Trabajo Fin de Máster

Pedido recomendado. Desarrollo de un modelo de recomendación de pedidos para distribuidores.

Víctor Manuel Valiñas Varela

Máster en Técnicas Estadísticas

Curso 2021-2022

Propuesta de Trabajo Fin de Máster

Título en galego: Orde recomendada. Desenvolvemento dun modelo de recomendación de pedidos para distribuidores.
Título en español: Pedido recomendado. Desarrollo de un modelo de recomendación de pedidos para distribuidores.
English title: Recommended order. Development of an order recommendation model for distributors.
Modalidad: Modalidad B.
Autor/a: Víctor Manuel Valiñas Varela, Universidad de Santiago de Compostela.
Director/a: Alberto Rodríguez Casal, Universidad de Santiago de Compostela.
Tutor/a: Jorge López Muñiz, Hijos de Rivera, S.A.U. (Estrella Galicia).
Breve resumen del trabajo: Desarrollo de un modelo de recomendación de pedidos para distribuidores de Estrella Galicia. Se tendrán en cuenta los históricos a nivel de cliente y producto. El modelo deberá recomendar el pedido a realizar por el distribuidor. Además, se debe realizar prospección de bases de datos de eventos o climatología.
Recomendaciones:
Otras observaciones:

Don Alberto Rodríguez Casal, profesor titular de la Universidad de Santiago de Compostela y don Jorge López Muñiz, Head of Business Analytics de Hijos de Rivera, S.A.U. (Estrella Galicia), informan que el Trabajo Fin de Máster titulado

Pedido recomendado. Desarrollo de un modelo de recomendación de pedidos para distribuidores

fue realizado bajo su dirección por don Víctor Manuel Valiñas Varela para el Máster en Técnicas Estadísticas. Estimando que el trabajo está terminado, dan su conformidad para su presentación y defensa ante un tribunal.

En Santiago de Compostela, a 9 de junio de 2022.

El director:
Don Alberto Rodríguez Casal

El tutor:
Don Jorge López Muñiz

El autor:
Don Víctor Manuel Valiñas Varela

Declaración responsable. Para dar cumplimiento a la Ley 3/2022, de 24 de febrero, de convivencia universitaria, referente al plagio en el Trabajo Fin de Máster (Artículo 11, Disposición 2978 del BOE núm. 48 de 2022), **el autor declara** que el Trabajo Fin de Máster presentado es un documento original en el que se han tenido en cuenta las siguientes consideraciones relativas al uso de material de apoyo desarrollado por otros/as autores/as:

- Todas las fuentes usadas para la elaboración de este trabajo han sido citadas convenientemente (libros, artículos, apuntes de profesorado, páginas web, programas,...)
- Cualquier contenido copiado o traducido textualmente se ha puesto entre comillas, citando su procedencia.
- Se ha hecho constar explícitamente cuando un capítulo, sección, demostración,... sea una adaptación casi literal de alguna fuente existente.

Y, acepta que, si se demostrara lo contrario, se le apliquen las medidas disciplinarias que correspondan.

Índice general

Resumen **IX**

1. Motivación del problema. **1**
1.1. Descripción del problema. . . . 1
1.2. Obtención y descripción de los datos. . . . 3
1.3. Código y software utilizado. . . . 8
1.4. Desarrollo del trabajo. . . . 8

2. Aprendizaje Estadístico. **9**
2.1. Introducción al Aprendizaje Estadístico. . . . 9
2.2. Evaluación del rendimiento del modelo. . . . 13
2.3. Equilibrio entre sesgo y varianza. . . . 19

3. *Random Forest.* **23**
3.1. Métodos basados en árboles. . . . 23
3.2. Introducción al método *Bootstrap*. . . . 31
3.3. *Bagging*. . . . 34
3.4. *Random Forest*. . . . 38

4. Redes Neuronales. **43**
4.1. Introducción a las Redes Neuronales. . . . 43
4.2. Funciones de activación. . . . 47
4.3. Ajuste de Redes Neuronales. . . . 51
4.4. Problemas relacionados con el ajuste de las Redes Neuronales. . . . 53

5. Modelado de los distribuidores. **57**
5.1. Comportamiento de los distribuidores . . . 57
5.2. Conclusiones. . . . 74
5.3. Posibles mejoras en la precisión de nuestros modelos. . . . 75

Bibliografía **77**

Resumen

Resumen en español

El objetivo principal de este trabajo es predecir la cartera de los distribuidores de Tipo B de Estrella Galicia para el próximo pedido. Para ello, hemos realizado predicciones sobre los tres productos más solicitados pertenecientes al grupo de la cerveza. En el Capítulo 1 introducimos y describimos los datos de nuestro problema. En el Capítulo 2 realizamos una introducción teórica a los conceptos generales que debemos de tener en cuenta a la hora de aplicar un método de aprendizaje estadístico. En el Capítulo 3 desarrollamos de forma teórica los métodos basados en árboles de decisión, centrándonos en el método *Random Forest*. Este método se utilizará, posteriormente, en el desarrollo práctico del trabajo. Además, ilustramos los diferentes procedimientos sobre ejemplos sencillos. En el Capítulo 4 se desarrolla el funcionamiento de las Redes Neuronales teniendo en cuenta los diferentes ajustes y funciones de activación. Para concluir, en el Capítulo 5 aplicamos, comentamos e ilustramos las diferentes técnicas de aprendizaje estadístico explicadas anteriormente. Además, exponemos los resultados obtenidos para cuatro distribuidores de Tipo B en distintas zonas geográficas. En nuestro caso hemos seleccionado la Zona A y la Zona B incluyendo una zona de interior y otra de costa en cada una. Finalmente, exponemos algunas sugerencias que podrían ser útiles para mejorar los resultados obtenidos en futuras líneas de trabajo.

English abstract

The main objective of this work is to predict the portfolio of Type B distributors of Estrella Galicia for the next order. To do this, we have made predictions about the three most requested products belonging to the beer group. In Chapter 1 we introduce and describe the data for our problem. In Chapter 2 we make a theoretical introduction to the general concepts that we must take into account when applying a statistical learning method. In Chapter 3 we theoretically develop decision tree-based methods, focusing on the Random Forest method. This method will be used in the practical development of the work. Furthermore, we illustrate the different procedures with simple examples. In Chapter 4 we develop Neural Network methods taking into account the different settings and activation functions. To conclude, in Chapter 5 we apply, discuss and illustrate the different statistical learning techniques previously explained. In addition, we present the results obtained for four Type B distributors in different geographical areas. In our case we have selected Zone A and Zone B including an interior zone and a coastal zone in each one. Finally, we expose some suggestions that could be useful to improve the results obtained in future lines of work.

Capítulo 1

Motivación del problema.

Este proyecto se ha desarrollado bajo la supervisión de Estrella Galicia, la empresa cervecera más importante de Galicia, una de las más importantes en España y probablemente de todo el mundo.

El nombre de la empresa es **Hijos de Rivera, S.L.**, sin embargo es más conocida por el nombre de su cerveza más popular, Estrella Galicia. Fue fundada en el año 1906 por José María Rivera Corral, Estrella Galicia (2022), después de su regreso de la emigración en México. En aquellos tiempos, apostaron por un producto de consumo muy reducido, sin embargo, en la actualidad forma parte de los hábitos de todos los españoles.

A lo largo de su historia, Estrella Galicia ha experimentado un gran crecimiento, tanto a nivel nacional como internacional. Actualmente, cuenta con una amplia cartera de productos como pueden ser la cerveza, el agua mineral, las sidras, los vinos o la sangría, con presencia en todo el territorio nacional y en más de 35 países de todo el mundo. Durante los últimos años, la compañía ha recibido múltiples premios nacionales e internacionales por sus productos.

Debido a esta gran expansión de mercado nacional e internacional, la compañía tiene claro que es necesario reestructurar el negocio y, es por ello que, la previsión de ventas se ha convertido en una necesidad para su departamento de *Business Intelligence* en la gestión de toma de decisiones.

1.1. Descripción del problema.

Antes de nada, es importante destacar que, por petición expresa de la compañía y con el fin de preservar la confidencialidad de los datos utilizados durante el trabajo, se ha decidido enmascarar toda aquella información que pueda resultar de interés comercial.

El objetivo principal de este proyecto consiste en predecir la cartera de productos, para los diferentes distribuidores de Estrella Galicia, en el próximo pedido. Para ello, haremos uso del histórico de pedidos. De esta forma, estaríamos ayudando a la empresa a tener una predicción sobre los diferentes pedidos que realizará cada distribuidor.

En Estrella Galicia, los distribuidores están categorizados en dos grupos bien diferenciados. Por un lado, la compañía cuenta con distribuidores de Tipo A, estos trabajan con el sistema interno de

la empresa y cada vez que registran un pedido, este aparece notificado de forma inmediata en las instalaciones de la fábrica. Por otro lado, cuenta con distribuidores de Tipo B, los cuáles se tienen que registrar en su página web cada vez que quieran realizar un pedido.

Debido a estas diferencias, los distribuidores de Tipo B realizan los pedidos de forma más esporádica, y es por ello que la empresa está interesada en tener una predicción precisa de la cartera de estos distribuidores para el siguiente pedido. Además, la información disponible sobre estos distribuidores es de menor calidad.

Es importante destacar que este problema puede tener una dimensión muy grande, pues en la Zona A, que es donde está la cartera de productos más amplia, basándose en el histórico de pedidos de los distribuidores, supera los 400 productos. Si extendemos nuestro análisis a la Zona B, la cartera más amplia estaría formada por más de 200 productos. Además nos encontramos con una amplia variabilidad en el porfolio de productos de los diferentes distribuidores.

Debido al problema de la dimensión, la falta de información sobre posibles segmentaciones de los distribuidores, más allá de las dos categorías comentadas, la falta de estudios previos sobre qué variables pueden ser más influyentes en el problema a tratar y la gran variabilidad en el tamaño de la cartera de nuestros distribuidores, hemos considerado algunas restricciones a la hora de resolver el problema.

En primer lugar, y atendiendo a las necesidades de la empresa, nos centraremos en analizar solo los distribuidores de Tipo B.

En segundo lugar, hemos decidido analizar solo aquellos distribuidores para los cuáles hayan transcurrido menos de 150 días desde su último pedido. De esta forma, estamos considerando que si no han realizado ningún pedido en este período de tiempo ya no deben de estar trabajando con la empresa, o al menos, que la predicción de un futuro pedido carece de relevancia práctica para Estrella Galicia.

En tercer lugar, para realizar el análisis de la cartera de nuestros distribuidores, solo nos vamos a centrar en los productos que pertenecen al grupo de la cerveza. Es importante destacar, que el único motivo que nos ha llevado a elegir los productos de este grupo es su gran importancia dentro de la compañía.

Sin embargo, a pesar de haber realizado todas estas restricciones, la cartera de productos todavía resultaba demasiado amplia. En consecuencia, y con el objetivo de tener un problema más manejable, hemos realizado un análisis para ver cuales son los tres productos más solicitados dentro de este grupo.

Finalmente, tenemos que en la Zona A los tres productos más demandados son: *Producto 1, Producto 2, Producto 3*, mientras que en la Zona B los más solicitados son: *Producto 3, Producto 4, Producto 1.* En consecuencia, estos serán los cuatro productos que consideraremos a la hora de realizar las predicciones sobre la cartera de los diferentes distribuidores de Tipo B de Estrella Galicia.

No obstante, es importante destacar que sería posible realizar el análisis desarrollado a lo largo de este trabajo para cualquier otro producto en el que estemos interesados, al igual que también se puede modificar la dimensión de la cartera que queremos analizar.

1.2. Obtención y descripción de los datos.

Los datos que vamos a tratar a lo largo de este proyecto se corresponden con el histórico de pedidos para los diferentes distribuidores de Tipo B de Estrella Galicia. Estos datos contienen los pedidos, tanto de productos propios de la marca de Estrella Galicia, como de productos para los cuales Estrella Galicia es la encargada de su distribución, principalmente en la Zona A.

Los datos abarcan el período comprendido entre el mes de Diciembre del año 2017 hasta el 1 de Septiembre del año 2021, fecha en la cual se dio inicio al proyecto. Es muy importante destacar que, durante este período de tiempo, nos hemos enfrentado a una pandemia mundial, originada por el virus SARS-CoV-2, habitualmente conocido como COVID-19.

Durante esta pandemia, declarada de forma oficial por la Organización Mundial de la Salud (OMS) el día 11 de Marzo de 2020 y que se extiende hasta prácticamente la actualidad, en España hemos sufrido un confinamiento estricto de la población, el cual abarca el período comprendido entre el día 15 de Marzo del año 2020 y el día 21 de Junio de 2020, además de diferentes períodos con múltiples restricciones, muchas de las cuales han tenido un carácter local, pudiendo afectar a unos distribuidores concretos y a otros no.

Todos estos períodos de incertidumbre e inestabilidad han tenido una gran repercusión en la hostelería, y como no podía ser menos, también han tenido un gran impacto en los pedidos de los diferentes distribuidores de Estrella Galicia. Por lo tanto, es muy probable que los resultados que obtengamos no sean tan precisos como nos hubiese gustado, pues los datos de los diferentes pedidos están altamente afectados por una situación sanitaria completamente atípica.

Los datos que vamos a tratar son diarios, incluso con múltiples pedidos en el mismo día, pues se registran por separado los diferentes productos solicitados. Para este proyecto se van a utilizar los datos de los diferentes distribuidores de la Zona A, además de los de la Zona B, aunque siempre dentro del canal HORECA (Hoteles, Restaurantes y Cafeterías).

Los datos se pueden agrupar en tres grupos bien diferenciados, como se muestra a continuación:

- **Agua**. En este grupo se encuentran los pedidos de agua, de las cuáles Estrella Galicia es propietaria.
- **Cerveza**. Este grupo comprende el sector principal de Estrella Galicia. En él se encuentran los pedidos de los diferentes tipos de cerveza comercializados por la compañía.
- **Productos distribuidos**. En esta categoría se engloban todos los productos externos a Estrella Galicia pero para los cuales se encarga de su distribución, principalmente en la Zona A. En este grupo podemos encontrar una gran variedad de bebidas de diferentes marcas, como pueden ser: refrescos, bebidas energéticas, zumos, vermuts, etc.

Los datos se procesan y se depuran diariamente para posteriormente ser almacenados en la base de datos SQL Server correspondiente.

De esta forma, nuestro primer objetivo consistirá en obtener los datos almacenados en la plataforma SQL Server. Posteriormente, tendremos que analizarlos, depurarlos y aplicarle los métodos que consideremos apropiados con el fin de predecir la cartera de los distribuidores considerados.

Debido a las restricciones causadas por el COVID-19, este proyecto se llevó a cabo totalmente de forma remota, originándose así una dificultad adicional para tener acceso a los datos, pues el acceso al servidor de la compañía era restringido y no siempre estaba disponible.

De este modo, para poder acceder a los datos, en primer lugar, era necesario conectarse mediante un usuario y contraseña a la VPN (*Virtual Private Network*) de Estrella Galicia y una vez conectados a la red de la compañía, mediante la utilización del escritorio remoto, conectarse al equipo correspondiente haciendo uso del usuario asignado. Una vez conectado al equipo, para poder extraer los datos era necesario conectar el software R al sistema de la base de datos SQL Server. Esta conexión se realizó haciendo uso de la función `odbcDriverConnect` de la librería `RODBC` del software R. Ver Ripley y Lapsley (2021). Para ello, nosotros hemos utilizado el siguiente código de R:

```
library(RODBC)
canal=odbcDriverConnect('driver={SQL Server};
  server="nombre del servidor de la compañia;
  database="nombre de la base de datos;
  trusted_connection=true')
```

Seguidamente, después de conectar el software R con el sistema de la base de datos SQL Server, se procedía a la extracción de los datos. Para ello se utilizó la función `sqlQuery` de la librería `RODBC`, ver Ripley y Lapsley (2021), en la cual hemos introducido el código SQL que nos ha permitido extraer los datos que necesitábamos. Para ello hemos hecho uso del siguiente código de R:

```
pedidos_tipo_B=sqlQuery(canal,"intoducimos la query de SQL necesaria")
```

Una vez habíamos realizado este proceso contábamos con un total de 116 167 filas y 13 columnas para los 24 distribuidores de la Zona A, mientras que para los 268 distribuidores de la Zona B teníamos un total de 400 613 filas y 13 columnas.

De este modo, cada fila de datos hace referencia al pedido de un determinado producto para un distribuidor y cada columna hace referencia a una variable o característica del pedido.

A continuación, vamos a examinar con detalle el significado de cada una de las variables que hemos extraído de la base de datos de la compañía.

- **`Distribuidor ID`**. Esta variable le asigna a cada distribuidor una identidad. Es única para cada distribuidor.
- **`Fecha pedido`**. Esta variable nos proporciona el año, el mes y el día en el que se realizó el pedido. Consta de 8 dígitos, los cuatro primeros indican el año, los dos siguientes indican el mes, y los dos últimos el día.
- **`Dia semana`**. Indica el día de la semana en el cual fue realizado el pedido.

- `Dia mes`. Indica el día del mes en el que se realizó el pedido.
- `Mes`. Indica el mes en el que se realizó el pedido.
- `Ano`. Indica el año del pedido.
- `Material desc`. Esta variable hace referencia al material solicitado en el pedido. Es única para cada producto.
- `Material cod`. Esta variable indica el código numérico que tiene asociado cada producto. Es única para cada producto.
- `Grupo`. Indica a que grupo pertenece el producto solicitado en el pedido. Recordemos que pueden ser productos de carácter propio como el **agua** o la **cerveza**, o productos distribuidos como los **refrescos**, las **bebidas energéticas**, los **zumos**, etc.
- `Ciudad desc`. Indica la ciudad en la cual se realizó el pedido.
- `Provincia desc`. Indica la provincia en la cual se realizó el pedido.
- `Cantidad pedido`. Hace referencia a la cantidad de lotes que se han solicitado para cada producto.
- `Litros pedidos`. Esta variable indica la cantidad de litros totales que se han solicitado para cada producto.

Posteriormente, una vez tenemos almacenados estos datos, nuestro objetivo será limpiar, depurar y tratar el conjunto de datos, pues cuanta más información nos aporte el conjunto de datos mejor será el ajuste y la precisión de los métodos de aprendizaje estadístico utilizados.

Entre los procesos de depuración y limpieza de datos destacamos la eliminación de valores NA's, es decir, valores no válidos. Los valores NA's pueden ser debido a errores de los distribuidores o pueden estar provocados por la falta de datos en los pedidos.

Entre los procesos de tratamiento de datos destaca la creación de nuevas variables que consideramos que pueden ser de gran interés para los modelos de aprendizaje estadístico. A continuación, se muestran y examinan de forma detallada. Para crear estas nuevas variables se ha hecho uso de las siguientes librerías del software R: `lubridate`, Grolemund y Wickham (2011), y `tidyverse`, Wickham *et al.* (2019). Es importante destacar que con la librería `tidyverse` también se carga la librería `dplyr`, Wickham *et al.* (2022), una de las más utilizadas para el tratamiento de datos con el software R.

Las nuevas variables se pueden separar en dos grupos. En un primer grupo podemos incluir todas las variables numéricas. Estas hacen referencia a los litros solicitados en los pedidos previos. Además, incluimos una variable que nos indica el tiempo que ha transcurrido, en días, desde que un distribuidor ha realizado su último pedido. Como hemos comentado anteriormente, no consideraremos a los distribuidores para los cuales hayan transcurrido 150 días o más desde su último pedido.

- **`Litros 1 pedido antes`**, **`Litros 2 pedidos antes`**, **`Litros 3 pedidos antes`**, **`Litros 4 pedidos antes`**, **`Litros 5 pedidos antes`**, **`Litros 6 pedidos antes`**, **`Litros 7 pedidos antes`**.
- **`Dias ultimo pedido`**.

En un segundo grupo podemos considerar todas las variables categóricas que se han creado a partir de los datos.

- **`Estacion`**. Esta variable hace referencia a la estación anual del año en la que nos encontramos en el momento del pedido. Toma 4 valores posibles: Primavera, Verano, Otoño e Invierno.
- **`Mes`**. Esta variable hace referencia al mes del año en el que nos encontramos. Tiene 12 valores diferentes y nos puede servir para detectar estacionalidad en los pedidos.
- **`Semana`**. Indica el número de semana del año en la que se ha realizado el pedido. Tiene 52 valores diferentes y nos puede servir para detectar estacionalidad en el histórico de los pedidos.
- **`Navidad`**. Hace referencia a si el pedido ha sido realizado durante la navidad o no. Es una variable dicotómica que toma el valor 0 si no es navidad y el valor 1 si es navidad. En nuestro caso hemos considerado que la navidad comienza el día 24 de Diciembre y termina el día 6 de Enero.
- **`Covid`**. Indica si el pedido se ha realizado durante alguno de los períodos en los cuales las restricciones ocasionadas por el COVID-19 eran más estrictas. Es una variable dicotómica que tomar el valor 0 si no se trata de un período con restricciones estrictas y el valor 1 si nos encontramos en un período con restricciones estrictas. En nuestro caso hemos considerado dos períodos estrictos diferentes. El primero se corresponde a los días comprendidos entre el 15 de Marzo de 2020 y el 21 de Junio de 2020, mientras que el segundo abarca desde el 25 de Octubre de 2020 hasta el 9 de Mayo de 2021.

A posteriori, en el momento de analizar a los distribuidores en particular, y con el objetivo de seguir mejorando los resultados obtenidos con los métodos de aprendizaje estadístico, se crean nuevas variables

- **`Promedio mes`**. Esta variable hace referencia al promedio del histórico de pedidos por mes.
- **`Promedio semana`**. Esta variable indica el promedio del histórico de pedidos por semana.
- **`Promedio mes distribuidores`**. Esta variable indica el promedio del histórico de todos los distribuidores, excepto el distribuidor que estamos analizando, por mes.
- **`Promedio semana distribuidores`**. Indica el promedio del histórico de todos los distribuidores, excepto el distribuidor que estamos analizando, por semana.

Finalmente, haciendo uso de la base de datos meteorológica y de la base de datos de eventos, ambas elaboradas a lo largo del proyecto, se crean las últimas variables que introduciremos en nuestros modelos de aprendizaje estadístico. Cabe destacar que la base de datos de eventos solo se ha construido para la Zona A. Estas variables son:

- **`Temperatura`**. Esta variable indica la temperatura media de cada provincia por días. Para obtener la temperatura media de cada provincia se ha utilizado el promedio de las diferentes estaciones meteorológicas ubicadas en cada una de las provincias.

- **`Eventos`**. Esta será una variable dicotómica, tomará el valor 0 si no hay evento, y el valor 1 si hay evento.

Para la elaboración de la base de datos meteorológica hemos utilizado la página web de Aemet, Aemet (2021), la descarga de los datos se ha realizado el día 15 de Septiembre del año 2021.

En primer lugar, para tener acceso a los datos meteorológicos hemos tenido que solicitar una clave denominada API Key.

Posteriormente, haciendo uso de la clave facilitada, hemos accedido a la página de Aemet y hemos descargado uno a uno los diferentes archivos "`json`". Estos archivos contenían la información meteorológica de España por estaciones meteorológicas y estaban organizados por meses. Es decir, para cada mes del año hay un archivo.

De esta forma, hemos tenido que descargar 45 archivos "`json`". Luego, haciendo uso de las librerías `jsonlite`, Ooms (2014), y `data.table`, Dowle y Srinivasan (2021), de R, hemos leído y transformado estos archivos en una única tabla que contenga la información meteorológica de España para el período comprendido entre el día 1 de Enero del año 2018 y el día 1 de Septiembre del año 2021. A continuación, a modo de ejemplo, mostramos el código utilizado para obtener la tabla que contenga la información meteorológica de España en el año 2018. Para los demás años se haría de forma análoga.

```
library(jsonlite)
library(data.table)

datos_2018=list.files(path="/Volumes/VICTOR/Practicas Estrella Galicia
  /Datos Meteorologicos/datos_2018",
  pattern = "*.json")
  setwd("/Volumes/VICTOR/Practicas Estrella Galicia/Datos
  ↪  Meteorologicos/datos_2018")
  multiples_json_2018=lapply(datos_2018, fromJSON)
  datos_meteorologicos_2018 <- rbindlist(multiples_json_2018)
```

Para la elaboración de la base de datos de los eventos hemos recolectado de forma manual la información relativa a todos los festivos de la Zona A, las principales fiestas gastronómicas, los principales festivales musicales y los partidos de fútbol, que como bien sabemos, representa el evento deportivo más seguido en toda España, y en particular, en nuestras zonas de interés.

1.3. Código y software utilizado.

El código que hemos utilizado para el desarrollo de este proyecto y mediante el cual se han obtenido los resultados que se ilustran a lo largo de este trabajo, ha sido desarrollado utilizando el software libre R.

R es un entorno de programación formado por un conjunto de herramientas flexibles y que pueden ampliarse fácilmente mediante paquetes, librerías o incluso definiendo nuestras propias funciones. Además, proporciona una gran variedad de métodos estadísticos y técnicas gráficas. Actualmente, es el software de referencia en el ámbito de la estadística.

Es notorio destacar, que para el desarrollo del trabajo se han utilizado numerosos paquetes de R, no obstante, para la redacción del mismo hemos decidido nombrar solo los más relevantes. Estas menciones se realizan en las secciones correspondientes.

En el siguiente enlace de GitHub, `https://github.com/VictorValinas/TFM`, ver Víctor Manuel Valiñas Varela (2022), el lector podrá consultar el código completo desarrollado a lo largo del trabajo.

1.4. Desarrollo del trabajo.

En primer lugar, en el Capítulo 2 se realiza una introducción general a los métodos de aprendizaje estadístico, haciendo hincapié en la evaluación del rendimiento de los modelos y en mantener un buen equilibrio entre el sesgo y la varianza. A continuación, en el Capítulo 3 se desarrollan los métodos basados en árboles de decisión, prestando principal atención a los modelos *Random Forest*. Además, se realiza una pequeña introducción al método *Bootstrap*. Posteriormente, en el Capítulo 4 se realiza una introducción teórica a las Redes Neuronales, incluyendo diferentes funciones de activación y algunos problemas relacionados con el ajuste de estos métodos. Finalmente, en el Capítulo 5 se ilustra, a modo de ejemplo, el funcionamiento de los métodos de aprendizaje estadístico desarrollados durante el trabajo sobre diferentes distribuidores de Tipo B de Estrella Galicia. Además, se incluyen algunas sugerencias que podrían ser útiles para mejorar los resultados obtenidos en futuras líneas de trabajo.

Capítulo 2

Aprendizaje Estadístico.

En este capítulo inicial nos vamos a centrar en introducir los conceptos básicos en los que se basa el marco teórico de los problemas de regresión. Para ello, principalmente se hace uso de James *et al.* (2014) y Hastie *et al.* (2001). En primer lugar, en la Sección 2.1, se realiza una **introducción al aprendizaje estadístico**. Luego, en la Sección 2.2, se expone como evaluar el **rendimiento** de los diferentes **modelos estadísticos**. Finalmente, en la Sección 2.3, se ilustra la importancia de mantener el **equilibrio entre el sesgo y la varianza**.

2.1. Introducción al Aprendizaje Estadístico.

Supongamos que queremos estudiar una variable respuesta cuantitativa Y y tenemos p variables predictoras diferentes, $X_1, \ldots, X_p$. Estamos asumiendo que existe alguna relación entre Y y $X = (X_1, X_2, \ldots, X_p)$. Esta relación se puede escribir generalmente de la siguiente forma:

$$Y = f(X) + \varepsilon. \tag{2.1}$$

En la relación (2.1) f es una función desconocida de $X_1, \ldots, X_p$ y ε es el término de error, que es independiente de X y tiene media cero. Por lo tanto, f representa la información sistemática que X proporciona sobre Y.

