Trabajo Fin de Máster

# Incorporación de Información de Redes Sociales en un Modelo de Propensión

María González Estévez

**Máster en Técnicas Estadísticas**

**Curso 2019-2020**

# Propuesta de Trabajo Fin de Máster

| |
|---|
| **Título en galego:** Incorporación de Información de Redes Sociais nun Modelo de Propensión |
| **Título en español:** Incorporación de Información de Redes Sociales en un Modelo de Propensión |
| **English title:** Incorporation of Social Networking Information into a Propensity Model |
| **Modalidad:** Modalidad B |
| **Autor:** María González Estévez, Universidad de A Coruña |
| **Director:** Rubén Fernández Casal, Universidad de A Coruña |
| **Tutor:** Juan Manuel Mazaira Gómez, ABANCA |
| **Breve resumen del trabajo:**<br>El principal objetivo de este Trabajo de Fin de Máster es mejorar la capacidad predictiva de un modelo de propensión de ahorro e inversión existente en el área de Inteligencia de Clientes de ABANCA introduciendo información no convencional procedente de RRSS, que a su vez, permite obtener un mayor conocimiento del cliente. Para ello será necesario: (i) realizar técnicas de *Web Scraping* para la obtención de información externa (RRSS) que enriquezca el modelo, (ii) explotación de bases de datos estructurados o bases de datos desestructurados mediante técnicas de *Text Mining*, (iii) validación y depuración de datos, diseño, entrenamiento y validación del modelo por medio de algoritmos *Machine Learning* para predecir resultados de acciones comerciales en el ámbito de trabajo del área, concretamente, en ahorro e inversión. |

Don Rubén Fernández Casal, Profesor Titular de la Universidad de A Coruña, don Juan Manuel Mazaira Gómez, Coordinador de ABANCA, informan que el Trabajo Fin de Máster titulado

**Incorporación de Información de Redes Sociales en un Modelo de Propensión**

fue realizado bajo su dirección por doña María González Estévez para el Máster en Técnicas Estadísticas. Estimando que el trabajo está terminado, dan su conformidad para su presentación y defensa ante un tribunal.

En A Coruña, a 12 de Febrero de 2020.

El director:

FERNANDEZ CASAL RUBEN - DNI 33317724Q

Firmado digitalmente por FERNANDEZ CASAL RUBEN - DNI 33317724Q
Nombre de reconocimiento (DN): c=ES, o=UNIVERSIDAD DE A CORUÑA, ou=CERTIFICADO ELECTRONICO DE EMPLEADO PUBLICO, ou=MATEMATICAS, ou=10458, serialNumber=IDCES-33317724Q, sn=FERNANDEZ CASAL, givenName=RUBEN, cn=FERNANDEZ CASAL RUBEN - DNI 33317724Q
Fecha: 2020.02.12 11:59:03 +01'00'

Don Rubén Fernández Casal

El tutor:

MAZAIRA GOMEZ JUAN MANUEL - 34997729D

Firmado digitalmente por MAZAIRA GOMEZ JUAN MANUEL - 34997729D
Fecha: 2020.02.12 14:06:59 +01'00'

Don Juan Manuel Mazaira Gómez

La autora:

María González Estévez

Firmado digitalmente por María González Estévez
Fecha: 2020.02.12 14:14:52 +01'00'

Doña María González Estévez

# Agradecimientos

A través de estas líneas quiero expresar mi más sincero agradecimiento, ya que este Trabajo de Fin de Máster no sería posible sin las recomendaciones y pautas sugeridas por Rubén Fernández Casal por parte de la Universidad de A Coruña, como por los consejos, continua dedicación y asesoramiento de los compañeros de ABANCA. Con especial interés al departamento de Modelos Predictivos y Prospección, a mi tutor Juan Manuel Mazaira Gómez, por la acertada orientación, soporte y discusión crítica que me permitió un buen aprovechamiento en el trabajo realizado, como al resto del equipo: José Piñeiro Abal, Belén Sánchez Robredo, María Oliveira Pérez y Marta García Astray, por su apoyo y confianza.

# Índice general

# Resumen


## Resumen en español

Desde el Departamento de modelos predictivos del área de Inteligencia de Clientes de la entidad ABANCA *Corporación Bancaria S.A.*, se realizan múltiples aplicaciones orientadas al desarrollo de modelos de propensión que abarcan gestiones comerciales de productos, como pueden ser: la propensión a la contratación, al abandono y en particular, a la contratación de productos de inversión.

Dentro del Departamento subyace el interés por entender los distintos patrones, significación y actuación de las variables de cada modelo. En el presente trabajo, se intentará cuantificar la mejora del modelo de propensión de ahorro e inversión al introducir una variable extraída de RRSS que permita un mejor conocimiento del cliente. De esta manera, el objetivo es añadir información no convencional, que la entidad no poseía, con el fin de enriquecer y mejorar la capacidad predictiva de un modelo de propensión.

Para ello, se utilizarán técnicas de *Web Scraping* y de *Text Mining*, así como diferentes algoritmos *Machine Learning* que permitan obtener un reflejo fiel del comportamiento de la nueva variable en los resultados de las acciones comerciales del modelo.

## English abstract

From the Predictive Models Department of the IC area of ABANCA *Corporación Bancaria S.A.*, some multiple applications are performed, oriented to the development of propensity models that cover commercial management of products, such as the propensity to contract, to abandon and especcially, to contract investment products.

Underlying the Department is an interest in understanding the different patterns, significance and performance of the variables in each model. In the current work, will try to show the improvement of the savings and investment propensity model by introducing a variable extracted from RRSS that allows a better understanding of the client. In this way, the objective is to add non-conventional information that the entity did not possess, with the purpose of enrich and improve the predictive capacity of a propensity model.

In order to do so, *Web Scraping* and *Text Mining* techniques will be used, as well different *Machine Learning* algorithms that allow us to obtain a faithful reflection of the behavior of the new variable in the results of the commercial actions of the model.

# Parte I

# Metodología

# Capítulo 1

# Motivación e Introducción al problema

## 1.1. Motivación

El área de Inteligencia de Clientes de la entidad ABANCA *Corporación Bancaria S.A.* es la principal encargada de la analítica de datos de toda la organización. Esta unidad es fundamental en la evolución estratégica de ABANCA, sobre todo para la función comercial. Se realizan, entre una gran variedad de tareas, modelos analíticos que cubren el XX % de los productos de carteras y otros modelos orientados a una visión más general del cliente, como son modelos predictivos y de propensión. Además, se realiza una labor de investigación comercial que proporciona un mayor conocimiento de clientes que, junto a tareas propias de analítica avanzada, permiten mejorar la comercialización de los productos. Este área está integrada por cuatro departamentos: Modelos Predictivos y Prospección, Analítica Avanzada, Investigación Comercial y Gestión de las Relaciones con Clientes (CRM por sus siglas en inglés).

En los últimos tres años, el departamento de Modelos Predictivos y Prospección, se ha especializado en el proceso de construcción de modelos y herramientas analíticas que abarcan gran parte de las gestiones comerciales de productos: propensión a la contratación, abandono, valor de cliente, ahorro e inversión, segmentaciones, etc.

En concreto, buscan la implicación de áreas de negocios para el desarrollo de sus modelos predictivos, y utilizan entre otras, las herramientas analíticas de IBM SPSS Modeler, SAS, Teradata, Aster, R o Python con algoritmos de aprendizaje estadístico.

La creación de cada uno de los modelos se realizará con técnicas de regresión o árboles de decisión con el fin de seleccionar el algoritmo que mejor funcione. Entre los modelos con los que trabajan, se encuentran:

- **Modelos familias**.
  En este caso, los modelos implementados están orientados a clientes personas físicas que no tengan contratados los productos financieros objetivo de cada modelo, o bien, en los que se pretenda incrementar posiciones en el mismo.
  - **Ahorro e inversión**.
    El modelo aprende del comportamiento pasado de los clientes e infiere en el comportamiento de los nuevos, estimando la probabilidad de éstos para contratar productos de inversión. De esta manera, se aporta a la entidad un doble beneficio, ya que un mejor conocimiento de las necesidades de sus clientes, le permite anticiparse a su competencia y ofrecer un catálogo de productos de acorde a la situación financiera presente y futura del cliente, favoreciendo a ambas partes.
  - **Consumo y tarjetas**.
    El modelo estima la probabilidad de contratar préstamos al consumo o tarjetas respectivamente, aportando a la entidad ventajas similares a los modelos de ahorro e inversión anteriores.

  - **Seguros**.
    Los modelos para este tipo de productos estiman la probabilidad de contratación de seguros en cada uno de los siguientes rangos: seguros de hogar, seguros de salud, seguros de auto, decesos y vida riesgo.
  - **Planes de pensiones**.
    El modelo estima la probabilidad de contratación/aportaciones a planes de pensiones (altas, reactivados, aportaciones extraordinarias, etc.).
  - **Abandono**.
    El modelo estima la probabilidad que los clientes presentarán hacia el abandono de la entidad.

- **Modelos empresas**.
  Estos modelos están orientados a personas jurídicas y permiten detectar clientes con necesidades de financiación que no se estén financiando actualmente en la entidad bancaria. El objetivo principal de este modelo es la captación de nuevos clientes, es decir, identificar aquéllos que tengan mayor atractivo comercial (mayor rentabilidad, mayor vinculación y mayor solvencia), estimando las necesidades de negocio de la empresa.

- **Aldia Empresas**.
  Modelo que tiene el objetivo de detectar oportunidades comerciales en tiempo real a través de rastreo de medios digitales. Por medio de técnicas *Web Scraping* se recoge información online y se transforma en eventos comerciales. Los eventos detectados se distribuyen a gestores comerciales para realizar acciones comerciales proactivas a clientes potenciales.
  De esta manera se consigue aumentar la eficiencia comercial, se adquiere un mayor conocimiento del cliente, se obtiene una anticipación a la competencia, se detectan hechos que mejoren la perspectiva de solvencia y se analizan las tendencias sectoriales.

- **Redes empresas**.
  Este modelo tiene como objetivo construir una red de relaciones (compra-venta) entre empresas, que permitan identificar oportunidades comerciales (captación o financiación especializada).

- **Clientes valor**.
  Se realizan procesos comerciales con el fin de construir una matriz de afinidad de clientes. Entre estos procesos, se consigue diferenciar grupos de clientes, calcular la penetración de cada producto, definir la afinidad con cada uno de ellos; y de esta forma, alimentar el planificador de la Agenda Comercial y huecos de negocio de la entidad.

Además de todo lo anterior, en el departamento de Modelos Predictivos de Inteligencia de Clientes de ABANCA se realizan técnicas *Text Mining* y *Web Scraping*. Técnicas avanzadas de clasificación de textos para identificar variables, clasificar y extraer información que aporte valor en la toma de decisiones de la entidad.

Dentro del Departamento de Inteligencia de Clientes de la entidad subyace el interés por entender los distintos patrones y comportamiento de las variables que influyen en los clientes para realizar determinadas acciones comerciales. Actualmente, la generación de datos en cada una de las tareas cotidianas por los clientes alcanza un volumen alto, y muchas de ellas se rebelan en redes sociales. Así mismo, sería interesante ser capaces de perfilar las relaciones relevantes entre esta profusión de datos, y tratar de forma correcta, el conocimiento implícito que aportan para ser capaces de mejorar la atención profunda a las necesidades específicas de cada cliente. Tras un periodo de investigación previo, analizando la variedad de información no convencional que se podía extraer de RRSS[1] para mejorar el conocimiento sobre los clientes de la entidad, Linkedin fue la plataforma de contenido elegida. Ya que proporciona los datos de una forma más concreta, útil y coherente, permitiendo relacionarlos con los datos personales de los clientes de la entidad. De esta forma, podremos cuantificar la capacidad predictiva que incorpora esta nueva información a alguno de los modelos de propensión.

En este trabajo, se intentará mejorar el modelo de familias de ahorro e inversión de los clientes procedentes del área metropolitana de A Coruña introduciendo una variable nueva extraída de Linkedin,

[1] A lo largo del trabajo, se utilizará la abreviatura de RRSS para referirse a las redes sociales.

la experiencia laboral de cada cliente, concretamente el empleo actual, que como veremos en los resultados de la Parte II, enriquece el modelo de propensión y mejora su precisión. Esta modelización y extracción aporta a la entidad bancaria una ventaja competitiva muy importante, ya que, dispondrá de una nueva variable predictora repleta de información hasta ahora desconocida. A partir de la misma, también podrían realizarse predicciones para anticipar el comportamiento de los clientes hacia seguros, planes de pensiones, consumo o tarjetas, e incluso para tareas realizadas en otros departamentos; informes de comercios, identificar clientes objetivo de acciones específicas o en modelos de recomendación y optimización en la comunicación, pudiendo dar lugar a una variedad de estudios posteriores para la entidad.

Para llevar a cabo esta modelización se abarcarán múltiples técnicas estadísticas; desde el uso de modelos de regresión o árboles de decisión, con el propósito de comparar y analizar sus resultados, a técnicas de *Text Mining* o de *Web Scraping*.

## 1.2. Variables del modelo

Como se ha comentado anteriormente, el objetivo principal del trabajo es obtener una mejora del modelo de ahorro e inversión de la entidad, introduciendo una nueva variable relacionada con el empleo de cada cliente procedente de una fuente de información hasta ahora no utilizada, que como veremos en la Parte II muestra relevancia en el modelo, lo que produce una mejora en la validez del análisis y genera predicciones más fiables. Este modelo de familias está basado en detectar propensiones a la contratación de productos de ahorro e inversión como pueden ser: planes de pensiones, valores, seguros de ahorro o fondos de inversión. Para ello, localiza de entre todos los clientes, aquéllos que tengan una necesidad de invertir en alguno de los productos nombrados, permitiendo a la empresa anticiparse a la competencia, asesorando al cliente en determinados productos y contribuyendo al aumento de rentabilidad.

Esta sección se centra en definir la variable objetivo del modelo y seleccionar las variables significativas para poder obtener resultados fiables en las predicciones. Además, también se determinará el conjunto de clientes para los que se realizará la predicción, así como el perímetro u horizonte temporal a analizar.

Destacar que la selección de las variables internas, se enfocó desde un primer momento en la búsqueda de aquellas variables importantes para el modelo, es decir, aquéllas que resulten de mayor utilidad para predecir la variable de interés. Este trabajo se realizó anteriormente por parte del departamento que tiene un conocimiento más extenso acerca de cuales son los productos financieros de la entidad, así como los más utilizados para este caso en particular.

### 1.2.1. Definición de la variable objetivo

La finalidad del modelo de ahorro e inversión es detectar características en los clientes de la entidad, que puedan identificar su propensión a la contratación de productos de ahorro e inversión y cuantificar la probabilidad de que se lleve a cabo.

Entendemos como productos de ahorro e inversión a cualquiera de estos productos:

- **Fondos de Inversión**.
  Institución de inversión colectiva, que consiste en reunir fondos de distintos inversores, naturales o jurídicos, para invertirlos en diferentes instrumentos financieros (su responsabilidad se delega al banco). Los fondos de inversión son una alternativa de inversión diversificada, ya que invierten en numerosos instrumentos reduciendo el riesgo en función del fondo de inversión que se elija.

- **Planes de Pensiones**.
  Producto financiero de ahorro orientado a la jubilación, en el cual el inversor efectúa aportaciones periódicas que le permiten disponer de un capital o una renta en el momento de jubilación, incapacidad o fallecimiento.

- **Depósitos de Valores**.
  Resultado de un contrato con el intermediario financiero por el que éste se compromete, bajo un número de cuenta determinado, a custodiar y administrar una cartera de valores, de renta variable o fija: acciones, obligaciones, etc.

- **Seguros de ahorro**.
  Producto comercializado únicamente por aseguradoras. El asegurado se compromete a abonar una prima y se establece por contrato una fecha en la que el dinero no se podrá rescatar.

El interés radica en conocer qué características de los clientes pueden llevar a realizar inversiones en algunos de estos productos.

La variable a predecir no diferenciará entre nuevas contrataciones y aportaciones a estos productos. La idea es predecir peculiaridades en los clientes que lleven a una inversión, bien sea mediante una nueva contratación, o bien mediante incrementos en sus posiciones en algunos de estos productos.

Si bien es cierto, clientes que posean perfil "ahorrador", su probabilidad de inversión será mayor que clientes que no posean estos productos contratados, aunque la diferencia es redundante. Así mismo, se estudiará también como variable objetivo la captación de saldos de otras entidades.

En resumen, la variable objetivo a la que llamaremos $IN\_EXITO$ es una variable indicadora que define los incrementos en las posiciones del cliente en productos de ahorro e inversión, $IN\_EXITO = 1$ si aumenta el número de participaciones o por lo contrario, $IN\_EXITO = 0$. Con los valores de esta variable objetivo observados durante el horizonte temporal escogido (anual, ver Subsección 1.2.4), el modelo aprende del comportamiento pasado de los clientes y estima el comportamiento de nuevos clientes (durante los 6 meses posteriores a su observación).

### 1.2.2. Elección de variables para la modelización

De entre todas las variables relativas a operaciones comerciales que nos ha facilitado la entidad para esta modelización, la extracción de datos va encaminada a aquellas que aportan información útil en el análisis que pretendemos llevar a cabo: fechas, ingresos, capacidad de ahorro, límites preconcedidos, si el cliente es autónomo o no, ámbito del cliente, gastos de tarjetas de débito, posesión de seguros, importe/-recibos básicos domiciliados o tarjeta/préstamos ajenos domiciliados.

A continuación se muestra la información disponible sobre cada variable, teniendo en cuenta que ésta solo es una pequeña parte de la información que posee la entidad. De este conjunto de variables, *Cliente_ID* y *FECHA* son imprescindibles para la creación del modelo, ya que permiten agrupar las transacciones de cada cliente durante el período de observación. También destacan, algunas variables categóricas que nos permiten agrupar clientes en subgrupos más pequeños, descritas a continuación:

- **CLIENTE_ID**.
  Tipo factor, id del cliente en estudio.

- **FECHA**.
  Tipo fecha, en la que se observa el cliente (las fechas de observación van desde el 31/05/2018 al 30/04/2019).

- **SEGMENTO_ID_DINAMICO**.
  Tipo factor, variable segmento del cliente que toma los valores: 1 (menores), 2 (promotores autónomos), 3 (autónomos y profesiones liberales), 4 (jóvenes), 5 (adultos comprendidos entre los 26 y 45 años de edad), 6 (adultos entre los 46 y 60 años de edad), 7 (senior), 8 (no asignado[2]) y 9 (autónomos y profesiones liberales pero potenciales).

- **ESTADO_ACTUAL_CLIENTE_ID**.
  Tipo factor, variable estado del cliente en la fecha de observación. Toma los valores: con actividad (CA), sin actividad (SA), perdido (PE) y no activado (NA).

- **SEXO**.
  Tipo factor, variable sexo.

[2]En el modelo los valores no asignados se corresponden a datos faltantes (NA's) que la empresa no posee acerca de un cliente en concreto.

- **EDAD**.
  Tipo factor, variable edad del cliente a la fecha de observación.

- **IMP_RECIBOS_BASICOS**.
  Tipo numérico, suma de las variables del número de los importes de recibos de agua, teléfono y electricidad. Es decir, la media de los importes de los recibos básicos en los últimos 4 meses.

- **VISTA**.
  Tipo numérico, variable saldo medio vista en los contratos en la fecha de observación.

- **PLAZO**.
  Tipo numérico, suma de los saldos medios a fecha de observación de imposiciones a plazo fijo (a corto y largo plazo) en los contratos como primer titular. Es decir, saldo medio a plazo del cliente.

- **NUM_SEG_SIN_OTROS**.
  Tipo factor, variable número de seguros de hogar, de coche y de vida en la fecha de observación.

- **SEGUROS_SIN_OTROS**.
  Tipo factor, variable indicadora que nos indica si el cliente tiene o no seguro de hogar, de coche y de vida en base a la variable $NUM\_SEG\_SIN\_OTROS$.

- **INGRESOS**.
  Tipo numérico, suma de ingresos siempre y cuando este dato sea anterior a la fecha de observación.

- **CAPACIDAD_DE_AHORRO**.
  Tipo numérico, capacidad de ahorro o estimación de la liquidez del cliente disponible después de hacer frente a todos sus gastos.

- **TRIAD_LIMITE_PP**.
  Tipo numérico, límite de cuota asumible para préstamo personal preconcedido del cliente.

- **GATOS_DEBITO**.
  Tipo numérico, suma de los gastos mensuales realizados con tarjetas de débito en el último año. Se obtiene la tenencia del servicio de débito y el gasto.

- **IN_TARJETA_AJENA**.
  Tipo factor, variable indicadora que muestra la tenencia de tarjetas ajenas o pagos a otras entidades financieras en función de adeudos del último año respecto a la fecha de observación.

- **AMBITO_CLIENTE**.
  Tipo factor, indicadora del ámbito del cliente en función del padrón municipal. Toma los valores: 1 (Rural), 2 (Semiurbano), 3 (Ciudad), 4 (Gran ciudad) y 5 (Área metropolitana).

### 1.2.3. Elección de la población de clientes

Es importante señalar, que los datos personales de los clientes han sido previamente anonimizados en base al cumplimiento de las normas de protección de datos y a la política de riesgo operativo de la entidad. En un primer momento, se intentó trabajar con la operativa total de los clientes procedentes de las distintas comunidades que posee la entidad, pero debido a la cantidad de variables a extraer con las técnicas de *Web Scraping*, explicadas en la siguiente sección, y al tiempo que conllevaría, se decidió reducir los datos al área metropolitana de A Coruña y a los distintos ayuntamientos que la forman.

En resumen, los modelos se aplicaron sobre los datos de los clientes bancarizados residentes en el área metropolitana de A Coruña, centrando nuestro estudio en los clientes gallegos de entre 22 y 55 años de edad[3] y considerando las transacciones realizadas tanto en comercios gallegos como en el resto de España.

[3]Se decide realizar el modelo sobre este conjunto de clientes, porque se supuso que al alcanzar alrededor de los 50 años de edad, apenas se disponen de cuentas de RRSS para extraer el contenido de la experiencia laboral. Para ver más información acerca de la elección ver Sección 2.2.

Concretamente, los diversos ayuntamientos en los que se ha efectuado el análisis del área metropolitana son: A Coruña, Culleredo, Arteixo, Oleiros, Cambre, Sada, Betanzos, Abegondo, Bergondo y Carral.

