FACULTAD DE ESTUDIOS ESTADÍSTICOS

MÁSTER EN MINERÍA DE DATOS E INTELIGENCIA DE NEGOCIOS

Curso 2019/2020

Trabajo de Fin de Máster

TÍTULO: Predicción de precio de Airbnb en Madrid utilizando técnicas de minería de datos

***Alumno*:** Laura Contreras Núñez

***Tutor*:** Cipriano Quirós Romero

Junio de 2020

UNIVERSIDAD COMPLUTENSE
MADRID

Índice

1. Introducción. 1

1.1 Estado del arte 3
1.2 Objetivos. 5

2. Metodología. 5

2.1 Modelización. 6
2.1.1 Modelos hedónicos. 7
2.1.2 Algoritmos de Machine Learning. 9

3. Preparación de los datos. 12

3.1 Origen de los datos. 12
3.2 Trabajo previo. 13
3.3 Análisis exploratorio. 16
3.3.1 Variables de intervalo. 16
3.3.2 Variables nominales. 18
3.3.3 Relación entre variables. 19

4. Análisis visual. 21

5. Modelización. 24

5.1 Selección de variables. 24
5.2 Regresión lineal. 26
5.3 Análisis dependencia espacial. 28
5.4 Regresiones espaciales. 30
5.5 Redes Neuronales. 33
5.6 Random Forest. 38
5.7 Gradient Boosting. 40
5.8 Xgboost. 42
5.9 Support Vector Machine. 44
5.9.1 Kernel lineal. 44
5.9.2 Kernel polinomial 45
5.9.3 Kernel radial 46
5.10 Evaluación de modelos. 47
5.11 Ensamblado. 48

6. Conclusiones. 50

7. Bibliografía. 53

8. Anexo. 57

8.1 Código creación de las variables sobre comodidades en R. 57
8.2 Código creación de la variable *accesibilidad* en Python. 58
8.3 Código creación mapas en R. 60
8.4 Código creación gráficos en Python. 62
8.5 Código selección de variables en SAS. 63
8.6 Código estimación modelos hedónicos y análisis espacial en R. 66
8.7 Código estimación algoritmos de Machine Learning en R y SAS. 69

8.7.1 Redes neuronales en SAS. 70
8.7.2 Redes neuronales en R. 75
8.7.3 Random forest. 76
8.7.4 Gradient boosting en SAS. 78
8.7.5 Gradient boosting en R. 80
8.7.6 Extreme gradient boosting. 81
8.7.7 Support Vector Machine. 83
8.7.8 Ensamblado en SAS. 85
8.7.9 Ensamblado en R. 86

Índice de figuras

Figura 1. Visualización alojamientos de Airbnb en Madrid *(Inside Airbnb, s.f.)* 2
Figura 2. Esquema metodología SEMMA *(Calviño, 2019)* 6
Figura 3. Esquema análisis espacial *(Tang et al., 2019)* 7
Figura 4. Ejemplo esquema red neuronal *(Portela, 2019).* 9
Figura 5. Kernels más frecuentes en SVM. 12
Figura 6. Descripción de variables. 13
Figura 7. Comodidades disponibles. 14
Figura 8. Histogramas comodidades disponibles. 15
Figura 9. Bucle obtención lugares más cercanos a cada Airbnb (Carillo, 2019) 16
Figura 10. Estadísticos variables de intervalo. 17
Figura 11. Histogramas variables de intervalo. 17
Figura 12. Estadísticos variables modificadas. 17
Figura 13. Datos atípicos. 18
Figura 14. Estadísticos variables de intervalo tras ser depuradas. 18
Figura 15. Estadísticos variables nominales. 18
Figura 16. Agrupaciones variables nominales. 19
Figura 17. Estadísticos variables nominales tras ser depuradas. 19
Figura 18. Correlación de las variables. 20
Figura 19. Valor de las variables. 20
Figura 20. Densidad de alojamientos Airbnb en Madrid (Elaboración propia en R). 21
Figura 21. Precio medio Airbnb por distritos en Madrid. 21
Figura 22. Cambio anual en el precio de Airbnb en Madrid (Elaboración propia en Python). 22
Figura 23. Anfitriones uniéndose a Airbnb y alojamientos recibiendo su primera reseña. 22
Figura 24. Número de anfitriones uniéndose a Airbnb cada mes. 23
Figura 25. Número de alojamientos recibiendo su primera reseña. 24
Figura 26. Diagrama de caja media MSE regresión con distintos sets de variables. 26
Figura 27. Diagrama de cajas media MSE regresión con distintos sets de variables. 26
Figura 28. Variables seleccionadas. 27
Figura 29. VIF variables modelo regresión. 28
Figura 30. Ejemplo código R para la triangulación y su distribución. 28
Figura 31. Resultado estadístico I de Moran. 29
Figura 32. Resultado test del Multiplicador de Lagrange. 29
Figura 33. RMSE y AIC según valor de k (k-NN). 30
Figura 34. Estadísticos modelo SAR. 30
Figura 35. Estadísticos modelo SEM. 31
Figura 36. Estimaciones modelo OLS, SAR y SEM. 31
Figura 37. Impactos directos, indirectos y totales en modelo SAR. 33
Figura 38. Diagrama de cajas MSE redes de 4 a 18 nodos con algoritmos Levmar y Bprop. 34
Figura 39. Diagrama de cajas MSE redes Bprop con distintas especificaciones. 35
Figura 40. Early stopping red con Bprop y Levmar. 35
Figura 41. Diagrama de cajas MSE distintas redes. 36
Figura 42. Diagrama de cajas MSE redes con distintas funciones de activación. 36
Figura 43. Ejemplo código R uso librería Caret y Parallel. 37
Figura 44. RMSE redes en función del número de nodos y el learning rate. 37

Figura 45. Diagrama de cajas MSE redes con distintas especificaciones. 38
Figura 46. RMSE Random Forest en función del número de variables a sortear. 38
Figura 47. Diagrama de cajas MSE Random Forest con 6, 7 y 8 variables a sortear. 39
Figura 48. MSE Random Forest en función del número de iteraciones. 39
Figura 49. Diagrama de cajas MSE Random Forest en función del tamaño muestral..... 39
Figura 50. Diagrama de cajas MSE Random Forest en función del tamaño de hoja. 40
Figura 51. Diagrama de cajas MSE distintos modelos Gradient Boosting. 40
Figura 52. Diagrama de cajas MSE Gradient Boosting con distintas iteraciones. 41
Figura 53. MSE Gradient Boosting con distinta profundidad y tamaño de hoja. 41
Figura 54. RMSE y MSE distintos modelos Gradient Boosting..... 41
Figura 55. Diagrama de cajas MSE Gradient Boosting con distintos tamaños de hoja. . 42
Figura 56. RMSE y MSE Gradient Boosting variando el número de iteraciones. 42
Figura 57. RMSE y MSE distintos modelos Xgboost..... 43
Figura 58. RMSE modelo Xgboost en función de gamma. 43
Figura 59. Diagrama de cajas MSE distintos Xgboost. 44
Figura 60. Diagrama de cajas MSE Xgboost en función del tamaño de hoja. 44
Figura 61. RMSE y MSE SVM lineal con distintos valores de C. 45
Figura 62. RMSE SVM polinomial en función del grado, escala y C. 45
Figura 63. Diagrama de cajas MSE distintos modelos SVM polinomial. 46
Figura 64. RMSE SVM radial en función de C y sigma. 46
Figura 65. Diagrama de cajas MSE distintos modelos SVM radial. 47
Figura 66. Comparación modelos de Machine Learning. 47
Figura 67. Diagrama de cajas ensamblados y algoritmos individuales. 49
Figura 68. Diagrama de cajas MSE 15 mejores modelos. 50
Figura 69. MSE Random Forest en función del número de variables a sortear en SAS. 57

1. Introducción.

La economía colaborativa se basa en el intercambio de bienes o servicios entre un proveedor y un consumidor a través de las plataformas digitales. Las características del proveedor y su oferta engloban diversas actividades, lo que hace difícil la formulación de una definición que capte todas ellas. Así, nos encontramos con tres distinciones principales. En primer lugar, si el proveedor ofrece sus bienes o servicios ocasionalmente, la literatura habla de intercambios «peer to peer» (P2P), como puede ser el caso de Wallapop. En segundo lugar, cuando se trata de un proveedor que actúa dentro de su área profesional, como Uber, nos encontramos en una zona gris. Por último, si es una empresa la que actúa como proveedor, esta no formaría parte de la economía colaborativa (Quirós, 2019). Airbnb, la plataforma que nos atañe en este trabajo, permite que propietarios ofrezcan sus casas o habitaciones desocupadas para el alquiler a corto plazo, y estaría, por lo tanto, bajo el sistema P2P.

El desarrollo de internet ha facilitado la capacidad de los individuos para conectarse entre unos y otros, lo que ha resultado en la migración de todas las formas de actividades a las plataformas online. Desde el 2009, debido a la recesión económica mundial, la creciente confianza en el mundo online y el desarrollo de los sistemas de pago online, la economía colaborativa ha vivido una gran expansión. La industria del alojamiento ha sido probablemente una de las más afectadas por esta expansión (Tang et al., 2019). En el caso de Airbnb, desde su creación en 2008, su tasa de crecimiento compuesta ha sido del 153%, lo que le ha llevado a situarse a día de hoy como la mayor plataforma de alquileres turísticos del mundo, ofreciendo mayor número de alojamientos que las cinco cadenas hoteleras más importantes juntas. Se estima que cada noche dos millones de personas se alojan en un Airbnb (*Airbnb Statistics [2020]: User & Market Growth Data*, 2019).

Aunque estas nuevas plataformas de alquiler de corto plazo han tenido impactos positivos en la economía local en general, y en el turismo en particular, también han tenido repercusiones negativas. Un ejemplo de ello son las investigaciones recientes que sugieren que los ingresos potenciales de estos alquileres han desplazado o exacerbado los problemas de asequibilidad de viviendas para los residentes locales. Adicionalmente, también se ha demostrado el impacto negativo en el sector hotelero (Deboosere et al.,

2019). Debido a estas repercusiones y al éxito de Airbnb, este se ha visto envuelto en grandes polémicas y muchas ciudades se han visto obligadas a regular el uso de esta plataforma con el fin de minimizar sus impactos. Bajo el lema "añadiendo datos al debate" y buscando facilitar el análisis del impacto de Airbnb, Murray Cox creó en 2016 la página web Inside Airbnb. Esta página recoge información disponible en la página de Airbnb, como el tipo de alojamiento, precio, número de camas/baños, localización, etc, a través de técnicas de «webscrapping» (*Inside Airbnb*, s. f.). Por el momento, esta página ofrece datos sobre 25 países, entre ellos España, que se coloca como el cuarto país con mayor número de alojamientos, después de Estados Unidos, Francia e Italia (*Airbnb Statistics [2020]: User & Market Growth Data*, 2019).

Esta gran presencia de alojamientos en España, llevó a que en el 2018 Airbnb generase un impacto económico directo de unos 6.000 millones de euros, cifra obtenida de la suma de los ingresos de los propietarios y del gasto estimado de los huéspedes (*España, el tercer país donde Airbnb genera más negocio*, 2019). En España, Barcelona es el caso más llamativo, ya que contando con 19,539 alojamientos turísticos en Airbnb y 1.62 millones de habitantes, dispone de un alojamiento por cada 82 residentes (Hernández, 2018). En Madrid, desde el 2015 la oferta de alojamientos se ha triplicado, pasando de unos 5,000 a unos 17,000. Además, el precio por noche ha experimentado una subida de alrededor del 50%, situándose en unos 92€/noche (Page, 2018).

Figura 1. Visualización alojamientos de Airbnb en Madrid *(Inside Airbnb, s.f.)*

Debido al incremento en la oferta y, por tanto, en la competitividad, un reto al que se enfrentan los propietarios (o anfitriones) es el de decidir el precio por noche óptimo. Un precio demasiado alto hará que los clientes prefieran hospedarse en otro alojamiento, y un precio demasiado bajo supondrá una pérdida en el beneficio potencial. De ahí surge

la necesidad de estimar modelos enfocados a la identificación de factores determinantes en el precio de los alojamientos de Airbnb.

1.1 Estado del arte

Los modelos de regresión hedónica desagregan el precio de un activo en las partes componentes del mismo para después emplear alguna forma de análisis de regresión de mínimos cuadrados ordinarios que permita examinar cómo cada pieza individual contribuye de forma exclusiva al valor general del activo (Sopranzetti, 2015). Estos modelos se usan frecuentemente en el análisis del mercado inmobiliario para separar los efectos de las características individuales del inmueble en el precio, partiendo de la suposición de que este puede ser descrito como la suma ponderada del valor de los distintos atributos de la vivienda. Las dos especificaciones hedónicas más utilizadas son la regresión lineal y la regresión logarítmica (Deboosere et al., 2019).

Con la rápida expansión de alquileres de corta estancia en los últimos años, se han empezado a examinar los factores determinantes en el precio de Airbnb. Esta empresa cuenta con su propio algoritmo hedónico, descrito por Hill (2015). Sus elementos principales son la similitud, las actividades recientes y la localización. El elemento de similitud predice el precio comparándolo con alojamientos con características similares; las actividades recientes ajustan el precio a estacionalidades; y el elemento de localización predice el impacto de la ubicación en el precio. Aunque Airbnb ponga a disposición de los propietarios herramientas como este algoritmo para fijar sus tarifas, estas son usadas como punto de referencia, dejando en manos del anfitrión la decisión del precio final.

El análisis hedónico en la literatura confirma la importancia del factor de similitud de Hill, pero los efectos no apuntan siempre en la misma dirección. Por ejemplo, en la mayoría de los casos la presencia de un mayor número de baños resulta en un precio mayor (Chattopadhyay & Mitra, 2019; Chica-Olmo et al., 2020; Deboosere et al., 2019; Moreno-Izquierdo et al., 2018; Önder et al., 2019; Perez-Sanchez et al., 2018; Tong & Gunter, 2020; Wang & Nicolau, 2017; Yuan Cai et al., 2019), pero también se han dado casos contrarios (Dudas et al., 2020; Tang et al., 2019). Lo mismo ocurre con el rating del alojamiento, donde Chica-Olmo (2020) y Zhang (2017) encontraban que una mayor puntuación afectaba negativamente al precio, en contra del resultado más habitual

(Chattopadhyay & Mitra, 2019; Deboosere et al., 2019; Önder et al., 2019; Perez-Sanchez et al., 2018; Tong & Gunter, 2020; Wang & Nicolau, 2017; Yuan Cai et al., 2019). Similarmente, solo Chattopadhyay & Mitra (2019) deducía que el incremento en el número de reseñas aumentaba el precio.

Donde sí hay completo acuerdo es en el efecto positivo del incremento en el número de habitaciones (Chica-Olmo et al., 2020; Deboosere et al., 2019; Dudas et al., 2020; Moreno-Izquierdo et al., 2018; Önder et al., 2019; Tang et al., 2019; Tong & Gunter, 2020; Wang & Nicolau, 2017; Yuan Cai et al., 2019) o de que el propietario sea un «superhost», categoría concedida a un anfitrión cuando ha completado 10 estancias, ha mantenido un ratio de respuesta del 90% o más y un ratio de cancelación del 1% o inferior, y una valoración general de 4,8. (Chica-Olmo et al., 2020; Deboosere et al., 2019; Moreno-Izquierdo et al., 2018; Tong & Gunter, 2020; Yuan Cai et al., 2019).

Respecto a elementos de localización, se coincide en el impacto negativo en el precio que supone el aumento en la distancia al centro de la ciudad (Chica-Olmo et al., 2020; Deboosere et al., 2019; Dudas et al., 2020; Önder et al., 2019; Tong & Gunter, 2020; Yuan Cai et al., 2019; Zhang et al., 2017). En este trabajo, además de estudiar el efecto en la distancia al centro de Madrid, se incluirá el de la distancia al aeropuerto de Barajas y el de la accesibilidad, medida como la distancia media a los 5 servicios más cercanos (restaurantes, supermercados, etc.). El impacto del aeropuerto no está presente en ninguno de los estudios mencionados anteriormente, y el de la accesibilidad solamente en los artículos de Önder (2019) y Perez-Sanchez (2018), que concluían que la presencia de un mayor número de servicios y la menor distancia a estos, respectivamente, influían positivamente en el precio.

A parte de modelos de regresión hedónica, también podemos encontrar en la literatura, aunque de forma muy reducida, la aplicación de modelos de regresión cuantil (Dudas et al., 2020; Perez-Sanchez et al., 2018; Wang & Nicolau, 2017), de mínimos cuadrados ponderados (Tong & Gunter, 2020; Zhang et al., 2017) y de modelos espaciales como el modelo autorregresivo espacial (SAR) y el modelo de error espacial (SEM) (Chica-Olmo et al., 2020; Tang et al., 2019). Igualmente, existe algún estudio donde se investigan modelos de machine learning como el random forest o las redes

neuronales y que demuestran la ventaja predictiva de estos algoritmos frente a los modelos hedónicos (Chattopadhyay & Mitra, 2019; Moreno-Izquierdo et al., 2018).

1.2 Objetivos.

El objetivo principal de este proyecto es la predicción del precio por noche de los alojamientos de Airbnb en la ciudad de Madrid. Esto se llevará a cabo tanto a través de un modelo hedónico y sus especificaciones espaciales SAR y SEM, como de algoritmos de Machine Learning (redes neuronales, random forest, gradient boosting, etc.). En la aplicación de estos modelos se pretende poder captar la correlación espacial entre los alojamientos, así como las posibles relaciones no lineales.

Para la realización con éxito del objetivo fundamental deberemos cumplir otros objetivos específicos de entre los que destacan:

- Depuración de los datos.
- Selección de las variables implicadas y creación de nuevas. En particular, se creará, a través de la API de Foursquare, una variable denominada accesibilidad que recoja la distancia de cada alojamiento a los cinco servicios más cercanos de tipo restaurantes, supermercado, etc.
- Investigación de la correlación espacial entre observaciones.
- Aplicación de modelos de predicción a partir de diferentes técnicas.
- Comparación y evaluación de los distintos modelos.

La incorporación de la variable de accesibilidad y el uso de modelos espaciales y de Machine Learning son los aspectos más novedosos de este trabajo frente a la literatura por no estar presentes en la mayoría de las investigaciones.

2. Metodología.

Los pasos a seguir para el desarrollo del proyecto y el cumplimiento de los objetivos se basan en la metodología SEMMA, compuesta de las siguientes fases:

- Sample (muestrear): Obtención de los datos y clasificación de las variables.
- Explore (explorar): Exploración de los datos gráfica y estadísticamente para identificar relaciones y anomalías. Fase llevada a cabo en SAS Enterprise Miner.

- Modify (modificar): Preparación de los datos a través del tratamiento de valores atípicos y faltantes, y de la transformación y creación de variables. Realizado en SAS Enterprise Miner, Python y R.

- Model (modelizar): Creación de modelos para la predicción de la variable objetivo, que en este caso es el precio por noche del alojamiento. La estimación de estos modelos se ha llevado a cabo en SAS y R se explica en el siguiente apartado.

- Asses (evaluar): Finalmente se evalúan las predicciones obtenidas y se comparan los modelos a través de estadísticos y técnicas de remuestreo como validación cruzada repetida.

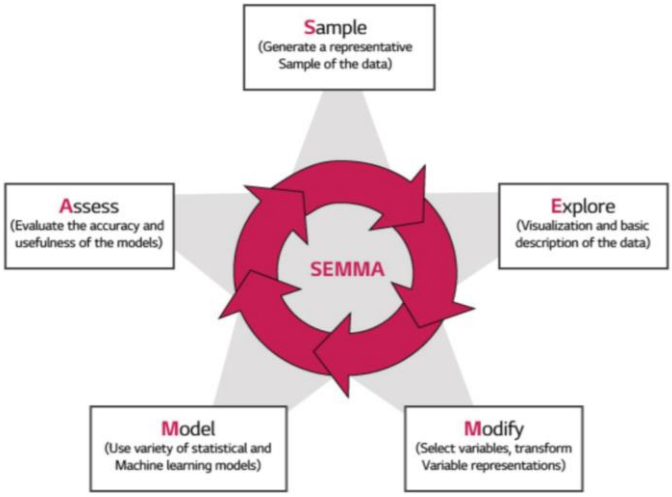


Figura 2. Esquema metodología SEMMA *(Calviño, 2019)*

2.1 Modelización.

Como se ha mencionado previamente, actualmente los modelos hedónicos son los más usados para investigar los factores determinantes en el precio de los inmuebles, los cuales asumen independencia entre las observaciones. Sin embargo, datos geográficos como los que se tratan en este trabajo pueden presentar dependencia espacial, ya que el precio de apartamentos cercanos tiene un impacto mayor que el precio de apartamentos más alejados (primera ley de la geografía de Tobler). Por ello, las estimaciones de la regresión de mínimos cuadrados ordinarios pueden ser ineficientes (Chica-Olmo et al., 2020). Para solventar este problema, en este trabajo se han implementado modelos espaciales hedónicos (SAR y SEM). Adicionalmente, estos modelos basados en regresiones no son capaces de detectar relaciones no lineales entre las variables, por lo que en este caso modelos no paramétricos son más efectivos. Por ello, también se han

incluido algoritmos de Machine Learning, específicamente: redes neuronales, random forest, gradient boosting, extreme gradient boosting, support vector machine y ensamblado. Todos los modelos mencionados se explican a continuación.

2.1.1 Modelos hedónicos.

La ecuación hedónica de precios sigue el siguiente modelo lineal:

$$y_i = \beta_0 + \sum_{k=1}^{K} \beta_k x_{ik} + \varepsilon_i \qquad [1]$$

Donde y_i es el precio por noche del alojamiento i; β_0 es el valor inicial; β_k son los coeficientes que recogen los precios implícitos de los atributos del Airbnb; x_{ik} son las variables independientes (características del alojamiento); ε_i es el error. Una vez estimado este modelo se realiza el análisis espacial siguiendo el siguiente esquema:

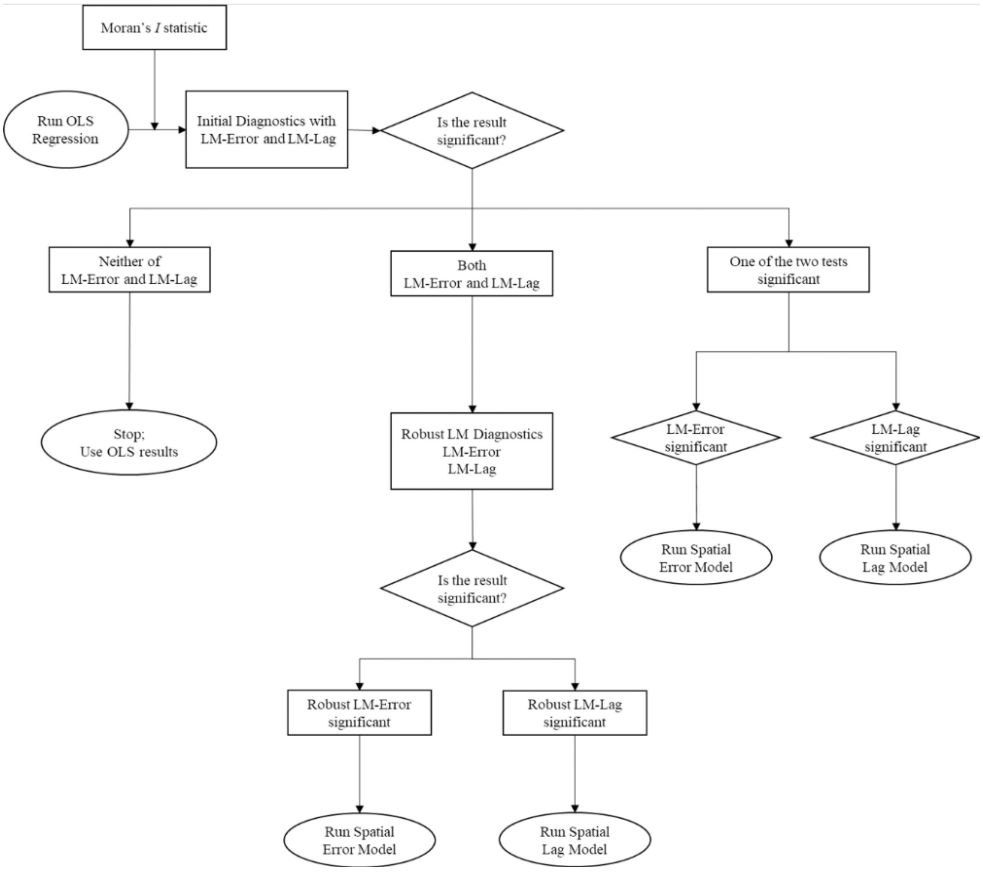


Figura 3. Esquema análisis espacial *(Tang et al., 2019)*

El primer paso tras obtener los resultados de la regresión es calcular el estadístico I de Moran, que es probablemente la técnica más usada para examinar dependencias espaciales, y toma la siguiente forma:

$$I = \left(\frac{e'We}{e'e}\right) \quad [2]$$

Donde $e = y - X\beta$ es un vector de los residuos de una regresión de mínimos cuadrados ordinarios $\beta = (X'X)^{-1}X'y$ y W es una matriz de pesos (Tang et al., 2019):

$$W = \begin{pmatrix} w_{11} & w_{12} & \cdots & w_{1n} \\ w_{21} & w_{22} & \cdots & w_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ w_{n1} & w_{n2} & \cdots & w_{nn} \end{pmatrix} \quad [3]$$

Donde n representa el número de observaciones; la entrada w_{ij} corresponde al par (i, j) de observaciones; los elementos diagonales de la matriz son cero, por convenio, mientras que los elementos no diagonales toman valores distintos de cero cuando las observaciones i y j se consideran vecinas (Borrego, 2018).

La I de Moran toma valores dentro del rango [-1, 1], siendo 1 correlación espacial perfecta positiva y 0 negativa. Si el resultado es estadísticamente significativo, indica fuerte correlación espacial en los residuos y se procede al paso 3, donde se realiza una prueba alternativa basada en el principio del multiplicador de Lagrange (LM) (Tang et al., 2019). Esta se calcula a través de los residuos de la regresión y es diferente según qué tipo de dependencia espacial queramos probar, ya que esta puede estar presente en:

- La variable dependiente, caso en el cual el modelo se denomina modelo autorregresivo espacial (SAR) y toma la siguiente forma:

$$y_i = \rho \sum_{j=1}^{n} y_j w_{ij} + \sum_{k=1}^{K} \beta_k x_{ik} + \varepsilon_i \quad [4]$$

  Donde el escalar ρ es un parámetro a estimar que determina el nivel de relación autorregresiva espacial entre y_i y $\sum_j y_j w_{ij}$.

- Los residuos (términos de error), entonces se llama modelo de error espacial (SEM) y viene dado por:

$$\varepsilon_i = \lambda \sum_{j=1}^{n} \varepsilon_j w_{ij} + u_i \quad [5]$$

  Donde λ es el parámetro autorregresivo y u_i un término aleatorio de error.

- Las variables explicativas. En este caso se denomina modelo de regresión cruzada y, al contrario que con los otros dos modelos, este no requiere procedimientos especiales para su estimación (Borrego, 2018).

Para estudiar el primer tipo de dependencia se usa el LM(error) y para el segundo, el LM(lag). Según los resultados de estos tests se elegirá un modelo SAR o SEM, a excepción del caso en que ambos sean estadísticamente significativos, que entonces se procederá a realizar un LM robusto para decantarse por el modelo más apropiado (Tang et al., 2019).

2.1.2 Algoritmos de Machine Learning.

Redes Neuronales.

Una red neuronal es un modelo de la forma $y = f(x_1, x_2, x_3, \ldots)$ donde f suele ser una función no lineal. Las redes neuronales se componen de nodos interconectados creando capas.

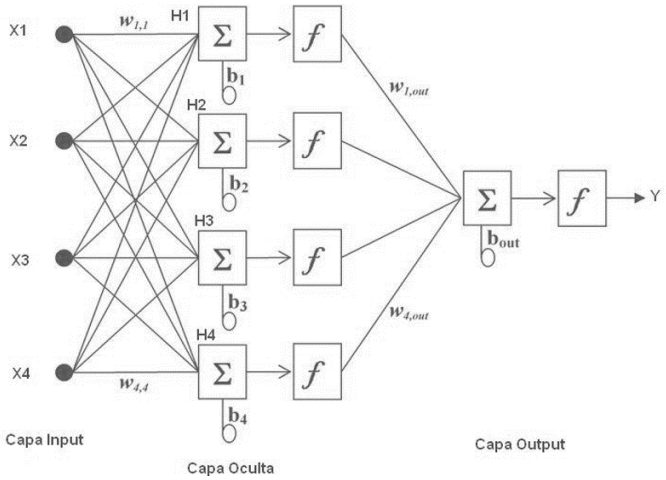


Figura 4. Ejemplo esquema red neuronal *(Portela, 2019)*.

En la capa input se encuentran los nodos de entrada (las variables independientes), que se conectan a los nodos de la capa oculta mediante la función de combinación, donde aparecen los pesos w_{ij} que hacen el papel de parámetros a estimar. Los valores de los pesos se van actualizando buscando reducir el valor de la función de error a través de funciones de optimización. Los nodos de la última capa oculta se conectan a la capa de salida donde obtenemos las predicciones de la variable dependiente.

La justificación de la red neuronal como modelo está basada en Teoremas de aproximación universal, que enuncia que cualquier función continua puede aproximarse

al nivel requerido con una red neuronal con al menos una capa oculta y un número de nodos a determinar (Portela, 2019).

Random Forest.

Random Forest es un modelo basado en árboles. Sigue el siguiente algoritmo:

Dados los datos de tamaño N,

1) Repetir m veces i), ii), iii):
   i. Seleccionar N observaciones con reemplazamiento de los datos originales
   ii. Aplicar un árbol de la siguiente manera: En cada nodo, seleccionar p variables de las k originales y de las p elegidas, escoger la mejor variable para la partición del nodo. Cuando p = k, esta técnica recibe el nombre de Bagging.
   iii. Obtener predicciones para todas las observaciones originales N
2) Promediar las m predicciones obtenidas en el apartado 1) (Portela, 2019).

Gradient Boosting.

Este algoritmo se basa en ir actualizando las predicciones en la dirección de decrecimiento de la función de error, siguiendo:

1) Hacer una primera predicción (p.ej. la media) de las observaciones.
2) Calcular los residuos para cada observación y modificar la predicción en función de la dirección del error. Para ello, se construye un árbol donde la variable dependiente es el residuo y las input son las x. Esto nos da unos residuos estimados.
3) La siguiente predicción es el valor de la primera predicción más los residuos multiplicados por una constante pequeña (“shrink”).
4) Volver al paso 2) hasta alcanzar convergencia o sobreajuste (Portela, 2019).

Xgboost.

Este modelo es una modificación del Gradient Boosting a la hora de construir cada árbol con una función de penalización basada en el número de hojas y el score-predicción en cada hoja. El algoritmo Xgboost prefija dos parámetros de regularización, gamma y lambda:

$$\Omega(f_t) = \gamma T + \frac{1}{2}\lambda \sum_{j=1}^{T} w_j^2 \qquad [6]$$

Number of leaves — L2 norm of leaf scores

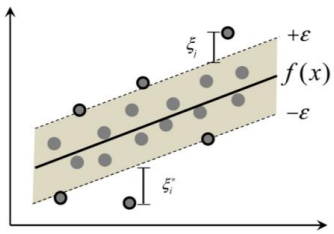
A la hora de construir cada árbol del Gradient Boosting, se tiene en cuenta esa penalización. Se utiliza un algoritmo secuencial para evaluar cómo se hace cada división de manera que la función objetivo sea la habitual, pero penalizada por la función anterior (Portela, 2019).

Support Vector Machine.

Se trata de seleccionar el hiperplano regresor que mejor se ajuste al conjunto de datos de entrenamiento. Se basa en:

1) Considerar una distancia margen ε, de modo que esperamos que todos los ejemplos se encuentren en una banda o tubo entorno a nuestro hiperplano, es decir, que disten una cantidad menor de ε del hiperplano. Pero también permitir observaciones fuera de este tubo, a través de una constante C > 0 que determina el equilibrio entre la regularidad de f y la cuantía hasta la cual toleramos desviaciones mayores que ε.

$$\begin{aligned} \text{mín} \quad & \frac{1}{2} \| w \|^2 + C \sum_{i=1}^{n} (\xi_i + \xi_i^*) \\ \text{s.a} \quad & y_i - (\langle w, x_i \rangle + b) \leq \varepsilon + \xi_i \quad i = 1, \ldots, n \\ & (\langle w, x_i \rangle + b) - y_i \leq \varepsilon + \xi_i^* \quad i = 1, \ldots, n \\ & \xi_i \geq 0, \quad \xi_i^* \geq 0 \quad i = 1, \ldots, n. \end{aligned}$$



Así pues, un valor muy grande de la constante C, en el caso límite (C → ∞) estaríamos considerando que el conjunto está perfectamente representado por nuestro hiperplano predictor (ξi → 0). Por contra, un número demasiado pequeño para C permitiría valores de ξi elevados, es decir, estaríamos admitiendo un número muy elevado de ejemplos mal representados (Martín, s. f.).

2) En los casos no lineales, se trabaja en un espacio de dimensión superior. Esto se consigue a través de una función llamada Kernel, que tiene que cumplir: K(x, y) =<φ(x), φ(y)>. Donde la función φ(x) representa una función que extrapola de la dimensión original a una superior (Portela, 2019). Los Kernels más frecuentes y que se emplean en este trabajo son:

Linear function	$K(\mathbf{x}_i \cdot \mathbf{x}_j) = \mathbf{x}_i^T \cdot \mathbf{x}_j$
Polynomial function	$K(\mathbf{x}_i \cdot \mathbf{x}_j) = (\gamma \mathbf{x}_i^T \cdot \mathbf{x}_j + r)^d$
RBF function	$K(\mathbf{x}_i \cdot \mathbf{x}_j) = \exp\left(-\frac{\|\mathbf{x}_i - \mathbf{x}_j\|^2}{2\sigma^2}\right)$ $= \exp\left(-\gamma \|\mathbf{x}_i - \mathbf{x}_j\|^2\right), \gamma = \frac{1}{2\sigma^2}$

Figura 5. Kernels más frecuentes en SVM.

Ensamblado.

Los métodos Ensemble consisten en la construcción de predicciones a partir de la combinación de varios modelos. Dos ideas básicas y que se pueden combinar son:

1) Promediar predicciones de diferentes algoritmos (regresión, red neuronal, random forest, etc)
2) Usar la predicción de un algoritmo como variable input de otro algoritmo (por ejemplo, tomar las predicciones obtenidas con una red como una nueva variable para un modelo de random forest)

Hay que tener en cuenta que cuanto menor sea la correlación entre clasificadores, más se reducirá el error con el ensamblado, pero para ello, conviene unir clasificadores que tengan sesgo suficientemente bajo y similar (Portela, 2019).

3. Preparación de los datos.

3.1 Origen de los datos.

Los datos utilizados en este proyecto han sido obtenidos de la página web Inside Airbnb, una plataforma que se dedica a hacer webscrapping de la página de Airbnb. En este caso usaremos los datos recogidos el 9 de noviembre de 2019 en la ciudad de Madrid. Nos limitaremos a los Airbnbs que tengan al menos una reseña para cerciorarnos de que el precio del alojamiento refleja el equilibrio del mercado en cierta medida. Ye et al. (2009) confirmaba la asociación entre reseñas online y reservas de habitaciones de hoteles, indicando que las reseñas sugieren transacciones reales.

Nuestra base de datos consta de 16,632 observaciones y 107 variables, de las cuales por motivos de utilidad y basándonos en otros estudios mencionados anteriormente, nos quedamos con las siguientes 32:

Variable	Descripción	
Anfitrión		
antigüedad	Nº de meses que lleva el anfitrión en Airbnb	
propiedades	Nº de apartamentos que tiene el anfitrión en Airbnb	
porcentaje_respuesta	Porcentaje de veces que responde	
tiempo_respuesta	Tiempo que tarda en contestar (una hora, varias, un día..)	
superanfitrión	T si es un "superhost", F si no	
verificado	T si su identidad ha sido verificada, F si no	
Localización		
dis_centro	Distancia al centro (Ref: Ayuntamiento de Madrid)	
dis_airp	Distancia al aeropuerto de Barajas	
accesibilidad	Distancia a los servicios más cercanos (bares, restaurantes…)	
exactitud	T si localización es exacta, F si no	
Comodidades		
AC	T si el alojamiento tiene AC, F si no	
ascensor	T si el alojamiento tiene ascensor, F si no	
checkin24h	T si el check-in es 24h, F si no	
anfitrión_recibe	T si el anfitrión te recibe, F si no	
balcón	T si el alojamiento tiene terraza, F si no	
permite_fumar	T si está permitido fumar, F si no	
apto_niños	T si el alojamiento es amigable para niños, F si no	
pequeños_electro	T si tiene pequeños electrodomésticos, F si no	
parking	T si el alojamiento tiene parking, F si no	
grandes_electro	T si el alojamiento tiene electrodomésticos, F si no	
accesible	T si es accesible, F si no	
permite_mascotas	T si se permiten mascotas, F si no	
Alojamiento		
tipo	Casa entera, habitación privada o compartida	
baños	Nº de baños	
capacidad	Capacidad personas	
política_cancelación	5 tipos de políticas de cancelación	
puntuación	Valoración general (0-100)	
noches_min	Nº mínimo de noches permitidas	
noches_max	Nº máximo de noches permitidas	
reseñas	Nº de reseñas	nominal
demanda	Nº de días del próximo mes en los que está reservado	intervalo
precio	Precio por noche	objetivo

Figura 6. Descripción de variables.

