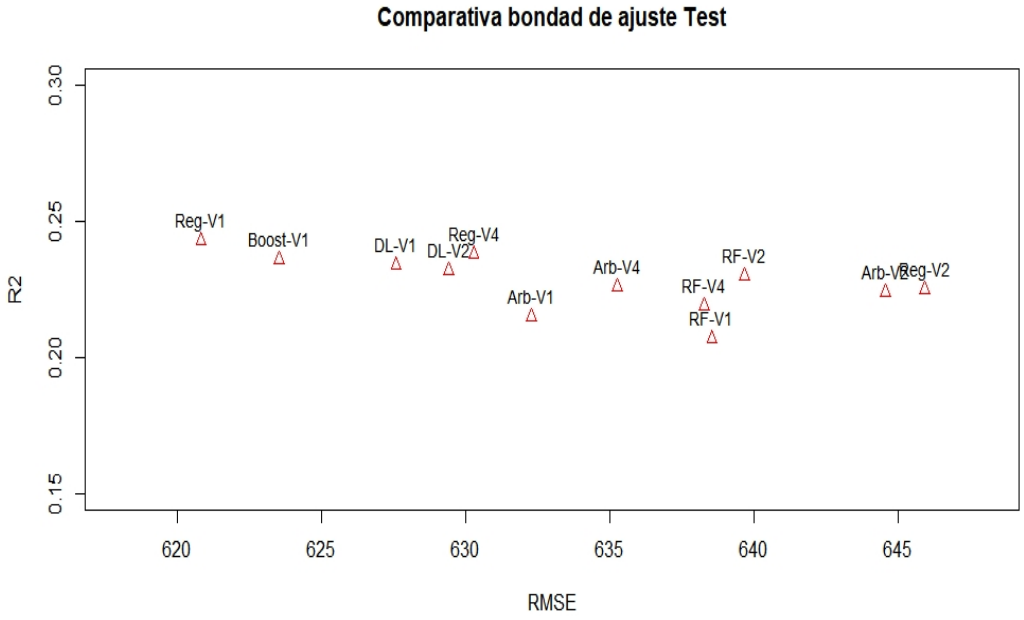
Si nos fijamos en el cuadrante inferior derecho y ampliamos la región para ver mejor los modelos que ahí se encuentran, podemos observar cómo los modelos de Árboles tienen un muy bajo rendimiento, junto con la Versión 2 de Regresión.

Igualmente debemos señalar el caso de la Versión 1 de Regresión, que aun sin tener un rendimiento del todo bueno, es de los modelos que mejor se comporta en este subconjunto estudiado.

Cuadro 8.3: Bondad de Ajuste Test

	MSE	RMSE	R2	PEMA
Regresion-V1	385411.9	620.815	0.243	64.252
Regresion-V2	417223.1	645.928	0.225	67.452
Regresion-V4	397242.8	630.272	0.238	65.842
Arboles-V1	399772.8	632.276	0.215	65.766
Arboles-V2	415457.9	644.560	0.224	67.475
Arboles-V4	403538.7	635.247	0.226	66.367
RF-V1	407740.0	638.545	0.207	63.972
RF-V2	409186.1	639.677	0.230	64.607
RF-V4	407374.8	638.259	0.219	64.342
DL-V1	393857.5	627.581	0.234	66.156
DL-V2	396157.8	629.411	0.232	66.537
Boosting-V1	388783.4	623.525	0.236	64.468

Figura 8.3: Bondad de Ajuste Test

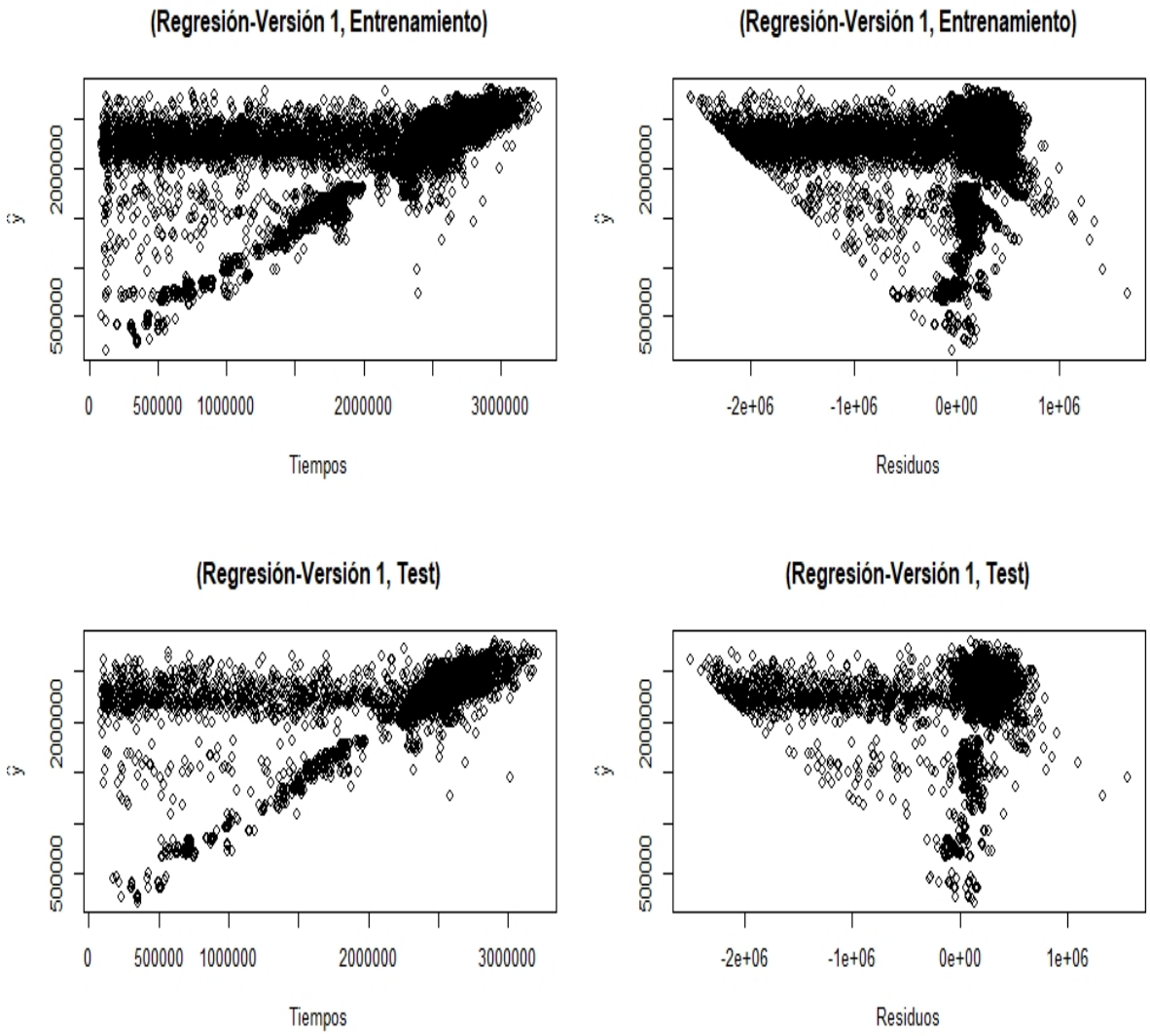


En este caso nos podríamos quedar con las Versiones 1 de Regresión, Boosting y Deep Learning, aunque siguen sin ser ninguno de los modelos del todo óptimos. Los que peor rendimiento vuelven a mostrar son las Versiones 2 de Regresión y Árboles.

Cabe destacar que las tres versiones de los modelos Random Forest ahora han pasado a ser de los que peor rendimiento muestran.

8.1.1.1. Gráficos

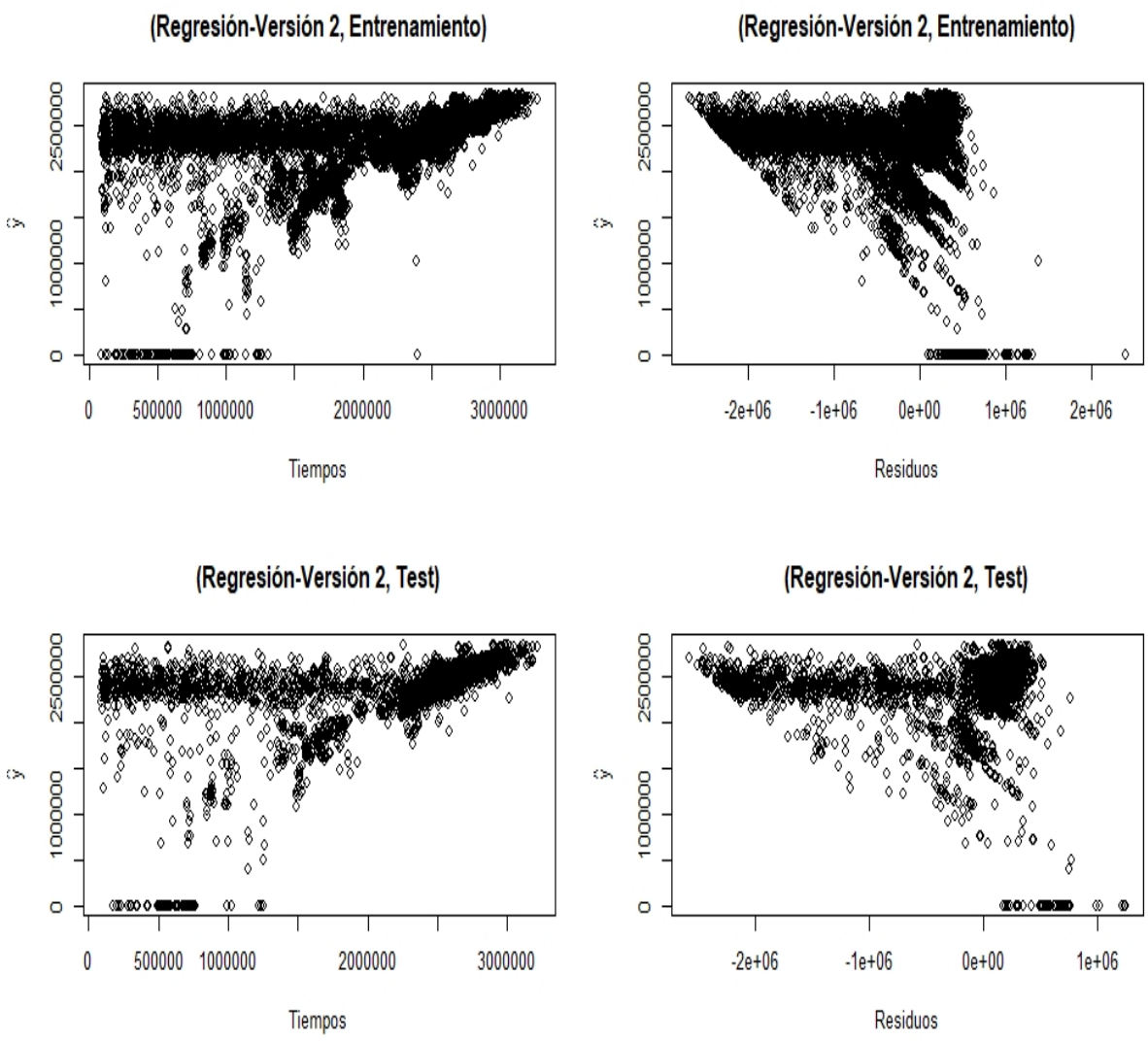
Figura 8.4: Gráficos bondad de ajuste Regresión-Versión 1



Representando los Tiempos frente a sus predicciones podemos ver cómo el modelo se ajusta mejor para los valores más altos. Esto tiene bastante lógica, puesto que los tiempos más bajos corresponderán a carreras divididas en dos por razones meteorológicas, de las cuales existen menos datos en los que basarse para poder predecir correctamente dichos resultados.

Representando los residuos frente a las predicciones podemos ver claramente (como ya intuimos al construir el *modelo*) que la hipótesis de homocedasticidad está siendo violada, puesto que los puntos se distribuyen en forma de embudo.

Figura 8.5: Gráficos bondad de ajuste Regresión-Versión 2

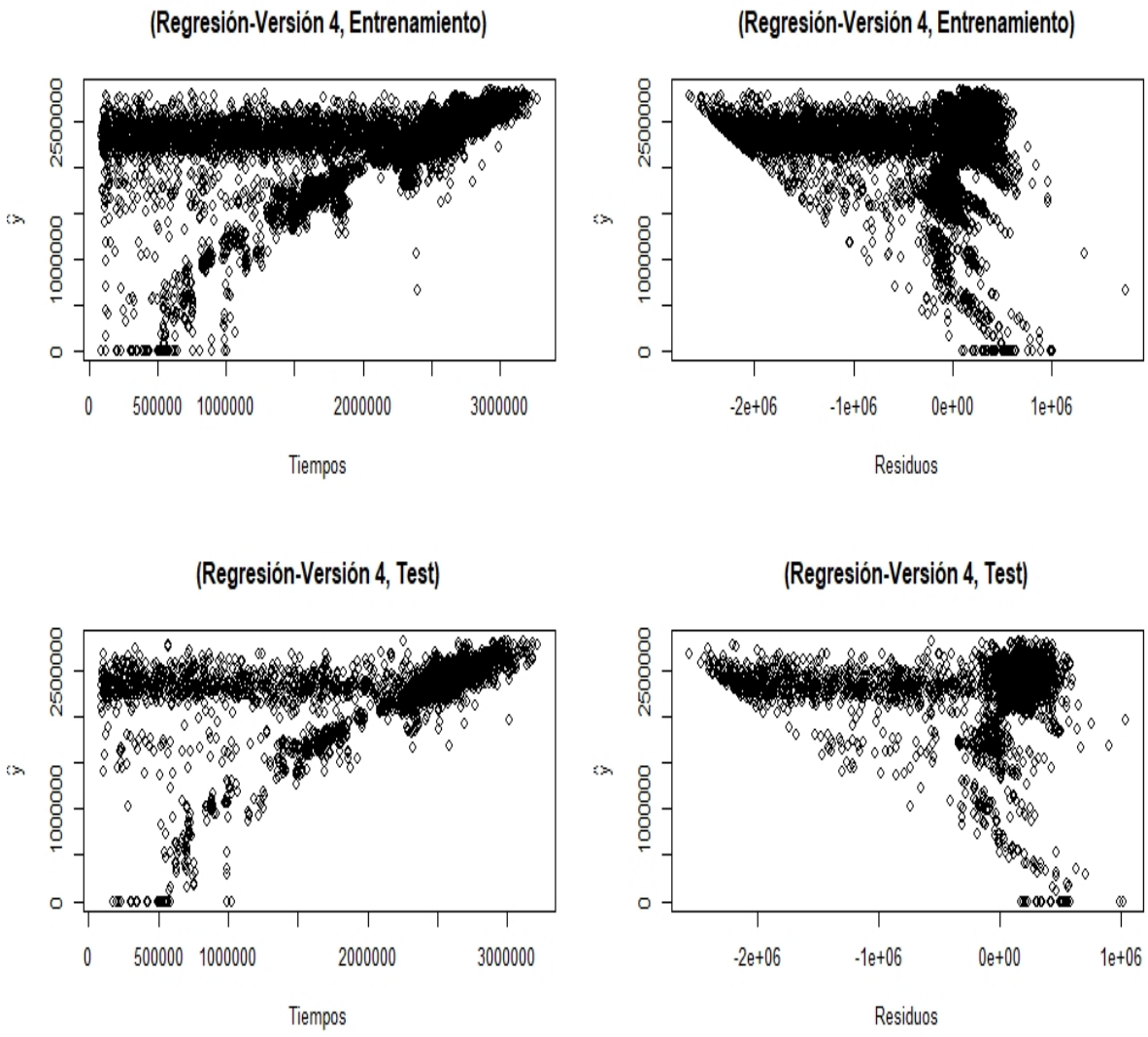


En este caso tenemos un panorama ligeramente diferente. Al representar tanto los Tiempos como los residuos frente a sus predicciones podemos ver cómo existe una línea en el eje horizontal que coincide con el 0: esto se debe al hecho de que al hacer el cambio inverso (elevando a 1/3) las predicciones negativas se han tenido que convertir previamente en un 0 para no obtener valores perdidos.

Por lo demás, las conclusiones que se obtienen en ambos gráficos (tanto en entrenamiento como en test) son bastante parecidas a las de la Versión 1.

Ahora existen algunos casos atípicos más notables que antes, pero sigue sin darse la condición de homocedasticidad y los residuos siguen estando, en su mayor parte, en torno al 0, lo cual es buena señal.