En la Tabla 1.1, se expone la información acerca del número de clientes así como el nombre de cada uno de los ayuntamientos, representando las dimensiones del problema al que nos enfrentamos.

| *AYUNTAMIENTO* | *AYUNTAMIENTO_ID* | *CLIENTES* |
|---|---|---|
| A Coruña | 030 | |
| Culleredo | 031 | |
| Oleiros | 058 | |
| Arteixo | 005 | |
| Cambre | 017 | |
| Sada | 075 | |
| Betanzos | 009 | |
| Bergondo | 008 | |
| Carral | 021 | |
| Abegondo | 001 | |
| TOTAL | 10 ayuntamientos | 64.066 |

Tabla 1.1: Tabla de información del conjunto de datos disponibles eliminada por confidencialidad.

### 1.2.4. Elección del intervalo temporal

Tras haberlo analizado en profundidad, se ha decidido trabajar con la operativa total de datos de un período anual, desde Mayo de 2018 hasta Abril de 2019, observando y validando los resultados de la variable objetivo los 6 meses posteriores a su observación, como se describe en la Tabla 1.2, es decir, hasta Octubre de 2019. La idea principal, es la que se muestra en la Figura 1.1.

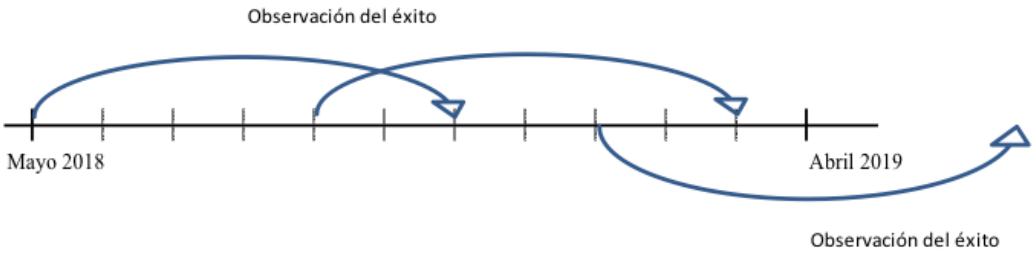



Figura 1.1: Ejemplo gráfico del perímetro temporal del modelo.

En pocas palabras, la variable objetivo se estima durante los posteriores 6 meses, considerando un éxito si se producen incrementos en posiciones de productos de inversión a lo largo de los 6 meses posteriores a cada momento de la observación.

De esta manera, un mismo cliente será observado a lo sumo 12 veces. El hecho de realizar observaciones durante un año completo, aporta ventajas con respecto a la observación de un único momento temporal, es decir, los resultados no estarían tan influenciados por una posible componente estacional. Por ejemplo, momentos puntuales dónde los saldos pueden fluctuar, véase Navidades o Semana Santa.

| Se observa cada una de las variables | ¿Ha invertido en los 6 meses siguientes? Si/No |
|---|---|
| 31/05/2018 | ¿Ha invertido en productos de la entidad entre el 1/06/2018 y 31/12/2018? Si/No |
| 30/06/2018 | ¿Ha invertido en productos de la entidad entre el 1/07/2018 y 31/01/2019? Si/No |
| ... | ... |
| 30/04/2019 | ¿Ha invertido en productos de la entidad entre el 1/05/2019 y 31/10/2019? Si/No |

Tabla 1.2: Tabla resumen procedimiento.

Expresado de forma matricial,

$$\begin{array}{c} \\ 31/05/2018 \\ 30/06/2018 \\ \cdots \\ 30/04/2019 \end{array} \begin{bmatrix} \text{Variable 1} & \text{Variable 2} & \cdots & \text{Variable 19} \\ X_{1,1} & X_{1,2} & \cdots & X_{1,19} \\ X_{2,1} & X_{2,2} & \cdots & X_{2,19} \\ \cdots & \cdots & \cdots & \cdots \\ X_{12,1} & X_{12,2} & \cdots & X_{12,19} \end{bmatrix} \rightarrow \begin{bmatrix} \text{¿Ha invertido a lo largo de los 6 meses siguientes?} \\ 1/0 \\ 1/0 \\ \cdots \\ 1/0 \end{bmatrix}$$

Finalmente, tras un estudio exhaustivo de cada una de las variables que entran en el modelo de propensión de ahorro e inversión (detrás de un tiempo dedicado para conocer el funcionamiento del mismo), la elección oportuna del intervalo temporal y la selección conveniente de la población de clientes, se procede al tratamiento de los datos.

En el Capítulo 2, se realizarán los pasos previos necesarios a la construcción del modelo. El Capítulo mencionado es de vital importancia para la resolución de este trabajo, ya que recoge las pautas necesarias para poder realizar la extracción de información de la RRSS, sin la cual no sería posible la mejora del modelo existente en la entidad.

# Capítulo 2

# Tratamiento de datos

Para satisfacer los objetivos de análisis y predicción de este trabajo, es necesario realizar una serie de pasos previos sobre las variables de estudio antes de la creación del modelo. Para comenzar, en la Sección 2.1 se desarrollará el procedimiento para la obtención de variables de la base de datos de la entidad. A continuación, se detallarán las medidas adoptadas para la creación de la nueva variable a introducir en el modelo. Primero en la Sección 2.2, se extraerá la información de redes sociales a través de técnicas de *Web Scraping*, para después, en la Sección 2.3 realizar la depuración y clasificación de la información desestructurada extraída con técnicas de *Text Mining*. Por último, una vez se obtiene la nueva variable del modelo, es necesario analizar detalladamente la relación entre las variables explicativas con respecto a la variable objetivo, y estudiar su comportamiento para comprender correctamente los resultados obtenidos en la Parte II del trabajo.

Cabe destacar que el objetivo de este trabajo se centra en enriquecer y mejorar el modelo de propensión de ahorro e inversión de la entidad, añadiendo la nueva variable correspondiente al empleo actual del cliente. Por ello, este Capítulo es muy relevante para su realización.

## 2.1. Extracción de las variables

Tal y como se ha comentado, se pretende construir un modelo que permita predecir la probabilidad de nuevos clientes de contratar productos de inversión a partir de los datos internos de ABANCA. Por tanto, será necesario construir métricas a partir de la información de negocio de la entidad accediendo a los sistemas informacionales que dispone, concretamente de un Data Warehouse (DWH) que se enriquece diariamente del informe que se genera en el operacional. Este sistema está diseñado como una base de datos relacional, ejecutando vistas o llamadas que cruzan diferentes tablas para facilitar el acceso y la compresión de la información almacenada. La extracción del contenido se realiza por medio de consultas del lenguaje SQL. A continuación, se muestran algunas de las principales instrucciones de este lenguaje.

- **SELECT**: Se utiliza para seleccionar registros de una base de datos. El resultado se guarda en una tabla de resultados, llamada *result-set*. Se puede seleccionar columnas concretas o todas las columnas (con el comando *).
- **SELECT DISTINCT**: Devuelve registros donde los valores de todas las columnas elegidas sean diferentes. Se utiliza cuando en una tabla, una columna contiene valores duplicados, y se quiere mostrar sólo los valores únicos.
- **FROM**: Cláusula en la que debemos indicar cuál es la tabla o subconsulta donde se encuentran las columnas declaradas en el SELECT.
- **WHERE**: Se usa para filtrar valores. Se extraen sólo los datos que cumplen unos determinados criterios. Los operadores lógicos de *WHERE*, pueden ser $=, <>, >, <, >=, <=$, *BETWEEN, LIKE, o IN*. Si los valores son de tipo *character*, se necesitan comillas simples, por el contrario, si son numéricos, no.

- **BETWEEN**: Filtra los valores de una columna (variable) de nuestra consulta entre el rango de los valores especificados. Los valores pueden ser números, texto o fechas.
- **LIKE**: Se usa en la cláusula *WHERE* para buscar un patrón específico en una columna.
- **IN**: Permite especificar múltiples valores en una cláusula *WHERE*.
- **AS**: Las alias se usan para renombrar temporalmente una tabla o columna, de esta forma son más legibles, especialmente para consultas complejas. Son útiles cuando: hay más de una tabla involucrada, se emplean funciones, los nombres de las columnas son largos o poco legibles, o se combinan dos o más columnas.
- **AND y OR**: Se usan para filtrar resultados basándose en más de una condición. El comando *AND* muestra los resultados si se cumplen la primera y la segunda condición, y *OR*, muestra los resultados si se cumple una de las condiciones. Se puede combinar tantas veces como se necesite para evitar errores y facilitar lectura. También, se suelen agrupar las sentencias reunidas por estos comandos utilizando paréntesis.
- **ORDER BY**: Se usa para ordenar los resultados por una o más columnas. Por defecto, se ordenan en orden ascendente *ASC*, para ordenar los resultados en orden descendente se usa la palabra *DESC*.
- **INSERT INTO**: Esta sentencia se utiliza para insertar nuevos valores en la tabla. Se pueden insertar de dos formas: sin especificar el nombre de las columnas donde se insertarán los datos, o especificando tanto las columnas como los valores a insertar.
- **UPDATE**: Se usa para actualizar datos en una tabla, se utiliza la cláusula *SET*, para indicar qué valor se quiere modificar. La cláusula *WHERE* es opcional, pero si no se indica, se actualizarán todos los valores de la tabla.
- **DELETE**: Se utiliza para eliminar filas de una tabla. La cláusula *WHERE* es opcional, pero si no se indica, se eliminarán todos los valores de la tabla. Si se quiere borrar todos los datos, se puede utilizar también *TRUNCATE*, ligeramente más rápido. Índices, tablas y bases de datos también se pueden eliminar con la sentencia *DROP*.
- **CREAT DATABASE**: Se usa para crear una base de datos. Sin embargo, para crear tablas en una base de datos se utiliza la sentencia *CREATE TABLE*.
- **GROUP BY**: Agrupa las variables seleccionados por los "niveles" de la variable o variables de agrupación escogidas.
- **HAVING**: Sentencia que contiene funciones de agregado y especifica las condiciones que se aplican a los campos que se resumen en la llamada *SELECT*.

Hay varias funciones que pueden usarse para manejar varios registros, a continuación, se muestran algunas de las funciones utilizadas con sentencias de agrupación de datos como *GROUP BY*:

- **COUNT**: Se utiliza para contar el número de filas de una o varias columnas.
- **AVG**: Se utiliza para calcular el promedio de un conjunto de valores.
- **MAX /MIN**: Para calcular el valor más alto, o el más pequeño.
- **SUM**: Calcula la suma de varias líneas.

Si queremos consultar datos de una tabla en función a ciertas columnas de otra o más tablas, es decir, asociar varias tablas con un campo común entre ellas en la misma consulta, tenemos que usar consultas conocidas como "multi-tabla". Estas consultas "multi-tabla" se realizan bajo los distintos tipos de *JOIN* que veremos a continuación:

- **INNER JOIN**: Devuelve aquellos registros/filas de las dos o más columnas, siempre y cuando exista una coincidencia entre las columnas en ambas tablas. Es el tipo de *JOIN* más común. Su sintaxis se correspondería con:

```
SELECT nombreColumna(s)
FROM table1
INNER JOIN table2
ON table1.nombreColumna=table2.nombreColumna;
```

- **LEFT JOIN**: Mantiene todas las filas de la tabla de la izquierda (table1), mostrándose únicamente los registros/filas de la tabla de la derecha que sean coincidentes. Si existen valores en la tabla de la izquierda pero no en la tabla de la derecha, se mostrarán como nulos. Su sintaxis se correspondería con:

```
SELECT nombreColumna(s)
FROM table1
LEFT JOIN table2
ON table1.nombreColumna=table2.nombreColumna;
```

- **RIGHT JOIN**: Lo contrario de *LEFT JOIN*, ahora se mantienen todas las filas de la tabla de la derecha (table2). Las filas de la tabla de la izquierda se presentarán si hay una coincidencia con las de la derecha. Si existen valores en la tabla de la derecha pero no en la tabla izquierda, se mostrarán como nulos. Su sintaxis se correspondería con:

```
SELECT nombreColumna(s)
FROM table1
RIGHT JOIN table2
ON table1.nombreColumna=table2.nombreColumna;
```

- **OUTER JOIN**: Devuelve todas las filas de la tabla de la izquierda (table1) y de la tabla de la derecha (table2). Combina el resultado de los *JOINS LEFT* y *RIGHT*. Aparecerán nulos en cada una de las tablas alternativamente cuando no exista coincidencia. Se puede usar en su lugar *FULL OUTER JOIN*, obteniéndose el mismo resultado. Su sintaxis se correspondería con:

```
SELECT nombreColumna(s)
FROM table1
OUTER JOIN table2
ON table1.nombreColumna=table2.nombreColumna;
```

Una forma útil y didáctica de ver el funcionamiento de los comandos anteriores es mediante el diagrama de Venn, como se muestra en la Figura 2.1.

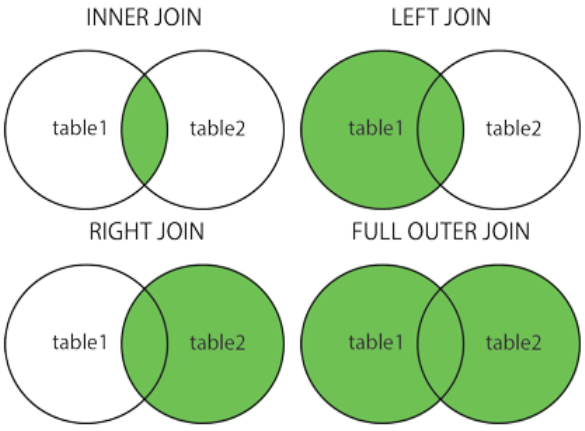



Figura 2.1: Diagrama de Venn para los comandos *JOIN*.

A continuación, se mostrarán algunos ejemplos de cómo utilizar algunas de las sentencias explicadas.

Imagínese, una empresa que contiene un sistema de bases de datos relacionales para analizar información generada por sus recibos trimensuales de electricidad, y se tienen las siguientes tablas:

- Tabla de clientes (*CLIENTES*), en la que se recogen los siguientes datos de clientes: [1]
  - **ClienteID**: *CLI_ID*
  - Nombre del Cliente: *NOM*
  - Apellidos del Cliente: *APE*
  - Teléfono de contacto: *TLF*
  - Provincia: *PROV*
  - Ayuntamiento: *AYUN*

| *CLI_ID* | *NOM* | *APE* | *TLF* | *PROV* | *AYUN* |
|---|---|---|---|---|---|
| 1 | María | Fernández Vázquez | 647876543 | A Coruña | Culleredo |
| 2 | Pablo | Rodríguez González | 654378923 | A Coruña | Oleiros |
| 3 | Isabel | Estévez López | 654763241 | A Coruña | Carral |
| 4 | José | Lago Ramos | 674563287 | A Coruña | Arteixo |
| 5 | Crsitina | Álvarez Alonso | 687564567 | A Coruña | A Coruña |

Tabla 2.1: Tabla Clientes.

- Tabla de pedidos (*RECIBOS*), en ella se recoge la información correspondiente a los pedidos efectuados:
  - **ReciboID**: *REC_ID*
  - ClienteID: *CLI_ID*
  - Factura: *FACT*
  - Importe: *IMP*
  - DIA: *DIA*

| *REC_ID* | *CLI_ID* | *FACT* | *IMP* | *DIA* |
|---|---|---|---|---|
| 234 | 5 | 30 | 140 | 17/05/2019 |
| 456 | 4 | 31 | 360 | 17/04/2019 |
| 236 | 3 | 90 | 1000 | 23/05/2019 |
| 478 | 1 | 40 | 800 | 16/05/2019 |
| 293 | 2 | 23 | 300 | 15/04/2019 |

Tabla 2.2: Tabla Pedidos.

[1] Destacar que el comando clave de la tabla se corresponde con el campo en negrita.

Si queremos saber cuantas facturas tienen un importe mayor o igual que 350€y en qué día se han realizado, la sentencia en SQL debería ser la siguiente:

**SELECT** $FACT$, $IMP$, $DIA$
**FROM** $RECIBO$
**WHERE** $IMP \geq 350$;

Si queremos saber a qué ayuntamiento del área metropolitana de A Coruña corresponden estos importes, la consulta debería ser:

**SELECT** $CLIENTES.AYUN$, $RECIBOS.FACT$, $RECIBOS.IMP$,$RECIBOS.IMP$
**FROM** $CLIENTES$
**INNER JOIN** $CLIENTES$
**ON** $CLIENTES.CLI\_ID = PEDIDOS.CLI\_ID$
**WHERE** $IMP \geq 350$;

Cuando se ejecuta la sentencia se muestra en pantalla la tabla de resultados, la correspondiente a la primera sentencia sería la Tabla 2.3.

| $FACT$ | $IMP$ | $DIA$ |
|---|---|---|
| 31 | 360 | 17/04/2019 |
| 90 | 1000 | 23/05/2019 |
| 40 | 800 | 16/05/2019 |

Tabla 2.3: Consulta Tabla Recibos.

Siguiendo este procedimiento, se realizaron las consultas con SQL a través de la plataforma de *Teradata*, seleccionando, fusionando y agrupando los datos que se consideraron significativos para el modelo. Como resultado, se elaboraron las variables internas analizadas en la Sección 2.4, utilizadas como las variables explicativas en el modelo. Para poder trabajar desde el software estadístico `R`, se realiza la llamada a SQL utilizando la librería `RODBC` para importar la tabla de datos correspondiente.

Para saber más acerca de SQL, se recomienda ver Alan Beaulieu (2005).

## 2.2. Web Scraping

Para la extracción de las variable categórica procedente de redes sociales se ha recurrido a la técnica conocida como *Web Scraping*, una herramienta muy común en el análisis de datos cuando se quiere incorporar a un modelo variables procedentes de datos no estructurados disponibles en la web. En nuestro caso, utilizaremos dicha técnica para recuperar información relativa al puesto de trabajo y formación de los clientes de la entidad presentes en la plataforma profesional de Linkedin.

Para esta tarea nos apoyaremos principalmente en el paquete `RSelenium`, ya que nos permite automatizar el comportamiento de un navegador web (en nuestro caso *Google Chrome*) mediante el lenguaje de programación `R`. Este paquete se sostiene en un proyecto con el mismo nombre, *Selenium*, que aglutina un conjunto de herramientas y bibliotecas que permiten, como ya se ha mencionado, la automatización de los navegadores web. El núcleo de Selenium es WebDriver, una interfaz que permite escribir conjuntos de instrucciones que se pueden ejecutar indistintamente en muchos navegadores, permitiéndonos controlar de forma remota las instancias del navegador y emular la interacción del usuario con el mismo. De este modo, mediante `RSelenium` podremos simular actividades comunes realizadas por los usuarios finales como pueden ser: ingresar texto en los campos, seleccionar valores desplegables en casillas de verificación, y hacer clic en los enlaces de los documentos. También proporciona muchos otros controles, como el movimiento del mouse, la ejecución arbitraria de JavaScript y mucho más.

En nuestra situación, utilizaremos `RSelenium` para automatizar, mediante un algoritmo, patrones de búsqueda de los clientes de ABANCA en la red social de Linkedin, concretamente mediante el nombre y los apellidos de los clientes que han cedido su voluntad para ello en base al reglamento de protección de datos de la entidad. De esta forma, se extraerá información relativa a sus características profesionales, como pueden ser: localidad, puesto, empleo actual, empresa, la fecha de comienzo del empleo o la duración, ubicación del empleo, educación, titulación, licencias y certificaciones, aptitudes y validaciones, correo electrónico, teléfono, fecha de nacimiento, etc.

No todas estas variables se incluirán en el modelo de propensión, si no que nos servirán de ayuda a la hora de comprobar si las personas encontradas mediante esta técnica son realmente los clientes de la entidad de los cuales queremos incluir esta información. No parece raro que a la hora de buscar en una red social por *"Manuel López Gonzalez"* las coincidencias no sean únicas, ya que se trata de un nombre común. Es por ello que debemos recuperar algunas variables que nos permitan a posteriori validar si los datos se corresponden con el cliente buscado, por ejemplo será de especial interés el correo electrónico o la fecha de nacimiento. A continuación, se expondrá un ejemplo que recoge en cierta medida la complejidad del algoritmo necesario para automatizar el comportamiento de los clientes de la entidad en Linkedin.

Primero, realizamos una llamada a SQL para extraer los nombres de cada cliente de ABANCA.

```
# Cargamos nombres de clientes desde llamada al SQL -----------------------

db <- odbcConnect("Teradata") #Abrir conexión con TERADATA
query<-"SELECT * FROM   mktg_usr_scraping_total WHERE (provincia_id='15'
AND ayuntamiento_id='030' AND consultado=0)"
output <- sqlQuery(db, query)
odbcClose(db) # Cerrar la conexión con TERADATA
head(output)

# Realizamos llamada a las columnas de nombre y apellidos y las unimos en una
output=within(output,nombre_completo<-paste(output$NOMBRE,output$PRIMER_APELLIDO,
output$SEGUNDO_APELLIDO,sep=""))
output$nombre_completo=str_to_lower(output$nombre_completo,locale="es")
head(output$nombre_completo)
str(output$nombre_completo)

# Selecionamos cada primera letra de nombre y apellido en mayúscula
# Creamos nuestro vector de nombres de todos los clientes de ABANCA
# con permiso para uso de datos personales
output$nombre_completo=unlist(lapply(output$nombre_completo, FUN=toTitleCase))
Cabanca <- output$nombre_completo
```

Después, cargaremos la librería de `RSelenium` e indicaremos la información del Driver de nuestro navegador y con el cual nos vamos a conectar, para ello los comandos serían los siguientes:

```
library("RSelenium")

# Cargando el drive y conexión con Internet
remDr <- remoteDriver(remoteServerAddr = "localhost",
port = 4444L, browserName = "chrome")
remDr$open(silent = TRUE)
```

Una vez cargada la librería y conectados al controlador, procedemos a acceder a la página web de la siguiente forma:

```
remDr$navigate("https://www.linkedin.com/feed/?
trk=guest_homepage-basic_sign-in-submit")
```

Lo que veríamos en nuestro navegador sería lo siguiente:

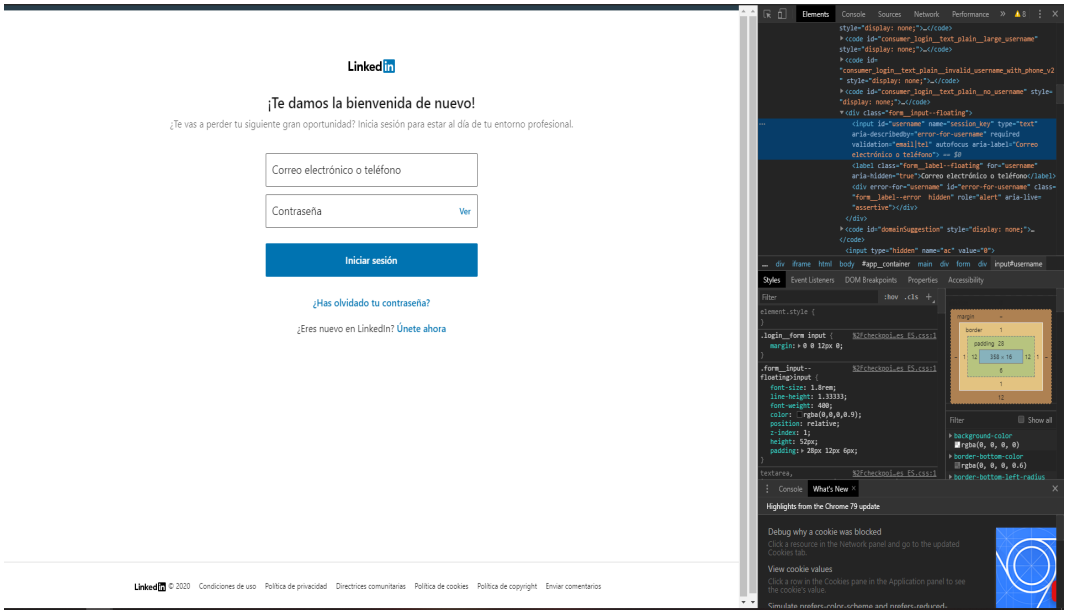



Figura 2.2: Inicio de sesión Linkedin.