3.2 Trabajo previo.

La variable *antigüedad* venía en formato fecha y ha sido modificada en Excel para representar el número de meses desde que el anfitrión se unió a Airbnb. Las variables *dis_centro* y *dis_airp* han sido calculadas en Excel a partir de las coordenadas de cada alojamiento y las coordenadas del Ayuntamiento de Madrid y del aeropuerto de Barajas respectivamente.

Las variables binarias sobre la presencia de ciertas comodidades han sido obtenidas en R a través de una variable que contenía una lista de comodidades para cada Airbnb, por ejemplo, el primer alojamiento en la base de datos tenía:

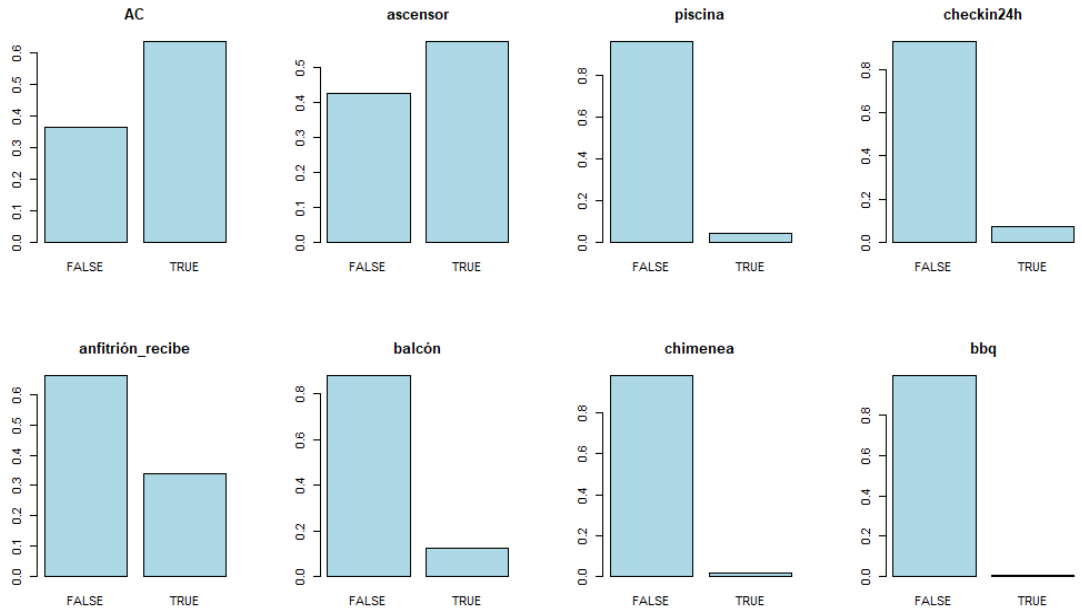
```
{Wifi, "Air conditioning", Kitchen, Elevator, Heating, "Family/kid friendly", Washer,
Essentials, Shampoo, Hangers, "Hair dryer", Iron, "Hot water", "Bed linens", "Extra
pillows and blankets", "Pocket wifi"}
```

Al crear una lista con todas las comodidades disponibles se obtiene la siguiente tabla:

wifi	air conditioning	ceiling fan
kitchen	elevator	espresso machine
heating	family/kid friendly	bbq grill
washer	essentials	outlet covers
shampoo	hangers	fixed grab bars bathtub with smart lock
hair dryer	iron	netflix
hot water	bed linens	formal dining area
extra pillows and blankets	pocket tv	ground floor
internet	free parking on premises	rain mudroom
doorman	first aid kit	heated towel rack
fire extinguisher	lock bedroom door	wine cooler
laptop friendly workspace	microwave	outdoor seating
coffee maker	refrigerator	amazon echo
dishes silverware	cooking basics	floors
oven	buzzer/wireless intercom	waterfront
24-hour check-in	luggage dropoff allowed	view
stove	host greets you	full keypad
long term stays cleaning before checkout	dog(s)	hallways
wheelchair accessible	dishwasher	entryway
smoking allowed	tub	suitable events
other pet(s)	single level home	lake access
street parking	children books toys	monitor
patio or balcony	no stairs steps to enter	fireplace
private living room	accessible-height toilet	sound system
high chair	garden backyard	handheld shower head
indoor fireplace	pool	game console
wide clearance shower	entrance	doorway bathroom
entrance for guests	dinnerware	step-free
gym	babysitter recommendations	baby bath
beachfront	ethernet connection	water kettle
well-lit path bathtub	body soap	bath towel
room-darkening shades	toilet paper	breakfast table

Figura 7. Comodidades disponibles.

De todas ellas escogimos las que inicialmente podrían suponerse que se relacionaban con el precio del alojamiento y las agrupamos en categorías para crear variables binarias que indiquen si esa comodidad está presente en cada alojamiento o no. Las variables y sus histogramas son:



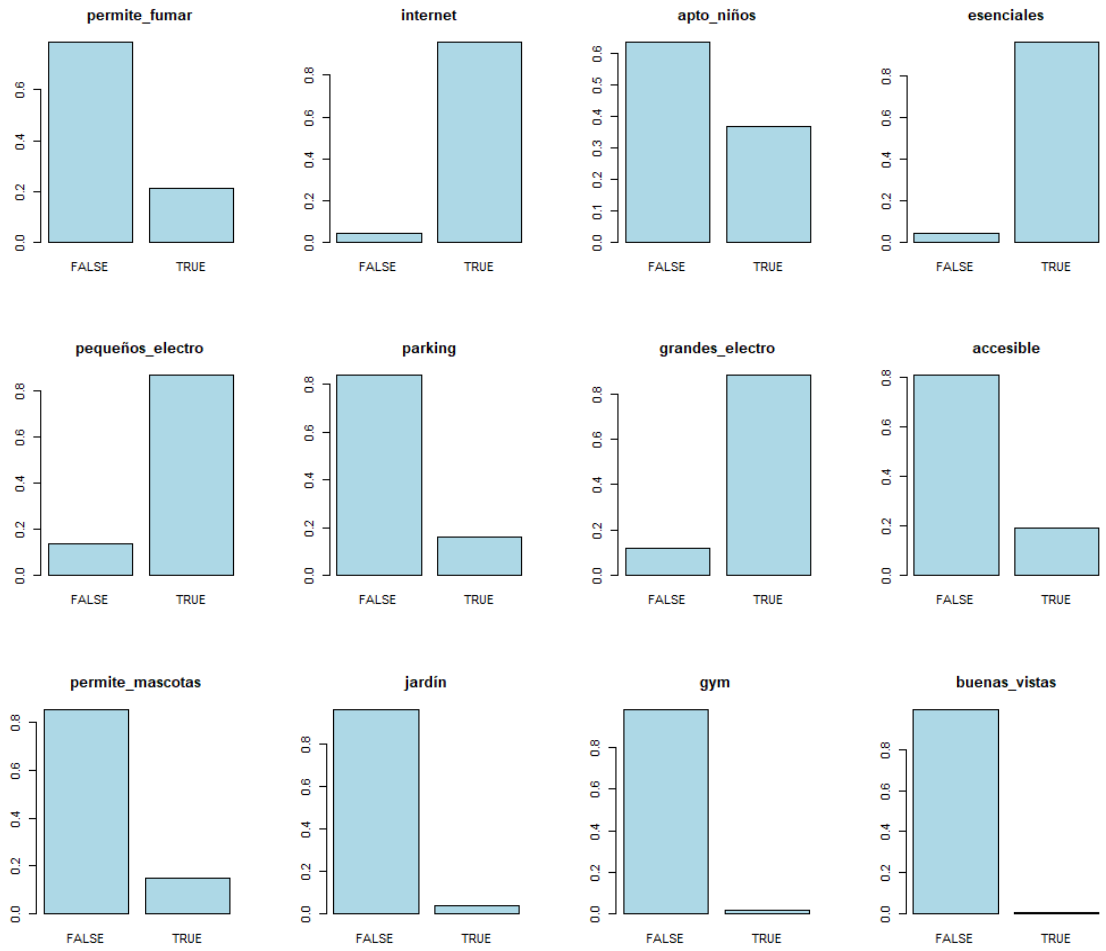


Figura 8. Histogramas comodidades disponibles.

Se descartaron aquellas variables que tuviesen menos del 5% de las observaciones en la categoría TRUE o FALSE. Estas eran: *piscina*, *chimenea*, *internet*, *bbq*, *esenciales*, *jardín*, *gym* y *buenas_vistas*. Añadimos las restantes a nuestra base de datos.

La variable *accesibilidad* ha sido creada en Python y mide la distancia media de cada alojamiento a los cinco lugares más cercanos dentro de las categorías restaurante, bar, supermercado, plaza o parque. El proceso para la creación de esta variable fue el siguiente:

1. Obtención de la información de los diez servicios más cercanos a cada alojamiento dentro del radio de un kilómetro, a través de la librería *requests*, el API de Foursquare y un bucle para automatizar el proceso.

```
# Foursquare Credentials
CLIENT_ID = '******************'
CLIENT_SECRET = '****************'
VERSION = '20200501' # Fecha en la que se realiza la búsqueda
LIMIT = 10 # Límite de lugares más cercanos a buscar por alojamiento

import requests
# function to loop through listings and get venues
def getNearbyVenues(names, latitudes, longitudes, radius):
    venues_list=[]
    for name, lat, lng in zip(names, latitudes, longitudes):
        print(name)

        # create the API request URL
        url = 'https://api.foursquare.com/v2/venues/explore?&client_id={}&client_secret={}&v={}&ll={},{}&radius={}&limit={}'
        .format(
            CLIENT_ID,
            CLIENT_SECRET,
            VERSION,
            lat,
            lng,
            radius,
            LIMIT)

        # make the GET request
        results = requests.get(url).json()['response']['groups'][0]['items']

         # return only relevant information for each nearby venue
        venues_list.append([(
            name,
            lat,
            lng,
            v['venue']['name'],
            v['venue']['location']['lat'],
            v['venue']['location']['lng'],
            v['venue']['categories'][0]['name']) for v in results])

    nearby_venues = pd.DataFrame([item for venue_list in venues_list for item in venue_list])
    nearby_venues.columns = ['ID',
                  'Listing Latitude',
                  'Listing Longitude',
                  'Venue',
                  'Venue Latitude',
                  'Venue Longitude',
                  'Venue Category']

    return(nearby_venues)
```

Figura 9. Bucle obtención lugares más cercanos a cada Airbnb (Carillo, 2019)

2. Tras obtener la información y hacer agrupaciones generales como englobar en una misma categoría a todos los restaurantes, encontramos que los cinco lugares más frecuentes son restaurantes, bares, supermercados, plazas y parques. Por ello nos limitamos a estas cinco categorías para estudiar el impacto en el precio.

3. Calculamos la distancia en kilómetros de cada alojamiento a estos lugares a través de la librería *geopy.distance*. A continuación, escogemos solo las distancias a los cinco servicios más cercanos (en caso de que un alojamiento no llegue a tener cinco de estos servicios en el límite de un kilómetro marcado en la función anterior, se penaliza marcando una distancia de 1.5 kilómetros en los faltantes). Por último, se calcula la media de las distancias y añadimos esta información a nuestra base de datos bajo el nombre *accesibilidad*.

3.3 Análisis exploratorio.

Tras el trabajo previo, cargamos los datos en SAS Miner para su depuración.

3.3.1 Variables de intervalo.

Se muestra un resumen de los principales estadísticos de las variables de intervalo.

Variable	Rol	Media	Desviación estándar	No ausente	Ausente	Mínimo	Mediana	Máximo	Asimetría
accesibilidad	INPUT	0.295797	0.167477	16632	0	0.045392	0.226214	1.5	1.451734
antiguedad	INPUT	44.20983	26.00214	16623	9	0	42	130	0.292187
demanda	INPUT	20.25728	9.334213	16632	0	0	22	30	-0.60141
dis_airp	INPUT	13.52413	2.359192	16632	0	1.840708	13.91554	26.10485	-0.95558
dis_centro	INPUT	2.477073	2.019054	16632	0	0.132375	1.664156	21.60102	1.880945
noches_max	INPUT	8100.123	865846.6	16632	0	1	1125	1.1111E8	127.1809
noches_min	INPUT	3.901575	19.9469	16632	0	1	2	1125	31.06541
porcentaje_respuesta	INPUT	1262.728	3320.227	16632	0	0	1	9999	2.25147
propiedades	INPUT	21.7521	240.2024	16632	0	0	2	9999	38.90692
puntuacion	INPUT	0.923093	0.09149	16369	263	0.2	0.95	1	-3.18108
resenas	INPUT	46.67923	68.74256	16632	0	1	18	619	2.691008
precios	TARGET	76.31854	58.76086	16632	0	8	62	450	2.482295

Figura 10. Estadísticos variables de intervalo.

Encontramos dos variables con valores ausentes y cuatro variables con valores máximos anómalos, por lo que nos fijamos en sus histogramas.

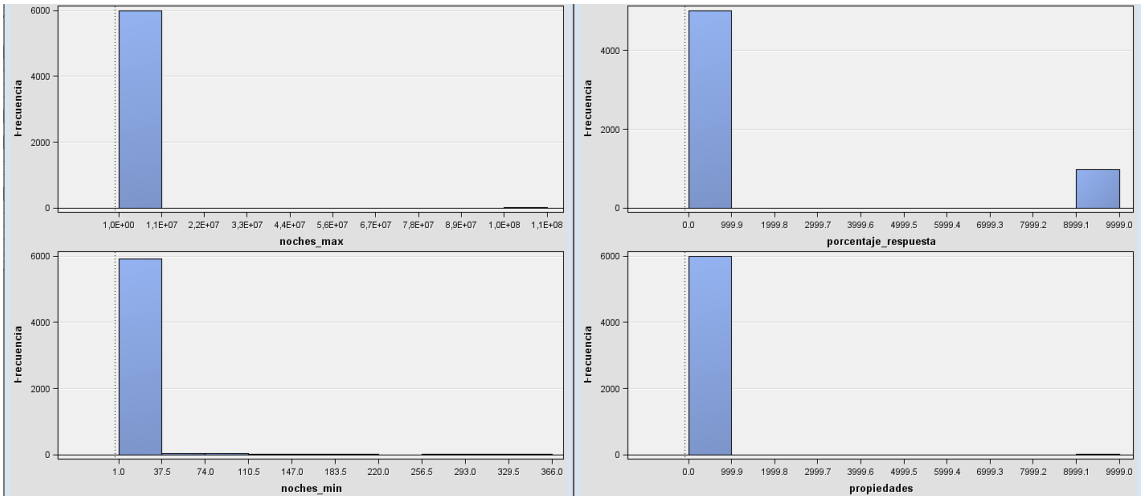


Figura 11. Histogramas variables de intervalo.

Probablemente los valores 9999 de las variables *propiedades* y *porcentaje_respuesta* corresponden a valores ausentes, por lo que fijamos su límite superior en 9998. La variable *noches_min* no la modificamos, pero estudiaremos sus datos atípicos más adelante. A la variable *noches_max* le fijamos un límite de 1500 para después estudiar sus datos atípicos.

Variable	Rol	Media	Desviación estándar	No ausente	Ausente	Mínimo	Mediana	Máximo	Asimetría
REP_noches_max	INPUT	727.3757	506.9451	16619	13	1	1125	1460	-0.55151
REP_porcentaje_respuesta	INPUT	0.949303	0.134205	14533	2099	0	1	1	-4.261
REP_propiedades	INPUT	16.35024	61.6463	16623	9	0	2	528	7.41366

Figura 12. Estadísticos variables modificadas.

Una vez modificados los límites superiores la variable *porcentaje_respuesta* está dentro de su rango [0-1], pero se han encontrado 2099 observaciones ausentes que tendremos que imputar. El número de ausentes de *propiedades* y *noches_max* es 9 y 13 respectivamente.

Pasamos a estudiar los datos atípicos de las variables *noches_min*, *noches_max* y *propiedades*. Si nos fijamos en las Figura 10 y 12 vemos que solo la primera es simétrica, por lo tanto, le aplicamos el método de desviación estándar. Las otras dos son asimétricas, por lo que usamos la desviación absoluta media.

Variable	Entrenamiento
REP_noches_max	0
REP_propiedades	3232
noches_min	87

Figura 13. Datos atípicos.

Para la variable *propiedades* se encuentran más del 5% de las observaciones, por lo que no pueden ser considerados datos atípicos. Para *noches_min* se encuentran 87 y para *noches_max* ninguno. Los outliers de *noches_min* se marcan como valores ausentes y, junto con el resto de missings de las otras variables, se imputan según la distribución. Además, para la variable *porcentaje_respuesta*, ya que tenía más del 5% de observaciones, se crea una variable indicadora que recoja si el valor ha sido imputado o no (*M_porcentaje_respuesta*). Tras imputar:

Variable	Rol	Media	Desviación estándar	No ausente	Ausente	Mínimo	Mediana	Máximo	Asimetría	Curtosis
IMP_REP_REP_noches_max	INPUT	727.6182	506.8958	16632	0	1	1125	1460	-0.55239	-1.62979
IMP_REP_REP_propiedades	INPUT	2.713865	2.607456	16632	0	0	2	11	1.678343	2.174257
IMP_REP_noches_min	INPUT	2.907047	4.933301	16632	0	1	2	63	6.3897	50.91068
IMP_REP_porcentaje_respuesta	INPUT	0.945497	0.148524	16632	0	0	1	1	-4.02267	17.79864
IMP_antiguedad	INPUT	44.21284	26.00285	16632	0	0	42	130	0.291799	-0.69619
IMP_puntuacion	INPUT	0.922947	0.092377	16632	0	0.2	0.95	1	-3.18727	16.32302
accesibilidad	INPUT	0.295797	0.167477	16632	0	0.045392	0.226214	1.5	1.451734	2.336796
demanda	INPUT	20.25728	9.334213	16632	0	0	22	30	-0.60141	-0.80627
dis_airp	INPUT	13.52413	2.359192	16632	0	1.840708	13.91554	26.10485	-0.95558	4.069383
dis_centro	INPUT	2.477073	2.019054	16632	0	0.132375	1.664156	21.60102	1.880945	3.839557
resenas	INPUT	46.67923	68.74256	16632	0	1	18	619	2.691008	9.464822
precios	TARGET	76.31854	58.76086	16632	0	8	62	450	2.482295	8.718962

Figura 14. Estadísticos variables de intervalo tras ser depuradas.

3.3.2 Variables nominales.

En la siguiente figura se muestra un resumen de los principales estadísticos de las variables nominales.

Nombre de la variable	Rol	Número de niveles	Ausente	Moda	Porcentaje moda	Moda2	Porcentaje Moda2
AC	INPUT	2	0	T	65.79	F	34.21
accesible	INPUT	2	0	F	78.10	T	21.90
anfitrion_recibe	INPUT	2	0	F	60.44	T	39.56
apto_ninos	INPUT	2	0	F	58.45	T	41.55
ascensor	INPUT	2	0	T	57.53	F	42.47
balcon	INPUT	2	0	F	85.96	T	14.04
banos	INPUT	23	4	1	71.78	2	15.54
capacidad	INPUT	16	0	2	32.68	4	25.29
checkin24h	INPUT	2	0	F	91.72	T	8.28
exactitud	INPUT	2	0	T	68.79	F	31.21
grandes_electro	INPUT	2	0	T	90.55	F	9.45
parking	INPUT	2	0	F	83.44	T	16.56
pequenos_electro	INPUT	2	0	T	89.86	F	10.14
permite_fumar	INPUT	2	0	F	79.98	T	20.02
permite_mascotas	INPUT	2	0	F	84.74	T	15.26
politica_cancelacion	INPUT	6	0	estricta_14_plazo_adicional	38.72	moderada	34.06
superanfitrion	INPUT	3	9	F	75.85	T	24.09
tiempo_respuesta	INPUT	5	2100	en una hora	65.69	en unas horas	13.41
tipo	INPUT	4	0	Alojamiento entero	64.98	Habitacion privada	32.27
verificado	INPUT	3	9	F	65.70	T	34.25

Figura 15. Estadísticos variables nominales.

Observamos algunos valores ausentes entre los que destaca los de la variable *tiempo_respuesta* por representar más del 5% de las observaciones, por lo que creamos un nuevo nivel para estos. Además, nos encontramos con variables con un número elevado de niveles que pasamos a agrupar ya que muchos de ellos tienen poca representatividad. Las agrupaciones son:

politica_cancelacion		
Nivel	Porcentaje	Nuevo
estricta_14_plazo_adicional	38.72	estricta
muy_estricta_30	0.50	
muy_estricta_60	0.22	
estricta	0.01	
moderada	34.06	moderada
flexible	26.50	flexible

tipo		
Nivel	Porcentaje	Nuevo
Habitacion privada	32.27	Habitacion
Habitacion hotel	1.69	
Habitacion compartida	1.06	
Alojamiento entero	64.98	Alojamiento entero

tiempo_respuesta		
Nivel	Porcentaje	Nuevo
en un dia	6.76	en un dia o mas
en un dia o mas	1.52	
en una hora	65.69	en una hora
en unas horas	13.41	en unas horas
	12.62	NA

capacidad		
Nivel	Porcentaje	Nuevo
1	12.23	1
2	32.68	2
3	9.21	3
4	25.29	4
5	5.28	5
6	9.71	
7	1.38	
8	2.42	
9	0.22	
10	0.79	
⋮	⋮	
16	0.19	

baños		
Nivel	Porcentaje	Nuevo
0	0.28	1
0.5	0.09	
1	71.78	
1.5	8.47	2
2	15.54	
2.5	1.08	3
3	1.62	
3.5	0.16	
7.5	0.01	
⋮	⋮	
16.5	0.01	
	0.02	NA

Figura 16. Agrupaciones variables nominales.

Como hemos mencionado previamente, se ha creado un nuevo nivel para los datos ausentes de la variable *tiempo_respuesta*. El resto de missings los imputamos según la distribución. Tras las modificaciones:

Nombre de la variable	Rol	Número de niveles	Ausente	Moda	Porcentaje moda	Moda2	Porcentaje Moda2
AC	INPUT	2	0	T	65.79	F	34.21
IMP_REP_banos	INPUT	3	0	1	72.16	2	24.01
IMP_REP_superanfitrion	INPUT	2	0	F	75.85	T	24.15
IMP_REP_verificado	INPUT	2	0	F	65.70	T	34.30
M_REP_porcentaje_respuesta	INPUT	2	0	0	87.38	1	12.62
REP_capacidad	INPUT	5	0	2	32.68	4	25.29
REP_politica_cancelacion	INPUT	3	0	estricta	39.44	moderada	34.06
REP_tiempo_respuesta	INPUT	4	0	en una hora	65.69	en unas horas	13.41
REP_tipo	INPUT	2	0	Alojamiento entero	64.98	Habitacion	35.02
accesible	INPUT	2	0	F	78.10	T	21.90
anfitrion_recibe	INPUT	2	0	F	60.44	T	39.56
apto_ninos	INPUT	2	0	F	58.45	T	41.55
ascensor	INPUT	2	0	T	57.53	F	42.47
balcon	INPUT	2	0	F	85.96	T	14.04
checkin24h	INPUT	2	0	F	91.72	T	8.28
exactitud	INPUT	2	0	T	68.79	F	31.21
grandes_electro	INPUT	2	0	T	90.55	F	9.45
parking	INPUT	2	0	F	83.44	T	16.56
pequenos_electro	INPUT	2	0	T	89.86	F	10.14
permite_fumar	INPUT	2	0	F	79.98	T	20.02
permite_mascotas	INPUT	2	0	F	84.74	T	15.26

Figura 17. Estadísticos variables nominales tras ser depuradas.

3.3.3 Relación entre variables.

Para poder identificar variables importantes, creamos una variable que tome valores aleatorios. De esta forma tendremos una referencia a la hora de fijarnos en los siguientes gráficos del valor de las variables y de correlación con la objetivo. Aquellas con menor relación que la variable aleatoria serán poco útiles.

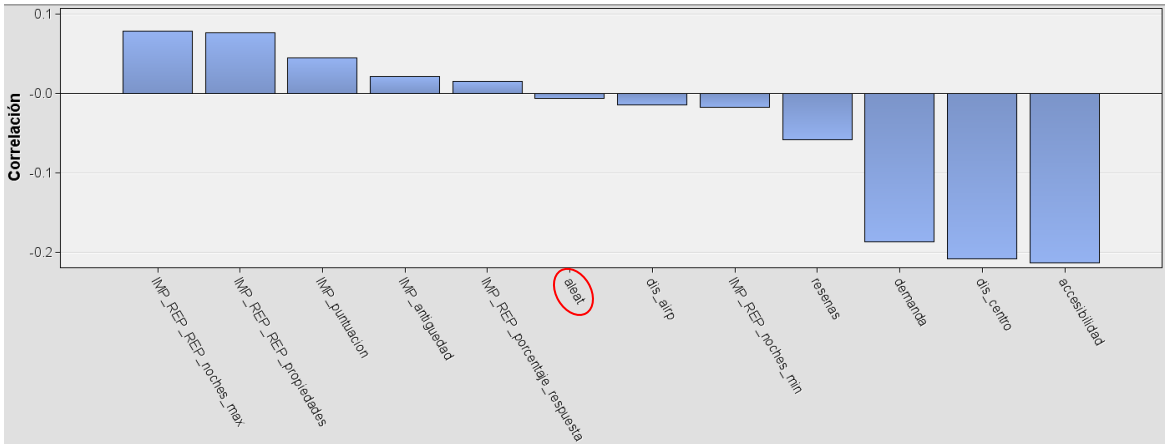


Figura 18. Correlación de las variables.

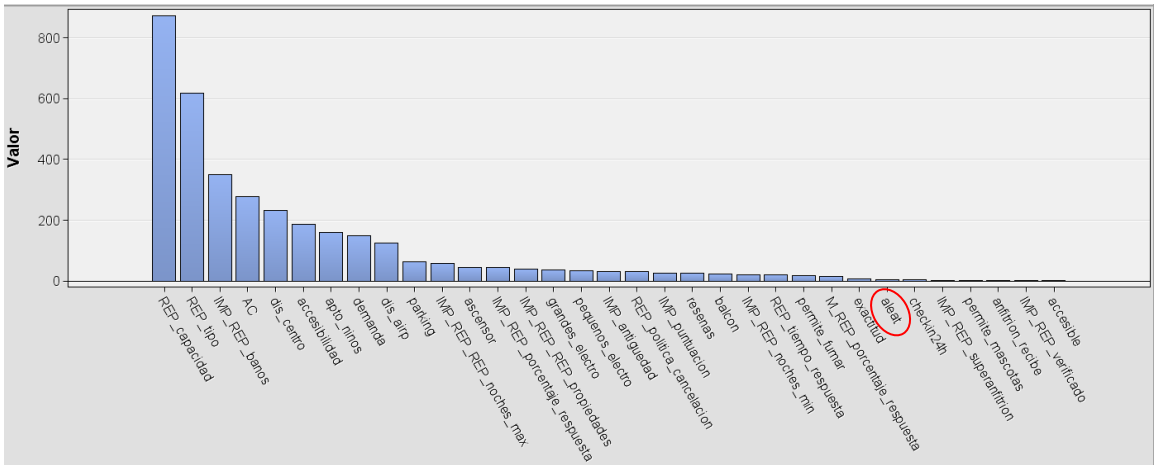


Figura 19. Valor de las variables.

En la Figura 19 observamos que las variables *capacidad* y *tipo* son las más importantes, lo que tiene sentido ya que determinan en gran medida el precio del alojamiento y son características básicas por las que se filtra en Airbnb al buscar. Sorprendentemente, vemos que la presencia o no de aire acondicionado tiene más valor que la distancia al centro o la accesibilidad. Las variables que parecen no ser útiles por tener menos valor que la aleatoria son *checkin24h*, *superanfitrion*, *permite_mascotas*, *anfitrion_recibe*, *verificado* y *accesible*.

De la Figura 18 destacamos que las variables continuas que guardan mayor correlación con la objetivo son *accesibilidad*, *dis_center* y *demanda* de forma negativa. Esto indica, como es lógico, que según aumenta la distancia a servicios como restaurantes o supermercados, y al centro de Madrid, bajan los precios, y que un alojamiento con mucha demanda en el próximo mes es porque tiene un precio menor. Aunque ninguna variable tiene menor correlación que la aleatoria, *dis_airp* y *porcentaje_respuesta* casi no guardan relación.

4. Análisis visual.

Una vez tenemos los datos depurados y previamente a la realización de las estimaciones de los distintos modelos, se propone un análisis más visual de los datos a través de gráficos y mapas con el fin de captar información que de otra forma pasaría desapercibida. Así, comenzamos graficando la distribución de los alojamientos en el municipio de Madrid. Esto nos puede servir, por ejemplo, para comprender mejor las estimaciones que obtendremos con los modelos hedónicos sobre las variables de localización.

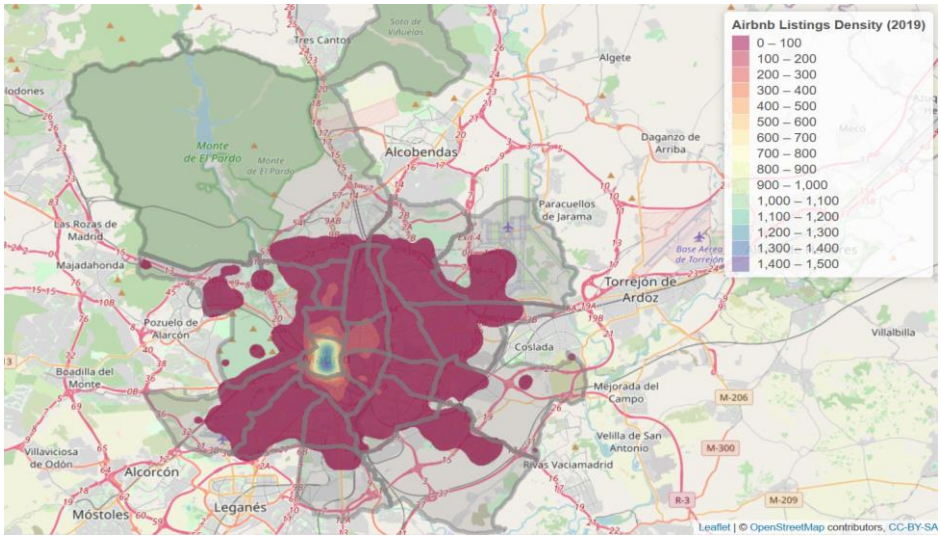


Figura 20. Densidad de alojamientos Airbnb en Madrid (Elaboración propia en R).

En el mapa se observa que la cantidad de alojamientos de Airbnb en el municipio de Madrid aumenta según nos acercamos a los distritos centrales. Además, en la siguiente figura podemos apreciar que el precio medio de los alojamientos también es mayor en estos distritos.

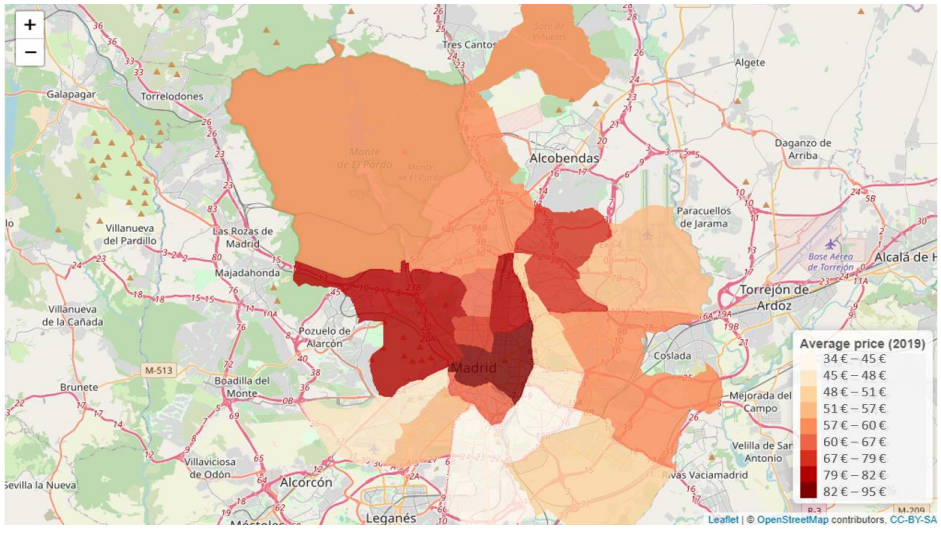


Figura 21. Precio medio Airbnb por distritos en Madrid.

En particular, los distritos del municipio de Madrid más caros son Centro, Salamanca y Retiro con un precio medio en el rango de 82-95€/noche.

Si deseamos ver la evolución del precio medio a lo largo de los años, podemos hacer una estimación usando la fecha en el que un alojamiento recibió su primera reseña como aproximación al año en el que este entró al mercado en Airbnb. Lo representamos a través del siguiente diagrama de cajas.

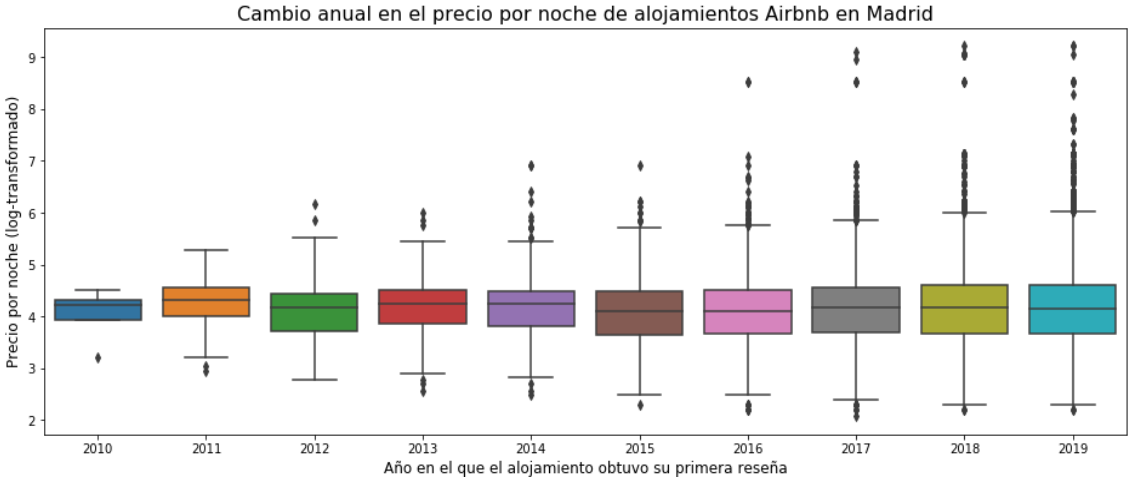


Figura 22. Cambio anual en el precio de Airbnb en Madrid (Elaboración propia en Python).

El precio medio de los alojamientos ha incrementado ligeramente a lo largo de los años, pero el extremo superior de los precios ha experimentado un gran aumento. Por ello, el cambio en el precio medio es mayor al de la mediana: en 2010 la media del precio fue 62.5€ y la mediana 67.5€, mientras que en 2019 estos valores fueron 118.6€ y 63€ respectivamente.

Siguiendo con las evoluciones temporales, podemos obtener un gráfico que recoja el número de anfitriones uniéndose a Airbnb y el número de alojamientos recibiendo su primera reseña a lo largo del tiempo.

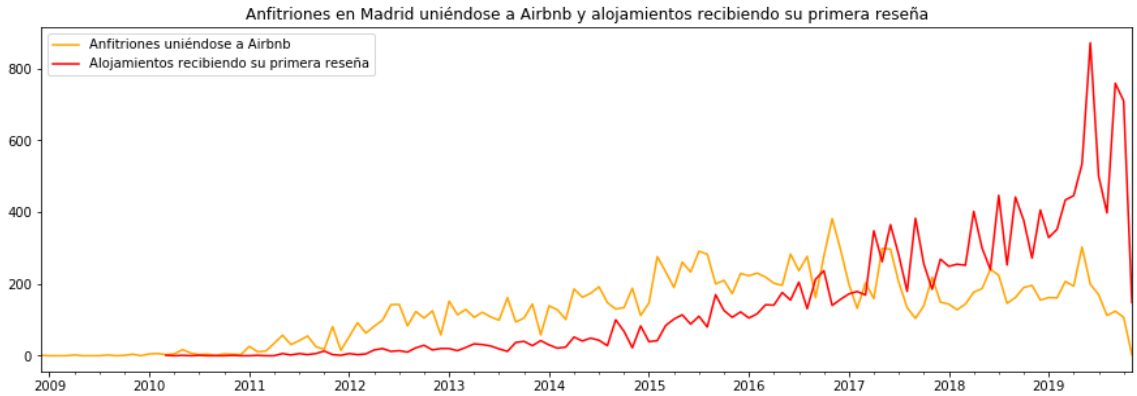


Figura 23. Anfitriones uniéndose a Airbnb y alojamientos recibiendo su primera reseña.