Figura 8.6: Gráficos bondad de ajuste Regresión-Versión 4

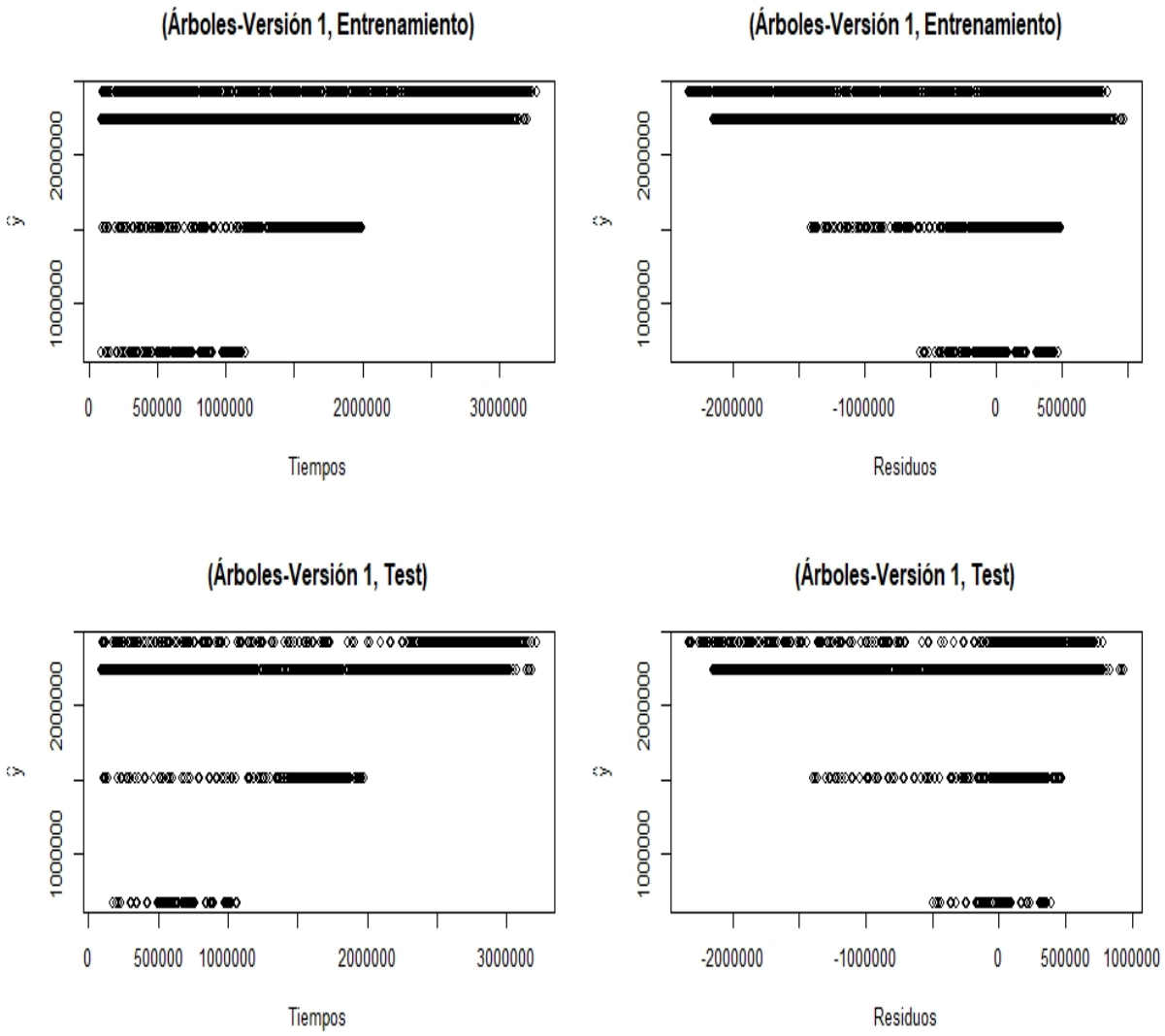


Nuevamente observamos esa línea horizontal en el 0, que vuelve a tener la misma explicación que en la versión anterior: el cambio inverso en este caso es la raíz cuadrada y sólo podemos aplicarla a valores estrictamente positivos.

La hipótesis de homocedasticidad sigue sin cumplirse, porque los Tiempos frente a sus predicciones siguén distribuyéndose en forma de embudo.

En este caso también podemos percatarnos de la existencia de algunos valores atípicos.

Figura 8.7: Gráficos bondad de ajuste Árboles-Versión 1

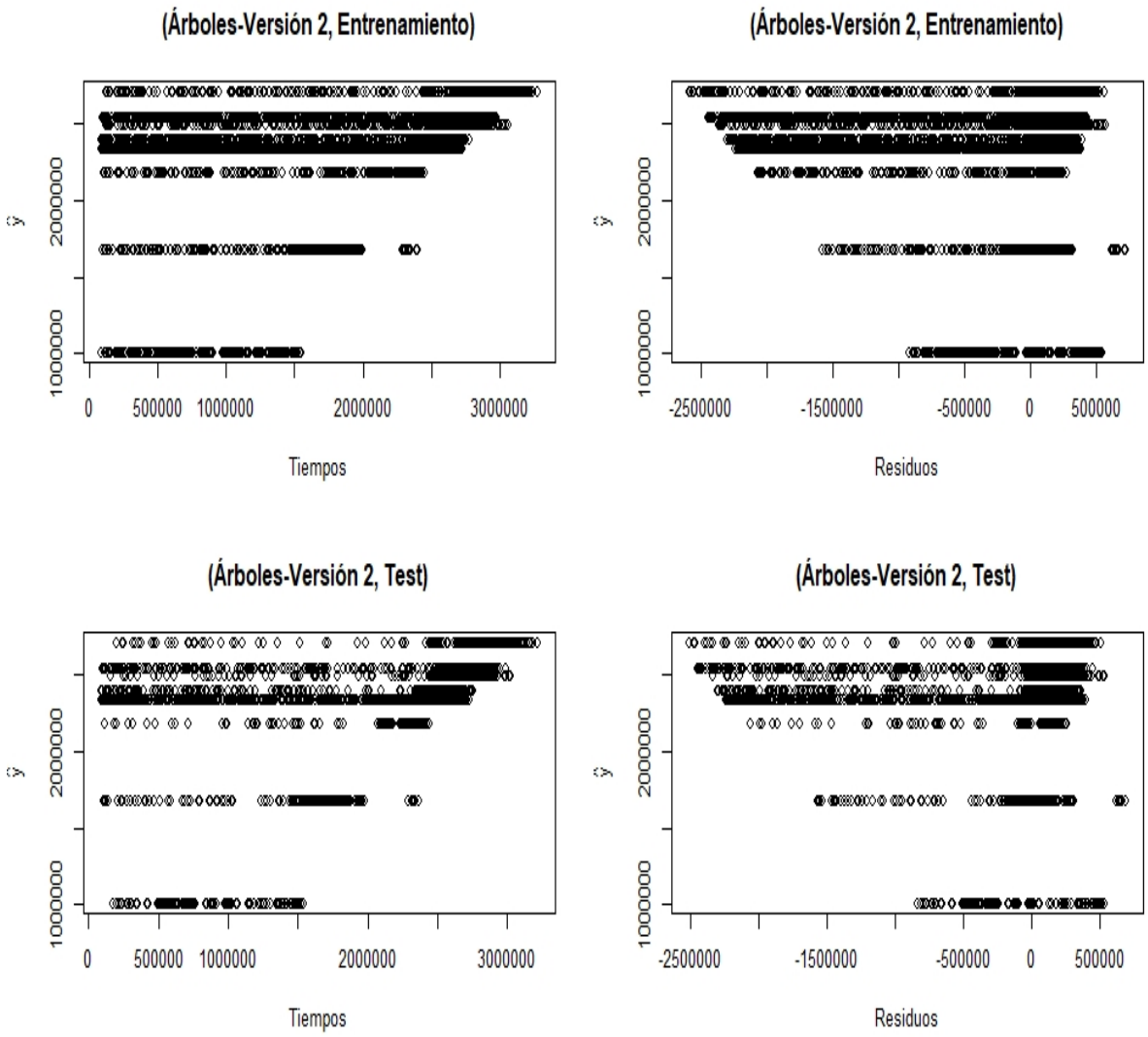


En esta ocasión, la representación es bastante diferente a las que hemos ido analizando antes. En el modelo de árboles vamos obteniendo resultados por "ramas", de ahí que los datos se distribuyan en esas cuatro líneas horizontales.

Aun así, lo que podemos decir es que las predicciones no son demasiado buenas, puesto que la mayoría de los datos se concentran en las dos líneas superiores, que se corresponden con valores de predicciones de Tiempo bastante elevadas.

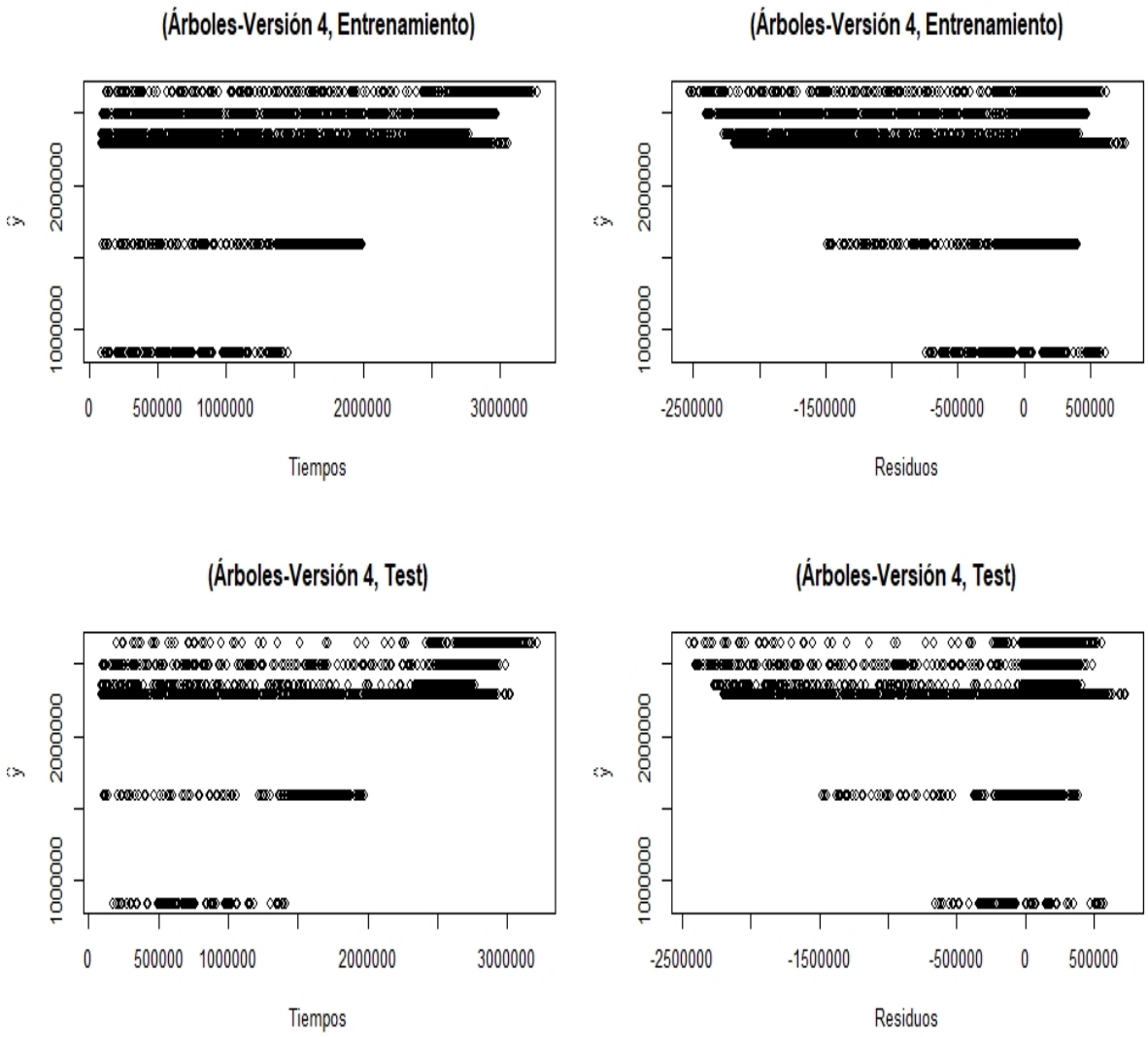
Con respecto a los residuos el comportamiento es bastante similar y no hay nada reseñable.

Figura 8.8: Gráficos bondad de ajuste Árboles-Versión 2



En esta ocasión podemos observar que disponemos de un mayor número de ramas, pero ello no hace que el comportamiento del modelo mejore demasiado, puesto que los residuos siguen estando bastante alejados del 0 y las predicciones siguen acumulándose en los valores altos, lo que hace que se cometa mayor error al hacerle corresponder predicciones menores.

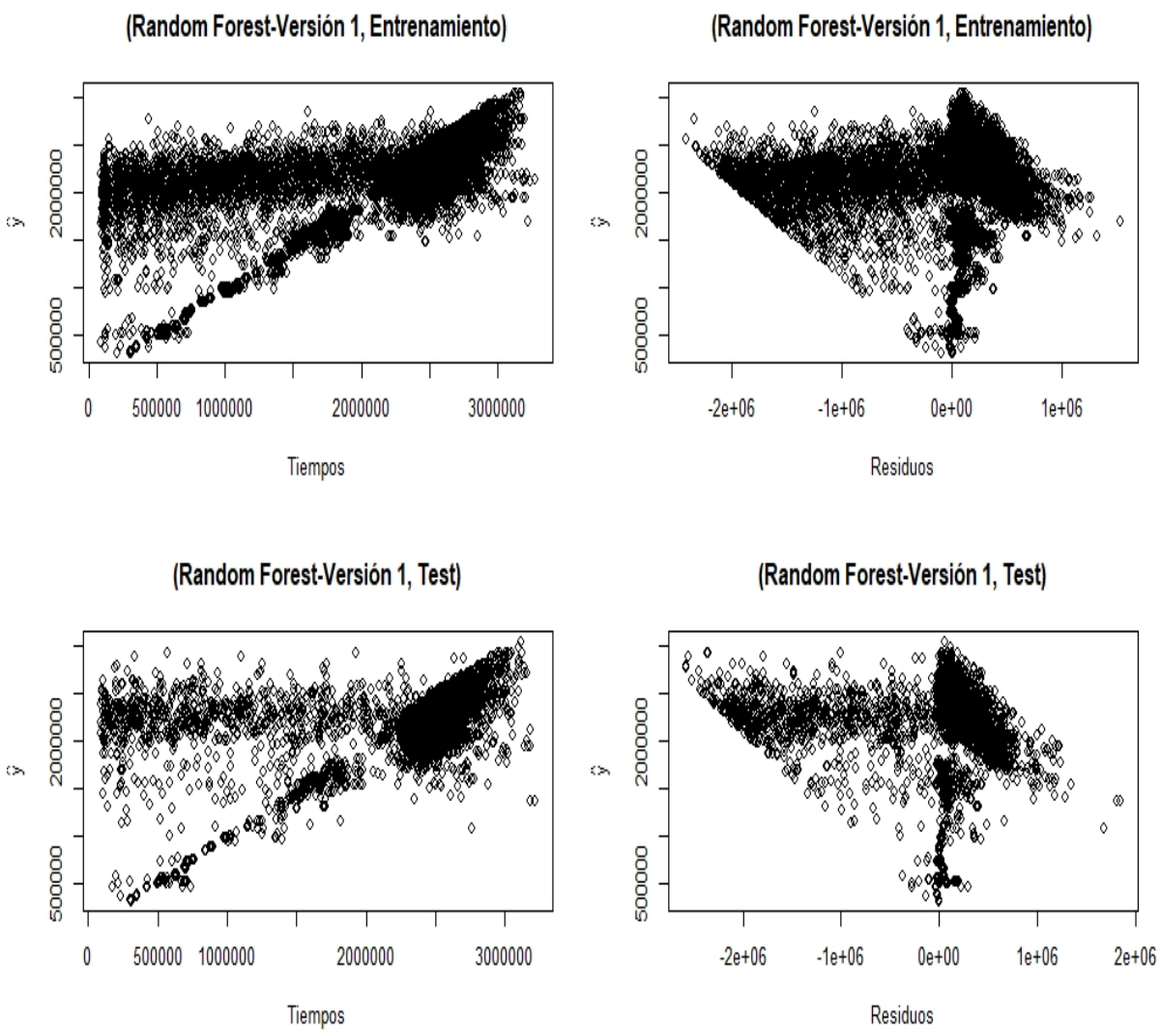
Figura 8.9: Gráficos bondad de ajuste Árboles-Versión 4



Esta vez el número de ramas vuelve a verse disminuido, pero sigue sin conseguirse un rendimiento óptimo del modelo.