En esencia, el aprendizaje estadístico es un conjunto de métodos para estimar f. Hay dos razones principales por las que tenemos interés en estimar f: *inferencia* y *predicción*.

Inferencia.

Cuando estamos interesados en la creación de un modelo estadístico, el interés radica en entender de la forma más sencilla posible, la relación que existe entre la variable respuesta Y y las variables predictoras $X_1, \ldots, X_p$, es decir, estamos interesados en entender como le afectan a la variable respuesta los cambios producidos en las variables predictoras. En este escenario, la estimación de f, denotada por $\hat{f}$, no puede ser tratada como una caja negra, pues necesitamos conocer su expresión exacta. Además, podemos estar interesados en dar respuesta a las siguientes cuestiones:

- ¿Qué predictores están asociados con la variable respuesta?
- ¿Cuál es la relación existente entre la variable respuesta y cada predictor?
- ¿La relación entre la variable respuesta y cada predictor puede ser explicada por un modelo lineal, como por ejemplo un modelo de regresión lineal, o la relación es más compleja?

A continuación nos vamos a centrar en la predicción, pues es el objetivo principal de este trabajo.

Predicción.

En muchos casos, es fácil disponer de un conjunto de datos de entrada X, pero no se puede obtener de forma sencilla la salida Y. En este escenario, dado que el término de error promedia a cero, podemos predecir Y utilizando

$$\hat{Y} = \hat{f}(X), \tag{2.2}$$

donde $\hat{f}$ representa la estimación de f e $\hat{Y}$ representa la predicción de Y. En este caso, $\hat{f}$ es tratada como una caja negra, en el sentido de que no nos preocupa demasiado entender la forma de $\hat{f}$, solo estamos interesados en que produzca predicciones precisas de Y.

Consideremos ahora un ejemplo, supongamos que, $X_1, \ldots, X_p$, son las características de la muestra de sangre de un paciente que pueden ser analizadas en un laboratorio, e Y es una variable que codifica el riesgo de un paciente a sufrir una reacción adversa a un fármaco particular. Es natural tratar de predecir Y utilizando X, pues de esta forma podemos evitar dar el fármaco a pacientes con alto riesgo de reacciones adversas; es decir, pacientes para quienes la estimación de Y es alta.

En general, $\hat{f}$ no será una estimación perfecta para f y esto introducirá errores. La precisión de $\hat{Y}$ como predicción para Y depende de dos cantidades, una que se denominará *error reducible* y otra que se denominará *error irreducible*. Los errores reducibles son aquellos que se pueden mejorar al aumentar la precisión de $\hat{f}$ utilizando las técnicas de aprendizaje estadístico más apropiadas para estimar f. Sin embargo, incluso si nosotros fuésemos capaces a encontrar una estimación perfecta para f, $\hat{Y} = f(X)$, nuestra predicción todavía tendría algún error. Esto se debe a que Y es también una función de ε, y que por definición no puede ser estimado a partir de X.

Por lo tanto, la variabilidad asociada con el término del error afecta a la precisión de nuestras predicciones. Esto es conocido como el *error irreducible*, no importa lo bien que se estime f, ya que el error introducido por ε no se puede reducir.

El *error irreducible* siempre es mayor que cero porque el término ε puede contener variables no medidas pero que son útiles para la estimación de Y. Además, ε también puede contener variaciones no medibles. Por ejemplo, el riesgo de una reacción adversa a un fármaco en un paciente determinado y en un día determinado puede variar dependiendo de la variación de fabricación del fármaco en sí o del bienestar del paciente ese día.

Consideremos ahora una estimación dada $\hat{f}$ y un conjunto de variables predictoras X, dando lugar a la predicción $\hat{Y} = \hat{f}(X)$. Asumamos por un momento que tanto $\hat{f}$ como X son fijos, de modo que la

única variabilidad viene introducida por ε. Entonces, es fácil demostrar que:

$$\begin{aligned}\mathbf{E}(Y-\hat{Y})^2 &= \mathbf{E}\left(f(X)+\varepsilon-\hat{f}(X)\right)^2 = \\ &= \mathbf{E}\left((f(X)-\hat{f}(X))^2+2\varepsilon(f(X)-\hat{f}(X))+\varepsilon^2\right) = \\ &= \mathbf{E}\left((f(X)-\hat{f}(X))^2\right)+2\mathbf{E}\left(\varepsilon(f(X)-\hat{f}(X))\right)+\mathbf{E}(\varepsilon^2) = \\ &= \underbrace{\left(f(x)-\hat{f}(X)\right)^2}_{Reducible}+\underbrace{Var(\varepsilon)}_{Irreducible},\end{aligned} \tag{2.3}$$

donde $\mathbf{E}\left(Y-\hat{Y}\right)^2$ representa el valor esperado de la diferencia al cuadrado entre el valor predicho y el valor real y Var(ε) representa la varianza asociada al término del error ε.

A continuación, nos centraremos en las técnicas de estimación de f con el objetivo de minimizar el *error reducible*. Debemos de tener en cuenta, que el *error irreducible* siempre va a proporcionar un límite superior sobre la precisión de nuestra predicción para Y. En la práctica, este límite es casi siempre desconocido.

Hay diferentes enfoques para estimar la función f. Nosotros siempre vamos a suponer que tenemos un conjunto de n observaciones diferentes. Estas observaciones se denominaran datos de entrenamiento, pues son los datos que utilizaremos parar entrenar, o enseñar, a nuestro modelo como estimar f. Consideraremos x_{ij} el j-ésimo predictor para la observación i, donde $i = 1, 2, \ldots, n$ y $j = 1, 2, \ldots, p$. En consecuencia, y_i representará la i-ésima observación de la variable respuesta. Por lo tanto, nuestros datos de entrenamiento se pueden escribir de la siguiente forma, $\{(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)\}$, donde $x_i = (x_{i1}, x_{i2}, \ldots, x_{ip})^T$.

Nuestro objetivo consiste en aplicar un método de aprendizaje estadístico al conjunto de los datos de entrenamiento para poder estimar la función desconocida f. En general, la mayoría de los métodos de aprendizaje estadístico utilizados para la estimación de la función desconocida f se pueden caracterizar como **métodos paramétricos** o **métodos no paramétricos**. A continuación, detallaremos brevemente éstos dos enfoques.

Métodos paramétricos.

Los métodos paramétricos tienen un enfoque basado en modelos de dos pasos. En primer lugar, tenemos que hacer una suposición sobre la función f. Se supone que f está determinada por una colección de parámetros. Por ejemplo, si $p = 1$, podemos suponer que $f(x) = \beta_0 + \beta_1 X$, siendo β_0 y β_1 sus parámetros desconocidos. De esta forma, una vez que nosotros hemos supuesto una forma paramétrica para f, el problema de la estimación de f ha sido simplificado, pues en general, es más sencillo estimar un conjunto finito de parámetros en vez de una función arbitraria. Después de seleccionar el modelo, necesitaremos un procedimiento que utilice los datos de entrenamiento para ajustar, o entrenar, nuestro modelo. La principal desventaja de los modelos paramétricos radica en que el modelo paramétrico elegido, por lo general, no coincide con la verdadera forma de la función desconocida f, y si el modelo elegido dista mucho de la verdadera función f, nuestra estimación será pobre.

Una posible solución para este problema es tratar de elegir modelos más flexibles y que puedan adaptarse a las diferentes formas funcionales de f. Sin embargo, ajustar modelos más flexibles implica ajustar un mayor número de parámetros y esto puede conducir al sobreajuste de los datos, lo cual significaría que los modelos siguen los errores, o el ruido, demasiado de cerca. Más adelante continuaremos hablando de este fenómeno denominado *overfitting*.

Métodos no paramétricos.

Los métodos no paramétricos no realizan ningún tipo de suposición explícita sobre la forma de la función f. Buscan una estimación de f que se acerque a los datos tanto como sea posible sin llegar a ser demasiado ondulada. Este tipo de enfoque puede tener una gran ventaja sobre el enfoque paramétrico, pues al no suponer ninguna forma particular para f tienen la capacidad de adaptarse con precisión a diferentes funciones posibles. Sin embargo, el enfoque no paramétrico tiene una importante desventaja, ya que no reduce el problema de estimar f y además, necesita muchas más observaciones que cualquier método paramétrico para realizar una estimación precisa de f.

Un ejemplo, en donde se puede ver una comparación entre el funcionamiento de los métodos paramétricos y los métodos no paramétricos, se puede encontrar en la Sección 2.1.2 de James *et al.* (2014).

Precisión vs Interpretabilidad.

A continuación, hablaremos del equilibrio entre la precisión de nuestro modelo y la interpretabilidad, pues parece razonable plantearse la siguiente pregunta: ¿por qué optamos por utilizar un método más restrictivo en lugar de un método más flexible?. Existen varias razones por las cuales muchas veces preferimos utilizar un modelo más restrictivo. Si además de la predicción, también estamos interesados en la interpretabilidad del modelo, los modelos más restrictivos son bastante apropiados, pues nos permiten entender fácilmente la relación existente entre la variable respuesta y las variables predictoras. Por el contrario, los enfoques más flexibles conllevan a estimaciones más complejas de f, por lo que resulta complicado entender como se relaciona de forma individual cada predictor con la variable respuesta. Por lo tanto, acabamos de ver que, cuando el objetivo es el modelado, los métodos de aprendizaje estadístico simples y relativamente inflexibles tienen claras ventajas.

Sin embargo, en algunos casos solo estamos interesados en la predicción, y la interpretabilidad del modelo no nos preocupa. Por ejemplo, si estamos interesados en desarrollar un algoritmo para predecir la cartera de nuestros distribuidores, nuestro objetivo principal radica en la calidad de la predicción, pues en este caso, la interpretabilidad del modelo no es una preocupación.

De este modo, se podría pensar que es mejor utilizar el modelo más flexible posible, pero sorprendentemente no siempre es el caso, pues a menudo se obtienen predicciones más precisas utilizando un método menos flexible, siempre que éste, se ajuste bien a la forma de f. Este fenómeno, que puede parecer contradictorio, es el denominado *overfitting*, y está muy presente en los métodos más flexibles. Más adelante, en la Sección 2.2, veremos reflejado este fenómeno mediante un ejemplo gráfico.

Aprendizaje supervisado vs Aprendizaje no supervisado.

Los problemas de aprendizaje estadístico se dividen en dos categorías: supervisados y no supervisados. Los conceptos que hemos introducido hasta ahora, y los que serán introducidos en lo que sigue, caen todos en el dominio del aprendizaje estadístico supervisado. Para cada observación del predictor x_i, $i = 1, \ldots, n$, hay una respuesta asociada y_i. En este caso, deseamos ajustar un modelo que relacione la variable respuesta con las variables predictoras, con el objetivo de predecir la variable respuesta para futuros valores o comprender mejor la relación entre ambas.

En contraposición, se encuentra el aprendizaje estadístico no supervisado. Éste describe una situación más desafiante, pues para cada observación $i = 1, \ldots, n$, observamos un vector de medidas x_i que no está asociado a la respuesta y_i. En este escenario, de alguna manera, estamos trabajando a ciegas, pues no tenemos una variable respuesta que pueda supervisar nuestro análisis. En nuestro caso particular, podríamos tener muchas variables aportando información sobre los pedidos de los distribuidores sin ninguna variable respuesta asociada. El objetivo podría ser tratar de agregar estos distribuidores en grupos similares, esperando que se comportaran de forma similar en términos de los pedidos. La herramienta estadística que se podría utilizar en este escenario es el *análisis cluster* o *clustering*. El objetivo de este análisis es determinar, sobre la base de $x_1, \ldots, x_n$, si las observaciones caen en los grupos diferenciados. Por lo tanto, este tipo de análisis nos sirve para poder entender las relaciones que existen entre las variables o entre las observaciones.

2.2. Evaluación del rendimiento del modelo.

En el mundo del aprendizaje estadístico es necesario introducir diferentes enfoques de aprendizaje en vez de un solo método, pues como se suele decir, *"There is no free lunch in statistics"*, es decir, ningún método domina a todos los demás sobre todos los conjuntos de datos posibles.

Por lo tanto, dado cualquier conjunto de datos, es importante decidir que método proporciona los mejores resultados. Seleccionar el mejor enfoque puede ser una de las tareas más desafiantes en la práctica del aprendizaje estadístico.

A continuación, vamos a introducir algunos de los conceptos más importantes que surgen a la hora de seleccionar un método de aprendizaje estadístico para un conjunto de datos particular.

Medición de la calidad del ajuste.

Dado un conjunto de datos, para evaluar el comportamiento de un método de aprendizaje estadístico necesitamos medir, de alguna forma, como de buenas son sus predicciones y si realmente coinciden con los datos observados. Es decir, necesitamos cuantificar, en que medida, el valor pronosticado de la variable respuesta, para una observación dada, se acerca al verdadero valor de la variable respuesta. En el contexto de la regresión, la medida más utilizada es el error cuadrático medio (MSE), que viene dado por la siguiente expresión:

$$\text{MSE} = \frac{1}{n} \sum_{i=1}^{n} \left(y_i - \hat{f}(x_i) \right)^2, \tag{2.4}$$

donde $\hat{f}(x_i)$ es la predicción dada por $\hat{f}$ para la observación i-ésima. De este modo, el error cuadrático medio será pequeño si las predicciones de la variable respuesta están muy cerca del verdadero valor y será grande si para algunas observaciones, el valor de la predicción y el valor verdadero se diferencian de forma considerable.

El MSE dado en (2.4) se calcula con los datos de entrenamiento que se han utilizado para ajustar el modelo, por lo tanto, si queremos hablar con mayor precisión deberíamos denominarlo MSE de entrenamiento. Sin embargo, en general, no nos importa como de bien funciona el método sobre los datos de entrenamiento, pues estamos interesados en la precisión de las predicciones cuando aplicamos nuestro método a un conjunto de datos que no ha sido utilizado para entrenar a nuestro modelo.

Reflejemos esto en nuestro caso particular. Supongamos que tenemos datos sobre los pedidos de nuestros distribuidores. De esta forma, podemos entrenar el modelo utilizando la información sobre los pedidos de los últimos tres años, aunque realmente no nos importa como de bien predice los pedidos en ese período de tiempo, sino que estamos preocupados por como de bien va a predecir los pedidos en el futuro. Es decir, no nos interesa mucho si el método predice con precisión los pedidos utilizados para entrenar el modelo, pues en esos casos ya conocemos toda la información sobre los pedidos.

A continuación, vamos a expresar esto en un lenguaje más general. Supongamos que ajustamos nuestro modelo de aprendizaje estadístico con las muestras de entrenamiento $\{(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)\}$, y obtenemos la estimación $\hat{f}$. Entonces, podemos calcular $\hat{f}(x_1), \hat{f}(x_2), \ldots, \hat{f}(x_n)$, y si estos valores son aproximadamente iguales a $y_1, y_2, \ldots, y_n$, el valor del MSE de entrenamiento dado por la expresión (2.4) será pequeño. Sin embargo, en realidad no nos interesa saber si $\hat{f}(x_i) \approx y_i$, sino que queremos saber si $\hat{f}(x_0)$ es aproximadamente igual a y_0, donde el par (x_0, y_0) es una observación nueva que no se ha utilizado para entrenar el método de aprendizaje estadístico. De esta forma, nosotros estamos buscando el método que proporcione el MSE más bajo, en lugar del MSE de entrenamiento más bajo. En otras palabras, si tuviésemos un amplio conjunto de observaciones nuevas, denominado conjunto de validación o test, $\{(x_{t_1}, y_{t_1}), \ldots, (x_{t_m}, y_{t_m})\}$, podríamos calcular el promedio del error cuadrático medio para estas observaciones. Esto es lo que se denomina MSE de test (TMSE).

Supongamos que tenemos m observaciones en el conjunto de validación, entonces el TMSE viene dado por,

$$\text{TMSE} = \frac{1}{m} \sum_{i=1}^{m} \left(\hat{f}(x_{t_i}) - y_{t_i} \right)^2 . \tag{2.5}$$

De esta forma, nuestro objetivo consiste en seleccionar el modelo que nos proporcione el menor TMSE posible utilizando la expresión (2.5).

Llegados a este punto, nos podemos plantear la siguiente cuestión: ¿cómo podemos seleccionar un método que minimice el TMSE? En algunos entornos podemos tener un conjunto de test disponible, es decir, podemos tener acceso a un conjunto de observaciones que no se han utilizado para entrenar nuestro modelo. De esta forma, podemos evaluar (2.5) solo en el conjunto de test y seleccionar el método de aprendizaje estadístico que nos proporcione el valor de TMSE más pequeño. Pero, ¿qué pasa si no hay observaciones de prueba disponibles? En este caso, uno tendría que seleccionar el método de aprendizaje estadístico que minimice el valor de MSE de entrenamiento dado por (2.4). Esto parece

que podría ser un enfoque bastante sensato, ya que el MSE del conjunto de entrenamiento y el MSE del conjunto de test parecen estar muy relacionados.

Desafortunadamente, existe un problema fundamental con esta estrategia, pues no hay ninguna garantía de que el método que nos proporciona el MSE de entrenamiento más bajo nos proporcione también el TMSE más bajo. En general, el problema es que muchos de los métodos estadísticos estiman específicamente los coeficientes para minimizar el MSE del conjunto de entrenamiento, y por lo tanto, para estos métodos, aunque el MSE del conjunto de entrenamiento pueda ser bastante pequeño, el MSE del conjunto de test suele ser mucho más grande. Para ver diferentes ejemplos sobre este fenómeno se puede consultar la Sección 2.2.1 de James *et al.* (2014).

En este punto, cabe destacar una propiedad fundamental del aprendizaje estadístico que se mantiene siempre, independientemente del conjunto de datos y del método que se este utilizando: *a medida que se incrementa la flexibilidad del modelo, el MSE de entrenamiento disminuirá.* Sin embargo, el MSE para el conjunto de test puede no hacerlo. Cuando ocurre este fenómeno, es decir, cuando el método que estamos utilizando produce un MSE de entrenamiento relativamente pequeño, pero un MSE de test grande, se dice que estamos sobreajustando los datos de entrenamiento, es decir, nos encontramos con el ya mencionado *overfitting*.

Este fenómeno sucede porque nuestro modelo de aprendizaje estadístico se centra demasiado en encontrar patrones en los datos de entrenamiento y puede que detecte algunos que sean fruto de la casualidad, en vez de ser causados por verdaderas propiedades de la función desconocida f. Cuando sobreajustamos los datos de entrenamiento, el valor del MSE en el conjunto de test será muy grande, pues los supuestos patrones que el método había detectado en los datos de entrenamiento no existen en los datos de validación.

A pesar de todo, debemos tener en cuenta que, independientemente de que haya ocurrido o no un sobreajuste, casi siempre se espera que el valor del MSE en el conjunto de entrenamiento sea más pequeño que el obtenido sobre el conjunto de test, ya que la mayoría de los métodos de aprendizaje estadístico buscan minimizar el error cuadrático medio del conjunto de entrenamiento, bien sea de forma directa o indirecta. El fenómeno del sobreajuste se refiere específicamente al caso en el que un modelo menos flexible habría producido un valor de TMSE más pequeño.

A continuación, ilustraremos con un ejemplo todos los conceptos que acabamos de ver. Para ello vamos a simular 200 observaciones siguiendo el modelo descrito en (2.1), para simplificarlo consideraremos que X es unidimensional.

Las simulaciones se llevarán a cabo bajo las siguientes hipótesis:

- $X \sim U(0,1)$
- La verdadera f viene dada por $f(X) = X$.
- $\varepsilon \sim N(0, \frac{1}{20})$.

De esta forma, tenemos que nuestro conjunto de entrenamiento estará formado por las observaciones simuladas, es decir, por el conjunto $\{(x_i, y_i)\}_{i=1}^{200}$. Sobre éste vamos a ajustar dos modelos diferentes,

en primer lugar, un modelo de regresión lineal, y en segundo lugar, un spline suavizado, este método se puede consultar en la Sección 7.5 de James *et al.* (2014). Es importante destacar, que los splines pertenecen al conjunto de los métodos no paramétricos, y bajo ciertas restricciones de suavidad minimizan el MSE dado en (2.4). Dicha suavidad se mide a través de los grados de libertad del spline. En términos generales, cuanto mayor sea el número de grados de libertad, más rugosa será su apariencia. El código de R necesario para realizar estos ajustes se puede consultar, respectivamente, en la Sección 6.1 y 8.7 de Venables y Ripley (2002a)

Debemos de notar que el número máximo de grados de libertad que se puede considerar en un spline, como el que vamos a ajustar, viene acotado por el tamaño de la muestra de entrenamiento, pues el número máximo no puede superar el tamaño de la muestra. En nuestro caso particular, el número máximo de grados de libertad que podemos considerar es 200. Se puede ver una explicación completa sobre el spline suavizado en la Sección 7.5 de James *et al.* (2014).

En nuestro ejemplo, hemos decidido ajustar un spline con 75 grados de libertad, mediante el cual, esperamos obtener una curva bastante aproximada a los datos, a diferencia de lo que se espera obtener con el modelo de regresión lineal. El diagrama de dispersión de los datos de entrenamiento se puede ver en la Figura 2.1, en él también se incluyen el ajuste de regresión lineal y el ajuste spline.

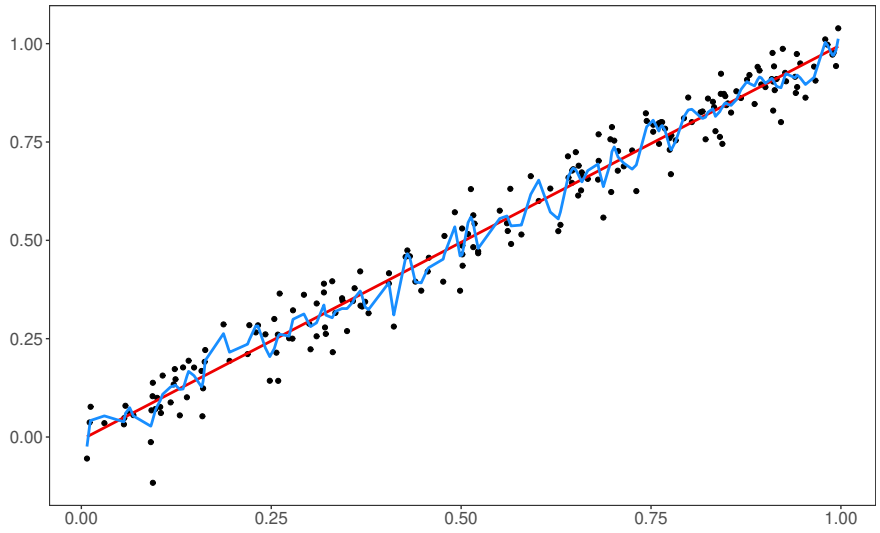


Figura 2.1: Los puntos representan las 200 simulaciones del modelo (2.1), con $X \sim U(0,1)$, $f(X) = X$ y $\varepsilon = N(0, \frac{1}{20})$. La recta roja representa el ajuste de regresión lineal. La curva azul representa el ajuste de suavización spline con 75 grados de libertad.

La recta roja es el ajuste de regresión lineal, que vemos que es inflexible. Sin embargo, capta adecuadamente la tendencia de la nube de puntos. La curva azul es el ajuste de suavización spline. En este caso particular, se puede ver que el modelo más flexible ajusta mejor los datos, de hecho, la curva azul coincide bastante bien con nuestros datos de entrenamiento. Cabe destacar que si ajustásemos diferentes niveles de flexibilidad para el ajuste spline produciríamos diferentes ajustes para estos datos.

Calculemos ahora el MSE de entrenamiento dado por (2.4) para cada uno de los ajustes. En este caso, con el ajuste de regresión lineal obtenemos un MSE igual a 0.0029. Sin embargo, en el caso del ajuste de suavización spline se obtiene un MSE igual a 0.0018. De esta forma, vemos que el MSE de entrenamiento disminuye para el ajuste de suavización spline. Este hecho era de esperar, pues como se puede ver en la Figura 2.1 la curva azul se aproxima mucho más a los puntos que la línea roja.

En este ejemplo, nosotros ya conocemos la verdadera función f, por lo que también podemos calcular el MSE sobre un conjunto de test. Para ello vamos a simular 5000 observaciones bajo las condiciones anteriores. En la Figura 2.2 se puede ver el diagrama de dispersión de estas nuevas observaciones junto con los dos ajustes realizados anteriormente. Ahora la situación parece diferente. Mientras que la recta de regresión parece que proporciona un ajuste razonable, vemos como la curva azul, que representa el spline de suavización, es un ajuste inapropiado para este nuevo conjunto de datos. Por lo tanto, vemos que estamos ante una clara situación de sobreajuste en los datos de entrenamiento, pues la curva azul encontró patrones imaginarios causados por el ruido, es decir, por ε en la expresión (2.1), y ahora está pagando el precio de ese sobreajuste cuando se enfrenta a un nuevo conjunto de datos.

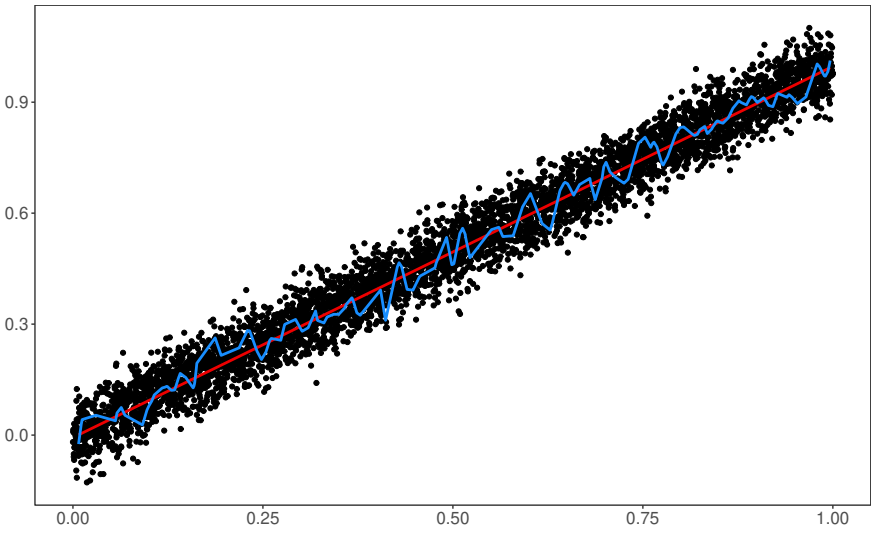


Figura 2.2: Los puntos representan las 5000 simulaciones del modelo (2.1), con $X \sim U(0,1)$, $f(X) = X$ y $\varepsilon = N(0, \frac{1}{20})$. La recta roja representa el ajuste de regresión lineal de la Figura 2.1. La curva azul representa el ajuste de suavización spline con 75 grados de libertad de la Figura 2.1

El MSE de test para la recta de regresión es de 0.0026, incluso un poco por debajo del obtenido con el conjunto de entrenamiento. Sin embargo, el ajuste de suavización spline tiene ahora un valor de MSE de 0.0040, superior al valor obtenido en el caso anterior. De esta forma, las Figuras 2.1 y 2.2 muestran que los métodos más flexibles no significan mejores resultados en términos el MSE en el conjunto de test.

Normalmente, en la práctica es muy sencillo de calcular el MSE sobre el conjunto de entrenamiento, sin embargo, estimar el MSE sobre un conjunto de test es bastante más complicado, pues normalmente no se dispone de un conjunto de validación o test. Para solucionar este inconveniente es habitual dividir

aleatoriamente el conjunto de todas las observaciones en dos conjuntos, un conjunto de entrenamiento, que será el que utilicemos para entrenar a nuestro modelo, y un conjunto de validación o test, para calcular el TMSE. Esto es lo que utilizaremos en todos los análisis prácticos y se conoce como el enfoque del conjunto de validación.