Aunque el primer anfitrión en unirse y que sigue estando activo, lo hizo en diciembre del 2008, la primera reseña en un alojamiento que sigue en el mercado data de marzo del 2010. Desde entonces, el número de reseñas ha ido aumentando considerablemente. Sin embargo, el número de nuevos anfitriones empezó a decaer a partir del 2017, cuando el Ayuntamiento de Madrid, en vista de los impactos de este tipo de plataformas en la industria del alquiler, contemplaba limitar el número de viviendas a uno por persona y con un tope de 60 días de alquiler, entre otras medidas (Hosteltur, 2017).

Los dos gráficos siguientes muestran la serie temporal de las dos variables del gráfico anterior, descompuesta en la tendencia general, la estacionalidad (que es importante en la industria del turismo) y los residuos. Esto se ha hecho a través de la función *seasonal_decompose* de la librería *statsmodel* de Python (Lewis, 2019). En ellos podemos apreciar una clara estacionalidad en los meses de verano que corresponde a la época del año con mayor turismo en Madrid, y que los propietarios aprovechan para generar ingresos poniendo sus viviendas en alquiler.

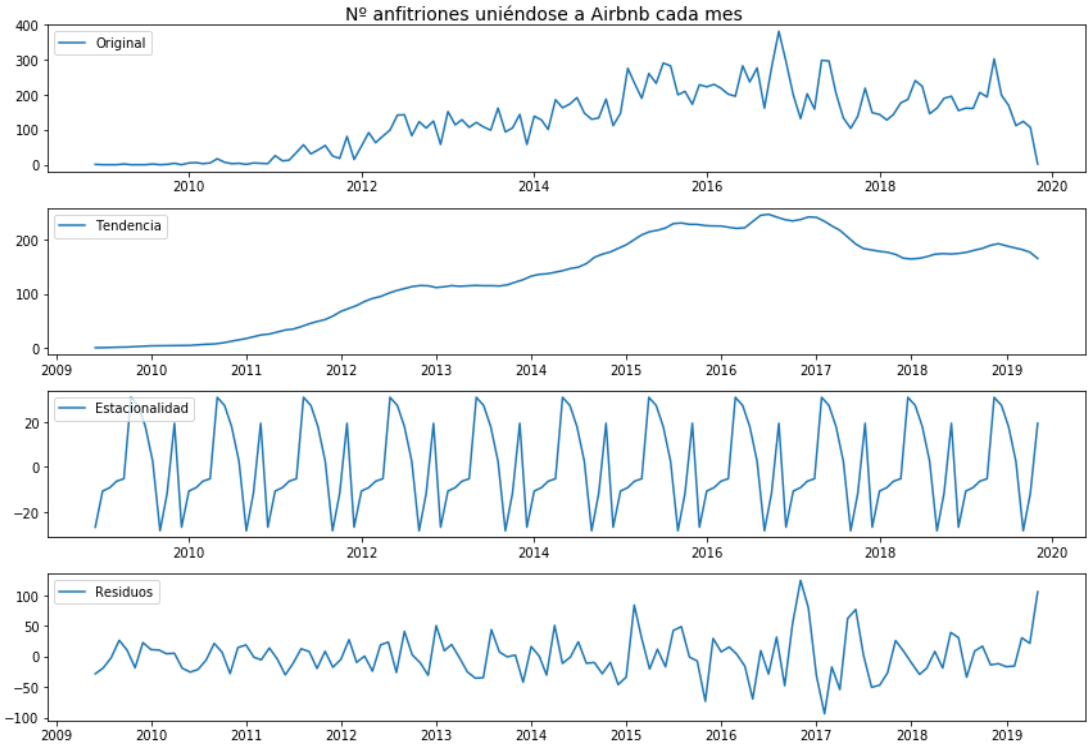


Figura 24. Número de anfitriones uniéndose a Airbnb cada mes.

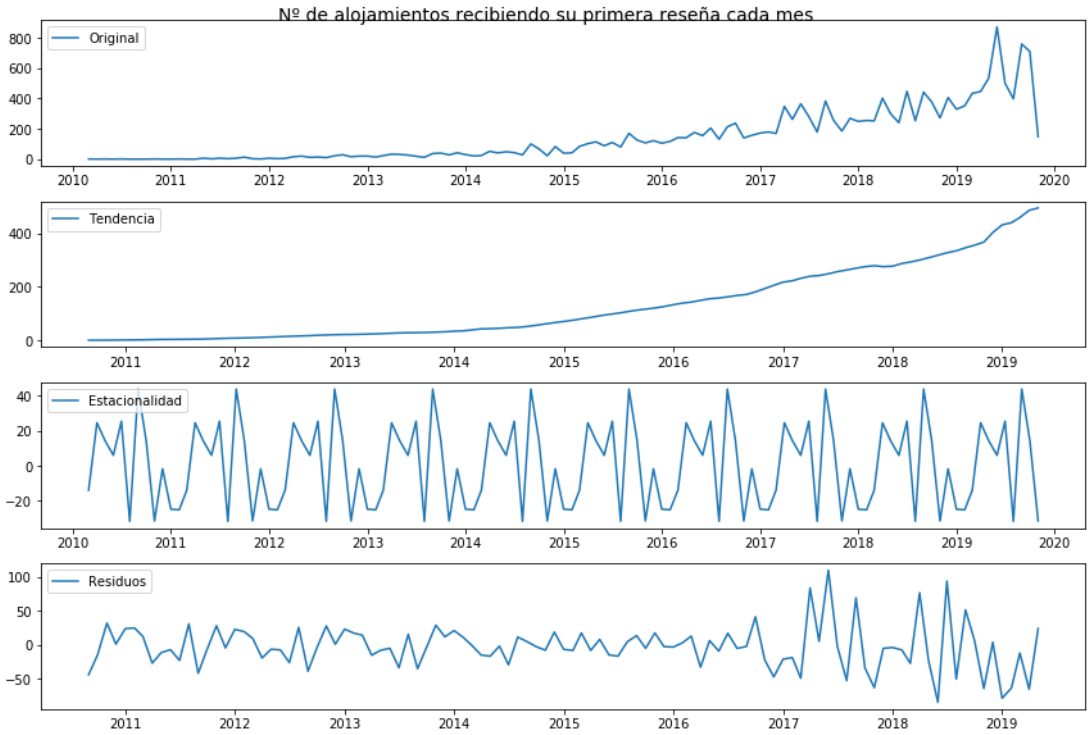


Figura 25. Número de alojamientos recibiendo su primera reseña.

5. Modelización.

En esta sección se explican las distintas estimaciones llevadas a cabo. Se estructura de la siguiente forma: En primer lugar, se realiza la selección de variables que se incorporarán a los modelos (apartado 5.1). A continuación, se estima el modelo hedónico y se estudia la dependencia espacial en sus residuos para pasar a estimar sus especificaciones espaciales SAR y SEM (apartados 5.2-5.4). Una vez hecho esto, se continúa con los algoritmos de Machine Learning donde para cada uno de ellos se investigan los parámetros a tunear (apartados 5.5-5.9). Por último, se comparan todos los modelos estimados hasta entonces y se realizan las pruebas de ensamblado para la selección del mejor modelo (apartados 5.10-5.11).

5.1 Selección de variables.

Para determinar qué variables formarán parte de nuestros modelos, elaboramos diez sets de variables obtenidos a través de distintos métodos que, en el apartado siguiente, compararemos a través de validación cruzada repetida para elegir el set que mejor se ajuste. Los diferentes métodos para la obtención son:

1. Todas las variables originales sin transformación alguna. La ventaja de no transformar las variables es la mejor interpretabilidad de las estimaciones del modelo hedónico.

2. Selección de variables. Con el nodo selección de variables de Miner rechazamos aquellas que no alcancen un valor de 0.005 de correlación con la variable objetivo.

3. Clustering de variables. A través del nodo clustering de variables de Miner agrupamos en un mismo cluster a las variables que se parezcan entre sí. Una vez creados los clusters seleccionamos las variables que pertenecerán a este basándonos en su correlación con las otras.

4. Regresión lineal backward (hacia atrás) con criterio de selección AIC, BIC y SBC en SAS. El método backward consiste en, partiendo del modelo con todas las variables, ir descartando una a una las menos influyentes hasta que todas las restantes sean significativas. Los criterios de selección de variables AIC, BIC y SBC seleccionan el modelo con el menor Criterio de Información de Akaike, menor Criterio Bayesiano de Schwarz y menor Criterio de Schwarz, respectivamente. Este último es el que más penaliza el número de parámetros.

5. Regresión lineal forward (hacia delante) con criterio de selección AIC, BIC y SBC en SAS. El método forward parte desde cero para ir introduciendo una a una las variables que mayor mejora produzcan en el modelo, hasta que no haya ninguna variable fuera que aporte información.

6. Regresión lineal stepwise (paso a paso) con criterio de selección AIC, BIC y SBC en SAS. Este método es similar al anterior, solo que se pueden eliminar las variables que han entrado de acuerdo al método backward. Por lo tanto, en cada paso se evalúan todas las posibles variables a eliminar e introducir.

Los tres últimos métodos se llevan a cabo en SAS, donde para cada una de las 9 regresiones se usa una partición de datos del 80% para entrenamiento y 20% para prueba, variando la semilla de la partición 55 veces para obtener resultados más fiables. De cada regresión se seleccionan los dos modelos más frecuentes, de tal forma que se

obtendrían 18 sets de variables. Debido a las repeticiones de modelos entre métodos, este número se ve reducido a siete. Estos siete más los tres obtenidos con los tres primeros métodos conforman nuestros diez sets de variables a probar.

5.2 Regresión lineal.

Para cada uno de los diez sets de variables obtenidos previamente, realizamos una regresión lineal con validación cruzada repetida de cuatro grupos y 15 semillas, y comparamos los resultados a través del error cuadrático medio (MSE).

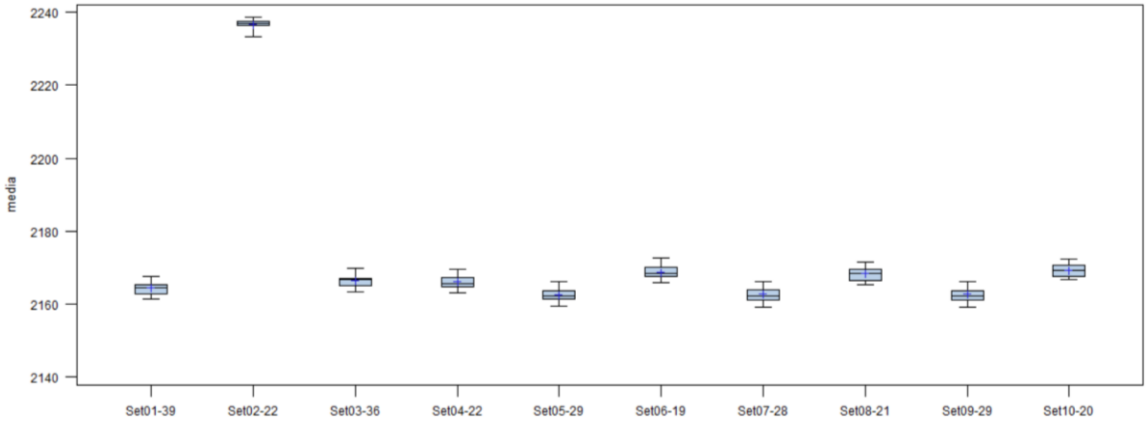


Figura 26. Diagrama de caja media MSE regresión con distintos sets de variables.

Rehacemos el gráfico descartando el segundo set de variables, que corresponde al obtenido en Miner a través del nodo de selección de variables.

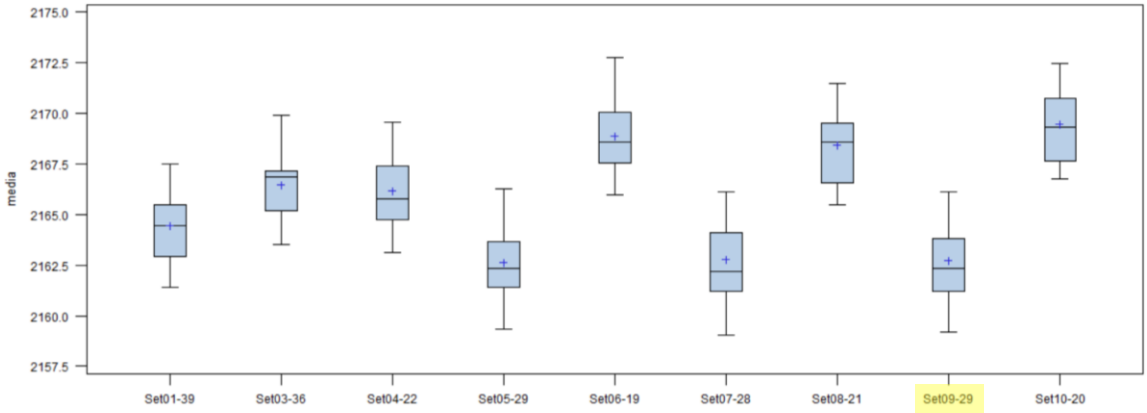


Figura 27. Diagrama de cajas media MSE regresión con distintos sets de variables.

El número de parámetros de cada modelo se indica con un guion en el nombre. La varianza no varía mucho, por lo que nos centramos en los modelos con mejor sesgo, que son el quinto, séptimo y noveno. Sus diferencias tanto en sesgo, varianza y número de parámetros son mínimas, por lo que escogemos el noveno por contener variables que pueden resultar más interesantes para las estimaciones. Este modelo se obtenía al aplicar cualquiera de los métodos forward, backward o stepwise con el criterio de selección

BIC. En la siguiente figura se muestran en verde las variables seleccionadas y en rojo las excluidas.

Variables
Anfitrión
antigüedad
propiedades
porcentaje_respuesta
tiempo_respuesta
superanfitrión
verificado
Localización
dis_centro
dis_airp
accesibilidad
exactitud
Comodidades
AC
ascensor
checkin24h
anfitrión_recibe
balcón
permite_fumar
apto_niños
pequeños_electro
parking
grandes_electro
accesible
permite_mascotas
Alojamiento
tipo
baños
capacidad
política_cancelación
puntuación
noches_min
noches_max
reseñas
demanda

Figura 28. Variables seleccionadas.

Una vez seleccionadas las variables procedemos a estimar el modelo hedónico en R con la función *lm* y una partición entrenamiento-prueba de 80%-20%. El modelo obtenido tiene un R^2 de 0.3847 y la raíz cuadrada del error cuadrático medio (RMSE) es 45.35. Las estimaciones de los parámetros se presentan más adelante en la Figura 36, ya que ahora lo que nos interesa analizar es la posible dependencia espacial.

Antes de pasar al estudio espacial, comprobamos que no haya problemas de multicolinealidad en las predicciones mediante el factor de inflación de la varianza (VIF), que mide la cantidad de la varianza de un coeficiente que ha sido aumentada como consecuencia de colinealidad en el modelo. Como norma general valores de VIF por encima de diez indican la existencia de un problema de multicolinealidad (Chattopadhyay & Mitra, 2019). En la siguiente tabla se observa que los valores están muy por debajo de diez, por lo que no tenemos este problema.

Variable	VIF
dis_centro	2.667
dis_airp	1.147
demanda	1.156
reseñas	1.315
accesibilidad	2.339
AC	1.306
ascensor	1.067
anfitrion_recibe	1.161
balcon	1.092
apto_niños	1.414
parking	1.581
grandes_electro	1.143
accesible	1.145
permite_mascotas	1.027
capacidad	2.608
politica_cancelacion	1.152
tiempo_respuesta	1.358
tipo	2.209
baños	1.212
superanfitrion	1.188
antigüedad	1.210
puntuacion	1.127

Figura 29. VIF variables modelo regresión.

5.3 Análisis dependencia espacial.

Tras haber estimado la regresión lineal, el siguiente paso en el análisis espacial siguiendo el esquema de la Figura 3, es calcular el estadístico I de Moran. En su fórmula [2] observamos que necesitamos una matriz de pesos W [3]. Para ello, primero debemos determinar qué observaciones se consideran vecinas entre ellas. Esto lo hacemos con la función de R *tri2nb* que determina los vecinos por triangulación, habiendo convertido previamente los datos en objetos espaciales a través de sus coordenadas. Los códigos de R de este apartado están basados en el tutorial de Sarmiento-Barbieri (2016).

```
library(spdep)
library(sf)
library(deldir) # install from CRAN if not got already
library(maptools)
library(geojsonio)

# Convert latitude and longitude variables to coordinates
dataSP_train <- st_as_sf(dataSP_train, coords= c("longitude", "latitude"))

# Convert sf object to sp
dataSP_train <- as(dataSP_train, "Spatial")

# Eliminate duplicates
dataSP_train <- dataSP_train[!duplicated(data.frame(coordinates(dataSP_train))),]

# Construct a neighbourhood list
coords <- coordinates(dataSP_train)
nbl = tri2nb(coords)
summary(nbl)

# Plot link distribution
Distritos <- geojson_read('distritos.geojson', what = "sp")
madrid <- st_as_sf(Distritos)
plot(st_geometry(madrid), border="dark blue", reset=FALSE)
plot(nbl, coords, add=TRUE)
```

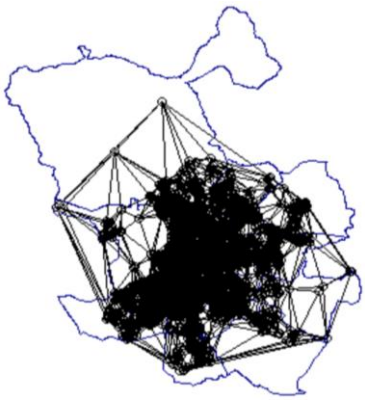


Figura 30. Ejemplo código R para la triangulación y su distribución.

En la figura anterior observamos la distribución obtenida. El número promedio de vecinos para cada observación es seis. El número mínimo de vecinos para una observación es tres y ocurre 148 veces. El número máximo de vecinos para una observación es 15 y se da en un caso.

A continuación, obtenemos la matriz de pesos W con la función *nb2listw*, y con esto y los residuos de la regresión lineal estimada anteriormente, calculamos el estadístico I de Moran a través de la función *morantest*. La hipótesis nula es que los datos son espacialmente independientes, ya que el p-valor << 0.05 la rechazamos.

```
        Global Moran I for regression residuals

data:
model: lm(formula = precio ~ ., data = dataSP_train@data)
weights: W

Moran I statistic standard deviate = 9.785, p-value < 2.2e-16
alternative hypothesis: two.sided
sample estimates:
Observed Moran I      Expectation         Variance
    4.914173e-02    -3.643159e-04     2.559745e-05
```

Figura 31. Resultado estadístico I de Moran.

Como ya se ha mencionado anteriormente, el resultado de este test nos indica que hay dependencia espacial, pero no qué tipo. Recordamos que la correlación espacial puede estar presente en la variable dependiente (modelo SAR) o en los residuos (modelo SEM). Para obtener más información sobre el tipo de dependencia presente en nuestros datos, usamos el test del Multiplicador de Lagrange con la función *LMtests*.

```
        Lagrange multiplier diagnostics for spatial dependence

data:
model: lm(formula = precio ~ ., data = dataSP_train@data)
weights: W

LMerr = 94.158, df = 1, p-value < 2.2e-16


        Lagrange multiplier diagnostics for spatial dependence

data:
model: lm(formula = precio ~ ., data = dataSP_train@data)
weights: W

LMlag = 105.4, df = 1, p-value < 2.2e-16


        Lagrange multiplier diagnostics for spatial dependence

data:
model: lm(formula = precio ~ ., data = dataSP_train@data)
weights: W

RLMerr = 2.6388, df = 1, p-value = 0.1043


        Lagrange multiplier diagnostics for spatial dependence

data:
model: lm(formula = precio ~ ., data = dataSP_train@data)
weights: W

RLMlag = 13.878, df = 1, p-value = 0.0001951
```

Figura 32. Resultado test del Multiplicador de Lagrange.

LMerr y LMlag miden la dependencia espacial en los residuos y en la variable dependiente, respectivamente. Dado que ambos resultados no se diferencian significativamente de 0, nos fijamos en sus equivalentes robustos, RLMerr y RLMlag. Estos sugieren que el modelo SAR es la alternativa más probable, sin embargo, estimaremos ambos modelos para estudiar sus diferencias.

5.4 Regresiones espaciales.

Puesto que los tests que han sido llevado a cabos indican que hay una fuerte correlación espacial y que el modelo que mejor resuelve este problema es el modelo autorregresivo espacial (SAR), estimaremos este modelo primero. Como podemos ver en su fórmula [4], se requiere una matriz de pesos W. Buscando optimizar el modelo, en vez de usar la anterior matriz donde se calcularon los vecinos por triangulación, usaremos los k vecinos más próximos, es decir:

$$W: \begin{cases} w_{ij} = 1 \text{ si el alojamiento } j \text{ es uno de los } k \text{ vecinos más próximos al alojamiento } i \\ w_{ij} = 0 \text{ si no} \end{cases}$$

Para determinar el valor de k se ha estimado el modelo SAR usando diferentes valores de los alojamientos k más próximos con un bucle de 5 a 60 vecinos de 5 en 5.

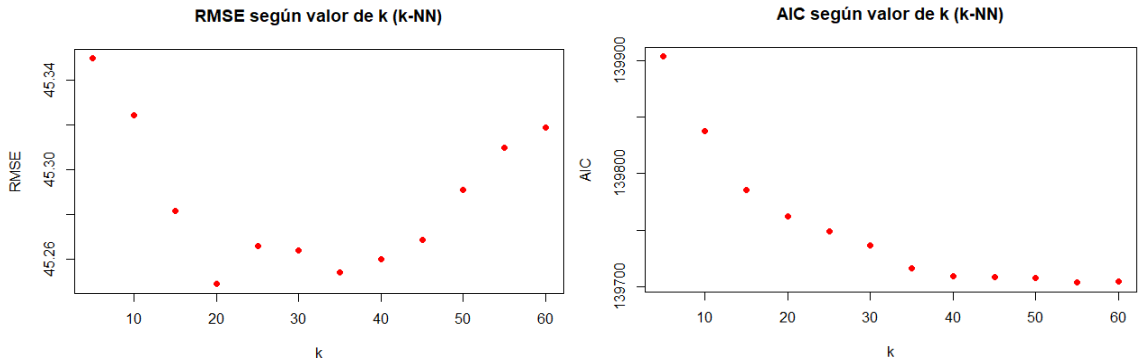


Figura 33. RMSE y AIC según valor de k (k-NN).

De todos ellos se seleccionaron 35 vecinos por obtener un valor bajo tanto en RMSE como en AIC, como podemos ver en la figura anterior. Una vez calculada la matriz, estimamos el modelo con la función *lagsarlm*.

```
Rho: 0.39482, LR test value: 341.86, p-value: < 2.22e-16
Asymptotic standard error: 0.020859
    z-value: 18.928, p-value: < 2.22e-16
Wald statistic: 358.25, p-value: < 2.22e-16

Log likelihood: -69826.29 for lag model
ML residual variance (sigma squared): 2113.6, (sigma: 45.974)
Number of observations: 13302
Number of parameters estimated: 32
AIC: 139720, (AIC for lm: 140060)
LM test for residual autocorrelation
test value: 2.6972, p-value: 0.10052
```

Figura 34. Estadísticos modelo SAR.

Los parámetros de los coeficientes se muestran en la Figura 36, pero en la anterior observamos que, como esperábamos, Rho (ρ), que indica autocorrelación espacial por ser el parámetro autorregresivo [4], es significativo. Vemos también que el test para la

autocorrelación en los residuos no es significativo, lo que sugiere que no podemos rechazar la hipótesis nula de que los residuos estén distribuidos aleatoriamente.

A continuación, calculamos el segundo tipo de modelo espacial, el modelo de error espacial (SEM). Para ello, usamos la matriz de pesos W calculada previamente y la función errorsarlm.

```
Lambda: 0.476, LR test value: 328.22, p-value: < 2.22e-16
Asymptotic standard error: 0.024807
    z-value: 19.188, p-value: < 2.22e-16
Wald statistic: 368.2, p-value: < 2.22e-16

Log likelihood: -69833.1 for error model
ML residual variance (sigma squared): 2110.5, (sigma: 45.941)
Number of observations: 13302
Number of parameters estimated: 32
AIC: 139730, (AIC for lm: 140060)
```

Figura 35. Estadísticos modelo SEM.

En este caso el parámetro autorregresivo espacial es Lambda (λ) [5] y vuelve a ser muy significativo.

	OLS Estimate	SAR Estimate	SEM Estimate
(Intercept)	70.079025 ***	54.998535 ***	71.7039779 ***
Anfitrión			
antigüedad	0.01465	0.0118699	0.0123513
tiempo_respuesta En unas horas	3.368478 **	3.4108774 **	1.2289451 **
tiempo_respuesta NA	11.282953 ***	11.6485436 ***	11.7416215 ***
tiempo_respuesta Un día o más	6.992745 ***	6.940149 ***	7.0744052 ***
superanfitrión True	2.829475 **	2.5224403 *	2.4786424 *
Localización			
dis_centro	-2.127185 ***	-0.6555929 *	-2.8519343 ***
dis_airp	-1.007593 ***	-0.4694105 *	-1.1796849 ***
accesibilidad	-13.170116 ***	0.3852597	-3.0243061
Comodidades			
AC True	7.73335 ***	6.9155317 ***	7.0015829 ***
ascensor True	5.805626 ***	4.4212428 ***	4.4909953 ***
anfitrión_recibe True	-1.445069	-1.4909554 .	-1.4745386 .
balcón True	3.409912 **	3.7270833 **	3.8322986 **
apto_niños True	5.493883 ***	4.5537332 ***	4.3471793 ***
parking True	-2.992371 *	-1.1091144 .	-2.280194
grandes_electro True	-9.183013 ***	-8.1355051 ***	-7.7468491 ***
accesible True	-2.117431 *	-1.9515759 .	-1.7390668 .
permite_mascotas True	-1.802299	-1.1380812	-1.1364257
Alojamiento			
demanda	-1.014054 ***	-0.9895895 ***	-1.0042547 ***
reseñas	-0.095752 ***	-0.0938523 ***	-0.0951635 ***
capacidad 2	11.356929 ***	10.6039749 ***	10.378057 ***
capacidad 3	16.373247 ***	15.5449472 ***	15.1804539 ***
capacidad 4	25.012957 ***	23.8236095 ***	23.5975766 ***
capacidad 5	46.583343 ***	45.6465735 ***	45.507503 ***
política_cancelación Flexible	1.814678 .	1.7379783	1.5870423
política_cancelación Moderada	-2.258638 *	-1.9040039 *	-2.0280706 *
tipo Habitación	-28.283493 ***	-27.9803551 ***	-28.545759 ***
baños 2	18.565123 ***	17.7459625 ***	17.6170074 ***
baños 3	52.58814 ***	50.3861545 ***	51.620759 ***
puntuación	34.595462 ***	33.3388507 ***	33.1625891 ***
Signif. codes: '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1			
Estadísticos			
R^2	0.38467	0.38645	0.38281
AIC	140056	139720	139730
RMSE	45.3532	45.2877	45.4220

Figura 36. Estimaciones modelo OLS, SAR y SEM.

Si comparamos los tres modelos estimados, no encontramos grandes diferencias entre ellos. Todas las variables son significativas al 95% excepto la antigüedad del anfitrión, la flexibilidad en la política de cancelación, la permisión de mascotas y el recibimiento del anfitrión. Donde se observa un mayor cambio es en los coeficientes de las variables que se refieren a la localización. A pesar de ello, estas mantienen su signo, a excepción de la accesibilidad, que además pasa de ser significativa en la regresión OLS, a no serlo en sus especificaciones espaciales. Lo mismo ocurre con que haya parking o que el alojamiento sea accesible para personas de movilidad reducida.

Respecto a los estadísticos, aunque los obtenidos por el modelo SAR sean ligeramente mejores, no difieren mucho de los otros dos. Sin embargo, al aplicar una prueba de razón de verosimilitud para determinar si la especificación SAR es significativamente mejor que las otras dos, se concluye (p-valor < 2.2e-16) que la bondad de ajuste de este modelo es superior a la del resto.

En cuanto a la interpretación de los coeficientes, la del modelo SEM es igual a la de una regresión lineal normal, pero no es el caso de las especificaciones SAR. En estas, cada observación recibirá un impacto directo de las variables predictivas, pero también uno indirecto de la información de observaciones vecinas. Esto implica que un cambio en el predictor de la región i puede afectar el resultado de la región j. Por consiguiente, hay tres cantidades de interés: el impacto directo, que es el equivalente a la interpretación tradicional; el impacto indirecto, que es el impacto del cambio en una variable explicativa sobre el precio de otros alojamientos próximos; y el impacto total, que es la suma de ambos (Chica-Olmo et al., 2020).

En la Figura 37 podemos ver estos tres impactos. De los efectos directos se puede inferir, por ejemplo, que al aumentar en un kilómetro la distancia al centro o al aeropuerto, el precio por noche disminuye ligeramente en menos de 1€. Un resultado parecido se obtiene en los efectos indirectos, lo que tiene sentido ya que sus alojamientos vecinos estarán en una situación geográfica parecida y, por lo tanto, ajustarán su precio a esta. En cuanto a las comodidades, que un alojamiento disponga de aire acondicionado aumenta el precio en unos 7€/noche, y esto, como consecuencia, influirá indirectamente en el precio por noche de alojamientos próximos en unos 4€. Algo similar ocurre con la presencia de ascensor o balcón.

Variable	Directo	Indirecto	Total
antigüedad*	0.011932	0.007682	0.019614
tiempo_respuesta En unas horas	3.428612	2.207518	5.636130
tiempo_respuesta NA	11.709111	7.538930	19.248041
tiempo_respuesta Un día o más	6.976235	4.491660	11.467895
superanfitrión True	2.535556	1.632522	4.168078
dis_centro	-0.659002	-0.424299	-1.083301
dis_airp	-0.471851	-0.303802	-0.775653
accesibilidad*	0.387263	0.249340	0.636603
AC True	6.951489	4.475728	11.427217
ascensor True	4.444231	2.861426	7.305657
anfitrión_recibe True*	-1.498708	-0.964945	-2.463653
balcón True	3.746462	2.412166	6.158628
apto_niños True	4.577411	2.947173	7.524584
parking True*	-1.114881	-0.717818	-1.832699
grandes_electro True	-8.177806	-5.265294	-13.443100
accesible True*	-1.961723	-1.263059	-3.224782
permite_mascotas True*	-1.143999	-0.736565	-1.880564
demanda	-0.994735	-0.640462	-1.635197
reseñas	-0.094340	-0.060741	-0.155081
capacidad 2	10.659111	6.862886	17.521997
capacidad 3	15.625774	10.060680	25.686454
capacidad 4	23.947481	15.418625	39.366107
capacidad 5	45.883915	29.542434	75.426349
política_cancelación Flexible*	1.747015	1.124818	2.871834
política_cancelación Moderada	-1.913904	-1.232270	-3.146174
tipo Habitación	-28.125840	-18.108868	-46.234709
baños 2	17.838234	11.485176	29.323409
baños 3	50.648140	32.609888	83.258027
puntuación	33.512198	21.576883	55.089081
*No significativa al 95%			

Figura 37. Impactos directos, indirectos y totales en modelo SAR.

Respecto a la capacidad del alojamiento, observamos que Airbnbs donde se pueden hospedar dos personas son unos 11€/noche más caros que para una persona, y que este número aumenta a 46€ cuando se trata de hospedar a cinco personas o más. Asimismo, las habitaciones son unos 28€ por noche más baratos que los alojamientos enteros.

Cabe destacar que, en todos los casos, el efecto indirecto es superior a la mitad del efecto directo, por lo que no son insignificantes e indican que el precio de un Airbnb se ve afectado por las características particulares de los alojamientos vecinos. Se podría decir que el anfitrión se esfuerza en mejorar su propiedad para poder competir en su vecindad. Por consecuente, es probable que alojamientos próximos entre sí tengan características similares (Chica-Olmo et al., 2020).

5.5 Redes Neuronales.

Tras haber estimado y comentado los modelos hedónicos, comenzamos con la investigación de los algoritmos de Machine Learning, los cuales tendremos que tunear debido a que se tratan de modelos paramétricos. Para estos, usaremos el mismo set de variables empleado en las regresiones anteriores.

Las redes neuronales tienen varios factores a tener en cuenta como la función de activación, el número de nodos o el algoritmo, que serán estudiados a través de prueba-error y validación cruzada repetida tanto en el programa SAS como R.

En SAS, los primeros factores a investigar son el número de nodos y el algoritmo usado para la optimización. Para el número de nodos, hay que tener en cuenta que en una red el número de parámetros es igual a $h(k + 1) + h + 1$, donde h es el número de nodos ocultos y k el número de nodos input, y que se recomiendan como mínimo 30 observaciones por parámetro para que no haya sobreajuste. Ya que disponemos de 16,632 observaciones y 29 parámetros, probaremos de 4 (133 observaciones por parámetro) a 18 nodos (30 observaciones por parámetro). Respecto al algoritmo de optimización nos centraremos en dos: Levenberg-Marquardt (Levmar) y Backpropagation (Bprop). Para cada uno de estos algoritmos probaremos modelos con 4 a 18 nodos, de dos en dos, para un total de 16 especificaciones de red distintas. Estas 16 redes se comparan a través de validación cruzada repetida y se muestran en el siguiente gráfico.

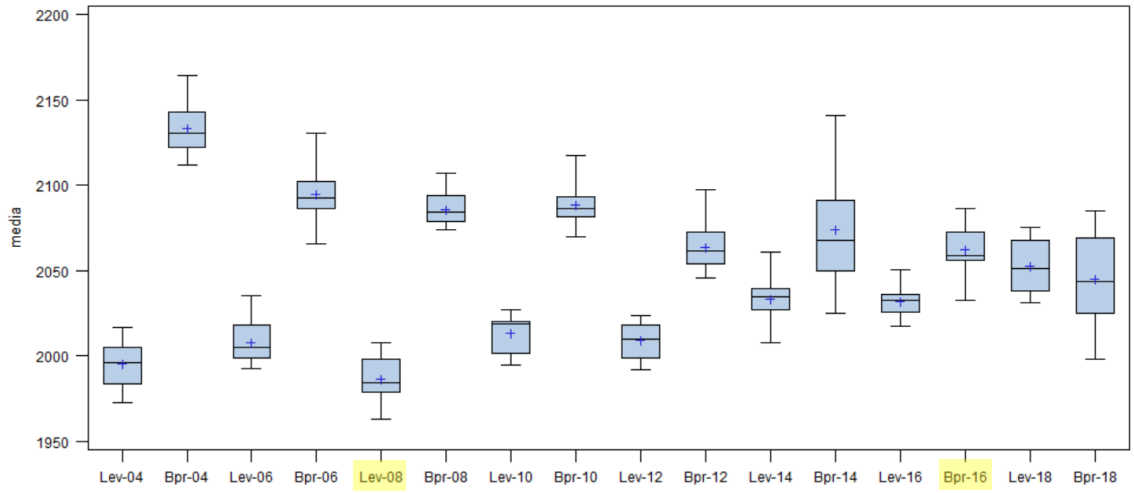


Figura 38. Diagrama de cajas MSE redes de 4 a 18 nodos con algoritmos Levmar y Bprop.

Observamos que con el algoritmo Levmar (impares) la mejor red se alcanza con ocho nodos, número a partir del cual el modelo empezaría a sobreajustar. Sin embargo, con Bprop (pares) este número pasa a ser 16. Adicionalmente, en principio, las redes con Levmar funcionarían mejor que con Bprop. No obstante, este último tiene dos parámetros a tunear: el learning rate que hace variar los pesos en la dirección de descenso del error, y el momentum (siempre menor que uno) que hace que en cada iteración del algoritmo los pesos se vayan cambiando algo menos para acercarse

lentamente al mínimo absoluto. En las redes anteriores estos parámetros se habían fijado a 0.1 y 0.2 respectivamente. Para estudiarlos, probamos nueve distintas especificaciones en las que se disminuye el learning rate y/o momentum, y se mantiene la elección de 16 nodos.