Los residuos siguen distribuyéndose muy alejados del cero, por lo que el error que se comete en las predicciones sigue siendo muy importante.

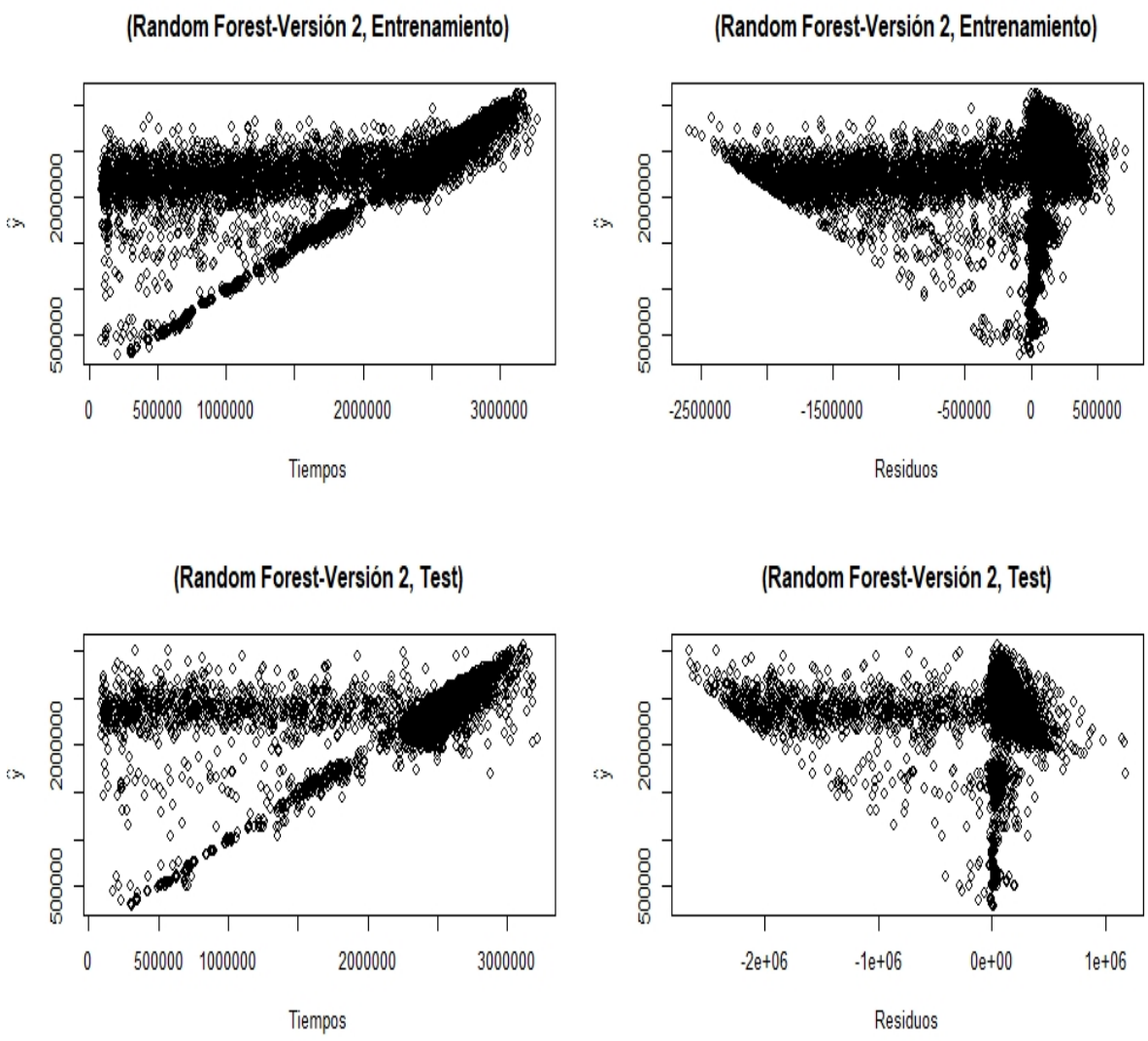
Figura 8.10: Gráficos bondad de ajuste Random Forest-Versión 1



Para este modelo se observa que los Tiempos y las predicciones se distribuyen, más o menos, de manera lineal y ajustada, aunque evidentemente sigue habiendo datos que se alejan de esta "distribución idónea".

Ello podemos verlo más claramente en el gráfico de los residuos: aunque vemos que en los alrededores del cero se acumulan muchas de las observaciones, siguen existiendo otras muchas bastante alejadas.

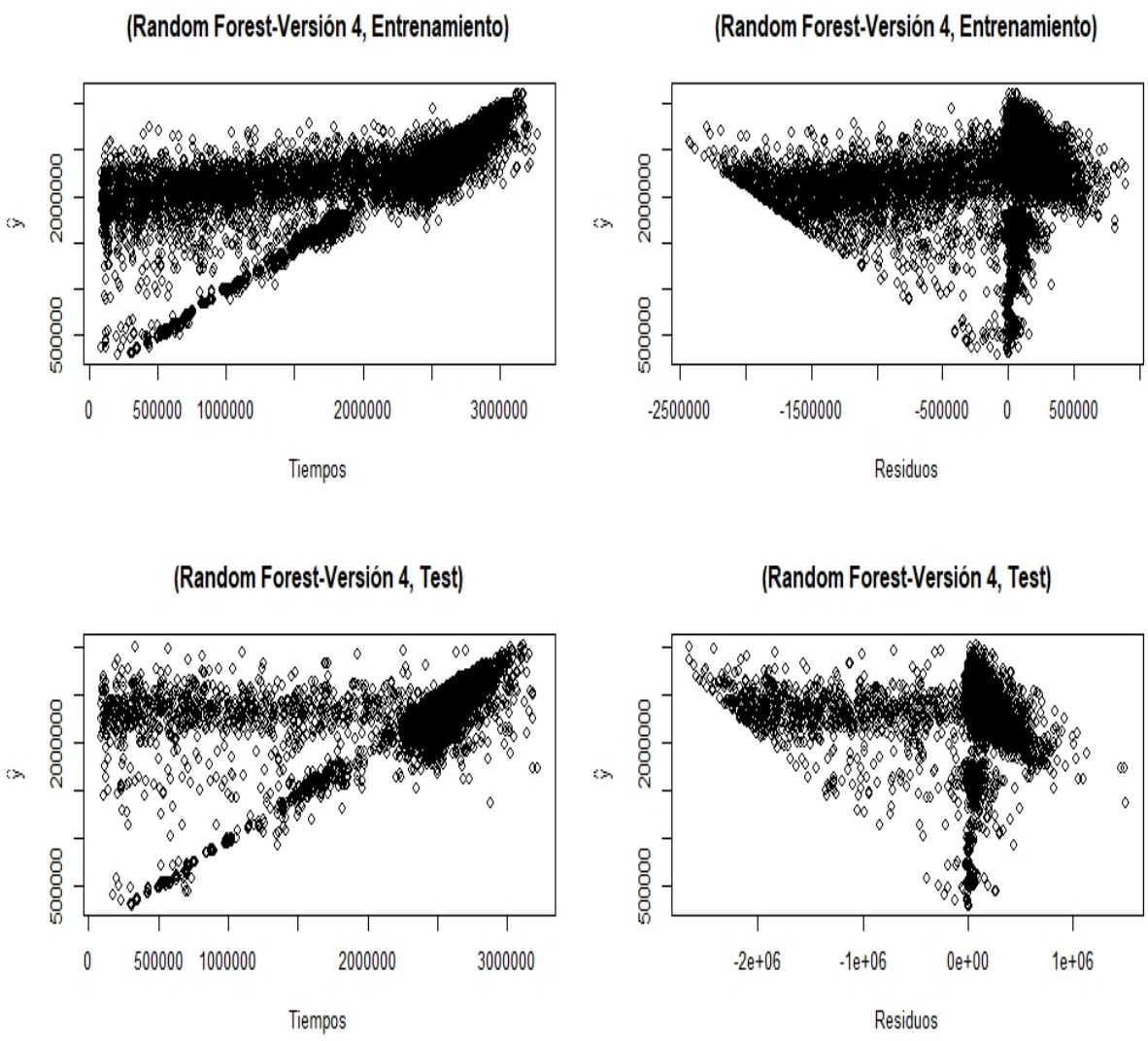
Figura 8.11: Gráficos bondad de ajuste Random Forest-Versión 2



Vemos ahora cómo hemos mejorado sustancialmente el ajuste de las predicciones en este caso, puesto que la línea se atisba cada vez más ajustada.

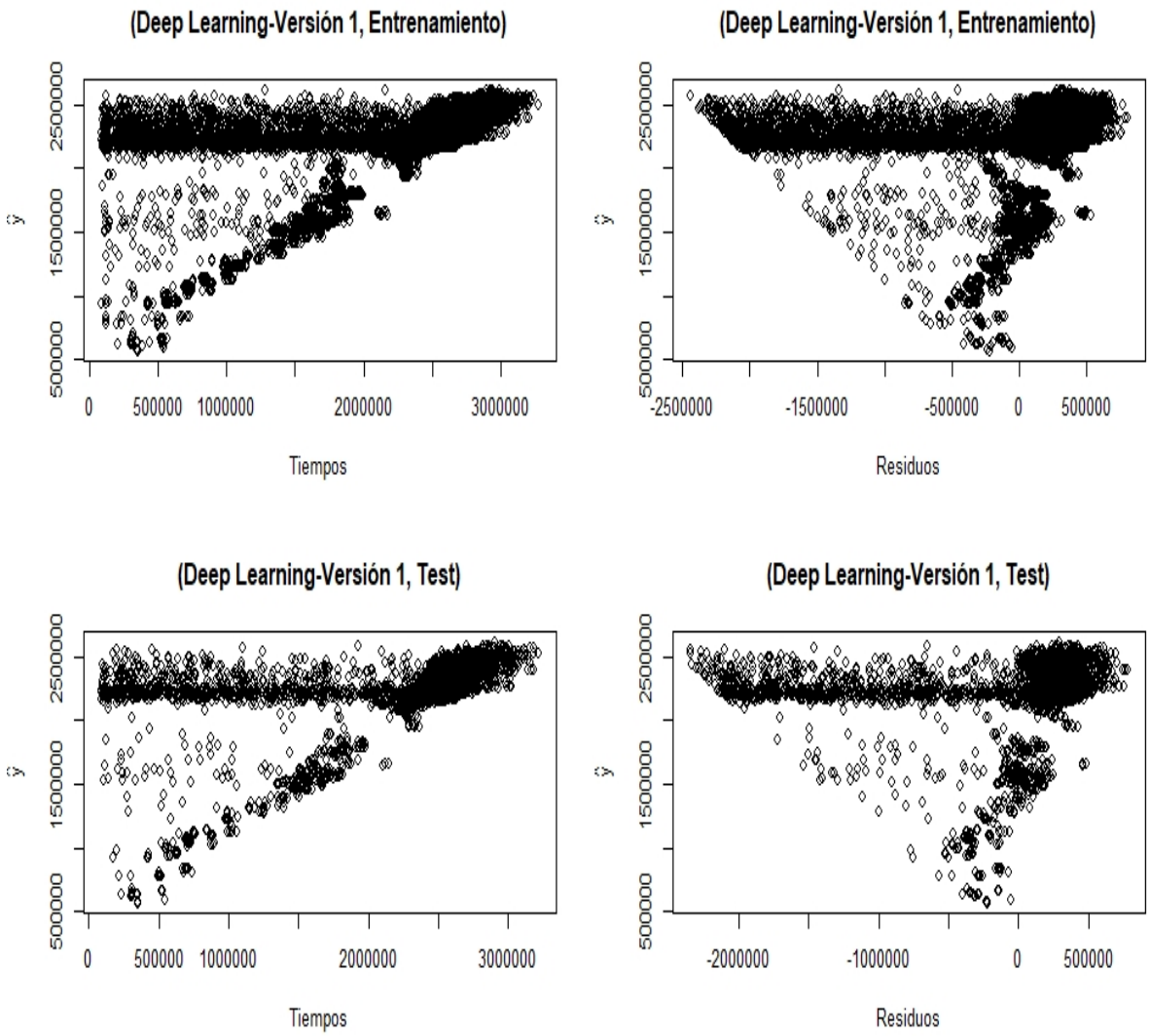
Evidentemente, siguen existiendo residuos bastante elevados, que son los que harán que el modelo siga sin resultar válido por completo.

Figura 8.12: Gráficos bondad de ajuste Random Forest-Versión 4



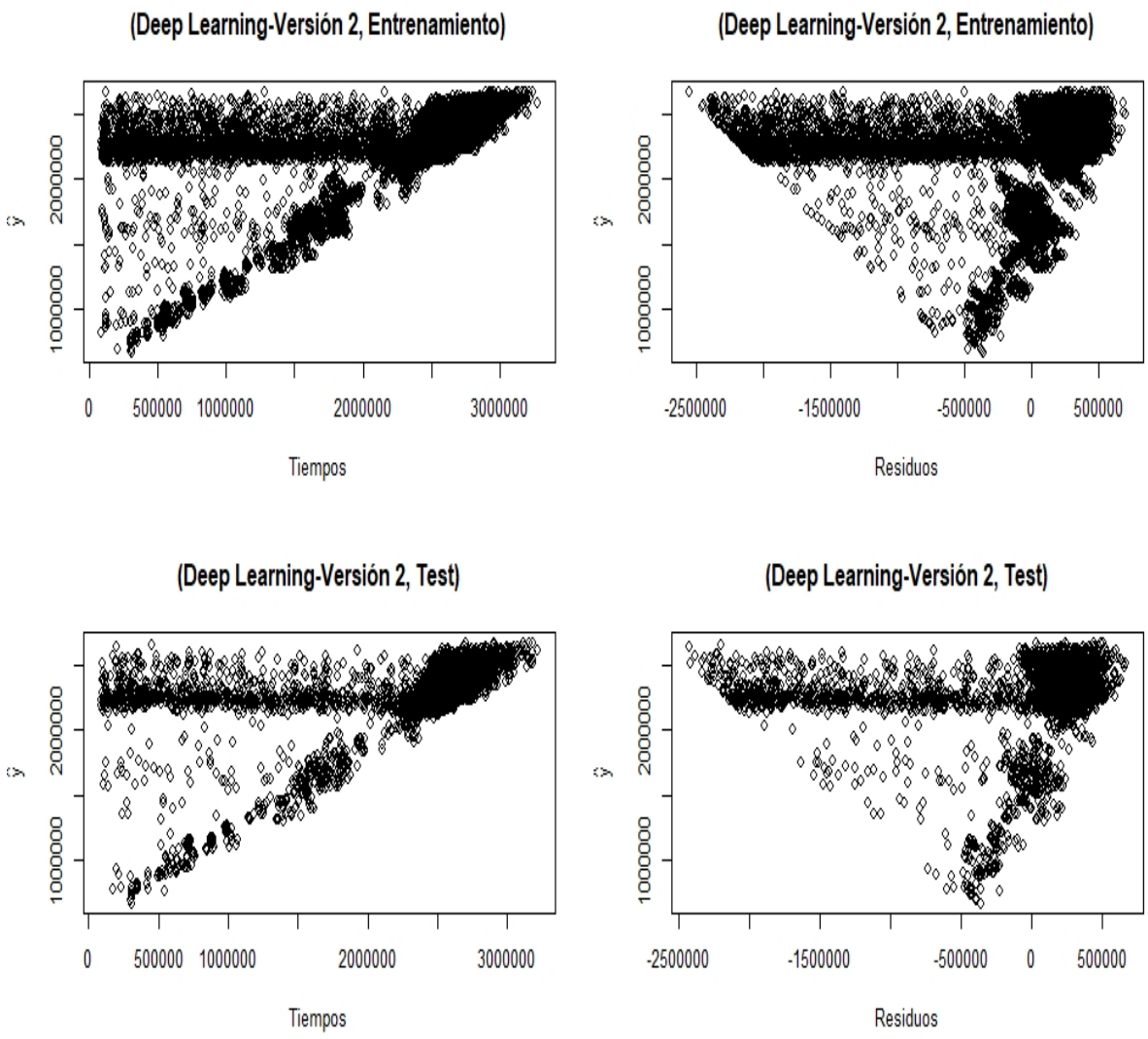
En esta última versión no observamos ningún cambio destacable, siguiendo válidas por tanto las mismas conclusiones hechas anteriormente.