De esta forma, nuestro modelo se ajustará utilizando el conjunto de entrenamiento y posteriormente, el modelo ajustado se utilizará para predecir la variable respuesta de las observaciones del conjunto de validación. La tasa de error sobre el conjunto de validación, que normalmente se evalúa mediante el error cuadrático medio (MSE), proporciona una estimación de la tasa de error de nuestro modelo. Aunque este es el enfoque que más se utiliza en el contexto del aprendizaje estadístico, somos conscientes de que pueden surgir algunos inconvenientes.

Si repetimos el proceso de dividir aleatoriamente el conjunto de observaciones, cada vez obtendremos una estimación ligeramente diferente para el MSE en el conjunto de test y por lo tanto:

- La estimación de la tasa de error en el conjunto de validación puede ser muy variable, dependiendo de las observaciones que se consideren en el conjunto de entrenamiento y de cuales se consideren en el conjunto de test.

- En el enfoque del conjunto de validación solo se utiliza un subconjunto de todas las observaciones para ajustar el modelo, es decir, solo se utilizan aquellas que están incluidas en el conjunto de entrenamiento.

Para resolver estos problemas se podría considerar el conocido enfoque de validación cruzada. A continuación, vamos a introducir brevemente este método, aunque es importante destacar que no lo consideraremos para el algoritmo general, pues en nuestro caso carece de sentido, ya que se perdería la dependencia temporal presente en los pedidos.

Enfoque de validación cruzada.

Este es un enfoque que implica dividir aleatoriamente el conjunto de observaciones en k grupos, aproximadamente del mismo tamaño. El primer grupo es el que se utilizará como conjunto de validación y el modelo se ajustará a los $k-1$ grupos restantes. De esta forma, el TMSE, que denotaremos por MSE_1, se calcula sobre las observaciones del grupo de validación. Este proceso proporciona k estimaciones del error, $\text{MSE}_1, \ldots, \text{MSE}_k$. Finalmente, la estimación dada por el enfoque de validación cruzada se calcula promediando estos k valores,

$$CV_{(k)} = \frac{1}{k} \sum_{i=1}^{k} \text{MSE}_i. \tag{2.6}$$

En la práctica, cuando se utiliza el método de validación cruzada se suele utilizar $k = 5$ o $k = 10$ para evitar problemas computacionales. Se puede ver una explicación más detallada del método de validación cruzada en la Sección 7.10 de Hastie *et al.* (2001).

2.3. Equilibrio entre sesgo y varianza.

A pesar de que no vamos a entrar en el detalle de la prueba matemática, pues se puede ver en la Sección 7.3 de Hastie *et al.* (2001), es fácil ver que, para un valor dado x_0, el MSE esperado del conjunto de entrenamiento siempre se puede descomponer en la suma de tres cantidades fundamentales: la varianza de $\hat{f}(x_0)$, el sesgo cuadrado de $\hat{f}(x_0)$ y la varianza del término del error. Esto es,

$$\mathbf{E}(y_0 - \hat{f}(x_0))^2 = \text{Var}(\hat{f}(x_0)) + \left[\text{Sesgo}(\hat{f}(x_0))\right]^2 + \text{Var}(\varepsilon). \tag{2.7}$$

Con esta notación $\mathbf{E}(y_0 - \hat{f}(x_0))^2$ define el MSE esperado para el conjunto de validación y se refiere al promedio del MSE de test que obtendríamos si estimamos repetidamente f utilizando una gran cantidad de conjuntos de entrenamiento y validamos cada uno con la observación x_0. El TMSE general esperado se puede calcular promediando $\mathbf{E}(y_0 - \hat{f}(x_0))^2$ sobre todos los valores posibles de x_0 en el conjunto de validación.

De esta forma, lo que nos dice la ecuación (2.7) es que para minimizar el error esperado necesitamos seleccionar un método de aprendizaje estadístico que consiga, de forma simultánea, bajo sesgo y baja varianza. Debemos de tener en cuenta que tanto la varianza como el sesgo al cuadrado siempre son cantidades no negativas. Por lo tanto, el MSE de test esperado nunca puede estar por debajo de $\text{Var}(\varepsilon)$, que ya vimos anteriormente en la expresión (2.3) que coincide con el error irreducible.

Dentro de los métodos de aprendizaje estadístico, la varianza se refiere a la cantidad en la que cambiaría $\hat{f}$ si la estimásemos utilizando otro conjunto de datos de entrenamiento. Dado que estos datos se utilizan para ajustar el método de aprendizaje estadístico, diferentes conjuntos de datos darán como resultado diferentes estimaciones de f. Idealmente, la estimación de f no debería de variar demasiado entre los conjuntos de entrenamiento. Sin embargo, si un método tiene una varianza alta, pequeños cambios en los datos de entrenamiento pueden resultar en grandes cambios en la estimación de f. En general, los métodos estadísticos más flexibles tienen una mayor varianza.

Consideremos, por ejemplo, las curvas de la Figura 2.1. En este gráfico se puede ver que la curva azul, correspondiente con el ajuste de suavización spline, es bastante flexible, pues sigue muy de cerca las observaciones. Por lo tanto, es esperable que tenga una varianza alta, pues el cambio de cualquiera de los datos puede provocar que la estimación de f cambie considerablemente. Por el contrario, la línea roja, correspondiente con el ajuste de regresión lineal, es relativamente inflexible y por lo tanto tiene una varianza baja, ya que mover una sola observación, sin cambiar el modelo, probablemente solo cause un ligero cambio en la posición de la recta.

Ahora vamos a explicar lo que se entiende por sesgo en el contexto de los métodos de aprendizaje estadístico. El sesgo es el error que se introduce al aproximar un problema de la vida real, que puede ser extremadamente complejo, por un modelo más simple. Por ejemplo, la regresión lineal clásica asume que existe una relación lineal entre la variable respuesta y los predictores. Sin embargo, es poco probable que cualquier problema de la vida real tenga una relación lineal tan simple, por lo que al aplicar un modelo de regresión lineal surgirá algún sesgo en la estimación de f. En general, los métodos más flexibles dan como resultado menos sesgo.

Ilustremos ahora este concepto sobre un ejemplo. En nuestro caso particular, como la verdadera f es lineal, si tenemos suficientes datos debería ser posible que la regresión lineal produzca una estimación bastante precisa. Esto se puede ver de forma gráfica en las Figuras 2.1 y 2.2. Sin embargo, si en el ejemplo anterior reemplazamos f por $f(X) = X^4$, que es sustancialmente no lineal, no importa cuantos datos de entrenamiento tengamos, pues no será posible realizar una estimación precisa utilizando un ajuste de regresión lineal. Por lo tanto, la regresión lineal daría como resultado un alto sesgo. Este hecho se puede ver en la Figura 2.3.

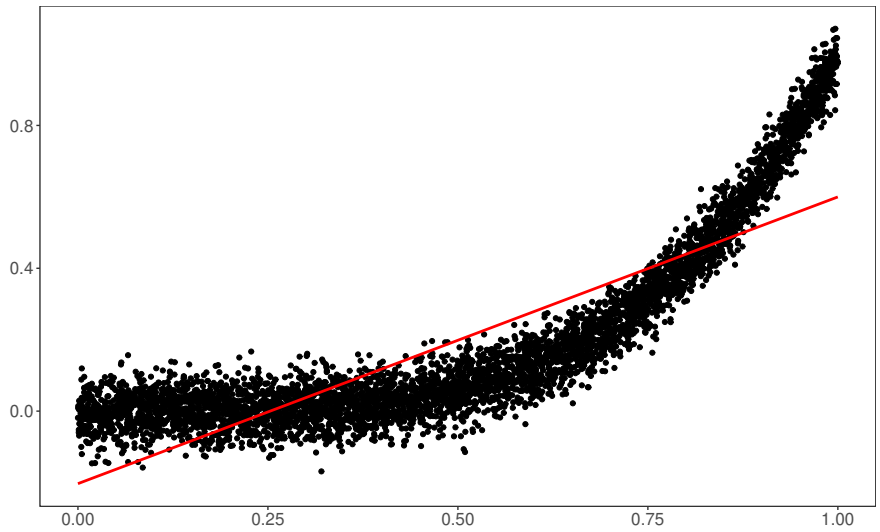


Figura 2.3: Los puntos representan las 5000 simulaciones del modelo (2.1), con $X \sim U(0,1)$, $f(X) = X^4$ y $\varepsilon = N(0, \frac{1}{20})$. La recta roja representa el ajuste de regresión lineal.

Como norma general, a medida que utilizamos métodos más flexibles, la varianza aumentará y el sesgo disminuirá. La tasa relativa de cambio de estas dos cantidades determina si el MSE del conjunto de validación aumenta o disminuye. A medida que aumentamos la flexibilidad de una clase de métodos, el sesgo tiende a disminuir inicialmente más rápido de lo que aumenta la varianza. En consecuencia, el MSE esperado sobre el conjunto de test disminuye. Sin embargo, en algún momento, aumentar la flexibilidad tiene poco impacto en el sesgo y comienza a aumentar la varianza de forma significativa, en el momento que sucede esto, el MSE de test aumenta. Este hecho se ilustra con varios conjuntos de datos en la Sección 2.2.2 de James *et al.* (2014).

La relación entre el sesgo, la varianza y el MSE del conjunto de validación, dada en la ecuación (2.7), se conoce como el compromiso entre el sesgo y la varianza. Para que un método de aprendizaje estadístico tenga un buen rendimiento sobre el conjunto de entrenamiento requiere una varianza baja y un sesgo cuadrático bajo.

Esto se conoce como compromiso porque es fácil obtener un método que tenga un sesgo extremadamente bajo pero una varianza muy alta. Para ello basta con dibujar una curva que pase por cada observación del conjunto de entrenamiento. Del mismo modo, es sencillo encontrar un método que presente una varianza muy baja pero con un sesgo elevado, basta con ajustar una recta horizontal al

conjunto de los datos. Por tanto, el desafío radica en encontrar un método para el cual tanto la varianza como el sesgo al cuadrado sean bajos. Este compromiso es una de las cuestiones más importantes en el aprendizaje estadístico.

Es importante destacar que, en una situación de la vida real en la que desconocemos f, en general no es posible calcular de forma explícita el MSE de test, el sesgo o la varianza para un método de aprendizaje estadístico. Sin embargo, siempre se debe de tener en cuenta el compromiso entre el sesgo y la varianza. Ya que, podemos considerar métodos extremadamente flexibles y que esencialmente eliminen el sesgo, pero esto no nos garantiza que vayan a superar a un método más simple como puede ser la regresión lineal. A continuación, lo ilustraremos con un ejemplo gráfico.

Supongamos que la verdadera f es lineal. En esta situación, la regresión lineal no tendrá ningún sesgo y por lo tanto será muy complicado para cualquier método flexible competir con esta. Este hecho ya se vio en la Figura 2.1. Por el contrario, si la verdadera f dista mucho de ser una función lineal y tenemos un conjunto amplio de observaciones, un enfoque flexible funcionará mejor. Esto se puede ver reflejado en la Figura 2.4. En dicha Figura, la verdadera f es, $f(X) = X^4$, la recta roja representa el ajuste de regresión lineal y la curva azul representa el ajuste de suavización spline con 75 grados de libertad.

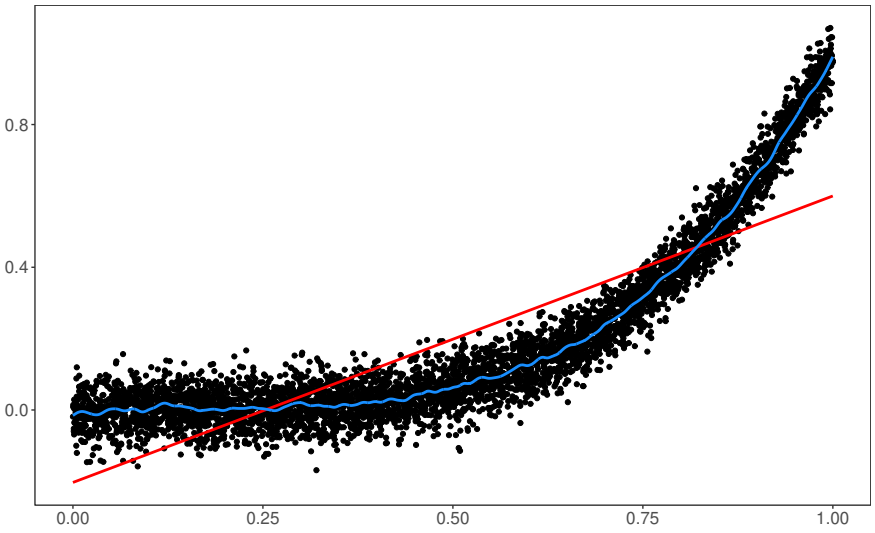


Figura 2.4: Los puntos representan las 5000 simulaciones del modelo (2.1), con $X \sim U(0,1)$, $f(X) = X^4$ y $\varepsilon = N(0, \frac{1}{20})$. La recta roja representa el ajuste de regresión lineal. La curva azul representa el ajuste de suavización spline con 75 grados de libertad.

En este caso, se puede ver como el ajuste de regresión lineal no es capaz a captar de forma adecuada la tendencia de la nube de puntos. Esto nos indica que la verdadera f no es lineal, como ya sabíamos de antemano. En la práctica, esta información es totalmente desconocida, pues se desconoce la verdadera función f. Sin embargo, se puede ver que el ajuste de suavización spline es capaz de captar perfectamente la tendencia de nuestra nube de puntos.

Capítulo 3

Random Forest.

En este capítulo vamos a realizar una introducción teórica a los métodos *Random Forest*. Para ello, principalmente se hace uso de James *et al.* (2014). Estos métodos se utilizarán en el Capítulo 5 durante el desarrollo práctico del trabajo.

Los métodos ***Random Forest***, o más conocidos como árboles aleatorios, son métodos que destacan por su sencilla implementación práctica y el buen comportamiento ante diversas situaciones. Además, en los últimos años han conseguido un gran impulso debido a los avances computacionales.

Estos métodos pertenecen a una amplia familia, habitualmente conocida como métodos basados en árboles. Por este motivo, y para poder tener una mayor comprensión de los *Random Forest*, consideramos adecuado realizar una introducción completa de esta familia de métodos. En la Sección 3.1 hacemos una introducción a los **métodos basados en árboles de decisión**. En la Sección 3.2 realizamos una breve introducción al **método *Bootstrap***. En la Sección 3.3, introducimos el **método *Bagging***. Finalmente, en la Sección 3.4 realizamos la introducción de los métodos ***Random Forest***.

3.1. Métodos basados en árboles.

En general, utilizar los métodos basados en árboles para regresión implica estratificar el espacio de las variables predictoras en varias regiones simples. De esta forma y considerando la variable respuesta como numérica, si queremos realizar una predicción para una observación dada, normalmente utilizaremos la media o la moda de las observaciones de entrenamiento, pertenecientes a la misma región que la observación que estamos tratando. El conjunto de reglas que se utiliza para estratificar el espacio predictor se puede resumir en un árbol.

Los métodos basados en árboles son simples y resultan muy útiles a la hora de realizar interpretaciones, sin embargo, no compiten con los mejores enfoques de aprendizaje supervisado en términos de precisión en la predicción. Por este motivo, surgen otros enfoques que implican la elaboración de múltiples árboles que finalmente se combinan para dar lugar a una sola predicción. Esta combinación de árboles produce una mejora significativa de la predicción, pero conlleva un alto coste en la pérdida de interpretación.

Los métodos basados en árboles pueden ser aplicados tanto a problemas de regresión como de clasificación, en este caso, nosotros solo consideraremos el caso de los problemas de regresión, pues son los que nos interesan para resolver nuestro problema.

Para motivar los árboles de regresión, vamos a empezar con un ejemplo sencillo aplicado a nuestros datos particulares.

Supongamos que queremos predecir el próximo pedido de uno de nuestros distribuidores para un producto particular. Para ello solo vamos a utilizar los litros de su último pedido, los litros de hace dos pedidos y la estación anual en la que nos encontramos. De esta forma, podemos ver que la variable predictora **`Litros 1 pedido antes`** y **`Litros 2 pedidos antes`** son variables numéricas, mientras que la variable **`Estación`** es una variable categórica que puede tomar cuatro valores posibles: **`Primavera`**, **`Verano`**, **`Otoño`** e **`Invierno`**. En la Figura 3.1 se puede ver un árbol de regresión ajustado para estos datos particulares.

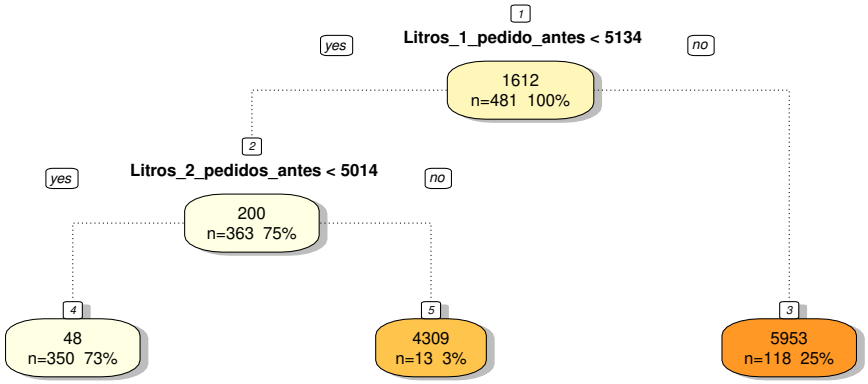


Figura 3.1: Árbol de regresión para un distribuidor de tipo B. El valor que se muestra en la parte superior de cada recuadro hace referencia a la predicción de litros del producto que estamos analizando para el próximo pedido. Los valores inferiores hacen referencia a la cantidad de datos pertenecientes a cada grupo. El valor de la izquierda viene dado en término absoluto, mientras que el de la derecha está dado en porcentaje.

Para la elaboración de la Figura 3.1 hemos utilizado la función `fancyRpartPlot` del paquete `rattle` de R, ver Williams (2011). A continuación, mostramos el código, aunque hemos omitido algunos parámetros de estilo.

```
fancyRpartPlot(rpart(Pedido~Litros_1_pedido_antes +
  Litros_2_pedidos_antes + Estacion,
  data = datos))
```

Este método consiste en una sucesión de reglas de división. En la parte superior del árbol, podemos ver como las observaciones que tienen una cantidad de litros en el pedido anterior inferior a 5134 litros son asignadas a la rama izquierda del árbol. Sin embargo, las observaciones que superan los 5134 en el pedido anterior, son asignadas a la rama derecha. Además el pedido previsto para este caso es de 5953 litros, que se corresponde con el valor medio de la variable respuesta para los pedidos que no cumplen la condición **`Litros 1 pedido antes`** < 5134. Asimismo, las observaciones que han sido asignadas a la rama izquierda, es decir, las observaciones que cumplen **`Litros 1 pedido antes`** < 5134 se vuelve a subdividir en función de la variable **`Litros 2 pedidos antes`**. En este caso, vemos que si esta cantidad no supera los 5014 litros, las observaciones se asignan a una rama izquierda y se estima un pedido de 48 litros, mientras que en el caso de que supere los 5014 litros se asignan a la rama derecha y se estima una cantidad de pedido igual a 4309 litros. Nuevamente, debemos destacar que estas cantidades estimadas se corresponden con la media de las observaciones que verifican las dos condiciones impuestas hasta ese momento. Además, cabe notar, que si tenemos en cuenta la cantidad de litros pedidos en nuestros datos, podríamos decir que una estimación de 48 litros podría ser considerada como un pedido nulo, es decir, que para el próximo pedido no se solicitará ningún litro del producto que estamos analizando.

De esta forma, vemos como el árbol estratifica los pedidos en tres regiones del espacio predictor que se pueden escribir matemáticamente de la siguiente manera:

- $R_1 = \{X | \texttt{Litros 1 pedido antes} > 5134\}$
- $R_2 = \{X | \texttt{Litros 1 pedido antes} < 5134, \texttt{Litros 2 pedidos antes} < 5014\}$
- $R_3 = \{X | \texttt{Litros 1 pedido antes} < 5134, \texttt{Litros 2 pedidos antes} > 5014\}$

En la Figura 3.2 se pueden ver estas tres regiones de forma gráfica. En la esquina inferior izquierda vemos la región R_2, en la esquina superior izquierda observamos la región R_3 y en la parte derecha de la imagen se encuentra la región R_1. Para su elaboración hemos utilizado las librerías `parsnip`, Kuhn y Vaughan (2022), y `parttree`, McDermott (2022), de R. A continuación, mostramos el código de R, aunque cabe destacar que se han omitido ciertos parámetros relativos al estilo del gráfico para evitar que este resulte demasiado engorroso:

```
tree=decision_tree() %>% set_engine("rpart") %>% set_mode("regression") %>%
  fit(Pedido~Litros_1_pedido_antes +
  Litros_2_pedidos_antes +
  Estacion,
  data = datos)
datos %>%
  ggplot(aes(x=Litros_1_pedido_antes, y=Litros_2_pedidos_antes),
  cex=1.5, col="Orange") +
  geom_jitter(aes(col=Pedido), alpha=0.8,cex=3) +
  geom_parttree(data = tree,alpha=0.2,cex=1.2,
  aes(fill=Pedido), show.legend = FALSE)
```

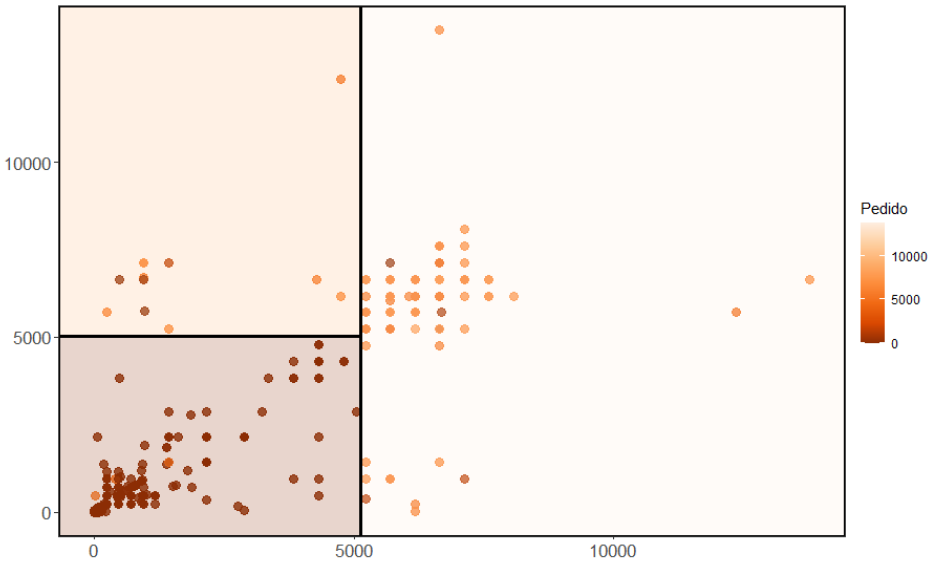


Figura 3.2: Regiones del espacio predictor asociadas al árbol de regresión. En el eje X se considera la variable **`Litros 1 pedido antes`**, mientras que en el eje Y se considera la variable **`Litros 2 pedidos antes`**.

Siguiendo con la analogía del árbol, las regiones R_1, R_2 y R_3 se conocen como **nodos terminales** u **hojas**, de esta forma vemos que los árboles de regresión como el que se muestra en la Figura 3.1 se suelen dibujar del revés, en el sentido de que las hojas están en la parte inferior del árbol. Los puntos en los que se divide el espacio predictor a lo largo del árbol se conocen como **nodos internos** y los segmentos que conectan los nodos entre sí se denominan **ramas**.

Para terminar con este ejemplo motivador vamos a interpretar la Figura 3.1. Los litros solicitados en el pedido anterior son el factor más importante pare determinar los futuros pedidos. Si el pedido anterior supera el umbral de litros fijado, en este caso 5134, la estimación para el siguiente pedido es mayor que en el caso de que no lo supere. De esta forma, parece que si la cantidad de litros del pedido anterior es muy grande, los litros solicitados hace dos pedidos no es relevante a la hora de predecir la cantidad del próximo pedido. Sin embargo, si la cantidad del pedido anterior no supera el umbral de 5134 litros, entonces la cantidad solicitada hace dos pedidos si que resulta relevante y en función de si ese pedido superó o no los 5014 litros tenemos una estimación diferente para el siguiente pedido. Es notable destacar que, en este caso particular, la variable factor **`Estación`** no influye en la estimación del siguiente pedido, es por ello, por lo que no aparece en el diagrama del árbol de regresión de la Figura 3.1.

En la Figura 3.3 se puede ver el mismo modelo aplicado sobre otro distribuidor, y en este caso, podemos apreciar que si influye la variable factor **`Estación`**.

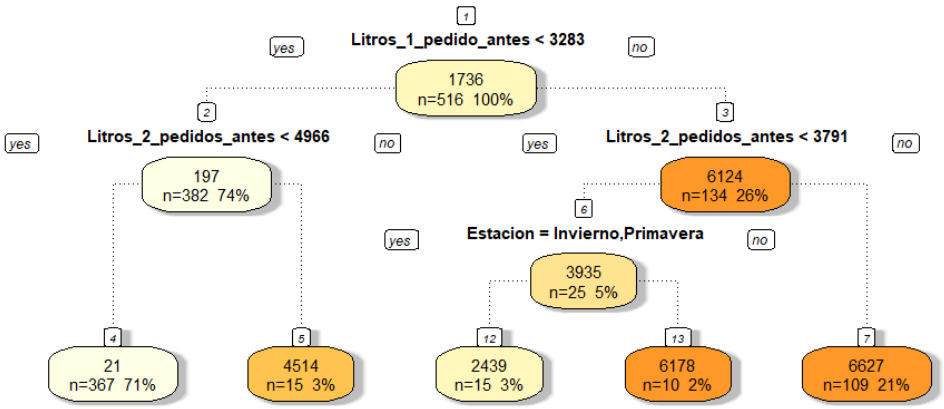


Figura 3.3: Árbol de regresión para un distribuidor de tipo B. El valor que se muestra en la parte superior de cada recuadro hace referencia a la predicción de litros del producto que estamos analizando para el próximo pedido. Los valores inferiores hacen referencia a la cantidad de datos pertenecientes a cada grupo. El valor de la izquierda viene dado en término absoluto, mientras que el de la derecha está dado en porcentaje.

Los árboles de regresión que se muestran en las Figuras 3.1 y 3.3 probablemente sean una simplificación excesiva de la verdadera relación entre la variable respuesta y las variables explicativas, sin embargo, nos permiten realizar una interpretación gráfica y sencilla de los modelos.

Después de realizar esta introducción gráfica a los métodos basados en árboles, vamos a centrarnos en el proceso de construcción del árbol de regresión. Fundamentalmente, se divide en dos pasos:

Paso 1. Dividimos el espacio predictor, es decir, el conjunto de posibles valores para $X_1, X_2, \ldots, X_p$ en J regiones disjuntas, $R_1, R_2, \ldots, R_J$, de tal forma que recubra todo el espacio de posibles valores $X_1, \ldots, X_p$.

Paso 2. Para cada observación que pertenezca a la región R_j realizamos la misma predicción, ésta se corresponde con la media de los valores de la variable respuesta para las observaciones de entrenamiento de la región R_J.