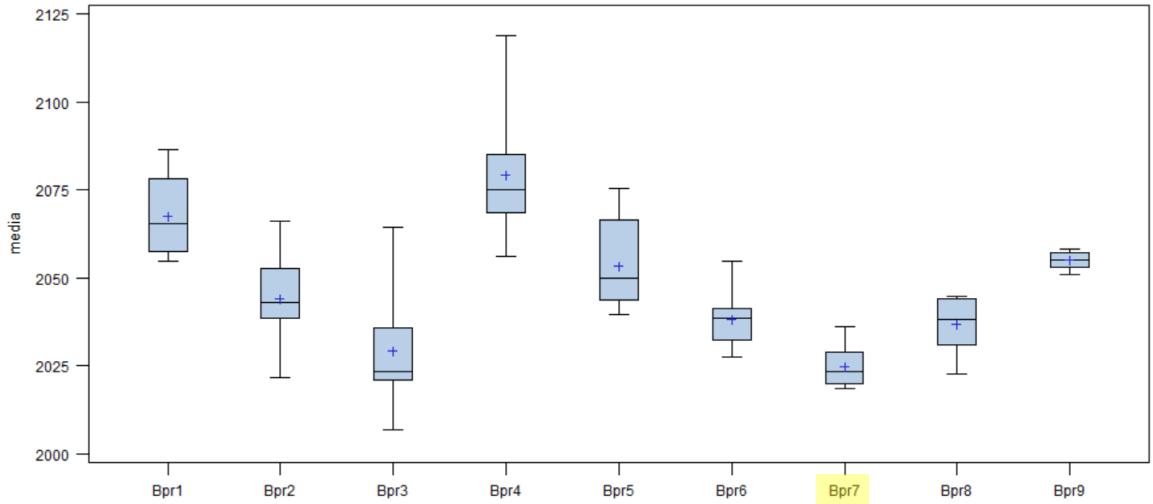


Figura 39. Diagrama de cajas MSE redes Bprop con distintas especificaciones.

La mejor red por obtener bajo sesgo y varianza en MSE es la séptima, que tiene momentum 0.2 y learning rate 0.01. A continuación, estudiamos el early stopping de este modelo y el de la red con Levmar y 8 nodos. La técnica de early stopping consiste en dividir los datos en entrenamiento y validación, y detener el proceso de estimación cuando el error en los datos de validación comienza a aumentar.

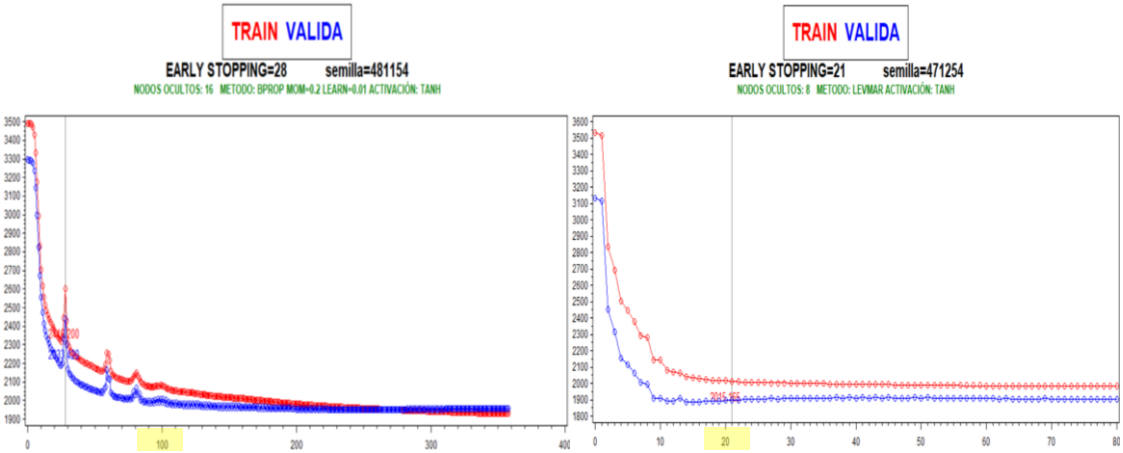


Figura 40. Early stopping red con Bprop y Levmar.

Tras varias iteraciones de la semilla para obtener un resultado más fiable, observamos que el error de los datos de validación para la red con Bprop se estabiliza a partir de las 100 iteraciones, que corresponden a las empleadas hasta el momento. En cambio, para la red con Levmar esto ocurre en torno a las 20 iteraciones, por lo que probamos a fijar

este número y lo comparamos con esta red sin early stopping y con el mejor modelo con Bprop a través de validación cruzada repetida.

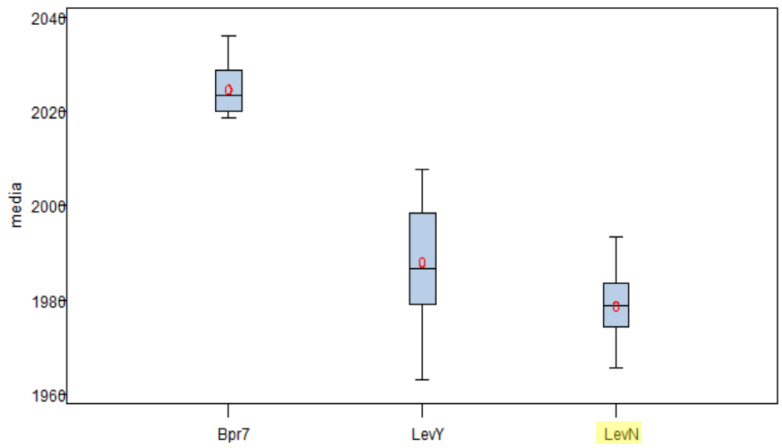


Figura 41. Diagrama de cajas MSE distintas redes.

Aunque al modificar las especificaciones de la red con Bprop se ha conseguido bajar la media de error de 2,060 a 2,025 aproximadamente, no se ha superado a las redes con Levmar. El mejor modelo de red es el tercero, con algoritmo Levmar, 8 nodos y 100 iteraciones. Hasta ahora, todas las redes se habían ejecutado con función de activación tanh, por lo que, como última modificación, probamos con otros tipos de funciones de activación. En particular, con función logística, arctan, lineal, sin, softmax y gauss.

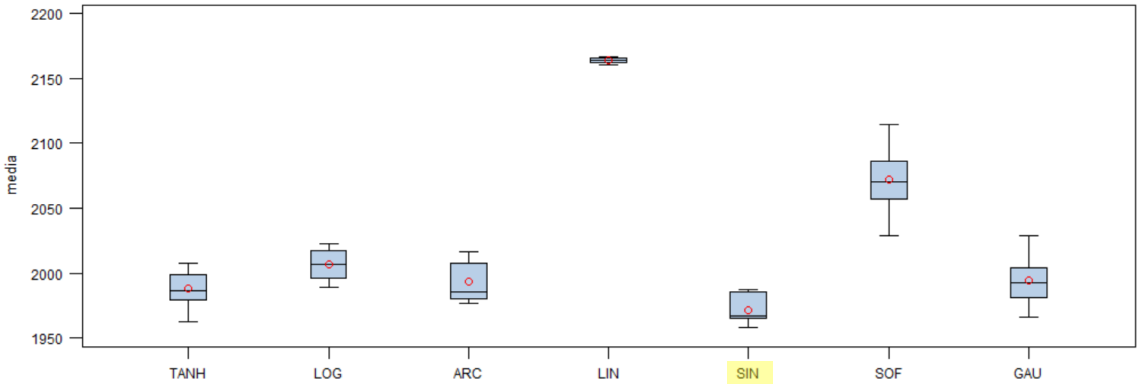


Figura 42. Diagrama de cajas MSE redes con distintas funciones de activación.

De entre estas redes destaca la quinta, con función de activación seno, por tener tanto menor sesgo como varianza. Por lo tanto, la mejor versión de red encontrada en SAS, es una red con algoritmo Levmar, ocho nodos, 100 iteraciones y función de activación seno.

Una vez estimada la mejor red en SAS, pasamos a R para usar la función *avnnet* de la librería *Caret*, que promedia distintas redes, donde para cada una se sortean los valores iniciales de los pesos. Además, *Caret* permite definir una rejilla con distintos

valores para el número de nodos ocultos y learning rate (o weight decay) que queremos probar. Con cada una de las combinaciones posibles a partir de la rejilla y usando validación cruzada repetida se estima una red. Una ventaja adicional de tunear algoritmos en R es el uso de la librería *Parallel* que permite usar los máximos cores del ordenador para realizar las pruebas de la rejilla en paralelo, lo que disminuye considerablemente el tiempo de ejecución. Por ello, utilizaremos estas dos librerías de ahora en adelante para tunear los distintos algortimos en R.

Comenzamos investigando el número de nodos en el rango [4-18] y el learning rate en el rango [0.001-0.1] a través de estas librerías. Para cada una las 24 especificaciones de red resultantes de las combinaciones de la rejilla, se promedian cinco redes.

```
library(caret)
library(parallel)
library(doParallel)

cluster <- makeCluster(detectCores() - 1)
registerDoParallel(cluster)

# Validación cruzada repetida
control<-trainControl(method = "repeatedcv",number=4,repeats=5,savePredictions = "all")

# Rejilla
avnnetgrid <-expand.grid(size=c(4,6,8,10,12,14,16,18),decay=c(0.01,0.1,0.001),bag=FALSE)

redavnnet<- train(precio~dis_centro+dis_airp+demanda+resenas+accesibilidad
                  +antiguedad+puntuacion+AC.T+ascensor.T+anfitrion_recibe.T+balcon.T
                  +apto_ninos.T+parking.T+grandes_electro.T+accesible.T+permite_mascotas.T+capacidad.2
                  +capacidad.3+capacidad.4+capacidad.5+politica_cancelacion.flexible+politica_cancelacion.moderada
                  +tiempo_respuesta.en_unas_horas+tiempo_respuesta.NA+tiempo_respuesta.un_dia_o_mas
                  +tipo.Habitacion+banos.2+banos.3+superanfitrion.T,data=estdata,
                  method="avNNet",linout = TRUE,maxit=100,trControl=control,repeats=5,tuneGrid=avnnetgrid)
```

Figura 43. Ejemplo código R uso librería Caret y Parallel.

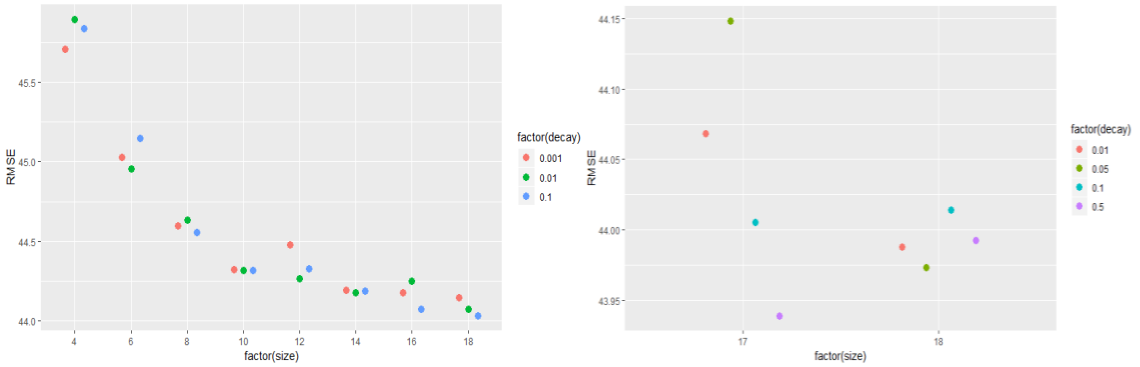


Figura 44. RMSE redes en función del número de nodos y el learning rate.

Tras observar, en el gráfico de la izquierda, que el error cometido disminuía al aumentar el número de nodos y el weight decay, se rehizo la rejilla para analizar las redes con 17 y 18 nodos y weight decays en el intervalo [0.01-0.5]. Los resultados de estas estimaciones se reflejan en el gráfico de la derecha. En principio, la red con 17 nodos y learning rate 0.5 sería la mejor, pero con el fin de obtener resultados más fiables, comparamos este modelo con otros cuatro de los mejores a través de validación cruzada repetida.

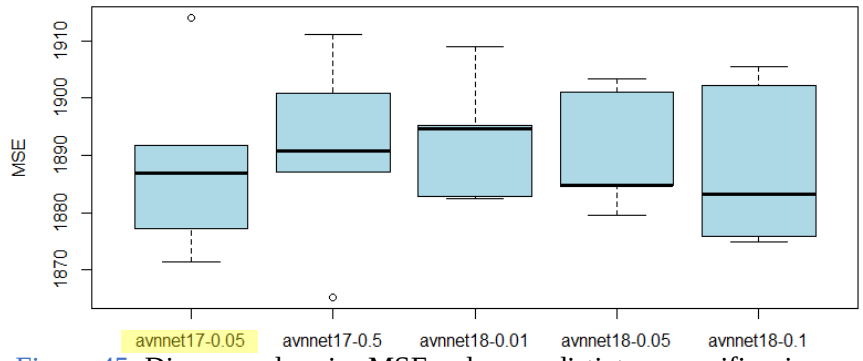


Figura 45. Diagrama de cajas MSE redes con distintas especificaciones.

Escogemos como mejor modelo al primero que tiene 17 nodos y un learning rate de 0.05. Este tiene mejor sesgo que el que parecía ser mejor en la Figura 43, de ahí la importancia de usar técnicas de resmuestreo como validación cruzada repetida. Más adelante, tras la elección de la mejor versión de cada algoritmo, estos se compararán junto a la regresión lineal.

5.6 Random Forest.

Este algoritmo basado en árboles tiene parámetros propios de estos a tunear, como la profundidad, y otros propios de esta técnica como el número de árboles a promediar o el número de variables a sortear en cada nodo. Tras realizar unas pruebas iniciales respecto a este último factor en SAS, se obtuvieron valores de MSE que duplicaban los obtenidos con los otros algoritmos, cuando en R veremos a continuación que no es así, por lo que no se confía del resultado obtenido en SAS y nos centramos en su modelización en R.

Comenzamos explorando el número óptimo de variables a sortear en cada nodo. El set de variables que escogimos anteriormente constaba de 29 parámetros, por lo que se prueban de 4 a 29 (bagging) variables a sortear. Observamos que el menor RMSE se obtiene con 6, 7 y 8 variables, por lo que comparamos modelos de random forest con estas tres distintas especificaciones a través de validación cruzada repetida.

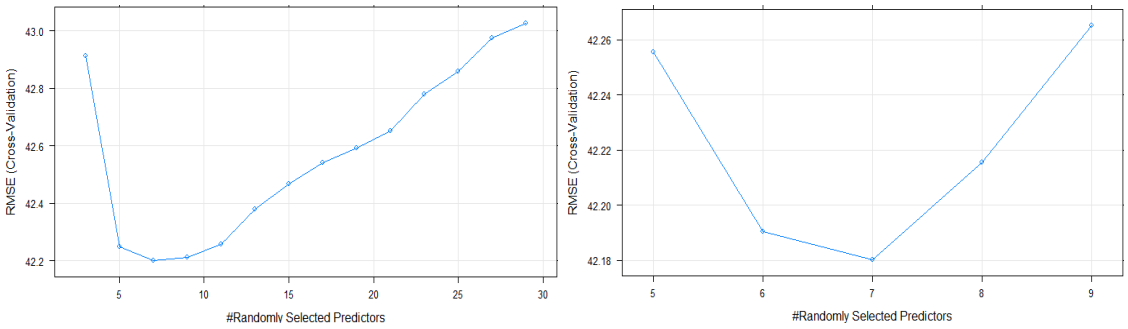


Figura 46. RMSE Random Forest en función del número de variables a sortear.

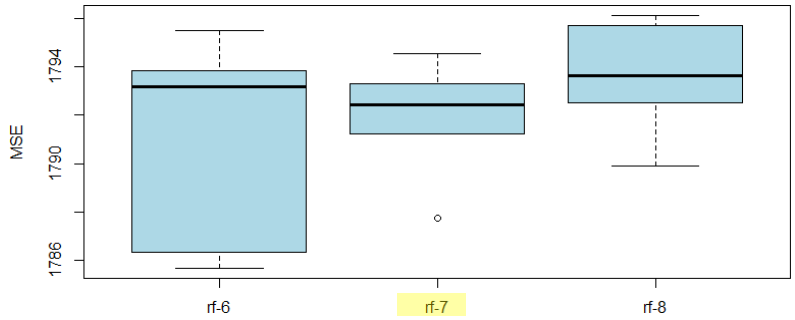


Figura 47. Diagrama de cajas MSE Random Forest con 6, 7 y 8 variables a sortear.

Escogemos el modelo con 7 variables por tener tanto menor sesgo como menor varianza en MSE. A continuación, estudiamos el número de árboles a promediar en el intervalo [0-5000]. En el primer gráfico observamos que el error se mantiene estable a partir de las 500 iteraciones aproximadamente, pero si comparamos con validación cruzada repetida un modelo con 500 árboles, otro con 700 y otro con 1000, vemos que tanto el sesgo como la varianza de este último es menor.

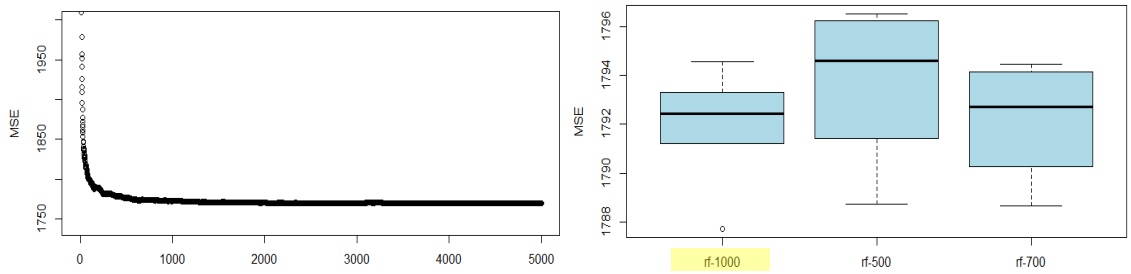


Figura 48. MSE Random Forest en función del número de iteraciones.

Buscando optimizar este modelo, investigamos el tamaño muestral necesario para obtener una buena performance. Para ello, comparamos random forests que utilizan del 13 al 75% de las observaciones, siendo esta la máxima muestra posible debido al uso de validación cruzada.

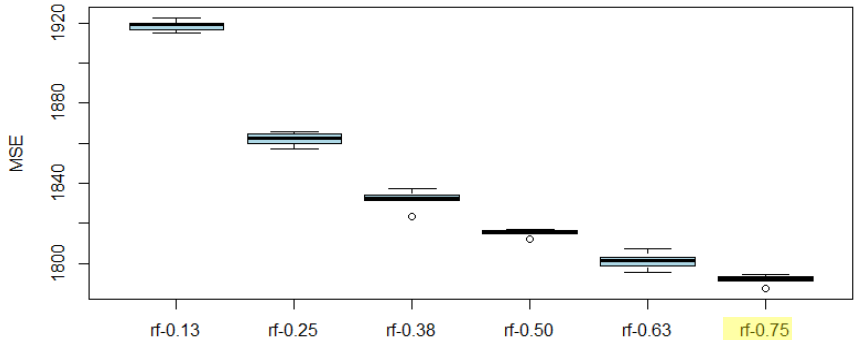


Figura 49. Diagrama de cajas MSE Random Forest en función del tamaño muestral.

Se puede apreciar cómo mejora la performance según aumenta el porcentaje de observaciones usadas, por lo que utilizamos el máximo tamaño muestral. Como última

modificación al modelo, estudiamos modelos de random forest con tamaño de hoja en el rango 5, 10, 15 y 20.

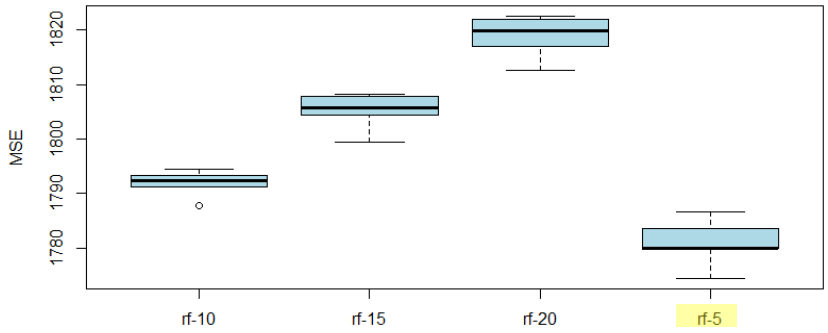


Figura 50. Diagrama de cajas MSE Random Forest en función del tamaño de hoja.

Según disminuye el número de observaciones en la hoja final, se produce una mejora significativa en el sesgo, por lo que escogemos el modelo con tamaño de hoja cinco. Por lo tanto, la mejor versión de este algoritmo encontrada en R, consta de siete variables, 1,000 iteraciones, máximo tamaño muestral y tamaño de hoja cinco.

5.7 Gradient Boosting.

Este algoritmo, además de contar con parámetros como el tamaño de hoja y el número de iteraciones al igual que en random forest, tiene la constante shrinkage, que recordamos es el valor por el cual se multiplican los residuos para obtener la siguiente predicción. Cuanto más pequeña es esta constante, más árboles se necesitan para que el modelo converja. Por lo que para empezar a tunear este algoritmo en SAS, probamos cinco modelos distintos con valores para las iteraciones en el rango [100-1,000] y shrinkage en el rango [0.01-0.1], especificando un mayor número de iteraciones según se reduce el shrinkage.

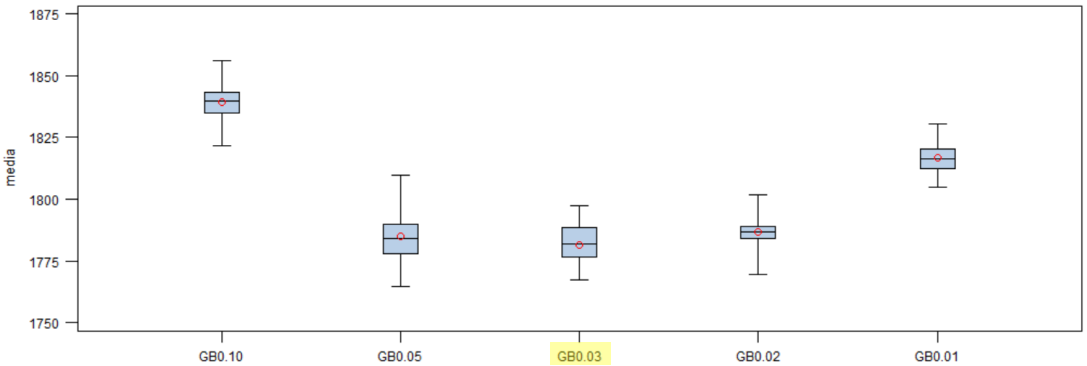


Figura 51. Diagrama de cajas MSE distintos modelos Gradient Boosting.

Elegimos el tercer modelo, con shrinkage 0.03 y 800 iteraciones, por obtener menor media en MSE y una buena varianza. A continuación, manteniendo el shrinkage en 0.03, variamos el número de iteraciones de 100 a 2,000.

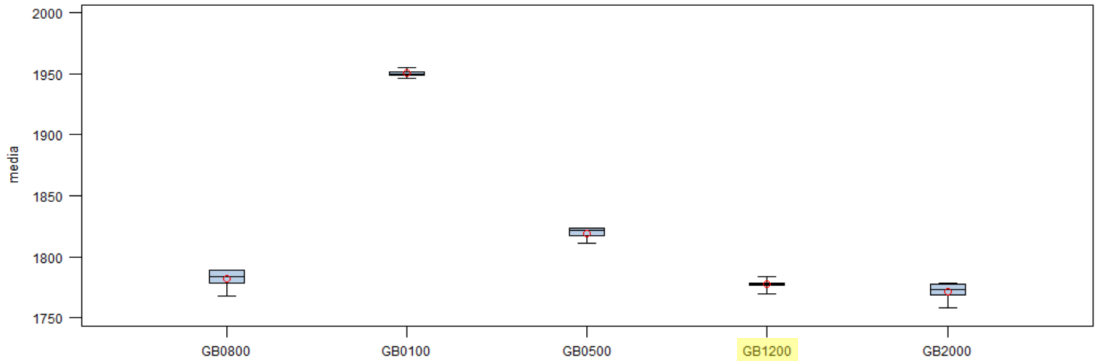


Figura 52. Diagrama de cajas MSE Gradient Boosting con distintas iteraciones.

Observamos como mejora el sesgo al aumentar el número de iteraciones. Escogemos el modelo con 1,200 y como última modificación, varíamos la profundidad de los árboles entre 4 y 8 y el tamaño de hoja entre 5 y 10. Las especificaciones que obtienen mejor resultado son profundidad 6 y tamaño de hoja 10. Por lo tanto, la versión final de gradient boosting en SAS tiene shrinkage 0.03, 1,200 iteraciones, profundidad 6 y tamaño de hoja 10.

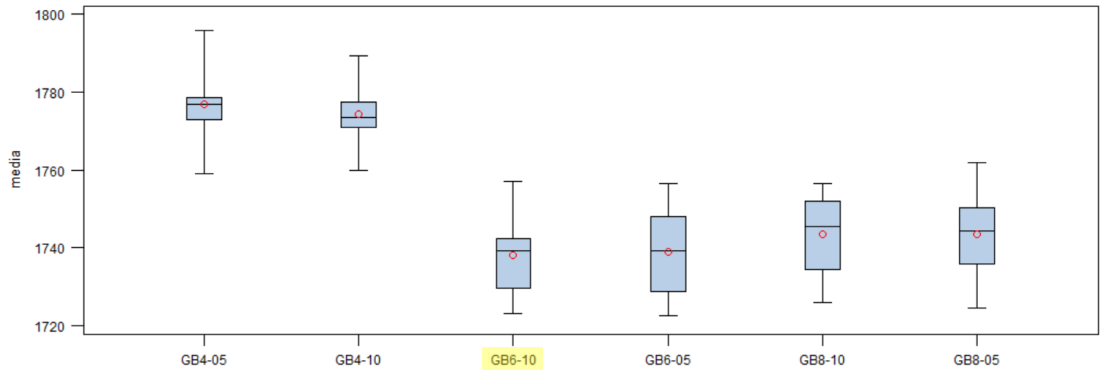


Figura 53. MSE Gradient Boosting con distinta profundidad y tamaño de hoja.

Una vez estimado el mejor modelo de gradient boosting en SAS, tuneamos este algoritmo en R. Comenzamos realizando pruebas simultáneas sobre el shrinkage en el rango [0.001-0.1], el número de árboles en el rango [100-5,000] y 5, 10 y 20 para el tamaño de hoja, para un total de 75 especificaciones distintas.

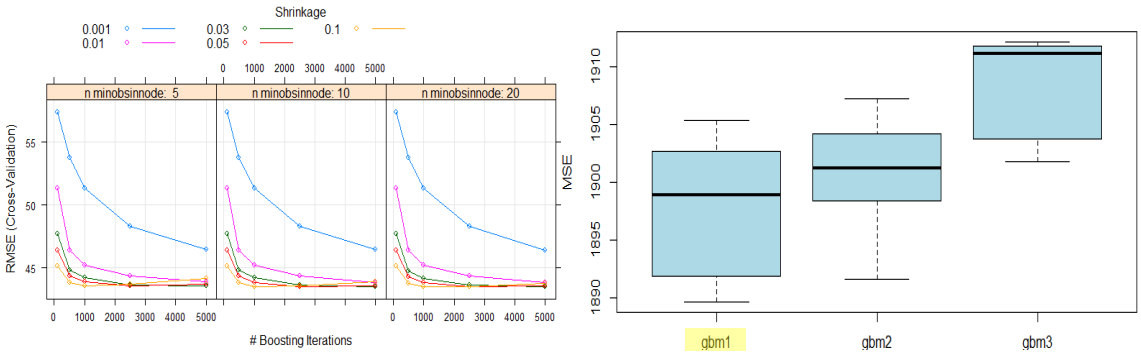


Figura 54. RMSE y MSE distintos modelos Gradient Boosting.

En el gráfico de la izquierda se aprecia claramente como disminuye el error cometido según aumenta el shrinakge y el número de iteraciones. Las tres especificaciones que obtienen mejor resultado y que se comparan en el gráfico de la derecha con validación cruzada repetida son:

- Shrinkage = 0.05, tamaño de hoja = 10, iteraciones = 2,500
- Shrinkage = 0.03, tamaño de hoja = 10, iteraciones = 500
- Shrinkage = 0.1, tamaño de hoja = 20, iteraciones = 1,000

De estos tres modelos, se selecciona el segundo por tener menor sesgo. A continuación, varíamos el tamaño de hoja a 5, 15 y 20, y tras la comparación en la siguiente figura, se escoge el modelo con mayor tamaño de hoja por lograr una menor varianza. Por último, puesto que este modelo consta de 2,500 iteraciones, estudiamos su early stopping.

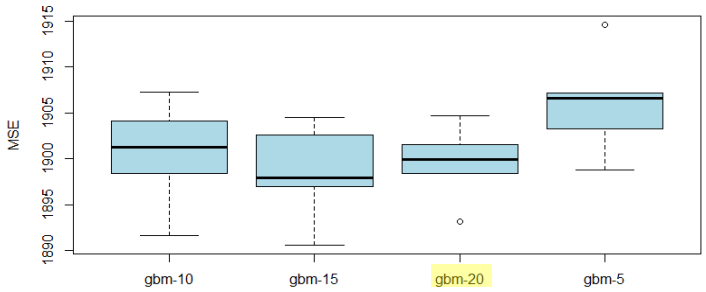


Figura 55. Diagrama de cajas MSE Gradient Boosting con distintos tamaños de hoja.

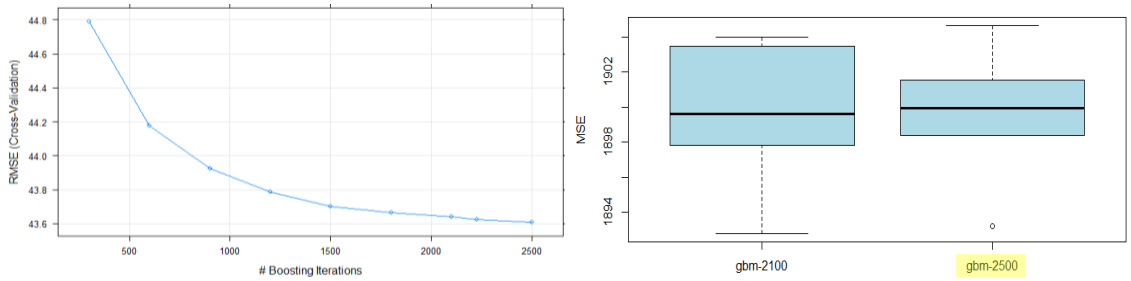


Figura 56. RMSE y MSE Gradient Boosting variando el número de iteraciones.

Aunque parece que el error cometido se estabiliza a partir de las 2,100 iteraciones aproximadamente, la varianza se reduce a la mitad cuando mantenemos las 2,500, por lo que mantenemos este número de iteraciones. Por lo tanto, la mejor versión de este algoritmo en R consta de shrinkage 0.05, tamaño de hoja 20 y 2,500 iteraciones.

5.8 Xgboost.

Este algoritmo, que es una modificación del anterior, se estimará solo en R ya que SAS no cuenta de momento con ninguna función para ello. Por lo tanto, siguiendo el mismo proceso y valores iniciales que para el tuneo de gradient boosting en R,

realizamos las mismas 75 especificaciones simultáneas sobre el shrinkage, el número de árboles y el tamaño de hoja.

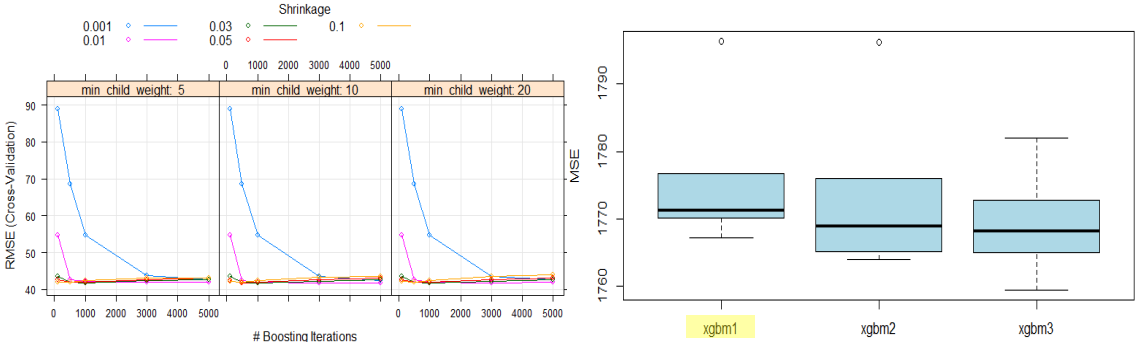


Figura 57. RMSE y MSE distintos modelos Xgboost.

Como era de esperar, los modelos con valores de shrinkage más pequeños necesitan de un mayor número de iteraciones para disminuir el error cometido. Los tres mejores modelos y que se comparan con validación cruzada en el diagrama de cajas constan de:

- Shrinkage = 0.05, tamaño de hoja = 10, iteraciones = 500
- Shrinkage = 0.01, tamaño de hoja = 10, iteraciones = 3,000
- Shrinkage = 0.01, tamaño de hoja = 20, iteraciones = 3,000

De entre estos tres, escogemos el primero por tener un sesgo parecido al resto y menor varianza, además de un menor número de iteraciones. Puesto que la novedad de extreme gradient boosting frente a gradient boosting es la función de penalización, evaluamos también el parámetro gamma que penaliza el número de hojas. Para este factor se prueban valores de 0 a 1, manteniendo el resto de especificaciones del modelo previamente seleccionado.

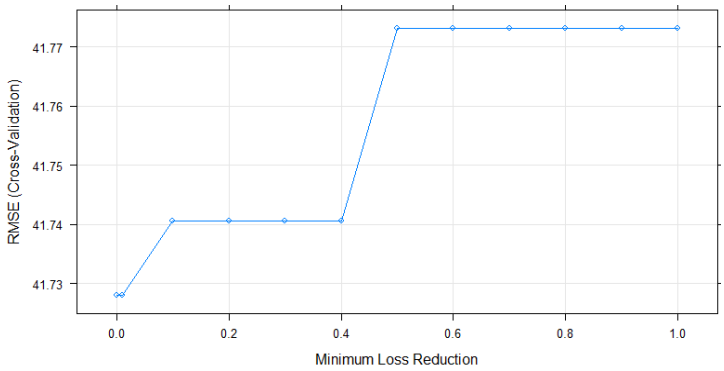


Figura 58. RMSE modelo Xgboost en función de gamma.

Observamos cómo aumenta el RMSE según aumenta gamma, por lo que lo mantenemos en cero. A continuación, con el fin de mejorar nuestro modelo, comparamos este con otro igual pero añadiendo el sorteo del 80% de las variables antes de cada árbol, y con otro donde se sortean el 80% de las observaciones.

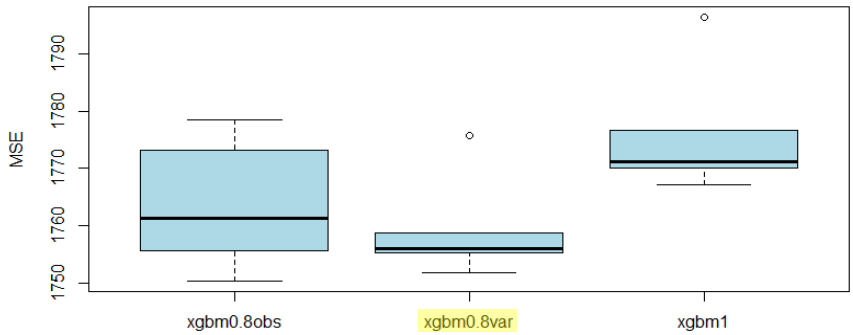


Figura 59. Diagrama de cajas MSE distintos Xgboost.

Tanto el sesgo como la varianza han experimentado una mejora considerable al sorterar el 80% de las variables. Por último, como última modificación, estudiamos el tamaño de hoja óptimo de entre los valores 5, 10, 15 y 20 con validación cruzada repetida.

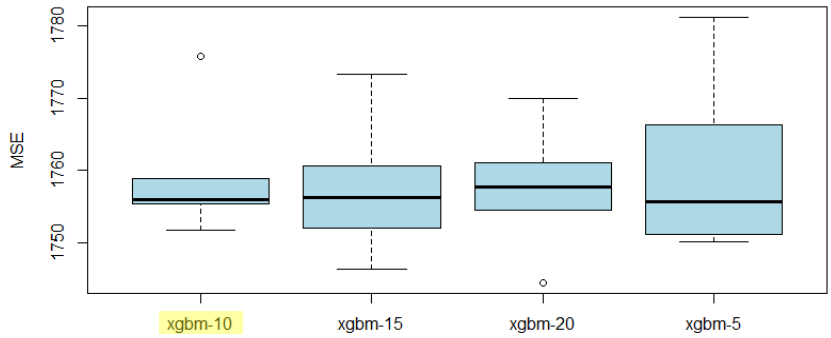


Figura 60. Diagrama de cajas MSE Xgboost en función del tamaño de hoja.

Se puede apreciar que, aunque el sesgo se mantiene prácticamente igual, la varianza del modelo que cuenta con mínimo diez observaciones en cada nodo es mejor, por lo que es el seleccionado. Por lo tanto, la versión elegida de extreme gradient boosting tiene shrinkage 0.05, 500 iteraciones, gamma cero, tamaño de hoja diez y sorteo del 80% de las variables antes de cada árbol.