Figura 8.13: Gráficos bondad de ajuste Deep Learning-Versión 1



Esta vez sí que se puede ver una distribución muy ajustada de los valores estimados con respecto a los originales, aunque en los residuos sí que seguimos contando con valores situados muy a la izquierda del eje, es decir, lejos del cero, que es lo ideal.

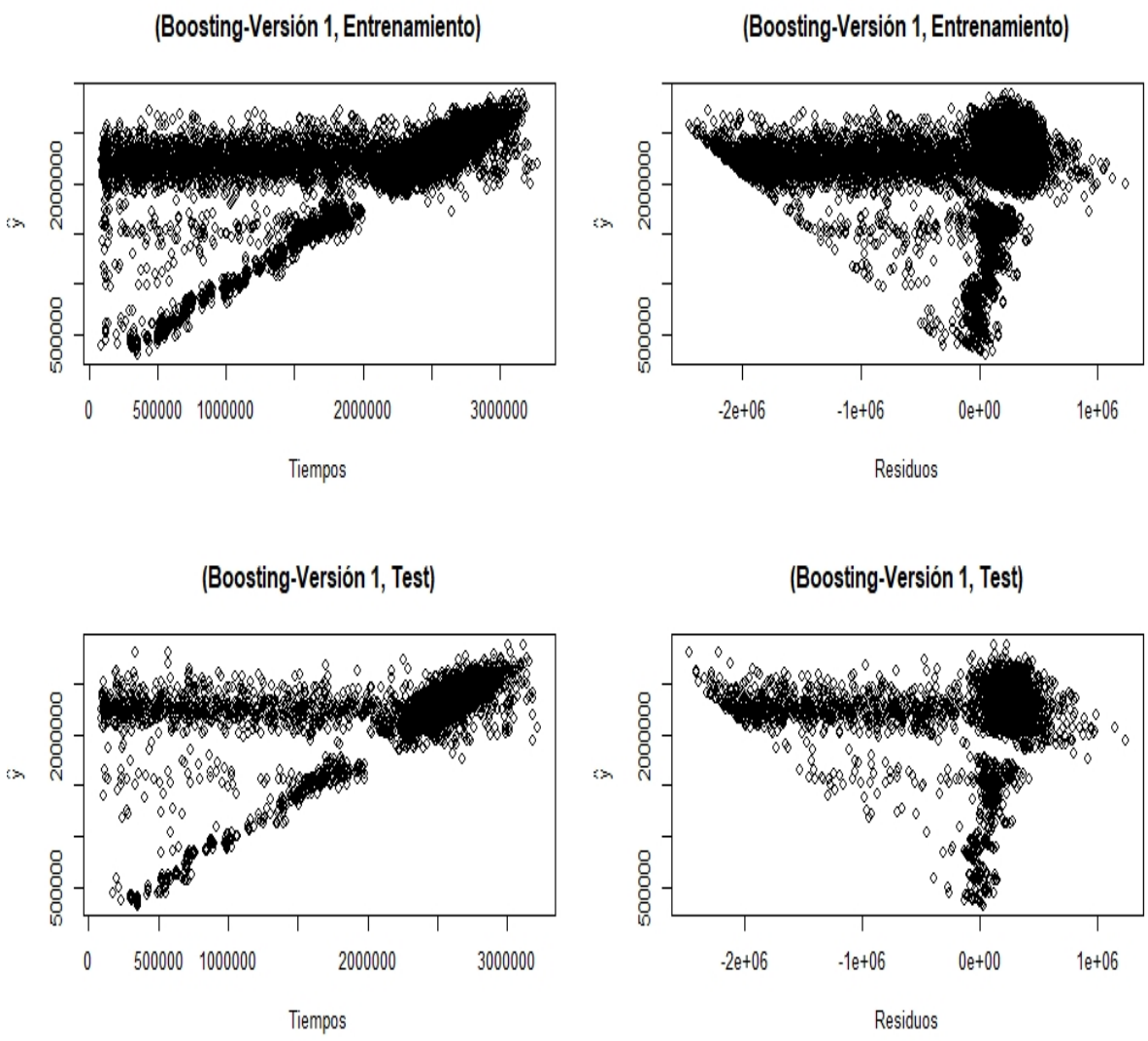
Figura 8.14: Gráficos bondad de ajuste Deep Learning-Versión 2



Nuevamente, en esta ocasión hemos mejorado un poco las predicciones sin llegar a ser un modelo óptimo, pues la existencia de residuos elevados sigue estando presente.

El hecho de que existan tantos residuos con valores negativos es indicativo de que las predicciones están resultando ser más elevadas que los datos originales: es decir, estamos dando a entender, en este caso concreto, que los pilotos lograrán peores tiempos, en términos de la competición, de lo que en un principio se esperaría viendo resultados anteriores.

Figura 8.15: Gráficos bondad de ajuste Boosting



Finalmente, en este modelo tenemos tanto un ajuste más o menos bueno, como datos atípicos, como residuos cercanos a cero, en su mayoría, y bastante alejados, por contraposición.

8.1.2. Importancia de variables

Cuadro 8.4: Ranking Versión 1

Variables	Regresión	Árboles	Random Forest
Distancia_comp	53.874	40.069	33.653
Tipo	26.770	13.933	15.125
Fabrica	6.393	9.812	14.998
GranPremio	5.465	8.199	3.499
Categoria	4.727	2.843	3.424
Condiciones_Meteo	1.173	1.107	1.367
Temp_Pista	0.457	6.339	7.540
MaxRecta	0.457	4.120	3.631
Temp_Aire	0.347	5.592	6.428
Fecha	0.253	4.549	6.898
Long_pista	0.085	3.436	3.437

En rasgos generales podemos concluir que tanto la distancia completa a recorrer en cada circuito como el tipo de carrera que se disputó (de una vez o dividida en dos por inclemencias meteorológicas) son las variables que más ponderan en las diferentes versiones 1 de los modelos estudiados.

En Random Forest podríamos incluir la Fábrica como una variable a tener en cuenta a la hora de construir el modelo.

Cuadro 8.5: Ranking Deep Learning-Versión 1

Variables	Deep Learning
Distancia_comp	17.924
Categoria	14.565
Condiciones_Meteo	12.306
Tipo	10.720
GranPremio	8.221
Long_pista	7.761
Fabrica	7.649
Fecha	7.072
MaxRecta	4.770
Temp_Aire	4.597
Temp_Pista	4.414

El análisis de Deep Learning lo haremos de manera separada porque, recordemos, la transformación de las variables no se hizo, aunque sí la selección de las mismas.

En este caso son más las variables que se consideran importantes en el estudio: coincidiendo la distancia completa y el tipo de carrera, pero añadiendo también la categoría

que se disputó y las condiciones meteorológicas bajo las que se realizaron las diferentes carreras.

Cuadro 8.6: Ranking Versión 2

Variables	Regresión	Árboles	Random Forest
Distancia_comp	31.592	26.631	26.187
Vueltas	22.686	19.430	13.494
GranPremio	12.900	8.314	3.316
Tipo	12.199	7.754	3.555
Categoria	5.484	3.982	3.352
Condiciones_Meteo	5.241	2.832	3.312
Fabrica	5.170	8.023	5.804
Nacionalidad	1.570	6.241	13.832
Fecha	0.974	3.387	5.804
sqrt(Temp_Pista)	0.915	4.325	6.399
log(MaxRecta)	0.667	3.318	3.067
log(Temp_Aire)	0.411	3.577	4.967
log(Curvas_izq)	0.193	2.188	1.946

La distancia completada vuelve a ser la que más pondera en los tres modelos, pero en este caso hemos de tener en cuenta también el número de vueltas que se dan, además del Gran Premio en cuestión que se disputa. Estas variables están fuertemente relacionadas, puesto que siempre se corre el mismo número de vueltas en cada circuito, a no ser que se haya realizado algún cambio importante, que no suele ocurrir con mucha frecuencia, y además el circuito siempre mide lo mismo, es decir, se recorre la misma distancia.

Destacable también es el hecho de que en Random Forest la nacionalidad de los pilotos se posiciona como muy importante, lo cual tampoco es sorprendente, puesto que los participantes son en su mayoría pilotos italianos y españoles.

Cuadro 8.7: Ranking Deep Learning-Versión 2

Variables	Deep Learning
Distancia_comp	12.314
Condiciones_Meteo	11.788
Vueltas	10.316
Categoria	9.478
Tipo	8.318
Fecha	7.470
Fabrica	7.168
Curvas_izq	6.921
GranPremio	6.407
Nacionalidad	5.596
MaxRecta	5.189
Temp_Pista	5.060
Temp_Aire	3.976

De nuevo hacemos el estudio de manera diferenciada porque las variables no están transformadas como en los modelos anteriormente estudiados.

En esta ocasión a la distancia completada y las condiciones meteorológicas se le suman las vueltas que se dan en cada circuito.

Cuadro 8.8: Ranking Versión 4

Variables	Regresión	Árboles	Random Forest
Distancia_comp	34.454	26.742	22.505
Vueltas	23.199	20.616	15.095
Tipo	15.221	8.798	4.566
GranPremio	10.615	7.229	3.069
Fabrica	5.141	7.483	11.722
Categoria	4.497	3.204	2.544
Condiciones_Meteo	2.865	1.428	1.929
Nacionalidad	1.602	6.710	15.269
sqrt(Temp_Pista)	0.684	4.581	6.706
log(MaxRecta)	0.617	3.234	3.012
Fecha	0.561	3.253	6.098
log(Temp_Aire)	0.415	4.064	5.600
log(Curvas_izq)	0.129	2.657	1.885

Tenemos ahora, además de la distancia completada, el número de vueltas, siendo éstas las dos variables que coinciden en importancia (aproximadamente) en los tres modelos.

En Random Forest la nacionalidad sigue teniendo un papel importante y la unimos con la fábrica, que vuelve a ser algo lógico puesto que son tres las marcas más competitivas: Honda, Yamaha y Ducati, en rasgos generales.

Cuadro 8.9: Ranking Boosting

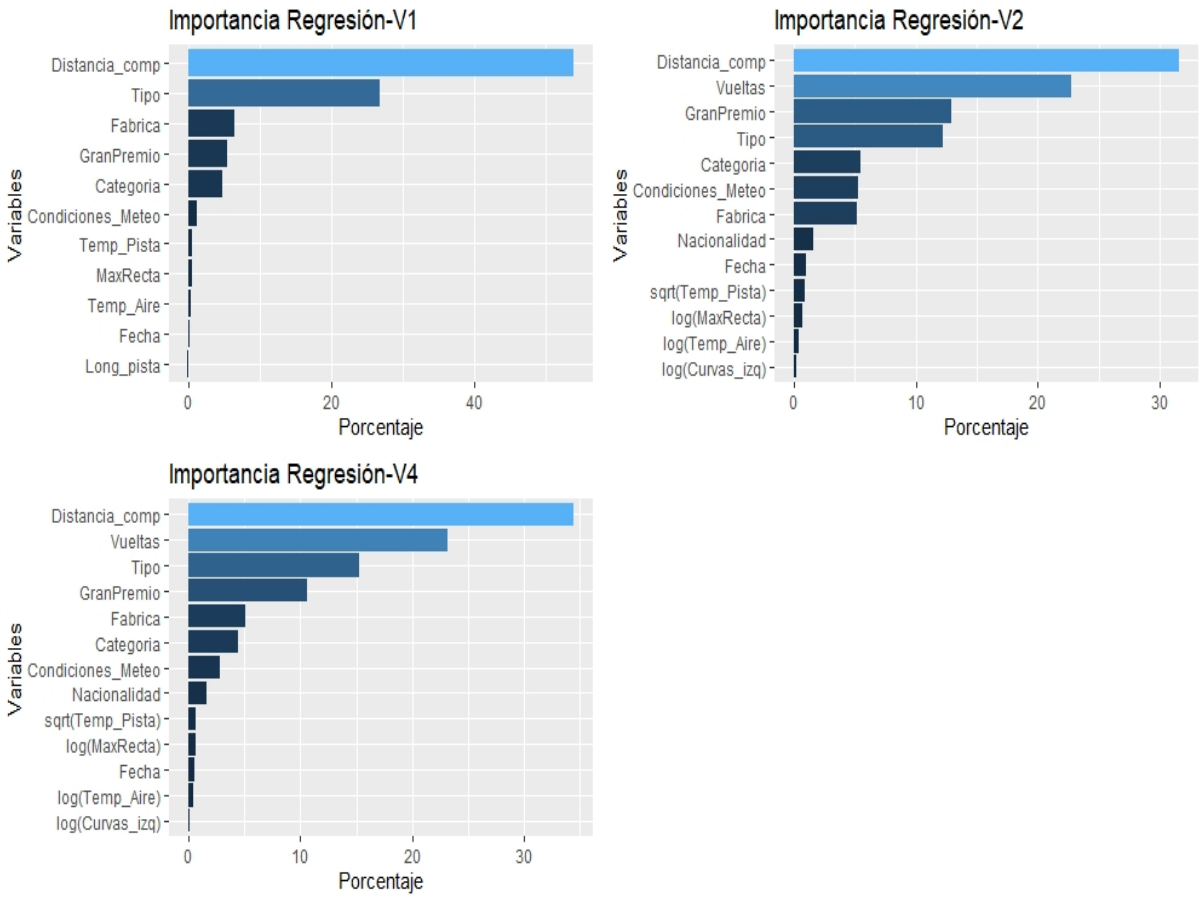
Variables	Boosting
Distancia_comp	60.963
Vueltas	9.956
Equipo	3.875
Piloto	3.596
Temp_Pista	3.373
Nacionalidad	2.311
Fabrica	2.228
Humedad	2.072
Tipo	1.613
Fecha	1.509
Condiciones_Meteo	1.491
MaxRecta	1.297
Temp_Aire	1.226
GranPremio	1.080
Long_pista	0.986
Curvas_dcha	0.723
Categoria	0.607
Anchura	0.598
Curvas_izq	0.497

En el caso de Boosting hacemos también el análisis de manera individual porque no hemos hecho siquiera la selección de variables, por lo que las conclusiones poco tienen que ver con las hechas anteriormente.

En esta ocasión la distancia completada no solo es importante, sino que prácticamente el modelo se basa en esta variable únicamente. Analizaremos este hecho más en profundidad en el gráfico correspondiente en el siguiente apartado.