Ahora apoyándonos en las opciones para desarrolladores de *Google Chrome*, que podemos ver en el margen derecho de la Figura 2.2, deberemos encontrar el elemento correspondiente al correo electrónico y contraseña.

Una vez identificados los elementos del código fuente de la página web, en `R` se ejecutaría lo siguiente:

```
# Correo electrónico

webElement=remDr$findElement(using = "xpath", "//input[contains(@id,'username')]")
webElement$clickElement()
webElement$sendKeysToElement(list("correoelectronico"))

# Contraseña

webElement=remDr$findElement(using = "xpath", "//input[contains(@id,'password')]")
webElement$clickElement()
webElement$sendKeysToElement(list("contraseña"))
```

Al iniciar la sesión, el siguiente paso sería entrar en la cuenta de usuario de Linkedin, tal y como se se muestra en la Figura 2.3 para un ejemplo concreto.

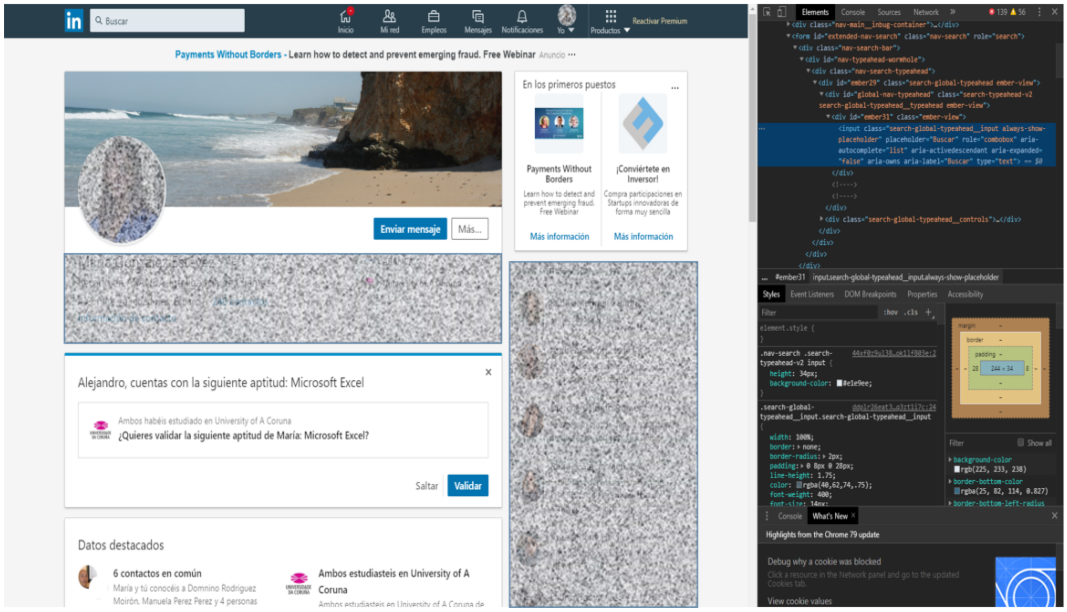



Figura 2.3: Inicio de sesión Linkedin.

En la parte superior izquierda de la imagen, deberemos identificar el elemento correspondiente a la barra de búsqueda e introducir ahí el nombre y apellidos de los clientes de ABANCA que son objeto de estudio. A continuación, se procedería a extraer su información, para ello en el navegador deberemos automatizar el siguiente comportamiento:

1. Acceder a la barra de búsqueda de Linkedin.

```
webElement=remDr$findElement(using = "xpath", "//input[contains(@class,'search-
    global-typeahead__input')]")
webElement$clickElement()
```

2. Insertar el nombre y apellidos de los clientes de ABANCA.

```
# Insertar nombre en la barra de búsqueda -----------------------

webElement$sendKeysToElement(list(Cabanca[k]))
webElement=remDr$findElement(using = "xpath",
"//div[contains(@class,'search-global-typeahead__controls')]")
webElement$clickElement()
```

3. Comprobación del número de resultados provenientes de la búsqueda.

```
dato=remDr$findElements(value =
"//span[@class='name actor-name']")
nombrevarios<-sapply(dato,function(x) x$getElementText())
vector=laply(nombrevarios,function(x) x)
```

4. Extraer la información requerida mencionada con anterioridad.

```
# Datos experiencia -----------------------

while (class(try(remDr$findElements(value =
"//h3[@class='t-16 t-black t-bold']"))) == "try-error"){
Sys.sleep(2)
}
exp=remDr$findElements(value =
"//h3[@class='t-16 t-black t-bold']")
exp<-sapply(exp,function(x) x$getElementText())
n=length(exp)
if(length(exp)!=0){
  result[4]<-paste(exp[1])
```

```
  value="\n"
  if(grepl(value,result[4])==TRUE){
    variable4<- grep("Cargo", data)
    if(length(variable4)!=0){
    result[4]<-strsplit(data[variable4[1]+1],"Cargo: ")
  }else{
    result[4]<-""}
  }
}else{
result[4]<-""}
```

5. Guardar los datos en SQL en la tabla creada como *mktg_usr.datos_scraping_linkedin*, donde identificaremos a cada cliente por su *Cliente_ID*.

```
# Guardar información en Teradata -----------------------

TERADATA <- odbcConnect("TeradataDSN",DBMSencoding = "UTF-8" )

for (r in 1:dim(my_data)[1]) {

  lanzamiento<-sqlQuery(TERADATA,paste0 ("insert into
  mktg_usr.datos_scraping_linkedin (CLIENTE_ID,CLIENTE,LOCALIDAD,PUESTO,
  EMPLEO_ACTUAL,EMPRESA,FECHA_COMIENZO_EMPLEO,DURACION_EMPLEO_ACTUAL,
  UBICACION_EMEPLO,EDUCACION,TITULACION,LICENCIAS_CERTIFICACIONES,
  APTITUDES,CORREO_ELECTRONICO,TELEFONO,FECHA_NACIMIENTO,URL,SELECCIONADO)
  VALUES(",my_data$Cliente_id[r],",'",my_data$Cliente[r],"','",
  my_data$Localidad[r],"','", my_data$Puesto[r],"','",
  my_data$Empleo_actual[r],"','", my_data$Empresa[r],"','",
  my_data$Fecha_comienzo_empleo[r],"','", my_data$Duración_empleo_actual[r],"','",
  my_data$Ubicación_empleo[r],"','",my_data$Educación[r],"','",
  my_data$Titulación[r],"','",my_data$Licencias_certificaciones[r],"','",
  my_data$Aptitudes_validaciones[r],"','",my_data$Correo_electrónico[r],"','",
  my_data$Teléfono[r],"','",my_data$Fecha_nacimiento[r],"','",
  my_data$URL[r],"',", 0,")"))

}
odbcClose(TERADATA)
```

Solamente se han puesto algunos ejemplos de cómo funcionaría el algoritmo de búsqueda, cuyo funcionamiento, se basa en detectar cuales son los elementos de la página web que debemos utilizar para extraer la información que buscamos. Todo ello se ha expuesto de manera sencilla y amena para favorecer su explicación, pero la complejidad del algoritmo es relativamente elevada, dado que es necesario contemplar todas las posibles casuísticas que puedan presentarse mediante sentencias condicionales, como puede ser que el cliente tenga o no Linkedin, o deje de tener alguno de los campos guardados, de tal forma que se pueda automatizar el proceso de búsqueda por completo.

Una vez hemos buscado la información de los clientes de la entidad objeto de estudio (concretamente 64 066 clientes) queda comprobar si los usuarios de Linkedin encontrados se corresponden realmente con dicho cliente. Llegados a este punto, es necesario recordar que mediante el algoritmo de búsqueda, un cliente de la entidad puede estar vinculado a varios perfiles de Linkedin. En concreto, para el total de clientes se han encontrado entorno a 45 000 perfiles de usuarios que se corresponderán con 24 500 clientes reales en la entidad, es decir, de los 64 066 clientes estudiados solamente 24 500 tenían asociado algún perfil de la red social.

A continuación, se muestra en la Tabla 2.4 el número de clientes y las veces que se repiten los mismos; de los 24 500 clientes a los cuales se les asignaba algún perfil de Linkedin, X solamente tenían asignado un perfil, X tenían asignado 2 perfiles y así sucesivamente.

| Número de repeticiones | Número de clientes |
|---|---|
| 1 | |
| 2 | |
| 3 | |
| 4 | |
| 5 | |
| 6 | |
| 7 | |
| 8 | |
| 9 | |
| 10 | |
| 11 | |

Tabla 2.4: Número de perfiles duplicados (modificada por confidencialidad.

Por consiguiente debido a esta situación, fue necesaria la programación de un algoritmo de comprobación que nos permitiese comprobar la fiabilidad real de cada uno de los perfiles extraídos. Para tal fin, se utiliza la base de datos de ABANCA para comparar variables como el correo electrónico, fecha de nacimiento, número de teléfono móvil o la empresa actual en la que trabaja cada cliente; de forma que se validen si los datos personales de la entidad coinciden con los datos personales de los perfiles de Linkedin. Hay que destacar, que para varios registros extraídos de la RRSS hay campos vacíos, por ello se realizará la comprobación cuando esto no ocurra. De esta manera el funcionamiento del algoritmo es el siguiente:

1. Se realiza la llamada a SQL para la tabla de datos de Linkedin y la tabla de datos de ABANCA.

```
# Selección de datos de Teradata ------------------------

# LLamada tabla de ABANCA
db <- odbcConnect("Teradata") #Abrir conexión con TERADATA
query_banco<-"SELECT * FROM   mktg_usr_scraping_total WHERE
(provincia_id='15' AND ayuntamiento_id IN ('021','031','030',
'058','005','017','075','009','008','001'))"
output_banco <- sqlQuery(db, query_banco)

# Llamada tabla de Linkedin
query_linkedin<-"SELECT * FROM mktg_usr.datos_scraping_linkedin
WHERE SELECCIONADO = 0"
output_linkedin <- sqlQuery(db, query_linkedin)

odbcClose(db) # Cerrar la conexión con TERADATA
```

2. Para cada uno de los registros de la tabla de Linkedin, se ejecuta lo siguiente:

   *a*) Se trabaja con *Cliente_id*, por tanto:

```
# Selección del Cliente_id
numero=which(output_empresa$CLIENTE_ID%in%output_linkedin$CLIENTE_ID[k])
```

   *b*) Se comprueba si el correo electrónico es el mismo en las dos tablas.

```
# Comprobación correo electrónico -----------------------

if(str_detect(output_linkedin$CORREO_ELECTRONICO[k], "@")==TRUE){

  # Selecionamos variable correo electronico
  if(output_linkedin$CORREO_ELECTRONICO[k]==
  output_banco$CORREO_ELECTRONICO[numero]){
```

```
  # Si coinciden los correos lo guardamos como VÁLIDO
  id=output_linkedin$CLIENTE_ID[k]
  TERADATA <- odbcConnect("Teradata")
  query1<-paste0("update mktg_usr.datos_scraping_linkedin set
  SELECCIONADO=1 WHERE correo_electronico=",sep="'",
  output_linkedin$CORREO_ELECTRONICO[k],sep="' ",
  "AND cliente_id=",id)
  outputsalida <- sqlQuery(TERADATA, query1)
  }
}
```

*c*) Se comprueba si la fecha de nacimiento coincide.

```
# Comprobación fecha de nacimiento -----------------------

if(str_detect(output_linkedin$FECHA_NACIMIENTO[k], "de")==TRUE){

  # modificar fechas de nacimiento para que tengan el mismo formato
  cumpleaños=output_banco$FECHA_NACIMIENTO[numero]
  cumpleaños=as.Date(cumpleaños,"%d-%m-%Y")
  cumpleaños=format(cumpleaños, format="%d de %B")
  cumpleaños=toTitleCase(cumpleaños)

  if( stringr::str_extract(cumpleaños, "^.{1}")==0){
  cumpleaños=str_replace(cumpleaños,"0","")
  }
  cumple=toTitleCase(output_linkedin$FECHA_NACIMIENTO[k])

  if(cumple==cumpleaños){

  # Si coinciden los correos lo guardamos como VÁLIDO
  id=output_linkedin$CLIENTE_ID[k]
  TERADATA <- odbcConnect("Teradata" )

  query2<-paste0("update   mktg_usr.datos_scraping_linkedin
  set SELECCIONADO=1 WHERE Fecha_nacimiento=",sep="'",
  output_linkedin$FECHA_NACIMIENTO[k],sep="'","AND cliente_id=",id)
  outputsalida <- sqlQuery(TERADATA, query2)
  }
}
```

*d*) Se comprueba si el teléfono móvil coincide.

```
# Comprobación teléfono móvil -----------------------

if(str_detect(output_linkedin$TELEFONO[k], "6")==TRUE){

# Selecionamos variable correo electronico
  if(output_linkedin$TELEFONO[k]==output_banco$TELEFONO_MOVIL[numero]){

  # Si coinciden los correos lo guardamos como VÁLIDO
  id=output_linkedin$CLIENTE_ID[k]
  cliente[k]=id
  TERADATA <- odbcConnect("Teradata")
  query3<-paste0("update   mktg_usr.datos_scraping_linkedin
  set SELECCIONADO=1 WHERE telefono=",sep="'",
  output_linkedin$TELEFONO[k],sep="'","AND cliente_id=",id)
  outputsalida <- sqlQuery(TERADATA, query3)}
}
```

De la misma manera, también se comprueba que la variable Empresa de Linkedin coincide con la que ABANCA guarda de algunos clientes.

Una vez realizado este proceso de selección ,se obtienen que de los 24 500 clientes extraídos de Linkedin, 19 000 se identificaron. De estos 19 000, 6 000 tienen el campo de empleo actual vacío, por tanto el conjunto de muestra que utilizaremos para la modelización es un total de 13 000 clientes.

Para conocer más detalles sobre esta técnica reconocida como *Web Scraping*, se recomienda ver el Capítulo 9 de Simon Munzert, Christian Rubba, Peter Meißner, Dominic Nyhuis (2015).

En la siguiente Sección 2.3, se realizará la depuración del texto de la variable empleo de cada uno de los clientes seleccionados, para ello se utilizarán técnicas de *Text Mining*.

## 2.3. Text Mining

El término *Data Mining* posee múltiples acepciones, pudiendo en general, interpretarse como un conjunto de métodos estadísticos que proporcionan información (correlaciones o patrones) cuando se dispone de muchos datos. Esta idea lleva a la siguiente estructura de conocimiento:

$$\text{Datos} + \text{Estadística} \rightarrow \text{Información}$$

Según varios autores, el *Data Mining* es aquella parte de la estadística que se usa para problemas que se presentan actualmente en el análisis de datos. Una peculiaridad de los problemas actuales, es que a diferencia de los problemas más clásicos, el número de datos a analizar es mucho mayor.

Generalmente, se basa en analizar una sobrecarga de información desde diferentes perspectivas con el objetivo de resumir los datos en segmentos de información útiles. La minería de texto permite examinar estas diferentes perspectivas ya que reúne técnicas de diversos campos, entre ellos: la lingüística, la estadística y el *Machine Learning*. Estas técnicas modelan y estructuran el texto de entrada que puede ser cualquier agrupación de documentos basados en texto, con la intención de extraer información de los patrones que siguen.

Es importante tener en cuenta que estas técnicas de minería de textos no se aplican directamente en las colecciones de documentos, sino que éstos necesitan una preparación anterior. Estas operaciones previas reciben el nombre de *preprocesado*, el cual incluye una gran variedad de pasos dependiendo del objetivo al que se pretenda llegar con los documentos.

Se debe diferenciar entre el término de *minería de datos* y *minería de textos*. Para el primero, la información se guarda en formatos estructurados, y su preprocesamiento se centra en la depuración y normalización de los datos. En cambio, en la minería de texto, el preprocesamiento se enfoca en reconocer y extraer características representativas para documentos en lenguaje natural, tales características pueden ser palabras claves relevantes, identificación de nombres, etc.

El análisis de datos obtenidos después del preprocesado, permite extraer información en formato texto de manera automatizada. Este es el objetivo principal, que normalmente se utiliza, para la toma de decisiones en el ámbito empresarial. De entre las múltiples aplicaciones de la minería de textos, en este trabajo se realizará la extracción de información y enriquecimiento de contenidos gestionando de forma eficaz la información extraída de Linkedin. Con ello, se revela la información semántica significativa que resume el contenido disponible.

El conjunto de datos con el que se va a trabajar es una colección de 13 000 muestras extraídas en la Sección 2.2. Sólo se realizará para la información de la variable empleo actual de los clientes identificados. Para llevar a cabo esta tarea, se partirá de un preprocesado de los datos, seguido de la elaboración de un clasificador que categorice a la variable en diferentes grupos de empleo para poder introducirla en los modelos como variable explicativa, Parte II.

Debido a que muchos de los clientes seleccionados tenían la información de Linkedin en otros idiomas diferentes del español o del gallego, antes de empezar con el análisis de la variable, fue necesaria la traducción de dichos registros de aquella información que no estuviese en los idiomas requeridos. Este proceso se lleva a cabo mediante el uso de *Google Cloud Plataform and APIs*, concretamente *Cloud Traslation API*. El proceso requiere de la creación de un proyecto con una solicitud para obtener la *key* y el usuario. Una vez se tiene acceso, es necesario automatizar el proceso de llamadas. El formato con el cual responde la API es el lenguaje *JSON*, por ello, se deberá utilizar el paquete `jsonlite`. Para más información acerca

de este lenguaje, se recomienda ver el Capítulo 1 de Deborah Nolan, Duncan Temple Lang (2014).

El algoritmo programado opera de la siguiente manera:

```
library(RODBC)
library(curl)
library(stringi)
library(jsonlite)
library(tcltk)

# Llamada tabla de Linkedin
query<-"SELECT * FROM mktg_usr.datos_scraping_linkedin
WHERE SELECCIONADO = 1"
salida <- sqlQuery(db, query)

# Bucle traducción ----------------
for (i in i:dim(salida)[1]) {
  # Se realiza la llamada para identificar idioma
  call_1 <- paste("https://www.googleapis.com/language/translate/v2/detect?key=***=",
  stri_encode(as.character(salida$EMPLEO_ACTUAL[i]), "", "UTF-8"), sep = "")
  con_1 <- URLencode(call_1)
  dioma <- jsonlite::fromJSON(con_1)

  # Si es español o gallego no se traduce
  if(idioma$data$detections[[1]]$language!="es"&idioma$data$detections[[1]]$language!=
  "gl"&idioma$data$detections[[1]]$confidence>0.5){

    # Hacemos la llamada para traducir
    call <- paste("https://www.googleapis.com/language/translate/v2?key=***=",
    stri_encode(as.character(salida$EMPLEO_ACTUAL[i]), "", "UTF-8"),
    "&target=es", sep = "" )
    # Nos conectamos
    con <- URLencode(call)
    # Descargamos el Json de la API
    prueba <- jsonlite::fromJSON(con)

    # Guardamos datos
    result <- data.frame(
    salida$CLIENTE_ID[i],
    prueba$data$translations[1],
    prueba$data$translations[2])
  }else{
    result <- data.frame(
    salida$CLIENTE_ID[i],
    salida$EMPLEO_ACTUAL[i] ,
    "es")
  }
  colnames(result) <- c("Cliente_id","Empleo", "Idioma")
  result2 <- rbind(result2,result)
  colnames(result2) <- c("Cliente_id","Empleo", "Idioma")
}
write_xlsx(result2,"empleo.xlsx")
```

Una vez se tiene toda la información en el mismo idioma, podemos empezar a realizar el *Data Mining*.

Para realizar este análisis de datos, se introducirá el código de las partes consideradas como más importantes. Principalmente, se usa el paquete `tm` ya que es el estándar actual para el análisis estadístico de texto en `R`. Este paquete facilita el uso de formatos de texto heterogéneos y cuenta con soporte de base de datos integrado para minimizar las demandas de memoria.

### 2.3.1. Preprocesamiento

El procesamiento de texto suele ser complejo. Esto se debe a la falta de estructura, contenido y naturaleza heterogénea de los datos con los que se trabaja. Por ello, se requiere de técnicas que los trate convenientemente.

Para poder trabajar con este tipo de documentos, se necesita de una entidad conceptual similar a una base de datos que contenga y gestione datos de texto de una manera genérica. Este concepto se denomina

corpus, que es una colección de documentos de texto plano.

Existen diversos tipos de corpus en `tm`. Los dos más importantes son el *Volatile Corpus* y el *Permanent Corpus*. Se utiliza uno u otro dependiendo de si se desea almacenar los datos en la memoria y que cuando el objeto `R` se borre, se elimine (VCorpus), o de si se quiere que los documentos se almacenen físicamente fuera de `R`, por ejemplo en una base de datos, y que no se destruya en caso de eliminarse (PCorpus).