5.9 Support Vector Machine.

En este apartado se estimarán los modelos de separación de hiperplanos con tres kernels distintos. Esto se realizará solamente en R.

5.9.1 Kernel lineal.

Comenzamos por el modelo más sencillo que usa un kernel lineal. Este modelo solamente requiere especificar la constante de regularización C, por lo que probamos nueve valores diferentes en el rango [0-50].

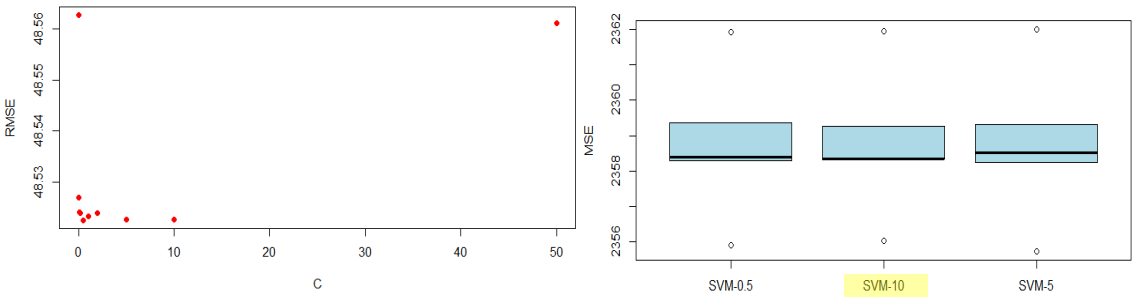


Figura 61. RMSE y MSE SVM lineal con distintos valores de C.

Los modelos con especificación C = 0.5, 5 y 10, cometen prácticamente el mismo error, por lo que los comparamos a través de validación cruzada repetida en el gráfico de la derecha. La performance vuelve a ser muy parecida, aunque el sesgo y la varianza del segundo modelo es ligeramente mejor que la de los otros dos.

5.9.2 Kernel polinomial

Esta especificación de SVM, además del parámetro C, requiere investigar el grado del polinomio y la escala. Para ello, se realizan pruebas con polinomios de grado dos y tres, valores de C en el rango [0.01-1] y valores para la escala en el rango [0.1-2] para un total de 40 especificaciones distintas.

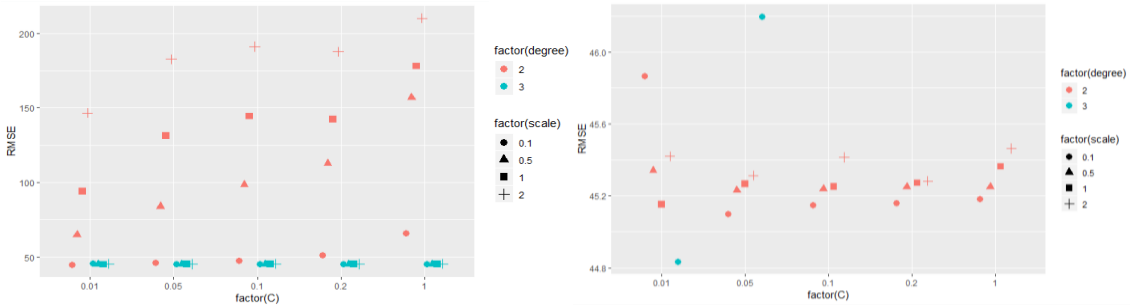


Figura 62. RMSE SVM polinomial en función del grado, escala y C.

En el gráfico de la izquierda se aprecia la interdependencia entre estos tres parámetros. A mayor grado, mayor C y mayor escala, mayor es el RMSE obtenido. Además, observamos que, a excepción de con escala 0.1, el kernel polinomial de grado tres comete bastante más error que el de grado dos. Por ello, en el gráfico de la derecha nos centramos solamente en los que obtienen un buen resultado. Las mejores tres especificaciones y que se comparan a continuación con validación cruzada repetida son:

- Grado = 3, C = 0.01, escala = 0.1
- Grado = 2, C = 0.05, escala = 0.1
- Grado = 2, C = 0.1, escala = 0.1

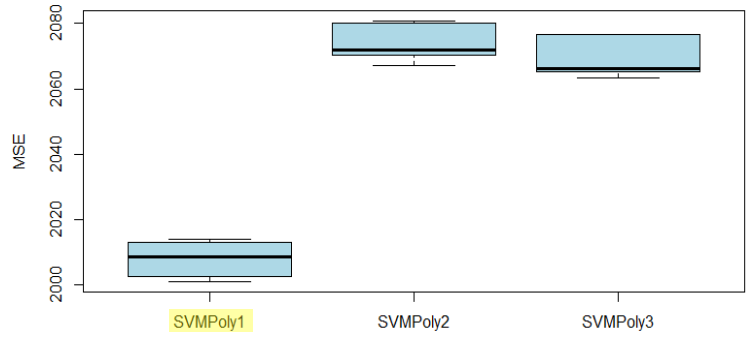


Figura 63. Diagrama de cajas MSE distintos modelos SVM polinomial.

El mejor modelo con diferencia es el primero con polinomio de grado tres y que es más agresivo que los otros dos modelos de grado dos.

5.9.3 Kernel radial

El uso del kernel radial, también llamado Gaussiano, es más general que el anterior y junto al parámetro C, solamente necesita especificarse el parámetro sigma. En principio, los modelos más agresivos, y por lo tanto con mejor sesgo, son aquellos con valores más altos de sigma, pero la interdependencia con C puede hacer que esto cambie. Para comprobarlo, probamos valores tanto de C como de sigma en el rango [0.01-30] para un total de 100 especificaciones distintas.

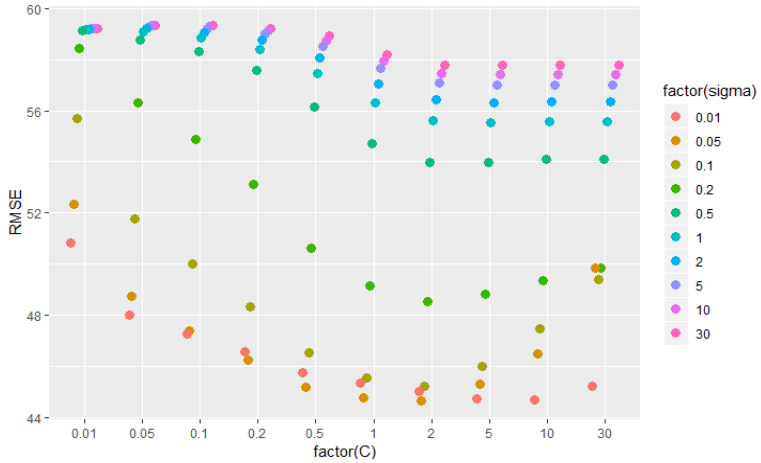


Figura 64. RMSE SVM radial en función de C y sigma.

La interdepdencia entres los factores es clara: a mayor C y menor sigma, menor RMSE. En particular, las especificaciones de los tres mejores modelos, y que se comparan en el siguiente gráfico de cajas, son (C 10, sigma 0.01), (C 2, sigma 0.05), (C 5, sigma 0.01).

- C = 10, sigma= 0.01
- C = 2, siga = 0.05
- C = 5, siga = 0.01

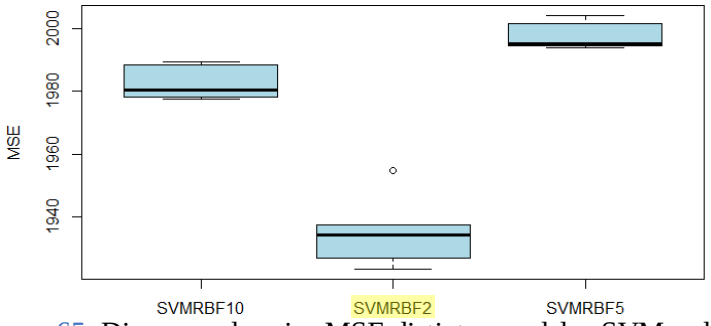


Figura 65. Diagrama de cajas MSE distintos modelos SVM radial.

El modelo seleccionado es el segundo por ser significativamente mejor en sesgo.

5.10 Evaluación de modelos.

Una vez estimados y tuneados los siete tipos de algoritmos de Machine Learning, los comparamos vía validación cruzada repetida de 4 grupos y 20 semillas. Se presentan en un mismo gráfico los modelos de R y de SAS (estos últimos subrayados en azul). Se añade regresión lineal en ambos programas como representación del modelo hedónico.

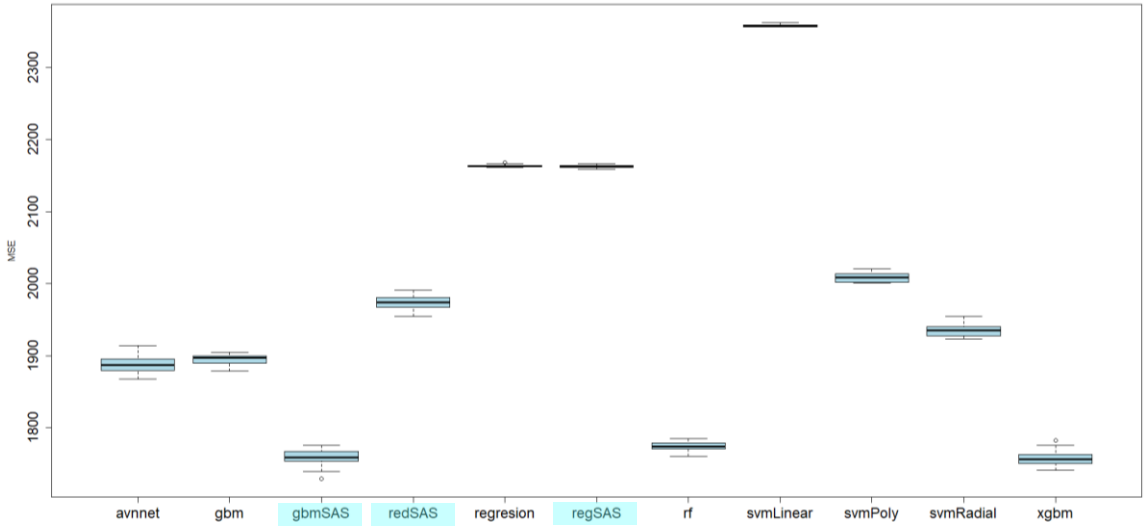


Figura 66. Comparación modelos de Machine Learning.

Observamos que todos los modelos de Machine Learning, a excepción del SVM con kernel lineal, obtienen mejores resultados que la regresión lineal, lo que indica que estamos tratando con datos con relaciones no lineales. De entre todos los algoritmos, destacan random forest y extreme gradient boosting en R, y gradient boosting en SAS.

Se aprecia también que la red en R comete menos error que la de SAS, que podría indicar la utilidad del uso de la función *avnnet* con la que promediamos cinco redes. En cambio, en gradient boosting obtiene mejores resultados el de SAS. Hay que tener en

cuenta que estas diferencias pueden debserse a la reordenación de grupos en validación cruzada y no al algoritmo en sí.

5.11 Ensamblado.

Finalmente, y para concluir con las estimaciones, investigamos el método Ensemble. Para este método usaremos la técnica Stacking que consiste en promediar las predicciones de distintos modelos. Los modelos a combinar son la mejor versión encontrada de cada algoritmo tanto en SAS como en R. En particular, se promediarán:

- En SAS:
    - Regresión lineal.
    - Red neuronal con algoritmo Levmar, ocho nodos, 100 iteraciones y función de activación seno.
    - Gradient boosting con shrinkage 0.03, 1,200 iteraciones, profundidad seis y hoja diez.
- En R:
    - Regresión lineal.
    - Cinco redes promediadas de 17 nodos y learning rate 0.05.
    - Random forest de siete variables a sortear, 1,000 iteraciones, máximo tamaño muestral y tamaño de hoja cinco.
    - Gradient boosting con shrinkage 0.05, 2,500 iteraciones y tamaño de hoja 20.
    - Extreme gradient boosting con shrinkage 0.05, 500 iteraciones, gamma cero, tamaño de hoja 10 y sorteando el 80% de las observaciones.
    - SVM lineal con C = 0.5.
    - SVM polinomial con grado = 3, C = 0.01 y escala = 0.1.
    - SVM radial con C = 2 y sigma = 0.05.

Se promedian de dos a tres algoritmos en SAS para un total de cinco ensamblados (subrayados en azul), y de dos a cinco algoritmos en R para un total de 62. Se presentan todos los ensamblados junto a los algoritmos individuales (subrayados en verde los de R y en rosa los de SAS) en el siguiente diagrama de cajas.

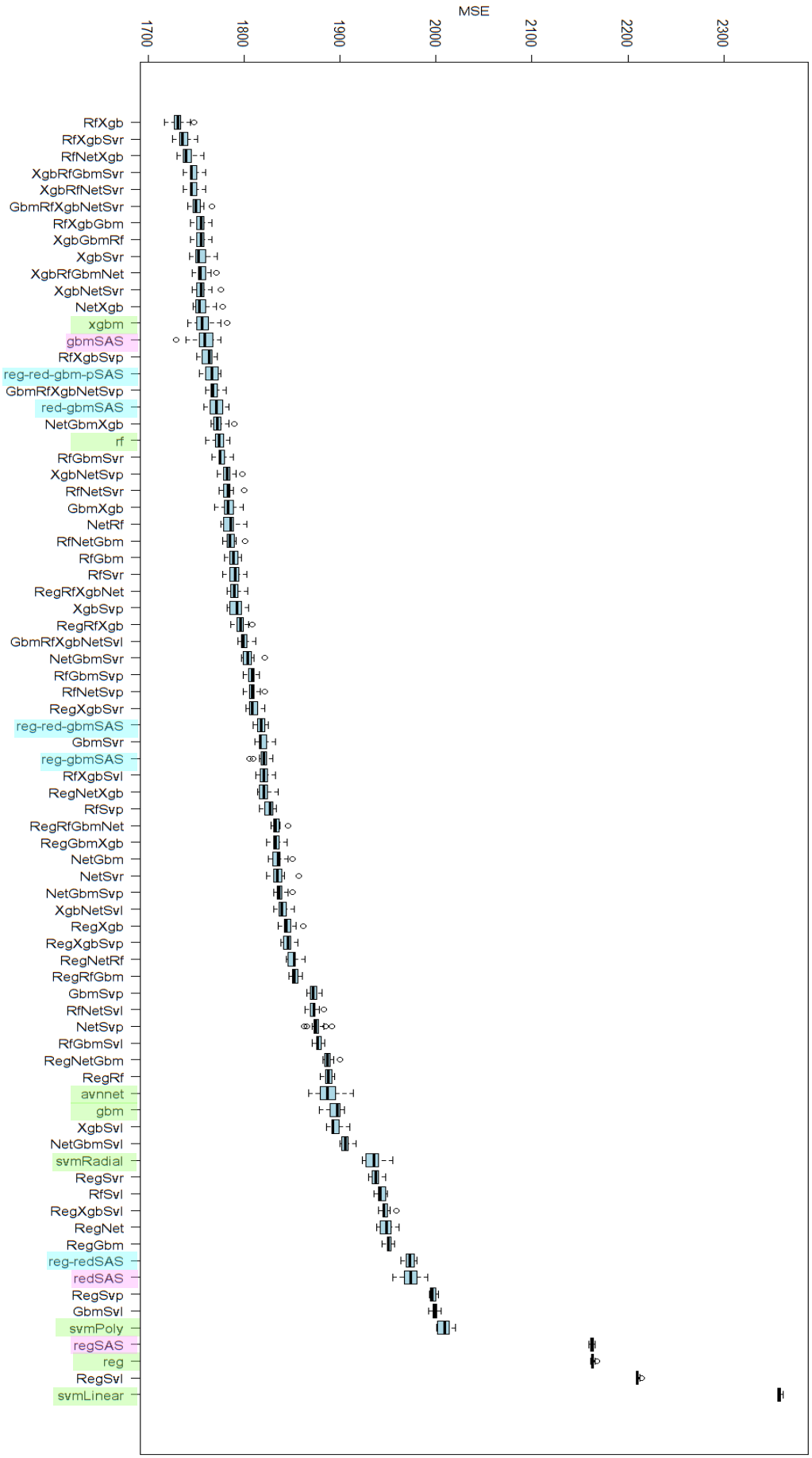


Figura 67. Diagrama de cajas ensamblados y algoritmos individuales.

La mayoría de algoritmos individuales son superados por los modelos de ensamblado, demostrando la eficacia de esta técnica. Los modelos que cometen mayor error son la regresión lineal y el SVM lineal, y por consecuente, el ensamblado entre ellos. Estos resultados reflejan la presencia de relaciones no lineales en los datos que no pueden ser captadas por modelos hedónicos.

Entre los mejores 15 modelos encontramos dos algoritmos individuales: el extreme gradient boosting de R y el gradient boosting de SAS. Nos fijamos a continuación en estos 15.

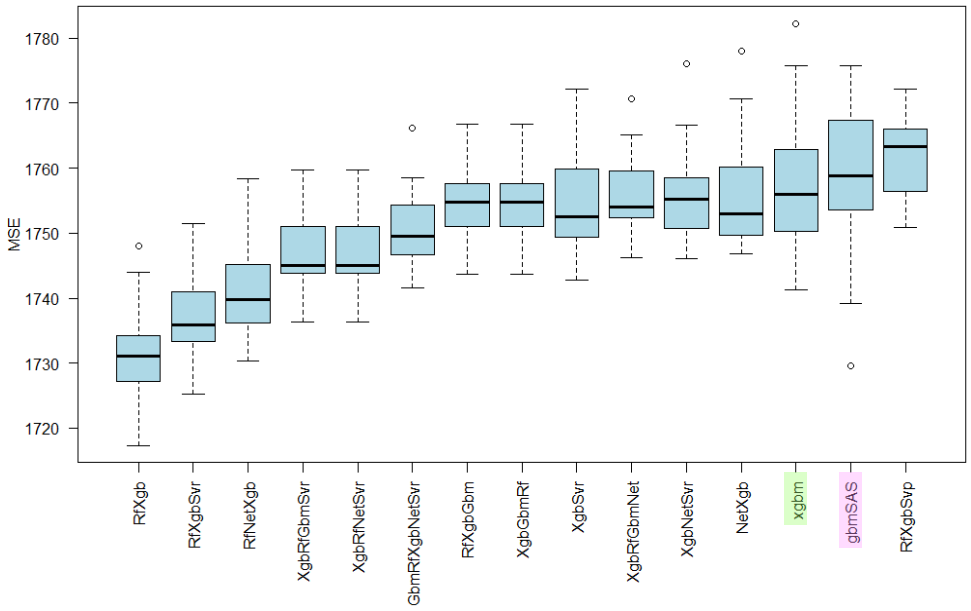


Figura 68. Diagrama de cajas MSE 15 mejores modelos.

Los mejores ensamblados tienen en común que promedian el modelo de extreme gradient boosting, mejorando tanto el sesgo como la varianza de este. El modelo ganador promedia este algoritmo junto a random forest, consiguiendo una media en MSE de 1731 y un R^2 de 0.49, mientras que en la regresión lineal estos valores eran 2163 y 0.39 respectivamente. Esto supone una mejora en RMSE del 5% y del 10% en R^2.

6. Conclusiones.

En este trabajo se han investigado tanto los factores determinantes en el precio de los alojamientos de Airbnb mediante modelos hedónicos espaciales, como la predicción del precio a través de seis técnicas de Machine Learning. El uso de modelos espaciales tenía como objetivo captar la correlación espacial presente en los datos, y el uso de algoritmos más avanzados, las relaciones no lineales. Las estimaciones se han llevado a

cabo a través de una muestra de 16,632 alojamientos de Airbnb en la ciudad de Madrid y de 32 variables explicativas, de las cuales cabe mencionar la creación de la variable *accesibilidad*, mediante el API de Foursquare, que recoge la distancia media de cada alojamiento a los cinco servicios (restaurantes, supermercados, etc) más cercanos.

Tras la depuración de los datos en la que se han realizado tareas como la imputación de valores ausentes o la agrupación de niveles con poca representatividad en variables categóricas, se ha llevado a cabo un análisis visual para recoger información que de otro modo se habría obviado. Así, mediante diversos tipos de mapas y gráficos se ha podido apreciar: la alta densidad de alojamientos de Airbnb en los distritos centrales de Madrid y su mayor precio medio con respecto al resto de distritos; el incremento en el precio medio anual; la estacionalidad en la oferta y demanda con un claro pico en los meses de verano; o el descenso en la oferta como probable consecuencia de las restricciones del ayuntamiento de Madrid a Airbnb.

Respecto a la estimación de modelos hedónicos espaciales, ha sido necesaria la triangulación de los alojamientos para determinar aquellos vecinos entre sí. Con esto y los residuos de la regresión OLS, se ha comprobado la presencia de dependencia espacial mediante el estadístico I de Moran. A continuación, con una prueba basada en el principio del multiplicador de Lagrange, se ha determinado que la especificación espacial más conveniente era el modelo autorregresivo espacial (SAR), aunque también se ha estimado el modelo de error espacial (SEM). Los resultados obtenidos con estas estimaciones, coinciden con los encontrados en la literatura. Así, se ha corroborado que la mayor capacidad del alojamiento, el mayor número de baños, la mayor puntuación y la pertenencia del anfitrión a la categoría de "superhosts" de Airbnb, tiene un efecto positivo en el precio. De igual forma, el aumento en el número de reseñas corresponde a alojamientos más baratos, lo que refleja una mayor frecuencia de uso en Airbnbs más económicos.

En cuanto a la localización de los alojamientos, se ha podido comprobar que aquellos situados en el distrito centro de Madrid tenían precios más elevados. No obstante, el impacto de la variable de distancia al centro en el modelo SAR se veía reducido con respecto a los otros dos modelos. Donde se percibió un mayor cambio entre las distintas estimaciones fue en la variable *accesibilidad*. Mientras que en la

regresión lineal esta parecía muy significativa y deducía que el precio por noche del alojamiento descendía en unos 13€/noche por cada kilómetro de más en la distancia a servicios como restaurantes o supermercados, esta variable perdía su significatividad en los modelos espaciales. Esta pérdida ocurría también en las variables indicadoras de que el alojamiento tuviese parking y de que fuese accesible para personas con movilidad reducida.

Las diferencias en la significatividad de las variables entre la regresión OLS y sus especificaciones espaciales pueden ser un indicador de que los resultados de las estimaciones del primero resultan ineficientes en casos con presencia de correlación espacial. Además, el uso del modelo SAR nos ha permitido verificar la existencia de impactos indirectos en el precio y cuantificar los efectos que un cambio en las características de un alojamiento tiene en sus alojamientos vecinos.

Desde el punto de vista predictivo, los estadísticos de ajuste no reflejaban ninguna diferencia sustancial entre las tres especificaciones de regresión. Sin embargo, tanto los tests de correlación espacial en los residuos del modelo hedónico, como las pruebas de razón de verosimilitud, determinaban que la bondad de ajuste del modelo autorregresivo espacial era superior a la de los otros dos. El R^2 de estos modelos era 39.65% y el RMSE 45.28%.

Buscando mejorar estos valores, se ha llevado a cabo una extensa investigación sobre cinco tipos de algoritmos de Machine Learning: Red neuronal, random forest, gradient boosting, extreme gradient boosting y support vector machine con kernel lineal, polinomial y radial. Adicionalmente, se han realizado 67 ensamblados entre las mejores versiones de cada algoritmo hasta concluir que el mejor modelo era un promedio entre las predicciones de random forest y extreme gradient boosting, con el que se conseguía un R^2 de 49.37% y un RMSE de 41.60%. Aunque estos valores siguen siendo bajos, sí que representan una mejora significativa frente a los modelos hedónicos, los cuales tienen la desventaja de no poder captar relaciones no lineales.

Al igual que otros estudios en los que se emplean los datos de Inside Airbnb, este trabajo tiene limitaciones. La variable dependiente correspondía al precio por noche marcado cuando se hizo el *webscrapping,* en vez del precio en el que el alojamiento fue

alquilado. Adicionalmente, no se han tenido en cuenta variables más cualitativas como por ejemplo socio-demográficas, que pueden influenciar el precio en plataformas colaborativas (Chattopadhyay & Mitra, 2019).

7. Bibliografía.

Airbnb Statistics [2020]: User & Market Growth Data. (s. f.). Recuperado 2 de enero de 2020, de https://ipropertymanagement.com/research/airbnb-statistics

Borrego, J. Á. (2018). *Modelos de Regresión para Datos Espaciales* [Trabajo Fin de Grado, Universidad de Sevilla]. Recuperado 4 de abril de 2020, de https://idus.us.es/bitstream/handle/11441/81660/Borrego%20S%E1nchez%20Jos%E9%20%C1ngel%20TFG.pdf;jsessionid=402DA454A8BB0963AF3718482A2D0814?sequence=1&isAllowed=y

Calviño, A. (2019). *Apuntes de la asignatura Técnicas y metodología de la minería de datos (SEMMA)*.

Carillo, G. (2019, noviembre 6). *Gracecarrillo/Predicting-Airbnb-prices-with-machine-learning-and-location-data*. GitHub. https://github.com/gracecarrillo/Predicting-Airbnb-prices-with-machine-learning-and-location-data

Chattopadhyay, M., & Mitra, S. K. (2019). Do airbnb host listing attributes influence room pricing homogenously? *International Journal of Hospitality Management*, *81*, 54-64. https://doi.org/10.1016/j.ijhm.2019.03.008

Chica-Olmo, J., González-Morales, J. G., & Zafra-Gómez, J. L. (2020). Effects of location on Airbnb apartment pricing in Málaga. *Tourism Management*, *77*, 103981. https://doi.org/10.1016/j.tourman.2019.103981

Deboosere, R., Kerrigan, D. J., Wachsmuth, D., & El-Geneidy, A. (2019). Location, location and professionalization: A multilevel hedonic analysis of Airbnb listing prices and revenue. *Regional Studies, Regional Science*, *6*(1), 143-156. https://doi.org/10.1080/21681376.2019.1592699

Dudas, G., Kovalcsik, T., Vida, G., Boros, L., & Nagy, G. (2020). Price determinants of Airbnb listing prices in Lake Balaton Touristic Region, Hungary. *European Journal of Tourism Research*, *24*, UNSP 2410.

España, el tercer país donde Airbnb genera más negocio. (2019, julio 1). Expansión.com. https://www.expansion.com/economia-digital/companias/2019/07/01/5d1a4a6de5fdead3778b4607.html

Hernández, Á. (2018, julio 5). *Diez años de Airbnb: Así ha invadido (y encarecido) las capitales españolas*. eldiario.es. https://www.eldiario.es/hojaderouter/internet/Airbnb-invadido-encarecido-capitales-espanolas_0_789521674.html

Hill, D. (2015). How much is your spare room worth? *IEEE Spectrum*, *52*(9), 32-58. https://doi.org/10.1109/MSPEC.2015.7226609

Inside Airbnb: Madrid. Adding data to the debate. (s. f.). Inside Airbnb. Recuperado 2 de enero de 2020, de http://insideairbnb.com/madrid

Hosteltur. (2017, mayo 4). *Airbnb pacta en San Francisco mientras negocia en Madrid | Hoteles y Alojamientos*. Hosteltur: Toda la información de turismo. https://www.hosteltur.com/121844_airbnb-pacta-san-francisco-mientras-negocia-madrid.html

Lewis, L. (2019, mayo 22). *Exploring Airbnb prices in London: Which factors influence price?* Medium. https://towardsdatascience.com/predicting-airbnb-prices-with-deep-learning-part-2-how-to-improve-your-nightly-price-50ea8bc2bd29

Martín, J. J. (s. f.) *Support Vector Regression: Propiedades y aplicaciones* [Trabajo Fin de Grado, Universidad de Sevilla]. Recuperado 15 de abril de 2020, de https://idus.us.es/bitstream/handle/11441/43808/Mart%C3%ADn%20Guare%C3%B1o%2C%20Juan%20Jos%C3%A9%20TFG.pdf?sequence=1&isAllowed=y

Moreno-Izquierdo, L., Egorova, G., Peretó-Rovira, A., & Más-Ferrando, A. (2018). Exploring the use of artificial intelligence in price maximisation in the tourism sector: Its application in the case of Airbnb in the Valencian Community. *Investigaciones Regionales; Madrid*, *42*, 113-128.

Önder, I., Weismayer, C., & Gunter, U. (2019). Spatial price dependencies between the traditional accommodation sector and the sharing economy. *Tourism Economics*, *25*(8), 1150-1166. https://doi.org/10.1177/1354816618805860

Page, D. (2018, mayo 17). *El negocio del alquiler turístico en Madrid: El precio de los pisos se dispara un 50% en tres años*. El Independiente. https://www.elindependiente.com/economia/2018/05/17/negocio-del-alquiler-turistico-madrid-precio-los-pisos-se-dispara-50-tres-anos/

Perez-Sanchez, V., Serrano-Estrada, L., Marti, P., & Mora-Garcia, R.-T. (2018). The What, Where, and Why of Airbnb Price Determinants. *Sustainability*, *10*(12), 4596. https://doi.org/10.3390/su10124596

Portela, J. (2019). *Apuntes de la asignatura Machine Learning.*

Pozzi, S. (2019, septiembre 19). *Airbnb se prepara para salir a Bolsa en 2020 | Economía | EL PAÍS*. Recuperado 2 de enero de 2020, de https://elpais.com/economia/2019/09/19/actualidad/1568908606_575608.html

Quirós, C. (2019): Nuevas implicaciones de las TIC: expansión de la economía colaborativa. *Working Paper No. 2. Foro Europa-Cuba. Jean Monnet Network*.

Sarmiento-Barbieri, I. (2016, abril). *An Introduction to Spatial Econometrics in R*. http://www.econ.uiuc.edu/~lab/workshop/Spatial_in_R.html

Sopranzetti, B. (2015). *Hedonic Regression Models*. 2119-2134. https://doi.org/10.1007/978-1-4614-7750-1_78

Tang, L. (Rebecca), Kim, J., & Wang, X. (2019). Estimating spatial effects on peer-to-peer accommodation prices: Towards an innovative hedonic model approach. *International Journal of Hospitality Management*, *81*, 43-53. https://doi.org/10.1016/j.ijhm.2019.03.012

Tong, B., & Gunter, U. (s. f.). Hedonic pricing and the sharing economy: How profile characteristics affect Airbnb accommodation prices in Barcelona, Madrid, and Seville. *Current Issues in Tourism*. https://doi.org/10.1080/13683500.2020.1718619

Wang, D., & Nicolau, J. L. (2017). Price determinants of sharing economy based accommodation rental: A study of listings from 33 cities on Airbnb.com. *International Journal of Hospitality Management*, *62*, 120-131. https://doi.org/10.1016/j.ijhm.2016.12.007

Ye, Q., Law, R., & Gu, B. (2009). The impact of online user reviews on hotel room sales. *International Journal of Hospitality Management*, *28*(1), 180-182. https://doi.org/10.1016/j.ijhm.2008.06.011

Yuan Cai, Yongbo Zhou, Jianyu (jenny) Ma, & Scott, N. (2019). Price Determinants of Airbnb Listings: Evidence from Hong Kong. *Tourism Analysis*, *24*(2), 227-242. https://doi.org/10.3727/108354219X15525055915554

Zhang, Z., Chen, R. J. C., Han, L. D., & Yang, L. (2017). Key Factors Affecting the Price of Airbnb Listings: A Geographically Weighted Approach. *Sustainability*, *9*(9), 1635. https://doi.org/10.3390/su9091635

8. Anexo.

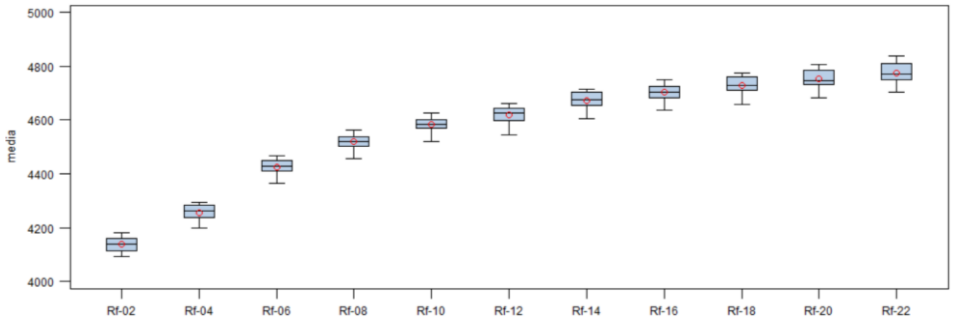


Figura 69. MSE Random Forest en función del número de variables a sortear en SAS.