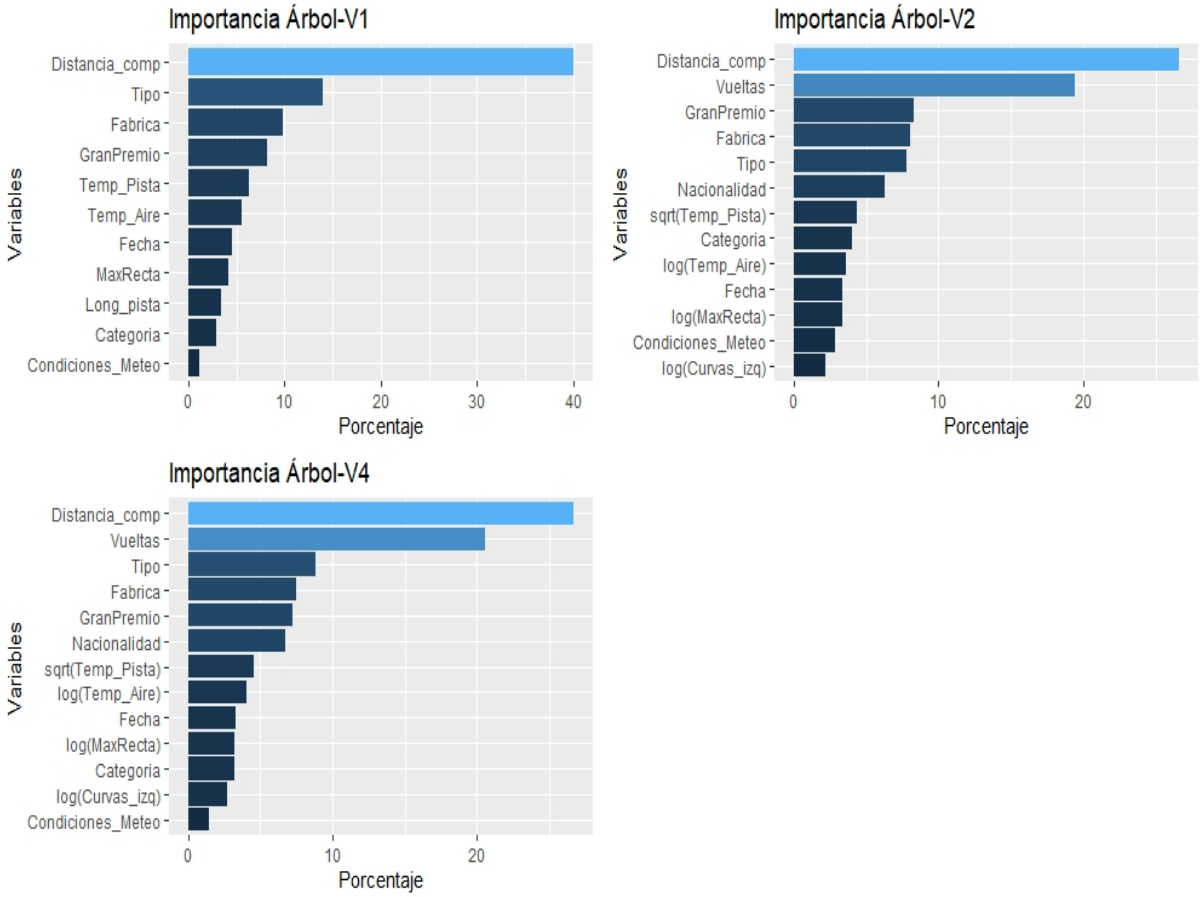
8.1.2.1. Gráficos

Figura 8.16: Gráficos Ranking Regresión



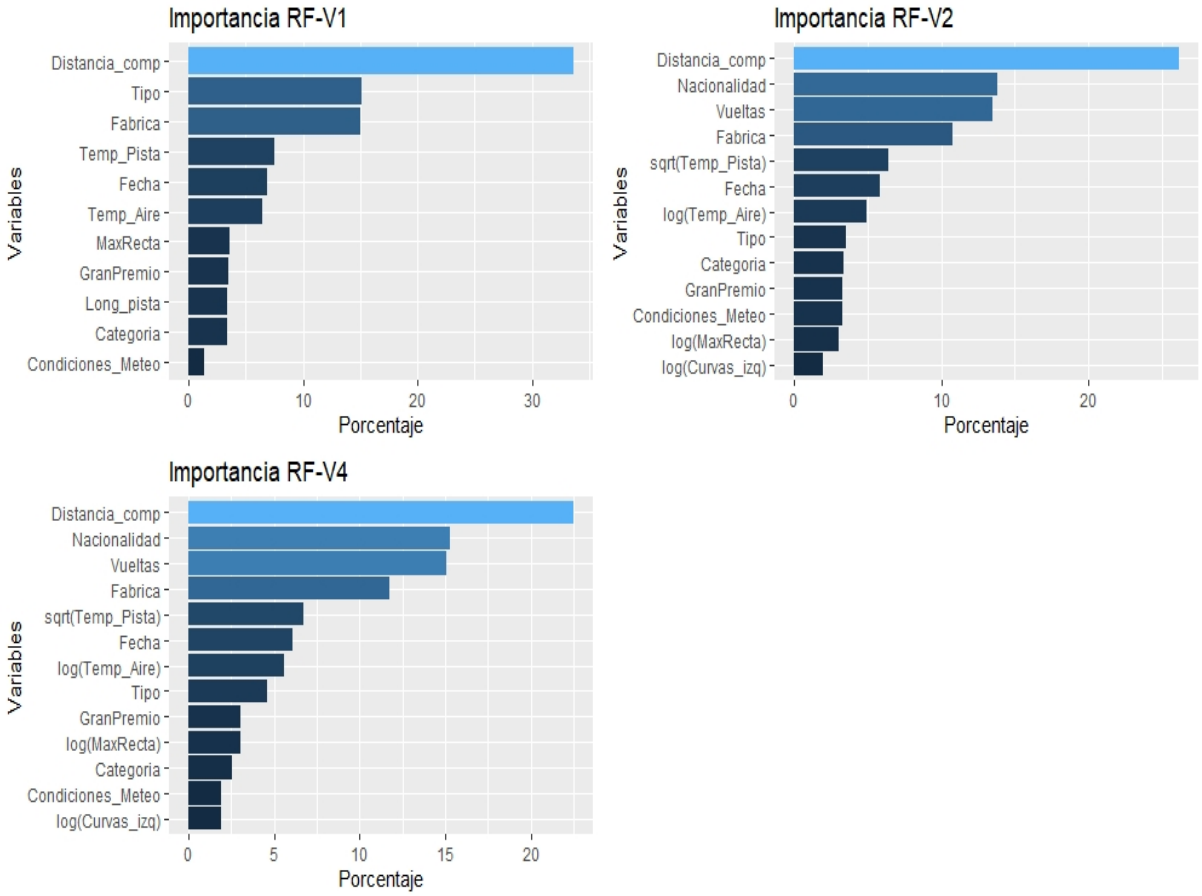
Comentamos ahora las diferencias entre las diferentes versiones del modelo de regresión: lo más reseñable es, quizás, el hecho de que en la versión 2 el modelo se basa en más variables con un porcentaje más repartido que en las otras dos.

Figura 8.17: Gráficos Ranking Árboles



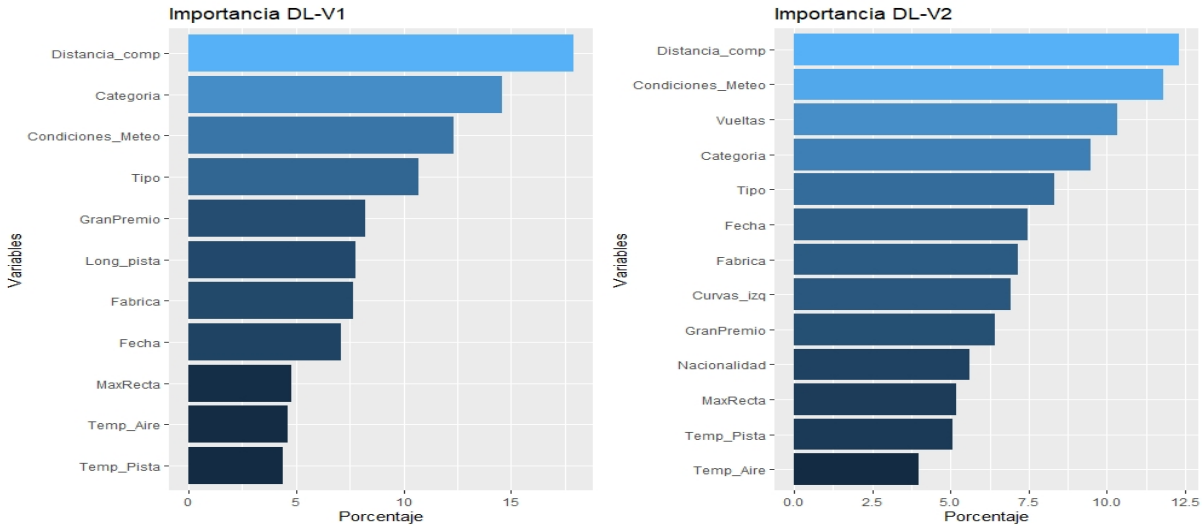
En el caso de los árboles la diferencia más notable la encontramos en la versión 1, puesto que la distancia tiene un papel muy relevante, mientras que en las otras dos le acompaña también el número de vueltas.

Figura 8.18: Gráficos Ranking Random Forest



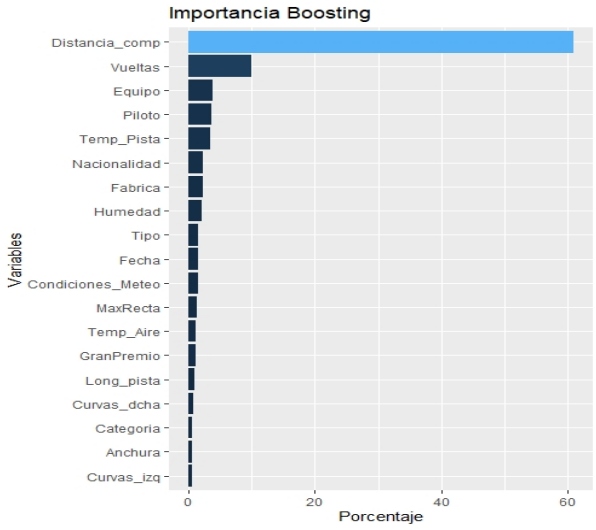
En Random Forest observamos cómo la inclusión de la nacionalidad en las versiones 2 y 4 hace que el modelo considere menos importantes algunas variables como la temperatura de la pista o el aire, para incluir la procedencia de los pilotos.

Figura 8.19: Gráficos Ranking Deep Learning



En Deep Learning se puede observar perfectamente cómo el modelo se construye de manera bastante equitativa, teniendo en cuenta todas las variables con las que se trabaja de una manera similar entre ellas.

Figura 8.20: Gráfico Ranking Boosting



En el caso de Boosting, al tener la distancia un porcentaje tan elevado, hace que en el momento en que se tengan valores algo atípicos en esta variable las predicciones van a obtenerse unos valores muy alejados de la realidad, es decir, es más posible cometer errores. Aun así, hemos visto que el rendimiento del modelo no es del todo malo, aunque sigue sin ser de los más óptimos estudiados.

8.1.3. Conclusión final

Como hemos podido ir analizando poco a poco, ninguno de los modelos estudiados es del todo fiable ni nos ofrece unos resultados buenos por completo, aunque los que mejor rendimiento hemos visto que tenían eran las Versiones 1 de Regresión, Boosting y Deep Learning. Por tanto, puestos a elegir algún modelo con el que seguir el estudio sería alguno de los anteriormente comentados.

Probablemente, si mejoráramos la base de datos obtendríamos unos modelos con resultados mucho más fiables y predicciones mucho más exactas. Por ejemplo:

- Incluyendo los tiempos que consiguen los pilotos durante los entrenamientos previos a la carrera, que son nada más y nada menos que tres sesiones además de la de clasificación, en la que pueden darse condiciones meteorológicas muy diversas.
- Incluyendo el estado físico de cada piloto, conociendo así la resistencia de cada uno de ellos bajo determinadas condiciones.
- Añadiendo también las lesiones que hayan podido sufrir los pilotos.
- Teniendo en cuenta si los peores tiempos logrados por un piloto han sido por algún error propio, de algún otro piloto o fallo mecánico.

Quizás de esta manera conseguiríamos un estudio más completo y por tanto, unas predicciones que se ajustaran mucho más a la realidad, logrando así intuir qué va a suceder en una determinada carrera, teniendo en cuenta todos los antecedentes que ya conoceríamos con exactitud.

Apéndice A

Código en R

A.1. Tratamiento de datos

Cargo las librerías con las que voy a trabajar:

```
library(dplyr)
library(tidyr)
```

Leemos los datos completos:

```
datos1=read.csv("Dataset.csv",header=TRUE,sep=",")
```

Elimino las variables que considero no son útiles para el estudio:

```
datos2=select(datos1, -TRK, -Track, -Date, -Points, -Time,
              -GP_avg_speed, -gp_dist, -m3_dist, -m2_dist,
              -Avg_Speed, -Position, -laps_completed)
```

Construyo una nueva columna que una el dorsal del piloto y su nombre:

```
datos3=unite(datos2, Pilot, c(8:9),  sep = " ", remove = TRUE)
```

Cambio los tipos de datos, donde corresponda, con los que trabajamos:

```
datos3$Pilot=as.factor(datos3$Pilot)
datos3$track_length_km=as.integer(as.character(datos3$track_length_km))
```

Renombro las carreras de 125cc como de Moto3 y las de 250cc como Moto2, por similitud:

```
datos3$Category=as.character(datos3$Category)
datos3$Category[datos3$Category == "125cc"] <- "Moto3"
datos3$Category[datos3$Category == "250cc"] <- "Moto2"
datos3$Category=as.factor(datos3$Category)
```

Elimino los espacios iniciales y finales en la variable Bike, para que los niveles no estén repetidos. Procedo de manera análoga para las nacionalidades y los equipos:

```
datos3$Bike=trimws(x = datos3$Bike, which = "right")
datos3$Bike=trimws(x = datos3$Bike, which = "left")
```

```
datos3$Bike=toupper(datos3$Bike)
datos3$Bike=as.factor(datos3$Bike)

datos3$Nationality=trimws(x = datos3$Nationality, which = "right")
datos3$Nationality=trimws(x = datos3$Nationality, which = "left")
datos3$Nationality=as.factor(datos3$Nationality)

datos3$Team_Name=trimws(x = datos3$Team_Name, which = "right")
datos3$Team_Name=trimws(x = datos3$Team_Name, which = "left")
datos3$Team_Name=as.factor(datos3$Team_Name)
```

Convierto los tiempos de llegada a milisegundos:

```
datos3$Finish_Time=as.character(datos3$Finish_Time)
S=strsplit(datos3$Finish_Time,"'")
A=substr(S,4,5)
B=substr(S,10,11)
C=substr(S,13,15)
datos3$Finish_Time=as.numeric(A)*60000+as.numeric(B)*1000+as.numeric(C)
```

Creo la variable distancia recorrida en cada categoría:

```
datos3$race_dist=datos3$laps*datos3$track_length_km
```

Vamos a arreglar el tema de las carreras en mojado: creo una nueva variable que sea "Tipo", que tomará el valor 1 cuando sea una carrera normal (RAC), 2 cuando se haya dividido en dos y sea la primera parte (RAC1) y 3 cuando sea la segunda parte (RAC2):

```
Tipo=c(NA)
datos4=cbind(datos3,Tipo)
datos4$Tipo[datos4$Session == "RAC"] <- 1
datos4$Tipo[datos4$Session == "RAC1"] <- 2
datos4$Tipo[datos4$Session == "RAC2"] <- 3
datos4$Tipo=as.factor(datos4$Tipo)
```

Reorganizo las variables y las columnas para que la información se vea mejor y las traduzco:

```
datos5=select(datos4, -Session)

datos=datos5[ , c(1,12,2,20,3:11,19,14,13,15:18)]
colnames(datos)=c("Fecha","GranPremio","Categoria","Tipo",
                  "Condiciones_Meteo","Temp_Pista",
                  "Temp_Aire","Humedad","Piloto",
                  "Nacionalidad","Equipo","Fabrica","Tiempo",
                  "Distancia_comp","Vueltas","Long_pista",
                  "Curvas_izq","Curvas_dcha","Anchura","MaxRecta")
# Traduzco también los niveles de las condiciones meteorológicas:
datos$Condiciones_Meteo=recode(datos$Condiciones_Meteo,"Dry"="Seco")
datos$Condiciones_Meteo=recode(datos$Condiciones_Meteo,"Wet"="Mojado")
datos$Condiciones_Meteo=recode(datos$Condiciones_Meteo,"Wet-Dry"="Mixta")
```

Descarto las nacionalidades que no sean representativas:

```
table(datos$Nacionalidad)
datos<-datos[datos$Nacionalidad!="CAN",]
datos<-datos[datos$Nacionalidad!="CRO",]
datos<-datos[datos$Nacionalidad!="DEN",]
datos<-datos[datos$Nacionalidad!="IND",]
datos<-datos[datos$Nacionalidad!="POL",]
datos<-datos[datos$Nacionalidad!="SVK",]
```