Dentro del constructor de corpus, la variable de texto debe estar encapsulada en un objeto fuente permitiendo así trabajar sin conocer las estructuras internas de los formatos del documento de entrada.

A continuación, se muestran los primeros pasos del análisis de texto de la variable empleo:

```
#  Creación del Corpus -----------------------
traducidos=read_excel("empleo.xlsx")
corpus<- VCorpus(VectorSource(traducidos$Empleo))
```

Se pueden utilizar comandos en formato lista para seleccionar documentos. Por ejemplo, para obtener un resumen de mensajes específicos. Para ello se utiliza la función `inspect()`:

```
inspect(corpus[1:3])
content(corpus[[1:3]])
```

obteniendo al compilar:

```
<<VCorpus>>
Metadata:  corpus specific: 0, document level (indexed): 0
Content:  documents: 3

[[1]]
<<PlainTextDocument>>
Metadata:  7
Content:  chars: 11

[[2]]
<<PlainTextDocument>>
Metadata:  7
Content:  chars: 21

[[3]]
<<PlainTextDocument>>
Metadata:  7
Content:  chars: 2
```

Para obtener el texto se puede aplicar la función `content()`:

```
content(corpus[[1]])
```

```
[1] "Desarrollador de software"
```

Una vez creado el corpus, el texto de la variable está en un formato que puede ser manipulado fácilmente.

A continuación, se aplican distintos métodos para limpiar y organizar los documentos. El paquete `tm` proporciona las herramientas necesarias para ello utilizando la función de envoltura `content_transformer()`.

- **Conversión a minúsculas**.
  Con el objetivo de normalizar los textos y que sean tratados de igual manera por los algoritmos, se convierte todo a minúsculas.

  ```
  corpus<-tm_map(corpus,content_transformer(tolower))
  ```

- **Eliminación de puntuación**.
  Se eliminan los símbolos del corpus, por espacios en blanco.

  ```
  corpus<-tm_map(corpus,removePunctuation)
  ```

  Esta función no elimina los signos "¡" y "¿", por ello se crea una función para eliminar casos particulares, como los signos que aparecen a continuación:

  ```
  toSpace <- content_transformer(function(x, pattern) gsub(pattern, " ", x))
  corpus <- tm_map(corpus, toSpace, "[[:punct:] ]")
  corpus <- tm_map(corpus, toSpace, "\\s*\\([^\\)]+\\)")
  corpus <- tm_map(corpus, toSpace, "\\?")
  ```

- **Eliminación de números**.
  Debido a que no proporcionan información relevante para el empleo de los cliente, se eliminan en los textos con la siguiente función:

```
corpus<-tm_map(corpus,removeNumbers)
```

- **Se eliminan las tildes**.
  Para lograr una mayor estandarización entre palabras se eliminan las tildes.

```
corpus<-tm_map(corpus, content_transformer(chartr), old= "áàéèíñìóòúù",
new="aaeeinioouu")
```

- **Se eliminan los *stopwords***.
  A este concepto se le conoce también como palabras vacías, que no aportan significado por sí solas, por lo tanto no tienen ningún significado adicional para el análisis. Pueden ser artículos, preposiciones o pronombres, considerados de escaso valor semántico.

  Desde el paquete `tm`, se puede especificar el idioma en el que se encuentran los textos para realizar una eliminación correcta de los *stopwords*. Para lograr una mayor estandarización, también se eliminarán las tildes de igual forma que con los textos.

```
stopwords_norm<-chartr(x=stopwords("spanish"), old= "áàéèíìóòúù",
new="aaeeiioouu")
corpus<-tm_map(corpus,removeWords,stopwords_norm)
```

  Como la traducción solo se realizó para los registros que se identificaron con idioma que no eran ni español ni gallego, se creó una lista de palabras gallegas que incluyen *stopwords* que no aportan información.

```
stopwords_gallego<-readLines("palabras_eliminar.txt")
stopwords_norm_gallego<-chartr(x=stopwords_gallego, old= "áàéèíìóòúù",
new="aaeeiioouu")
corpus<-tm_map(corpus,removeWords,stopwords_norm_gallego)
```

- **Se eliminan espacios vacíos**.
  Muchos de ellos, introducidos en las transformaciones anteriores, pero es necesario eliminarlos para limpiar sus resultados intermedios y obtener una representación mejor del texto.

```
corpus <- tm_map(corpus, stripWhitespace)
```

- **Stemming**.
  El *stem* de una palabra es la base o raíz de la misma. En el paquete `tm` se encuentra la función `StemDocument()`, que elimina las terminaciones morfológicas e inflexionales comunes de las palabras. El objetivo es unificar aquellos términos que aportan la misma información por ser derivaciones de la misma palabra o palabras muy relacionadas semánticamente, reduciendo la complejidad sin pérdida importante de información para aplicaciones típicas.

```
corpus<-tm_map(corpus, stemDocument,language="spanish")
```

Como ejemplo de este preprocesamiento de los datos, se eligió un registro al azar para comparar los resultados antes y después:

```
traducidos$Empleo[2881]
```

```
[1]"Auditor administrador de Calidad de Servicio y Gestion"
```

```
content(corpus[[2881]])
```

```
[1] "auditor administrador calidad servicio gestion"
```

Para la construcción del corpus y el preprocesamiento de las variables de textos puede consultarse Eric D. Kolaczyk, Gábor Csárdi (2014).

Antes de poder realizar la clasificación de la colección de documentos depurados, se realizó un análisis estadístico del mismo con la intención de ver los empleos más comunes y la frecuencia con la que se repetían. El siguiente apartado se centrará en ello para después, poder obtener los niveles de la variable con mejor precisión.

### 2.3.2. Análisis exploratorio del texto

Una vez se tenga depurado el corpus, un enfoque común en la minería de textos es la creación de la matriz de documento-término, donde las filas corresponden a los documentos y las columnas a los términos[2]. De esta forma, se transforma un conjunto de datos textuales en un conjunto de datos estructurados. En dicha matriz, las celdas contienen un recuento del número de apariciones de un término específico en un texto dado. Los términos con mayor número de ocurrencias en un texto son catalogados como importantes.

Para la creación de esta matriz en `R` se utiliza el comando `DocumentTermMatrix(x, control=list())`;con el valor x representando al corpus y control, una lista de opciones: funciones de transformación de limpieza, reducción de la dimensión de la matriz y funciones de peso capaz de manejar la matriz de documento-término. Con la función `inspect()` se obtienen los resultados.

```
dtm<- DocumentTermMatrix(corpus)
inspect(dtm)
```

```
<<DocumentTermMatrix (documents: 13001, terms: 2905)>>
Non-/sparse entries: 25050/37126995
Sparsity           : 100%
Maximal term length: 33
Weighting          : term frequency (tf)
Sample             :
Terms
Docs    administrativo comerci director estudiant gerent ingeniero jefe profesor respons tecnico
10823              0       0        0         0      1         0    0        0       0       0
11401              0       0        0         0      0         0    0        0       0       0
3323               0       0        0         0      0         0    0        1       0       0
4498               0       0        0         0      0         0    0        0       0       0
4704               0       0        0         1      0         0    0        0       0       0
5563               0       0        0         0      0         0    0        0       1       0
5575               0       0        0         0      0         1    0        0       0       1
6662               0       0        0         0      0         0    0        0       0       1
7586               0       0        0         0      0         0    0        0       0       0
9873               0       0        0         0      0         0    0        0       0       0
```

Como se puede observar, la matriz de documento-término arroja una pequeña muestra, mientras que con `as.matrix` se produciría la matriz completa en formato denso. La matriz resultante está formada por 13 001 documentos y 2 905 términos (en este caso stems). El hecho de que cada celda en la tabla sea cero, implica que los términos no aparecen en los documentos que corresponden a las filas del corpus.

Otra manera de estudiar la importancia de cada término se basa en utilizar diferentes ponderaciones denominadas TF-IDF (*Term Frequency-Inverse Document Frecuency*). Una de las grandes ventajas de estas ponderaciones, es que se pueden aplicar a cualquier idioma. Estas ponderaciones se calculan como producto de dos medidas: la frecuencia de aparición del término (tf) y la frecuencia inversa del documento (idf).

La fórmula matemática para esta métrica es la siguiente:

$$tfidf(t,d,D) = tf(t,d) * idf(t,D) \tag{2.1}$$

con $t$ el término, $d$ cada documento, $D$ espacio total de documentos y $tdidf$ peso asignado a ese término del documento correspondiente.

La combinación de los valores de $tf$ e $idf$, proporciona una métrica que permite saber cómo de únicas son las palabras de un documento. La ponderación asigna peso alto a cada término frecuente en la colección completa, por lo contrario, si aparece pocas veces se le asigna un peso menor.

Se deduce de forma lógica, que el peso aumenta proporcionalmente al número de veces que una palabra aparece en el documento, pero es compensada por la frecuencia de la palabra en la colección de documentos, lo que permite filtrar las palabras más comunes.

[2]Si se quisiera términos como filas y documentos como columnas, se genera con la función `TermDocumentMatrix()`.

**Frecuencia de aparición del término (tf)**

Para medir la frecuencia se define $tf(t,d)$, que asigna un peso local debido a que la importancia del término $t$ solo dependerá de las frecuencias en el documento $d$ y no de otros documentos. Existen distintas formas de medir la frecuencia, entre las que destacan:

- **Recuento**.
  Número de veces que el término $t$ aparece en el documento $d$. Se utiliza para favorecer a las palabras comunes y documentos largos.
  $$tf(t,d) = n_{t,d}$$
- **Forma binaria**.
  Asigna a cada palabra del mismo documento la misma importancia.
  $$tf(t,d) = \begin{cases} 1 \text{ si t aparece en d} \\ 0 \quad \text{en otro caso} \end{cases}$$
- **Frecuencia normalizada**.
  Como cada documento tiene diferente longitud, la idea es mitigar la anomalía de observar frecuencias de términos más altas en documentos más largos que tienden a repetir la mismas palabras una y otra vez. Por ello, se divide por el número total de términos.
  $$tf(d,f) = \frac{n_{t,d}}{\sqrt{\sum_{t' \in D} n_{t',d}^2}}$$
- **Logarítmica escalada**.
  Se realiza una transformación de los valores de las frecuencias aplicando logaritmos. Se utiliza, para no dar importancia lineal a las palabras que aparezcan n veces más que otras.
  $$tf(t,d) = \begin{cases} 1 + log(n_{t,d}) \text{ si } n_{t,d} > 0 \\ 0 \quad \text{en otro caso} \end{cases}$$

**Frecuencia inversa del documento(idf)**

No siempre todos los términos tienen igual importancia, por ello, es necesario considerar ciertos casos con términos que aparecen muchas veces pero tienen poca importancia. Esta medida completa el análisis de $tf(t,d)$ considerando todos los términos con el mismo valor y actuando como corrector, de esta forma, asigna con un peso global, ya que asocia valores más bajos a términos que aparecen en muchos documentos.

- **Unitario**.
  No se llevaría a cabo ninguna corrección.
  $$idf(t,D) = 1$$
- **Frecuencia Inversa**.
  Medida de cuánta información proporciona una palabra. Si el término es común, el *idf* definido a continuación, es probable que sea bajo, si es raro será alto.
  $$idf(t,D) = log\left(\frac{N}{df(t)}\right)$$
  con $N = |D|$ número de documentos en el corpus, y $df(t) = |\{d \in D : t \in d\}|$ número de documentos en la colección que contienen el término $t$.

- **Modificación de la frecuencia inversa utilizando el máximo**.

$$idf(t,D) = max\left\{0, log\left(\frac{N - df(t)}{df(t)}\right)\right\}$$

En la práctica, las ponderaciones más utilizadas son:

- **Binaria**.
  Se define como el producto de la $tf(t,d)$ binaria y la $idf(t,D)$ unitaria. En `R` se ejecuta de la siguiente manera:
  ```
  weightBin(dtm)
  ```

- **Estándar**.
  Se define como el producto de la $tf(t,d)$ recuento o $tf(t,d)$ normalizada, con la inversa $idf(t,D)$[3].
  ```
  weightTfIdf(dtm,normalize=FALSE) o weightTfIdf(dtm,normalize=TRUE)
  ```

- **Ponderaciones en notación *SMART***.
  Los parámetros de esta función son la matriz de documento-término y *spec*, cadena de tres caracteres, siendo el primero, el esquema de frecuencia de término, el segundo, esquema de frecuencia de documento y el tercero, de normalización. Existen diferentes combinaciones para este parámetro, los más utilizados son:

  - Método binario "bnn".
  - Frecuencia logarítmica "lnn".
  - Frecuencia aumentada de términos normalizados "ann".
  - Frecuencia inversa del documento sin normalización "ntn".

  En este caso, la empleada fue la ponderación correspondiente al método binario, debido a que el interés se enfocaba en comprobar si aparecía o no el término en cada documento. Para ello, se ejecutó en `R` el código siguiente:
  ```
  weightSMART(dtm, spec = "bnn")
  ```

Además, se pueden utilizar ponderaciones personalizadas dependiendo de lo que se desee. Este apartado se ha construido siguiendo los artículos Manning, Christopher D. and Raghavan, Prabhakar and Schütze, Hinrich (2008) y Sparck, Jones, Karen (1972).

Las matrices de documento-término en grandes colecciones de documentos, como ocurre en este trabajo, suelen tener enormes dimensiones y muchas de las combinaciones son cero. Por ello, es necesario utilizar técnicas de reducción de dimensión y selección de variables importantes, con el objetivo de reducir su tamaño y el ruido de los datos, sin perder la información significativa.

La elección de las palabras claves se realiza descartando aquellas que aparecen de forma ocasional y las que aparecen muy frecuentemente en el corpus. Con el fin de realizar el filtrado, se intentan suprimir los términos que no tengan significado, para ello se eliminan aquéllos con una longitud menor de 2 letras y mayor de 20, ya que se supone que los términos de más de 20 letras serán agrupaciones de caracteres sin sentido.

```
dtm <- DocumentTermMatrix(corpus, list(wordLengths= c(2,15)))
```

También puede resultar muy útil eliminar los términos que aparecen en muy pocos documentos antes de proceder a la clasificación. Para ello, se utiliza el comando `removeSparseTerms()` que conserva todos los términos que cumplan:

$$df(t) > N(1 - sparse)$$

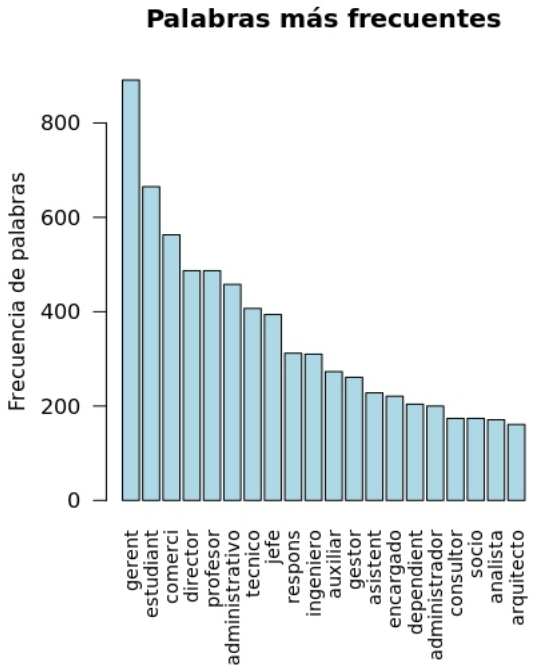
con $df$ frecuencia de documentos del término $t$ y $N$ número de vectores. El parámetro $sparse$ toma valores entre 0 y 1. Si el umbral de escasez se considera al 0.95 %, se toman los términos que aparecen en más del 5 % de documentos. De esta forma:

```
dtm <- removeSparseTerms(dtm, sparse= 0.95)
```

[3]En `R`, para calcular la frecuencia inversa, se utiliza en logaritmo en base 2, por ello si se realiza a mano, tendría que tenerse en cuenta para que los resultados fuesen comparables.

### 2.3.3. Métodos gráficos

Para poder continuar trabajando con la clasificación, se necesita tener una idea clara de cómo son los textos con los que se trabaja. Para ello, se puede mostrar un diagrama de barras o una nube de palabras sobre la matriz de documento-término una vez modelada con los pasos anteriores del preprocesado de texto.




(a) Diagrama de barras de la variable empleo.


(b) Nube de palabras.

En la Figura 2.4a, se muestran únicamente los 20 términos más frecuentes en el corpus. La nube de palabras o *worldcloud* que se observa en la Figura 2.4b, es una representación que visualmente resulta más atractiva que el diagrama de barras, en este caso se representaron 150 términos donde los más frecuentes se representan por mayor tamaño. Como se puede apreciar en los dos gráficos, aparecen las raíces de cada uno de los términos, debido a que ya se ha realizado el preprocesado necesario.

Existen diversas formas de representar gráficamente el texto analizado por el corpus, si se muestra interés en estos métodos se recomienda leer el Capítulo 8 de Viswanathan, Viswa and Viswanathan, Shanthi and Gohil, Atmajitsinh and Yu-Wei, Chiu David Chiu (2016).

### 2.3.4. Clasificación

En los apartados anteriores se han realizado los pasos previos a la clasificación de la colección de texto. Existen dos tipos de clasificación: la supervisada y la no supervisada. La primera, en la que la respuesta es conocida y en la segunda, también llamada *cluster*, no se tienen las posibles respuestas. No es posible compararlas ya que cada una de ellas tiene unas propiedades concretas, por lo que dependiendo de cuál sea el objetivo, se emplea una u otra.

La clasificación supervisada requiere especificar las categorías para el algoritmo de clasificación. El inconveniente es que estos clasificadores pueden necesitar mano de obra, debido a la gran variedad de datos de entrenamiento.

Los métodos de clasificación no supervisada generan ellos mismos categorías. Esto es interesante para el estudio de las principales líneas de división en el corpus lingüístico. Además, no necesita una codificación manual de los datos. Pero también tiene inconvenientes; requiere que se especifiquen el número concreto de categorías en las que se va a agrupar el corpus o que se establezcan un criterio que determine ese

número, y además, la interpretación a posteriori de los resultados de la estimación suele ser muy compleja.

Ante la duda para utilizar un método u otro, existen algunas propiedades que se pueden usar como guía para seleccionar el correcto. Suele ser una elección difícil, por ello, lo mejor es que se pruebe directamente con un simple ensayo observando los errores. A lo largo de este trabajo se han probado las dos alternativas, pero con la clasificación no supervisada han surgido muchos problemas debido al gran número de observaciones y a su amplio campo semántico, lo que ha llevado a realizar la categorización por un algoritmo determinístico.

Los principales problemas del método *cluster* han sido:

- Dficultad de encontrar un algoritmo para conjuntos de datos de alta dimensión y que fuese eficiente computacionalmente.
- Dificultad a la hora de interpretar los resultados debido a la cantidad de grupos que se generan. Para poder visualizar los resultados con claridad, fue necesario prescindir de palabras que aparecen en más del 40 % de los documentos.
- Aunque también se han probado bigramas, la relación entre los términos más frecuentes que hay en la observación, no se correspondían con las categorías lógicas.
  Por ejemplo, un doctor en cirugía no es igual que un doctor en matemáticas. Al realizar la clasificación no supervisada, si se eliminan las palabras cirugía y matemáticas, estos dos clientes de la entidad se clasificarían en el mismo nivel.

El entorno de las ocupaciones cambian sustancialmente cada año, principalmente desde el punto de vista de las nuevas tecnologías y en especial, Internet, que ha producido un impacto importante en los métodos de trabajo, destacando el sector de las TIC (por sus siglas, *Tecnologías de la Información y la Comunicación*). Escoger una clasificación adecuada ha sido laborioso, sobre todo por el desconocimiento que se tiene de los empleos que existen en la actualidad.

Para solventar el problema, este trabajo se apoyó en la clasificación estadística nacional de ocupaciones, realizada por el INE (*Instituto Nacional de Estadística*)[4]. El cual garantiza su mantenimiento y actualización, mediante la elaboración y publicación de cuantas notas explicativas y normas interpretativas sean necesarias, para un uso en el ámbito estadístico. El objetivo de esta clasificación, es garantizar el tratamiento uniforme de los datos estadísticos sobre ocupaciones en el ámbito nacional y su comparabilidad internacional y comunitaria. De esta forma, el listado de ocupaciones que proporciona es público y actualizado.

Los niveles principales de la clasificación del INE, son los siguientes:

- Grupo 1: Directores y gerentes.
- Grupo 2: Técnicos y profesionales científicos e intelectuales.
- Grupo 3: Técnicos; profesionales de apoyo
- Grupo 4: Empleados contables, administrativos y otros empleados de oficina.
- Grupo 5: Trabajadores de los servicios de restauración, personales, protección y vendedores.
- Grupo 6: Trabajadores cualificados en el sector agrícola, ganadero, forestal, y pesquero.
- Grupo 7: Artesanos y trabajadores cualificados de las industrias manufactureras y la construcción.
- Grupo 8: Operadores de instalaciones y maquinarias, y montadores.
- Grupo 9: Ocupaciones elementales.
- Grupo 0: Ocupaciones militares.

[4]Para más información ver https://www.ine.es/daco/daco42/clasificaciones/cno11_notas.pdf.

A partir del análisis descriptivo de los datos, se construyó un algoritmo determinístico para clasificar la variable empleo de los clientes de la entidad. Este algoritmo selecciona algunos de los niveles principales anteriores y algunos los subdivide en los grupos de nivel dos del INE, con el objetivo de intentar conseguir que la variable sea lo más homogénea posible. La finalidad de esta subdivisión es obtener un resultado del trabajo más fiable. También se añaden un grupo para los autónomos, otro para los que no tengan experiencia laboral (estudiantes, becarios, voluntarios), otro de jubilados, en paro y un quinto de no clasificados, para poder automatizar el proceso. Los grupos resultantes serían los que se muestran en la Tabla 2.5.

| | |
|---|---|
| 1-1 directores ejecutivos y legislativos | 1-2 directores administrativos y comerciales |
| 1-3 gerentes de producción | 1-4 gerentes de restauración y comercio |
| 1-5 propietarios-gerentes de establecimiento | 1-6 responsables con cargo altos |
| 2 profesionales intelectuales de salud | 3 profesionales de enseñanza |
| 4 científicos e intelectuales | 5 profesionales del derecho |
| 6-1 administración de empresas | 6-2 tecnologías de la información |
| 6-3 ciencias sociales | 6-4 cultura y espectáculo |
| 6-5 administración publica | 6-6 técnicos sanitarios |
| 7 técnicos y profesionales de apoyo | 8-1 oficinistas que no atienden al publico |
| 8-2 empleados de oficina que al publico | 9-1 trabajadores de restauración y comercio |
| 9-2 trabajadores de servicios de salud y cuidados personales | 9-3 trabajadores de protección |
| 10 trabajadores cualificados en el sector agrícola | 11 artesanos y trabajadores de industrias manufacturas |
| 12 operadores de instalaciones | 13 ocupaciones elementales |
| 14 ocupaciones militares | 15 pensionistas |
| 16 sin experiencia laboral | 17 en paro |
| 18 autónomos | 19 no clasificados |

Tabla 2.5: Niveles de la clasificación.