Reflejemos esto con un ejemplo sencillo. Supongamos que en el Paso 1 obtenemos 2 regiones, denominadas R_1 y R_2, y que la respuesta media de las observaciones de entrenamiento que caen en la región R_1 es 10, mientras que en la región R_2 es 20. De esta forma, dada una observación $X = x$, si $x \in R_1$ realizaremos una predicción de 10, y si $x \in R_2$ predecimos un valor de 20.

Llegados a este punto, nos preguntaremos como construimos las diferentes regiones $R_1, R_2, \ldots, R_J$. En principio, las regiones podrían adoptar cualquier forma, no estarían sujetas a ninguna forma particular, sin embargo, se opta por dividir el espacio predictor en rectángulos o cajas de alta dimensión

para simplificar el problema. Además, esto facilita la interpretación del modelo resultante. De esta forma, el objetivo es encontrar las cajas, $R_1, R_2, \ldots, R_J$, que minimicen la suma de los residuos al cuadrado (RSS), que viene dada por la siguiente expresión,

$$RSS = \sum_{j=1}^{J} \sum_{i \in R_j} (y_i - \hat{y}_{R_j})^2, \tag{3.1}$$

donde $\hat{y}_{R_j}$ es la media de la variable respuesta para las observaciones de entrenamiento que pertenecen a la j-ésima caja. Es importante destacar que, esta medida se basa en una idea similar a la utilizada en la medida MSE descrita en la ecuación (2.4).

Desgraciadamente, por problemas computacionales, es imposible considerar todas las particiones posibles del espacio predictor en J cajas. Por este motivo, se adopta un enfoque ambicioso de arriba hacia abajo que se conoce como **división binaria recursiva**.

Este enfoque es de arriba hacia abajo porque se empieza en la parte superior del árbol, notemos que en ese punto todas las observaciones pertenecen a una sola región, y luego se divide sucesivamente el espacio predictor. Cada división se va indicando a través de dos nuevas ramas. Asimismo, se selecciona la mejor división posible para cada paso en la construcción del árbol, en lugar de elegir una división que pueda producir un árbol mejor en algún paso futuro.

Para realizar la división binaria recursiva se selecciona el predictor X_j y el punto de corte s de tal forma que si dividimos el espacio predictor en las regiones $\{X|X_j < s\}$ y $\{X|X_j \geq s\}$ esto supone la mayor reducción posible en la suma de los residuos al cuadrado. Esto es, se consideran todos los predictores, $X_1, X_2, \ldots, X_p$, y todos los valores posibles del punto de corte s para cada uno de ellos y se elige el predictor y el punto s que nos proporcionen el árbol con menor RSS.

De forma rigurosa, para cualquier j y s, definimos el par de semiplanos

$$R_1(j,s) = \{X|X_j < s\} \quad \text{y} \quad R_2(j,s) = \{X|X_j \geq s\}, \tag{3.2}$$

y tenemos que encontrar los valores de j y s que minimicen la ecuación

$$\sum_{i:\, x_i \in R_1(j,s)} (y_i - \hat{y}_{R_1})^2 + \sum_{i:\, x_i \in R_2(j,s)} (y_i - \hat{y}_{R_2})^2, \tag{3.3}$$

donde $\hat{y}_{R_1}$ es la media de la variable respuesta para las observaciones de entrenamiento que pertenecen al semiplano $R_1(j,s)$, y $\hat{y}_{R_2}$ es la media de la variable respuesta para las observaciones de entrenamiento que pertenecen al semiplano $R_2(j,s)$.

Luego, tenemos que repetir el proceso de tal forma que siempre busquemos el mejor predictor y el mejor punto de corte para continuar dividiendo los datos, pero siempre teniendo en cuenta nuestro objetivo, que no es otro que el de minimizar el RSS dentro de cada una de las regiones resultantes. Sin embargo, ahora en vez de dividir todo el espacio predictor tenemos que dividir una de las dos regiones identificadas anteriormente, de tal forma que obtengamos tres regiones. Una vez tengamos estas tres regiones, buscaremos dividir una vez más una de ellas para minimizar el RSS. Este proceso continua

reiteradamente hasta que se alcance un criterio de parada prefijado anteriormente. Por ejemplo, como criterio se podría establecer que se continuará realizando la división binaria recursiva hasta que ninguna región contenga más de un número $m \in \mathbb{N}$ de observaciones. Otro posible criterio de parada podría consistir en fijar de antemano el número de regiones J.

Una vez que tengamos identificadas las regiones, $R_1, \ldots, R_J$, dada una observación de prueba podremos realizar la predicción de su variable respuesta. Para ello, solo utilizaremos la media de las observaciones de entrenamiento de la región a la cuál pertenece la observación dada.

En la Figura 3.2 se puede ver un ejemplo con 3 regiones, las cuales se corresponden con las obtenidas a partir del árbol de regresión de la Figura 3.1.

El procedimiento que acabamos de describir puede obtener buenas predicciones sobre el conjunto de entrenamiento, pero es muy probable que sobreajuste los datos, lo cual conlleva a un rendimiento deficiente sobre el conjunto de validación o test. Esto puede ser debido a que el árbol resultante es muy complejo. Si utilizásemos un árbol más pequeño, es decir, con menos divisiones, lo cuál implicaría menos regiones, $R_1, \ldots, R_J$, podríamos generar una varianza más baja y una mejor interpretación pagando el precio de obtener un pequeño sesgo.

Una posible solución al método que acabamos de describir sería construir el árbol mientras la reducción de RSS asociada a cada nueva división supere un umbral relativamente alto. Esta nueva estrategia generará árboles más pequeños, pero tiene un enfoque demasiado local, ya que una división aparentemente ineficaz al principio podría ser seguida por una división muy buena que conduzca a una gran reducción del RSS más adelante.

Por lo tanto, parece que la mejor opción es dejar crecer el árbol, T_0, y posteriormente podarlo para obtener un subárbol que produzca mejores resultados sobre el conjunto de validación. Pero ahora surge la siguiente pregunta: ¿cuál es la mejor forma de podar el árbol?.

Recordemos que nuestro objetivo es seleccionar un subárbol que produzca la menor tasa de error sobre el conjunto de test. Dado un subárbol, podemos estimar su error sobre el conjunto de test utilizando la validación cruzada o el enfoque del conjunto de validación. Sin embargo, estimar el error de validación cruzada para cada subárbol posible podría ser una tarea demasiado compleja, pues hay una cantidad extremadamente grande de posibilidades. De esta forma, se necesita crear un procedimiento para seleccionar un conjunto de subárboles limitado que nos permita considerarlos. Es aquí donde surge la *poda de complejidad de costes*, también llamada *poda del eslabón más débil*.

De esta forma, en lugar de considerar todos los posibles subárboles solo vamos a considerar una secuencia de árboles indexados por un parámetro de ajuste no negativo α. A cada valor de α le corresponde un subárbol $T \subset T_0$ tal que

$$\sum_{m=1}^{|T|} \sum_{i:\, x_i \in R_m} (y_i - \hat{y}_{R_m})^2 + \alpha\, |T| \tag{3.4}$$

sea lo más pequeño posible. En esta expresión $|T|$ denota el número de nodos terminales del árbol T, R_m es el rectángulo, es decir, el subconjunto del espacio predictor, correspondiente al nodo terminal m,

y $\hat{y}_{R_m}$ es la predicción de la variable respuesta, es decir, la media de las observaciones de entrenamiento que pertenecen al subconjunto R_m. El parámetro de ajuste α controla el equilibrio entre la complejidad del subárbol y el ajuste a los datos.

Cabe destacar, que cuando $\alpha = 0$ el subárbol T es igual a T_0, ya que en este caso la ecuación (3.4) solo mide el error para el conjunto de entrenamiento. Sin embargo, a medida que se aumenta el valor de α, tenemos que pagar un precio por tener un árbol con muchos nodos terminales, y de esta forma, el resultado de (3.4) tenderá a minimizarse para un subárbol más pequeño.

A medida que se aumenta el valor de α en la ecuación (3.4), las ramas del árbol se eliminan de forma predecible y anidada, de tal forma que es fácil obtener la secuencia completa de subárboles en función de α.

Para seleccionar un valor de α podemos utilizar un conjunto de validación o simplemente el método de validación cruzada. Posteriormente, con el conjunto de datos completo y con el valor de α seleccionado obtenemos el subárbol correspondiente. Todo este proceso se ve resumido en el Algoritmo 1.

Algoritmo 1. *Construcción de un árbol de regresión.*

1. *Utilizaremos la división binaria recursiva para crear un árbol grande sobre los datos de entrenamiento, este algoritmo solo se detendrá cuando se verifique el criterio de parada seleccionado.*
2. *Aplicaremos la poda de complejidad de costes sobre el árbol para poder obtener una secuencia de los mejores subárboles en función del valor de α.*
3. *Utilizaremos la validación cruzada para seleccionar el valor de α. Para ello tenemos que dividir las observaciones de entrenamiento en K grupos. Para cada $k = 1, \ldots, K$:*
   a) *Repetimos los pasos 1 y 2 sobre todos los grupos menos en el k-ésimo.*
   b) *Evaluamos, en función de α, el error cuadrático medio de predicción de los datos sobre el k-ésimo grupo. Promediamos los resultados obtenidos para cada valor de α y elegimos el α que minimice el error promedio.*
4. *Devolvemos el subárbol del paso 2 que se corresponde al valor elegido de α.*

Hasta ahora hemos mencionado que los métodos basados en árboles tienen ciertos inconvenientes como pueden ser el nivel de precisión en la predicción o que son muy poco robustos, pues un pequeño cambio en los datos puede causar un gran cambio en la estimación final. Sin embargo, hasta el momento no hemos propuesto ningún método que solucione estos problemas. Es en este momento, donde surgen métodos como el ***Bagging***, los ***Random Forest*** o el ***Boosting***. Estos métodos agregan múltiples árboles de regresión con el objetivo de mejorar el rendimiento predictivo. El método *Boosting* se puede consultar en la Sección 8.2.3 de James *et al.* (2014).

Antes de entrar a explicar en detalle estos métodos, necesitamos introducir el ***Bootstrap***, pues esta herramienta es necesaria para poder comprender estos métodos.

3.2. Introducción al método *Bootstrap*.

El *Bootstrap* es un método de remuestreo propuesto por Efron en el año 1979. Es una herramienta estadística muy poderosa y de amplia aplicación que se puede utilizar para cuantificar la incertidumbre asociada con un estimador dado o un método de aprendizaje estadístico.

Lo que realmente hace muy poderoso al método *Bootstrap* es que se puede aplicar fácilmente a una amplia gama de métodos de aprendizaje estadístico, incluidos algunos para los cuales es difícil obtener una medida de la variabilidad. Además, los avances computacionales han favorecido e impulsado enormemente la aplicación de este método, pues es bastante costoso en términos de computación.

A continuación, vamos a ilustrar el método *Bootstrap* con un ejemplo muy sencillo, en el cual, queremos determinar la mejor asignación de inversión bajo un modelo simple.

Supongamos que deseamos invertir una cantidad de dinero en dos activos que generan rendimientos de X e Y, respectivamente, donde X e Y son cantidades aleatorias. De esta forma, invertiremos una proporción de dinero α en el activo X y una proporción $1-\alpha$ en el activo Y.

Debido a que existe una variabilidad asociada a los rendimientos de estos activos, queremos elegir el α de manera que se minimice el riesgo total de nuestra inversión. Formalmente, lo que queremos es minimizar $\mathrm{Var}\,(\alpha X + (1-\alpha)Y)$. Si desarrollamos esta expresión de la varianza, la derivamos con respecto de α e igualamos a cero, se puede demostrar fácilmente que el valor de α que minimiza el riesgo viene dado por:

$$\alpha = \frac{\sigma_Y^2 - \sigma_{XY}}{\sigma_X^2 + \sigma_Y^2 - 2\sigma_{XY}}, \tag{3.5}$$

donde $\sigma_X^2 = \mathrm{Var}\,(\mathrm{X})$, $\sigma^2 = \mathrm{Var}\,(\mathrm{Y})$ y $\sigma_{XY} = \mathrm{Cov}\,(X, Y)$.

Cabe destacar, que las cantidades σ_X^2, σ_Y^2 y σ_{XY} en realidad son desconocidas, sin embargo, podemos calcular unas estimaciones, $\hat{\sigma}_X^2$, $\hat{\sigma}_Y^2$ y $\hat{\sigma}_{XY}$, utilizando un conjunto de datos que contenga mediciones pasadas para X e Y. De esta forma, el valor de α estimado, $\hat{\alpha}$, que minimiza la varianza de nuestra inversión viene dado por:

$$\hat{\alpha} = \frac{\hat{\sigma}_Y^2 - \hat{\sigma}_{XY}}{\hat{\sigma}_X^2 + \hat{\sigma}_Y^2 - 2\hat{\sigma}_{XY}}. \tag{3.6}$$

En nuestro ejemplo, vamos a suponer que tanto X como Y siguen una distribución normal. Además, tomaremos $\sigma_X^2 = 1$, $\sigma_Y^2 = 1.25$ y $\sigma_{XY} = 0.5$, por lo que nuestro verdadero valor de α será 0.6. Simularemos 100 pares de rendimientos para las inversiones X e Y y posteriormente utilizaremos estos rendimientos para estimar $\hat{\sigma}_X^2$, $\hat{\sigma}_Y^2$ y $\hat{\sigma}_{XY}$. Una vez obtenidas estas estimaciones, las sustituiremos en la ecuación (3.6) y obtendremos la estimación de α. En nuestro caso, hemos repetido este proceso 5 veces, obteniendo los siguientes valores de α para cada conjunto de datos: $0,596$, $0,546$, $0,628$, 0.635 y $0,646$.

Llegados a este punto, es normal querer cuantificar la precisión de nuestra estimación de α. Para ello, hemos repetido el proceso de simular 100 pares de observaciones X e Y y estimar α utilizando la ecuación (3.6) 1000 veces. Por lo tanto, hemos obtenido 1000 estimaciones para α, las cuales hemos

denotado por $\hat{\alpha}_1, \hat{\alpha}_2, \ldots, \hat{\alpha}_{1000}$. En la Figura 3.4 se muestra un histograma de las estimaciones resultantes junto con su estimación kernel. Además, representamos la media muestral de las estimaciones con una línea vertical discontinua de color negro, esta media muestral se puede calcular de la siguiente forma:

$$\bar{\alpha} = \frac{1}{1000} \sum_{i=1}^{1000} \hat{\alpha}_i = 0.5996. \tag{3.7}$$

Podemos ver que este valor está muy cerca del verdadero valor de α, que sabemos que es 0.6, y la desviación típica de esta estimación es

$$\sqrt{\frac{1}{1000-1} \sum_{i=1}^{1000} (\hat{\alpha}_i - \bar{\alpha})^2} = 0.0816. \tag{3.8}$$

Esto nos sirve para hacernos una idea de la precisión de $\hat{\alpha}$, pues en términos generales, para una muestra aleatoria de la población, esperamos que $\hat{\alpha}$ difiera, en promedio, 0.0816 del verdadero α.

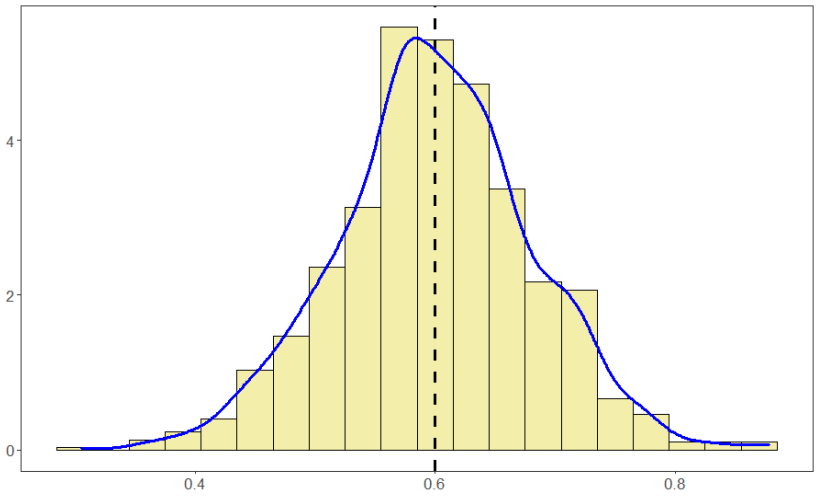


Figura 3.4: Histograma de las estimaciones de α obtenidas al simular 1000 conjuntos de datos de la población. La línea vertical discontinua de color negro representa la media muestral de dichas estimaciones. La curva de color azul representa la estimación kernel para las estimaciones $\hat{\alpha}_1, \ldots, \hat{\alpha}_{1000}$.

Sin embargo, en la práctica, el procedimiento que acabamos de describir para estimar la desviación típica de $\hat{\alpha}$ no es factible, pues para unos datos reales no podemos generar nuevas muestras a partir de la población original, pues esta es desconocida. Con todo, el enfoque *Bootstrap* nos permite utilizar un ordenador para simular el proceso de obtener nuevos conjuntos de datos y de esta forma, poder estimar la variabilidad de $\hat{\alpha}$ sin tener que generar unas muestras poblacionales adicionales. En vez de obtener conjuntos de datos independientes de la población original, si utilizamos este enfoque lo que estamos haciendo es muestrear reiteradamente el conjunto de datos obteniendo diferentes conjuntos a partir del original.

Imaginemos que tenemos un conjunto de datos Z formado por n observaciones. Seleccionemos de forma aleatoria n datos de este conjunto Z. Estas observaciones darán lugar al conjunto Z_1^*. Notemos

que el muestreo se tiene que realizar con reemplazamiento y bajo equiprobabilidad, lo cuál significa que la misma observación puede ocurrir más de una vez en el nuevo conjunto de datos. Debemos de tener en cuenta que, si una observación esta contenida en Z_1^* se incluyen sus valores X e Y. Podemos utilizar Z_1^* para obtener una nueva estimación *Bootstrap* de α, la denotaremos mediante $\hat{\alpha}_1^*$. Este procedimiento se repetirá de forma iterativa B veces, siendo B un número natural grande. El objetivo es generar B conjuntos de datos diferentes, los cuales denotaremos por $Z_1^*, Z_2^*, \ldots, Z_B^*$, y las correspondientes B estimaciones de α, denotadas por $\hat{\alpha}_1^*, \hat{\alpha}_2^*, \ldots, \hat{\alpha}_B^*$. Para calcular la desviación típica de estas B estimaciones *Bootstrap* utilizamos la siguiente fórmula:

$$\sqrt{\frac{1}{B-1}\sum_{j=1}^{B}\left(\hat{\alpha}_j^* - \frac{1}{B}\sum_{j'=1}^{B}\hat{\alpha}_{j'}^*\right)^2}. \tag{3.9}$$

El resultado de esta expresión nos sirve como una estimación del error estándar de $\hat{\alpha}$.

La estimación de $\hat{\alpha}$ utilizando el procedimiento de aproximación *Bootstrap* se puede ver en la Figura 3.5, en ella se muestra un histograma con 1000 estimaciones *Bootstrap* de α, $\hat{\alpha}_1^*, \ldots, \hat{\alpha}_{1000}^*$, junto con la estimación kernel correspondiente. Además, se representa la media muestral de las estimaciones *Bootstrap* utilizando una línea vertical discontinua de color negro. Cada una de las estimaciones ha sido calculada utilizando un conjunto de datos *Bootstrap* distinto, aunque todos han sido extraídos a partir de un único conjunto poblacional. De esta forma, el gráfico ha sido generado a partir de un único conjunto de datos, por lo que este método se podría aplicar a cualquier conjunto de datos reales.

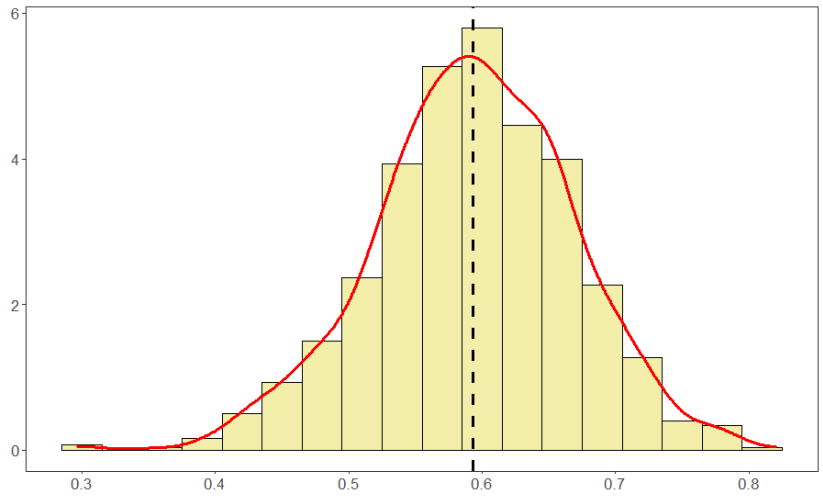


Figura 3.5: Histograma de las estimaciones de α obtenidas al simular 1000 remuestras *Bootstrap* a partir del conjunto de datos poblacional. La línea vertical discontinua de color negro representa la media muestral de dichas estimaciones. La curva de color rojo representa la estimación kernel para las estimaciones *Bootstrap* $\hat{\alpha}_1^*, \ldots, \hat{\alpha}_{1000}^*$.

Cabe destacar que el histograma de la Figura 3.5 se parece mucho al histograma basado en datos poblacionales de la Figura 3.4. Recordemos que estas estimaciones se obtuvieron después de simular

1000 conjuntos de datos de la población real. En concreto, la estimación *Bootstrap* obtenida en la ecuación (3.9) es 0.0745, muy cercana a la obtenida mediante la simulación de 1000 conjuntos de datos poblacionales, recordemos que esta estimación era de 0.0816.

Esta similitud, se puede apreciar mejor en la Figura 3.6, en la cual se representan y comparan las diferentes estimaciones kernel.

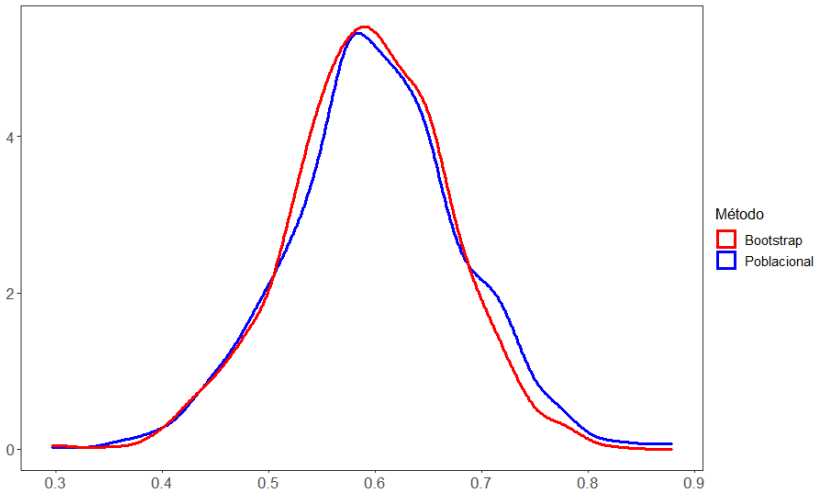


Figura 3.6: La línea azul representa la estimación kernel obtenida para $\hat{\alpha}$ después de simular 1000 muestras poblacionales, mientras que la línea roja representa la estimación kernel para $\hat{\alpha}$ utilizando 1000 remuestras *Bootstrap* a partir de la poblacional original.

El procedimiento que acabamos de exponer e ilustrar en la Figura 3.5 es conocido como el ***Bootstrap Naive***, y forma parte de una familia de métodos que siguen un razonamiento parecido al anterior. Para obtener más información acerca de los diferentes métodos *Bootstrap* existentes se puede consultar Davison y Hinkley (1997).

3.3. *Bagging.*

El procedimiento *Bootstrap* que acabamos de exponer es una herramienta muy poderosa que se suele utilizar cuando es muy difícil de calcular la desviación estándar de una cantidad que nos resulta de interés. Sin embargo, también se puede aplicar sobre otros contextos diferentes, de hecho, nosotros lo utilizaremos para mejorar los métodos de aprendizaje estadístico basados en árboles de regresión.

Los árboles de regresión expuestos anteriormente tienen alta varianza. Esto supone que si nosotros dividimos el conjunto de entrenamiento de forma aleatoria en dos subconjuntos y ajustamos un árbol de regresión a cada subconjunto, los resultados que vamos a obtener pueden ser muy diferentes. Sin embargo, si fuesen métodos con baja varianza, los resultados obtenidos en cada subconjunto serian muy similares. La agregación *Bootstrap*, o más conocida como procedimiento ***Bagging***, es una técnica cuyo objetivo principal radica en reducir la varianza de un método de aprendizaje estadístico. Esta técnica es muy utilizada en el ámbito de los árboles de regresión.

Ahora debemos de recordar que, dado un conjunto formado por n observaciones independientes, $Z_1, Z_2, \ldots, Z_n$, con varianza σ^2, la varianza de la media de las observaciones, $\text{Var}\,(\bar{Z})$, viene dada por σ^2/n. Por lo tanto, vemos que una forma natural de reducir la varianza, y en consecuencia aumentar la precisión de la predicción de un método de aprendizaje estadístico es considerar muchos conjuntos de entrenamiento de la población, construir, por separado, un modelo de predicción para cada conjunto y finalmente promediar las predicciones resultantes.

Es decir, calculamos, $\hat{f}_1(x), \hat{f}_2(x), \ldots, \hat{f}_B(x)$, utilizando B conjuntos de entrenamiento diferentes y promediamos los B valores para obtener un único modelo de aprendizaje estadístico con baja varianza, dado por

$$\hat{f}_{\text{avg}}(x) = \frac{1}{B}\sum_{b=1}^{B} \hat{f}_b(x). \tag{3.10}$$

Es importante destacar, que este procedimiento no es práctico, pues en la práctica no tenemos acceso a múltiples conjuntos de entrenamiento. Sin embargo, podemos hacer uso del *Bootstrap*, tomando muestras con reemplazamiento del único conjunto de datos del que disponemos. De esta forma, estamos generando B conjuntos de entrenamiento *Bootstrap* diferentes. Posteriormente, entrenamos nuestro modelo con el conjunto de entrenamiento *Bootstrap* b y obtenemos la estimación $\hat{f}_b^*$. Tenemos que realizar este procedimiento para cada $b = 1, \ldots, B$. Finalmente, promediamos todas las predicciones para obtener

$$\hat{f}_{\text{bag}}(x) = \frac{1}{B}\sum_{b=1}^{B} \hat{f}_b^*(x). \tag{3.11}$$

Esto es lo que se denomina ***Bagging***.

Para aplicar el *Bagging* simplemente tenemos que construir B árboles de regresión, para ello utilizaremos B conjuntos de entrenamiento *Bootstrap* diferentes y promediaremos las predicciones obtenidas. Estos árboles se dejan crecer, no se podan, por lo que, individualmente, cada árbol tiene una varianza elevada, pero en consecuencia, un sesgo bajo. Esto no supone un problema, pues ya hemos visto anteriormente que al promediar estos B árboles se reduce la varianza.

En James *et al.* (2014) se afirma que la técnica del *Bagging* mejora significativamente la precisión de la predicción, pues combina cientos, o incluso miles, de árboles en un solo procedimiento.

El número de árboles, B, no es un parámetro crítico a la hora de ajustar un método *Bagging*, pues utilizar un valor muy grande de B no nos llevará al sobreajuste. En la práctica, se suele utilizar un valor de B suficientemente grande para conseguir que se establezca el error.

Veamos ahora como estimar el error del conjunto de test cuando utilizamos un método *Bagging*. En este caso, no hace falta utilizar la validación cruzada ni el enfoque del conjunto de validación o test. Basta recordar que en el *Bagging* los árboles se ajustan repetidamente a subconjuntos de las observaciones obtenidos mediante *Bootstrap*. En la Observación 3.3.1 se demuestra que, para un n suficientemente grande, en promedio, cada árbol *Bagging* utiliza aproximadamente dos tercios de las observaciones.