Descarto las fábricas que no sean representativas:

```
table(datos$Fabrica)
datos<-datos[datos$Fabrica!="APR",]
datos<-datos[datos$Fabrica!="BAKKER HONDA",]
datos<-datos[datos$Fabrica!="BCL",]
datos<-datos[datos$Fabrica!="BQR-FTR",]
datos<-datos[datos$Fabrica!="BULLET",]
datos<-datos[datos$Fabrica!="CASTROL HONDA",]
datos<-datos[datos$Fabrica!="FGR",]
datos<-datos[datos$Fabrica!="FRIBA",]
datos<-datos[datos$Fabrica!="FTR KTM",]
datos<-datos[datos$Fabrica!="HAOUJE",]
datos<-datos[datos$Fabrica!="HAOJUE",]
datos<-datos[datos$Fabrica!="IAMT",]
datos<-datos[datos$Fabrica!="ILMOR GP",]
datos<-datos[datos$Fabrica!="ILMOR X3",]
datos<-datos[datos$Fabrica!="INMOTEC",]
datos<-datos[datos$Fabrica!="MIR HONDA",]
datos<-datos[datos$Fabrica!="MIR RACING",]
datos<-datos[datos$Fabrica!="MISTRAL 610",]
datos<-datos[datos$Fabrica!="MTA TM RACING",]
datos<-datos[datos$Fabrica!="MZ FTR",]
datos<-datos[datos$Fabrica!="ORAL",]
datos<-datos[datos$Fabrica!="RBB",]
datos<-datos[datos$Fabrica!="RSV",]
datos<-datos[datos$Fabrica!="S&B SUTER",]
datos<-datos[datos$Fabrica!="SEEL",]
datos<-datos[datos$Fabrica!="TAYLOR MADE",]
datos<-datos[datos$Fabrica!="TEN KATE",]
datos<-datos[datos$Fabrica!="TRANSFORMIERS",]
datos<-datos[datos$Fabrica!="TSR 6",]
datos<-datos[datos$Fabrica!="TVR",]
```

Me quedo sólo con los casos completos:

```
datos=datos[complete.cases(datos),]
```

Elimino los niveles no usados:

```
datos$Fabrica=droplevels(datos$Fabrica)
datos$Nacionalidad=droplevels(datos$Nacionalidad)
```

A.2. Conjuntos test y entrenamiento

```
datos=datos[sample(1:nrow(datos)),] #permuto primero
n <- nrow(datos)
set.seed(75911896)
library(caret)
indices=1:nrow(datos)
indient <- createDataPartition(datos$Nacionalidad,
                               p=.75, list=FALSE, times=1)
inditest <- setdiff(indices,indient)
ent <- datos[indient,]
test <- datos[inditest,]

dim(ent); dim(test)
```

A.3. Selección y transformación de variables

A.3.1. Versión 1

```
m1=lm(Tiempo~., data=ent)
modelo1=step(m1,direction="both")
```

A.3.2. Versión 2

```
library(alr3)

summary(powerTransform(cbind(Tiempo,Fecha,Temp_Pista,Temp_Aire,Humedad,
                             Long_pista,Distancia_comp,Curvas_izq,
                             Curvas_dcha,Anchura,MaxRecta)~1,
                       data=ent))

m2=lm(Tiempo^3 ~ Fecha + sqrt(Temp_Pista) + log(Temp_Aire) + Humedad +
                 1/Long_pista + Distancia_comp^4 + log(Curvas_izq) +
                 Curvas_dcha + log(Anchura) + log(MaxRecta) + Vueltas +
                 GranPremio + Categoria + Tipo + Condiciones_Meteo +
                 Piloto + Nacionalidad + Equipo + Fabrica,
       data=ent)

modelo2=step(m2,direction="both")
```

A.3.3. Versión 3

```
summary(powerTransform(cbind(Fecha,Temp_Pista,Temp_Aire,Humedad,
                             Long_pista,Distancia_comp,Curvas_izq,
                             Curvas_dcha,Anchura,MaxRecta)~1,
                       data=ent))

m3=lm(Tiempo ~ Fecha + sqrt(Temp_Pista) + log(Temp_Aire) + Humedad +
               1/Long_pista + Distancia_comp^4 + log(Curvas_izq) +
               Curvas_dcha + log(Anchura) + log(MaxRecta) + Vueltas +
               GranPremio + Categoria + Tipo + Condiciones_Meteo +
               Piloto + Nacionalidad + Equipo + Fabrica,
       data=ent)

inverseResponsePlot(m3,key=TRUE)

m33=lm(Tiempo^3 ~ Fecha + sqrt(Temp_Pista) + log(Temp_Aire) + Humedad +
               1/Long_pista + Distancia_comp^4 + log(Curvas_izq) +
               Curvas_dcha + log(Anchura) + log(MaxRecta) + Vueltas +
               GranPremio + Categoria + Tipo + Condiciones_Meteo +
               Piloto + Nacionalidad + Equipo + Fabrica,
      data=ent)

modelo3=step(m33,direction="both")
```

A.3.4. Versión 4

```
library(MASS)

A=boxcox(m3)
A$x[which.max(A$y)]

m4=lm(Tiempo^2 ~ Fecha + sqrt(Temp_Pista) + log(Temp_Aire) + Humedad +
                 1/Long_pista + Distancia_comp^4 + log(Curvas_izq) +
                 Curvas_dcha + log(Anchura) + log(MaxRecta) + Vueltas +
                 GranPremio + Categoria + Tipo + Condiciones_Meteo +
                 Piloto + Nacionalidad + Equipo + Fabrica,
       data=ent)

modelo4=step(m4,direction="both")
```

A.4. Función bondad de ajuste

```
bondad <- function (y,pred,nom) {
  n=length(y)
  pred<-ifelse(is.na(pred),0,pred)
```

```
  residuos=y-pred
  MSE=mean(residuos^2)/1000000
  RMSE=sqrt(MSE)
  R2=cor(y,pred)^2
  PEMA=(sum(100*abs(y-pred)/y))/n

  tabla=data.frame(MSE,RMSE,R2,PEMA); print(tabla)

  plot(y, pred, main=nom, xlab="Tiempos", ylab=expression(hat(y)))

  plot(pred, residuos, main=nom,
       xlab=expression(hat(y)), ylab="Residuos")
}
```

A.5. Modelo regresión

A.5.1. Versión 1

```
summary(modelo1)

y1_reg_ent=ent$Tiempo
pred1_reg_ent=predict(modelo1,ent)
bondad(y1_reg_ent,pred1_reg_ent,
       "(Regresión-Versión 1, Conjunto entrenamiento)")

y1_reg_test=test$Tiempo
pred1_reg_test=predict(modelo1,test)
bondad(y1_reg_test,pred1_reg_test,"(Regresión-Versión 1, Conjunto test)")

library(relaimpo)
imp1=calc.relimp(modelo1, type="lmg")$lmg
Porcent1=round(100*imp1/sum(imp1),3)
imp1=as.data.frame(Porcent1)
imp1=rownames_to_column(imp1, var="variable")
graf_imp_reg1 <- ggplot(data = imp1,
                        aes(x = reorder(variable, Porcent1),
                            y = Porcent1,
                            fill = Porcent1)) +
                  labs(x = "Variables", y = "Porcentaje",
                       title = "Importancia Regresión-V1") +
                  geom_col(show.legend = FALSE) +
                  coord_flip()
```

A.5.2. Versión 2

```
summary(modelo2)

y2_reg_ent=ent$Tiempo
pred2_reg_ent=predict(modelo2,ent)^(1/3)
bondad(y2_reg_ent,pred2_reg_ent,
       "(Regresión-Versión 2, Conjunto entrenamiento)")

y2_reg_test=test$Tiempo
pred2_reg_test=predict(modelo2,test)^(1/3)
bondad(y2_reg_test,pred2_reg_test,"(Regresión-Versión 2, Conjunto test)")

imp2=calc.relimp(modelo2, type="lmg")$lmg
Porcent2=round(100*imp2/sum(imp2),3)
imp2=as.data.frame(Porcent2)
imp2=rownames_to_column(imp2, var="variable")
graf_imp_reg2 <- ggplot(data = imp2,
                        aes(x = reorder(variable, Porcent2),
                            y = Porcent2,
                            fill = Porcent2)) +
                  labs(x = "Variables", y = "Porcentaje",
                       title = "Importancia Regresión-V2") +
                  geom_col(show.legend = FALSE) +
                  coord_flip()
```

A.5.3. Versión 4

```
summary(modelo4)

y4_reg_ent=ent$Tiempo
pred4_reg_ent=sqrt(predict(modelo4,ent))
bondad(y4_reg_ent,pred4_reg_ent,
       "(Regresión-Versión 4, Conjunto entrenamiento)")

y4_reg_test=test$Tiempo
pred4_reg_test=sqrt(predict(modelo4,test))
bondad(y4_reg_test,pred4_reg_test,"(Regresión-Versión 4, Conjunto test)")

imp4=calc.relimp(modelo4, type="lmg")$lmg
Porcent4=round(100*imp4/sum(imp4),3)
imp4=as.data.frame(Porcent4)
imp4=rownames_to_column(imp4, var="variable")
graf_imp_reg4 <- ggplot(data = imp4,
                        aes(x = reorder(variable, Porcent4),
                            y = Porcent4,
                            fill = Porcent4)) +
```

```
                labs(x = "Variables", y = "Porcentaje",
                     title = "Importancia Regresión-V4") +
                geom_col(show.legend = FALSE) +
                coord_flip()
```

A.6. Modelo árboles

A.6.1. Versión 1

```
arbol1_raw <- rpart(Tiempo ~ Fecha + GranPremio + Categoria + Tipo +
                Condiciones_Meteo + Temp_Pista + Temp_Aire +
                Fabrica + Distancia_comp + Long_pista +
                MaxRecta,
                cp=0.0001, ent)
arbol1_raw
library(DMwR)
arbol1=rt.prune(arbol1_raw)
arbol1
summary(arbol1)

library(rpart.plot)
rpart.plot(arbol1, roundint=FALSE)

y1_arb_ent=ent$Tiempo
pred1_arb_ent=predict(arbol1,ent)
bondad(y1_arb_ent,pred1_arb_ent,
       "(Árboles-Versión 1, Conjunto entrenamiento)")

y1_arb_test=test$Tiempo
pred1_arb_test=predict(arbol1,test)
bondad(y1_arb_test,pred1_arb_test,"(Árboles-Versión 1, Conjunto test)")

importancia1=arbol1_raw$variable.importance
Porc1=round(100*importancia1/sum(importancia1),3)
importancia1=as.data.frame(Porc1)
importancia1=rownames_to_column(importancia1, var="variable")
graf_imp_arb1 <- ggplot(data = importancia1,
                        aes(x = reorder(variable, Porc1),
                            y = Porc1,
                            fill = Porc1)) +
                labs(x = "Variables", y = "Porcentaje",
                     title = "Importancia Árbol-V1") +
                geom_col(show.legend = FALSE) +
                coord_flip()
```

A.6.2. Versión 2

```
arbol2_raw <- rpart(Tiempo^3 ~ Fecha + sqrt(Temp_Pista) + log(Temp_Aire) +
                Distancia_comp + log(Curvas_izq) + log(MaxRecta) +
                Vueltas + GranPremio + Categoria + Tipo +
                Condiciones_Meteo + Fabrica + Nacionalidad,
                cp=0.0001, ent)
arbol2_raw
arbol2=rt.prune(arbol2_raw)
arbol2
summary(arbol2)

rpart.plot(arbol2, roundint=FALSE)

y2_arb_ent=ent$Tiempo
pred2_arb_ent=predict(arbol2,ent)^(1/3)
bondad(y2_arb_ent,pred2_arb_ent,
       "(Árboles-Versión 2, Conjunto entrenamiento)")

y2_arb_test=test$Tiempo
pred2_arb_test=predict(arbol2,test)^(1/3)
bondad(y2_arb_test,pred2_arb_test,"(Árboles-Versión 2, Conjunto test)")

importancia2=arbol2_raw$variable.importance
Porc2=round(100*importancia2/sum(importancia2),3)
importancia2=as.data.frame(Porc2)
importancia2=rownames_to_column(importancia2, var="variable")
graf_imp_arb2 <- ggplot(data = importancia2,
                        aes(x = reorder(variable, Porc2),
                            y = Porc2,
                            fill = Porc2)) +
                 labs(x = "Variables", y = "Porcentaje",
                      title = "Importancia Árbol-V2") +
                 geom_col(show.legend = FALSE) +
                 coord_flip()
```

A.6.3. Versión 4

```
arbol4_raw <- rpart(Tiempo^2 ~ Fecha + sqrt(Temp_Pista) + log(Temp_Aire) +
                Distancia_comp + log(Curvas_izq) + log(MaxRecta) +
                Vueltas + GranPremio + Categoria + Tipo +
                Condiciones_Meteo + Fabrica + Nacionalidad,
                cp=0.0001, ent)
arbol4_raw
arbol4=rt.prune(arbol4_raw)
arbol4
summary(arbol4)
```

```
rpart.plot(arbol4, roundint=FALSE)

y4_arb_ent=ent$Tiempo
pred4_arb_ent=sqrt(predict(arbol4,ent))
bondad(y4_arb_ent,pred4_arb_ent,
       "(Árboles-Versión 4, Conjunto entrenamiento)")

y4_arb_test=test$Tiempo
pred4_arb_test=sqrt(predict(arbol4,test))
bondad(y4_arb_test,pred4_arb_test,"(Árboles-Versión 4, Conjunto test)")

importancia4=arbol4_raw$variable.importance
Porc4=round(100*importancia4/sum(importancia4),3)
importancia4=as.data.frame(Porc4)
importancia4=rownames_to_column(importancia4, var="variable")
graf_imp_arb4 <- ggplot(data = importancia4,
                        aes(x = reorder(variable, Porc4),
                            y = Porc4,
                            fill = Porc4)) +
                 labs(x = "Variables", y = "Porcentaje",
                      title = "Importancia Árbol-V4") +
                 geom_col(show.legend = FALSE) +
                 coord_flip()
```

A.7. Modelo Random Forest

A.7.1. Codificación variables categóricas

```
ent.RF=ent
test.RF=test

library(CatEncoders)

factors_ent <- names(which(sapply(ent, is.factor)))
for (i in factors_ent){
  encode <- LabelEncoder.fit(ent.RF[, i])
  ent.RF[, i] <- transform(encode, ent.RF[, i])
}
ent.RF

factors_test <- names(which(sapply(test, is.factor)))
for (i in factors_test){
  encode <- LabelEncoder.fit(test.RF[, i])
  test.RF[, i] <- transform(encode, test.RF[, i])
}
test.RF
```

A.7.2. Versión 1

```
library(randomForest)
RF1 <- randomForest(Tiempo ~ Fecha + GranPremio + Categoria + Tipo +
                      Condiciones_Meteo + Temp_Pista + Temp_Aire +
                      Fabrica + Distancia_comp + Long_pista + MaxRecta,
                    ent)
RF1
summary(RF1)
library(caret)
y1_RF_ent=ent.RF$Tiempo
pred1_RF_ent=predict(RF1,ent.RF)
bondad(y1_RF_ent,pred1_RF_ent,
       "(Random Forest-Versión 1, Conjunto entrenamiento)")

y1_RF_test=test.RF$Tiempo
pred1_RF_test=predict(RF1,test.RF)
bondad(y1_RF_test,pred1_RF_test,
       "(Random Forest-Versión 1, Conjunto test)")

library(tidyverse)
importancia_pred1=as.data.frame(importance(RF1, scale = TRUE))
importancia_pred1=rownames_to_column(importancia_pred1, var = "variable")
Porcentaje1=round(100*importancia_pred1$IncNodePurity/
                    sum(importancia_pred1$IncNodePurity),3)
importancia_pred1=cbind(importancia_pred1, Porcentaje1)
graf_imp_RF1 <- ggplot(data = importancia_pred1,
                       aes(x = reorder(variable, Porcentaje1),
                           y = Porcentaje1,
                           fill = Porcentaje1)) +
                labs(x = "Variables", y = "Porcentaje",
                     title = "Importancia RF-V1") +
                geom_col(show.legend = FALSE) +
                coord_flip()
```

A.7.3. Versión 2

```
RF2 <- randomForest(Tiempo^3 ~ Fecha + sqrt(Temp_Pista) + log(Temp_Aire) +
                      Distancia_comp + log(Curvas_izq) + log(MaxRecta) +
                      Vueltas + GranPremio + Categoria + Tipo +
                      Condiciones_Meteo + Fabrica + Nacionalidad,
                    ent)
RF2
summary(RF2)

y2_RF_ent=ent.RF$Tiempo
pred2_RF_ent=predict(RF2,ent.RF)^(1/3)
```

```
bondad(y2_RF_ent,pred2_RF_ent,
       "(Random Forest-Versión 2, Conjunto entrenamiento)")

y2_RF_test=test.RF$Tiempo
pred2_RF_test=predict(RF2,test.RF)^(1/3)
bondad(y2_RF_test,pred2_RF_test,
       "(Random Forest-Versión 2, Conjunto test)")

importancia_pred2=as.data.frame(importance(RF2, scale = TRUE))
importancia_pred2=rownames_to_column(importancia_pred2, var = "variable")
Porcentaje2=round(100*importancia_pred2$IncNodePurity/
                    sum(importancia_pred2$IncNodePurity),3)
importancia_pred2=cbind(importancia_pred2, Porcentaje2)
graf_imp_RF2 <- ggplot(data = importancia_pred2,
                       aes(x = reorder(variable, Porcentaje2),
                           y = Porcentaje2,
                           fill = Porcentaje2)) +
                labs(x = "Variables", y = "Porcentaje",
                     title = "Importancia RF-V2") +
                geom_col(show.legend = FALSE) +
                coord_flip()
```