8.1 Código creación de las variables sobre comodidades en R.

```
### ----- Amenities ----- ###
library(readxl)
datos <- read_excel("listings.xlsx")
# Creo un único string con todas las amenities
amenities_list <- list(datos$amenities)
amenities_list_string <- paste(unlist(amenities_list), collapse='')
# Quito corchete inicial y final
amenities_list_string <- gsub("{", "", amenities_list_string, fixed = T)
amenities_list_string <- gsub("}", ",", amenities_list_string, fixed = T)
# Quito las comillas
amenities_list_string <- gsub('"', '', amenities_list_string, fixed=T)
# Separo bien los elementos
amenities_list_string <- gsub(',', ', ', amenities_list_string, fixed=T)

# Función para quitar las palabras repetidas del string
rem_dup_word <- function(x){
  x <- tolower(x)
  paste(unique(trimws(unlist(strsplit(x,split=" ",fixed=F,perl=T)))),collapse = " ")
}
# Quito las palabras repetidas
amenities_list_string <- rem_dup_word(amenities_list_string)
# Convierto el string en lista
amenities_set <- strsplit(amenities_list_string, ",")

# Creación de variables binarias para cada comodidad elegida
datos$AC <- grepl("Air conditioning", datos$amenities, fixed=TRUE)
datos$elevator <- grepl("Elevator", datos$amenities, fixed=TRUE)
datos$pool <- grepl("Pool", datos$amenities, fixed=TRUE)
datos$checkin24h <- grepl("24-hour check-in", datos$amenities, fixed=TRUE)
datos$host_greeting <- grepl("Host greets you", datos$amenities, fixed=TRUE)
datos$balcony <- grepl("balcony", datos$amenities, fixed=TRUE)
datos$fireplace <- grepl("fireplace", datos$amenities, fixed=TRUE)
datos$bbq <- grepl("BBQ", datos$amenities, fixed=TRUE)
datos$smoking_allowed <- grepl("Smoking allowed", datos$amenities, fixed=TRUE)
datos$internet <- grepl("Wifi|Internet|Ethernet", datos$amenities, ignore.case = TRUE)
datos$kid_friendly <- grepl("kid friendly|toys|crib", datos$amenities, ignore.case = TRUE)
datos$essentials <- grepl("essentials|shampoo|soap|towel|paper", datos$amenities, ignore.case = TRUE)
datos$small_appliances <- grepl("hair dryer|microwave|iron|coffee maker|kettle|espresso",
datos$amenities, ignore.case = TRUE)
datos$parking <- grepl("free parking|street parking", datos$amenities, ignore.case = TRUE)
datos$white_goods <- grepl("oven|refrigerator|washer|dishwash", datos$amenities, ignore.case = TRUE)
```

```
datos$accesible <- grepl("no stairs|wheelchair|single level|step-free", datos$amenities, ignore.case =
TRUE)
datos$pets_allowed <- grepl("dog|pet", datos$amenities, ignore.case = TRUE)
datos$garden <- grepl("garden|outdoor seating", datos$amenities, ignore.case = TRUE)
datos$gym <- grepl("gym|exercise equipment", datos$amenities, ignore.case = TRUE)
datos$great_views <- grepl("beachfront|lake|waterfront|view", datos$amenities, ignore.case = TRUE)

#Nos fijamos en las proporciones de TRUE y FALSE
summary(datos[45:64])
names = c("AC", "ascensor", "piscina", "checkin24h", "anfitrión_recibe", "balcón", "chimenea", "bbq",
"permite_fumar", "internet", "apto_niños", "esenciales", "pequeños_electro", "parking",
"grandes_electro", "accesible", "permite_mascotas", "jardín", "gym", "buenas_vistas")
par(mfrow=c(2,4))
j = 1
for (i in 45:64){
  barplot(prop.table(table(datos[i])), main = names[j], col = "lightblue")
  j = j + 1
}

# Descartamos aquellas con menos del 5% de observaciones en una categoría
datosAm <- datos[,-c(4,41,45,46,47,49,51,57,58,59,60)]
summary(datosAm)
```

8.2 Código creación de la variable *accesibilidad* en Python.

```
### ---- Accesibilidad ---- ###
import pandas as pd
tfm_df = pd.read_excel('C:/Users/lcn97/OneDrive/Documentos/DatosSelVar.xlsx')
# Foursquare Credentials para el API
CLIENT_ID = '******************'
CLIENT_SECRET = '*****************'
VERSION = '20200501' # Fecha en la que se realiza la búsqueda
LIMIT = 10 # Límite de lugares más cercanos a buscar por alojamiento

import requests
# function to loop through listings and get venues
def getNearbyVenues(names, latitudes, longitudes, radius):
    venues_list=[]
    for name, lat, lng in zip(names, latitudes, longitudes):
        print(name)
        # create the API request URL
        url =
'https://api.foursquare.com/v2/venues/explore?&client_id={}&client_secret={}&v={}&ll={},{}&radius
={}&limit={}'
        .format(
            CLIENT_ID,
            CLIENT_SECRET,
            VERSION,
            lat,
            lng,
            radius,
            LIMIT)
        # make the GET request
        results = requests.get(url).json()['response']['groups'][0]['items']
         # return only relevant information for each nearby venue
        venues_list.append([(
            name,
```

```
            lat,
            lng,
            v['venue']['name'],
            v['venue']['location']['lat'],
            v['venue']['location']['lng'],
            v['venue']['categories'][0]['name']) for v in results])

    nearby_venues = pd.DataFrame([item for venue_list in venues_list for item in venue_list])
    nearby_venues.columns = ['ID',
                'Listing Latitude',
                'Listing Longitude',
                'Venue',
                'Venue Latitude',
                'Venue Longitude',
                'Venue Category']
    return(nearby_venues)

# Cojo los 10 POI's más cercanos a cada apartamento poniendo de máximo 1km
# El servicio gratuito del API solo permite 950 requests al día y tengo 16632 observaciones
# así que me toca hacerlo por partes y en varios días
madrid_venues_tfm1 = getNearbyVenues(names=tfm_df['id'][0:949],
                             latitudes=tfm_df['latitude'][0:949],
                             longitudes=tfm_df['longitude'][0:949],
                            radius = 1000)
# Guardo en csv
madrid_venues_tfm1.to_csv('C:/Users/lcn97/Documents/TFM/Madrid_Venues_tfm1.csv')

# Una vez hecho para todos los alojamientos, uno todos los archivos
frames = [madrid_venues_tfm1, madrid_venues_tfm2, madrid_venues_tfm3, madrid_venues_tfm4,
madrid_venues_tfm5, madrid_venues_tfm6, madrid_venues_tfm7, madrid_venues_tfm8,
madrid_venues_tfm9, madrid_venues_tfm10, madrid_venues_tfm11, madrid_venues_tfm12,
madrid_venues_tfm13, madrid_venues_tfm14, madrid_venues_tfm15, madrid_venues_tfm16,
madrid_venues_tfm17, madrid_venues_tfm18]
madrid_venues_tfm0 = pd.concat(frames)
madrid_venues_tfm0.to_csv('C:/Users/lcn97/Documents/TFM/Madrid_Venues_tfm0.csv')

# Categorías de sitios recogidos
len(madrid_venues_tfm0['Venue Category'].unique()) #249 tipos de categorías para los lugares
madrid_venues_tfm0['Venue Category'].value_counts()[:30]
# Agrupamos categorías parecidas
freq_Venues = madrid_venues_tfm0.copy()
freq_Venues.loc[freq_Venues['Venue Category'].str.contains('Restaurant|Bistro|fast
food|pizza|bbq|sandwich|burger', case=False), 'Venue Category'] = 'Restaurant'
freq_Venues.loc[freq_Venues['Venue Category'].str.contains('Bar|brewery', case=False), 'Venue
Category'] = 'Bar'
freq_Venues.loc[freq_Venues['Venue
Category'].str.contains('Market|Supermarket|grocery|food|gourmet|convenience', case=False), 'Venue
Category'] = 'Grocery store'
freq_Venues.loc[freq_Venues['Venue Category'].str.contains('Hotel|Hostel', case=False), 'Venue
Category'] = 'Hotel'
freq_Venues.loc[freq_Venues['Venue Category'].str.contains('Café|Bakery|coffee|breakfast', case=False),
'Venue Category'] = 'Bar'
freq_Venues.loc[freq_Venues['Venue Category'].str.contains('gym', case=False), 'Venue Category'] =
'Gym'
freq_Venues.loc[freq_Venues['Venue Category'].str.contains('Theater', case=False), 'Venue Category'] =
'Theater'
freq_Venues.loc[freq_Venues['Venue Category'].str.contains('park|garden', case=False), 'Venue
Category'] = 'Park'
```

```
freq_Venues.loc[freq_Venues['Venue Category'].str.contains('Museum', case=False), 'Venue Category']
= 'Museum'
freq_Venues.loc[freq_Venues['Venue Category'].str.contains('pub|nightlife|club', case=False), 'Venue
Category'] = 'Pub'
freq_Venues.loc[freq_Venues['Venue Category'].str.contains('art', case=False), 'Venue Category'] = 'Art
Gallery'
freq_Venues.loc[freq_Venues['Venue Category'].str.contains('historic|landmark', case=False), 'Venue
Category'] = 'Monument'

freq_Venues['Venue Category'].value_counts()[:10]
# Las 5 categorías más comunes son bar, supermercado, parque, plaza y restaurante
# Nos quedamos con estas 5 categorías
terms = ['Bar', 'Grocery Store', 'Park', 'Plaza', 'Restaurant']
df_pois = freq_Venues[freq_Venues['Venue Category'].str.contains('|'.join(terms))]
df_pois.reset_index(inplace=True)
df_pois.drop(columns=['index'],inplace=True)

# Calculamos la distancia de cada alojamiento a sus POIs
import geopy.distance
df_pois.columns
df_pois['Distance'] = ""
for i in range(0,104009):
    coords_1 = (df_pois['Listing Latitude'][i], df_pois['Listing Longitude'][i])
    coords_2 = (df_pois['Venue Latitude'][i], df_pois['Venue Longitude'][i])
    df_pois['Distance'][i] = geopy.distance.distance(coords_1, coords_2).km
df_pois.to_csv('C:/Users/lcn97/Documents/TFM/4quare_amenities.csv', index=id, header=True)

# Creamos otro dataframe en el que para alojamiento se guardan las distancias a los 5 pois más cercanos
# en caso de que no llegue a 5 pois, se añade 1.5 en los faltantes como penalización
pois5 =  pd.DataFrame(columns=('Listing','Distance'))
namesdis = list(df_pois['Listing'].unique())
pois5['Listing'] = namesdis
for i in range(0,len(namesdis)):
    dist5 = np.sort(df_pois['Distance'].loc[df_pois['Listing']==namesdis[i]])[:5]
    if len(dist5) == 5:
        pois5['Distance'][i] = dist5
    elif len(dist5) < 5:
        l = [1.5] * (5-len(dist5))
        dist5=list(dist5)
        dist5.extend(l)
        pois5['Distance'][i] = dist5

pois5['Avg_distance'] = ""
for i in range(0,len(namesdis)):
    pois5['Avg_distance'][i] = np.mean(pois5['Distance'][i])
pois5['Avg distance'].hist(figsize=(10,11))

# Añadimos la info a nuestra base de datos y guardamos
tfm_df_merged = pd.merge(tfm_df, pois5, on='id', how='left')
tfm_df_merged.to_excel('C:/Users/lcn97/OneDrive/Documentos/TFM/DatosAvg4square.xlsx', index=id,
header=True)
```

8.3 Código creación mapas en R.

```
### ----- Mapas ----- ###
# -- Preparación datos -- #
library(sas7bdat)
library(spdep)
```

```
library(maptools)
library(geojsonio)
library(leaflet)
# Cargamos los datos
datosMap <- read.sas7bdat('datosmapas_train.sas7bdat')
summary(datosMap)

# Cargamos mapa distritos de Madrid como objeto espacial
Distritos <- geojson_read('distritos.geojson', what = "sp")
levels(datosMap$Neighbourhood)
# Hacer bucle con los niveles en este orden
names <- c("Fuencarral - El Pardo", "Moncloa - Aravaca", "Villa de Vallecas", "Vicalvaro", "Barajas",
"Hortaleza", "Retiro", "Moratalaz", "Tetuan","Centro", "Chamberi", "Chamartin", "Salamanca",
"Arganzuela", "Carabanchel", "Latina", "Usera", "Puente de Vallecas", "Ciudad Lineal", "San Blas -
Canillejas", "Villaverde")
atrdistr <- matrix(0, nrow=21, ncol = 2)

j <- 1
for(i in names){
  cat(paste0(i,"\n"))
  atrdistr[j,1] <- sum(datosMap$Neighbourhood==i)
  atrdistr[j,2] <- mean(datosMap[datosMap$Neighbourhood==i,]$precios)
  j <- j + 1
}
row.names(atrdistr) <- names
colnames(atrdistr)  <- c("listings","avgprice", "avgpeople", "avgpriceperperson")
atrdistr <- as.data.frame(atrdistr)

# Convertimos los datos en objetos espaciales a través de las coordenadas
data <- st_as_sf(datosMap, coords= c("longitude", "latitude"))
class(data)

# -- Mapas de colores -- #
require(RColorBrewer)

# Mapa de color de precio medio de airbnbs en cada distrito
qpal1<-colorQuantile("OrRd", atrdistr$avgprice, n=9)
leaflet(Distritos) %>%
  addPolygons(stroke = FALSE, fillOpacity = .8, smoothFactor = 0.2, color = ~qpal1(atrdistr$avgprice))
%>%  addTiles() %>%
  addLegend("bottomright", pal = qpal1, values = ~atrdistr$avgprice,
            title = "Average price (2019)", opacity = 1)

# Mapa de color de nº alojamientos
qpal4<-colorQuantile("OrRd", atrdistr$listings, n=9)
leaflet(Distritos) %>%
  addPolygons(stroke = FALSE, fillOpacity = .8, smoothFactor = 0.2, color = ~qpal4(atrdistr$listings))
%>% addTiles() %>%
  addLegend("bottomright", pal = qpal4, values = ~atrdistr$listings,
            title = "Number of listings(2019)", opacity = 1)

# -- Mapa de densidad -- #
library("sp")
library("rgdal")
library("KernSmooth")
library("raster")
```

```
kde <- bkde2D(datos[ , c("longitude", "IMP_REP_latitude")],bandwidth=c(.0039247, .003114062),
gridsize = c(1000,1000))

# Create Raster from Kernel Density output
KernelDensityRaster <- raster(list(x=kde$x1 ,y=kde$x2 ,z = kde$fhat))
# Set low density cells as NA so we can make them transparent
KernelDensityRaster@data@values[which(KernelDensityRaster@data@values < 1)] <- NA
# create pal function for coloring the raster
palRaster <- colorBin("Spectral", bins = 11, domain = KernelDensityRaster@data@values, na.color =
"transparent")
# Leaflet map with raster
leaflet(Distritos) %>% addTiles() %>%
  addPolygons(stroke = TRUE, fillOpacity = .25, smoothFactor = 0.2, color = "grey") %>%
  addRasterImage(KernelDensityRaster, colors = palRaster, opacity = .8) %>%
  addLegend(pal = palRaster, values = KernelDensityRaster@data@values,
            title = "Airbnb Listings Density (2019)")
```

8.4 Código creación gráficos en Python.

```
### ---- Gráficos ---- ###
import pandas as pd
tfm_df = pd.read_excel('C:/Users/lcn97/OneDrive/Documentos/DatosTFM.xlsx')
pd.set_option('display.max_columns', len(tfm_df.columns))
pd.set_option('display.max_rows', 100)
tfm_df.head(3)
tfm_df.columns

%matplotlib inline  # Para que muestre los gráficos en pantalla
import matplotlib.pyplot as plt

# Gráfico línea anfitriones uniéndose a Airbnb y alojamientos recibiendo su primera reseña
plt.figure(figsize=(15,5))
tfm_df.set_index('f_host_since').resample('MS').size().plot(label='Anfitriones uniéndose a Airbnb',
color='orange')
tfm_df.set_index('f_first_review').resample('MS').size().plot(label='Alojamientos recibiendo su primera
reseña', color='red')
plt.title('Anfitriones en Madrid uniéndose a Airbnb y alojamientos recibiendo su primera reseña')
plt.legend()
plt.xlabel('')
plt.ylabel('')

# Creación de dataframes para series temporales
ts_host_since = pd.DataFrame(tfm_df.set_index('f_host_since').resample('MS').size())
ts_first_review = pd.DataFrame(tfm_df.set_index('f_first_review').resample('MS').size())
ts_host_since = ts_host_since.rename(columns={0: 'hosts'})
ts_host_since.index.rename('month', inplace=True)
ts_first_review = ts_first_review.rename(columns={0: 'reviews'})
ts_first_review.index.rename('month', inplace=True)

# Función para descomponer serie temporal
from statsmodels.tsa.seasonal import seasonal_decompose
def decompose_time_series(df, title=''):
    # Decomposing the time series
    decomposition = seasonal_decompose(df)
    # Getting the trend, seasonality and noise
    trend = decomposition.trend
    seasonal = decomposition.seasonal
```

```
    residual = decomposition.resid
    # Plotting the original time series and the decomposition
    plt.figure(figsize=(12,8))
    plt.suptitle(title, fontsize=14, y=1)
    plt.subplots_adjust(top=0.80)
    plt.subplot(411)
    plt.plot(df, label='Original')
    plt.legend(loc='upper left')
    plt.subplot(412)
    plt.plot(trend, label='Tendencia')
    plt.legend(loc='upper left')
    plt.subplot(413)
    plt.plot(seasonal,label='Estacionalidad')
    plt.legend(loc='upper left')
    plt.subplot(414)
    plt.plot(residual, label='Residuos')
    plt.legend(loc='upper left')
    plt.tight_layout()

# Usamos función en el nº de anfitriones y las primeras reseñas
decompose_time_series(ts_host_since, title='Nº anfitriones uniéndose a Airbnb cada mes')
decompose_time_series(ts_first_review, title='Nº de alojamientos recibiendo su primera reseña cada
mes')

import numpy as np
import seaborn as sns
# Diagrama de cajas número alojamientos por propietario
plt.figure(figsize=(16,6))
sns.boxplot(tfm_df.f_host_since.dt.year, np.log(tfm_df.host_total_listings_count))
plt.xlabel('Año en el que el anfitrión se unió a Airbnb', fontsize=12)
plt.ylabel('Nº de alojamientos por anfitrión (log-transformado)', fontsize=12)
plt.title('Cambio anual en el nº de alojamiento por anfitrión en Airbnb en Madrid', fontsize=16)
# Diagrama de cajas precio por noche
plt.figure(figsize=(16,6))
sns.boxplot(tfm_df.f_first_review.dt.year, np.log(tfm_df.price))
plt.xlabel('Año en el que el alojamiento obtuvo su primera reseña', fontsize=12)
plt.ylabel('Precio por noche (log-transformado)', fontsize=12)
plt.title('Cambio anual en el precio por noche de alojamientos Airbnb en Madrid', fontsize=16)

# Precio medio de alojamientos cada año
print(round(tfm_df.set_index('f_first_review').price.resample('YS').mean(),2))
# Mediana del precio de alojamientos cada año
print(round(tfm_df.set_index('f_first_review').price.resample('YS').median(),2))
```

8.5 Código selección de variables en SAS.

```
libname discoc 'C:\Users\lcn97\OneDrive\Documentos\TFM\depuracion_listings';
data uno;set discoc.selvarfinal_train;run;
proc freq data=uno;run;
proc contents data=uno out=sal;run;quit;
data;set sal;put name @@;run;
options mprint=0;

/* Renombrar variables*/
data uno;
```

```
set discoc.selvarfinal_train (rename= (IMP_REP_REP_noches_max=noches_max
IMP_REP_REP_propiedades=propiedades IMP_REP_banos=banos
IMP_REP_noches_min=noches_min IMP_REP_porcentaje_respuesta=porcentaje_respuesta
IMP_REP_superanfitrion=superanfitrion IMP_REP_verificado=verificado IMP_antiguedad=antiguedad
IMP_puntuacion=puntuacion M_REP_porcentaje_respuesta=M_porcentaje_respuesta
REP_capacidad=capacidad REP_politica_cancelacion=politica_cancelacion
REP_tiempo_respuesta=tiempo_respuesta REP_tipo=tipo precios=precio));
run;
proc contents data=uno out=sal;run;quit;
data;set sal;put name @@;run;
options mprint=0;

/*STEPWISE*/
%randomselect(data=uno, vardepen=precio,
listclass= AC M_porcentaje_respuesta accesible anfitrion_recibe apto_ninos ascensor balcon banos
capacidad checkin24h exactitud grandes_electro parking pequenos_electro permite_fumar
permite_mascotas politica_cancelacion superanfitrion tiempo_respuesta tipo verificado,
modelo=AC M_porcentaje_respuesta accesibilidad accesible aleat anfitrion_recibe antiguedad
apto_ninos ascensor balcon banos capacidad checkin24h demanda dis_airp dis_centro exactitud
grandes_electro noches_max noches_min parking pequenos_electro permite_fumar
permite_mascotas politica_cancelacion porcentaje_respuesta propiedades puntuacion resenas
superanfitrion tiempo_respuesta tipo verificado,selection=stepwise,criterio=AIC,
sinicio=12346, sfinal=12400, fracciontrain=0.8,directorio=C:\Users\lcn97\Documents\PracticaML);

%randomselect(data=uno, vardepen=precio,
listclass= AC M_porcentaje_respuesta accesible anfitrion_recibe apto_ninos ascensor balcon banos
capacidad checkin24h exactitud grandes_electro parking pequenos_electro permite_fumar
permite_mascotas politica_cancelacion superanfitrion tiempo_respuesta tipo verificado,
modelo=AC M_porcentaje_respuesta accesibilidad accesible aleat anfitrion_recibe antiguedad
apto_ninos ascensor balcon banos capacidad checkin24h demanda dis_airp dis_centro exactitud
grandes_electro noches_max noches_min parking pequenos_electro permite_fumar
permite_mascotas politica_cancelacion porcentaje_respuesta propiedades puntuacion resenas
superanfitrion tiempo_respuesta tipo verificado,selection=stepwise,criterio=BIC,
sinicio=12346, sfinal=12400, fracciontrain=0.8,directorio=C:\Users\lcn97\Documents\PracticaML);

%randomselect(data=uno, vardepen=precio,
listclass= AC M_porcentaje_respuesta accesible anfitrion_recibe apto_ninos ascensor balcon banos
capacidad checkin24h exactitud grandes_electro parking pequenos_electro permite_fumar
permite_mascotas politica_cancelacion superanfitrion tiempo_respuesta tipo verificado,
modelo=AC M_porcentaje_respuesta accesibilidad accesible aleat anfitrion_recibe antiguedad
apto_ninos ascensor balcon banos capacidad checkin24h demanda dis_airp dis_centro exactitud
grandes_electro noches_max noches_min parking pequenos_electro permite_fumar
permite_mascotas politica_cancelacion porcentaje_respuesta propiedades puntuacion resenas
superanfitrion tiempo_respuesta tipo verificado,selection=stepwise,criterio=SBC,
sinicio=12346, sfinal=12400, fracciontrain=0.8,directorio=C:\Users\lcn97\Documents\PracticaML);

/*FORWARD*/
%randomselect(data=uno, vardepen=precio,
listclass= AC M_porcentaje_respuesta accesible anfitrion_recibe apto_ninos ascensor balcon banos
capacidad checkin24h exactitud grandes_electro parking pequenos_electro permite_fumar
permite_mascotas politica_cancelacion superanfitrion tiempo_respuesta tipo verificado,
modelo=AC M_porcentaje_respuesta accesibilidad accesible aleat anfitrion_recibe antiguedad
apto_ninos ascensor balcon banos capacidad checkin24h demanda dis_airp dis_centro exactitud
grandes_electro noches_max noches_min parking pequenos_electro permite_fumar
permite_mascotas politica_cancelacion porcentaje_respuesta propiedades puntuacion resenas
superanfitrion tiempo_respuesta tipo verificado,selection=forward,criterio=AIC,
sinicio=12346, sfinal=12400, fracciontrain=0.8,directorio=C:\Users\lcn97\Documents\PracticaML);
```

%***randomselect***(data=uno, vardepen=precio,
listclass= AC M_porcentaje_respuesta accesible anfitrion_recibe apto_ninos ascensor balcon banos capacidad checkin24h exactitud grandes_electro parking pequenos_electro permite_fumar permite_mascotas politica_cancelacion superanfitrion tiempo_respuesta tipo verificado,
modelo=AC M_porcentaje_respuesta accesibilidad accesible aleat anfitrion_recibe antiguedad apto_ninos ascensor balcon banos capacidad checkin24h demanda dis_airp dis_centro exactitud grandes_electro noches_max noches_min parking pequenos_electro permite_fumar permite_mascotas politica_cancelacion porcentaje_respuesta propiedades puntuacion resenas superanfitrion tiempo_respuesta tipo verificado,selection=forward,criterio=BIC,
sinicio=**12346**, sfinal=**12400**, fracciontrain=**0.8**,directorio=C:\Users\lcn97\Documents\PracticaML);

%***randomselect***(data=uno, vardepen=precio,
listclass= AC M_porcentaje_respuesta accesible anfitrion_recibe apto_ninos ascensor balcon banos capacidad checkin24h exactitud grandes_electro parking pequenos_electro permite_fumar permite_mascotas politica_cancelacion superanfitrion tiempo_respuesta tipo verificado,
modelo=AC M_porcentaje_respuesta accesibilidad accesible aleat anfitrion_recibe antiguedad apto_ninos ascensor balcon banos capacidad checkin24h demanda dis_airp dis_centro exactitud grandes_electro noches_max noches_min parking pequenos_electro permite_fumar permite_mascotas politica_cancelacion porcentaje_respuesta propiedades puntuacion resenas superanfitrion tiempo_respuesta tipo verificado,selection=forward,criterio=SBC,
sinicio=**12346**, sfinal=**12400**, fracciontrain=**0.8**,directorio=C:\Users\lcn97\Documents\PracticaML);

/*BACKWARD*/
%***randomselect***(data=uno, vardepen=precio,
listclass= AC M_porcentaje_respuesta accesible anfitrion_recibe apto_ninos ascensor balcon banos capacidad checkin24h exactitud grandes_electro parking pequenos_electro permite_fumar permite_mascotas politica_cancelacion superanfitrion tiempo_respuesta tipo verificado,
modelo=AC M_porcentaje_respuesta accesibilidad accesible aleat anfitrion_recibe antiguedad apto_ninos ascensor balcon banos capacidad checkin24h demanda dis_airp dis_centro exactitud grandes_electro noches_max noches_min parking pequenos_electro permite_fumar permite_mascotas politica_cancelacion porcentaje_respuesta propiedades puntuacion resenas superanfitrion tiempo_respuesta tipo verificado,selection=backward,criterio=AIC,
sinicio=**12346**, sfinal=**12400**, fracciontrain=**0.8**,directorio=C:\Users\lcn97\Documents\PracticaML);

%***randomselect***(data=uno, vardepen=precio,
listclass= AC M_porcentaje_respuesta accesible anfitrion_recibe apto_ninos ascensor balcon banos capacidad checkin24h exactitud grandes_electro parking pequenos_electro permite_fumar permite_mascotas politica_cancelacion superanfitrion tiempo_respuesta tipo verificado,
modelo=AC M_porcentaje_respuesta accesibilidad accesible aleat anfitrion_recibe antiguedad apto_ninos ascensor balcon banos capacidad checkin24h demanda dis_airp dis_centro exactitud grandes_electro noches_max noches_min parking pequenos_electro permite_fumarpermite_mascotas politica_cancelacion porcentaje_respuesta propiedades puntuacion resenas superanfitrion tiempo_respuesta tipo verificado,selection=backward,criterio=BIC,
sinicio=**12346**, sfinal=**12400**, fracciontrain=**0.8**,directorio=C:\Users\lcn97\Documents\PracticaML);

%***randomselect***(data=uno, vardepen=precio,
listclass= AC M_porcentaje_respuesta accesible anfitrion_recibe apto_ninos ascensor balcon banos capacidad checkin24h exactitud grandes_electro parking pequenos_electro permite_fumar permite_mascotas politica_cancelacion superanfitrion tiempo_respuesta tipo verificado,
modelo=AC M_porcentaje_respuesta accesibilidad accesible aleat anfitrion_recibe antiguedad apto_ninos ascensor balcon banos capacidad checkin24h demanda dis_airp dis_centro exactitud grandes_electro noches_max noches_min parking pequenos_electro permite_fumar permite_mascotas politica_cancelacion porcentaje_respuesta propiedades puntuacion resenas superanfitrion tiempo_respuesta tipo verificado,selection=backward,criterio=SBC,
sinicio=**12346**, sfinal=**12400**, fracciontrain=**0.8**,directorio=C:\Users\lcn97\Documents\PracticaML);

8.6 Código estimación modelos hedónicos y análisis espacial en R.

```
### ----- Modelos Hedónicos ----- ###

# -- Preparación datos -- #
library(sas7bdat)
datos <- read.sas7bdat('selvarfinal_train.sas7bdat')
summary(datos)

# Quitamos las variables que no han sido seleccionadas
datos <- datos[,-c(3, 9, 12, 14, 26, 27, 28, 29, 30, 33, 34)]

# Renombramos
colnames(datos)[colnames(datos) == 'precios'] <- 'precio'
colnames(datos)[colnames(datos) == 'REP_capacidad'] <- 'capacidad'
colnames(datos)[colnames(datos) == 'REP_politica_cancelacion'] <- 'politica_cancelacion'
colnames(datos)[colnames(datos) == 'REP_tiempo_respuesta'] <- 'tiempo_respuesta'
colnames(datos)[colnames(datos) == 'REP_tipo'] <- 'tipo'
colnames(datos)[colnames(datos) == 'IMP_REP_banos'] <- 'banos'
colnames(datos)[colnames(datos) == 'IMP_REP_superanfitrion'] <- 'superanfitrion'
colnames(datos)[colnames(datos) == 'IMP_antiguedad'] <- 'antiguedad'
colnames(datos)[colnames(datos) == 'IMP_puntuacion'] <- 'puntuacion'

# Mal puestas: capacidad y baños
datos$capacidad<- as.factor(datos$capacidad)
datos$banos<- as.factor(datos$banos)
summary(datos)

# -- Regresión lineal -- #
library(glmnet)
library(lmSupport)
library(caret)

# Separamos en train y test
set.seed(2611)
partitionIndex <- createDataPartition(datos$precio, p=0.8, list=FALSE)
data_train <- datos[partitionIndex,]
data_test <- datos[-partitionIndex,]

# Regresión
Reg <- lm(precio~., data=data_train)
summary(Reg)
Reg$rank
RMSE(predict(Reg, data_test), data_test$precio)
R2(predict(Reg, data_test), data_test$precio)
modelEffectSizes(Reg)
barplot(sort(abs(Reg[["effects"]][2:30]),decreasing =T),col="light blue",las=2,cex.names=0.65,
main="Importancia de las variables")

# Multicolinealidad
car::vif(Reg)

# -- Análisis espacial -- #
library(spdep)
library(sf)

# Añadimos las columnas de latitud y longitud
latylon <- read.sas7bdat('lat&lon.sas7bdat')
```

```
datosSP <- cbind(datos,latylon)
summary(datosSP)

# Partición
set.seed(2611)
partitionIndex1 <- createDataPartition(datosSP$precio, p=0.8, list=FALSE)
dataSP_train <- datosSP[partitionIndex1,]
dataSP_test <- datosSP[-partitionIndex1,]

# Convertir latitud y longitud en coordenadas
dataSP_train <- st_as_sf(dataSP_train, coords= c("longitude", "latitude"))
# Convertir objeto sf a sp
dataSP_train <- as(dataSP_train, "Spatial")
# Eliminamos observaciones con coordenadas duplicadas porque si no nos saldrán errores
table(duplicated(data.frame(coordinates(dataSP_train)))) # 5 duplicados
dataSP_train <- dataSP_train[!duplicated(data.frame(coordinates(dataSP_train))),]

# OLS normal
abnb.ols<-lm(precio~., data=dataSP_train@data)
summary(abnb.ols)
AIC(abnb.ols)
abnb.ols$rank
RMSE(predict(abnb.ols, dataSP_test), dataSP_test$precio)
R2(predict(abnb.ols, dataSP_test), dataSP_test$precio)

library(deldir)
library(maptools)
library(geojsonio)

# Triangulación
coords <- coordinates(dataSP_train)
nbl = tri2nb(coords)
summary(nbl)
# Mapa distritos de Madrid
Distritos <- geojson_read('distritos.geojson', what = "sp")
madrid <- st_as_sf(Distritos)
# Graficar distribución de la triangulación
plot(st_geometry(madrid), border="dark blue", reset=FALSE)
plot(nbl, coords, add=TRUE)
# Construir matriz de pesos W
W<-nb2listw(nbl, style="W", zero.policy=TRUE) # La opción W estandariza la matriz W

# Moran´s I Test
moran.lm<-lm.morantest(abnb.ols, W, alternative="two.sided")
print(moran.lm)
# p-value << 0.05  -> Rechazar Ho: Datos son especialmente independientes

# Lagrange Multiplier Test
LM<-lm.LMtests(abnb.ols, W, test="all") # test=all para que haga tanto SAR como SEM
print(LM)
# SAR alternativa más probable

# -- Estimación SAR & SEM -- #
library(spatialreg)

id <- row.names(as(dataSP_train, "data.frame"))
coords_train <- coordinates(dataSP_train)
id_test <- row.names(as(dataSP_test, "data.frame"))
```

```
coords_test <- coordinates(dataSP_test)

# Bucle k vecinos de 5 a 60 en 5 en 5
df3<- data.frame(matrix(ncol = 4, nrow = 11))
x <- c("k", "AIC", "RMSE", "R^2")
colnames(df2) <- x
j <- 1
for (kn in seq(5,60,5))
{
  print(kn)
  nb <- knn2nb(knearneigh(coords_train, k = kn), row.names = id)
  W_train <-nb2listw(nb, style="W", zero.policy=TRUE)
  abnb.sar <- lagsarlm(precio~., data=dataSP_train@data, W_train)
  test_nb <- knn2nb(knearneigh(coords_test, k = kn), row.names = id_test)
  W_test <-nb2listw(test_nb, style="W", zero.policy=TRUE)
  df2[j,1] <- kn
  df2[j,2] <- AIC(abnb.sar)
  predic_sar = as.vector(predict(abnb.sar, dataSP_test@data,W_test))
  df2[j,3] <- sqrt(sum((dataSP_test@data$precio - predic_sar)^2)/length(predic_sar))
  df2[j,4] <- 1 - (sum((predic_sar-dataSP_test@data$precio)^2) /sum((dataSP_test@data$precio-
mean(dataSP_test@data$precio))^2))
  j <- j + 1
  print(kn)
}
df3
df3 <- na.omit(df3)

plot(df3$k,df3$AIC,col="red",pch=19,main="AIC según valor de k (k-NN)",xlab="k", ylab="AIC")
plot(df3$k,df3$`R^2`,col="red",pch=19,main="RMSE según valor de k (k-NN)",xlab="k",
ylab="RMSE")
plot(df3$k,df3$RMSE,col="red",pch=19,main="R^2 según valor de k (k-NN)",xlab="k", ylab="R^2")
# k = 35 ganador

# Estimación SAR y SEM con este número
nb <- knn2nb(knearneigh(coords_train, k = 35), row.names = id)
plot(st_geometry(madrid), border="dark blue", reset=FALSE)
plot(nb, coords_train, add=TRUE)
W_train <-nb2listw(nb, style="W", zero.policy=TRUE)

abnb.sar <- lagsarlm(precio~., data=dataSP_train@data, W_train)
summary(abnb.sar)
# RMSE, AIC Y R2 SAR
test_nb <- knn2nb(knearneigh(coords_test, k = 35), row.names = id_test)
W_test <-nb2listw(test_nb, style="W", zero.policy=TRUE)
AIC(abnb.sar)
predic_sar = as.vector(predict(abnb.sar, dataSP_test@data,W_test))
sqrt(sum((dataSP_test@data$precio - predic_sar)^2)/length(predic_sar))
1 - (sum((predic_sar-dataSP_test@data$precio)^2)/sum((dataSP_test@data$precio-
mean(dataSP_test@data$precio))^2))

abnb.sem <- errorsarlm(precio~., data=dataSP_train@data, W_train)
summary(abnb.sem)
# RMSE, AIC Y R2 SEM
AIC(abnb.sem)
predic_sem = as.vector(predict(abnb.sem, dataSP_test@data,W_test))
sqrt(sum((dataSP_test@data$precio - predic_sem)^2)/length(predic_sem))
1 - (sum((predic_sem-dataSP_test@data$precio)^2)/sum((dataSP_test@data$precio-
mean(dataSP_test@data$precio))^2))
```

```
# RMSE Y R2 OLS
RMSE(predict(abnb.ols, dataSP_test@data), dataSP_test@data$precio)
sqrt(sum((dataSP_test@data$precio - as.vector(predict(abnb.ols,
dataSP_test@data)))^2)/length(as.vector(predict(abnb.ols, dataSP_test@data))))
R2(predict(abnb.ols, dataSP_test@data), dataSP_test@data$precio)
1 - (sum((predict(abnb.ols, dataSP_test@data)-dataSP_test@data$precio)^2)
/sum((dataSP_test@data$precio-mean(dataSP_test@data$precio))^2))

# LR test para bondad de ajuste
library(lmtest)
lrtest(abnb.sem, abnb.sar) #SAR mejor
lrtest(abnb.ols, abnb.sar) #SAR mejor

# Ver impactos directos e indirectos modelo SAR
W2 <- as(W_train, "CsparseMatrix")
trMC <- trW(W2, type="MC")
im<-impacts(abnb.sar, tr=trMC, R=100)
sums<-summary(im,  zstats=T)
data.frame(sums$res)

# Significatividad de los impactos directos e indirectos
data.frame(sums$pzmat)
```

8.7 Código estimación algoritmos de Machine Learning en R y SAS.

```
# -- Preparación datos  -- #
library(nnet)
library(h2o)
library(dummies)
library(MASS)
library(reshape)
library(caret)

# Lectura y esquema de variables
dput(names(datos))
continuas<-c("dis_centro", "dis_airp", "demanda", "resenas", "accesibilidad", "antiguedad",
"puntuacion")
categoricas<-c("AC", "ascensor", "anfitrion_recibe", "balcon", "apto_ninos", "parking",
"grandes_electro", "accesible", "permite_mascotas", "capacidad", "politica_cancelacion",
"tiempo_respuesta", "tipo", "banos", "superanfitrion")

# Pasar las categóricas a dummies
datos2<- dummy.data.frame(datos, categoricas, sep = ".")
# Estandarizar las variables continuas
means <-apply(datos2[,continuas],2,mean)
sds<-sapply(datos2[,continuas],sd)
estdata<-scale(datos2[,continuas], center = means, scale = sds) # Continuas estandarizadas
numerocont<-which(colnames(datos2)%in%continuas) # Índices
estdata<-cbind(estdata,datos2[,-numerocont]) # Unir las estandarizadas a las dummies

# El archivo estdata ya está preparado
dput(names(estdata))

# Renombramos
colnames(estdata)[colnames(estdata) ==  "tiempo_respuesta.en unas horas"] <-
"tiempo_respuesta.en_unas_horas"
```

```
colnames(estdata)[colnames(estdata) == "tiempo_respuesta.en una hora"] <-
"tiempo_respuesta.en_una_hora"
colnames(estdata)[colnames(estdata) == "tiempo_respuesta.un dia o mas"] <-
"tiempo_respuesta.un_dia_o_mas"
colnames(estdata)[colnames(estdata) == "tipo.Alojamiento entero"] <- "tipo.Alojamiento_entero"

listconti=c("dis_centro", "dis_airp", "demanda", "resenas", "accesibilidad", "antiguedad", "puntuacion",
"AC.T", "ascensor.T", "anfitrion_recibe.T",  "balcon.T", "apto_ninos.T", "parking.T",
"grandes_electro.T",  "accesible.T", "permite_mascotas.T",  "capacidad.2",
"capacidad.3", "capacidad.4", "capacidad.5","politica_cancelacion.flexible",
"politica_cancelacion.moderada", "tiempo_respuesta.en_unas_horas", "tiempo_respuesta.NA",
"tiempo_respuesta.un_dia_o_mas", "tipo.Habitacion",  "banos.2","banos.3","superanfitrion.T")
# Quitamos dummies que no usaremos en los modelos
estdata <- estdata[,-c(9,11,13,15,17,19,21,23,25,26,31,34,38,40,44)]
```