Observación 3.3.1. Anteriormente, hemos visto que en el *Bootstrap Naive* el muestreo se realiza bajo las condiciones de reemplazamiento y equiprobabilidad. Además, en nuestro caso, hemos supuesto que

cada muestra contiene n observaciones.

De esta forma, bajo las condiciones que acabamos de describir, la probabilidad de que la observación i-ésima ocupe la primera posición de la muestra *Bootstrap* es $\frac{1}{n}$. Como estamos actuando bajo reemplazamiento, ocurre lo mismo para todas las posiciones de la muestra.

Además, sabemos que la probabilidad de que la observación i-ésima no ocupe la primera posición de la muestra *Bootstrap* es $1 - \frac{1}{n}$. Análogamente para las demás posiciones, pues estamos bajo reemplazamiento.

Por lo tanto, la probabilidad de que la observación i-ésima no esté en la muestra *Bootstrap* es $(1 - \frac{1}{n})^n$ y es de sobra conocido que esta expresión converge a $\frac{1}{e}$ cuando n tiende a infinito.

De esta forma, para un tamaño de muestra n, grande, la probabilidad de que la i-ésima observación esté en la muestra *Bootstrap* es $1 - \frac{1}{e}$.

Si ahora expresamos el número de observaciones de la muestra *Bootstrap* como una suma de n variables Bernoulli con probabilidad de éxito $1 - \frac{1}{e}$, el número de observaciones esperadas en la muestra es $(1 - \frac{1}{e})\, n \sim \frac{2}{3}\, n$.

El tercio de las observaciones restantes que no se utiliza para ajustar un árbol se denominan observaciones fuera de la bolsa y se denotan por **OOB**. De esta forma, podemos predecir la variable respuesta para la observación i-ésima utilizando los árboles para los cuales esa observación fue OOB, lo cuál, producirá cerca de $B/3$ predicciones para la observación i-ésima. Como es natural, para obtener una sola predicción, basta con promediar las $B/3$ predicciones, esto conduce a una única predicción OOB para la observación i-ésima. Aplicando este procedimiento se puede obtener una predicción OOB para cada una de las n observaciones, a partir de las cuales se puede calcular el MSE OOB. El error OOB obtenido es una estimación válida del error del método *Bagging*, pues para cada observación, la variable respuesta ha sido predicha utilizando solo los árboles que no se ajustaron con esa observación.

El enfoque OOB para estimar el error es particularmente ventajoso cuando se aplica el método *Bagging* sobre conjuntos de datos muy grandes, para los cuales, la validación cruzada sería muy pesada computacionalmente.

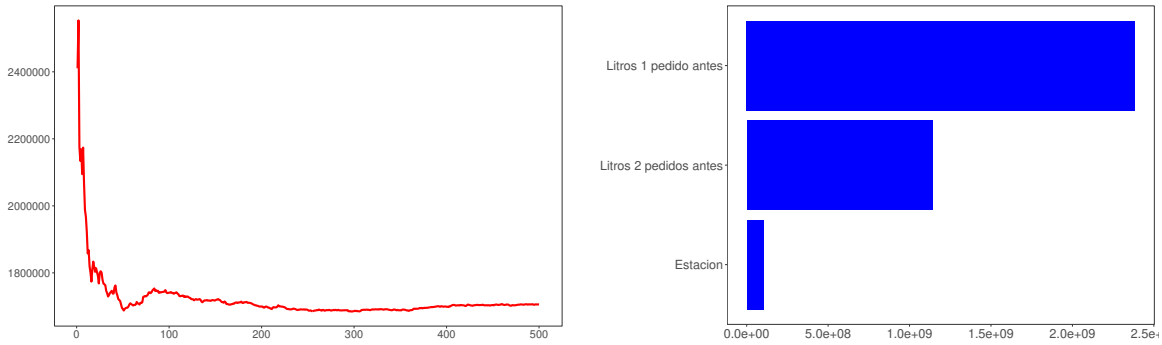
De esta forma, y en consonancia con lo que venimos exponiendo, la aplicación del método *Bagging*, generalmente, proporciona mejores resultados de precisión si lo comparamos con el uso de un único árbol de regresión. Sin embargo, no todo podían ser buenas noticias, pues esta mejora en la precisión de la predicción conlleva consigo una pérdida de la interpretabilidad del modelo resultante, pues al aplicar el método *Bagging*, ya no es posible representar el procedimiento en un único árbol, y en consecuencia, ya no tenemos información interpretable sobre las variables más importantes para nuestro modelo. Por lo tanto, ahora es natural que aparezca la siguiente cuestión, ¿cómo sabemos cuales son las variables más importantes si aplicamos el procedimiento *Bagging*?.

Aunque el procedimiento *Bagging* es mucho más complicado de interpretar que la aplicación de un único árbol de regresión, se puede obtener un resumen general de la importancia de cada variable explicativa utilizando el RSS definido en (3.1). Dado un predictor, podemos cuantificar cuanto se reduce

el RSS total debido a las divisiones, posteriormente, promediamos este valor sobre los B árboles. Si como resultado obtenemos un valor grande, esto nos indicará que se trata de un predictor importante. Si por el contrario, el valor que obtenemos es pequeño, indicará que el predictor que estamos analizando no es importante para nuestro modelo.

A continuación, vamos a ilustrar lo que venimos comentando del procedimiento *Bagging*. En la Figura 3.7(a) se pueden ver los resultados del método *Bagging* sobre los datos que han originado el árbol de regresión de la Figura 3.1. La tasa de error se muestra en función de B, que recordemos que es el número de árboles construidos utilizando conjuntos de entrenamiento *Bootstrap*. De esta forma, podemos ver que la tasa de error del procedimiento *Bagging* es bastante menor que la tasa de error obtenida con un único árbol. Notemos que esta se corresponde con el error correspondiente para $B = 1$. Además, se puede ver que a partir de $B = 100$ la tasa del error se estabiliza.

Para los mismos datos, se puede ver una representación gráfica de las variables más importantes en la Figura 3.7(b). De este modo, podemos ver que la variable **`Litros 1 pedido antes`** es la que provoca una mayor disminución media en el RSS y, en consecuencia, es la más importante para nuestro modelo. Además, cabe destacar, que en este caso la variable **`Estacion`** si que adquiere importancia, pese a ser la variable menos importante.



(a) Tasa de error del método *Bagging*.

(b) Variables más importantes del método *Bagging*.

Figura 3.7: Aplicación del método *Bagging* sobre los datos de la Figura 3.1

Para ajustar el modelo *Bagging* y crear estos gráficos hemos utilizado la librería `randomForest` de R, ver Liaw y Wiener (2002). Cabe destacar, que para ajustar el modelo *Bagging*, el parámetro `mtry` tiene que ser igual al número de variables predictoras utilizadas en el modelo. A continuación, mostramos el código de R utilizado. Es importante destacar que, se han decidido suprimir algunos parámetros relativos al estilo de los gráficos para evitar que el este resultase demasiado engorroso.

```
# Modelo Bagging
modelo_bagging=randomForest(Pedido~Litros_1_pedido_antes+
  Litros_2_pedidos_antes+
  Estacion,
  data = datos,
  mtry=3,
  importante=TRUE,
  ntree = 500)
# Gráfica del error
datos_error_1=data.frame(Árboles=1:500,Error=modelo_bagging$mse,Método="Bagging")
grafica_bagging=ggplot(datos_error_1,aes(x=Árboles,y=Error))+
  geom_line(col="red", lwd=1.2)
# Gráfico predictores
lista_predictores=data.frame("Predictor"= c("Litros 1 pedido antes",
  "Litros 2 pedidos antes", "Estacion"),
  "Valor"=unname(modelo_bagging$importance))
  importancia_bagging=lista_predictores %>% dplyr::arrange(desc(Valor)) %>%
  ggplot(aes(reorder(Predictor, Valor), Valor)) +
  geom_col(fill="blue") + coord_flip()
```

3.4. *Random Forest.*

Los métodos *Random Forest* constituyen una variante mejorada de los métodos *Bagging* mediante un pequeño ajuste que evita la correlación entre los diferentes árboles de regresión. En este caso, al igual que hacíamos en el método *Bagging*, construimos B árboles de regresión a partir de B muestras de entrenamiento *Bootstrap*, pero la diferencia radica en que al construir estos árboles, cada vez que consideramos una división en un árbol, elegimos como candidatos a la división una muestra aleatoria de m predictores sobre los p posibles, siendo $m < p$. Cabe destacar, que la división solo puede utilizar uno de esos m predictores.

Como norma general, según James *et al.* (2014), en cada división se suele tomar de forma aleatoria una muestra de $m \approx \sqrt{p}$, es decir, el número de predictores que se considera en cada división es aproximadamente la raíz cuadrada del número total de predictores.

De esta forma, cuando construimos un *Random Forest*, en cada división del árbol, el algoritmo no considera ni la mitad de los predictores posibles. Así de primeras esto puede parecer raro, pero tiene su explicación.

Supongamos, por ejemplo, que en nuestro conjunto de datos tenemos un predictor muy influyente junto con otros predictores bastante importantes. Bajo estas hipótesis, si aplicamos un método *Bagging* la mayoría de los árboles utilizarán el predictor más influyente para la primera división, y de este modo

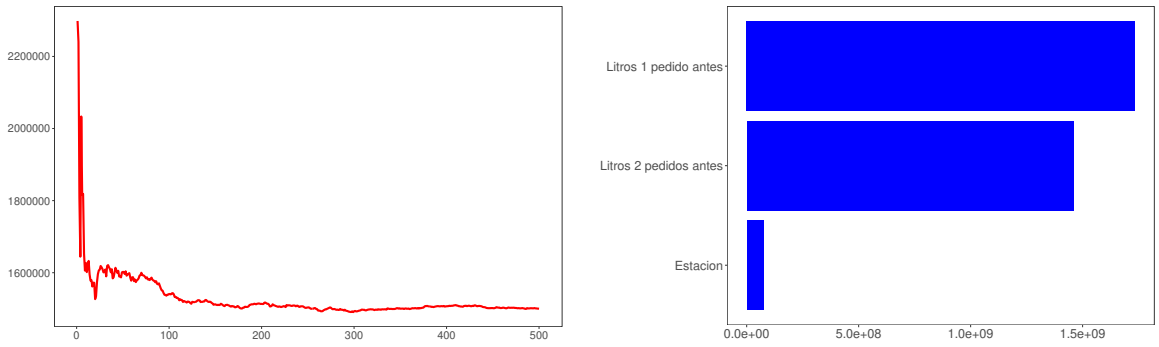
todos serán bastante parecidos entre sí. En consecuencia, las predicciones de los árboles *Bagging* estarán bastante correlacionadas, y desafortunadamente promediar variables muy correlacionadas no conlleva una reducción tan grande de la varianza como promediar variables incorrelacionadas.

Sin embargo, los *Random Forest* evitan este problema al imponer la condición de que en cada división se debe considerar solo un subconjunto de los p predictores. De esta forma, tenemos que, en promedio $\frac{p-m}{p}$ divisiones no consideran el predictor más influyente, lo cuál provoca que los demás predictores tengan más oportunidades. Este proceso evita una gran correlación entre los árboles, lo que provoca que el promedio sea menos variable y en consecuencia produzca estimaciones más estables.

De esta forma, podemos ver que la diferencia principal entre el procedimiento *Bagging* y el *Random Forest* es la elección del número de predictores que se pueden considerar en cada una de las divisiones del árbol de regresión. Es importante destacar, que si consideramos un *Random Forest* con $m = p$ esto es equivalente a aplicar un método *Bagging*. Además, cuando tengamos una gran cantidad de predictores correlacionados será útil utilizar un valor pequeño de m para construir el método *Random Forest*.

Al igual que sucedía con el método *Bagging*, en los *Random Forest* no se cometerá un sobreajuste por aumentar el tamaño de las muestras *Bootstrap* B, así que en la práctica utilizaremos un valor de B suficientemente grande, para que la tasa de error permanezca constante. Además, cabe destacar que en general, para el método *Random Forest* vamos a necesitar más árboles que para el método *Bagging*, aunque esto no va afectar a la eficiencia computacional, pues al considerar solo unos pocos predictores en cada división, la construcción de cada árbol es bastante más rápida.

A continuación, aplicaremos el método *Random Forest* sobre el mismo conjunto de datos que hemos utilizado para el método *Bagging* y veremos las diferencias que venimos comentando.



(a) Tasa de error del método *Random Forest.* (b) Variables más importantes del método *Random Forest.*

Figura 3.8: Aplicación del método *Random Forest* sobre los datos de la Figura 3.1

En la Figura 3.8(a) se pueden ver los resultados del método *Random Forest* sobre los datos que han originado el árbol de regresión de la Figura 3.1. La tasa de error se muestra en función del número de árboles construidos utilizando diferentes conjuntos de entrenamiento *Bootstrap*. Podemos ver que

en este caso la tasa de error es menor que en el caso del método *Bagging*, aunque esta comparación se verá mejor en la Figura 3.9.

En la Figura 3.8(b) se puede ver una represetación gráfica de las variables más importantes para el método *Random Forest*. En este caso, podemos ver que existe una menor diferencia entre la variable más influyente, **`Litros 1 pedido antes`**, y la segunda variable más importante, **`Litros 2 pedidos antes`**. Además, parece que se mantiene la importancia de la variable **`Estacion`**.

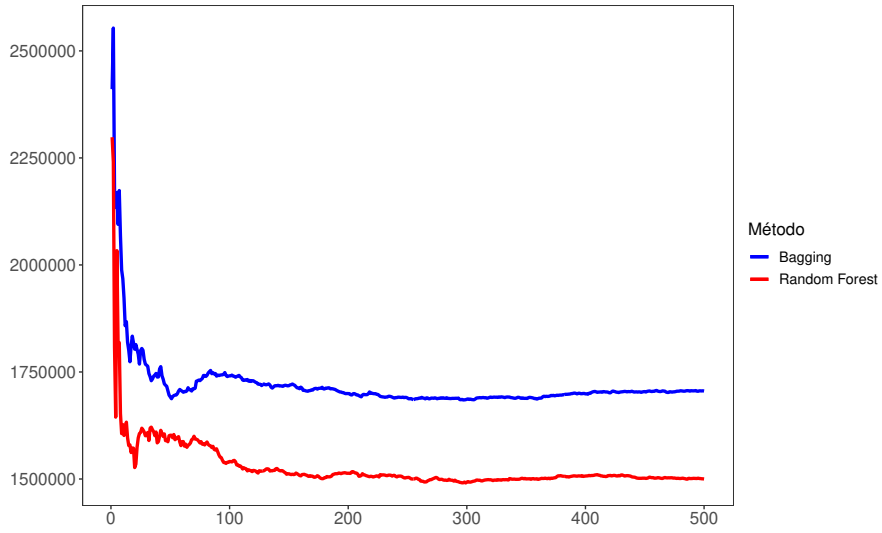


Figura 3.9: Comparación de la función de la tasa de error entre el método *Bagging* y el *Random Forest*.

Para ajustar el modelo *Random Forest* y crear los gráficos hemos utilizado la librería **`randomForest`** de R, ver Liaw y Wiener (2002). A continuación, mostramos el código de R utilizado. Se han omitido algunos parámetros de estilo para evitar que el código resulte demasiado engorroso.

```
# Modelo Random Forest
modelo_rf=randomForest(Pedido~Litros_1_pedido_antes+
  Litros_2_pedidos_antes+
  Estacion,
  data = datos,
  importante=TRUE,
  ntree = 500)

# Gráfica del error
datos_error_2=data.frame(Árboles=1:500,Error=modelo_rf$mse,
Método="Random Forest")
grafica_rf=ggplot(datos_error_2,aes(x=Árboles,y=Error))+
  geom_line(col="red", lwd=1.2)
```

```
# Gráfico predictores
lista_predictores=data.frame("Predictor"= c("Litros 1 pedido antes",
  "Litros 2 pedidos antes",
  "Estacion"),
  "Valor"=unname(modelo_rf$importance))
importancia_rf=lista_predictores %>%
  dplyr::arrange(desc(Valor)) %>%
  ggplot(aes(reorder(Predictor, Valor), Valor)) +
  geom_col(fill="blue") +
  coord_flip()
```

Capítulo 4

Redes Neuronales.

En este capítulo vamos a realizar una introducción teórica al funcionamiento de las *Neural Networks*, habitualmente conocidas como **Redes Neuronales**. Para ello, principalmente, se hace uso de James *et al.* (2014) y de Hastie *et al.* (2001). Este método será utilizado en el Capítulo 5 durante el desarrollo práctico del trabajo.

Las **Redes Neuronales** son una serie de algoritmos que buscan relaciones en un conjunto de datos. Constan de nodos interconectados que que le dan la apariencia de una red neuronal biológica, de donde proviene su nombre. Este método es más complejo que los árboles aleatorios, ya que necesita ajustar diferentes hiperparámetros, pero destaca por su utilidad en diferentes aplicaciones como pronóstico de ventas, gestión de riesgos, reconocimiento de voz, reconocimiento de imágenes...

En la Sección 4.1 realizamos la **introducción teórica a las Redes Neuronales**. En la Sección 4.2 se muestran **diferentes funciones de activación** utilizadas en las Redes Neuronales. En la Sección 4.3 se trata el **ajuste** de las **Redes Neuronales**. Finalmente, en la Sección 4.4 se abordan los diferentes **problemas** que surgen a la hora de **ajustar las Redes Neuronales**.

4.1. Introducción a las Redes Neuronales.

En esta sección, vamos a describir una clase de métodos de aprendizaje estadístico que se desarrollaron por separado en diferentes campos, como pueden ser la estadística o la inteligencia artificial, pero que esencialmente estaban basados en modelos idénticos.

Las Redes Neuronales aparecieron con mucha fuerza a finales de la década de 1980. Sin embargo, llegaron métodos como los *SVM* (*Support Vector Machine*), el *Boosting* y los *Random Forest* que provocaron que estas cayeran un poco en desuso. En gran parte, este desuso fue causado porque las Redes Neuronales requerían muchos retoques, mientras que los nuevos métodos eran más automáticos y, en muchos problemas, superaban el rendimiento de las Redes Neuronales mal entrenadas.

Posteriormente, alrededor del año 2010, resurgieron las Redes Neuronales, este hecho fue posible gracias a la disponibilidad de conjuntos de datos de entrenamiento cada vez más grandes y a la gran

digitalización de la ciencia y la industria. Las Redes Neuronales tienen un gran éxito en problemas de clasificación de imágenes o modelado de voz y texto.

La idea central de este método es extraer combinaciones de las variables explicativas como características derivadas, y posteriormente, modelar la variable respuesta como una función no lineal de estas características. El resultado es un poderoso método de aprendizaje estadístico, con amplias aplicaciones en diferentes campos.

Cabe destacar que las Redes Neuronales están basadas en el modelo de regresión de la búsqueda de proyección. Este modelo evolucionó en el ámbito de la estadística semiparamétrica. En nuestro caso, no vamos a explicar el modelo de la búsqueda de proyección, pero este se puede ver explicado en la Sección 11.2 de Hastie *et al.* (2001).

En este trabajo nos centraremos principalmente en las Redes Neuronales de una sola capa, pues consideramos que son suficientes para abordar la complejidad de nuestro problema.

Redes Neuronales de una sola capa.

Las Redes Neuronales abarcan una gran clase de modelos y métodos de aprendizaje estadístico. Aquí empezaremos describiendo la más utilizada, habitualmente denominada, red de propagación hacia atrás de una sola capa oculta. Es importante destacar, que a pesar de todo el misterio que envuelve a las Redes Neuronales, estas solo son modelos estadísticos no lineales muy parecidos al modelo de regresión de búsqueda de proyección, mencionado anteriormente.

Las Redes Neuronales son un modelo de clasificación o regresión en dos etapas y se representan por un diagrama de red como el que se puede ver en la Figura 4.1. La capa del lado izquierdo se denomina capa de entrada (*input layer*) y representa las variables predictoras para nuestro modelo, la capa del medio es la que se denomina capa oculta (*hidden layer*) y la capa que se encuentra en el lado derecho es la capa de salida (*output layer*) que representa nuestra variable respuesta. Es importante destacar, que toda red neuronal tiene una capa input y una output, pero puede tener más de una capa intermedia. La red de la Figura 4.1 es la que se obtiene si aplicamos una red neuronal con una única capa intermedia a los datos que hemos estado utilizando para los ejemplos de los *Random Forest*.

En este caso, para ajustar el modelo de Redes Neuronales hemos utilizado la función `nnet` de la librería `nnet` de R, ver Venables y Ripley (2002b). Sin embargo, durante el desarrollo práctico del trabajo hemos decidido utilizar la librería `neuralnet`, ver Fritsch *et al.* (2019), pues esta última permite realizar una mayor personalización de los parámetros. A continuación, mostramos el código utilizado.

```
# Ajuste Red Neuronal
neural=nnet(Pedido~Litros_1_pedido_antes+Litros_2_pedidos_antes+Estacion,
  data=datos,
  size=2,
  linout=FALSE, skip=TRUE,
  algorithm="rprop+",
  decay=1e-5, err.fct = "sse")
```

```
# Gráfico Red Neuronal
plotnet(neural,
  x_names = c("Litros 1 pedido antes","Litros 2 pedidos antes","Otoño",
  "Primavera", "Verano"),
  y_names="Pedido")
```

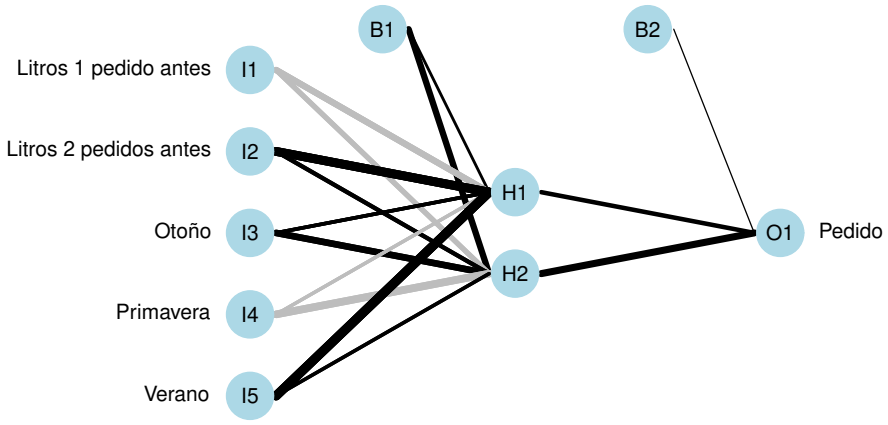


Figura 4.1: Diagrama de una red neuronal con una única capa oculta. Las conexiones con pesos asociados positivos se representan de color negro, además cuanto más positivo es el peso asociado, más gruesa es la línea de la conexión. Análogamente, las conexiones con pesos asociados negativos se representan de color gris.

Las capas de las Redes Neuronales están formadas por elementos más básicos, denominados **neuronas**. La función de estos elementos es aplicar una transformación a los datos recibidos en la capa input para predecir un valor continuo o categórico en la capa output. De esta forma, las capas pueden ser clasificadas según la interconexión entre las neuronas. Por ejemplo, cuando cada neurona de la capa intermedia se conecta a todas las neuronas de las capas anterior y posterior, decimos que se trata de una capa intermedia de tipo denso, habitualmente denominada *fully connected layer*. En la Sección 10.3 de James *et al.* (2014) se pueden encontrar diferentes tipos de capas intermedias. En la Sección 10.3.1 tenemos las capas de convolución y en la 10.3.2 se encuentran las capas de agrupación, este tipo de capas intermedias son utilizadas en las **Redes Neuronales convolucionales**. Además, en la Sección 10.5, se encuentran las **Redes Neuronales recurrentes**, cuya capa intermedia es una secuencia.

Estos modelos se aplican tanto a la regresión como a la clasificación. En el caso de la regresión, normalmente, se utiliza $K = 1$ y solo tendremos una unidad de salida, Y_1, como se puede ver en el lado derecho de la Figura 4.1. Sin embargo, las Redes Neuronales pueden manipular múltiples variables respuesta cuantitativas sin ningún problema. Es por esto, por lo que expondremos el caso general.

Para la clasificación de K-clases, hay k unidades en la parte derecha del diagrama, y la unidad k-ésima modela la probabilidad de la clase k. Tenemos K valores respuesta Y_k, $k = 1 \ldots, K$, siendo cada uno de los valores codificado como una variable binaria 0 o 1 para la k-ésima clase.

Las características derivadas, Z_m, se crean a partir de combinaciones lineales de las variables de entrada, y la respuesta Y_k, cuya salida asociada se denota por T_k, se modela como una función de combinaciones lineales de Z_m.

Veamos ahora la formulación matemática de las Redes Neuronales.

$$\begin{aligned} Z_m &= \sigma(\alpha_{0m} + \alpha_m^T X),\ m = 1 \ldots, M, \\ T_k &= \beta_{0k} + \beta_k^T Z,\ k = 1 \ldots, K, \\ f_k(X) &= g_k(T),\ k = 1, \ldots, K, \end{aligned} \tag{4.1}$$

donde $Z = (Z_1, Z_2, \ldots, Z_M)$, y $T = (T_1, T_2, \ldots, T_K)$. Cabe destacar que X, α_m y β_k son vectores columna.

Generalmente, se utiliza la función de activación sigmoide, $\sigma(v) = 1/(1 + e^{-v})$. Esta función se puede ver representada en la Figura 4.2. En la siguiente sección se pueden ver diferentes funciones de activación muy utilizadas en la práctica. La función de salida $g_k(T)$ nos permite realizar una transformación final del vector respuesta T.

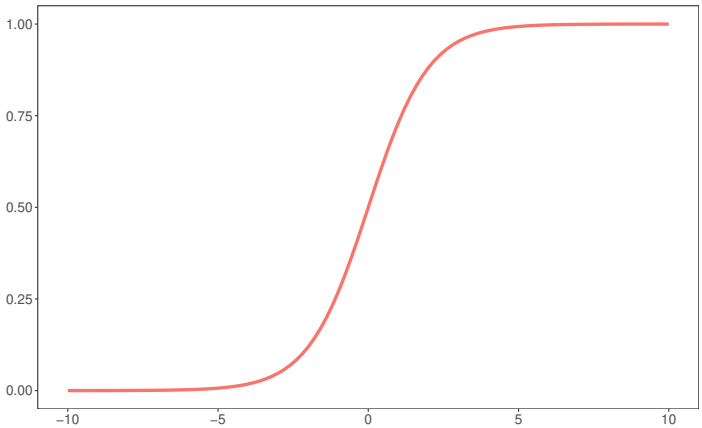


Figura 4.2: Gráfica de la función sigmoide.

Generalmente, para la regresión, escenario en el que estamos interesados, se suele utilizar la función identidad, es decir, $g_k(T) = T_k$. Incluso, para los primeros trabajos de clasificación en K clases también se utilizó la función identidad, aunque luego se abandonó en favor de la función *softmax*, explicada brevemente en la Sección 4.2.

Las unidades en el medio de la red se denominan unidades ocultas, pues los valores Z_m no se observan directamente. En la red de la Figura 4.1 vemos que hay dos unidades ocultas. En general, puede haber más de una capa oculta, pero nosotros solo consideraremos ese caso para realizar un breve comentario. Entonces, se puede pensar en las Z_m como una base de expansión de las variables de entrada originales X; de esta forma, la red neuronal es un modelo lineal estándar, o un modelo multilogit lineal, que utiliza estas transformaciones como entradas.