A.7.4. Versión 4

```
RF4 <- randomForest(Tiempo^2 ~ Fecha + sqrt(Temp_Pista) + log(Temp_Aire) +
                      Distancia_comp + log(Curvas_izq) + log(MaxRecta) +
                      Vueltas + GranPremio + Categoria + Tipo +
                      Condiciones_Meteo + Fabrica + Nacionalidad,
                    ent)
RF4
summary(RF4)

y4_RF_ent=ent.RF$Tiempo
pred4_RF_ent=sqrt(predict(RF4,ent.RF))
bondad(y4_RF_ent,pred4_RF_ent,
       "(Random Forest-Versión 4, Conjunto entrenamiento)")

y4_RF_test=test.RF$Tiempo
pred4_RF_test=sqrt(predict(RF4,test.RF))
bondad(y4_RF_test,pred4_RF_test,
       "(Random Forest-Versión 4, Conjunto test)")

importancia_pred4=as.data.frame(importance(RF4, scale = TRUE))
importancia_pred4=rownames_to_column(importancia_pred4, var = "variable")
Porcentaje4=round(100*importancia_pred4$IncNodePurity/
                    sum(importancia_pred4$IncNodePurity),3)
importancia_pred4=cbind(importancia_pred4, Porcentaje4)
```

```
graf_imp_RF4 <- ggplot(data = importancia_pred4,
                       aes(x = reorder(variable, Porcentaje4),
                           y = Porcentaje4,
                           fill = Porcentaje4)) +
                labs(x = "Variables", y = "Porcentaje",
                     title = "Importancia RF-V4") +
                geom_col(show.legend = FALSE) +
                coord_flip()
```

A.8. Modelo Boosting

A.8.1. Codificación variables categóricas

```
ent.boost=ent
test.boost=test

library(CatEncoders)

factors_ent <- names(which(sapply(ent, is.factor)))
for (i in factors_ent){
  encode <- LabelEncoder.fit(ent.boost[, i])
  ent.boost[, i] <- transform(encode, ent.boost[, i])
}
ent.boost

factors_test <- names(which(sapply(test, is.factor)))
for (i in factors_test){
  encode <- LabelEncoder.fit(test.boost[, i])
  test.boost[, i] <- transform(encode, test.boost[, i])
}
test.boost

ent.boost=data.matrix(ent.boost)
test.boost=data.matrix(test.boost)
```

A.8.2. Configuración de parámetros

```
library(caret)

nrounds <- 1000

tune_grid <- expand.grid(
  nrounds = seq(from = 200, to = nrounds, by = 50),
  eta = c(0.025, 0.05, 0.1, 0.3),
  max_depth = c(2, 3, 4, 5, 6),
  gamma = 0,
```

```
  colsample_bytree = 1,
  min_child_weight = 1,
  subsample = 1
)

tune_control <- trainControl(
  method = "cv",
  number = 3,
  verboseIter = FALSE,
  allowParallel = TRUE
)

xgb_tune <- train(
  x = ent.boost[,-13],
  y = ent.boost[,13],
  trControl = tune_control,
  tuneGrid = tune_grid,
  method = "xgbTree",
  verbose = TRUE
)

tuneplot <- function(x, probs = .90) {
  ggplot(x) +
    coord_cartesian(ylim = c(quantile(x$results$RMSE, probs = probs),
                             min(x$results$RMSE))) +
    theme_bw()
}

tuneplot(xgb_tune)

xgb_tune$bestTune

tune_grid2 <- expand.grid(
  nrounds = seq(from = 50, to = nrounds, by = 50),
  eta = xgb_tune$bestTune$eta,
  max_depth = ifelse(xgb_tune$bestTune$max_depth == 2,
                     c(xgb_tune$bestTune$max_depth:4),
                     xgb_tune$bestTune$max_depth -
                       1:xgb_tune$bestTune$max_depth + 1),
  gamma = 0,
  colsample_bytree = 1,
  min_child_weight = c(1, 2, 3),
  subsample = 1
)

xgb_tune2 <- train(
  x = ent.boost[,-13],
```

```
  y = ent.boost[,13],
  trControl = tune_control,
  tuneGrid = tune_grid2,
  method = "xgbTree",
  verbose = TRUE
)

tuneplot(xgb_tune2)

xgb_tune2$bestTune

tune_grid3 <- expand.grid(
  nrounds = seq(from = 50, to = nrounds, by = 50),
  eta = xgb_tune$bestTune$eta,
  max_depth = xgb_tune2$bestTune$max_depth,
  gamma = 0,
  colsample_bytree = c(0.4, 0.6, 0.8, 1.0),
  min_child_weight = xgb_tune2$bestTune$min_child_weight,
  subsample = c(0.5, 0.75, 1.0)
)

xgb_tune3 <- train(
  x = ent.boost[,-13],
  y = ent.boost[,13],
  trControl = tune_control,
  tuneGrid = tune_grid3,
  method = "xgbTree",
  verbose = TRUE
)

tuneplot(xgb_tune3, probs = .95)

xgb_tune3$bestTune

tune_grid4 <- expand.grid(
  nrounds = seq(from = 50, to = nrounds, by = 50),
  eta = xgb_tune$bestTune$eta,
  max_depth = xgb_tune2$bestTune$max_depth,
  gamma = c(0, 0.05, 0.1, 0.5, 0.7, 0.9, 1.0),
  colsample_bytree = xgb_tune3$bestTune$colsample_bytree,
  min_child_weight = xgb_tune2$bestTune$min_child_weight,
  subsample = xgb_tune3$bestTune$subsample
)

xgb_tune4 <- train(
  x = ent.boost[,-13],
  y = ent.boost[,13],
```

```
  trControl = tune_control,
  tuneGrid = tune_grid4,
  method = "xgbTree",
  verbose = TRUE
)

tuneplot(xgb_tune4)

xgb_tune4$bestTune

tune_grid5 <- expand.grid(
  nrounds = seq(from = 100, to = 10000, by = 100),
  eta = c(0.01, 0.015, 0.025, 0.05, 0.1),
  max_depth = xgb_tune2$bestTune$max_depth,
  gamma = xgb_tune4$bestTune$gamma,
  colsample_bytree = xgb_tune3$bestTune$colsample_bytree,
  min_child_weight = xgb_tune2$bestTune$min_child_weight,
  subsample = xgb_tune3$bestTune$subsample
)

xgb_tune5 <- train(
  x = ent.boost[,-13],
  y = ent.boost[,13],
  trControl = tune_control,
  tuneGrid = tune_grid5,
  method = "xgbTree",
  verbose = TRUE
)

tuneplot(xgb_tune5)

xgb_tune5$bestTune
```

A.8.3. Versión 1

```
library(xgboost)

xgb.fit <- xgboost(data = ent.boost[,-13], label=ent.boost[,13],
                   max.depth = xgb_tune5$bestTune$max_depth,
                   eta = xgb_tune5$bestTune$eta,
                   nrounds = xgb_tune5$bestTune$nrounds,
                   gamma = xgb_tune5$bestTune$gamma,
                   colsample_bytree = xgb_tune5$bestTune$colsample_bytree,
                   min_child_weight = xgb_tune5$bestTune$min_child_weight,
                   subsample = xgb_tune5$bestTune$subsample,
                   objective="reg:squarederror")
```

```
y1_bst_ent=ent.boost[,13]
pred1_bst_ent=predict(xgb.fit, ent.boost[,-13])
bondad(y1_bst_ent,pred1_bst_ent,
       "(Boosting-Versión 1, Conjunto entrenamiento)")

y1_bst_test=test.boost[,13]
pred1_bst_test=predict(xgb.fit, test.boost[,-13])
bondad(y1_bst_test,pred1_bst_test,"(Boosting-Versión 1, Conjunto test)")

importance1=xgb.importance(feature_names = colnames(ent.boost[,-13]),
                           model= xgb.fit)
importance11=as.data.frame(cbind(Variables=importance1$Feature,
                    Porcentaje=round(100*importance1$Gain/
                                       sum(importance1$Gain),3)),
                    stringsAsFactors=FALSE)
importance11$Porcentaje=as.numeric(importance11$Porcentaje)

graf_imp_boost <- ggplot(data = importance11,
                         aes(x = reorder(Variables, Porcentaje),
                             y = Porcentaje,
                             fill = Porcentaje)) +
                  labs(x = "Variables", y = "Porcentaje",
                       title = "Importancia Boosting") +
                  geom_col(show.legend = FALSE) +
                  coord_flip()
```

A.9. Modelo Deep Learning

A.9.1. Codificación variables categóricas

```
datos=datos[ , c(13,1,6:8,14:20,2:5,9:12)]
ent=ent[ , c(13,1,6:8,14:20,2:5,9:12)]
test=test[ , c(13,1,6:8,14:20,2:5,9:12)]

summary(ent[,2:12])

summary(datos$Tiempo) #max=3275570

ent2=ent
ent2$Tiempo=ent$Tiempo/3275570
test2=test
test2$Tiempo=test$Tiempo/3275570

ent.DL=ent2
test.DL=test2
```

```
library(CatEncoders)

factors_ent <- names(which(sapply(ent2, is.factor)))
for (i in factors_ent){
  encode <- LabelEncoder.fit(ent.DL[, i])
  ent.DL[, i] <- transform(encode, ent.DL[, i])
}
ent.DL

factors_test <- names(which(sapply(test2, is.factor)))
for (i in factors_test){
  encode <- LabelEncoder.fit(test.DL[, i])
  test.DL[, i] <- transform(encode, test.DL[, i])
}
test.DL

library(h2o)
h2o.init()

ent.hex<-as.h2o(ent.DL)
test.hex<-as.h2o(test.DL)
```

A.9.2. Versión 1

```
DL1 <- h2o.deeplearning(x = c(2,13:16,3,4,20,6,8,12),
                        y = 1,
                        standardize = TRUE,
                        training_frame = ent.hex,
                        validation_frame = test.hex,
                        distribution="gaussian",
                        activation = 'RectifierWithDropout',
                        hidden = c(250, 250),
                        hidden_dropout_ratio = c(0.5, 0.5),
                        input_dropout_ratio = 0.2,
                        epochs = 50,
                        l1 = 1e-5,
                        l2 = 1e-5,
                        rho = 0.99,
                        epsilon = 1e-8,
                        variable_importances = TRUE)

y1_DL_ent=(ent.DL$Tiempo)*3275570
pred1_DL_ent=as.data.frame(h2o.predict(DL1,
                                       newdata=ent.hex))$predict*3275570
bondad(y1_DL_ent,pred1_DL_ent,
       "(Deep Learning-Versión 1, Conjunto entrenamiento)")
```

```
y1_DL_test=(test.DL$Tiempo)*3275570
pred1_DL_test=as.data.frame(h2o.predict(DL1,
                                        newdata=test.hex))$predict*3275570
bondad(y1_DL_test,pred1_DL_test,
       "(Deep Learning-Versión 1, Conjunto test)")

impDL1=as.data.frame(cbind(h2o.varimp(DL1)$variable,
                           round(h2o.varimp(DL1)$percentage*100,3)),
                     stringsAsFactors=FALSE)
impDL1$V2=as.numeric(impDL1$V2)
impDL1

graf_imp_DL1 <- ggplot(data = impDL1,
                       aes(x = reorder(V1, V2),
                           y = V2,
                           fill = V2)) +
                labs(x = "Variables", y = "Porcentaje",
                     title = "Importancia DL-V1") +
                geom_col(show.legend = FALSE) +
                coord_flip()
```

A.9.3. Versión 2

```
DL2 <- h2o.deeplearning(x = c(2:4,6,9,12,7,13:16,20,18),
                        y = 1,
                        standardize = TRUE,
                        training_frame = ent.hex,
                        validation_frame = test.hex,
                        distribution="gaussian",
                        activation = 'RectifierWithDropout',
                        hidden = c(250, 250),
                        hidden_dropout_ratio = c(0.5, 0.5),
                        input_dropout_ratio = 0.2,
                        epochs = 50,
                        l1 = 1e-5,
                        l2 = 1e-5,
                        rho = 0.99,
                        epsilon = 1e-8,
                        variable_importances = TRUE)

y2_DL_ent=(ent.DL$Tiempo)*3275570
pred2_DL_ent=as.data.frame(h2o.predict(DL2,
                                       newdata=ent.hex))$predict*3275570
bondad(y2_DL_ent,pred2_DL_ent,
       "(Deep Learning-Versión 2, Conjunto entrenamiento)")

y2_DL_test=(test.DL$Tiempo)*3275570
```

```
pred2_DL_test=as.data.frame(h2o.predict(DL2,
                                        newdata=test.hex))$predict*3275570
bondad(y2_DL_test,pred2_DL_test,
       "(Deep Learning-Versión 2, Conjunto test)")

impDL2=as.data.frame(cbind(h2o.varimp(DL2)$variable,
                           round(h2o.varimp(DL2)$percentage*100,3)),
                     stringsAsFactors=FALSE)
impDL2$V2=as.numeric(impDL2$V2)
impDL2

graf_imp_DL2 <- ggplot(data = impDL2,
                       aes(x = reorder(V1, V2),
                           y = V2,
                           fill = V2)) +
                labs(x = "Variables", y = "Porcentaje",
                     title = "Importancia DL-V2") +
                geom_col(show.legend = FALSE) +
                coord_flip()
```