Una vez realizado el análisis de texto de la Subsección 2.3.2 y adquirido el conocimiento necesario acerca de la frecuencia de los términos del corpus, se ejecuta el algoritmo y se obtiene una clasificación de 32 grupos, consiguiendo de esta forma, reducir de manera notable la heterogeneidad presente en la variable empleo inicial. Por consiguiente, se obtiene la nueva variable a introducir en el modelo.

Para realizar el objetivo del trabajo, en la Parte II se implementará esta variable constituida por información de redes sociales en el modelo de propensión de ahorro e inversión vigente en la entidad, y se cuantificará el aumento de su precisión frente al modelo actual por medio de una comparativa. En la siguiente sección, el interés se enfocará en realizar un análisis del comportamiento de la relación existente para cada variable explicativa y la variable respuesta, necesario antes de la construcción del modelo.

Esta sección se ha construido siguiendo el Capítulo 1 de Graham Williams (2011) y el Capítulo 10 de Simon Munzert, Christian Rubba, Peter Meißner, Dominic Nyhuis (2015).

## 2.4. Análisis de las variables

Antes de entrenar un modelo predictivo, o incluso antes de realizar cualquier cálculo con un nuevo conjunto de datos, es muy importante realizar una exploración descriptiva de los mismos. Este proceso permite entender mejor qué información contiene cada variable, así como detectar posibles errores y dar pistas sobre qué variables pueden no ser adecuadas como predictores en el modelo.

Una vez extraída y estructurada la nueva variable correspondiente al empleo actual de los clientes, en esta sección, se realizará un análisis del comportamiento de cada una de las variables que entrarán en los distintos modelos de clasificación vistos en el Capítulo 3.

Una de las primeras comprobaciones que hay que hacer, es verificar que cada variable se ha almacenado con el tipo de valor que le corresponde, es decir, que las variables numéricas sean *números* y las cualitativas *factor*. También es muy importante, estudiar la distribución de la variable objetivo, ya que a fin de cuentas, es lo que nos interesa predecir.

Para que un modelo predictivo sea útil, debe de tener un porcentaje de acierto superior a lo esperado por azar o a un determinado nivel basal. En problemas de clasificación, el nivel basal es el que se obtiene si se asignan todas las observaciones a la clase mayoritaria (la moda). En este problema como se muestra en la Figura 2.5, dado que para el XX % de las obsrvaciones de los clientes no se produce inversión, si siempre se predice $Y = 0$ el porcentaje de aciertos será aproximadamente del XX %. Este es el porcentaje mínimo que hay que intentar superar con los modelos de predicción.

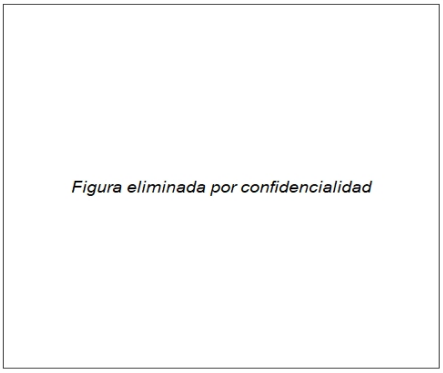



Figura 2.5: Comportamiento de la variable objetivo.

Cada mes se observan 19 variables de 13.000 clientes de la entidad. Cómo se muestra en la Tabla 2.6, la tasa de éxito se sitúa en torno al XX %.

| Fecha | Nº clientes | Éxitos | % éxitos |
|---|---|---|---|
| 31/05/2018 | | | % |
| 30/06/2018 | | | % |
| 31/07/2018 | | | % |
| 31/08/2018 | | | % |
| 30/10/2018 | | | % |
| 31/10/2018 | | | % |
| 30/11/2018 | | | % |
| 31/12/2018 | | | % |
| 31/01/2019 | | | % |
| 28/02/2019 | | | % |
| 31/03/2019 | | | % |
| 30/04/2019 | | | % |

Tabla 2.6: Tasa de éxitos en el horizonte temporal modificada por confidencialidad.
El análisis concreto de cada variable, se ha eliminado por confidencialidad.

### 2.4.1. Correlación entre variables

La representación gráfica de la distribución de las variables en función de si los clientes invierten o no, ayuda a tener una idea de qué variables pueden ser buenos predictores para el modelo y cuales no aportan información, o la que aportan, es redundante. Aunque la creación de un buen modelo debe entenderse como un proceso iterativo, en el que se van ajustando y probando, existen ciertas herramientas de análisis que pueden ayudar a realizar una selección inicial adecuada.

Entre estas herramientas, se encuentra el estudio de correlación. Si dos variables numéricas están muy correlacionadas, añaden información redundante al modelo, por lo tanto, no conviene incorporar ambas. Si esto ocurre, se puede excluir aquella que se considere como que no está realmente asociada con la variable respuesta, combinarlas para recoger toda su información en una única variable o aplicar otras técnicas como: componentes principales, proyección pursuit, etc.

Para los modelos de predicción se considera una correlación alta entre variables a un valor superior a 0.7. Tras el estudio de correlación de las variables vistas en la Sección 2.4, no hay ningún par que presente una correlación lineal problemática. Como se muestra en la Figura 2.6, las variables que más presentan correlación, son las variables $VISTA$ o $PLAZO$ con $CAPACIDAD\_AHORRO$.

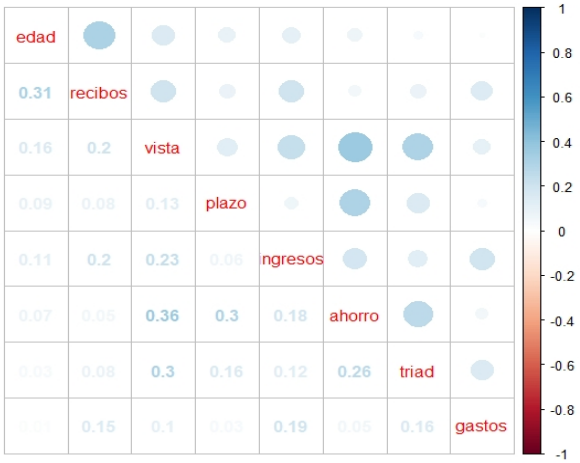



Figura 2.6: Correlación de las variables numéricas del modelo.

# Capítulo 3

# Modelos de clasificación

Los métodos predictivos como la regresión lineal, generan modelos globales en los que una única ecuación se aplica a todo el espacio muestral. Cuando el análisis implica múltiples predictores que tienen una relación compleja y no lineal con la variable respuesta, es muy difícil encontrar un modelo global que sea capaz de reflejar buenos resultados. Existen métodos de ajuste no necesariamente lineal, como los modelos lineales generalizados, vistos en la Subsección 3.1.2, que realizan los ajustes necesarios dependiendo de la distribución de las variables para llevar la linealidad a otra escala. Sin embargo, para determinados problemas utilizar otros modelos de predicción puede mejorar las estimaciones de los efectos de los métodos más clásicos y por tanto, sus resultados.

En el transcurso práctico de este trabajo se aplicarán diferentes modelos de predicción que plantearán alternativas interesantes cuando la linealidad de los datos sea una hipótesis alejada de la realidad, como los Árboles de decisión CART (del inglés, *Classification And Regression Trees*), una idea sencilla e intuitiva. Los métodos estadísticos basados en árboles, engloban un conjunto de técnicas supervisadas no paramétricas que consiguen segmentar el espacio de las variables predictoras en regiones simples, dentro de las cuales es más sencillo manejar las interacciones.

La metodología apoyada en Árboles de decisión, se ha convertido en uno de los referentes dentro del ámbito predictivo debido a los buenos resultados que generan en áreas muy diversas. En la Sección 3.2 de este Capítulo, se examina la complejidad de la estructura sobre la que se construyen los árboles, sus propiedades y las singularidades de sus variedades. A lo largo de los años, esta técnica ha ido evolucionando en modelos más complejos y eficientes, como los métodos *Boosting* y *Bagging* que se contemplan en la Sección 3.3.

Con la finalidad de examinar las diferentes técnicas de modelización en el desarrollo de este Capítulo, se representa como $X$ al conjunto total de datos definidos en la Sección 1.2 más los datos extraídos por técnicas de *Web Scraping*, y como $X_1, \cdots, X_p$ al conjunto de variables explicativas. La idea es poder predecir la variable respuesta en función de los resultados obtenidos por las variables explicativas, en este caso, la variable objetivo $Y$ es un indicador de incrementos en productos de ahorro o inversión con valores dicotómicos:

$$Y = \begin{cases} 1 \text{ (se produce incrementos en productos de inversión)} \\ 0 \text{ (en otro caso)} \end{cases}$$

La variable respuesta en este problema toma dos valores, por lo que el objetivo se centra en agrupar las predicciones como $Y = 1$ o $Y = 0$, a este proceso se le reconoce como clasificación supervisada. En este tipo de aprendizaje, el método se entrena con un histórico de datos y en base al comportamiento a lo largo del tiempo de las observaciones, aprende a asignar la etiqueta de la variable objetivo adecuada a un nuevo valor.

Cómo se ha comentado, existe un gran número de paradigmas desarrollados bien en el campo de la Estadística (como la regresión logística), o bien por modelos de aprendizaje automatizado (del inglés, *Machine Learning*, como los Árboles de decisión), capaces de realizar las tareas propias de la clasificación. En este Capítulo, se explicarán cada uno de estos métodos para poder entender su funcionamiento y utilizarlos correctamente en la práctica.

**Partición de los datos**

Paso previo a aplicar un método de clasificación supervisada y en general, en cualquier proceso de modelización, consiste en dividir el conjunto de datos inicial en conjuntos de datos más pequeños que serán utilizados con los siguientes fines: entrenamiento y validación. El subconjunto de datos de entrenamiento se utiliza para estimar los parámetros del modelo y el subconjunto de datos de validación se emplea para comprobar el comportamiento predictivo del modelo estimado. Cada registro de la base de datos debe aparecer en uno de los dos subconjuntos, por ello, la división se realiza a partir de la variable $CLIENTE\_ID$, para que no se repita el mismo individuo en muestras diferentes y mitigar el efecto de la dependencia. Para separar el conjunto de datos, se utiliza un procedimiento de muestreo aleatorio, simple o estratificado.

Normalmente, se separa[1] el conjunto de datos en el 70 % para el entrenamiento y el 30 % para validación, pero esto puede generar en la implementación de algunos modelos[2] una dependencia entre los dos subconjuntos, provocando que las predicciones no sean del todo fiables. Esto ocurre porque ciertos métodos requieren establecer hiperparámetros que moldean su estructura, por tanto es recomendable dividir la muestra de entrenamiento en dos y validar el resultado en una muestra totalmente nueva.

Como lo ideal es entrenar un modelo con un conjunto de datos independiente de los datos con los que realizamos la validación, en este trabajo se decidió separar los datos en tres grupos: entrenamiento 60 %, test 20 % y validación 20 %, realizando la predicción sobre la muestra de validación independiente de la muestra de entrenamiento y de la muestra de test. La simple comparación del resultado obtenido en la muestra de validación permite validar el modelo en término de error de predicción.

## 3.1. Modelos de regresión

Las técnicas de clasificación supervisada son una parte esencial del *Machine Learning*. Actualmente existe una gran variedad de algoritmos predictivos, sin embargo, la regresión logística continúa siendo un método clásico, pero robusto. En particular, define un método de regresión útil para resolver problemas de clasificación binaria ya que describe y estima, la relación entre una variable dicotómica dependiente y las variables independientes.

El hilo de la exposición de esta sección comenzará por comprender el funcionamiento de modelos más simples, en concreto, el modelo de regresión múltiple que nos permitirá entender las peculiaridades de modelos más complejos como es el caso de la regresión logística, siendo ambos modelos, casos particulares de los modelos de regresión lineal múltiple generalizada.

### 3.1.1. Modelo de regresión lineal múltiple

El modelo de regresión lineal múltiple es una extensión del modelo lineal simple, el cual sirve para expresar la dependencia de una variable respuesta $Y$, con respecto a un conjunto de variables explicativas $X_1, \ldots, X_p$ de forma lineal, como bien indica su nombre. La función de regresión, entendida como la medida de la variable respuesta condicionada a los valores de $X$, presenta la siguiente forma:

$$f(X) = \mathbb{E}(Y \mid X) = \beta_0 + \sum_{j=1}^{p} X_j \beta_j \tag{3.1}$$

siendo $\beta = (\beta_1, \ldots, \beta_p)^T$ el conjunto de parámetros desconocidos. De este modo, la variable $Y$ se puede explicar en función de su media condicionada más un error no observable $\varepsilon$. De tal forma, que el modelo de regresión lineal múltiple se puede formular como:

$$Y = f(X) + \varepsilon = \beta_0 + \sum_{j=1}^{p} X_j \beta_j + \varepsilon \tag{3.2}$$

[1] Se podría emplear validación cruzada en su lugar.
[2] Concretamente, para los métodos *Boosting* y *Bagging*.

donde $\varepsilon$ habitualmente es un variable aleatoria gaussiana con media 0 y varianza $\sigma^2$ constante, $\varepsilon \sim N(0, \sigma^2)$.

Reescribiendo la Ecuación (3.2) de manera matricial, obtenemos que $X$ se corresponde con una matriz de diseño de dimensión $N \times (p+1)$ cuya $i$-ésima fila, con $i = 1, \cdots, N$, se corresponde con la $i$-ésima observación, la primera columna está formada por unos y la $j$-ésima columna, $j = 2, \cdots, p+1$, recoge los valores de la variable explicativa $j-1$. Si se denota por $Y$ al vector de observaciones de la respuesta, por $\beta$ al vector de parámetros y por $\varepsilon$ al vector de errores, la Ecuación (3.2) puede escribirse como $Y = X\beta + \varepsilon$, descomponiéndose en la siguiente expresión:

$$\begin{pmatrix} Y_1 \\ Y_2 \\ \vdots \\ Y_N \end{pmatrix} = \begin{pmatrix} 1 & X_{1,1} & X_{1,2} & \dots & X_{1,p} \\ 1 & X_{2,1} & X_{2,2} & \dots & X_{2,p} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & X_{N,1} & X_{N,2} & \dots & X_{N,p} \end{pmatrix} \begin{pmatrix} \beta_0 \\ \beta_1 \\ \vdots \\ \beta_p \end{pmatrix} + \begin{pmatrix} \varepsilon_1 \\ \varepsilon_2 \\ \vdots \\ \varepsilon_N \end{pmatrix} \tag{3.3}$$

con $Y \in M_{N\times 1}$, $X \in M_{N\times(p+1)}$ y $\beta \in M_{(p+1)\times 1}$. Donde ahora $Y$ y $X$ serán respectivamente un vector y una matriz de valores conocidos dados por los datos muestrales.

#### Hipótesis del modelo y diagnosis

Para realizar la estimación o el contraste sobre los parámetros de la Ecuación (3.2), el procedimiento habitual asume las siguientes hipótesis:

- **Linealidad**: Se asume que $\mathbb{E}(X \mid Y)$ es lineal.
- **Homocedasticidad**: Se supone que la varianza del error será la misma para cualesquiera valores de las variables explicativas $Var(\widehat{\varepsilon} \sim (X_1, \cdots, X_p)) = \sigma^2$.
- **Normalidad**: Los residuos siguen una distribución normal tal que $\widehat{\varepsilon} \sim N(0, \sigma^2)$.
- **Independencia**: Los residuos del modelo, $\widehat{\varepsilon}_1, \ldots, \widehat{\varepsilon}_N$, deben ser independientes.

Para más información acerca de los modelos más sencillos de regresión lineal se recomienda ver el Capítulo 3 de Ruppert, David (2004) y el Capítulo 3 de Trevor Hastie, Robert Tibshirani, Jerome Friedman (2013).

Nos centraremos en la regresión logística utilizada en la Parte II como modelo de predicción. Es un caso particular de los GLM, por ello en la siguiente sección se profundizará en sus propiedades.

### 3.1.2. Modelo lineal generalizado (GLM)

Los modelos de regresión lineal generalizado (GLM, por sus siglas en inglés *Generalized Linear Models*), buscan unificar diferentes modelos estadísticos de regresión dentro de un marco global que incluya las distintas posibilidades de incorporar variables, tanto discretas como continuas de distribución no gausiana en la respuesta. Dentro de este grupo, se encuentran modelos diferentes como la regresión logística utilizada en la parte práctica de este trabajo.

El modelo generalizado es mucho más versátil que el modelo de regresión lineal múltiple ya que no se impone que la distribución de las $Y_i$ (condicionada por las $X_i$) sea necesariamente normal, sino una familia exponencial con esperanza condicional $\mu_i = \mathbb{E}[Y_i \mid X_i]$ con $i = 1, \cdots, N$. Tampoco se impone una relación lineal directa entre la variable respuesta y las variables explicativas, por todo ello, es factible utilizar los GLM para modelar la variable respuesta $Y_i$ de tipo continuo, de recuento de observaciones o de tipo binario.

Más en concreto, considerando $Y_i$ una variable i.i.d. $Y$ y $X_i$ la i-ésima fila de la matriz del modelo X, conocida una vez que se obtiene una muestra de las covariables (la cual pasa a denotarse por $\mathbf{X}$), en los GLM se asume que la distribución de las $Y_i \mid X_i$ se enmarca de manera más general en la familia exponencial como se define a continuación.

**Definición 3.1.1.** *En los modelos lineales generalizados la distribución de las $Y_i \mid X_i$ pertenece a la familia exponencial y su función de densidad admite la expresión:*

$$f\big(Y_i \mid \theta_i, \xi\big) = exp\left\{ \frac{Y_i\theta_i - b(\theta_i)}{a(\xi)} + c(Y_i, \xi) \right\} \tag{3.4}$$

*donde $\theta_i$ se denomina parámetro canónico de localización, $\xi$ parámetro de escala y, $a(.)$, $b(.)$ y $c(.)$ funciones que determinan el tipo de familia exponencial (ver Wood, Simon N. (2017, pág: 104)).*

Como se ha mencionado en un modelo lineal generalizado la forma en que las covariables suministran información sobre la media $\mu_i$ de la variable dependiente, ya no es necesariamente lineal. Denótese por $\eta_i = X_i^T\beta$ (normalmente llamado predictor lineal). Entonces, la forma en la cual las covariables suministran información sobre $\mu_i$ se establece mediante una función *link* $g(.)$, monótona y diferenciable, que relaciona el predictor lineal con la media de la siguiente manera:

$$\eta_i = g(\mu_i) = X_i^T\beta \quad con\ i = 1, \ldots, N \tag{3.5}$$

Denotando por $h(.)$ a la inversa de la función link, $h = g^{-1}$, se tiene:

$$\mu_i = h(\eta_i) = h(X_i^T\beta) \quad con\ i = 1, \ldots, N \tag{3.6}$$

Por tanto, un modelo de regresión lineal generalizado vendrá especificado por la selección de un tipo de distribución de la familia exponencial para la distribución condicionada $Y_i \mid X_i$, por la función link $g$, y por el vector o matriz de diseño $X$.

Para cada familia exponencial existe una función link que iguala al parámetro natural o canónico $\theta_i$ con el predictor lineal $X_i^T\beta$, de modo que:

$$\theta_i = g(\mu_i) = \eta_i = X_i^T\beta \tag{3.7}$$

Destacar, que la media de cualquier variable aleatoria perteneciente a una distribución de la familia exponencial viene dada por la primera derivada del parámetro específico $b$, en términos de $\theta$. Esta expresión, $b'(\theta) = \mu_i$, es la clave que permite relacionar los parámetros $\beta$ a ajustar en el modelo GLM con los parámetros canónicos de la familia exponencial.

En un principio $a$ puede ser cualquier función de $\xi$, lo cual no da problemas cuando dicho parámetro es conocido. Sin embargo si este dato es desconocido, el desarrollo se vuelve mucho más complejo a menos que se pueda escribir $a(\xi) = \xi/w$, donde $w$ representa una constante conocida que puede variar de observación a observación y $\xi = \sigma^2$ un valor constante que se conoce como parámetro de dispersión. Esta última suposición, recoge las distribuciones tradicionales pertenecientes a la familia exponencial y además, permite heterocedasticidad en la varianza, ya que se reescribiría en términos de parámetros distribucionales como $Var(Y) = b''(\theta)\sigma^2/w$.

Para conocer más acerca de estos modelos se recomienda ver el Capítulo 3 de Wood, Simon N. (2017). Una vez vistas las características principales de los GLM se explicará la regresión logística.

### Regresión logística

Como caso particular de los modelos lineales generalizados se va a mostrar el desarrollo de la regresión logística, la cual tiene gran importancia dentro del contexto de los métodos de clasificación y por tanto, tiene gran interés conocer sus características y ventajas.

La regresión logística estudia problemas de regresión donde la variable respuesta es dicotómica, a diferencia de la regresión lineal donde se consideraba continua. Este modelo se crea con el objetivo de

conseguir afrontar una toma de decisiones ante dos posibles sucesos. Un ejemplo muy utilizado sería emplearlo para dictaminar si un paciente padece o no una enfermedad en base a los valores de ciertos parámetros especiales relacionados con esta.