Cabe destacar, que si σ es la función identidad, el modelo colapsa a un modelo lineal en las entradas. De esta forma, una red neuronal se puede pensar como una generalización no lineal del modelo lineal, tanto para el caso de la regresión, como para la clasificación.

En la Figura 4.3 se puede ver que la tasa de activación de la función sigmoide depende de la norma euclídea de α_m, $\|\alpha_m\|$. De esta forma, si $\|\alpha_m\|$ es muy pequeña, la unidad o neurona estará operando en la "parte lineal" de su función de activación.

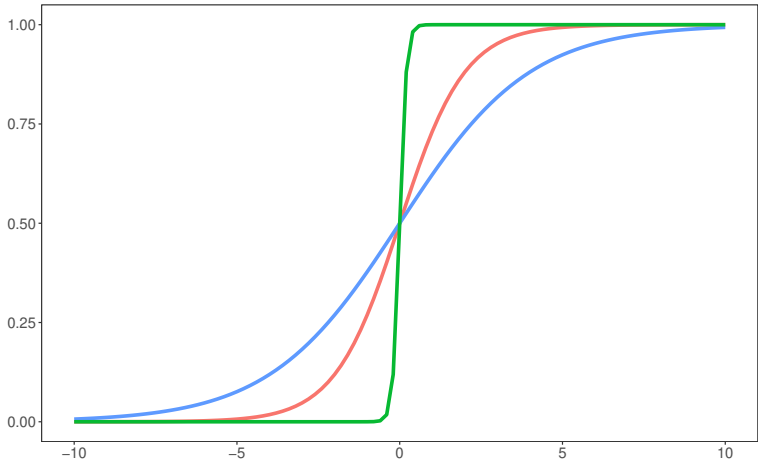


Figura 4.3: Gráfica de la función sigmoide, $\sigma(v) = \frac{1}{(1+e^{-v})}$, (curva roja). Se incluye la curva $\sigma(s\,v)$ para $s = \frac{1}{2}$, (curva azul) y para $s = 10$, (curva verde).

El parámetro de escala, s, utilizado en la Figura 4.3 controla la velocidad de activación. Además, se puede ver que para valores grandes es equivalente a una activación dura en $v = 0$. También cabe destacar, que $\sigma(s(v - v_0))$ cambia el umbral de activación de 0 a v_0.

4.2. Funciones de activación.

A continuación, vamos a detallar las diferentes funciones de activación, $\sigma(v)$, que se suelen utilizar en la práctica.

La función identidad.

La función identidad se define como:

$$\sigma(v) = v, \tag{4.2}$$

donde v es la combinación lineal de las variables de entrada. Esta función se suele utilizar cuando nos interesa que nuestro modelo sea capaz de percibir patrones lineales en las capas ocultas, y cuando las predicciones de la capa final sean de carácter numérico.

Función sigmoide.

La función sigmoide se define como:

$$\sigma(v) = \frac{1}{1 + e^{-v}}. \tag{4.3}$$

Esta es la función de activación más utilizada en la capa *output* dentro de los modelos de clasificación binaria. La idea subyacente de esta función radica en transformar un *output* numérico en un valor comprendido entre 0 y 1 que siga una distribución Bernouilli, pues estamos buscando predecir la probabilidad de una variable de tipo categórica para dos categorías distintas.

Rectified Linear Unit (ReLU).

La función ReLU, presentada por Hahnloser *et al.* (2000) para explicar el funcionamiento de circuitos de silicona inspirados en la interacción entre neuronas dentro de una corteza cerebral, es la función de activación de capas ocultas más conocida en el ámbito de las Redes Neuronales. La función ReLU se define como:

$$\sigma(v) = \text{máx}\{0, v\}. \tag{4.4}$$

Esta función iguala a cero todo *input* negativo. A diferencia de otras funciones de activación, es una función sin límites. Se puede ver representada en la Figura 4.4.

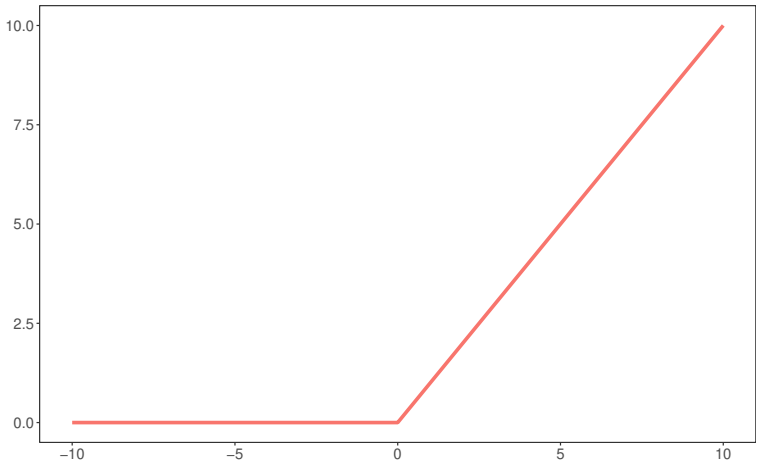


Figura 4.4: Gráfica de la función ReLU.

En Zeiler *et al.* (2013) se pueden ver varios motivos por los cuales la función ReLU es una de las funciones de activación más popular. Entre ellos destaca que: son fáciles de optimizar, son rápidas de procesar en el ordenador y mitigan el problema del sobreajuste. Es importante destacar, que su uso se relaciona con la implementación en las capas ocultas de los modelos Deep Learning. En Agarap (2018) se propone la utilización de esta función en la capa final, dando como resultado un método de clasificación multinomial y ofreciendo resultados similares a los obtenidos con la función *softmax*, que es la función más utilizada para ese tipo de problemas.

Por ahora solo hemos comentado las ventajas de las funciones ReLU, sin embargo, este tipo de funciones también tienen sus problemas. Una de las desventajas más importantes es el problema del ReLU moribundo, conocido como *dying ReLU problem*. Este problema consiste en que mientras se entrena una red neuronal, puede pasar que las neuronas que se están utilizando se desactiven, es decir, solo nos devuelvan un valor igual a cero durante el resto del proceso de estimación, y por lo tanto, el modelo no sería capaz de captar patrones subyacentes en los datos y en consecuencia, se vería afectada la estimación del modelo.

Otro problema bastante común con estas funciones se expone en Clevert *et al.* (2015), y consiste en que las funciones ReLU pueden devolver, en término medio, un valor mayor que cero a lo largo de la red neuronal, lo cuál introduciría un sesgo positivo en las siguientes capas. Este podría acumularse a medida que la información se transforma a través de las diferentes capas y afectaría a la estimación del modelo.

Para evitar estos problemas se han propuesto algunas variantes de la función ReLU, entre ellas destacan las funciones *Leaky ReLU*, *PReLU* y *ELU*.

Leaky ReLU y PReLU.

La función Leaky ReLU se define como:

$$\sigma(v) = \begin{cases} \alpha\, v & \text{si } z \leq 0 \\ \\ v & \text{en otro caso} \end{cases}, \tag{4.5}$$

donde α es un hiperparámetro que toma valores entre 0 y 1. Esta función permite que las neuronas devuelvan valores negativos, aunque sean atenudados por α para conservar las propiedades de la función ReLU original. Este hecho ayuda a reducir el sesgo positivo. Se puede ver representada en la Figura 4.5.

Si el hiperparámetro α es escogido a mano, la función de activación (4.5) se denomina Leaky ReLU, mientras que si el hiperparámetro α es optimizado, se denomina PReLU o Parametric ReLU. Es importante destacar, que para el caso de la función PReLU el valor de α pertenece al intervalo $[-\infty, +\infty]$.

En términos generales, la función PReLU proporciona mejores resultados que la función Leaky ReLU y genera menos sobreajuste.

Exponential Linear Unit o ELU.

La función de activación se define como:

$$\sigma(v) = \begin{cases} v & \text{si } z > 0 \\ \alpha\left(e^z - 1\right) & \text{si } z \leq 0 \end{cases}, \tag{4.6}$$

donde α es un hiperparámetro. De nuevo, vemos que esta función también introduce valores negativos en su definición. Se puede ver representada en la Figura 4.5.

La función ELU fue propuesta por Clevert *et al.* (2015) y presenta mejores propiedades que las variantes comentadas anteriormente, especialmente si estamos trabajando con Redes Neuronales con cinco capas o más.

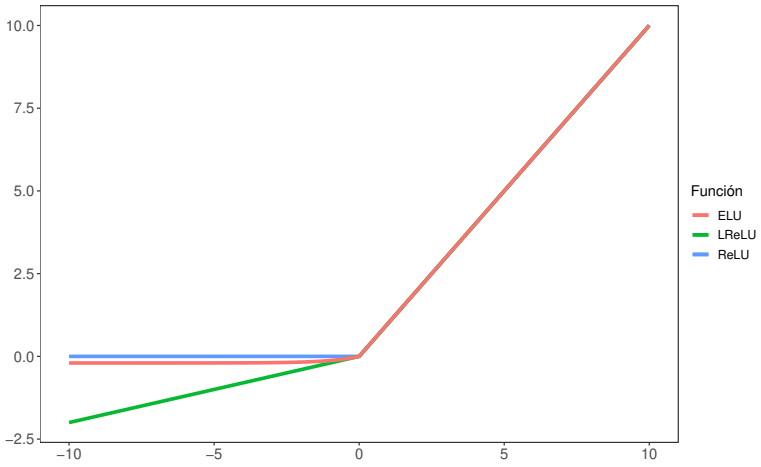


Figura 4.5: Gráfica de la función ReLU, LReLU y ELU. Para las funciones LReLU y ELU se ha considerado el hiperparámetro $\alpha = 0.2$

Función softmax.

La función softmax se define como:

$$\sigma(v_i) = \frac{\exp(v_i)}{\sum_{j=1}^{C} \exp(v_j)}, \tag{4.7}$$

donde $v \in \mathbb{R}^C$, C es el número de categorías posibles a predecir, v_i es el valor inferido por la red para el elemento i-ésimo del vector v correspondiente a la categoría i y v_j son los valores restantes del vector v.

Esta función es la generalización de la función sigmoide para C categorías, devolviendo un vector con las probabilidades asignadas a cada clasificación y cuya suma es uno.

Normalmente, esta función se utiliza para la clasificación de imágenes en la capa final de la red neuronal. Su mayor ventaja es la interpretación de la predicción numérica en términos de probabilidad, se puede ver detalladamente en Shen y Liu (2018).

4.3. Ajuste de Redes Neuronales.

Las Redes Neuronales tienen parámetros desconocidos, denominados pesos, y para los cuales se buscan valores que consigan que el modelo se ajuste bien a los datos de entrenamiento. El conjunto completo de pesos se denota por θ, y consta de

$$\begin{aligned} \{\alpha_{0m}, \alpha_m;\ m = 1, 2, \ldots, M\}\ M(p+1) \text{ pesos}, \\ \{\beta_{0k}, \beta_k;\ k = 1, 2, \ldots, K\}\ K(M+1) \text{ pesos}. \end{aligned} \tag{4.8}$$

Para el caso de la regresión se utiliza como medida de ajuste la suma de los errores al cuadrado. Esta medida viene dada por:

$$R(\theta) = \sum_{k=1}^{K} \sum_{i=1}^{N} (y_{ik} - f_k(x_i))^2. \tag{4.9}$$

En general, no se quiere minimizar el $R(\theta)$ global, pues es probable que esta sea una solución sobre-ajustada. Lo que se pretende conseguir es una regularización a través de un término de penalización o por un criterio de parada anticipado.

Generalmente, se utiliza el descenso por gradiente para minimizar $R(\theta)$, en este contexto el método se suele denominar retropropagación (*back-propagation*). En virtud de la composición del modelo, el gradiente se puede derivar fácilmente utilizando la regla de la cadena para la diferenciación. Esto se puede calcular realizando un barrido hacia delante y hacia atrás en la red neuronal, teniendo en cuenta solo las cantidades locales de cada unidad o neurona.

Ahora, vamos a ver con detalle el método de retropropagación. Consideremos $z_{mi} = \sigma(\alpha_{0m} + \alpha_m^T x_i)$ y $z_i = (z_{1i}, z_{2i}, \ldots, z_{Mi})$. De esta forma, tenemos que:

$$R(\theta) \equiv \sum_{i=1}^{N} R_i = \sum_{i=1}^{N} \sum_{k=1}^{K} (y_{ik} - f_k(x_i))^2, \tag{4.10}$$

cuyas derivadas parciales son:

$$\begin{aligned} \frac{\partial R_i}{\partial \beta_{km}} &= -2(y_{ik} - f_k(x_i))\, g_k'\, (\beta_k^T z_i)\, z_{mi}, \\ \frac{\partial R_i}{\partial \alpha_{ml}} &= -\sum_{k=1}^{K} 2(y_{ik} - f_k(x_i))\, g_k'\, (\beta_k^T z_i)\, \beta_{km}\, \sigma'(\alpha_m^T x_i)\, x_{il}. \end{aligned} \tag{4.11}$$

Con estas derivadas, la actualización en la iteración $(r+1)$ del método de descenso por gradiente

tiene la siguiente forma:

$$\begin{aligned}\beta_{km}^{(r+1)} &= \beta_{km}^{(r)} - \gamma_r \sum_{i=1}^{N} \frac{\partial R_i}{\partial \beta_{km}^{(r)}},\\ \alpha_{ml}^{(r+1)} &= \alpha_{ml}^{(r)} - \gamma_r \sum_{i=1}^{N} \frac{\partial R_i}{\partial \alpha_{ml}^{(r)}},\end{aligned} \tag{4.12}$$

donde, γ_r representa la tasa de aprendizaje.

Si ahora escribimos las derivadas parciales (4.11) como

$$\begin{aligned}\frac{\partial R_i}{\partial \beta_{km}} &= \delta_{ki} z_{mi},\\ \frac{\partial R_i}{\partial \alpha_{ml}} &= s_{mi} x_{il}.\end{aligned} \tag{4.13}$$

las cantidades δ_{ki} y s_{mi} son errores del modelo en las neuronas de la capa de salida y la capa oculta, respectivamente.

Por lo tanto, a partir de sus definiciones, estos errores satisfacen

$$s_{mi} = \sigma'(\alpha_m^T x_i) \sum_{k=1}^{K} \beta_{km} \delta_{ki}, \tag{4.14}$$

conocidas como ecuaciones de retropropagación.

Si utilizamos esto, las actualizaciones expuestas en las ecuaciones (4.12) se pueden implementar en un algoritmo de dos pasos, uno hacia delante y otro hacia atrás. En el paso hacia delante, se fijan los pesos actuales y se calculan los valores $\hat{f}_k(x_i)$ a partir de la ecuación (4.14). En el paso hacia atrás, se calculan los errores δ_{ki} y luego se realiza la retropropagación, utilizando la ecuación (4.14), para obtener los errores s_{mi}. Ambos conjuntos de errores se utilizan posteriormente para calcular los gradientes de las actualizaciones en (4.12), utilizando las ecuaciones (4.13).

Este procedimiento de dos pasos que acabamos de exponer es lo que se conoce, en el ámbito de las Redes Neuronales, como algoritmo de retropropagación. También se le ha denominado *regla delta*.

Las ventajas de este algoritmo radican en su naturaleza local y simple. Pues en el, cada unidad oculta solo pasa y recibe información con las unidades que comparte conexión.

La tasa de aprendizaje, γ_r, se suele tomar como una constante, pero también se puede optimizar mediante una búsqueda de línea que minimice la función de error dada en cada actualización. Con el aprendizaje en línea, γ_r debería disminuir a cero cuando $r \to \infty$. En este caso, siempre que $\gamma_r \to 0$, $\sum_r \gamma_r = \infty$ y $\sum_r \gamma_r^2 < \infty$ tenemos asegurada la convergencia del método. Un ejemplo muy habitual es utilizar $\gamma_r = \frac{1}{r}$.

La retropropagación puede llegar a ser muy lenta y por este motivo aparecen otros métodos de optimización como pueden ser el método *stochastic gradiant descent*, se puede ver en Amari (1993), el método *RMSprop*, se puede ver en Dauphin *et al.* (2015) y el método *Adam*, se puede consultar en Kingma y Ba (2014).

4.4. Problemas relacionados con el ajuste de las Redes Neuronales.

Ahora vamos a comentar algunos de los problemas más importantes que pueden aparecer a la hora de entrenar las Redes Neuronales. Este proceso es todo un arte pues, generalmente, el modelo está sobreparametrizado y el problema de optimización suele ser no convexo e inestable.

Valores iniciales.

Cabe destacar, que si los pesos son cercanos a cero, la parte operativa de la función sigmoide, ilustrada en la Figura 4.3, es aproximadamente lineal, lo cual provoca que la red neuronal colapse en un modelo aproximadamente lineal.

En general, los valores iniciales para los pesos se eligen aleatoriamente y próximos a cero. Por lo tanto, el modelo comienza de forma casi lineal y se vuelve no lineal a medida que se incrementan los pesos.

Las unidades individuales localizan las direcciones e introducen no linealidades donde sea necesario. Si utilizamos pesos iguales a cero, obtendremos derivadas nulas y una simetría perfecta, lo cuál provocará que nuestro algoritmo no se mueva. Sin embargo, si comenzamos utilizando pesos demasiado grandes, obtendremos soluciones pobres.

Sobreajuste de las Redes Neuronales.

Las Redes Neuronales tienen demasiados pesos y sobreajustan los datos en el mínimo global de R. En los primeros desarrollos de las Redes Neuronales, se utilizaba un criterio de parada para evitar el sobreajuste.

De esta forma, se entrena el modelo durante un tiempo y se detiene antes de acercarse al mínimo global. Debido a que, inicialmente, los pesos comienzan en una solución aproximadamente lineal, esto tiene el efecto de reducir el modelo final hacia un modelo lineal. En este caso, el conjunto de validación es útil para determinar cuando parar, pues esperamos que el error de validación empiece a aumentar.

Un método más explícito para la regularización es la disminución del peso, conocida como *weight decay*. Esta es análoga a la regresión de crestas utilizada para los modelos lineales y se puede ver en la Sección 3.4.1 de Hastie *et al.* (2001).

De esta forma, añadimos una penalización a la función del error $R(\theta) + \lambda\, J(\theta)$, donde

$$J(\theta) = \sum_{km} \beta_{km}^2 + \sum_{ml} \alpha_{ml}^2, \tag{4.15}$$

y $\lambda \geq 0$ es el parámetro de ajuste.

Es importante destacar, que si consideramos valores grandes de λ, los pesos tenderán a reducirse hacia cero. Normalmente, se suele utilizar la validación cruzada para estimar el valor de λ.

El efecto de la penalización radica en añadir los términos $2\beta_{km}$ y $2\alpha_{km}$ a las expresiones del gradiente vistas en (4.12).

Además de la penalización de la ecuación (4.15), se han propuesto otras formas de penalización, como por ejemplo,

$$J(\theta) = \sum_{km} \frac{\beta_{km}^2}{1+\beta_{km}^2} + \sum_{ml} \frac{\alpha_{ml}^2}{1+\alpha_{ml}^2}, \tag{4.16}$$

conocida como penalización por eliminación de peso (*weight elimination panlty*). A diferencia de la penalización expuesta en (4.15), esta tiene la capacidad de reducir más los pesos.

Al final de la Sección 11.5.2 de Hastie *et al.* (2001) se puede ver un ejemplo dónde se muestra como la disminución del peso mejora el rendimiento de la red neuronal para un caso particular.

Reescalado de las variables de entrada.

Debido a que el escalado de las variables de entrada establece el escalado de los pesos en la capa *input*, esto puede tener un gran impacto en la calidad de la solución final.

De esta forma, es mejor estandarizar todas las entradas de manera que tengan media cero y desviación típica uno, pues esto nos garantiza que todas las entradas van a ser tratadas por igual en el proceso de regularización. Además, nos va a permitir elegir un rango significativo para los pesos iniciales, que recordemos que deben ser aleatorios. Generalmente, si estandarizamos las entradas, es habitual tomar, de manera aleatoria, pesos uniformes en el rango $[-0.7, 0.7]$.

Número de unidades y capas ocultas.

En general, es mejor utilizar demasiadas unidades ocultas, en vez de utilizar muy pocas. Pues, con muy pocas unidades, el modelo podría no tener suficiente flexibilidad para captar todas las no linealidades subyacentes en los datos. Sin embargo, si utilizamos demasiadas unidades ocultas, y tenemos una regularización adecuada, los pesos adicionales van a reducirse a cero.

En general, el número de unidades ocultas oscila entre 5 y 100, aumentando el número de estas unidades a medida que aumentan el número de variables de entrada y el número de datos de entrenamiento. Lo más habitual es utilizar un número razonablemente grande de unidades y entrenarlas con regularización.

Algunos investigadores hacen uso de la validación cruzada para estimar el número óptimo de neuronas ocultas, pero esto no parece estrictamente necesario si utilizamos la validación cruzada para estimar el parámetro de regularización.

La elección del número de capas es un proceso basado en los conocimientos previos y la experiencia. Cada capa extrae características de las variables de entrada tanto para la regresión como para la clasificación. La utilización de múltiples capas ocultas nos va a permitir construir características jerárquicas a diferentes niveles. En la Sección 11.6 de Hastie *et al.* (2001) se puede ver un ejemplo en el cuál es eficaz la utilización de múltiples capas.

Problema de múltiples mínimos.

En esta situación, la función de error $R(\theta)$ no es convexa y posee muchos mínimos locales. En consecuencia, la solución que obtenemos depende bastante de la elección de los pesos iniciales.

En este caso, deberíamos probar varias configuraciones aleatorias de pesos iniciales y elegir aquella solución que nos proporcione el menor error penalizado. Probablemente, una mejor aproximación sería utilizar el *Bagging* en Redes Neuronales, este método promedia las predicciones de las Redes Neuronales de entrenamiento, obtenidas a partir de versiones perturbadas, aleatoriamente, de los datos de entrenamiento. Este procedimiento se describe detalladamente en la Sección 8.7 de Hastie *et al.* (2001).

Capítulo 5

Modelado del comportamiento de los distribuidores de Estrella Galicia.

En este capítulo vamos a aplicar, ilustrar y comentar las técnicas de aprendizaje estadístico expuestas en los Capítulos 2,3 y 4 sobre diferentes distribuidores de Tipo B de Estrella Galicia.

En la Sección 5.1 consideraremos diferentes distribuidores de la Zona A y de la Zona B e iremos comentando los resultados obtenidos para cada uno de ellos. Además, veremos como varía la importancia de las variables predictoras en función del producto solicitado y de la zona en la que actúa cada distribuidor.

En la Sección 5.2 se presentarán las conclusiones generales que hemos obtenido, mientras que en la Sección 5.3 se comentarán posibles mejoras que se podrían aplicar en un futuro para conseguir aumentar la precisión de las predicciones.

5.1. Comportamiento de los distribuidores de Estrella Galicia en función de la zona de actuación.

En esta sección nos centraremos en aplicar y comentar los resultados obtenidos al utilizar los métodos de aprendizaje estadístico descritos en los Capítulos 3 y 4 sobre diferentes distribuidores de Tipo B de Estrella Galicia.

Aprovechando la segmentación geográfica presente en la naturaleza de nuestros distribuidores, a modo de ejemplo, ilustraremos el funcionamiento de los modelos de aprendizaje estadístico sobre dos distribuidores cuya zona de actuación sea de interior y dos cuya zona sea costera.

En primer lugar, analizaremos un distribuidor del interior de la Zona A. En segundo lugar, vamos a analizar un distribuidor de la costa de la Zona A. En tercer lugar, consideraremos un distribuidor

situado en el interior de la Zona B. Finalmente, para terminar con este análisis sobre los diferentes distribuidores hemos seleccionado uno de la Zona B cuya zona de actuación sea de costa.

Antes de comenzar a aplicar los modelos *Random Forest*, explicados en el Capítulo 3, y los modelos de Redes Neuronales, explicados en el Capítulo 4, vamos a exponer ciertas consideraciones prácticas que hemos llevado a cabo para obtener los resultados de las tablas expuestas a lo largo de este capítulo.

- Después de realizar varias pruebas, hemos considerado las siguientes variables predictoras: **`Litros 1 pedido antes`**, **`Litros 2 pedidos antes`**, **`Litros 3 pedidos antes`**, **`Litros 4 pedidos antes`**, **`Semana`**, **`Mes`**, **`Ano`**, **`Promedio mes`**, **`Promedio mes distribuidores`**, **`Promedio semana`**, **`Promedio semana distribuidores`**, **`Estacion`**, **`Covid`**, **`Navidad`** y **`Temperatura`**. Todas estas variables se encuentran explicadas en la Sección 1.2 del Capítulo 1.

- Como variable respuesta hemos considerado la variable **`Cantidad pedido`**. Podríamos haber escogido la variable **`Litros pedidos`** pero, finalmente, nos hemos decantado por utilizar la cantidad de pedido, pues de esta forma nos aseguramos de que las predicciones obtenidas tengan sentido. En caso de utilizar la variable **`Litros pedidos`** pudiera ocurrir que los litros obtenidos en las predicciones no fuesen plausibles, pues probablemente estas predicciones no fuesen un múltiplo de los litros del pack de cada producto.

- Para los distribuidores de Tipo A consideraremos la siguiente cartera de productos: *Producto 1, Producto 2, Producto 3*.

- Para los distribuidores de Tipo B consideraremos la siguiente cartera de productos: *Producto 3, Producto 4, Producto 1*.

- Para cada distribuidor hemos considerado dividir su histórico de pedidos en dos subconjuntos. El primero estará formado por el 80 % de los datos, se denominará conjunto de entrenamiento. El 20 % restante formará el segundo subconjunto, el cual denominaremos conjunto de validación o test.

- Dentro de los modelos *Random Forest* hemos considerado dos formulaciones ligeramente diferentes.

  En la primera opción se ha ajustado un *Random Forest* básico. En este caso, se ha ajustado el modelo mediante el conjunto de entrenamiento y se han obtenido las predicciones sobre el conjunto de validación.

  En la segunda opción se ha ajustado un *Random Forest* retroalimentado, para ello hemos utilizado un bucle, en la primera iteración se ajusta el *Random Forest* sobre el conjunto de entrenamiento y se obtiene la predicción sobre el primer dato del conjunto de validación. Es importante destacar que, en este caso, el conjunto de validación siempre estará formado por un único pedido. De esta forma, en la segunda iteración se realizaría lo mismo, pero en este caso el conjunto de entrenamiento estaría formado por el 80 % inicial y se le añadiría el dato utilizado para el conjunto

de validación en la iteración anterior. Ahora, para el conjunto de validación se utilizaría el segundo pedido del conjunto de validación original. Este proceso se repetirá de forma iterativa hasta obtener un conjunto de entrenamiento formado por los $n-1$ pedidos históricos, donde n denota el total de pedidos del distribuidor en el período considerado.

- Para los modelos de Redes Neuronales se ha utilizado un procedimiento similar al utilizado para los modelos *Random Forest*. En primer lugar, se optó por ajustar un modelo de Redes Neuronales básico, mientras que luego se ajustó un modelo de Redes Neuronales retroalimentado utilizando un procedimiento análogo al empleado en el modelo *Random Forest* retroalimentado.

- Para obtener las predicciones hemos realizado un post-procesado de las mismas en el que hemos aplicado un redondeo, pues la variable `Cantidad pedido` solo admite valores enteros. Además, hemos considerado que todas las predicciones que fuesen menores que el mínimo de los pedidos se estimarían como nulas, es decir, como `Cantidad pedido` igual a cero.

- Para cuantificar el rendimiento predictivo de nuestros modelos hemos utilizado el MAE (Error Absoluto Medio), se puede consultar en la Sección 3.3 de Shmueli (2016). Es importante destacar que esta medida se aplica sobre el conjunto de validación, pues ya hemos visto que este permite evaluar la precisión predictiva de nuestros modelos. Recordemos que los datos del conjunto de validación no se utilizan para seleccionar las variables predictoras ni para estimar los parámetros de los modelos.