Apéndice B

Gráficas

B.1. Boosting

Figura B.1: Tune Max Tree Depth

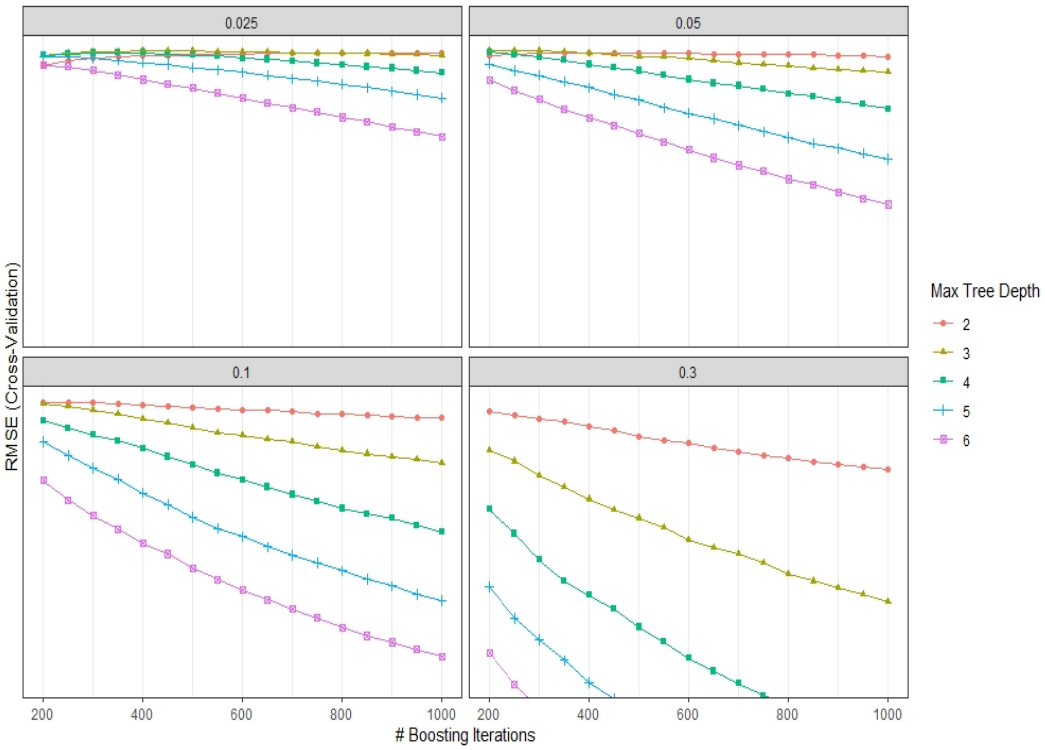


Figura B.2: Tune Minimum Sum of Instance Weight

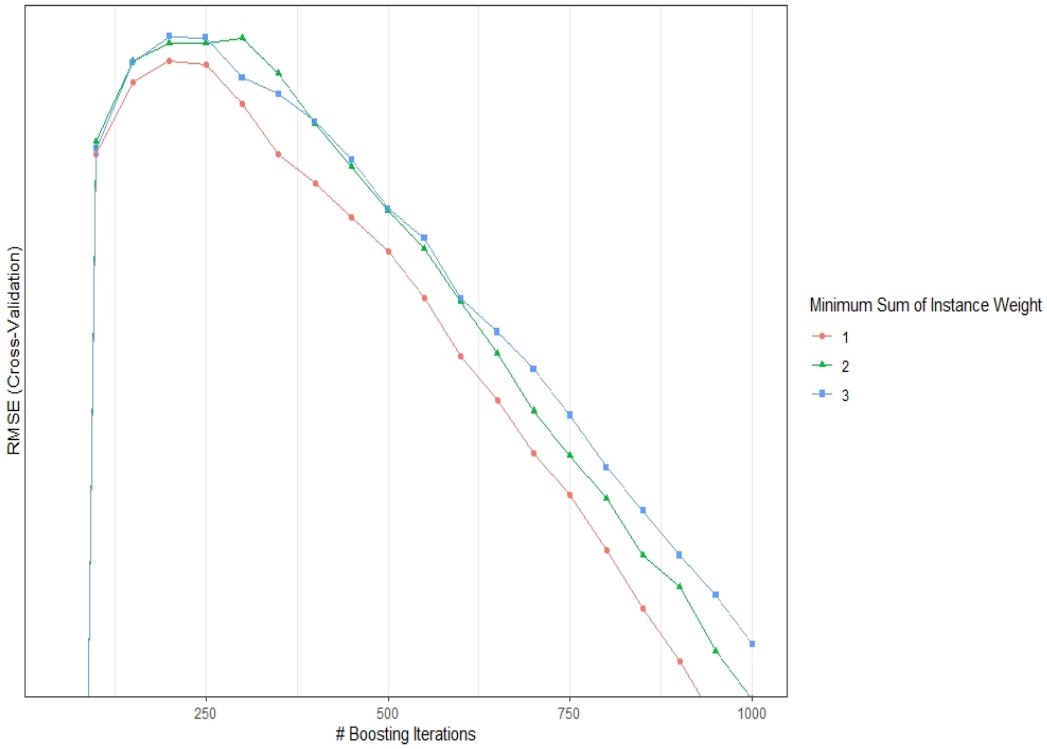


Figura B.3: Tune Subsample Ratio of Columns

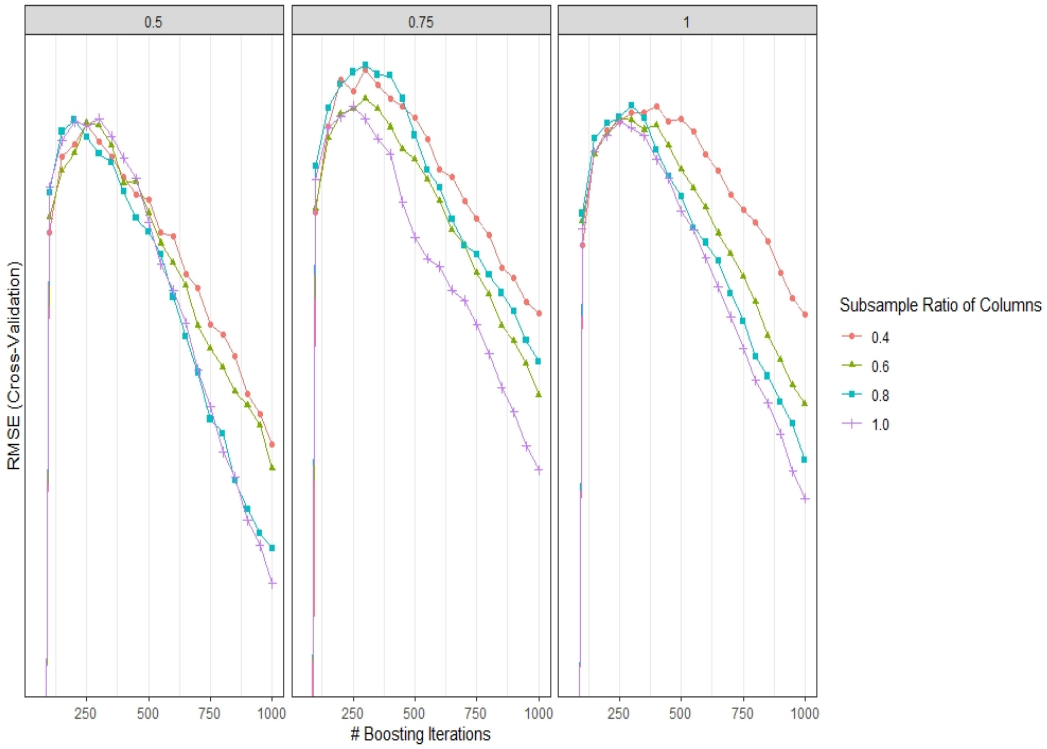


Figura B.4: Tune Minimum Loss Reduction

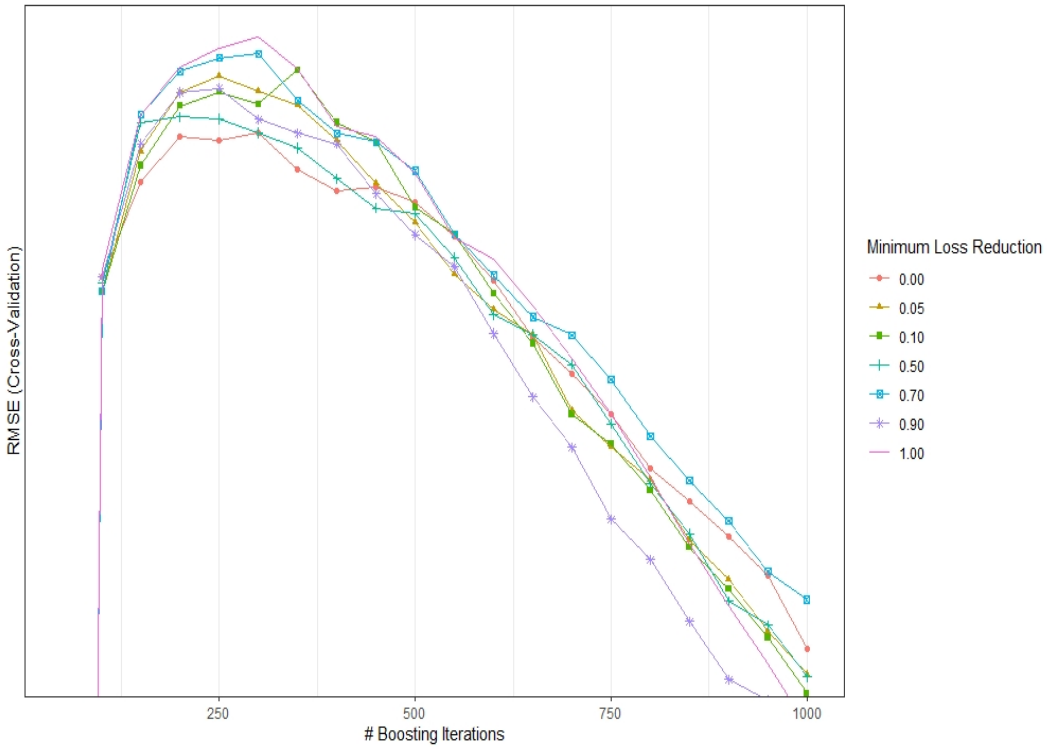
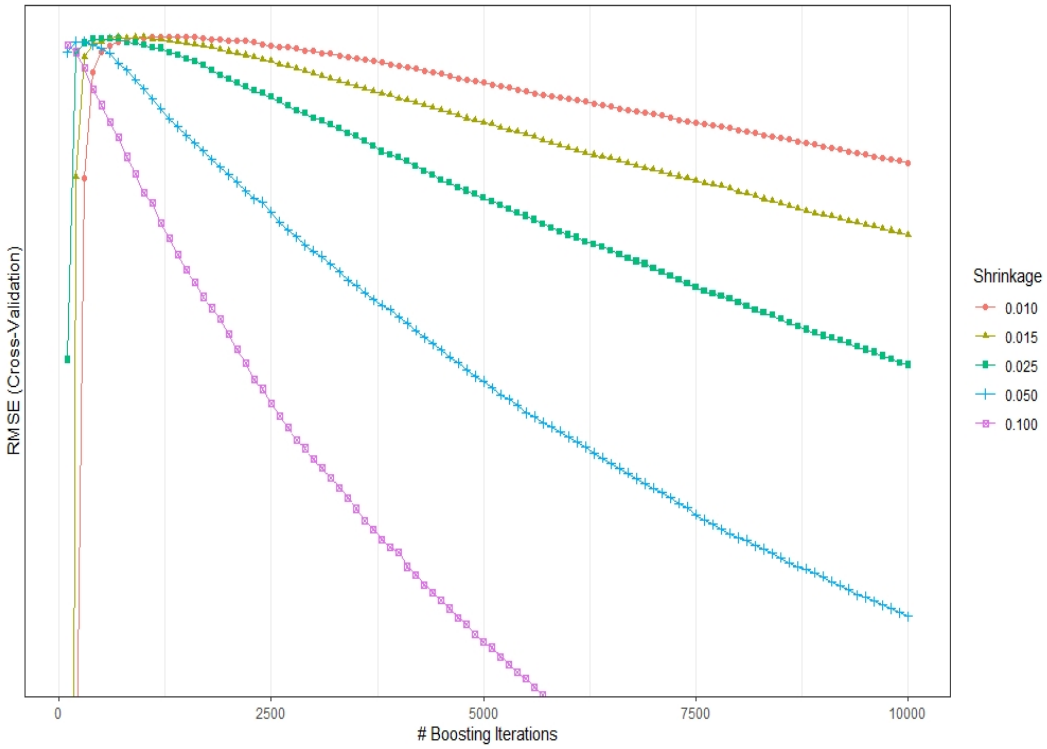


Figura B.5: Tune Shrinkage



Bibliografía

JJ Allaire, Yihui Xie, Jonathan McPherson, Javier Luraschi, Kevin Ushey, Aron Atkins, Hadley Wickham, Joe Cheng, Winston Chang, and Richard Iannone. *rmarkdown: Dynamic Documents for R*, 2019. URL https://CRAN.R-project.org/package=rmarkdown. R package version 1.12.

Johanna Orellana Alvear. *Arboles de decision y Random Forest*, 2018. URL https://bookdown.org/content/2031/ensambladores-random-forest-parte-i.html.

Anton Antonov Baptiste Auguie. *gridExtra: Miscellaneous Functions for "Grid" Graphics*, 2017. URL https://CRAN.R-project.org/package=gridExtra. R package version 2.3.

Navdeep Gill Erin LeDell, Spencer Aiello, Anqi Fu, Arno Candel, Cliff Click, Tom Kraljevic, Tomas Nykodym, Patrick Aboyoun, Michal Kurka, and Michal Malohlava. *h2o: R Interface for the 'H2O' Scalable Machine Learning Platform*, 2020. URL https://CRAN.R-project.org/package=h2o. R package version 3.28.0.4.

Cristina Gil. *Árboles de decisión y métodos de ensemble*, 2018. URL https://rpubs.com/Cristina_Gil/arboles_ensemble.

Hadley Wickham Kirill Müller. *tibble: Simple Data Frames*, 2020. URL https://CRAN.R-project.org/package=tibble. R package version 3.0.0.

Max Kuhn. *caret: Classification and Regression Training*, 2020. URL https://CRAN.R-project.org/package=caret. R package version 6.0-86.

José Francisco López. *Coeficiente de determinación (R cuadrado)*, 2020. URL https://economipedia.com/definiciones/r-cuadrado-coeficiente-determinacion.html.

Pedro L. Luque-Calvo. *Escribir un Trabajo Fin de Estudios con R Markdown*, 2017. URL http://destio.us.es/calvo.

Lino Manjarrez. *Gráfico Redes Neuronales*, 2014. URL https://www.researchgate.net/figure/Figura-III4-Capas-de-una-Red-Neuronal-Capa-de-entrada-neuronas-que-reciben-datos-o_fig3_315762548.

Stephen Milborrow. *rpart.plot: Plot 'rpart' Models: An Enhanced Version of 'plot.rpart'*, 2019. URL https://CRAN.R-project.org/package=rpart.plot. R package version 3.0.8.

nl zhang. *CatEncoders: Encoders for Categorical Variables*, 2017. URL https://CRAN.R-project.org/package=CatEncoders. R package version 0.1.1.

Fortran original by Leo Breiman, R port by Andy Liaw Adele Cutler, and Matthew Wiener. *randomForest: Breiman and Cutler's Random Forests for Classification and Regression*, 2018. URL https://CRAN.R-project.org/package=randomForest. R package version 4.6-14.

Brian Ripley, Bill Venables, Douglas M. Bates, Kurt Hornik, Albrecht Gebhardt, and David Firth. *MASS: Support Functions and Datasets for Venables and Ripley's MASS*, 2019. URL https://CRAN.R-project.org/package=MASS. R package version 7.3-51.5.

Joaquin Amat Rodrigo. *Arboles de predicción: bagging, random forest, boosting y C5.0*, 2017. URL https://rpubs.com/Joaquin_AR/255596.

Joaquin Amat Rodrigo. *Machine Learning con H2O y R*, 2018. URL https://rpubs.com/Joaquin_AR/406480.

Joaquín Amat Rodrigo. *Introducción a la Regresión Lineal Múltiple*, 2016. URL https://rpubs.com/Joaquin_AR/226291.

Brian Ripley Terry Therneau, Beth Atkinson. *rpart: Recursive Partitioning and Regression Trees*, 2019. URL https://CRAN.R-project.org/package=rpart. R package version 4.1-15.

Michael Benesty Tianqi Chen, Tong He. *xgboost: Extreme Gradient Boosting*, 2020. URL https://CRAN.R-project.org/package=xgboost. R package version 1.0.0.2.

Luis Torgo. *DMwR: Functions and data for "Data Mining with R"*, 2013. URL https://CRAN.R-project.org/package=DMwR. R package version 0.4.1.

Lehrkamp Matthias Ulrike Groemping. *relaimpo: Relative Importance of Regressors in Linear Models*, 2018. URL https://CRAN.R-project.org/package=relaimpo. R package version 2.2-3.

Ankur Vishwakarma. *Predicting MotoGP Race Finish Times Using Linear Regression*, 2018a. URL https://medium.com/@Vishwacorp/predicting-motogp-race-finish-times-using-linear-regression-6dc3d1e1c2a4.

Ankur Vishwakarma. *Predicting MotoGP Race Finish Times Using Linear Regression*, 2018b. URL https://github.com/Vishwacorp/motogp_regression.

Sanford Weisberg. *alr3: Data to Accompany Applied Linear Regression 3rd Edition*, 2018. URL https://CRAN.R-project.org/package=alr3. R package version 2.0.8.

Hadley Wickham. *tidyverse: Easily Install and Load the 'Tidyverse'*, 2019. URL https://CRAN.R-project.org/package=tidyverse. R package version 1.3.0.

Hadley Wickham. *ggplot2: Create Elegant Data Visualisations Using the Grammar of Graphics*, 2020. URL https://CRAN.R-project.org/package=ggplot2. R package version 3.3.0.

Hadley Wickham and Lionel Henry. *tidyr: Tidy Messy Data*, 2020. URL https://CRAN.R-project.org/package=tidyr. R package version 1.0.2.

Hadley Wickham, Romain François, Lionel Henry, and Kirill Müller. *dplyr: A Grammar of Data Manipulation*, 2020. URL https://CRAN.R-project.org/package=dplyr. R package version 0.8.5.

Wikipedia. *Error cuadrático medio*, 2019a. URL https://es.wikipedia.org/wiki/Error_cuadr%C3%A1tico_medio.

Wikipedia. *Raíz del error cuadrático medio*, 2019b. URL https://es.wikipedia.org/wiki/Ra%C3%ADz_del_error_cuadr%C3%A1tico_medio.

Wikipedia. *Campeonato Mundial de Motociclismo*, 2020. URL https://es.wikipedia.org/wiki/Campeonato_Mundial_de_Motociclismo.

Yihui Xie. *knitr: A General-Purpose Package for Dynamic Report Generation in R*, 2019. URL https://CRAN.R-project.org/package=knitr. R package version 1.22.

Hao Zhu. *kableExtra: Construct Complex Table with 'kable' and Pipe Syntax*, 2019. URL https://CRAN.R-project.org/package=kableExtra. R package version 1.1.0.