De este modo, se considera la variable respuesta $Y$ como variable binaria o dicotómica tomando dos valores representados por 0 y 1 respectivamente. Por ser binaria, su distribución es una Bernoulli cuya media será una probabilidad de éxito, obteniéndose:

$$\mathbb{E}\left(Y_i|X = X_i\right) = \mathbb{P}\left(Y_i = 1|X = X_i\right) \tag{3.8}$$

$$Var\left(Y_i|X = X_i\right) = \mathbb{P}\left(Y_i = 1|X = X_i\right)\left[1 - \mathbb{P}\left(Y_i = 1|X = X_i\right)\right] \tag{3.9}$$

siendo ahora $Y_i$ el valor de la variable respuesta para el individuo i-ésimo y $X_i \in \mathbb{R}^p \ \forall p \in \mathbb{N}$ un vector que contiene los valores de las variables explicativas para $i \in \{1, \cdots, N\}$.

Un modelo lineal representaría la respuesta de la siguiente manera:

$$Y_i = X_i^T\beta + \varepsilon_i \text{ para } i \in \{1, \cdots, N\}$$

donde $\beta$ es el vector de coeficientes y $\varepsilon_i$ el error.

Este modelo es inadecuado puesto que no se cumplen las suposiciones básicas que se tenían en cuenta para construir un modelo lineal múltiple. No se tiene linealidad, pues si se intenta expresar la media de $Y$ como función lineal, es decir $\mathbb{E}(Y_i|X = X_i) = X_i^T\beta$, provoca que en muchos casos las predicciones no estén en el intervalo $[0, 1]$ en el cual tienen que estar siempre, pues la Ecuación (3.8) es una probabilidad de éxito. Por otra parte, ahora el modelo es heterocedástico ya que por la Ecuación (3.9) se observa que la varianza no es constante sino que depende de cada observación. Finalmente se observa que al estar ante una distribución de Bernoulli no se cumple la normalidad y que la hipótesis de independencia sería la única que podría cumplirse (será necesario para el modelo logístico). Bajo estas condiciones es obvio considerar otro modelo.

Un enfoque diferente que consiga solucionar estos problemas podría ser construir un modelo para la probabilidad de éxito condicionada a cada valor de la variable explicativa, la cual coincide con la función:

$$\pi(X) = \mathbb{P}(Y = 1|X = X_i)$$

Si se desea considerar un modelo lineal habría que aplicar a $\pi(X)$ una función que transforme el intervalo $[0, 1]$ de definición en toda la recta real, de forma que se trabajaría con:

$$g(\pi(X, \beta)) = X^T\beta$$

donde $g$ es la función *link* de los modelos lineales generalizados. En esta situación, teniendo una variable respuesta dicotómica se suele considerar la función *logit* que tiene la siguiente forma:

$$g(p) = log\left(\tfrac{p}{1-p}\right) \ \forall p \in [0, 1]$$

Sustituyendo $p$ por la probabilidad de éxito, aplicar la función *logit* se basa en aplicar un logaritmo (en base $e$) al cociente entre la probabilidad de éxito y la probabilidad de fracaso. Este cociente se conoce como la *odds* (disparidad) y se puede formalizar como:

$$Odds(Y) = \frac{\mathbb{P}(Y = 1)}{\mathbb{P}(Y = 0)}$$

Si la probabilidad de éxito toma valores en $[0, 1]$ la *odds* toma en $[0, +\infty]$. Al aplicarle el logaritmo a la *odds* se toma los valores en el intervalo $(-\infty, \infty)$ pudiendo ser, por consiguiente, explicada mediante un modelo lineal. De esta forma, el modelo consistirá en expresar el logaritmo de la *odds* de variable respuesta $Y$ como función lineal de la variable explicativa,

$$log\left(\frac{\pi(X, \beta)}{1 - \pi(X, \beta)}\right) = X^T\beta$$

Si se representa el modelo cómo la probabilidad de éxito es necesario invertir la función *logit*. Esto es posible, ya que produce una correspondencia biunívoca y creciente entre $[0,1]$ y el intervalo $(-\infty,\infty)$, siendo tanto ella como su inversa funciones suaves, es decir, derivables con derivadas continuas. La inversa de la función *logit* es:

$$g^{-1}(X) = \frac{e^X}{1+e^X}$$

Por tanto, el modelo logístico expresa la probabilidad de éxito de la siguiente manera:

$$\pi(X,\beta) = g^{-1}(X,\beta) = \frac{e^{X^T\beta}}{1+e^{X^T\beta}} > 0 \tag{3.10}$$

con valor positivo y mayor que cero, ya que $e^{X^T\beta} > 0$.

Destacar que si $X^T\beta = 0$ la probabilidad será de 0.5, si $X^T\beta > 0$ será mayor de 0.5 teniendo como asíntota el valor 1 y recíprocamente, si es menor será más pequeña de 0.5 teniendo como asíntota el 0. Por ello, el modelo logístico parece adecuado para representar el comportamiento de la función de regresión con variable respuesta dicotómica.

**Interpretación de los coeficientes de regresión**

Para explicar la interpretación de los coeficientes $\beta$ de la regresión logística consideraremos el caso unidimensional y dos situaciones, que la variable explicativa $X$ sea discreta o que sea continua. Para el primer caso, contemplando el caso más sencillo de que $X$ sólo tome dos valores, $X = 1$ o $X = 0$, la estructura del modelo de regresión sería:

$$\pi(0,\beta) = \mathbb{P}(Y=1|X=0) = \frac{e^{\beta_0}}{1+e^{\beta_0}}$$
$$\pi(1,\beta) = \mathbb{P}(Y=1|X=1) = \frac{e^{\beta_0+\beta_1}}{1+e^{\beta_0+\beta_1}}$$

con $\beta_0$ intercepto asociado al grupo de referencia y $\beta_1$ parámetro que representa el incremento de pasar del primer grupo al segundo.

Si se quiere estimar la probabilidad de éxito para el grupo de referencia $X = 0$, se tiene:

$$\hat{p_0} = \frac{\sum_{i=1}^N I_{\{X_i=0\}} Y_i}{\sum_{i=1}^N I_{\{X_i=0\}}}$$

Pensando que $\beta$ será tal que $\hat{p_0} = \pi(0,\hat{\beta})$, para obtener el valor de $\hat{\beta}_0$ basta tener en cuenta:

$$e^{\hat{\beta}_0} = \frac{\pi(0,\hat{\beta})}{1-\pi(0,\hat{\beta})} = \frac{\hat{p}_0}{1-\hat{p}_0} \Rightarrow \hat{\beta}_0 = \log\left(\frac{\hat{p}_0}{1-\hat{p}_0}\right)$$

De esta manera, el intercepto $\beta_0$ resulta ser el logaritmo de la *odds* de la variable respuesta condicionada al grupo de referencia. Si $\beta_0$ es negativo, la probabilidad de éxito es menor de 0.5; mientras que si es positivo es mayor de 0.5 y si fuese nulo sería exactamente 0.5.

Actuando de forma similar con el otro grupo se tiene que:

$$e^{\hat{\beta}_0+\hat{\beta}_1} = \frac{\pi(1,\hat{\beta})}{1-\pi(1,\hat{\beta})} = \frac{\hat{p}_1}{1-\hat{p}_1}$$

con qué $e^{\hat{\beta}_0+\hat{\beta}_1}$ es la *odds* correspondiente al segundo grupo. Para extraer el parámetro $\hat{\beta}_1$ es necesario efectuar el cociente entre ambas *odds*:

$$e^{\hat{\beta}_1} = \frac{e^{\hat{\beta}_0+\hat{\beta}_1}}{e^{\hat{\beta}_0}} = \frac{\hat{p}_1/(1-\hat{p}_1)}{\hat{p}_0/(1-\hat{p}_0)}$$

este cociente se conoce como la *oddsratio* que se define para dos poblaciones respecto de una variable binaria $Y$ como el cociente de la *odds* en una y otra población, de la forma:

$$OddsRatio = \frac{Odds(Y|\text{ Prob. 2})}{Odds(Y|\text{Prob.1})} = \frac{\mathbb{P}(Y=1|\text{ Prob. 2})/\mathbb{P}(Y=0|\text{ Prob .2})}{\mathbb{P}(Y=1|\text{ Prob. 1})/\mathbb{P}(Y=0|\text{ Prob. 1})}$$

y se estima por:

$$\hat{OddsRatio} = e^{\hat{\beta}_1} = \frac{\hat{p}_1/(1-\hat{p}_1)}{\hat{p}_0/(1-\hat{p}_0)}$$

indicando el incremento de la *odds* del primer grupo al pasar al segundo. Los coeficientes de regresión pueden presentar signo negativo o positivo, mientras que la *odds* y la *oddsratio* siempre toman valores positivos, y se valora si son mayores o menores que uno.

En el otro caso particular para una variable explicativa continua $X$ no existen expresiones explicitas para los estimadores, pero se interpretan a través de la *odds* y *oddsratio*. Así, bajo este modelo:

$$Odds\,(X=X_0) = \frac{e^{\beta_0+\beta_1 X_0}/\left(1+e^{\beta_0+\beta_1 X_0}\right)}{1-e^{\beta_0+\beta_1 X_0}/\left(1+e^{\beta_0+\beta_1 X_0}\right)} = e^{\beta_0+\beta_1 X_0}$$

$$OddsRatio(X=X_0+1 \text{ frente a } X=X_0) = \frac{e^{\beta_0+\beta_1(X_0+1)}}{e^{\beta_0+\beta_1 X_0}} = e^{\beta_1}$$

de tal forma que $e^{\beta_1}$ representa la *odds* al haber incrementado la variable explicativa en una unidad. La diferencia con la regresión lineal es que hay que tener en cuenta que se incrementa de manera multiplicativa.

**Estimación de los parámetros del modelo**

Volviendo al caso multidimensional general para un vector de variables explicativas con $X_i$ la i-ésima fila de la matriz del modelo $X$ , se emplea el método de máxima verosimilitud para estimar los parámetros del modelo. Para una muestra aleatoria simple $(X_1, Y_1), \cdots (X_N, Y_N)$ donde cada $Y_i \in \text{Bernoulli}(\pi(X_i, \beta))$ su función de masa de probabilidad se define como:

$$P(Y=Y_i|X=X_i) = p_i^{Y_i}\,(1-p_i)^{1-Y_i} \text{ con } Y_i \in \{0,1\}$$

siendo $p_i = \pi\,(X_i, \beta)$ la probabilidad de éxito. De esta manera, la función de verosimilitud es:

$$L(\beta) = \prod_{i=1}^{N}[\pi\,(X_i,\beta)^{Y_i}\,(1-\pi\,(X_i,\beta))^{1-Y_i}]$$

y su logaritmo

$$\log(L(\beta)) = \sum_{i=1}^{N}[Y_i \log(\pi(X_i,\beta)) + (1-Y_i)\log(1-\pi(X_i,\beta))] \tag{3.11}$$

Teniendo en cuenta que el vector de derivadas se calcula como:

$$\frac{\partial}{\partial\beta}\left[Y_i \log\left(\pi\left(X_i,\beta\right)\right) + \left(1-Y_i\right)\log\left(1-\pi\left(X_i,\beta\right)\right)\right] = \frac{\partial\pi\left(X_i,\beta\right)}{\partial\beta}\left(\frac{Y_i-\pi\left(X_i,\beta\right)}{\pi\left(X_i,\beta\right)\left(1-\pi\left(X_i,\beta\right)\right)}\right)$$

y derivando la Ecuación (3.11) respecto de $\beta$, se obtiene:

$$\frac{\partial\log(L(\beta))}{\partial\beta} = \sum_{i=1}^{N}\frac{\partial\pi(X_i,\beta)}{\partial\beta}\frac{1}{\pi(X_i,\beta)(1-\pi(X_i,\beta))}[Y_i-\pi(X_i,\beta)] \tag{3.12}$$

Si la función de regresión paramétrica $\pi(X,\beta)$ adopta la forma de $\frac{e^{X^T\beta}}{1+e^{X^T\beta}}$, tras los cálculos de sustitución se tiene:

$$\frac{\partial\pi(X,\beta)}{\partial\beta} = X^T\pi(X,\beta)(1-\pi(X,\beta)) \tag{3.13}$$

Si se sustituye la Ecuación (3.13) en la Ecuación (3.12) e igualando a cero para obtener el máximo se alcanza:

$$\frac{\partial \log(L(\beta))}{\partial \beta} = \sum_{i=1}^{N} X_i^T \left[Y_i - \pi\left(X_i, \beta\right)\right] = 0$$

para que $\beta$ sea el estimador se ha de cumplir que esta derivada esté igualada a cero. A esta ecuación se le conoce como la ecuación de verosimilitud.

Estas ecuaciones de verosimilitud no tienen una solución explícita pues $\pi(X, \beta)$ no es una función lineal respecto de $\beta$, de modo que para resolverlas va a ser necesario recurrir a la información proporcionada por la matriz hessiana, cuya formulación es:

$$\frac{\partial^2 \log L(\beta)}{\partial \beta^2} = -\sum_{i=1}^{N} X_i X_i^T \pi\left(X_i, \beta\right)\left(1 - \pi\left(X_i, \beta\right)\right)$$

con $\partial\beta^2 = \partial\beta\partial\beta^T$. La expresión $X_i X_i^T$ es una matriz simétrica, semidefinida positiva y de rango uno, debido a que se multiplica por $\pi(X_i, \beta)(1 - \pi(X_i, \beta))$ que es mayor que cero, como se puede ver en la estructura vista en la Ecuación (3.10). Por lo que la suma será definida positiva cuando los $X_i$ no estén en un espacio lineal de dimensión inferior, de forma que los vectores $X_i$ serán independientes entre sí. En consecuencia, la matriz hessiana será semidefinida o definida negativa respectivamente obteniendo un estimador de máxima verosimilitud, porque en el último caso, la raíz de ecuaciones de verosimilitud sería un máximo de la función de verosimilitud.

En forma matricial quedaría como:

$$\frac{\partial^2 \log L(\beta)}{\partial \beta^2} = -\sum_{i=1}^{N} X_i X_i^T \pi\left(X_i, \beta\right)\left(1 - \pi\left(X_i, \beta\right)\right) = -\mathbf{X}^T\mathbf{V}\mathbf{X}$$

Recordando las ecuaciones de verosimilitud anteriores si se expresan de forma matricial se obtiene:

$$\frac{\partial log L(\beta)}{\partial \beta} = \mathbf{X}^T(\mathbf{Y} - \pi(\mathbf{X}, \beta)) = 0$$

Para este sistema de ecuaciones de verosimilitud no se va a tener habitualmente una solución analítica por lo que deberá resolverse de forma numérica mediante un método iterativo. El más habitual es el método de mínimos cuadrados ponderados $IRLS$, también denominado método de las marcas de Fisher (*Fisher Scoring*), otra alternativa es el método de Newton-Raphson.

Concretamente, el método de Newton-Raphson se basa en que a partir de un iterante inicial ($\beta_0$), escogido adecuadamente, se calculan los sucesivos iterantes. Un buen iterante podría ser un estimador $\hat{\beta}$ obtenido empleando el método de mínimos cuadrados en el problema. Pero la matriz $\mathbf{X}^T\mathbf{V}\mathbf{X}$ está mal condicionada impidiendo: la convergencia del método, y que la matriz hessiana no sea definida negativa.

Esta sección se ha construido siguiendo lo expuesto en el Capítulo 4 de Trevor Hastie, Robert Tibshirani, Jerome Friedman (2013) y en el Capítulo 3 de Wood, Simon N. (2017).

### 3.1.3. Problemas ante un conjunto grande de variables explicativas

Tanto en los modelos lineales múltiples como en los modelos lineales generalizados, es necesario afrontar el problema de la posible *colinealidad* entre sus covariables. Una solución para evitar estos problemas es reducir el número de variables explicativas que entran en juego, determinando cuáles son las más eficaces a la hora de predecir la variable respuesta y de cuáles se puede prescindir, provocando que el modelo sea más sencillo, más fácil de interpretar, aumentando la precisión de estimación y evitando problemas como la multicolinealidad.

Para obtener el modelo "óptimo" lo ideal sería evaluar todos los modelos posibles. Si el número de variables explicativas es grande, en lugar de emplear una búsqueda exhaustiva se puede emplear un criterio por pasos:

- **Selección progresiva (forward)**.
  Partiendo de la situación en la que no hay ninguna variable y en cada paso se incluye una aplicando un *criterio de entrada*, hasta que ninguna de las restantes lo verifiquen.

- **Eliminación progresiva (backward)**.
  Se parte del modelo con todas las variables y en cada paso se elimina una aplicando un *criterio de salida*, hasta que ninguna de las incluidas lo verifican.

- **Regresión paso a paso (stepwise)**.
  El más utilizado, se combina un criterio de entrada y uno de salida. Normalmente se parte sin ninguna variable y en cada paso puede haber una inclusión y una exclusión (forward/backward).

Cuando el número de variables explicativas es muy grande, o si el tamaño de la muestra es pequeño en comparación, pueden aparecer problemas al emplear los métodos anteriores, incluso pueden no ser aplicables. Como alternativa se utilizan los métodos de regularización, que imponen restricciones adicionales a los parámetros que los "retraen" hacia cero, produciendo una reducción en la varianza de predicción (a costa del sesgo). Estos modelos consideran todas las variables explicativas desde el inicio y añaden una penalización proporcional al número de parámetros. Se pueden clasificar en:

- **Ridge**.
  Este modelo de regularización expone un método que suma una cantidad positiva a los autovalores para lograr que la matriz hessiana sea siempre invertible, realizando una penalización del tamaño de los coeficientes en la regresión a la hora de minimizar la suma residual de cuadrados. Se le conoce como regularización tipo $L_2$. Reduce los problemas que surgen cuando las variables están muy correlacionadas, puesto que cuando esto sucede, un coeficiente muy grande y positivo puede ser anulado por otro de magnitud similar y negativo, a causa de que ambas variables a las que acompaña están muy correlacionadas entre sí. En definitiva, se trata de estimar:

$$\max_{\beta} \left\{ \sum_{i=1}^{N} \left[ Y_i \left( \beta_0 + \beta^T X_i \right) - \log \left( 1 + e^{\beta_0 + \beta^T X_i} \right) \right] - \lambda \sum_{j=1}^{p} \left( \|\beta_j\|_2 \right)^2 \right\}$$

  donde $\lambda \geq 0$ se define como el parámetro que controla la cantidad de reducción. Para hallar su valor se puede recurrir a métodos de validación cruzada. Cuanto más grande es el valor de $\lambda$, mayor es la reducción y por tanto el modelo se queda con menos variables explicativas ya que los coeficientes se aproximan a cero.

  Se puede escribir como:

$$\hat{\beta}^{RR} = \max_{\beta} \sum_{i=1}^{N} [Y_i(\beta_0 + \beta^T X_i) - \log(1 + e^{\beta_0 + \beta^T X_i})]$$

  sujeto a $\sum_{j=1}^{p} \beta_j^2 \leq t$, existiendo entre $\lambda$ y $t$ una relación proporcionalmente inversa.

  Empleando el algoritmo de Newton-Raphson se resuelve esta penalización y se consigue que la matriz hessiana deje de ser singular, garantizando que se pueda estimar $\beta$ tomando como $\mathbf{X}^T\mathbf{V}\mathbf{X}$ la nueva matriz generada.

- **Lasso**.
  Minimiza la suma residual de cuadrados pero imponiendo una restricción en la suma de los valores absolutos de los coeficientes. Esta restricción implementa la posibilidad de que algunos de los coeficientes sean nulos y se consiga un modelo con interpretación más sencilla. Al igual que el método Ridge, busca un estimador que deje de ser insesgado pero que a cambio se reduzca la varianza de los valores estimados y se mejore en conjunto la precisión de la predicción.

  En este caso se impone una penalización de tipo $L_1$ donde se tendrá que resolver un problema de programación cuadrático, con la ventaja de que permite reducir el número de parámetros no nulos

de forma más eficiente. Se buscaría encontrar:

$$\max_{\beta} \left\{ \sum_{i=1}^{N} \left[ Y_i \left( \beta_0 + \beta^t X_i \right) - \log \left( 1 + e^{\beta_0 + \beta^T X_i} \right) \right] - \lambda \sum_{j=1}^{p} |\beta_j| \right\}$$

o lo que es análogo, resolver el problema:

$$\hat{\beta}^{\mathrm{RL}} = \max_{\beta} \sum_{i=1}^{n} \left[ Y_i \left( \beta_0 + \beta^T X_i \right) - \log \left( 1 + e^{\beta_0 + \beta^T X_i} \right) \right]$$

sujeto a $\sum_{j=1}^{p} |\beta_j| \leq t$. De igual forma que en la regularización Ridge, se establece una relación inversa entre el coeficiente $\lambda$ y $t$.

Se puede ver la idea de la regularización Lasso frente a la Ridge en la Figura 3.1 para un caso bidimensional, donde las regiones azules se corresponden con las áreas comprendidas en $|\beta_1| + |\beta_2| \leq t$ y en $\beta_1^2 + \beta_2^2 \leq t^2$; y las elipses rojas representan los contornos de la función de mínimos cuadrados.

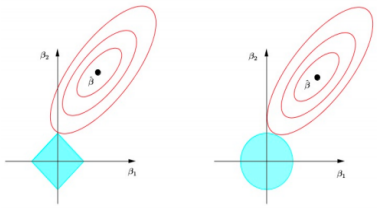


Figura 3.1: Gráfica extraída de Trevor Hastie, Robert Tibshirani, Jerome Friedman (2013).

Se observa cómo en el caso de la regresión Lasso los contornos de la función de mínimos cuadrados cortan a la región azul en un vértice situado en un eje, provocando que la otra componente sea nula ($\beta_1 = 0$), resultando ser, más fácil de interpretar.

El parámetro $\lambda$ al igual que en el caso de la penalización Ridge, se escoge mediante validación cruzada. Se puede resolver de nuevo por el algoritmo de Newton-Raphson así como por otras técnicas numéricas. La matriz $\mathbf{X}^T\mathbf{V}\mathbf{X}$ obtenida en este problema se va a poder invertir, por lo menos se va a poder implementar en el algoritmo para obtener una estimación del parámetro $\beta$.

La principal diferencia práctica entre Lasso y Ridge es que el primero consigue que algunos coeficientes sean exactamente cero, por lo que realiza selección de variables predictoras, mientras que el segundo no llega a excluir ninguno. Esto supone una ventaja notable de Lasso en escenarios donde no todos las variables son importantes para el modelo y se desea que las menos influyentes queden excluidas.

Por otro lado cuando existen variables altamente correlacionadas (linealmente), Ridge reduce la influencia de todos ellas a la vez y de forma proporcional, mientras que Lasso tiende a seleccionar una de ellas dándole todo el peso y excluyendo al resto. En presencia de correlaciones, esta selección varía mucho con pequeñas perturbaciones incitando a que las soluciones de Lasso sean muy inestables.