  El **MAE** nos proporciona el promedio de los errores en términos absolutos y tiene una fórmula similar a la métrica TMSE descrita en la ecuación (2.5) y utilizada para medir la calidad de los ajustes.

$$\text{TMAE} = \frac{1}{m}\sum_{i=1}^{m}\left|y_{t_i} - \hat{f}(x_{t_i})\right| = \frac{1}{m}\sum_{i=1}^{m}\left|y_i - \hat{y}_i\right|, \tag{5.1}$$

  donde y_i representa el valor real, $\hat{y}_i$ el valor de la predicción y m hace referencia al tamaño del conjunto de validación.

  Cabe destacar que para los modelos retroalimentados siempre vamos a tener $m = 1$. De esta forma, vamos a obtener m valores distintos del MAE.

- Además de los modelos *Random Forest* y los modelos de Redes Neuronales, también hemos creado un modelo base. Este nos servirá para hacernos una idea de lo útiles que son nuestros modelos de aprendizaje estadístico.

  Los modelos base se utilizan, principalmente, con dos propósitos:

  1. Obtener las previsiones reales de los pedidos. Los modelos base son fáciles de entender y se implementan con relativa facilidad. Además, en ocasiones, pueden alcanzar unos niveles de precisión bastante elevados. Por lo tanto, muchas veces, estos modelos se consideran verdaderos competidores frente a otros más sofisticados, como los analizados en este trabajo.

2. Se utilizarán como una línea de partida. Notemos que, al calcular el rendimiento predictivo de un cierto modelo es muy importante poder compararlo con un modelo base o de referencia. De esta forma, los modelos base deben ser considerados como una línea de partida y la ventaja comparativa de cualquier otro modelo que consideremos debe ser evidente para justificar su utilización.

Del mismo modo que sucede con los otros modelos de aprendizaje estadístico, la capacidad predictiva de los modelos base se evalúa sobre el conjunto de validación.

En nuestro caso, para la creación del modelo base hemos utilizado, para cada distribuidor, la media anual del histórico de litros pedidos para cada uno de los productos de su cartera.

A continuación, ilustraremos y comentaremos los resultados obtenidos utilizando los diferentes modelos de aprendizaje estadístico, expuestos a lo largo de este trabajo. Estos métodos serán aplicados sobre diferentes distribuidores de Tipo B de Estrella Galicia.

Distribuidor del interior de la Zona A.

En primer lugar, vamos a analizar un distribuidor del interior de la Zona A. Para este distribuidor y en el espacio temporal que hemos considerado, recordemos que abarca desde Diciembre de 2017 hasta el 1 de Septiembre de 2021, contamos con un histórico total de 481 pedidos. Entre ellos, podemos destacar 133 pedidos de *Producto 3*, 143 de *Producto 1* y 156 de *Producto 2*.

En la Tabla 5.1 podemos ver los resultados del MAE obtenidos sobre el conjunto de validación después de aplicar los modelos de aprendizaje estadístico. Además, se puede ver que, en los casos estudiados, los modelos *Random Forest* son los que proporcionan los mejores resultados.

Método	**MAE (Error Absoluto Medio)**		
	Producto 3	*Producto 1*	*Producto 2*
Modelo base	319,21	22,35	85,23
Random Forest	31,42	9,20	34,25
Random Forest retroalimentado	33,17	9,11	35,49
Red Neuronal	116,54	32.20	58,90
Red Neuronal retroalimentada	113,59	16,13	74,19

Tabla 5.1: MAE obtenido después de aplicar los diferentes métodos de aprendizaje estadístico.

Es importante destacar que, para este distribuidor particular, tenemos una gran cantidad de pedidos nulos, pues el 72,35 % de los pedidos del *Producto 3*, el 70,27 % del *Producto 1* y el 67,57 % del *Producto 2* son nulos. Por lo tanto, debido a la gran cantidad de pedidos nulos, estamos interesados en obtener la proporción de acierto de nuestros modelos sobre estos pedidos. Recordemos que con pedido nulo nos estamos refiriendo a aquel pedido en el cual el distribuidor no solicita el producto que estamos analizando. Para ello vamos a utilizar la expresión dada en la ecuación (5.2), donde $\mathbb{I}$ representa la

función indicadora, y_i el valor real del pedido, $\hat{y}_i$ el valor de la predicción y m hace referencia al tamaño del conjunto de validación.

$$P = \frac{\sum_{i=1}^{m} \mathbb{I}\left(y_i = 0,\, \hat{y}_i = 0\right)}{\sum_{i=1}^{m} \mathbb{I}(y_i = 0)}. \tag{5.2}$$

En la Tabla 5.2 mostramos la proporción de acierto sobre los pedidos nulos. Es importante destacar que el modelo no tiene conocimiento, a priori, sobre si se encuentra ante un pedido nulo o no. Además, en este caso, no tiene sentido considerar el modelo base, pues este nunca va a predecir un valor nulo. Nuevamente, se puede ver que los modelos *Random Forest* son los que tienen una mayor capacidad para detectar los pedidos nulos, pues en todos los casos tienen una proporción de acierto superior al 75 %, llegando, en algún caso, a obtener el 100 %.

Método	**Proporción acierto pedidos nulos**		
	Producto 3	*Producto 1*	*Producto 2*
Random Forest	100 %	77,36 %	83,33 %
Random Forest retroalimentado	100 %	79,25 %	85,18 %
Red Neuronal	100 %	69,81 %	66,67 %
Red Neuronal retroalimentada	84,62 %	50,94 %	57,41 %

Tabla 5.2: Proporción de acierto sobre los pedidos nulos para el distribuidor del interior de la Zona A.

Debido a la gran cantidad de pedidos nulos presente en el histórico de nuestro distribuidor y la alta proporción de acierto en estos, es muy probable que la medida del MAE, dada anteriormente, se vea altamente afectada, pues esto provoca que el MAE medio se reduzca. Por lo tanto, para tener una visión más general de los errores cometidos vamos a utilizar el MAPE (Error Porcentual Absoluto Medio), se puede consultar en la Sección 3.3 de Shmueli (2016).

El MAPE nos da una puntuación porcentual de cómo se desvían, en promedio, las predicciones de los valores reales, aunque es importante destacar que esta medida solo se puede aplicar a los pedidos no nulos, es decir, sobre aquellos en los que la cantidad de pedido es positiva. Podemos ver su expresión en la ecuación (5.3), donde $\mathbb{I}$ representa la función indicadora, y_i el valor real real del pedido e $\hat{y}_i$ el valor de la predicción.

$$\text{TMAPE} = \frac{1}{\sum \mathbb{I}(y_i > 0)} \sum_{i:\, y_i > 0} \left| \frac{y_{t_i} - \hat{f}(x_{t_i})}{y_{t_i}} \right| \times 100 = \frac{1}{\sum \mathbb{I}(y_i > 0)} \sum_{i:\, y_i > 0} \left| \frac{y_i - \hat{y}_i}{y_i} \right| \times 100. \tag{5.3}$$

En la Tabla 5.3 podemos ver los resultados del MAPE obtenidos con los diferentes modelos. Estos valores se han obtenido sobre los datos del conjunto de validación cuya cantidad de pedido es no nula. De nuevo, se puede ver que en los casos estudiados los modelos *Random Forest* son los que proporcionan los mejores resultados en cuanto a MAPE. Además, cabe destacar que para el *Producto 1* el modelo base funciona mejor que la Red Neuronal, mientras que si consideramos el *Producto 2* tanto la Red

Neuronal retroalimentada como el modelo base nos proporcionan resultados similares.

Método	**MAPE (Error Porcentual Absoluto Medio)**		
	Producto 3	*Producto 1*	*Producto 2*
Modelo base	66,02 %	73,77 %	69,24 %
Random Forest	14,51 %	43,98 %	31,34 %
Random Forest retroalimentado	15,78 %	40,56 %	35,10 %
Red Neuronal	52,35 %	91,20 %	43,57 %
Red Neuronal retroalimentada	37,52 %	58,07 %	68,24 %

Tabla 5.3: MAPE obtenido sobre los pedidos no nulos para el distribuidor del interior de la Zona A.

Para poder hacernos una idea de como son los errores cometidos en los pedidos del conjunto de validación, en la Figura 5.1 vamos a ilustrar el histograma y la función de densidad correspondiente a los errores cometidos utilizando el modelo *Random Forest* retroalimentado y el *Producto 1*.

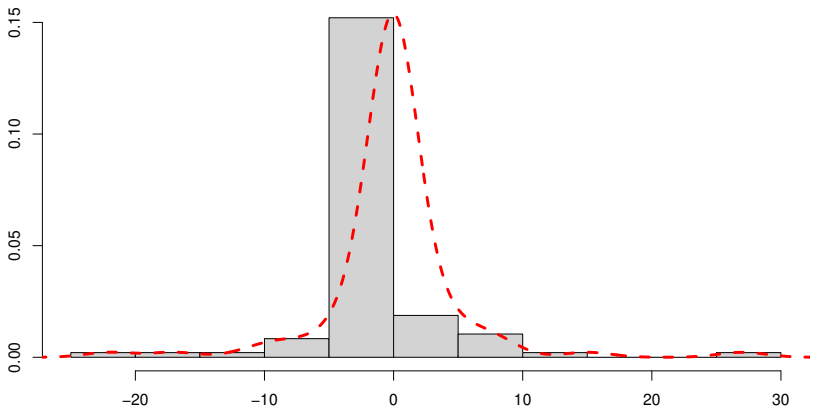


Figura 5.1: Histograma de las diferencias entre las predicciones y los pedidos reales del distribuidor del interior de la Zona A, considerando el material *Producto 1* y el modelo *Random Forest* Retroalimentado.

A partir del histograma de la Figura 5.1 se puede afirmar que lo más frecuente es obtener estimaciones inferiores al pedido real, esto nos indica que nuestro modelo no es capaz de captar de forma adecuada los incrementos en los pedidos de nuestro distribuidor.

En la Figura 5.2 se muestran las variables más importantes para el modelo *Random Forest*. De esta forma se puede ver que, para este distribuidor y considerando el material *Producto 1*, las variables más influyentes son las relativas a los litros de los cuatro pedidos anteriores, aunque la más influye es la variable `Litros 2 pedidos antes`. Del mismo modo, la variable menos influyente es la `Navidad`.

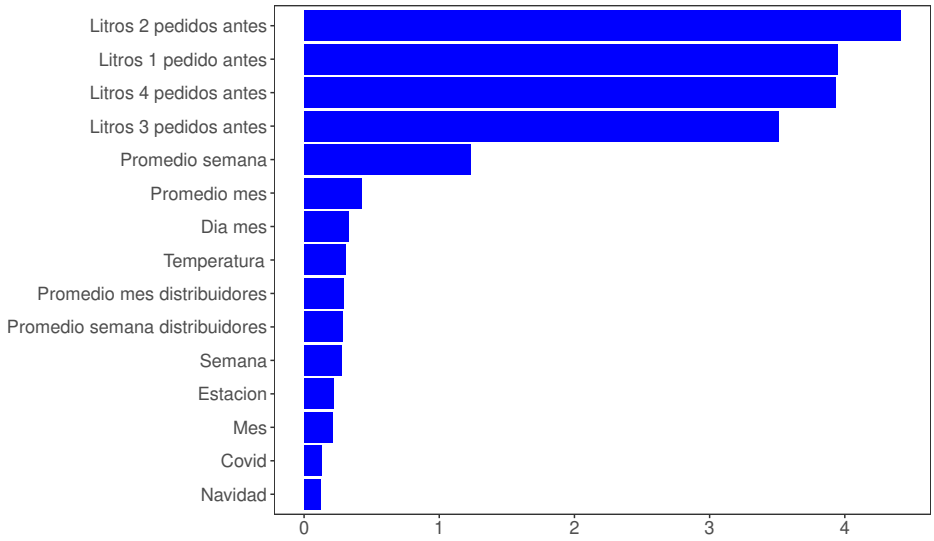


Figura 5.2: Diagrama de barras para las variables más importantes del modelo *Random Forest*, considerando el material *Producto 1* y el distribuidor del interior de la Zona A.

Para terminar con el análisis de este distribuidor, en la Tabla 5.4 se muestra el funcionamiento del modelo *Random Forest* retroalimentado utilizado para la predicción de la cartera de los distribuidores. Hemos obtenido la predicción de la cartera para sus últimos cinco pedidos. En ella solo hemos considerado los tres productos destacados: *Producto 3*, *Producto 1* y *Producto 2*. Sin embargo, este procedimiento se puede generalizar a más productos.

Pedido	***Producto 3***		***Producto 1***		***Producto 2***	
n	Valor Real	Predicción	Valor Real	Predicción	Valor Real	Predicción
477	0	4	0	0	0	0
478	720	674	0	7	120	121
479	0	0	0	0	0	0
480	0	0	0	0	0	0
481	0	0	0	0	0	0

Tabla 5.4: Tabla comparativa, en unidades, de los pedidos reales y las predicciones obtenidas para los diferentes productos considerando el distribuidor del interior de la Zona A.

De esta forma, a partir de la Tabla 5.4 se realiza la siguiente interpretación. Consideremos el pedido $n = 478$, en este caso, nuestro modelo predice que el distribuidor nos va a solicitar 674 unidades de *Producto 3*, 7 de *Producto 1* y 121 de *Producto 2*. Mientras que el valor del pedido real son 720 unidades de *Producto 3*, 0 de *Producto 1* y 120 de *Producto 2*.

Distribuidor de una zona costera de la Zona A.

En segundo lugar, hemos decidido analizar un distribuidor de Tipo B situado en la costa de la Zona A. En este caso contamos con un histórico total de 513 pedidos. Entre ellos, podemos destacar 183 pedidos de *Producto 3*, 174 de *Producto 1* y 197 de *Producto 2*.

En la Tabla 5.5 podemos ver los resultados del MAE obtenidos sobre el conjunto de validación después de aplicar los modelos de aprendizaje estadístico. Se puede observar que, en los productos estudiados, los modelos que obtienen los mejores resultados en cuanto a MAE son los *Random Forest*.

Método	**MAE (Error Absoluto Medio)**		
	Producto 3	*Producto 1*	*Producto 2*
Modelo base	327,10	15,68	74,77
Random Forest	46,87	3,61	17,26
Random Forest retroalimentado	43,34	3,80	18,24
Red Neuronal	113,72	11,94	49,41
Red Neuronal retroalimentada	122,22	14,65	55,59

Tabla 5.5: MAE obtenido después de aplicar los diferentes métodos de aprendizaje estadístico.

En este caso, al igual que sucedía con el distribuidor anterior, volvemos a tener una gran cantidad de pedidos nulos, pues el 64,33 % de los pedidos del *Producto 3*, el 66,08 % del *Producto 1* y el 62,57 % del *Producto 2* son nulos. Por lo tanto, siguiendo un razonamiento análogo y utilizando la ecuación (5.2) vamos a calcular la proporción de acierto en estos pedidos.

En la Tabla 5.6 mostramos la proporción de acierto sobre los pedidos nulos para este distribuidor. Se puede ver que, en todos los casos estudiados, excepto si consideramos la Red Neuronal y el *Producto 3*, los modelos *Random Forest* son los que tienen una mayor capacidad para detectar los pedidos nulos, pues en cualquiera de los productos estudiados obtiene una probabilidad de acierto superior al 89 %. Además, cabe destacar que en el caso particular en el que estos métodos son superados por la Red Neuronal la diferencia en la proporción de acierto no llega al 4 %.

Método	**Proporción acierto pedidos nulos**		
	Producto 3	*Producto 1*	*Producto 2*
Random Forest	96,77 %	100 %	89,40 %
Random Forest retroalimentado	96,78 %	100 %	89,40 %
Red Neuronal	100 %	70,59 %	72,72 %
Red Neuronal retroalimentada	87,09 %	67,65 %	87,10 %

Tabla 5.6: Proporción de acierto sobre los pedidos nulos para el distribuidor situado en la costa de la Zona A.

La gran cantidad de pedidos nulos presente en el histórico de nuestro distribuidor y la alta proporción de acierto en estos pedidos distorsiona los resultados obtenidos con el MAE en la Tabla 5.5. Por lo tanto, para tener una visión más general de los errores cometidos con los diferentes modelos vamos a utilizar el MAPE.

En la Tabla 5.7 mostramos los resultados del MAPE obtenidos con los diferentes modelos de aprendizaje estadístico. Estos valores han sido obtenidos sobre los datos del conjunto de validación cuya cantidad de pedido es positiva. Nuevamente, se puede observar que, en los casos estudiados, los modelos *Random Forest* son los que proporcionan un MAPE más bajo. Cabe destacar que para el *Producto 1* la Red Neuronal retroalimentada proporciona peores resultados que nuestro modelo base.

Método	**MAPE (Error Porcentual Absoluto Medio)**		
	Producto 3	*Producto 1*	*Producto 2*
Modelo base	53,87 %	65,13 %	61,20 %
Random Forest	13,59 %	36,34 %	16,94 %
Random Forest retroalimentado	12,42 %	37,98 %	18,51 %
Red Neuronal	42,98 %	62,39 %	45,91 %
Red Neuronal retroalimentada	35,01 %	70,15 %	35,02 %

Tabla 5.7: MAPE obtenido sobre los pedidos no nulos para el distribuidor situado en la costa de la Zona A.

A continuación, vamos a obtener el histograma y la función de densidad correspondiente para los errores cometidos utilizando el modelo *Random Forest* retroalimentado y el material *Producto 3*. Este se puede ver en la Figura 5.3.

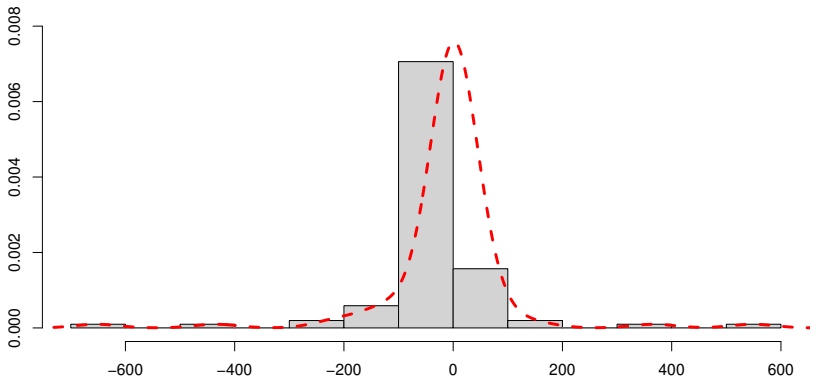


Figura 5.3: Histograma de las diferencias entre las predicciones y los pedidos reales del distribuidor de costa de la Zona A, considerando el material *Producto 3* y el modelo *Random Forest* retroalimentado.

En esta figura se puede ver que, nuevamente, nuestro modelo tiende a predecir por debajo del valor real de los pedidos. Además, tanto en la parte derecha del gráfico como en la izquierda tenemos valores muy grandes, lo cual nos indica que nuestro modelo no es capaz a captar de forma correcta los patrones de subida y bajada para este distribuidor. Esto puede ser debido a la ausencia de alguna covariable influyente o determinante para la predicción de nuestro modelo.

En la Figura 5.4 se pueden ver cuales son las variables más importantes para el modelo *Random Forest* en el caso de considerar el material *Producto 3* y este distribuidor.

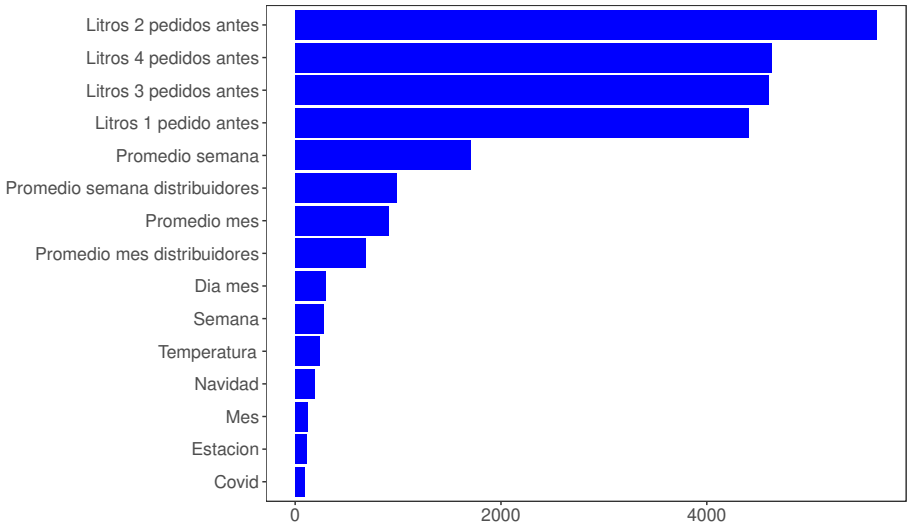


Figura 5.4: Diagrama de barras para las variables más importantes del modelo *Random Forest* considerando el material *Producto 3*

Al igual que en el caso anterior, se puede apreciar que las variables más importantes son las relativas a los litros de los cuatro pedidos anteriores, aunque en este caso varía ligeramente el orden. Nuevamente, destaca la variable `Litros 2 pedidos antes`. Sin embargo, en este caso, la variable menos importante es el `Covid`.

Por último, en la Tabla 5.8 ilustraremos el funcionamiento del modelo *Random Forest* retroalimentado utilizado para la predicción de la cartera de los diferentes distribuidores de Tipo B de Estrella Galicia. En este caso, hemos considerado la predicción de los tres productos analizados en la Tabla 5.5 para los últimos cinco pedidos.

De esta forma, a partir de la Tabla 5.8 se puede ver que, para este distribuidor, con los productos seleccionados y para los últimos cinco pedidos, nuestro modelo es capaz de interpretar, correctamente, todos los pedidos nulos.

Pedido	***Producto 3***		***Producto 1***		***Producto 2***	
n	Valor Real	Predicción	Valor Real	Predicción	Valor Real	Predicción
509	0	0	0	0	0	0
510	660	692	120	107	24	18
511	0	0	0	0	0	0
512	0	0	0	0	0	0
513	0	0	0	0	0	0

Tabla 5.8: Tabla comparativa, en unidades, de los pedidos reales y las predicciones obtenidas para los diferentes productos considerando el distribuidor costero de la Zona A.

Distribuidor del interior de la Zona B.

En tercer lugar, hemos considerado un distribuidor de Tipo B del interior de la Zona B. Este distribuidor cuenta con un histórico total de 481 pedidos. Entre ellos, podemos destacar 267 pedidos de *Producto 3*, 230 de *Producto 4* y 235 de *Producto 1*.

En la Tabla 5.9 podemos ver los resultados del MAE obtenidos sobre el conjunto de validación después de aplicar los modelos de aprendizaje estadístico. Una vez más, en los casos estudiados, los modelos *Random Forest* son los que proporcionan los mejores resultados. En este caso, cabe destacar que el MAE obtenido al aplicar la Red Neuronal sobre el *Producto* 4 supera el obtenido con el modelo base.

Método	**MAE (Error Absoluto Medio)**		
	Producto 3	*Producto 4*	*Producto 1*
Modelo base	433, 42	96, 96	21, 01
Random Forest	251, 64	34, 60	11, 33
Random Forest retroalimentado	238, 28	32, 42	11, 99
Red Neuronal	342, 43	115, 90	14, 06
Red Neuronal retroalimentada	370, 32	79, 10	18, 84

Tabla 5.9: MAE obtenido después de aplicar los diferentes métodos de aprendizaje estadístico.

Al igual que sucedía con los distribuidores de la Zona A, también tenemos pedidos nulos, aunque para este distribuidor particular la proporción de pedidos nulos y no nulos es más homogénea, pues, en este caso, tenemos que el 44, 49 % de los pedidos del *Producto 3*, el 52, 18 % del *Producto 4* y el 51, 14 % del *Producto 1* son nulos. Por lo tanto, para obtener la proporción de acierto en los pedidos nulos vamos a repetir el proceso llevado a cabo en los dos distribuidores anteriores.

En la Tabla 5.10 mostramos la proporción de acierto obtenida con los diferentes modelos sobre los pedidos nulos. En este caso podemos ver que en todos los casos estudiados, excepto si consideramos la Red Neuronal y el *Producto 3*, los modelos *Random Forest* son los que tienen una mayor capacidad para detectar estos pedidos. Además, cabe destacar que para este distribuidor se reduce la proporción de acierto, independientemente del modelo utilizado.

Método	**Proporción acierto pedidos nulos**		
	Producto 3	*Producto 4*	*Producto 1*
Random Forest	27,50 %	83,30 %	80,00 %
Random Forest retroalimentado	30,00 %	85,19 %	76,00 %
Red Neuronal	62,50 %	9,25 %	68,00 %
Red Neuronal retroalimentada	20,00 %	24,07 %	28,00 %

Tabla 5.10: Proporción de acierto sobre los pedidos nulos para el distribuidor situado en el interior de la Zona B.

En la Tabla 5.11 podemos ver los resultados del MAPE obtenidos sobre el conjunto de validación después de aplicar los diferentes modelos de aprendizaje estadístico sobre los pedidos no nulos. Una vez más, se puede ver que, en los casos estudiados, los modelos *Random Forest* son los que proporcionan los mejores resultados, aunque es importante resaltar que para este distribuidor los resultados generales del MAPE empeoran y las diferencias entre modelos no son tan significativas. Además, destaca que si consideramos el *Producto 4* el modelo base obtiene mejores resultados que la Red Neuronal, mientras que si consideramos el *Producto 3* el modelo base mejora el rendimiento de los dos modelos de Redes Neuronales.

Método	**MAPE (Error Porcentual Absoluto Medio)**		
	Producto 3	*Producto 4*	*Producto 1*
Modelo base	63,20 %	44,65 %	47,50 %
Random Forest	46,53 %	32,98 %	39,39 %
Random Forest retroalimentado	52,08 %	32,86 %	37,23 %
Red Neuronal	72,06 %	58,77 %	45,44 %
Red Neuronal retroalimentada	68,32 %	40,63 %	41,32 %

Tabla 5.11: MAPE obtenido sobre los pedidos no nulos para el distribuidor situado en el interior de la Zona B.

A continuación, vamos a obtener el histograma y la función de densidad correspondiente a los errores cometidos utilizando el modelo *Random Forest* retroalimentado y el material *Producto 1*. Este se puede ver en la Figura 5.5.

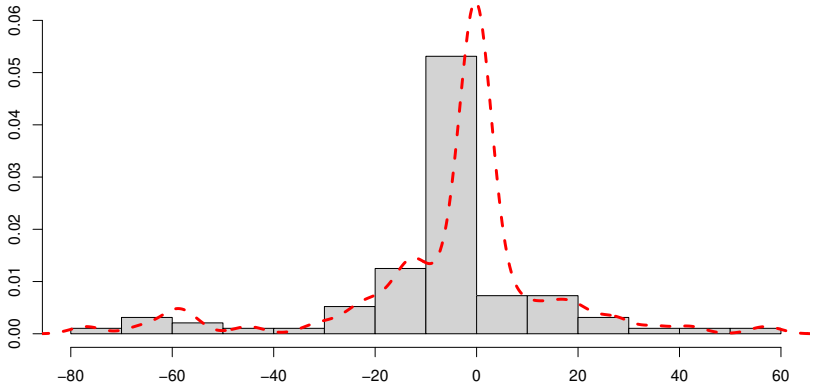


Figura 5.5: Histograma de las diferencias entre las predicciones y los pedidos reales del distribuidor situado en el interior de la Zona B, considerando el material *Producto 1* y el modelo *Random Forest* retroalimentado.