8.7.1 Redes neuronales en SAS.

```
/*----REDES ----*/
/*algoritmos y numero de nodos con vc de 4 grupos y 15 semillas*/
/* 4 nodos levmar y luego bprop */
%cruzadaneural(archivo=uno,vardepen=precio, conti=accesibilidad antiguedad demanda dis_airp
dis_centro puntuacion resenas, categor=AC accesible anfitrion_recibe apto_ninos ascensor balcon
banos capacidad grandes_electro parking permite_mascotas politica_cancelacion superanfitrion
tiempo_respuesta tipo,ngrupos=4,sinicio=12345,sfinal=12359,nodos=4,early=,algo=levmar,
directorio=C:\Users\lcn97\Documents\TFM);
data final1;set final;modelo='Lev-04';

%cruzadaneural(archivo=uno,vardepen=precio,conti=accesibilidad antiguedad demanda dis_airp
dis_centro puntuacion resenas,categor=AC accesible anfitrion_recibe apto_ninos ascensor balcon banos
capacidad grandes_electro parking permite_mascotas politica_cancelacion superanfitrion
tiempo_respuesta tipo,ngrupos=4,sinicio=12345, sfinal=12359,nodos=4,early=,algo=bprop mom=0.2
learn=0.1, directorio=C:\Users\lcn97\Documents\TFM);
data final2;set final;modelo='Bpr-04';

/* 6 nodos levmar y luego bprop */
%cruzadaneural(archivo=uno,vardepen=precio,conti=accesibilidad antiguedad demanda dis_airp
dis_centro puntuacion resenas,categor=AC accesible anfitrion_recibe apto_ninos ascensor balcon banos
capacidad grandes_electro parking permite_mascotas politica_cancelacion superanfitrion
tiempo_respuesta tipo, ngrupos=4,sinicio=12345, sfinal=12359,nodos=6,early=,algo=levmar,
directorio=C:\Users\lcn97\Documents\TFM);
data final3;set final;modelo='Lev-06';

%cruzadaneural(archivo=uno,vardepen=precio,conti=accesibilidad antiguedad demanda dis_airp
dis_centro puntuacion resenas,categor=AC accesible anfitrion_recibe apto_ninos ascensor balcon banos
capacidad grandes_electro parking permite_mascotas politica_cancelacion superanfitrion
tiempo_respuesta tipo,ngrupos=4,sinicio=12345, sfinal=12359,nodos=6,early=,algo=bprop mom=0.2
learn=0.1, directorio=C:\Users\lcn97\Documents\TFM);
data final4;set final;modelo='Bpr-06';

/* 8 nodos levmar y luego bprop */
%cruzadaneural(archivo=uno,vardepen=precio,conti=accesibilidad antiguedad demanda dis_airp
dis_centro puntuacion resenas,categor=AC accesible anfitrion_recibe apto_ninos ascensor balcon banos
capacidad grandes_electro parking permite_mascotas politica_cancelacion superanfitrion
tiempo_respuesta tipo,ngrupos=4,sinicio=12345, sfinal=12359,nodos=8,early=,algo=levmar,
directorio=C:\Users\lcn97\Documents\TFM);
data final5;set final;modelo='Lev-08';
```

```
%cruzadaneural(archivo=uno,vardepen=precio,conti=accesibilidad antiguedad demanda dis_airp
dis_centro puntuacion resenas,categor=AC accesible anfitrion_recibe apto_ninos ascensor balcon banos
capacidad grandes_electro parking permite_mascotas politica_cancelacion superanfitrion
tiempo_respuesta tipo,ngrupos=4,sinicio=12345, sfinal=12359,nodos=8,early=,algo=bprop mom=0.2
learn=0.1, directorio=C:\Users\lcn97\Documents\TFM);
data final6;set final;modelo='Bpr-08';
/* 10 nodos levmar y luego bprop */
%cruzadaneural(archivo=uno,vardepen=precio,conti=accesibilidad antiguedad demanda dis_airp
dis_centro puntuacion resenas,categor=AC accesible anfitrion_recibe apto_ninos ascensor balcon banos
capacidad grandes_electro parking permite_mascotas politica_cancelacion superanfitrion
tiempo_respuesta tipo,ngrupos=4,sinicio=12345, sfinal=12359,nodos=10,early=,algo=levmar,
directorio=C:\Users\lcn97\Documents\tfm);
data final7;set final;modelo='Lev-10';

%cruzadaneural(archivo=uno,vardepen=precio,conti=accesibilidad antiguedad demanda dis_airp
dis_centro puntuacion resenas,categor=AC accesible anfitrion_recibe apto_ninos ascensor balcon banos
capacidad grandes_electro parking permite_mascotas politica_cancelacion superanfitrion
tiempo_respuesta tipo,ngrupos=4,sinicio=12345, sfinal=12359,nodos=10,early=,algo=bprop mom=0.2
learn=0.1, directorio=C:\Users\lcn97\Documents\TFM);
data final8;set final;modelo='Bpr-10';
/* 12 nodos levmar y luego bprop */
%cruzadaneural(archivo=uno,vardepen=precio,conti=accesibilidad antiguedad demanda dis_airp
dis_centro puntuacion resenas,categor=AC accesible anfitrion_recibe apto_ninos ascensor balcon banos
capacidad grandes_electro parking permite_mascotas politica_cancelacion superanfitrion
tiempo_respuesta tipo,ngrupos=4,sinicio=12345, sfinal=12359,nodos=12,early=,algo=levmar,
directorio=C:\Users\lcn97\Documents\TFM);
data final9;set final;modelo='Lev-12';

%cruzadaneural(archivo=uno,vardepen=precio,conti=accesibilidad antiguedad demanda dis_airp
dis_centro puntuacion resenas,categor=AC accesible anfitrion_recibe apto_ninos ascensor balcon banos
capacidad grandes_electro parking permite_mascotas politica_cancelacion superanfitrion
tiempo_respuesta tipo,ngrupos=4,sinicio=12345, sfinal=12359,nodos=12,early=,algo=bprop mom=0.2
learn=0.1, directorio=C:\Users\lcn97\Documents\TFM);
data final10;set final;modelo='Bpr-12';
/* 14 nodos levmar y luego bprop */
%cruzadaneural(archivo=uno,vardepen=precio,conti=accesibilidad antiguedad demanda dis_airp
dis_centro puntuacion resenas,categor=AC accesible anfitrion_recibe apto_ninos ascensor balcon banos
capacidad grandes_electro parking permite_mascotas politica_cancelacion superanfitrion
tiempo_respuesta tipo,ngrupos=4,sinicio=12345, sfinal=12359,nodos=14,early=,algo=levmar,
directorio=C:\Users\lcn97\Documents\TFM);
data final11;set final;modelo='Lev-14';

%cruzadaneural(archivo=uno,vardepen=precio,conti=accesibilidad antiguedad demanda dis_airp
dis_centro puntuacion resenas,categor=AC accesible anfitrion_recibe apto_ninos ascensor balcon banos
capacidad grandes_electro parking permite_mascotas politica_cancelacion superanfitrion
tiempo_respuesta tipo,ngrupos=4,sinicio=12345, sfinal=12359,nodos=14,early=,algo=bprop mom=0.2
learn=0.1, directorio=C:\Users\lcn97\Documents\TFM);
data final12;set final;modelo='Bpr-14';
/* 16 nodos levmar y luego bprop */
%cruzadaneural(archivo=uno,vardepen=precio,conti=accesibilidad antiguedad demanda dis_airp
dis_centro puntuacion resenas,categor=AC accesible anfitrion_recibe apto_ninos ascensor balcon banos
capacidad grandes_electro parking permite_mascotas politica_cancelacion superanfitrion
tiempo_respuesta tipo,ngrupos=4,sinicio=12345, sfinal=12359,nodos=16,early=,algo=levmar,
directorio=C:\Users\lcn97\Documents\TFM);
data final13;set final;modelo='Lev-16';
```

```
%cruzadaneural(archivo=uno,vardepen=precio,conti=accesibilidad antiguedad demanda dis_airp
dis_centro puntuacion resenas,categor=AC accesible anfitrion_recibe apto_ninos ascensor balcon banos
capacidad grandes_electro parking permite_mascotas politica_cancelacion superanfitrion
tiempo_respuesta tipo,ngrupos=4,sinicio=12345, sfinal=12359,nodos=16,early=,algo=bprop mom=0.2
learn=0.1, directorio=C:\Users\lcn97\Documents\TFM);
data final14;set final;modelo='Bpr-16';
/* 18 nodos levmar y luego bprop */
%cruzadaneural(archivo=uno,vardepen=precio,conti=accesibilidad antiguedad demanda dis_airp
dis_centro puntuacion resenas,categor=AC accesible anfitrion_recibe apto_ninos ascensor balcon banos
capacidad grandes_electro parking permite_mascotas politica_cancelacion superanfitrion
tiempo_respuesta tipo,ngrupos=4,sinicio=12345, sfinal=12359,nodos=18,early=,algo=levmar,
directorio=C:\Users\lcn97\Documents\TFM);
data final15;set final;modelo='Lev-18';

%cruzadaneural(archivo=uno,vardepen=precio,conti=accesibilidad antiguedad demanda dis_airp
dis_centro puntuacion resenas,categor=AC accesible anfitrion_recibe apto_ninos ascensor balcon banos
capacidad grandes_electro parking permite_mascotas politica_cancelacion superanfitrion
tiempo_respuesta tipo,ngrupos=4,sinicio=12345, sfinal=12359,nodos=18,early=,algo=bprop mom=0.2
learn=0.1, directorio=C:\Users\lcn97\Documents\TFM);
data final16;set final;modelo='Bpr-18';
ods graphics off;
data union;set final1 final2 final3 final4 final5 final6 final7 final8 final9 final10 final11 final12
final13 final14 final15 final16 ;
proc boxplot data=union;plot media*modelo;run;
%cruzadaneural(archivo=uno,vardepen=precio,conti=accesibilidad antiguedad demanda dis_airp
dis_centro puntuacion resenas,categor=AC accesible anfitrion_recibe apto_ninos ascensor balcon banos
capacidad grandes_electro parking permite_mascotas politica_cancelacion superanfitrion
tiempo_respuesta tipo,ngrupos=4,sinicio=12345, sfinal=12359,nodos=20,early=,algo=bprop mom=0.2
learn=0.1, directorio=C:\Users\lcn97\Documents\TFM);
data final17;set final;modelo='Bpr-20';
/* LEVMAR 8 NODOS; BPROP 16 */
/* CAMBIAR ESPECIFICACIONES BPROP */
%cruzadaneural(archivo=uno,vardepen=precio,conti=accesibilidad antiguedad demanda dis_airp
dis_centro puntuacion resenas,categor=AC accesible anfitrion_recibe apto_ninos ascensor balcon banos
capacidad grandes_electro parking permite_mascotas politica_cancelacion superanfitrion
tiempo_respuesta tipo,ngrupos=4,sinicio=12345 ,sfinal=12354,nodos=16,early=,algo=bprop mom=0.2
learn=0.1, directorio=C:\Users\lcn97\Documents\TFM);
data final18;set final;modelo='Bpr1';

%cruzadaneural(archivo=uno,vardepen=precio,conti=accesibilidad antiguedad demanda dis_airp
dis_centro puntuacion resenas,categor=AC accesible anfitrion_recibe apto_ninos ascensor balcon banos
capacidad grandes_electro parking permite_mascotas politica_cancelacion superanfitrion
tiempo_respuesta tipo,ngrupos=4,sinicio=12345, sfinal=12354,nodos=16,early=,algo=bprop mom=0.2
learn=0.05, directorio=C:\Users\lcn97\Documents\TFM);
data final19;set final;modelo='Bpr2';

%cruzadaneural(archivo=uno,vardepen=precio,conti=accesibilidad antiguedad demanda dis_airp
dis_centro puntuacion resenas,categor=AC accesible anfitrion_recibe apto_ninos ascensor balcon banos
capacidad grandes_electro parking permite_mascotas politica_cancelacion superanfitrion
tiempo_respuesta tipo,ngrupos=4,sinicio=12345, sfinal=12354,nodos=16,early=,algo=bprop mom=0.2
learn=0.02, directorio=C:\Users\lcn97\Documents\TFM);
data final20;set final;modelo='Bpr3';

%cruzadaneural(archivo=uno,vardepen=precio,conti=accesibilidad antiguedad demanda dis_airp
dis_centro puntuacion resenas,categor=AC accesible anfitrion_recibe apto_ninos ascensor balcon banos
capacidad grandes_electro parking permite_mascotas politica_cancelacion superanfitrion
```

```
tiempo_respuesta tipo,ngrupos=4,sinicio=12345, sfinal=12354,nodos=16,early=,algo=bprop mom=0.1
learn=0.1, directorio=C:\Users\lcn97\Documents\TFM);
data final21;set final;modelo='Bpr4';

%cruzadaneural(archivo=uno,vardepen=precio,conti=accesibilidad antiguedad demanda dis_airp
dis_centro puntuacion resenas,categor=AC accesible anfitrion_recibe apto_ninos ascensor balcon banos
capacidad grandes_electro parking permite_mascotas politica_cancelacion superanfitrion
tiempo_respuesta tipo,ngrupos=4,sinicio=12345, sfinal=12354,nodos=16,early=,algo=bprop mom=0.1
learn=0.05, directorio=C:\Users\lcn97\Documents\TFM);
data final22;set final;modelo='Bpr5';

%cruzadaneural(archivo=uno,vardepen=precio,conti=accesibilidad antiguedad demanda dis_airp
dis_centro puntuacion resenas,categor=AC accesible anfitrion_recibe apto_ninos ascensor balcon banos
capacidad grandes_electro parking permite_mascotas politica_cancelacion superanfitrion
tiempo_respuesta tipo,ngrupos=4,sinicio=12345, sfinal=12354,nodos=16,early=,algo=bprop mom=0.1
learn=0.02, directorio=C:\Users\lcn97\Documents\TFM);
data final23;set final;modelo='Bpr6';

%cruzadaneural(archivo=uno,vardepen=precio,conti=accesibilidad antiguedad demanda dis_airp
dis_centro puntuacion resenas,categor=AC accesible anfitrion_recibe apto_ninos ascensor balcon banos
capacidad grandes_electro parking permite_mascotas politica_cancelacion superanfitrion
tiempo_respuesta tipo,ngrupos=4,sinicio=12345, sfinal=12354,nodos=16,early=,algo=bprop mom=0.2
learn=0.01, directorio=C:\Users\lcn97\Documents\TFM);
data final24;set final;modelo='Bpr7';

%cruzadaneural(archivo=uno,vardepen=precio,conti=accesibilidad antiguedad demanda dis_airp
dis_centro puntuacion resenas,categor=AC accesible anfitrion_recibe apto_ninos ascensor balcon banos
capacidad grandes_electro parking permite_mascotas politica_cancelacion superanfitrion
tiempo_respuesta tipo,ngrupos=4,sinicio=12345, sfinal=12354,nodos=16,early=,algo=bprop mom=0.1
learn=0.01, directorio=C:\Users\lcn97\Documents\TFM);
data final25;set final;modelo='Bpr8';

%cruzadaneural(archivo=uno,vardepen=precio,conti=accesibilidad antiguedad demanda dis_airp
dis_centro puntuacion resenas,categor=AC accesible anfitrion_recibe apto_ninos ascensor balcon banos
capacidad grandes_electro parking permite_mascotas politica_cancelacion superanfitrion
tiempo_respuesta tipo,ngrupos=4,sinicio=12345 ,sfinal=12354,nodos=16,early=,algo=bprop mom=0.2
learn=0.005, directorio=C:\Users\lcn97\Documents\TFM);
data final26;set final;modelo='Bpr9';
data union;set final18 final19 final20 final21 final22 final23 final24 final25 final26;
proc boxplot data=union;plot media*modelo;run;
/*mom 0.2 y learn 0.01 - bpr7- final24*/
/*Early stopping bpr7 y lev8*/
%redneuronal(archivo=uno,listclass=AC accesible anfitrion_recibe apto_ninos ascensor balcon banos
capacidad grandes_electro ,listconti=accesibilidad antiguedad demanda dis_airp dis_centro puntuacion
resenas, vardep=precio,porcen=0.80,semilla=471154, ocultos=16, algo=BPROP MOM=0.2
LEARN=0.01,acti=TANH);

%redneuronal(archivo=uno,listclass=AC accesible anfitrion_recibe apto_ninos ascensor balcon banos
capacidad grandes_electro ,listconti=accesibilidad antiguedad demanda dis_airp dis_centro puntuacion
resenas,vardep=precio,porcen=0.80,semilla=471254,ocultos=16, algo=BPROP MOM=0.2
LEARN=0.01,acti=TANH);

%redneuronal(archivo=uno,listclass=AC accesible anfitrion_recibe apto_ninos ascensor balcon banos
capacidad grandes_electro ,listconti=accesibilidad antiguedad demanda dis_airp dis_centro puntuacion
resenas,vardep=precio,porcen=0.80,semilla=481154,ocultos=16, algo=BPROP MOM=0.2
LEARN=0.01,acti=TANH);
```

```
%redneuronal(archivo=uno,listclass=AC accesible anfitrion_recibe apto_ninos ascensor balcon banos
capacidad grandes_electro ,listconti=accesibilidad antiguedad demanda dis_airp dis_centro puntuacion
resenas,vardep=precio,porcen=0.80,semilla=4714,ocultos=16, algo=BPROP MOM=0.2
LEARN=0.01,acti=TANH);
/*100 -> No early stopping*/
%redneuronal(archivo=uno,listclass=AC accesible anfitrion_recibe apto_ninos ascensor balcon banos
capacidad grandes_electro ,listconti=accesibilidad antiguedad demanda dis_airp dis_centro puntuacion
resenas,vardep=precio,porcen=0.80,semilla=471154, ocultos=8, algo=LEVMAR,acti=TANH);

%redneuronal(archivo=uno,listclass=AC accesible anfitrion_recibe apto_ninos ascensor balcon banos
capacidad grandes_electro ,listconti=accesibilidad antiguedad demanda dis_airp dis_centro puntuacion
resenas,vardep=precio,porcen=0.80,semilla=471254,ocultos=8, algo=LEVMAR,acti=TANH);

%redneuronal(archivo=uno,listclass=AC accesible anfitrion_recibe apto_ninos ascensor balcon banos
capacidad grandes_electro ,listconti=accesibilidad antiguedad demanda dis_airp dis_centro puntuacion
resenas,vardep=precio,porcen=0.80,semilla=481154,ocultos=8, algo=LEVMAR,acti=TANH);

%redneuronal(archivo=uno,listclass=AC accesible anfitrion_recibe apto_ninos ascensor balcon banos
capacidad grandes_electro ,listconti=accesibilidad antiguedad demanda dis_airp dis_centro puntuacion
resenas,vardep=precio,porcen=0.80,semilla=4714, ocultos=8,algo=LEVMAR,acti=TANH);
/*20 aprox*/
/*Comparamos con y sin early Levmar + mejor Bprop*/
%cruzadaneural(archivo=uno,vardepen=precio,conti=accesibilidad antiguedad demanda dis_airp
dis_centro puntuacion resenas,categor=AC accesible anfitrion_recibe apto_ninos ascensor balcon banos
capacidad grandes_electro parking permite_mascotas politica_cancelacion superanfitrion
tiempo_respuesta tipo,ngrupos=4,sinicio=12345, sfinal=12354,nodos=8,early=,algo=levmar,
directorio=C:\Users\lcn97\Documents\TFM);
data final27;set final;modelo='LevY';

%cruzadaneural(archivo=uno,vardepen=precio,conti=accesibilidad antiguedad demanda dis_airp
dis_centro puntuacion resenas,categor=AC accesible anfitrion_recibe apto_ninos ascensor balcon banos
capacidad grandes_electro parking permite_mascotas politica_cancelacion superanfitrion
tiempo_respuesta tipo,ngrupos=4,sinicio=12345, sfinal=12354,nodos=8,early=20,algo=levmar,
directorio=C:\Users\lcn97\Documents\TFM);
data final28;set final;modelo='LevN';
data union;set final24 final27 final28;
proc boxplot data=union;plot media*modelo;run;
/* LevY */
/*función activación*/
%macro activalcruza;
%let lista='TANH LOG ARC LIN SIN SOF GAU';
%let nume=7;
%do  i=1 %to &nume;
data _null_;activa=scanq(&lista,&i);call symput('activa',left(activa));run;
 %cruzadaneural(archivo=uno,
    vardepen=precio,conti=accesibilidad antiguedad demanda dis_airp dis_centro puntuacion resenas,
    categor=AC accesible anfitrion_recibe apto_ninos ascensor balcon banos capacidad grandes_electro
    parking permite_mascotas politica_cancelacion superanfitrion tiempo_respuesta tipo,ngrupos=4,
sinicio=12345,sfinal=12354,nodos=8,early=,algo=levmar,directorio=C:\Users\lcn97\Documents\TFM,a
cti=&activa);
 data final&i;set final;modelo="&activa";put modelo=;run;
%end;
data union;set %do i=1 %to &nume; final&i %end;
%mend;

%activalcruza;
proc print data=union;run;
```

```
proc boxplot data=union;plot media*modelo;run;
/*gana seno*/
```

8.7.2 Redes neuronales en R.

```
#  ***  Redes Neuronales *** #
library(caret)
library(parallel)
library(doParallel)
cluster <- makeCluster(detectCores() - 1)
registerDoParallel(cluster)

# Validación cruzada repetida
control<-trainControl(method = "repeatedcv",number=4,repeats=5,savePredictions = "all")
# Rejilla
avnnetgrid <-expand.grid(size=c(4,6,8,10,12,14,16,18),decay=c(0.01,0.1,0.001),bag=FALSE)
redavnnet<- train(precio~.,data=estdata,  method="avNNet",linout = TRUE, maxit=100,
trControl=control, repeats=5,tuneGrid=avnnetgrid)
redavnnet

# Graficamos los resultados
library(ggplot2)
dat<-as.data.frame(redavnnet$results)
ggplot(dat, aes(x=factor(size), y=RMSE, color=factor(decay))) +
geom_point(position=position_dodge(width=0.5),size=3)

# Exploramos intervalos 16-18; 0.01-0.1
avnnetgrid <-expand.grid(size=c(17, 18),decay=c(0.01,0.05,0.1,0.5),bag=FALSE)

redavnnet1<- train(precio~.,data=estdata,method="avNNet",linout = TRUE,maxit=100,
trControl=control,repeats=5,tuneGrid=avnnetgrid)
redavnnet1
stopCluster(cluster) # shut down the cluster
registerDoSEQ(); # force R to return to single threaded processing

# Graficamos los resultados
dat<-as.data.frame(redavnnet1$results)
ggplot(dat, aes(x=factor(size), y=RMSE, color=factor(decay))) +
geom_point(position=position_dodge(width=0.5),size=3)
# Mejor 17 con 0.5, pero puede ser azar. Comparamos con validacion cruzada repetida.
medias1<-cruzadaavnnet(data=estdata,vardep="precio", listconti=listconti, listclass=c(""),
grupos=4,sinicio=1234,repe=5,size=c(17),decay=c(0.05),repeticiones=5,itera=100)
medias1$modelo="avnnet17-0.05"

medias2<-cruzadaavnnet(data=estdata,vardep="precio",listconti=listconti, listclass=c(""),
grupos=4,sinicio=1234,repe=5,size=c(18),decay=c(0.05),repeticiones=5,itera=100)
medias2$modelo="avnnet18-0.05"

medias3<-cruzadaavnnet(data=estdata,vardep="precio",listconti=listconti, listclass=c(""),
grupos=4,sinicio=1234,repe=5,size=c(18),decay=c(0.1),repeticiones=5,itera=100)
medias3$modelo="avnnet18-0.1"

medias5<-cruzadaavnnet(data=estdata,vardep="precio",listconti=listconti,listclass=c(""),
grupos=4,sinicio=1234,repe=5, size=c(18),decay=c(0.01),repeticiones=5,itera=100)
medias5$modelo="avnnet18-0.01"

medias6<-cruzadaavnnet(data=estdata,vardep="precio", listconti=listconti,listclass=c(""),
grupos=4,sinicio=1234,repe=5,size=c(17),decay=c(0.5),repeticiones=5,itera=100)
```

```
medias6$modelo="avnnet17-0.5"

union1<-rbind(medias1,medias2,medias3,medias5,medias6)
par(cex.axis=1)
boxplot(data=union1,error~modelo, col="light blue", ylab="MSE")
# 17 0.05

# Iteraciones
control<-trainControl(method = "repeatedcv",number=4,repeats=1,savePredictions = "all")
avnnetgrid <-expand.grid(size=c(17),decay=c(0.05),bag=FALSE)
df<- data.frame(matrix(ncol = 2, nrow = 0))
x <- c("maxit", "RMSE")
j = 1
for (maxit in seq(40,1000,20)) {
  redavnnet2<- train(precio~.,data=estdata,  method="avNNet",linout = TRUE,maxit=maxit,
trControl=control,repeats=5,tuneGrid=avnnetgrid)
  df[j,1] = maxit
  df[j,2] = redavnnet2$results$RMSE
  j = j+1
}
plot(df[,1], df[,2], cex.lab=1, cex.axis=1, cex.sub=1, xlab="Iteraciones",
ylab="RMSE",col="red",pch=19)

# Mejores 940, 800 y 440. Comparamos con validacion cruzada.
medias6$modelo="avnnet-100"
medias60<-cruzadaavnnet(data=estdata,vardep="precio",listconti=listconti,listclass=c(""),
grupos=4,sinicio=1234,repe=5,size=c(17),decay=c(0.5),repeticiones=5,itera=440)
medias60$modelo="avnnet-440"

medias61<-cruzadaavnnet(data=estdata,vardep="precio",listconti=listconti,listclass=c(""),
grupos=4,sinicio=1234,repe=5,size=c(17),decay=c(0.5),repeticiones=5,itera=800)
medias61$modelo="avnnet-800"

medias62<-cruzadaavnnet(data=estdata,vardep="precio",listconti=listconti, listclass=c(""),
grupos=4,sinicio=1234,repe=5,size=c(17),decay=c(0.5),repeticiones=5,itera=940)
medias62$modelo="avnnet-940"

union1<-rbind(medias6, medias60, medias61, medias62)
par(cex.axis=1)
boxplot(data=union1,error~modelo, col="light blue", ylab="MSE")
```

8.7.3 Random forest.

```
#  ***  Random Forest *** #
library(randomForest)
cluster <- makeCluster(detectCores() - 1)
registerDoParallel(cluster)

rfgrid<-expand.grid(mtry=c(3,5,7,9,11,13,15,17,19,21,23,25,27,29))
control<-trainControl(method = "cv",number=4,savePredictions = "all",classProbs=TRUE)
rf<- train(precio~.,data=estdata, method="rf",trControl=control,tuneGrid=rfgrid,
          ntree=1000, nodesize=10, replace=TRUE,importance=TRUE)
plot(rf)

# Exploramos intervalo 5-9
rfgrid<-expand.grid(mtry=c(5,6,7,8,9))
control<-trainControl(method = "cv",number=4,savePredictions = "all",classProbs=TRUE)
```

```
rf1<- train(precio~.,data=estdata, method="rf",trControl=control,tuneGrid=rfgrid,
        ntree=1000,nodesize=10,replace=TRUE,importance=TRUE)
plot(rf1)
stopCluster(cluster)
registerDoSEQ();

# Compramos 6, 7 y 8 con validación cruzada repetida
medias7<-cruzadarf(data=estdata,vardep="precio",listconti=listconti, listclass=c(""),  grupos=4,
sinicio=1234, repe=5, nodesize=10, replace=TRUE, ntree=1000, mtry=6)
medias7$modelo="rf-6"

medias8<-cruzadarf(data=estdata,vardep="precio", listconti=listconti,   listclass=c(""),
grupos=4,sinicio=1234,repe=5,nodesize=10,replace=TRUE,ntree=1000,mtry=7)
medias8$modelo="rf-0.75"

medias9<-cruzadarf(data=estdata,vardep="precio", listconti=listconti, listclass=c(""),
grupos=4,sinicio=1234,repe=5,nodesize=10,replace=TRUE,ntree=1000,mtry=8)
medias9$modelo="rf-8"

union1<-rbind(medias7,medias8,medias9)
par(cex.axis=1)
boxplot(data=union1,error~modelo,col="light blue",ylab="MSE")
# rf 7 (medias8) estudiamos iteraciones

rfbis<-randomForest(precio~.,data=estdata, mtry=7, ntree=5000, sampsize=12474, nodesize=10,
replace=TRUE)
plot(rfbis$mse,ylim=c(1740,2000),ylab="MSE")

medias10<-cruzadarf(data=estdata,vardep="precio",listconti=listconti, listclass=c(""),
grupos=4,sinicio=1234,repe=5,nodesize=10,replace=TRUE,ntree=500,mtry=7)
medias10$modelo="rf-500"

medias11<-cruzadarf(data=estdata,vardep="precio",listconti=listconti, listclass=c(""),
grupos=4,sinicio=1234,repe=5,nodesize=10,replace=TRUE,ntree=700,mtry=7)
medias11$modelo="rf-700"

union1<-rbind(medias8,medias10,medias11)
par(cex.axis=1)
boxplot(data=union1,error~modelo,col="light blue",ylab="MSE")
# 1000 iteraciones (medias8)

# Tamaño muestral
medias12<-cruzadarf(data=estdata,vardep="precio", listconti=listconti, listclass=c(""),
grupos=4,sinicio=1234,repe=5,nodesize=10,replace=TRUE,ntree=700,mtry=7, sampsize=10395)
medias12$modelo="rf-0.63"

medias13<-cruzadarf(data=estdata,vardep="precio",listconti=listconti, listclass=c(""),
grupos=4,sinicio=1234,repe=5,nodesize=10,replace=TRUE,ntree=700,mtry=7, sampsize=8316)
medias13$modelo="rf-0.50"

medias14<-cruzadarf(data=estdata,vardep="precio", listconti=listconti, listclass=c(""),
grupos=4,sinicio=1234,repe=5,nodesize=10,replace=TRUE,ntree=700,mtry=7, sampsize=6237)
medias14$modelo="rf-0.38"

medias15<-cruzadarf(data=estdata,vardep="precio", listconti=listconti,listclass=c(""),
grupos=4,sinicio=1234,repe=5,nodesize=10,replace=TRUE,ntree=700,mtry=7, sampsize=4158)
medias15$modelo="rf-0.25"
```

```
medias16<-cruzadarf(data=estdata,vardep="precio",listconti=listconti,listclass=c(""),
grupos=4,sinicio=1234,repe=5,nodesize=10,replace=TRUE,ntree=700,mtry=7, sampsize=2079)
medias16$modelo="rf-0.13"

union1<-rbind(medias8,medias12,medias13,medias14,medias15,medias16)
par(cex.axis=1)
boxplot(data=union1,error~modelo,col="light blue",ylab="MSE")
# max tamaño muestral (medias8)

# tamaño hoja. medias 8-> tamaño 10
medias17<-cruzadarf(data=estdata,vardep="precio",listconti=listconti,listclass=c(""),
grupos=4,sinicio=1234,repe=5,nodesize=5,replace=TRUE,ntree=700,mtry=7)
medias17$modelo="rf-5"

medias18<-cruzadarf(data=estdata,vardep="precio",listconti=listconti,listclass=c(""),
grupos=4,sinicio=1234,repe=5,nodesize=15,replace=TRUE,ntree=700,mtry=7)
medias18$modelo="rf-15"

medias19<-cruzadarf(data=estdata,vardep="precio",listconti=listconti,listclass=c(""),
grupos=4,sinicio=1234,repe=5,nodesize=20,replace=TRUE,ntree=700,mtry=7)
medias19$modelo="rf-20"

union1<-rbind(medias8,medias17,medias18,medias19)
par(cex.axis=1)
boxplot(data=union1,error~modelo,col="light blue",ylab="MSE")
# 5 hojas medias17
```

8.7.4 Gradient boosting en SAS.

```
/* ------------ GRADIENT BOOSTING ---------------- */
%cruzadatreeboost(archivo=uno,vardepen=precio,
conti=accesibilidad antiguedad demanda dis_airp dis_centro puntuacion resenas, categor=AC accesible
anfitrion_recibe apto_ninos ascensor balcon banos capacidad grandes_electro parking permite_mascotas
politica_cancelacion superanfitrion tiempo_respuesta tipo, ngrupos=4,sinicio=12345, sfinal=12354,
leafsize=5, iteraciones=100, shrink=0.1, maxbranch=2, maxdepth=4, mincatsize=15,minobs=20);
data final89;set final;modelo='GB0.10';

%cruzadatreeboost(archivo=uno,vardepen=precio, conti=accesibilidad antiguedad demanda dis_airp
dis_centro puntuacion resenas, categor=AC accesible anfitrion_recibe apto_ninos ascensor balcon banos
capacidad grandes_electro parking permite_mascotas politica_cancelacion superanfitrion
tiempo_respuesta tipo, ngrupos=4, sinicio=12345, sfinal=12354, leafsize=5, iteraciones=600,
shrink=0.05,maxbranch=2,  maxdepth=4, mincatsize=15, minobs=20);
data final90;set final;modelo='GB0.05';

%cruzadatreeboost(archivo=uno,vardepen=precio, conti=accesibilidad antiguedad demanda dis_airp
dis_centro puntuacion resenas, categor=AC accesible anfitrion_recibe apto_ninos ascensor balcon banos
capacidad grandes_electro parking permite_mascotas politica_cancelacion superanfitrion
tiempo_respuesta tipo,ngrupos=4,sinicio=12345,sfinal=12354,leafsize=5,iteraciones=800, shrink=0.03,
maxbranch=2, maxdepth=4,mincatsize=15,minobs=20);
data final91;set final;modelo='GB0.03';

%cruzadatreeboost(archivo=uno,vardepen=precio,conti=accesibilidad antiguedad demanda dis_airp
dis_centro puntuacion resenas,categor=AC accesible anfitrion_recibe apto_ninos ascensor balcon banos
capacidad grandes_electro parking permite_mascotas politica_cancelacion superanfitrion
tiempo_respuesta tipo, ngrupos=4, sinicio=12345, sfinal=12354,leafsize=5,iteraciones=900,
shrink=0.02,maxbranch=2, maxdepth=4, mincatsize=15,minobs=20);
data final92;set final;modelo='GB0900';
```

```
%cruzadatreeboost(archivo=uno,vardepen=precio, conti=accesibilidad antiguedad demanda dis_airp
dis_centro puntuacion resenas, categor=AC accesible anfitrion_recibe apto_ninos ascensor balcon banos
capacidad grandes_electro parking permite_mascotas politica_cancelacion superanfitrion
tiempo_respuesta tipo,ngrupos=4, sinicio=12345,  sfinal=12354,leafsize=5,iteraciones=1000,
shrink=0.01,maxbranch=2,maxdepth=4,mincatsize=15,minobs=20);
data final93;set final;modelo='GB0.01';

data union;set final89 final90 final91 final92 final93;
proc boxplot data=union;plot media*modelo;run;
/*final 91*/
/*iteraciones*/
%cruzadatreeboost(archivo=uno,vardepen=precio,conti=accesibilidad antiguedad demanda dis_airp
dis_centro puntuacion resenas,categor=AC accesible anfitrion_recibe apto_ninos ascensor balcon banos
capacidad grandes_electro parking permite_mascotas politica_cancelacion superanfitrion
tiempo_respuesta tipo,  ngrupos=4,sinicio=12345, sfinal=12349,leafsize=5,iteraciones=800,
shrink=0.03,maxbranch=2,maxdepth=4, mincatsize=15,minobs=20);
data final91;set final;modelo='GB0800';

%cruzadatreeboost(archivo=uno,vardepen=precio,conti=accesibilidad antiguedad demanda dis_airp
dis_centro puntuacion resenas,categor=AC accesible anfitrion_recibe apto_ninos ascensor balcon banos
capacidad grandes_electro parking permite_mascotas politica_cancelacion superanfitrion
tiempo_respuesta tipo, ngrupos=4,sinicio=12345, sfinal=12349,leafsize=5,iteraciones=100,
shrink=0.03,maxbranch=2,maxdepth=4,mincatsize=15,minobs=20);
data final93;set final;modelo='GB0100';

%cruzadatreeboost(archivo=uno,vardepen=precio,conti=accesibilidad antiguedad demanda dis_airp
dis_centro puntuacion resenas,categor=AC accesible anfitrion_recibe apto_ninos ascensor balcon banos
capacidad grandes_electro  parking permite_mascotas politica_cancelacion superanfitrion
tiempo_respuesta tipo,ngrupos=4, sinicio=12345, sfinal=12349,leafsize=5,iteraciones=500,
shrink=0.02,maxbranch=2,  maxdepth=4, mincatsize=15, minobs=20);
data final94;set final;modelo='GB0500';

%cruzadatreeboost(archivo=uno,vardepen=precio,conti=accesibilidad antiguedad demanda dis_airp
dis_centro puntuacion resenas,categor=AC accesible anfitrion_recibe apto_ninos ascensor balcon banos
capacidad grandes_electro parking permite_mascotas politica_cancelacion superanfitrion
tiempo_respuesta tipo,ngrupos=4, sinicio=12345, sfinal=12349,leafsize=5,iteraciones=1200,
shrink=0.02,maxbranch=2,maxdepth=4, mincatsize=15,minobs=20);
data final95;set final;modelo='GB1200';

%cruzadatreeboost(archivo=uno,vardepen=precio,conti=accesibilidad antiguedad demanda dis_airp
dis_centro puntuacion resenas,categor=AC accesible anfitrion_recibe apto_ninos ascensor balcon banos
capacidad grandes_electro parking permite_mascotas politica_cancelacion superanfitrion
tiempo_respuesta tipo,ngrupos=4,sinicio=12345, sfinal=12349,leafsize=5,iteraciones=2000,
shrink=0.02,maxbranch=2,maxdepth=4, mincatsize=15,minobs=20);
data final96;set final;modelo='GB2000';
data union;set final91 final93 final94 final95 final96;
proc boxplot data=union;plot media*modelo;run;
/*1200 - final95*/
/* profundidad y tamaño de hoja*/
%cruzadatreeboost(archivo=uno,vardepen=precio,conti=accesibilidad antiguedad demanda dis_airp
dis_centro puntuacion resenas,categor=AC accesible anfitrion_recibe apto_ninos ascensor balcon banos
capacidad grandes_electro parking permite_mascotas politica_cancelacion superanfitrion
tiempo_respuesta tipo,ngrupos=4,sinicio=12345, sfinal=12354,leafsize=5,iteraciones=1200,
shrink=0.03,maxbranch=2,maxdepth=4, mincatsize=15,minobs=20);
data final97;set final;modelo='GB4-05';
```

```
%cruzadatreeboost(archivo=uno,vardepen=precio,conti=accesibilidad antiguedad demanda dis_airp
dis_centro puntuacion resenas,categor=AC accesible anfitrion_recibe apto_ninos ascensor balcon banos
capacidad grandes_electro parking permite_mascotas politica_cancelacion superanfitrion
tiempo_respuesta tipo,ngrupos=4,sinicio=12345, sfinal=12354,leafsize=10,iteraciones=1200,
shrink=0.03,maxbranch=2,maxdepth=4, mincatsize=15,minobs=20);
data final98;set final;modelo='GB4-10';

%cruzadatreeboost(archivo=uno,vardepen=precio,conti=accesibilidad antiguedad demanda dis_airp
dis_centro puntuacion resenas,categor=AC accesible anfitrion_recibe apto_ninos ascensor balcon banos
capacidad grandes_electro parking permite_mascotas politica_cancelacion superanfitrion
tiempo_respuesta tipo,ngrupos=4,sinicio=12345, sfinal=12354,leafsize=10, iteraciones=1200,
shrink=0.03,maxbranch=2,maxdepth=6, mincatsize=15,minobs=20);
data final99;set final;modelo='GB6-10';

%cruzadatreeboost(archivo=uno,vardepen=precio, conti=accesibilidad antiguedad demanda dis_airp
dis_centro puntuacion resenas, categor=AC accesible anfitrion_recibe apto_ninos ascensor balcon banos
capacidad grandes_electro parking permite_mascotas politica_cancelacion superanfitrion
tiempo_respuesta tipo,ngrupos=4,sinicio=12345, sfinal=12354,leafsize=5,iteraciones=1200,
shrink=0.03,maxbranch=2,maxdepth=6, mincatsize=15,minobs=20);
data final100;set final;modelo='GB6-05';

%cruzadatreeboost(archivo=uno,vardepen=precio,conti=accesibilidad antiguedad demanda dis_airp
dis_centro puntuacion resenas,categor=AC accesible anfitrion_recibe apto_ninos ascensor balcon banos
capacidad grandes_electro parking permite_mascotas politica_cancelacion superanfitrion
tiempo_respuesta tipo,ngrupos=4,sinicio=12345, sfinal=12354,leafsize=10,iteraciones=1200,
shrink=0.03,maxbranch=2,maxdepth=8,mincatsize=15,minobs=20);
data final101;set final;modelo='GB8-10';

%cruzadatreeboost(archivo=uno,vardepen=precio,conti=accesibilidad antiguedad demanda dis_airp
dis_centro puntuacion resenas,categor=AC accesible anfitrion_recibe apto_ninos ascensor balcon banos
capacidad grandes_electro parking permite_mascotas politica_cancelacion superanfitrion
tiempo_respuesta tipo,ngrupos=4,sinicio=12345, sfinal=12354,leafsize=5,iteraciones=1200,
shrink=0.03,maxbranch=2,maxdepth=8, mincatsize=15,minobs=20);
data final102;set final;modelo='GB8-05';
data union;set final97 final98 final99 final100 final101 final102;
proc boxplot data=union;plot media*modelo;run;
/* final 99 */
```