- **Regularización con penalización Elastic Net**.

  Para conseguir un equilibrio óptimo entre la presencia de correlaciones y selección de variables menos influyentes se recurre a la penalización Elastic Net[3], la cual es un compromiso entre ambas y tiene la siguiente forma:

$$\max_{\beta} \left\{ \sum_{i=1}^{N} \left[ Y_i \left( \beta_0 + \beta^t X_i \right) - \log \left( 1 + e^{\beta_0 + \beta^t X_i} \right) \right] - \lambda \sum_{j=1}^{p} \left( \alpha \left| \beta_j \right| + (1 - \alpha) \left\| \beta_j \right\|_2^2 \right) \right\} \tag{3.14}$$

[3]Será la penalización que se utilice en la creación del modelo de regresión logística visto en la Parte II.

con $\alpha \in (0, 1)$.

El parámetro $\lambda$ regula el peso dado a la regularización impuesta por Ridge y Lasso. En el caso de que exista cierta colinealidad entre varias variables predictoras, Elastic Net tenderá a escoger una o todas en función de cómo se ha parametrizado.

Es fácil ver, que es equivalente hallar la solución de la Ecuación (3.14) a resolver:

$$\hat{\beta}' = \max_{\beta} \sum_{i=1}^{N} [Y_i(\beta_0 + \beta^t X_i) - \log(1 + e^{\beta_0 + \beta^t X_i})]$$

sujeto a $\sum_{j=1}^{p} (\alpha|\beta_j| + (1-\alpha)\beta_j^2) \leq t$.

Al igual que ocurría con las penalizaciones de $L_1$ y $L_2$, teniendo en cuenta que es una restricción promedio, se puede hallar la solución mediante métodos numéricos como Newton-Raphson. Esta modificación también resuelve la no invertibilidad de la matriz $\mathbf{X^T V X}$, hallando un estimador de $\beta$.

Para más información acerca de métodos de regularización se recomienda ver el Capítulo 3 de Trevor Hastie, Robert Tibshirani, Jerome Friedman (2013).

Si la relación entre los predictores y la variable respuesta es aproximadamente lineal, entonces un modelo de regresión lineal dará buenos resultados. Si de lo contrario, existe una relación altamente no lineal y compleja, los métodos basados en árboles de decisión que se explicarán en la siguiente sección, pueden superar los resultados obtenidos por los métodos más clásicos.

## 3.2. Árboles de Decisión

Los Árboles de decisión son métodos de clasificación supervisados que permiten una interpretación sencilla, pues segmentan el espacio definido por las variables predictoras en un número simple de regiones (por ejemplo rectángulo para el caso bidimensional) que se caracterizan por incluir observaciones lo más similares posibles en cuanto a la variable respuesta. Para obtener las predicciones, se suele usar la media o la moda de las observaciones de entrenamiento en la región en la que cada observación a predecir pertenece. Son conceptualmente simples pero potentes. Esta idea fue introducida por Leo Breiman en el año 1984, y ha ido evolucionando a lo largo de los años derivando en técnicas más eficientes y complejas como los métodos *Bagging* y *Boosting*.

Es uno de los métodos más utilizados para la predicción de una variable respuesta, dependiendo de la categoría de dicha variable, los Árboles de decisión se clasifican en dos conjuntos: Árboles de regresión cuando la variable respuesta es continua, y Árboles de clasificación si por el contrario, la variable respuesta es discreta.

La idea en las dos categorías consiste en desarrollar el modelo en forma de árbol, es decir, en primer lugar se empieza con todo el conjunto de datos y mediante divisiones binarias se va dividiendo el conjunto a través de ramificaciones sobre alguna de las variables explicativas hasta alcanzar una condición de parada, de manera que, el árbol dejará de ramificarse y el conjunto de datos que queda en ese nodo se denominará *terminal*.

A continuación, se describe un método popular tanto para Árboles de regresión como para Árboles de clasificación llamado CART, desarrollado por matemáticos de la universidad de Berkeley y Stanford (Breiman, Friedman, Olshen y Stone) a mediados de los 80. Esta técnica propone segmentar la base de datos obteniendo una estructura de árbol compleja, y por tanto, árboles con mayor profundidad. El objetivo del particionamiento recursivo es acabar en nodos terminales que sean homogéneos. La completa homogeneidad de los nodos terminales es un idea raramente alcanzada en el análisis de datos reales. De esta manera, el criterio de parada de CART se basa en la idea de hacer las variables resultantes en los

nodos terminales tan homogéneas como sea posible.

Una medida cuantitativa de la homogeneidad es la noción de impureza, el cociente de número de sujetos que cumplen la característica en el nodo entre el número total de sujetos en el nodo. Así se selecciona el corte que conduce al mayor decrecimiento de la impureza, consiguiendo descendientes homogéneos en la variable respuesta. Por ello, es necesario conocer las diferentes condiciones de parada y las medidas empleadas para los criterios de división.

Con la finalidad de comprender la idea principal de los Árboles de decisión, se considera un ejemplo particular de un problema de regresión con una variable respuesta continua $Y$ y las variables explicativas $X^T = \{X_1, X_2\}$, cada una de ellas tomando valores en el intervalo unitario. La imagen superior izquierda de la Figura 3.2, muestra una partición del espacio de características por líneas paralelas a los ejes de coordenadas. En cada elemento de la partición se modela la variable respuesta $Y$ con una constante diferente. Aunque en cada partición la línea tiene una descripción simple, algunas de las regiones resultantes son complicadas de describir.

Para simplificar se restringe a particiones binarias recursivas como se muestra en la imagen superior derecha de la Figura 3.2, realizando primero una división del espacio en dos regiones y después, modelando la respuesta por la media de $Y$ en cada una de las regiones resultantes. De esta manera se eligen la variable y el punto de corte de la división para lograr un mejor ajuste, produciendo la partición de una o ambas regiones en dos regiones o más, continuando el proceso hasta aplicar una condición de parada. Por ejemplo en la imagen superior derecha, primero se selecciona el punto de corte $X_1 = t_1$, la región $X_1 \leq t_1$ se divide en $X_2 = t_2$ y la región $X_1 > t_1$ en $X_1 = t_3$. Finalmente, la región $X_1 = t_3$ se fracciona en $X_2 = t_4$. El resultado de este proceso es una partición en las cinco regiones $R_i$ con $i = 1, \cdots, 5$.

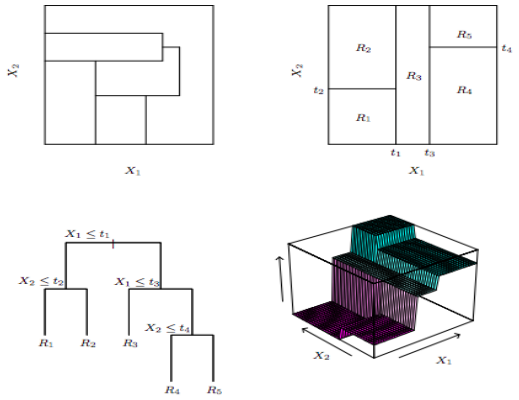



Figura 3.2: Particiones y CART (Trevor Hastie, Robert Tibshirani, Jerome Friedman (2013)).

El correspondiente modelo de regresión divide el espacio de las covariables en hiper-rectángulos donde todas las observaciones que queden dentro de la misma región $R_j$ tendrán el mismo valor estimado $c_j$, de esta manera se tiene:

$$\hat{f}(X) = \sum_{j=1}^{5} c_j I_{\{(X_1, X_2) \in R_j\}}$$

Este modelo se representa en la imagen inferior izquierda de la Figura 3.2 por el árbol binario. Una ventaja de esta estructura recursiva es su interpretación. Se encuentran los siguientes componentes: nodos, ramas y hojas; los nodos son las variables de entrada que cumplen una condición, las ramas representan los posibles valores de las variables considerando el punto de corte y las hojas o nodos hijos, muestran los valores de las variables que cumplan esta restricción. Por último la imagen inferior derecha de la Figura 3.2, es un diagrama en perspectiva de la superficie de regresión de este modelo.

Existen otros métodos diferentes para árboles de decisión, entre ellos el método C4.5. Para más información acerca de ellos se recomienda ver el Capítulo 9 de Trevor Hastie, Robert Tibshirani, Jerome Friedman (2013).

### 3.2.1. Árboles de regresión

Para los árboles de regresión se tienen $p$ variables explicativas y una variable respuesta $Y$ para cada una de las $N$ observaciones, es decir, $(X_i, Y_i)$ para $i = 1, \cdots, N$ con $X_i = (X_{i,1}, \cdots, X_{i,p})$. El problema estadístico es establecer una relación entre la variable objetivo y las variables explicativas de tal forma que sea posible predecir $Y$ basado en los valores de $X$. Expresado de otra forma, se quiere estimar la esperanza condicional de $Y$ tal que:

$$\mathbb{E}(Y|X = X_i)$$

El algoritmo decide automáticamente sobre las variables y los puntos de división, así como la topología o forma que debe tener el árbol.

Es fácil ver en la Figura 3.2 que las $R_j$ regiones se corresponden con los nodos terminales. Para una respuesta continua, una elección natural de la impureza para un nodo o una región $R_j$, es el criterio de mínimos cuadrados $imp(R_j) = \sum_{i \in R_j}(Y_i - \hat{f}(X_i))^2$ con $\hat{f}(X_i)$ el promedio de los $Y_i's$ dentro de dicha región. De esta forma se puede escribir:

$$N_j = \sum_{i=1}^{p} I_{\{X_i \in R_j\}}$$

$$\hat{f}(X_i) = \frac{1}{N_j} \sum_{X_i \in R_j} Y_i$$

Encontrar la mejor partición binaria en términos de mínimos cuadrados, es por lo general, computacionalmente inviable. Por ello se suele realizar mediante un algoritmo de partición. Se empieza considerando una variable $k$ y un punto $t$ de división y luego se define el par de regiones de la siguiente forma:

$$R_L(k,t) = \{X|X_k \leq t\} \text{ y } R_R(k,t) = \{X|X_k > t\}$$

Para dividir una región en dos subregiones, por ejemplo $R_L$ y $R_R$, se busca la variable $k$ y el punto $t$ que resuelvan:

$$\min_{k,t} \left[ \min_{\hat{f}_L} \sum_{X_i \in R_L(k,t)} \left(Y_i - \hat{f}_L(X_i)\right)^2 + \min_{\hat{f}_R} \sum_{X_i \in R_R(k,t)} \left(Y_i - \hat{f}_R(X_i)\right)^2 \right]$$

escogiendo $\hat{f}_L(X_i)$ y $\hat{f}_R(X_i)$ como los promedios de los $Y_i's$ en cada subregión.

Una vez escogidas las mejores variables de división, se vuelve a repetir el proceso separando los datos en las dos regiones resultantes y así sucesivamente. Un árbol muy grande podría sobreajustar los datos mientras que un árbol pequeño podría no explicar la estructura de los datos. Por ello, se puede decir que el tamaño del árbol es un parámetro de ajuste complejo y puede suavizarse utilizando un criterio de parada.

Para entender el procedimiento de parada o poda del árbol, se define un subárbol $b \subset b_0$, se denotan a los nodos terminales por $j$ representados por la región $R_j$, y a $J$ al número de nodos terminales en $b$. Entonces se le llama medida de impureza o error promedio cuadrático entre la variable objetivo y el promedio en la región, a $Q_j(b)$ expresado como:

$$Q_j(b) = \frac{1}{N_j} \sum_{X_i \in R_j} \left(Y_i - \hat{f}(X_i)\right)^2 \tag{3.15}$$

Además, se define el criterio de complejidad de costes como:

$$C_\gamma(b) = \sum_{j=1}^{J} N_j Q_j(b)) + \gamma|J| \tag{3.16}$$

La idea del criterio de parada es resolver $min_{b \subset b_0} C_\gamma(b)$, donde $b$ es cualquier subárbol que se puede obtener de $b_0$ al ramificarlo y $\gamma$ un hiperparámetro que penaliza el tamaño de los árboles. De esta forma para valores altos de $\gamma$ se construyen árboles más pequeños y si $\gamma = 0$ se obtiene el árbol $b_0$. Este parámetro se puede determinar usando validación cruzada.

### 3.2.2. Árboles de clasificación

Si el objetivo es un resultado de clasificación para la variable respuesta $Y$, la cual toma los valores $1, 2, \cdots, K$, la lógica a seguir es la misma que en los árboles de regresión cambiando solamente los criterios de división y el criterio de parada. Para el Árbol de regresión se utiliza el error cuadrático medio como medida de impureza para realizar la división, pero no es adecuado para clasificación que se tendrá que hacer sobre otras cantidades ligadas a la nueva naturaleza de estas variables.

El primer cambio respecto a los Árboles de regresión será sobre la cantidad asignada como la media aritmética $\hat{f}(X_i)$ en la región $R_j$. En este nuevo contexto la clase con mayor número de participantes será la que ponga etiqueta a la región, y será el valor que se dará en la predicción a los registros que caigan en la misma. Para el Árbol de clasificación en un nodo $j$, que representa la región $R_j$ con $N_j$ observaciones de $X_i$, se define:

$$\hat{p}_{jk} = \frac{1}{N_j} \sum_{X_i \in R_j} I_{\{Y_i = k\}}$$

como la proporción de observaciones de la clase $k$ en el nodo $j$. De manera que se clasifican las observaciones $X_i$ en el nodo $j$ como $\hat{G}(X_i) = arg\ max_k\ \hat{p}_{jk}$, con $\hat{G}$ clasificador que representa la clase mayoritaria en dicho nodo. Las medidas más empleadas como funciones de impureza para la construcción de árboles de clasificación son:

$$\begin{aligned}
&\text{Error de clasificación (err):} && imp(R_j) = 1 - max_k(\hat{p}_{jk}) \\
&\text{Índice de Gini:} && imp(R_j) = \sum_{k \neq k'} \hat{p}_{jk}\hat{p}_{jk'} = \sum_{k=1}^{K} \hat{p}_{jk}\,(1 - \hat{p}_{jk}) \\
&\text{Entropía:} && imp(R_j) = -\sum_{k=1}^{K} \hat{p}_{jk} \log(\hat{p}_{jk})
\end{aligned}$$

El *Error de clasificación* se basa en el porcentaje de observaciones que no pertenece a la clase más común del nodo, considerando como clase, cada uno de los posibles grupos de la variable respuesta $Y$. El *Índice de Gini* es una métrica que mide la frecuencia con la que un elemento elegido al azar se identifica incorrectamente; el objetivo es obtener un valor bajo debido a que nos indica una mayor pureza del nodo. Según Breiman, Leo (2001), este método de división se emplea en diversas técnicas como en el *Random Forest* (ver Subsección 3.3.2). Por último, la *Entropía* es una manera de contabilizar la dispersión de cada clase, si una división tiene valores de una sola clase se considerará una división pura y por tanto su valor es cero; pero si la frecuencia de cada clase es la misma, el valor de la *Entropía* alcanza el valor máximo de 1. Por tanto, cuanto más bajo sea este valor mejores resultados se obtendrán.

Para el caso particular con $Y$ variable dicotómica con valores 0 y 1, se tiene que para el nodo menos impuro la impureza es 0 y debe tener como resultado $\mathbb{P}[Y = 1|R_j] = 0$ o $\mathbb{P}[Y = 1|R_j] = 1$. El nodo es más impuro cuando su impureza es 1 con $\mathbb{P}[Y = 1|R_j] = 1/2$. Por tanto, la función impureza tiene una forma cóncava y se puede definir como $imp(\{Y = 1|R_j\})$. Si $p$ es la proporción en $Y = 1$, las tres medidas de impureza serían $1 - max(p, 1 - p)$, $p(1 - p)$ y $-plogp - (1 - p)\log(1 - p)$ respectivamente. Las tres medidas son similares, pero la *Entropía* y el *Índice de Gini* son diferenciables y por tanto, más susceptibles a la optimización numérica. Al igual que en el Árbol de regresión, se necesitan ponderar las medidas de impureza de cada nodo por el número de observaciones que entren en él.

Para realizar la división de un Árbol de clasificación, en Gareth James; Trevor Hastie; Robert Tibshirani; Daniela Witten (2013), se recomienda utilizar las últimas dos medidas comentadas debido a que el *Error de clasificación* no es suficientemente sensible para la creación de buenos árboles.

Debido a que por motivos computacionales no es factible comprobar todas las posibilidades de división, se recurre al método comentado en la Figura 3.2, particiones binarias recursivas (en inglés *recursive binary splitting*). De esta forma no se evalúan todas las posibles divisiones pero si que se alcanza un buen equilibrio entre el resultado y el proceso computacional.

En cuanto al criterio de parada para los Árboles de clasificación, al igual que en los Árboles de regresión consiste en encontrar el subárbol del árbol saturado con la mejor calidad en cuanto a que sea más predictivo de los resultados y menos vulnerable al ruido en los datos. Para alcanzar el objetivo de parada se debe tener certeza de que los nodos terminales son homogéneos, es decir, la calidad de un árbol es simplemente la calidad de sus nodos terminales. Por tanto, para un Árbol de clasificación se pueden utilizar las medidas de impureza descritas anteriormente como métrica de observaciones mal clasificadas en un nodo. Sustituyendo $Q_j$ por $imp(R_j)$ en la Ecuación (3.16), se obtendría el criterio de complejidad.

A continuación, se detallan los pasos del algoritmo de división de un árbol con variable objetivo dicotómica:

1. Todas las observaciones pertenecen a la misma región. Se suele tomar el conjunto de entrenamiento para la construcción del árbol y el conjunto de validación para comprobar el buen funcionamiento. De esta forma se comprueba a posteriori la eficacia del árbol, ya que puede funcionar bien en los datos usados para la construcción pero no actuar de la misma manera para realizar predicciones (produciéndose un sobreajuste).

2. Se identifican todos los puntos de corte $t$ sobre los que se divide el árbol para cada una de las variables explicativas. En el caso de que sean cualitativas, los puntos de corte serán cada uno de sus niveles, y para valores continuos, se ordenan de menor a mayor los valores muestrales y se selecciona el punto medio entre cada par de valores como punto de corte.

3. Se calcula la medida deseada para los dos conjuntos que se formarían en la división, definidos como $R_L$ y $R_R$, de la siguiente manera:

| *REGRESION* | *CLASIFICACION* |
|---|---|
| RSS $\rightarrow Qj = \frac{1}{N_j} \sum_{X_i \in R_j} \left(Y_i - \hat{f}(X_i)\right)^2$ | Índice de Gini $\rightarrow imp(R_j) = 2\hat{p}_j(1-\hat{p}_j)$ |
| | Entropía $\rightarrow imp(R_j) = -(\hat{p}_j log(\hat{p}_j) + (1-\hat{p}_j) klog(1-\hat{p}_j))$ |
| | Error rate $\rightarrow imp(R_j) = 1 - max(\hat{p}_j, 1-\hat{p}_j)$ |

Tabla 3.1: Medidas de división para árbol de regresión y árbol de clasificación con $j = \{R, L\}$.

   Se ponderaría como $N_L * imp(R_L) + N_R * imp(R_R)$, siendo $N_j$ el promedio de observaciones dentro de cada nodo $j$.

4. El menor valor obtenido para las medidas ponderadas del paso anterior se selecciona como división óptima, y se realiza la partición.

5. Se repite el proceso para cada una de las regiones que se han creado en la división anterior hasta alcanzar algún criterio de parada.

Esta sección se ha construído con la ayuda del Capítulo 9 de Trevor Hastie, Robert Tibshirani, Jerome Friedman (2013) y el Capítulo 2 de Leo Breiman, Jerome Friedman, Richard A. Olshen, Charles J. Stone (1984).

### 3.2.3. Parámetros del modelo y como evitar sobreajuste en árboles de decisión

El sobreajuste es uno de los desafíos más importantes en el proceso de modelización de árboles de decisión. Si no se definen límites, el árbol tendrá un 100 % de precisión en el conjunto de datos de entrenamiento. Existen dos formas particulares de evitar el sobreajuste:

- Definir restricciones sobre el tamaño del árbol, permitiendo ramificaciones sólo cuando la reducción en el error supera un cierto umbral.
  - Definir un mínimo de observaciones para dividir un nodo.
  - Definir un mínimo de observaciones para un nodo terminal.

  - Determinar la máxima profundidad del árbol.
  - Máximo número de ramas del árbol.

- Podar el árbol.
  Consiste en la construcción de un árbol de decisión sin prácticamente condiciones de parada para posteriormente seleccionar el mejor subárbol, entendiendo como el mejor subárbol la partición del árbol de decisión que consigue un error más bajo en la muestra de entrenamiento.

Cuando se tiene creado y ajustado el árbol de decisión, se procede a la realización de la predicción sobre el conjunto de datos de validación. Para Árboles de regresión el valor de la predicción se corresponde con la media de la variable cuantitativa y para los Árboles de clasificación con la moda de la variable cualitativa. Los Árboles de decisión suelen funcionar mejor con el método de clasificación, y en este caso puede aportarse el porcentaje de cada clase en el nodo terminal que puede tomarse como la probabilidad estimada de pertenencia a cada clase, proporcionando información acerca de la confianza de predicción.

## 3.3. Modelos Bagging y Boosting

Los árboles de decisión suelen presentar problemas con el equilibrio entre el sesgo y la varianza. Por un lado, el sesgo indica la diferencia en promedio entre las predicciones y los valores reales, mientras que la varianza indica la variación de las estimaciones dependiendo de la muestra empleada en el entrenamiento. Recordando la Sección 3.2, a medida que se trabaja con árboles más ramificados la complejidad del modelo aumenta, se dispone de menos sesgo debido a que se asemeja más al conjunto de entrenamiento. Sin embargo, se tiene una mayor varianza ya que el árbol funciona mal ante futuras predicciones (sobreajuste). Construir árboles más sencillos tampoco es una opción, puesto que no se representa bien la combinación entre variables, produciendo mucho sesgo pues las predicciones no serán parecidas al conjunto de entrenamiento, y poca varianza al tratarse de una estructura tan sencilla. En la Figura 3.4, se explica gráficamente lo comentado:

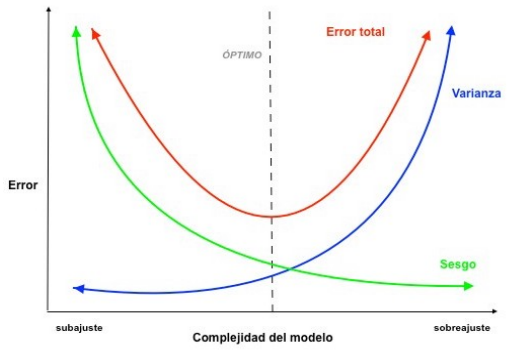



Figura 3.3: Comportamiento del sesgo y la varianza.