A partir del histograma de la Figura 5.5 se puede ver que nuestro modelo tiende a predecir por debajo del valor real de la cantidad de pedido. Además, la escala negativa es más amplia que la positiva, lo cual nos indica, nuevamente, que nuestro modelo no es capaz a interpretar de forma correcta las tendencias al alza en la cantidad de los pedidos.

En la Figura 5.6 se pueden ver cuales son las variables más importantes para el modelo *Random Forest* en el caso de considerar el material *Producto 1* y este distribuidor.

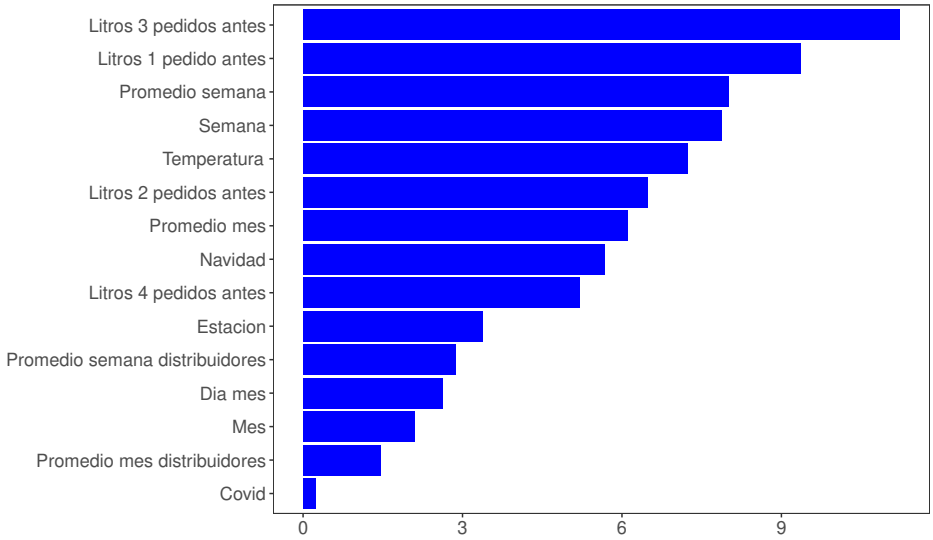


Figura 5.6: Diagrama de barras para las variables más importantes del modelo *Random Forest* considerando el material *Producto 1*.

Cabe destacar que, en este caso, entre las cuatro variables más importantes se encuentran la `Semana` y el `Promedio semana`, aunque la variable más importante es `Litros 3 pedidos antes`. Esto nos puede indicar que se trata de un distribuidor con un patrón bastante estacional, de hecho, la variable `Estación` también tiene cierta relevancia. Además, la variable `Temperatura` también se encuentra entre las variables más importantes.

Es bastante destacable que entre las variables menos relevantes se encuentre el `Promedio mes distribuidores`, esto nos puede indicar que se trata de un distribuidor peculiar. Nuevamente, la variable menos importante es el `Covid`.

Finalmente, en la Tabla 5.12 ilustramos el funcionamiento del método *Random Forest* retroalimentado para la predicción de la cartera de este distribuidor. Al igual que en los casos anteriores, lo haremos para los últimos cinco pedidos, aunque en este caso hemos considerado los productos: *Producto 3*, *Producto 4* y *Producto 1*.

Pedido	***Producto 3***		***Producto 4***		***Producto 1***	
n	Valor Real	Predicción	Valor Real	Predicción	Valor Real	Predicción
477	960	486	70	80	30	32
478	60	448	0	50	0	17
479	540	546	70	73	36	25
480	360	370	140	126	42	30
481	0	0	0	57	0	24

Tabla 5.12: Tabla comparativa, en unidades, de los pedidos reales y las predicciones obtenidas para los diferentes productos considerando el distribuidor del interior de la Zona B.

Por lo tanto, si a partir de la Tabla 5.12 consideramos, a modo de ejemplo, el pedido $n = 480$, nuestro modelo predice que el distribuidor nos va a solicitar 370 unidades de *Producto 3*, 126 de *Producto 4* y 30 de *Producto 1*. Sin embargo, el pedido real fueron 360 unidades de *Producto 3*, 140 de *Producto 4* y 42 de *Producto 1*.

Distribuidor situado en la costa de la Zona B.

En último lugar, hemos considerado un distribuidor de Tipo B situado en una zona costera y turística de la Zona B. Este cuenta con un histórico de 664 pedidos. Entre ellos, podemos destacar 450 pedidos del *Producto 3*, 322 del *Producto 4* y 456 del *Producto 1*.

En la Tabla 5.13 podemos ver los resultados del MAE, sobre el conjunto de validación, obtenidos con los diferentes modelos de aprendizaje estadístico. Nuevamente, en los productos estudiados, se observa que los métodos que proporcionan un MAE más pequeño son los *Random Forest*, aunque en este caso, excepto para el *Producto 4*, la mejora con respecto al modelo base ya no es tan significativa. Además, cabe destacar que los modelos de Redes Neuronales no logran obtener una clara mejoría con respecto al modelo base, pues para el caso del *Producto* 1 los resultados obtenidos son similares.

Método	**MAE (Error Absoluto Medio)**		
	Producto 3	*Producto 4*	*Producto 1*
Modelo base	886,08	154,44	182,1
Random Forest	651,53	32,33	137,11
Random Forest retroalimentado	667,01	29,70	143,23
Red Neuronal	766,32	94,98	180,45
Red Neuronal retroalimentada	836,32	96,02	181,15

Tabla 5.13: MAE obtenido después de aplicar los diferentes métodos de aprendizaje estadístico.

Al igual que sucedía con todos los distribuidores previos, en este caso también tenemos pedidos nulos, aunque es importante destacar que este es el distribuidor con menor proporción, pues solo el 32,22 % de los pedidos del *Producto 3*, el 51,55 % del *Producto 4* y el 31,33 % del *Producto 1* son nulos. Por lo tanto, aplicaremos un procedimiento análogo al que venimos desarrollando para calcular la proporción de acierto sobre estos pedidos.

En la Tabla 5.14 mostramos la proporción de acierto sobre los pedidos nulos para este distribuidor. Nuevamente, podemos ver que en todos los casos estudiados, excepto si consideramos la Red Neuronal y el *Producto 3*, los modelos *Random Forest* son los que tienen una mayor capacidad para detectar los pedidos nulos, aunque en este caso, a excepción del *Producto 4*, esta proporción baja considerablemente con respecto a los distribuidores anteriores. Esto probablemente sea debido a que se trata del distribuidor con menor proporción de pedidos nulos.

Método	**Proporción acierto pedidos nulos**		
	Producto 3	*Producto 4*	*Producto 1*
Random Forest	11,9 %	92,94 %	20,40 %
Random Forest retroalimentado	16,67 %	92,94 %	30,61 %
Red Neuronal	47,62 %	74,12 %	22,45 %
Red Neuronal retroalimentada	14,29 %	87,71 %	14,29 %

Tabla 5.14: Proporción de acierto sobre los pedidos nulos para el distribuidor situado en la costa de la Zona B.

En la Tabla 5.15 podemos ver los resultados del MAPE después de aplicar los diferentes modelos de aprendizaje estadístico sobre los pedidos no nulos. Cabe destacar que, en este caso, el MAPE obtenido con la Red Neuronal y el *Producto 3* es más pequeño que el obtenido con los modelos *Random Forest*, aunque la diferencia es mínima. En los demás casos, el *Random Forest* vuelve a obtener mejores resultados. Es importante destacar que si consideramos el *Producto 4* el modelo base mejora el rendimiento del modelo de Red Neuronal retroalimentada, mientras que si consideramos el *Producto 1* el modelo base obtiene mejores resultados que cualquiera de los modelos de Redes Neuronales.

Método	**MAPE (Error Porcentual Absoluto Medio)**		
	Producto 3	*Producto 4*	*Producto 1*
Modelo base	90,62 %	60,20 %	45,05 %
Random Forest	76,10 %	15,14 %	38,49 %
Random Forest retroalimentado	77,74 %	16,88 %	43,59 %
Red Neuronal	74,16 %	42,15 %	70,84 %
Red Neuronal retroalimentada	84,41 %	61,41 %	52,57 %

Tabla 5.15: MAPE obtenido sobre los pedidos no nulos para el distribuidor situado en la costa de la Zona B.

A continuación, vamos a obtener el histograma y la función de densidad correspondiente a los errores cometidos utilizando el modelo *Random Forest* retroalimentado y el material *Producto 3*. Este se puede ver en la Figura 5.7.

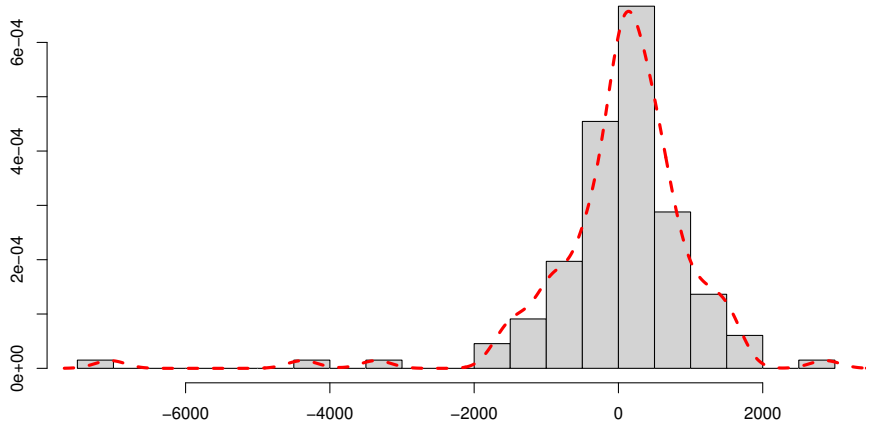


Figura 5.7: Histograma de las diferencias entre las predicciones y los pedidos reales del distribuidor situado en la costa de la Zona B, considerando el material *Producto 3* y el modelo *Random Forest* retroalimentado.

En la Figura 5.7 se aprecia, una vez más, que nuestro modelo tiende a predecir por debajo del valor real de la cantidad de pedido. Además, la escala negativa es mucho más amplia que la positiva, lo cual nos indica que nuestro modelo no es capaz de interpretar, correctamente, cuando aumenta la cantidad de pedido. Asimismo, si nos fijamos en la parte izquierda del gráfico llama la atención que para algún pedido esporádico la estimación se queda muy por debajo del pedido real, este hecho no es habitual y por lo tanto podríamos pensar que esos pedidos son atípicos o que se produjo algún error a la hora de almacenar los datos.

En la Figura 5.8 se muestran las variables más importantes para el modelo *Random Forest* considerando este distribuidor y el material *Producto 3*.

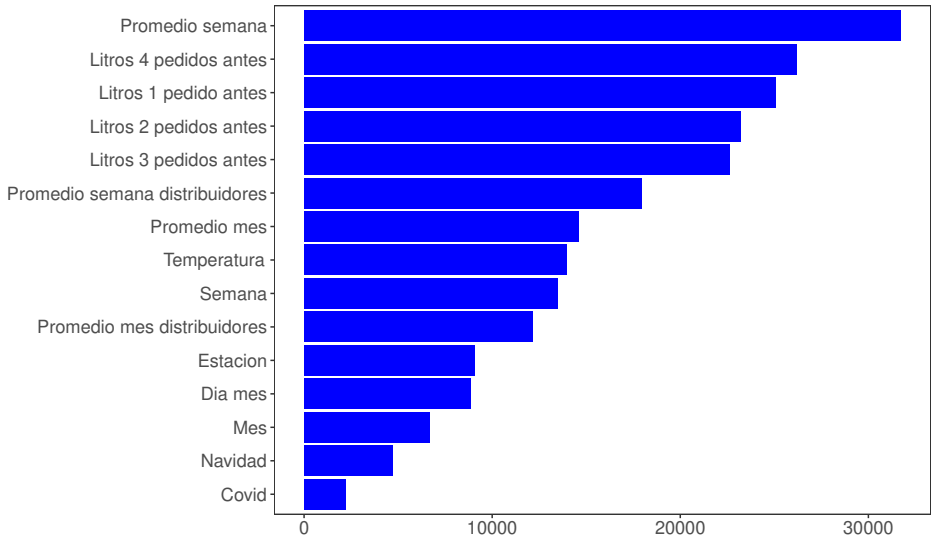


Figura 5.8: Diagrama de barras para las variables más importantes del modelo *Random Forest* considerando el material *Producto 3*.

Entre las variables más importantes se encuentran los litros de los cuatro pedidos anteriores, aunque en este caso cabe destacar que la variable más importante es `Promedio semana`. Esto parece indicar que, para este material, el distribuidor es bastante estable en sus pedidos semanales.

Entre las variables menos importantes se encuentran la `Navidad` y el `Covid`. Una vez más, esta última es la menos relevante para el modelo utilizado.

En la Tabla 5.16 ilustramos el funcionamiento del método *Random Forest* retroalimentado para la predicción de la cartera de este distribuidor. Del mismo modo que para los demás distribuidores, la predicción se realizará para los últimos cinco pedidos.

Pedido	***Producto 3***		***Producto 4***		***Producto 1***	
n	Valor Real	Predicción	Valor Real	Predicción	Valor Real	Predicción
660	1440	838	0	0	0	64
661	240	846	0	0	0	221
662	1440	505	0	0	0	42
663	480	727	140	157	192	197
664	1440	942	140	148	288	290

Tabla 5.16: Tabla comparativa, en unidades, de los pedidos reales y las predicciones obtenidas para los diferentes productos considerando el distribuidor costero de la Zona B.

Para finalizar el análisis de este distribuidor vamos a considerar, a modo de ejemplo, la predicción para el último pedido, $n = 664$. En este caso, como se puede ver en la Tabla 5.16, esperamos recibir un pedido de 942 unidades de *Producto 3*, 148 de *Producto 4* y 290 de *Producto 1*. No obstante, el pedido real fue de 1440 unidades de *Producto 3*, 140 de *Producto 4* y 288 de *Producto 1*. En los demás pedidos se realizaría una interpretación análoga.

5.2. Conclusiones.

A continuación, vamos a exponer las conclusiones generales que hemos obtenido.

- A partir de las Tablas 5.1, 5.5, 5.9 y 5.13 podemos ver que, en todos los productos estudiados para los diferentes distribuidores de Tipo B de Estrella Galicia, los modelos que nos proporcionan los mejores resultados en cuanto a MAE son los *Random Forest*. Independientemente del *Random Forest* que consideremos.

- A partir de las Tablas 5.3, 5.7, 5.11 y 5.15 podemos ver que en todos los productos estudiados para los diferentes distribuidores de Tipo B de Estrella Galicia, excepto en el caso del distribuidor situado en la costa de la Zona B considerando el *Producto 3*, los modelos que nos proporcionan los mejores resultados en cuanto a MAPE son los *Random Forest*. Independientemente del *Random Forest* que consideremos.

- Cabe destacar que, en algunos casos tanto el MAE como el MAPE obtenidos aplicando los modelos *Random Forest* retroalimentados es mayor. Esto parece un poco sorprendente, pues estos modelos deberían proporcionar un mejor rendimiento ya que su conjunto de entrenamiento cada vez es mayor y por lo tanto, disponen de más información acerca de los distribuidores. Sin embargo, este hecho puede ser debido a lo siguiente:

  1. La información más reciente, es decir, los datos del conjunto de validación están altamente influenciados por el COVID-19, lo cuál podría provocar confusión en los modelos. Para poder mitigar este efecto deberíamos de realizar un estudio más detallado de los períodos de tiempo más afectados por el COVID-19.
  2. El alto grado de aleatoriedad presente en las diferentes etapas de los modelos que estamos utilizando. Este hecho puede provocar que al reentrenar muchas veces el modelo obtenido con los nuevos datos este empeore. Para resolver esta cuestión deberíamos analizar el número de pedidos que deberían transcurrir antes de volver a reentrenar nuestro modelo sin que este proceso conlleve un efecto negativo en la precisión de nuestras predicciones.

- Teniendo en cuenta las Tablas 5.2, 5.6, 5.10 y 5.14 podemos ver que en todos los productos estudiados para los diferentes distribuidores de Tipo B de Estrella Galicia, excepto si consideramos los distribuidores de la Zona B y el *Producto 3*, los modelos *Random Forest* son los que nos proporcionan una mayor proporción de aciertos sobre los pedidos nulos.

- Es importante destacar que en la Zona A tenemos una mayor cantidad de pedidos nulos en el histórico de nuestros distribuidores y por lo tanto, nuestros modelos obtienen una proporción

de acierto más elevada en estos pedidos. Sin embargo, como en la Zona B tenemos una menor cantidad de pedidos nulos esta proporción de acierto baja considerablemente.

- A la vista de las predicciones obtenidas en las Tablas 5.4, 5.8, 5.12, 5.16, al valor del MAE medio obtenido en las Tablas 5.1, 5.5, 5.9 y 5.13 y a las Figuras 5.1, 5.3, 5.5 y 5.7 parece que nuestro modelo interpreta mejor los patrones de los pedidos en los distribuidores de la Zona A, sin embargo, esto podría ser debido a la gran cantidad de pedidos nulos y a la gran proporción de acierto en estos pedidos.
- En las Figuras 5.2, 5.4, 5.6 y 5.8 se puede ver que las variables más importantes para los modelos *Random Forest* varían dependiendo del distribuidor y del producto que estemos analizando. Este hecho dificulta encontrar un modelo general que se ajuste correctamente a todos los distribuidores.
- Teniendo en cuenta las Figuras 5.1, 5.3, 5.5 y 5.7 se puede ver que, en todos los casos analizados, nuestro modelo tiende a predecir por debajo del valor real de los pedidos, esto puede estar motivado por la ausencia de alguna covariable determinante para la predicción de nuestro modelo.

5.3. Posibles mejoras en la precisión de nuestros modelos.

En esta sección vamos a mostrar algunas ideas que consideramos útiles para mejorar los resultados obtenidos en futuras líneas de trabajo. Además, se muestran algunos enfoques que hemos llevado a cabo y que se podrían modificar para intentar optimizar los resultados obtenidos para los diferentes distribuidores de Tipo B de Estrella Galicia.

- Puede ser importante realizar un estudio más detallado de la temperatura. En nuestro caso hemos utilizado la temperatura media de cada día por provincia. Para ello, hemos calculado el promedio de temperatura de las diferentes estaciones meteorológicas situadas en cada provincia. Sin embargo, podría parecer útil emplear la temperatura de la zona de ocupación de cada distribuidor. Otra opción, que se podría valorar, es utilizar la temperatura máxima en lugar de la temperatura media.
- Habría que realizar un estudio mucho más detallado de los eventos y ver como estos le podrían afectar a los pedidos de los diferentes distribuidores. Esta variable puede llegar a ser bastante útil para explicar las grandes diferencias que se llegan a producir entre las predicciones y el valor real de algunos pedidos. Para ello, tendríamos que realizar un análisis para estimar con cuanto tiempo de antelación se preparan los distribuidores para los diferentes eventos y estudiar cuales son los eventos más relevantes para ellos. Desafortunadamente, nosotros no fuimos capaces de lograr que esta variable fuese significativa en nuestros modelos de aprendizaje estadístico.
- Otra cuestión a tener en cuenta, para futuras líneas de trabajo, puede consistir en estudiar de que forma se podrían agrupar los diferentes distribuidores. Estas agrupaciones se podrían realizar en función de lo parecido que sean sus pedidos, de la variedad de su cartera o de su zona de ocupación. Para ello, en primer lugar, habría que pensar en construir una métrica que nos permita relacionar a los distribuidores en función del criterio que hayamos elegido para posteriormente poder agruparlos. En nuestra opinión, consideramos que la información de distribuidores

semejantes puede ser mucho más relevante que considerar, por ejemplo, el promedio de todos los distribuidores.

- La selección de las variables para nuestros modelos se hizo mediante prueba y error. Es decir, hemos utilizado los predictores con los que hemos obtenido los mejores resultados. Sin embargo, podríamos pensar en realizar la selección de las variables del modelo utilizando diferentes metodologías, como pueden ser las metodologías tipo *wrapper* o *embedded*, y de esta forma podríamos tener en cuenta un gran número de posibles variables predictoras. En Guyon y Elisseeff (2003) afirman que la utilización de estos métodos mejora el desempeño predictivo de los métodos de aprendizaje estadístico, aunque también indican que estas mejoras no siempre son significativas.

- La evaluación de los diferentes modelos se realizó utilizando un conjunto de entrenamiento formado por los primeros datos, en particular el 80 %, y un conjunto de validación formado por los últimos datos, en nuestro caso el 20 %. Sin embargo, esto no es exactamente el enfoque del conjunto de validación. Posiblemente, un mejor enfoque podría haber sido utilizar la validación cruzada formada por K grupos. En nuestro caso, la división aleatoria del conjunto de datos en un conjunto de entrenamiento y otro de validación carecía de sentido, pues se perdía la dependencia temporal de los pedidos.

- Se podría haber utilizado una versión más sofisticada del modelo base considerado. Por ejemplo, podríamos considerar la media de los pedidos de cada distribuidor por estación y año.

- Probablemente, se pueda realizar un estudio más exhaustivo de las restricciones provocadas por el COVID-19 en cada una de las zonas de actuación de los diferentes distribuidores de Tipo B de Estrella Galicia. Consideramos que esta variable debería de jugar un papel relevante en nuestros modelos, pues la pandemia provocada por el COVID-19 abarca un gran espacio temporal en el estudio que hemos realizado durante este trabajo, en particular, sobre los conjuntos de validación que hemos considerado. Es por ello que optimizar esta variable nos puede ayudar a entender algunas desviaciones significativas en los pedidos de nuestros distribuidores, lo cuál nos ayudaría a reducir el error en esas predicciones.

- Otra posible línea de trabajo podría consistir en realizar un estudio de la correlación entre las diferentes variables predictoras utilizadas en los modelos de aprendizaje estadístico. Pues a priori, por ejemplo, parece claro que va existir correlación entre el `Mes`, la `Semana` y la `Estacion`.

- Finalmente, además de todas las cuestiones planteadas, podríamos pensar en estudiar si con algún otro procedimiento estadístico lograríamos mejorar los resultados obtenidos, especialmente, con la intención de obtener mejores resultados en los pedidos no nulos.

Bibliografía

[1] Aemet (2021). http://www.aemet.es/es/datos_abiertos/AEMET_OpenData, 15 de Septiembre de 2021.

[2] Agarap, A. F. (2018). Deep Learning using Rectified Linear Units (ReLU). arXiv. URL: https://arxiv.org/abs/1803.08375.

[3] Amari, S. (1993). Backpropagation and stochastic gradient descent method. *Neurocomputing*, 5(4):185–196. URL: https://www.sciencedirect.com/science/article/pii/092523129390006O.

[4] Clevert, D., Unterthiner, T., y Hochreiter, S. (2015). Fast and Accurate Deep Network Learning by Exponential Linear Units (ELUs). arXiv. URL: https://arxiv.org/abs/1511.07289.

[5] Dauphin, Y., de Vries, H., y Bengio, Y. (2015). Equilibrated adaptive learning rates for non-convex optimization. URL: https://arxiv.org/abs/1502.04390, doi = 10.48550/ARXIV.1502.04390.

[6] Davison, A. C. y Hinkley, D. V. (1997). *Bootstrap Methods and their Applications*. Cambridge Series in Statistical and Probabilistic Mathematics. Cambridge University Press.

[7] Dowle, M. y Srinivasan, A. (2021). *data.table: Extension of 'data.frame'*. R package version 1.14.2.

[8] Estrella Galicia (2022). https://estrellagalicia.es/compania/, 6 de Junio de 2022.

[9] Fritsch, S., Guenther, F., y Wright, M. N. (2019). *neuralnet: Training of Neural Networks*. R package version 1.44.2.

[10] Grant Slatton, T. (2014). *A comparison of dropout and weight decay for regularizing deep neural networks*. Thesis, University of Arkansas.

[11] Grolemund, G. y Wickham, H. (2011). Dates and times made easy with lubridate. *Journal of Statistical Software*, 40(3):1–25.

[12] Guyon, I. y Elisseeff, A. (2003). An Introduction to Variable and Feature Selection. *Journal of Machine Learning Research, 3*, 1157-1182. URL: https://www.jmlr.org/papers/volume3/guyon03a/guyon03a.pdf.

[13] Hahnloser, R. H. R., Sarpeshkar, R., Mahowald, M. A., Douglas, R. J., y Seung, H. S. (2000). Digital selection and analogue amplification coexist in a cortex-inspired silicon circuit. *Nature*, 405: 947–951. URL: https://doi.org/10.1038/35016072.

[14] Hastie, T., Tibshirani, R., y Friedman, J. (2001). *The Elements of Statistical Learning*. Springer texts in Statistics. Springer.

[15] James, G., Witten, D., Hastie, T., y Tibshirani, R. (2014). *An Introduction to Statistical Learning*. Springer texts in Statistics. Springer.

[16] Kingma, D. P. y Ba, J. (2014). Adam: A method for stochastic optimization. URL: https://arxiv.org/abs/1412.6980.

[17] Kuhn, M. y Vaughan, D. (2022). *parsnip: A Common API to Modeling and Analysis Functions*. R package version 0.2.1.

[18] Liaw, A. y Wiener, M. (2002). Classification and regression by randomforest. *R News*, 2(3):18–22. R package version 4.7-1.1.

[19] McDermott, G. (2022). *parttree: Visualise simple decision tree partitions*. R package version 0.0.1.9000.

[20] Ooms, J. (2014). The jsonlite package: A practical and consistent mapping between json data and r objects. *arXiv:1403.2805 [stat.CO]*.

[21] Ripley, B. y Lapsley, M. (2021). *RODBC: ODBC Database Access*. R package version 1.3-19.

[22] Shen, W. y Liu, R. (2018). Tackling Early Sparse Gradients in Softmax Activation Using Leaky Squared Euclidean Distance. *CoRR*, abs/1811.10779. URL: http://arxiv.org/abs/1811.10779.

[23] Shmueli, G. (2016). *Practical Time Series Forecasting: A Hands-On Guide [3rd Edition]*. Practical Analytics. Axelrod Schnall.

[24] Venables, W. N. y Ripley, B. D. (2002a). *Modern Applied Statistics with S*. Springer, New York, fourth edición. ISBN 0-387-95457-0.

[25] Venables, W. N. y Ripley, B. D. (2002b). *Modern Applied Statistics with S*. Springer, New York, fourth edición. ISBN 0-387-95457-0, R package version 7.3-17.

[26] Víctor Manuel Valiñas Varela (2022). https://github.com/VictorValinas/TFM, 12 de Junio de 2022.

[27] Wickham, H., Averick, M., Bryan, J., Chang, W., McGowan, L. D., François, R., Grolemund, G., Hayes, A., Henry, L., Hester, J., Kuhn, M., Pedersen, T. L., Miller, E., Bache, S. M., Müller, K., Ooms, J., Robinson, D., Seidel, D. P., Spinu, V., Takahashi, K., Vaughan, D., Wilke, C., Woo, K., y Yutani, H. (2019). Welcome to the tidyverse. *Journal of Open Source Software*, 4(43):1686.

[28] Wickham, H., François, R., Henry, L., y Müller, K. (2022). *dplyr: A Grammar of Data Manipulation*. R package version 1.0.9.

[29] Williams, G. J. (2011). *Data Mining with Rattle and R: The art of excavating data for knowledge discovery*. Use R! Springer.

[30] Zeiler, M., Ranzato, M., Monga, R., Mao, M., Yang, K., Le, Q., Nguyen, P., Senior, A., Vanhoucke, V., Dean, J., y Hinton, G. (2013). On rectified linear units for speech processing. En *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*, pp. 3517–3521. URL: http://ieeexplore.ieee.org/document/6638312/.