8.7.5 Gradient boosting en R.

```
# *** Gradient Boosting *** #
cluster <- makeCluster(detectCores() - 1)
registerDoParallel(cluster)

gbmgrid<-expand.grid(shrinkage=c(0.1,0.05,0.03,0.01,0.001),n.minobsinnode=c(5,10,20),
                n.trees=c(100,500,1000,2500,5000),interaction.depth=c(2))
control<-trainControl(method = "cv",number=4,savePredictions = "all")
gbm<- train(precio~.,data=estdata, method="gbm",trControl=control,tuneGrid=gbmgrid,
         distribution="gaussian", bag.fraction=1,verbose=FALSE)
gbm
plot(gbm)

stopCluster(cluster) # shut down the cluster
registerDoSEQ(); # force R to return to single threaded processing
medias20 <-cruzadagbm(data=estdata,vardep="precio", listconti=listconti, listclass=c(""),
grupos=4,sinicio=1234,repe=5,n.minobsinnode=10,shrinkage=0.03,n.trees=5000, interaction.depth=2)
medias20$modelo="gbm1"
```

```
medias21 <-cruzadagbm(data=estdata,vardep="precio", listconti=listconti,   listclass=c(""),
grupos=4,sinicio=1234,repe=5,n.minobsinnode=10,shrinkage=0.05,n.trees=2500, interaction.depth=2)
medias21$modelo="gbm-10"

medias22 <-cruzadagbm(data=estdata,vardep="precio", listconti=listconti,  listclass=c(""),
grupos=4,sinicio=1234,repe=5,n.minobsinnode=20,shrinkage=0.1,n.trees=1000, interaction.depth=2)
medias22$modelo="gbm3"

union1<-rbind(medias20,medias21,medias22)
par(cex.axis=1)
boxplot(data=union1,error~modelo,col="light blue",ylab="MSE")
# Gbm 2 shrinkage 0.05 y 2500 iteraciones y 10 observaciones; medias 21

# Tamaño hoja
medias23 <-cruzadagbm(data=estdata,vardep="precio",listconti=listconti, listclass=c(""),
grupos=4,sinicio=1234,repe=5,n.minobsinnode=5,shrinkage=0.05,n.trees=2500, interaction.depth=2)
medias23$modelo="gbm-5"

medias24 <-cruzadagbm(data=estdata,vardep="precio",listconti=listconti, listclass=c(""),
grupos=4,sinicio=1234,repe=5,n.minobsinnode=15,shrinkage=0.05,n.trees=2500, interaction.depth=2)
medias24$modelo="gbm-15"

medias25 <-cruzadagbm(data=estdata,vardep="precio",listconti=listconti,listclass=c(""),
grupos=4,sinicio=1234,repe=5,n.minobsinnode=20,shrinkage=0.05,n.trees=2500, interaction.depth=2)
medias25$modelo="gbm-2500"

union1<-rbind(medias21,medias23,medias24,medias25)
par(cex.axis=1)
boxplot(data=union1,error~modelo,col="light blue",ylab="MSE")
# 20 - medias 25
# Gana el segundo modelo, pero tiene 2500 árboles. Estudiamos early stopping
cluster <- makeCluster(detectCores() - 1)
registerDoParallel(cluster)
gbmgrid<-expand.grid(shrinkage=c(0.05), n.minobsinnode=c(20),
                     n.trees=c(300,600,900,1200,1500,1800,2100,2225,2500),interaction.depth=c(2))
control<-trainControl(method = "cv",number=4,savePredictions = "all")
gbm<- train(precio~.,data=estdata, method="gbm",trControl=control,tuneGrid=gbmgrid,
          distribution="gaussian", bag.fraction=1,verbose=FALSE)
gbm
plot(gbm)
stopCluster(cluster) # shut down the cluster
registerDoSEQ(); # force R to return to single threaded processing
#2100?

medias26 <-cruzadagbm(data=estdata,vardep="precio",listconti=listconti, listclass=c(""),
grupos=4,sinicio=1234,repe=5,n.minobsinnode=20,shrinkage=0.05,n.trees=2100, interaction.depth=2)
medias26$modelo="gbm-2100"

union1<-rbind(medias26,medias25)
par(cex.axis=1)
boxplot(data=union1,error~modelo,col="light blue",ylab="MSE")
```

8.7.6 Extreme gradient boosting.

```
# *** Extreme Gradient Boosting *** #
cluster <- makeCluster(detectCores() - 1)
registerDoParallel(cluster)
```

```
xgbmgrid<-expand.grid(min_child_weight=c(5,10,20),eta=c(0.1,0.05,0.03,0.01,0.001),
nrounds=c(100,500,1000,3000,5000),max_depth=6,gamma=0,colsample_bytree=1, subsample=1)
control<-trainControl(method = "cv",number=4,savePredictions = "all",classProbs=TRUE) xgbm<-
train(precio~.,data=estdata,method="xgbTree",trControl=control,tuneGrid=xgbmgrid, verbose=FALSE)
xgbm
plot(xgbm)

stopCluster(cluster) # shut down the cluster
registerDoSEQ(); # force R to return to single threaded processing
medias27<-cruzadaxgbm(data=estdata,vardep="precio",listconti=listconti, listclass=c(""),grupos=4,
sinicio=1234,repe=5,min_child_weight=10,eta=0.05,nrounds=500,
max_depth=6,gamma=0,colsample_bytree=1,subsample=1)
medias27$modelo="xgbm1"

medias28<-cruzadaxgbm(data=estdata,vardep="precio", listconti=listconti,
listclass=c(""),grupos=4,sinicio=1234,repe=5, min_child_weight=10,eta=0.01,nrounds=3000,
max_depth=6,gamma=0,colsample_bytree=1,subsample=1)
medias28$modelo="xgbm2"

medias29<-cruzadaxgbm(data=estdata,vardep="precio", listconti=listconti, listclass=c(""),grupos=4,
sinicio=1234,repe=5, min_child_weight=20,eta=0.01,nrounds=3000,
max_depth=6,gamma=0,colsample_bytree=1,subsample=1)
medias29$modelo="xgbm3"

union1<-rbind(medias27,medias28,medias29)
par(cex.axis=1)
boxplot(data=union1,error~modelo,col="light blue",ylab="MSE")
# medias27. estudiamos gamma
cluster <- makeCluster(detectCores() - 1)
registerDoParallel(cluster)
xgbmgrid<-expand.grid(min_child_weight=c(10),eta=c(0.05),nrounds=c(500), max_depth=6,
gamma=c(0,0.01,0.1,0.2,0.3,0.4,0.5,0.6,0.7,0.8,0.9,1),colsample_bytree=1, subsample=1)
control<-trainControl(method = "cv",number=4,savePredictions = "all",classProbs=TRUE)
xgbm<- train(precio~.,data=estdata,
        method="xgbTree",trControl=control,tuneGrid=xgbmgrid,verbose=FALSE)
xgbm
plot(xgbm)
stopCluster(cluster) # shut down the cluster
registerDoSEQ(); # force R to return to single threaded processing
#gamma 0.

medias30<-cruzadaxgbm(data=estdata,vardep="precio",listconti=listconti,listclass=c(""),
grupos=4,sinicio=1234,repe=5,min_child_weight=10,eta=0.05,nrounds=500,max_depth=6,
gamma=0,colsample_bytree=0.8,subsample=1)
medias30$modelo="xgbm-10"

medias31<-cruzadaxgbm(data=estdata,vardep="precio",listconti=listconti, listclass=c(""),
grupos=4,sinicio=1234,repe=5,min_child_weight=10,eta=0.05,nrounds=500,max_depth=6,
gamma=0,colsample_bytree=1,subsample=0.8)
medias31$modelo="xgbm0.8obs"

union1<-rbind(medias27,medias30,medias31)
par(cex.axis=1)
boxplot(data=union1,error~modelo,col="light blue",ylab="MSE")
# medias 30

# tamaño hoja
```

```
medias32<-cruzadaxgbm(data=estdata,vardep="precio",listconti=listconti,listclass=c(""),
grupos=4,sinicio=1234,repe=5,min_child_weight=5,eta=0.05,nrounds=500,max_depth=6,
gamma=0,colsample_bytree=0.8,subsample=1)
medias32$modelo="xgbm-5"

medias33<-cruzadaxgbm(data=estdata,vardep="precio",listconti=listconti,listclass=c(""),
grupos=4,sinicio=1234,repe=5,min_child_weight=15,eta=0.05,nrounds=500,max_depth=6,
gamma=0,colsample_bytree=0.8,subsample=1)
medias33$modelo="xgbm-15"

medias34<-cruzadaxgbm(data=estdata,vardep="precio",listconti=listconti,listclass=c(""),
grupos=4,sinicio=1234,repe=5,min_child_weight=20,eta=0.05,nrounds=500,max_depth=6,
gamma=0,colsample_bytree=0.8,subsample=1)
medias34$modelo="xgbm-20"

union1<-rbind(medias30,medias32,medias33,medias34)
par(cex.axis=1)
boxplot(data=union1,error~modelo,col="light blue",ylab="MSE")
# xgbm10 medias30
```

8.7.7 Support Vector Machine.

```
# *** SVM Lineal *** #
cluster <- makeCluster(detectCores() - 1)
registerDoParallel(cluster)

SVMgrid<-expand.grid(C=c(0.01,0.05,0.5,1,2,5,10,50))
control<-trainControl(method = "cv",number=4, savePredictions = "all")
SVM<- train(precio~.,data=estdata,
method="svmLinear",trControl=control,tuneGrid=SVMgrid,verbose=FALSE)
SVM$results
plot(svmlineal$C,svmlineal$RMSE,col="red",pch=19,ylab='RMSE',xlab='C')

stopCluster(cluster) # shut down the cluster
registerDoSEQ(); # force R to return to single threaded processing
medias35<-cruzadaSVM(data=estdata,vardep="precio",
                listconti=listconti,listclass=c(""),grupos=4,sinicio=1234,repe=5,C=0.5)
medias35$modelo="SVM-0.5"

medias36<-cruzadaSVM(data=estdata,vardep="precio",
                listconti=listconti,listclass=c(""),grupos=4,sinicio=1234,repe=5,C=5)
medias36$modelo="SVM-5"

medias37<-cruzadaSVM(data=estdata,vardep="precio",
                listconti=listconti,listclass=c(""),grupos=4,sinicio=1234,repe=5,C=10)
medias37$modelo="SVM-10"

union1<-rbind(medias35,medias36,medias37)
par(cex.axis=1)
boxplot(data=union1,error~modelo,col="light blue",ylab="MSE")

# *** SVM Polinomial *** #
SVMgrid<-expand.grid(C=c(0.01,0.05,0.1,0.2,1), degree=c(2,3), scale=c(0.1,0.5,1,2))
control<-trainControl(method = "cv",number=2,savePredictions = "all")
SVM<- train(precio~.data=estdata,method="svmPoly",trControl=control,
tuneGrid=SVMgrid,verbose=FALSE)
SVM
SVM$results
```

```
dat<-as.data.frame(SVM$results)
library(ggplot2)
ggplot(dat, aes(x=factor(C), y=RMSE, color=factor(degree),pch=factor(scale))) +
  geom_point(position=position_dodge(width=0.5),size=3)
ggplot(dat[1:22, aes(x=factor(C), y=RMSE, color=factor(degree),pch=factor(scale))) +
  geom_point(position=position_dodge(width=0.5),size=3)

medias38<-cruzadaSVMpoly(data=estdata,vardep="precio",listconti=listconti,listclass=c(""),
                    grupos=4,sinicio=1234,repe=5,C=0.01,degree=3,scale=0.1)
medias38$modelo="SVMPoly1"

medias39<-cruzadaSVMpoly(data=estdata,vardep="precio",listconti=listconti,listclass=c(""),
                     grupos=4,sinicio=1234,repe=5,C=0.05,degree=2,scale=0.1)
medias39$modelo="SVMPoly2"

medias40<-cruzadaSVMpoly(data=estdata,vardep="precio",listconti=listconti,listclass=c(""),
                     grupos=4,sinicio=1234,repe=5,C=0.1,degree=2,scale=0.1)
medias40$modelo="SVMPoly3"
union1<-rbind(medias38,medias39,medias40)
par(cex.axis=1)
boxplot(data=union1,error~modelo,col="light blue",ylab="MSE")

# *** SVM Radial *** #
cluster <- makeCluster(detectCores() - 1)
registerDoParallel(cluster)
SVMgrid<-expand.grid(C=c(0.01,0.05,0.1,0.2,0.5,1,2,5,10,30),
sigma=c(0.01,0.05,0.1,0.2,0.5,1,2,5,10,30))
control<-trainControl(method = "cv",number=2,savePredictions = "all")
SVM<- train(precio~.,data=estdata, method="svmRadial", trControl=control, tuneGrid=SVMgrid,
verbose=FALSE)
SVM
dat<-as.data.frame(SVM$results)
ggplot(dat, aes(x=factor(C), y=RMSE, color=factor(sigma)))+
  geom_point(position=position_dodge(width=0.5),size=3)
# C     sigma  RMSE
# 2 0.05    44.65187
# 10    0.01    44.71032
# 5 0.01    44.71648
stopCluster(cluster) # shut down the cluster
registerDoSEQ(); # force R to return to single threaded processing

medias50<-cruzadaSVMRBF(data=estdata,vardep="precio",listconti=listconti, listclass=c(""),
                    grupos=4, sinicio=1234, repe=5,C=2,sigma=0.05)
medias50$modelo="SVMRBF2"

medias51<-cruzadaSVMRBF(data=estdata,vardep="precio",listconti=listconti,listclass=c(""),
                    grupos=4,sinicio=1234,repe=5,C=10,sigma=0.01)
medias51$modelo="SVMRBF10"

medias52<-cruzadaSVMRBF(data=estdata, vardep="precio",listconti=listconti,listclass=c(""),
                    grupos=4,sinicio=1234,repe=5,C=5,sigma=0.01)
medias52$modelo="SVMRBF5"
union1<-rbind(medias50,medias51,medias52)
par(cex.axis=1)
boxplot(data=union1,error~modelo,col="light blue",ylab="MSE")
```

8.7.8 Ensamblado en SAS.

```
/* MACRO ENSAMBLADO */
%macro cruzadastacksin(archivo=,vardepen=,listclass=,listconti=,ngrupos=,seminicio=,semifinal=,
nodos=8,algo=levmar,rediter=100,act=sin,/*red*/
bleafsize=20,iterations=1200,bmaxbranch=2,bmaxdepth=6,shrinkage=0.03/* g boosting*/);
data sal1;data sal2;data sal3; run;
data final;run;
*proc printto print='c:\ca.txt' log='c:\loga.txt';run;
%do semilla=&seminicio %to &semifinaldata dos;set &archivo;u=ranuni(&semilla);
proc sort data=dos;by u;run;
data dos (drop=nume);
retain grupo 1;
set dos nobs=nume;
if _n_>grupo*nume/&ngrupos then grupo=grupo+1;
run;
data fantasma;run;
data unionsalfin;run;
data unifin;run;
%do exclu=1 %to &ngrupos;
data tres;set dos;semilla=&semilla;if grupo ne &exclu then vardep=&vardepen*1;run;

/* REGRESIÓN */
proc glm data=tres noprint;
        %if &listclass ne %then %do;class &listclass;model vardep=&listconti &listclass;%end;
        %else %do;model vardep=&listconti;%end;
        output out=sal p=predi1;run;

        data sal1;set sal;run;

/*RED */
PROC DMDB DATA=tres dmdbcat=catatres;
target vardep ;
var vardep &listconti;
%if &listclass ne %then %do;class &listclass;%end;
;run;
proc neural data=tres dmdbcat=catatres ;
input &listconti/ id=i;
%if &listclass ne %then %do;input &listclass /level=nominal;%end;
target vardep/ id=o ;
hidden &nodos/ id=h act=&act;
netoptions randist=normal ranscale=0.15 random=15459;
prelim 15 preiter=10 ;
train maxiter=&rediter technique=&algo;
score data=tres out=salred;
run;

data sal2 (keep=&vardepen predi2 grupo vardep semilla);set salred;predi2=p_vardep;run;

/*GRADIENT BOOSTING */
    proc treeboost data=tres
    exhaustive=1000 intervaldecimals=max
    leafsize=&bleafsize iterations=&iterations maxbranch=&bmaxbranch
    maxdepth=&bmaxdepth mincatsize=15 missing=useinsearch shrinkage=&shrinkage
    splitsize=10;
    %if &listclass ne %then %do;input &listclass /level=nominal;%end;
    input &listconti/level=interval;
```

```
    target vardep ;
    subseries largest;
    score out=salboost;
run;
data sal3 (keep=&vardepen predi3 grupo vardep);set salboost;predi3=p_vardep;run;

/* PRUEBAS CON STACKING */
data unionsal (drop=ygorro);merge sal1 sal2 sal3;
predi4=(predi1+predi2)/2; /* RED -LOG */
predi5=(predi2+predi3)/2;/* RED -BOOST*/
predi6=(predi1+predi3)/2;/* LOG-BOOST */
predi7=(predi1+predi2+predi3)/3;/* RED -LOG-BOOST*/
predi8=(predi1*0.1+predi2*0.4+predi3*0.5);/* RED-LOG-BOOST ponderado*/
run;
data salfin (keep=&vardepen vardep predi1-predi8 grupo);set unionsal;if grupo=&exclu then output;run;
data unionsalfin;set unionsalfin salfin;run;
data salbis(drop=i);
array predi{8};
array resi{8};
set salfin;
do i=1 to 8;
 resi{i}=(&vardepen-predi{i})**2;
end;
run;
proc means data=salbis;output out=salbos mean=mediaresi1-mediaresi8;
var resi1-resi8;
run;
data fantasma;set fantasma salbos;run;
%end;
/* FIN GRUPOS */
proc means data=fantasma noprint;var mediaresi1-mediaresi8;
output out=mediaresi mean=ase1-ase8 ;
run;
data mediaresi;set mediaresi;semilla=&semilla;run;
data final (keep=ase1-ase8 semilla);set final mediaresi;if ASE1=. then delete;run;
data unifin;set unifin unionsalfin;run;
%end;
proc printto; run;
proc print data=final;run;
%mend;
%cruzadastacksin
(archivo=uno,vardepen=precio,
listclass=AC accesible anfitrion_recibe apto_ninos ascensor balcon banos capacidad grandes_electro
parking permite_mascotas politica_cancelacion superanfitrion tiempo_respuesta tipo,
listconti=accesibilidad antiguedad demanda dis_airp dis_centro puntuacion resenas,
ngrupos=4,seminicio=12345,semifinal=12364,nodos=8,algo=levmar,rediter=100,act=sin,/*red*/
bleafsize=20,iterations=1200,bmaxbranch=2,bmaxdepth=6,shrinkage=0.03/* g boosting*/);
```

8.7.9 Ensamblado en R.

```
archivo<-estdata
vardep<-"precio"
grupos<-4
sinicio<-1234
repe<-20
# Aplicación cruzadas para ensamblar
```

```
medias1<-cruzadalin(data=archivo,vardep=vardep,listconti=listconti, listclass=listclass,  grupos=grupos,
sinicio=sinicio,repe=repe)
medias1bis<-as.data.frame(medias1[1])
medias1bis$modelo<-"regresion"
predi1<-as.data.frame(medias1[2])
predi1$reg<-predi1$pred

medias2<-cruzadaavnnet(data=archivo,vardep=vardep,listconti=listconti, listclass=listclass,
grupos=grupos, sinicio=sinicio,repe=repe,size=c(17),decay=c(0.05),repeticiones=5,itera=100)
medias2bis<-as.data.frame(medias2[1])
medias2bis$modelo<-"avnnet"
predi2<-as.data.frame(medias2[2])
predi2$avnnet<-predi2$pred

medias3<-cruzadarf(data=archivo,vardep=vardep,listconti=listconti,listclass=listclass,
grupos=grupos,sinicio=sinicio,repe=repe,mtry=7,ntree=1000,nodesize=5,replace=TRUE)
medias3bis<-as.data.frame(medias3[1])
medias3bis$modelo<-"rf"
predi3<-as.data.frame(medias3[2])
predi3$rf<-predi3$pred

medias4<-cruzadagbm(data=archivo,vardep=vardep,listconti=listconti,listclass=listclass, grupos  =
grupos, sinicio=sinicio,repe=repe,
n.minobsinnode=20,shrinkage=0.05,n.trees=2500,interaction.depth=2)
medias4bis<-as.data.frame(medias4[1])
medias4bis$modelo<-"gbm"
predi4<-as.data.frame(medias4[2])
predi4$gbm<-predi4$pred

medias5<-cruzadaxgbm(data=archivo,vardep=vardep,listconti=listconti,  listclass=listclass,
grupos=grupos, sinicio=sinicio,repe=repe,min_child_weight=10,eta=0.05,nrounds=500, max_depth=6,
gamma=0,colsample_bytree=0.8,subsample=1,alpha=0,lambda=0,lambda_bias=0)
medias5bis<-as.data.frame(medias5[1])
medias5bis$modelo<-"xgbm"
predi5<-as.data.frame(medias5[2])
predi5$xgbm<-predi5$pred

medias6<-cruzadaSVM(data=archivo,vardep=vardep,listconti=listconti,listclass=listclass,
grupos=grupos, sinicio=sinicio,repe=repe,C=0.5)
medias6bis<-as.data.frame(medias6[1])
medias6bis$modelo<-"svmLinear"
predi6<-as.data.frame(medias6[2])
predi6$svmLinear<-predi6$pred

medias7<-cruzadaSVMpoly(data=archivo,vardep=vardep,listconti=listconti,listclass=listclass,
grupos=grupos, sinicio=sinicio,repe=repe,C=0.01,degree=3,scale=0.1)
medias7bis<-as.data.frame(medias7[1])
medias7bis$modelo<-"svmPoly"
predi7<-as.data.frame(medias7[2])
predi7$svmPoly<-predi7$pred

medias8<-cruzadaSVMRBF(data=archivo,vardep=vardep,listconti=listconti,listclass=listclass,
grupos=grupos, sinicio=sinicio,repe=repe,C=2,sigma=0.05)
medias8bis<-as.data.frame(medias8[1])
medias8bis$modelo<-"svmRadial"
predi8<-as.data.frame(medias8[2])
predi8$svmRadial<-predi8$pred
```

```
#mediasSAS
library(readxl)
mediasSAS <- read_excel("~/PracticaML/cosas varias (ML).xlsx", sheet = "Hoja18")
union1<-rbind(medias1bis,medias2bis,medias3bis,medias4bis,medias5bis,medias6bis,
medias7bis,medias8bis)
union2<-rbind(union1,mediasSAS[1:45,])
par(cex.axis=1)
boxplot(data=union2,error~modelo,col="light blue",ylab="MSE")

# Construcción ensamblados
unipredi<-cbind(predi1,predi2,predi3,predi4,predi5,predi6,predi7,predi8)
# Esto es para eliminar columnas duplicadas
unipredi<- unipredi[, !duplicated(colnames(unipredi))]
# Ensamblados
unipredi$RegNet<-(unipredi$reg+unipredi$avnnet)/2
unipredi$RegRf<-(unipredi$reg+unipredi$rf)/2
unipredi$RegGbm<-(unipredi$reg+unipredi$gbm)/2
unipredi$RegXgb<-(unipredi$reg+unipredi$xgbm)/2
unipredi$RegSvl<-(unipredi$reg+unipredi$svmLinear)/2
unipredi$RegSvp<-(unipredi$reg+unipredi$svmPoly)/2
unipredi$RegSvr<-(unipredi$reg+unipredi$svmRadial)/2
unipredi$NetRf<-(unipredi$avnnet+unipredi$rf)/2
unipredi$NetGbm<-(unipredi$avnnet+unipredi$gbm)/2
unipredi$NetXgb<-(unipredi$avnnet+unipredi$xgbm)/2
unipredi$NetSvl-(unipredi$avnnet+unipredi$svmLinear)/2
unipredi$NetSvp<-(unipredi$avnnet+unipredi$svmPoly)/2
unipredi$NetSvr<-(unipredi$avnnet+unipredi$svmRadial)/2
unipredi$RfGbm<-(unipredi$rf+unipredi$gbm)/2
unipredi$RfXgb<-(unipredi$rf+unipredi$xgbm)/2
unipredi$RfSvl<-(unipredi$rf+unipredi$svmLinear)/2
unipredi$RfSvp<-(unipredi$rf+unipredi$svmPoly)/2
unipredi$RfSvr<-(unipredi$rf+unipredi$svmRadial)/2
unipredi$GbmXgb<-(unipredi$gbm+unipredi$xgbm)/2
unipredi$GbmSvl<-(unipredi$gbm+unipredi$svmLinear)/2
unipredi$GbmSvp<-(unipredi$gbm+unipredi$svmPoly)/2
unipredi$GbmSvr<-(unipredi$gbm+unipredi$svmRadial)/2
unipredi$XgbSvl<-(unipredi$xgbm+unipredi$svmLinear)/2
unipredi$XgbSvp<-(unipredi$xgbm+unipredi$svmPoly)/2
unipredi$XgbSvr<-(unipredi$xgbm+unipredi$svmRadial)/2
unipredi$RegNetRf<-(unipredi$reg+unipredi$avnnet+unipredi$rf)/3
unipredi$RegNetGbm<-(unipredi$reg+unipredi$avnnet+unipredi$gbm)/3
unipredi$RegNetXgb<-(unipredi$reg+unipredi$avnnet+unipredi$xgbm)/3
unipredi$RegRfGbm<-(unipredi$reg+unipredi$rf+unipredi$gbm)/3
unipredi$RegRfXgb<-(unipredi$reg+unipredi$rf+unipredi$xgbm)/3
unipredi$RegGbmXgb<-(unipredi$reg+unipredi$gbm+unipredi$xgbm)/3
unipredi$RegXgbSvl<-(unipredi$reg+unipredi$xgbm+unipredi$svmLinear)/3
unipredi$RegXgbSvp<-(unipredi$reg+unipredi$xgbm+unipredi$svmPoly)/3
unipredi$RegXgbSvr<-(unipredi$reg+unipredi$xgbm+unipredi$svmRadial)/3
unipredi$RfGbmSvl<-(unipredi$rf+unipredi$gbm+unipredi$svmLinear)/3
unipredi$RfGbmSvp<-(unipredi$rf+unipredi$gbm+unipredi$svmPoly)/3
unipredi$RfGbmSvr<-(unipredi$rf+unipredi$gbm+unipredi$svmRadial)/3
unipredi$RfXgbSvl<-(unipredi$rf+unipredi$xgbm+unipredi$svmLinear)/3
unipredi$RfXgbSvp<-(unipredi$rf+unipredi$xgbm+unipredi$svmPoly)/3
unipredi$RfXgbSvr<-(unipredi$rf+unipredi$xgbm+unipredi$svmRadial)/3
unipredi$RfNetGbm<-(unipredi$rf+unipredi$avnnet+unipredi$gbm)/3
unipredi$RfNetXgb<-(unipredi$rf+unipredi$avnnet+unipredi$xgbm)/3
unipredi$RfNetSvl<-(unipredi$rf+unipredi$avnnet+unipredi$svmLinear)/3
unipredi$RfNetSvp<-(unipredi$rf+unipredi$avnnet+unipredi$svmPoly)/3
```

```
unipredi$RfNetSvr<-(unipredi$rf+unipredi$avnnet+unipredi$svmRadial)/3
unipredi$RfXgbGbm<-(unipredi$rf+unipredi$xgbm+unipredi$gbm)/3
unipredi$NetGbmSvl<-(unipredi$avnnet+unipredi$gbm+unipredi$svmLinear)/3
unipredi$NetGbmSvp<-(unipredi$avnnet+unipredi$gbm+unipredi$svmPoly)/3
unipredi$NetGbmSvr<-(unipredi$avnnet+unipredi$gbm+unipredi$svmRadial)/3
unipredi$NetGbmXgb<-(unipredi$avnnet+unipredi$gbm+unipredi$xgbm)/3
unipredi$XgbGbmRf<-(unipredi$xgbm+unipredi$gbm+unipredi$rf)/3
unipredi$XgbNetSvl<-(unipredi$xgbm+unipredi$avnnet+unipredi$svmLinear)/3
unipredi$XgbNetSvp<-(unipredi$xgbm+unipredi$avnnet+unipredi$svmPoly)/3
unipredi$XgbNetSvr<-(unipredi$xgbm+unipredi$avnnet+unipredi$svmRadial)/3
unipredi$RegRfGbmNet<-(unipredi$reg+unipredi$rf+unipredi$gbm+unipredi$avnnet)/4
unipredi$RegRfXgbNet<-(unipredi$reg+unipredi$rf+unipredi$xgbm+unipredi$avnnet)/4
unipredi$XgbRfGbmNet<-(unipredi$xgbm+unipredi$rf+unipredi$gbm+unipredi$avnnet)/4
unipredi$XgbRfGbmSvr<-(unipredi$xgbm+unipredi$rf+unipredi$gbm+unipredi$svmRadial)/4
unipredi$XgbRfNetSvr<-(unipredi$xgbm+unipredi$rf+unipredi$gbm+unipredi$svmRadial)/4
unipredi$GbmRfXgbNetSvr<-
(unipredi$gbm+unipredi$rf+unipredi$xgbm+unipredi$avnnet+unipredi$svmRadial)/5
unipredi$GbmRfXgbNetSvp<-
(unipredi$gbm+unipredi$rf+unipredi$xgbm+unipredi$avnnet+unipredi$svmPoly)/5
unipredi$GbmRfXgbNetSvl<-
(unipredi$gbm+unipredi$rf+unipredi$xgbm+unipredi$avnnet+unipredi$svmLinear)/5
dput(names(unipredi))
# Recorto los modelos de la lista de variables
listado<-c("reg", "avnnet", "rf", "gbm", "xgbm", "svmLinear", "svmPoly", "svmRadial", "RegNet",
"RegRf", "RegGbm", "RegXgb", "RegSvl", "RegSvp", "RegSvr", "NetRf", "NetGbm", "NetXgb",
"NetSvp", "NetSvr", "RfGbm", "RfXgb", "RfSvl", "RfSvp", "RfSvr", "GbmXgb", "GbmSvl", "GbmSvp",
"GbmSvr", "XgbSvl", "XgbSvp", "XgbSvr", "RegNetRf", "RegNetGbm", "RegNetXgb", "RegRfGbm",
"RegRfXgb", "RegGbmXgb", "RegXgbSvl", "RegXgbSvp", "RegXgbSvr", "RfGbmSvl", "RfGbmSvp",
"RfGbmSvr", "RfXgbSvl", "RfXgbSvp", "RfXgbSvr", "RfNetGbm", "RfNetXgb", "RfNetSvl",
"RfNetSvp", "RfNetSvr", "RfXgbGbm", "NetGbmSvl", "NetGbmSvp", "NetGbmSvr", "NetGbmXgb",
"XgbGbmRf","XgbNetSvl", "XgbNetSvp", "XgbNetSvr", "RegRfGbmNet", "RegRfXgbNet",
 "XgbRfGbmNet", "XgbRfGbmSvr", "XgbRfNetSvr", "GbmRfXgbNetSvr", "GbmRfXgbNetSvp",
"GbmRfXgbNetSvl")
repeticiones<-nlevels(factor(unipredi$Rep))
unipredi$Rep<-as.factor(unipredi$Rep)
unipredi$Rep<-as.numeric(unipredi$Rep)
# Calculo el MSE para cada repeticion de validaciÃ³n cruzada
medias0<-data.frame(c())
for (prediccion in listado)
{
  print(prediccion)
  paso <-unipredi[,c("obs",prediccion,"Rep")]
  paso$error<-(paso[,c(prediccion)]-paso$obs)^2
  paso<-paso %>%
    group_by(Rep) %>%
    summarize(error=mean(error))
  paso$modelo<-prediccion
  medias0<-rbind(medias0,paso)
}
# Finalmente boxplot
mediasfinal <- rbind(medias0,mediasSAS)
par(cex.axis=0.5,las=2)
boxplot(data=mediasfinal,outcex=0.3,error~modelo,col="light blue",ylab="MSE")
# Ordeno gráfico
mediasfinal$modelo <- with(mediasfinal,reorder(modelo,error, mean))
par(mar=c(8.5,4,4,2))
par(cex.axis=1,las=2)
boxplot(data=mediasfinal,error~modelo,col="light blue",ylab="MSE")
```

```
# 15 mejores
listadobis<-c("RfXgb", "RfXgbSvr", "RfNetXgb","XgbRfGbmSvr",  "XgbRfNetSvr",
"GbmRfXgbNetSvr",  "RfXgbGbm", "XgbGbmRf","XgbSvr","XgbRfGbmNet", "XgbNetSvr",
"NetXgb", "xgbm",   "gbmSAS", "RfXgbSvp"      )
mediasfinal$modelo<-as.character(mediasfinal$modelo)
mediasver<-mediasfinal[mediasfinal$modelo %in% listadobis,]
mediasver$modelo <- with(mediasver,reorder(modelo,error, mean))
par(cex.axis=1,las=2)
boxplot(data=mediasver,error~modelo,col="light blue",ylab="MSE")

# Calculo R2
paso <-unipredi[1:16632,c("obs","RfXgb","Rep")]
num <- mean((paso[,c("xgbm")]-mean(paso$obs))^2)
den <- mean((paso$obs-mean(paso$obs))^2)
paso$R2 <- num/den
```