A lo largo de esta sección se desarrollarán métodos llamados *ensemble*, basados en la creación de múltiples árboles para reducir el error del modelo, es decir, encontrar un equilibrio entre el sesgo y la varianza.

Los métodos combinados (*métodos de ensemble*) utilizan múltiples algoritmos de aprendizaje para obtener un rendimiento predictivo que mejore el que podría obtenerse por medio de cualquiera de los algoritmos de aprendizaje individuales que lo constituyen, evidentemente, esto produce una elevación en los costes computacionales. Este término de "*ensemble*" se suele reservar para aquellas combinaciones que hacen uso de modelos pertenecientes a una misma familia, mientras que se usa el término más general de "*sistemas de aprendizaje múltiples*" cuando las hipótesis que se combinan provienen de diversas familias. Este trabajo se centra sólo en los métodos del primer caso, concretando para la familia de Árboles de decisión.

Los métodos más extendidos y comunes son los métodos de agregación *Bootstrap*, conocidos como métodos *Bagging*, y los métodos *Boosting*. Aunque estos dos métodos presentan la misma finalidad, trabajan de manera prácticamente opuesta.

En el método *Bagging*, se emplean modelos con muy poco sesgo pero mucha varianza, agregando muchos de estos modelos se consigue reducir la varianza sin apenas inflar el sesgo. En cambio, en el método *Boosting* se emplean modelos con muy poca varianza pero mucho sesgo, ajustando secuencialmente muchos modelos se reduce el sesgo. A continuación, se muestran cada uno de los modelos, con más detalle.

### 3.3.1. Bagging

La técnica *Bagging* fue introducida por Breiman en 1996, es un método de agregación de modelos de la misma familia basado en el promedio. Así como *Bootstrap* es una forma de evaluar la precisión de un estimador o una predicción, el método *Bagging* utiliza remuestreo *Bootstrap* para mejorar la estimación de la predicción en sí.

Se considera primero el caso particular del problema de regresión, para ello se denota un modelo de Árboles de regresión con datos de entrenamiento $Z = \{(X_1, Y_1), \cdots, (X_N, Y_N)\}$ y con función de predicción $\hat{f}(X)$. Entonces la técnica *Bagging* promedia $\hat{f}(X)$ sobre una colección de muestras independientes *Bootstrap*, reduciendo así su varianza. De esta forma, se tiene que para cada muestra *Bootstrap* $Z_b^*$ con $b = 1, \cdots, B$, se ajusta el modelo generando la predicción $\hat{f}_b^*(X)$. La estimación *Bagging* se define como:

$$\hat{f}_{bag}(X) = \frac{1}{B} \sum_{b=1}^{B} \hat{f}_b^*(X) \tag{3.17}$$

Así, se puede minimizar la varianza y aumentar el poder de predicción separando el conjunto de entrenamiento en B muestras *Bootstrap* y, con cada uno de estos subconjuntos construir un árbol para luego promediar el resultado de predicción en todos los árboles construidos.

Si por lo contrario se plantea un problema de clasificación, es necesario definir un clasificador $\hat{G}(X)$ para una respuesta de clase $k$. Considerando $\hat{p}_k^*$ la proporción de observaciones que se predicen con clase k (revisar la Subsección 3.2.2). Entonces la estimación *Bagging* $\hat{f}_{bag}(X)$ es un K-vector de la forma $[\hat{p}_1^*(X), \hat{p}_2^*(X), \cdots, \hat{p}_K^*(X)]$. En este caso, el clasificador *Bagging* escoge como predicción final para dicha observación, la clase común de entre las $B$ predicciones, siendo $\hat{G}_{bag}(X) = arg\ max_k\ \hat{p}_k^*(X)$.

Este método reduce el error al combinar varios árboles con alta varianza, ya que el error promedio que obtiene sobre la muestra de entrenamiento $Z$ es mayor o igual al error obtenido por el estimador *Bagging*.

A continuación, se resume el algoritmo *Bagging*:

1. Se construyen B muestras *Bootstrap* $Z_1^*, \cdots, Z_B^*$ independientes por técnicas de remuestreo.

2. Se crean de manera independiente $B$ árboles, uno para cada una de las muestras, que crecerán sin condiciones de parada y por tanto, sin someterse a poda, presentando poco sesgo y mucha varianza.

3. Se realiza la predicción individual de cada uno de los B árboles creados, es decir los estimadores $\hat{f}^{*1}(X), \cdots, \hat{f}^{*B}(X)$.

4. Se calcula el estimador *Bagging* $\hat{f}_{bag}(X)$ para regresión o $\hat{G}_{bag}(X)$ para clasificación, obteniendo la media de las B predicciones en el caso de variables continuas o la moda de las B predicciones en variables categóricas (que como en los Árboles de decisión, proporcionan una probabilidad numérica para realizar la predicción).

La Figura 3.5 nos muestra un ejemplo práctico de como funciona el procedimiento *Bagging*.

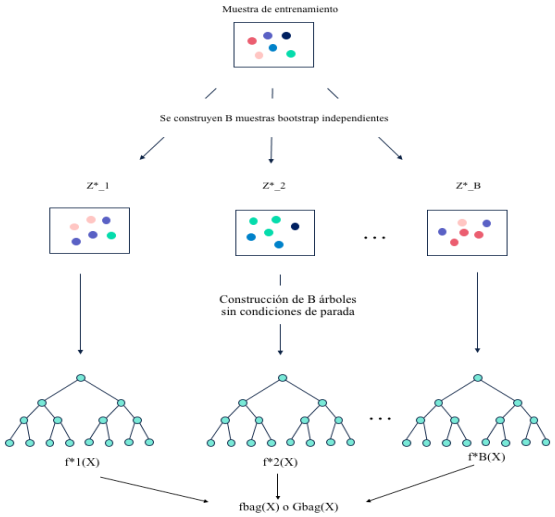



Figura 3.4: Procedimiento del método *Bagging* .

### Out-of-Bag Error

Debido a la naturaleza del proceso, es posible estimar de forma directa el *test error* sin necesidad de recurrir a validación cruzada (del inglés, *cross-validation*). El hecho de que los árboles se ajusten de forma repetida empleando muestras generadas por *Bootstrapping* conlleva que, en promedio según Breiman, Leo (2001), cada una de las B muestras ajustadas estén formadas por aproximadamente 2/3 de las observaciones originales. Al tercio restante se le llama *out-of-bag (OOB)*. Este resultado viene de tomar el límite:

$$\lim_{N\to\infty}\left(1-\frac{1}{N}\right)^{N}=e^{-1}\approx 0.36$$

pues $(1-\frac{1}{N})^N$ es la probabilidad de que un dato no se seleccione para una muesta *Bootstrap* de tamaño $N$ cuando $N\to\infty$ ya que, a cada elección con reemplazamiento la probabilidad de seleccionar un dato concreto es claramente $(1-\frac{1}{N})$, de donde al repetir de manera independiente $N$ veces se obtiene el resultado.

Si para cada árbol ajustado en el proceso *Bagging* se registran las observaciones empleadas, se puede predecir la respuesta de la observación $i$ haciendo uso de aquellos árboles en los que la observación ha sido excluida *(OOB)*. Siguiendo este proceso, se pueden obtener las predicciones para las N observaciones y con ellas calcular el *OOB-mean square error o OOB-classification error*, es decir, con las observaciones que no se utilizaron para la construcción de un árbol se puede aprovechar para realizar una validación interna de éste. Esto ofrece la posibilidad de poder ajustar la estructura del árbol por medio de hiperparámetros sin necesidad de realizar un conjunto de validación externo.

### Importancia de las variables

Aunque el proceso *Bagging* consigue mejorar la capacidad predictiva en comparación con los modelos de Árbol de decisión, la interpretación del modelo se reduce, esto se debe a la combinación de varios árboles sin poder obtener una gráfica sencilla del modelo, causando que no sea inmediato identificar de forma visual que predictores son más importantes.

Existen diferentes estrategias para cuantificar la importancia de las variables, las más utilizadas para modelos *Bagging* y también para modelos *Boosting*, explicados en la Sección 3.4, pueden ser:

- **Para regresión**:
  - **Incremento de pureza de nodos (IncNodePurity[4]).**
    En cada partición de cada árbol, se cuantifica el incremento total en la pureza de los nodos

[4]Entre paréntesis, los nombres de la rutina de las librerías de `R`, como `randomForest`.

debido a divisiones en las que participa el predictor. En cada división de los árboles, se registra la reducción en la medida de impureza empleada (MSE)[5] cuando se seleccionó la variable $X_i$, $i \in \{1, \cdots, p\}$. Para cada una de las variables, se calcula el promedio de las reducciones conseguidas en el conjunto total de los B árboles. De forma evidente, cuanto mayor sea la reducción total de error cuadrático medio que ha conseguido cada variable en el conjunto de todos los árboles, más importante será.

- **Incremento del error cuadrático medio (%IncMSE) empleando *OOB*.**
  Esta medida utiliza una permutación en el cálculo del *Mean Squared Error* para valorar la influencia que tiene cada variable. Se calcula siguiendo los siguientes pasos:
  1. Para cada árbol $b$ ajustado por el modelo, con $b = 1, \cdots, B$, se calcula el $OOBmse$ como la media de las desviaciones cuadráticas de la variable respuesta $Y_i$ de los datos que no se utilizaron en la creación del árbol *(OOB)* con respecto a sus predicciones:

  $$OOBmse_b = \frac{1}{N_{OOB,b}} \sum_{i=1}^{N} (Y_i - \hat{f}_b^*(X_i))^2$$

  Si este cálculo se realiza haciendo una transformación en los datos aleatorizando los valores de una de las variables de entrada, se producirá una modificación en el valor de las predicciones $\hat{f}_b^*(X_i)$ y, en consecuencia, una variación en la medida del error, que será mayor cuanto más importante sea dicha variable.

  2. Se repite este cálculo para cada árbol y cada variable de entrada, cada vez realizando una aleatorización en los valores de una de dichas variables de entrada:

  $$OOBmse_b(X_i^*) = \frac{1}{N_{OOB,b}} \sum_{i=1}^{N} (Y_i - \hat{f}_b^*(X_i^*))^2$$

  3. Tras esto, para cada variable $X_i^*$[6] en cada árbol $b$, se calcula el incremento del error debido a la aleatorización de los valores del predictor:

  $$\%IncMSE = \frac{(OOBmse_b(X_i^*) - OOBmse_b)}{OOBmse_b} \times 100$$

  De forma que cuanto mayor sea el incremento de cada variable, más importante será. Este proceso se repite con todas las variables y luego éstas se ordenan de acuerdo a los cambios que produjeron cada una en los errores.

- **Para clasificación**:
  - **Incremento de pureza de nodos (*MeanDecreaseGini*).**
    De la misma forma que en regresión, pero se calcula cuantificando la disminución promedio de la medida de impureza empleada como criterio de división, *Índice de Gini* o *Entropía*, que ha conseguido el predictor en el conjunto de todos los árboles que forman el modelo.
  - **Incremento de la tasa de error (*MeanDecreaseAccuracy*) empleando *OOB*.**
    Se calcula de la misma manera que %IncMSE pero con la tasa de error de clasificación en el resultado de cada división.

El cálculo %IncMSE se considera la mejor medida de la importancia de las variables según diversos autores, para más información ver Grömping, Ulrike (2009). La utilización de estas estrategias, y otras no nombradas, dependen de la utilización del paquete de `R` para la realización del modelo. La parte práctica de este trabajo se realizó con la librería `h2o`, que utiliza como método predeterminado para calcular la importancia relativa de cada variable explicativa del modelo, el incremento de pureza de nodos sin emplear la medida *out-of-bag*.

Para más información acerca de los métodos Bagging se recomienda ver en el Capítulo 8 de Gareth James; Trevor Hastie; Robert Tibshirani; Daniela Witten (2013).

[5] En el resto de trabajo, se denotará por MSE, al error cuadrático medio (del ingés, *Mean Square Error*).
[6] La notación $X_i^*$ con $i \in \{1, \cdots, p\}$, se corresponde a la variable $X_i$ tras aleatorizar sus valores.

### 3.3.2. Random Forest

Breiman en 2001 propone el algoritmo *Random Forest*, árboles aleatorios, que combina las técnicas CART y *Bagging*. Tiene como propósito incorporar la aleatoriedad en las distintas etapas de la construcción de un árbol obtenido por CART.

Este método consigue mejores resultados que el método *Bagging*, ya que decorrelaciona los árboles generados. Recordando que el proceso *Bagging* se basa en el hecho de que promediando un conjunto de modelos se consigue reducir la varianza, esto no siempre es cierto, ya que si los modelos agregados están correlacionados, la reducción de la varianza que se puede lograr es muy pequeña.

Si se considera un conjunto de datos con un predictor muy influyente en comparación con los otros predictores, todos o casi todos los árboles creados por el método *Bagging* estarán dominados por el mismo predictor y serán muy parecidos entre ellos, debido a esta alta correlación, apenas se conseguirá disminuir la varianza ni mejorar el modelo. *Random Forest* evita este problema realizando una selección aleatoria de $m$ predictores antes de realizar cada división. Entonces, la mayoría de divisiones no contemplará el predictor influyente permitiendo que otros puedan ser seleccionados y consiguiendo reducir la varianza.

El algoritmo de *Random Forest* es igual al algoritmo *Bagging*, con la pequeña diferencia de que antes de realizar cada división se seleccionan aleatoriamente $m$ predictores. El resultado dependerá del valor $m$; si $m = p$ entonces el resultado será el mismo al obtenido por el método *Bagging*. Se recomienda utilizar, $m \simeq \sqrt{p}$ siendo $p$ número de predictores totales, aunque la mejor forma sería evaluar el *out-og-bag* para diferentes valores de $m$. Con este proceso, *Random Forest* permite romper la correlación entre los árboles intermedios ayudando a tener predicciones más fiables.

A continuación, se muestra el algoritmo correspondiente al *Random Forest* tanto para regresión como clasificación:

1. Para $b = 1, \cdots, B$:

   *a*) Se construyen B muestras *Bootstrap* $Z_1^*, \cdots, Z_B^*$ independientes por técnicas de remuestreo.

   *b*) Se crean de manera independiente $B$ árboles *Random-Forest*, uno para cada una de las muestras, repitiendo recursivamente los siguientes pasos para cada nodo del árbol hasta alcanzar el tamaño mínimo de nodo $n_{min}$.

   - Se seleccionan $m$ variables al azar de las $p$ variables posibles.
   - Se elije la mejor variable y el mejor punto de división entre los $m$.
   - Se divide el nodo en dos regiones.

2. Se realiza la predicción individual de cada uno de los $B$ árboles creados, es decir los estimadores $\hat{f}_1^*(X), \cdots, \hat{f}_b^*(X)$.

Se calcula el estimador de predicción *Bagging* para el *Random Forest*:

Regresión: $\hat{f}_{\text{rf}}(X) = \dfrac{1}{B}\displaystyle\sum_{b=1}^{B} \hat{f}_B^*(X)$

Classificación: Sea $\hat{p}_k^*$ proporción de observaciones que predicen cada árbol *Random Forest* con clase k. Entonces $\hat{G}_{\text{rf}}(X) = arg\ max_k\ \hat{p}_k^*(X)$.

Gráficamente, la idea del método *Random Forest* se muestra en la Figura 3.6.

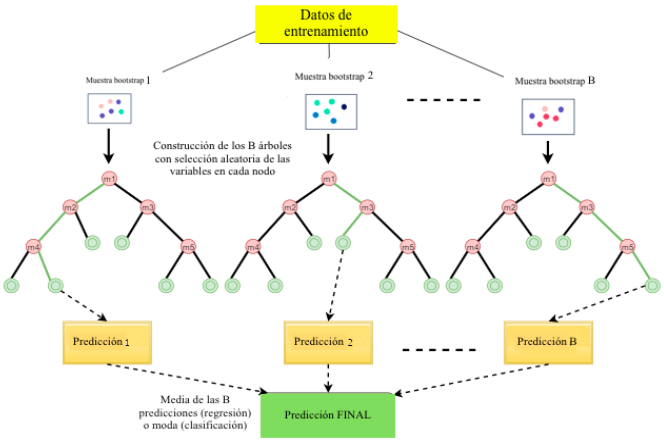



Figura 3.5: Procedimiento del Random Forest .

En este marco, veremos que ocurre con la varianza cuando aumenta el número de árboles suponiendo varianzas iguales $\sigma^2$ con correlación $\rho$. La varianza del estimador *Random Forest* será:

$$Var\left(\frac{1}{B}\sum_{i=1}^{B}\hat{f}^{*i}(X)\right) = \frac{1}{B^2}\sum_{i=1}^{B}\sum_{j=1}^{B}Cov(\hat{f}^{*i}(X),\hat{f}^{*j}(X)) = \frac{1}{B^2}\sum_{i=1}^{B}\sum_{j\neq i}^{B}Cov(\hat{f}^{*i}(X),\hat{f}^{*j}(X))+Var(\hat{f}^{*i}(X)))$$

donde sustituyendo,

$$Var\left(\frac{1}{B}\sum_{i=1}^{B}\hat{f}^{*i}(X)\right) = \frac{1}{B^2}\sum_{i=1}^{B}((B-1)\rho\sigma^2+\sigma^2) = \frac{\sigma^2\rho(B-1)}{B}+\frac{\sigma^2}{B} = \rho\sigma^2+\frac{\sigma^2(1-\rho)}{B}$$

Cuando $B\to\infty$ la varianza del estimador solo depende de $\rho\sigma^2$. Si la correlación es alta, como en los métodos *Bagging* cuando hay variables cuya importancia es muy superior al resto, el valor de dicha cantidad será alto indicando alta variabilidad. Mientras que, en el *Random Forest* se disminuye el número de variables candidatas a considerar en una división, y de este modo, al aleatorizar más la selección de variables, el valor de $\rho$ será menor y consecuentemente se reducirá la varianza en virtud de la ecuación anterior.

El *Random Forest* es uno de los algoritmos con mejores resultados en los problemas de aprendizaje, en particular, en aquellos que cuentan con una cantidad importante de variables explicativas. Sin embargo si sólo algunas pocas son de relevancia, el *Boosting* suele ser más eficaz. Para ver más información acerca del proceso y características del *Random Forest* se recomienda ver el Capítulo 15 de Trevor Hastie, Robert Tibshirani, Jerome Friedman (2013).

### Características del Random Forest

A continuación, se presentan algunas características del *Random Forest* así como la selección adecuada de los hiperparámetros.

- El *Random Forest* proporciona su propio estimador *out-of-bag (OOB)*, midiendo tanto el error como la correlación e importancia de variables sin necesidad de utilizar medidas de validación externas.
- Es computacionalmente más rápido que las técnicas *Bagging* y *Boosting* (ver Sección 3.4).
- La precisión del algoritmo es bueno para grandes bases de datos y en ocasiones mejor que algunas técnicas *Boosting*.
- Es robusto a variables de ruido y datos atípicos.
- Aporta estimaciones de qué variables son importantes en la clasificación.

Para este método es muy importante ajustar correctamente los valores prefijados (hiperparámetros) antes de la construcción del árbol. Los hiperparámetros más importantes para ajustar son el número de variables candidatas a evaluar en cada ramificación, y el número de árboles individuales que forma el algoritmo. Sin embargo, existen algunos adicionales que deben considerarse:

- **Número de árboles B (ntree)**.
  Tiene efecto en la precisión de la predicción, por ello, se pretende estabilizar el error con un número óptimo de árboles pero usar demasiados produce un alto coste computacional. Por defecto, en el *Random Forest* se considera B=500.

- **Número mínimo de variables para cada ramificación** (**mtry**).
  Número de variables predictoras, $m$, seleccionados aleatoriamente en cada árbol. Al reducir el valor de `mtry` se reduce la correlación entre los árboles, debido a que en cada nodo se tienen menos posibilidades de variables candidatas para cada ramificación. Sin embargo, reducir su valor también puede reducir la precisión de cada árbol individual.
  Las recomendaciones para el valor de `mtry` son: para clasificación $\sqrt{p}$ y para regresión $p/3$. En la práctica, este hiperparámetro depende del problema. Por ello conviene, probar simulaciones de distintos valores.

- **min_rows**.
  Mínimo número de muestras dentro de los nodos terminales. Equilibrio entre sesgo-varianza. En la práctica se recomienda utilizar un tamaño pequeño, por defecto se utiliza `min_rows`=1 para clasificación y `min_rows`=5 para regresión.

- **max_depth**.
  Máxima profundidad del árbol.

De la elección aleatoria con reemplazamiento de las N observaciones que forman la muestra inicial, se suelen quedar fuera en torno al 36 % del total. El *out-of-bag* estima el error de predicción teniendo en cuenta estas observaciones. Según Breiman, Leo (2001), *OOB* tiene dependencia importante con los parámetros del modelo `ntree` y `mtry`. La influencia del valor `mtry` en el error depende del número de variables de entrada en el modelo, sin embargo, este error se reduce de forma asintótica con el número de árboles, como se puede ver en la Figura 3.7. De igual forma ocurre para el método *Bagging*.

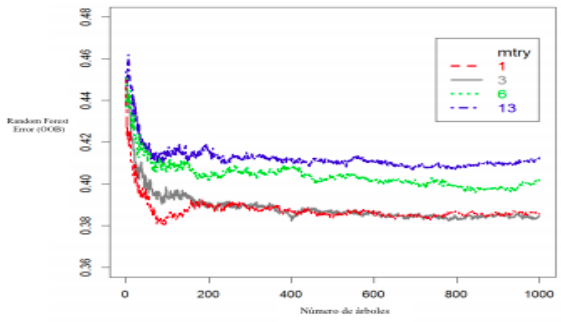



Figura 3.6: Variación del OOB con `ntree` para diferentes valores de `mtry` (ver Trevor Hastie, Robert Tibshirani, Jerome Friedman (2013)).

Es posible que en determinados casos una elección incorrecta de estos hiperparámetros, sobre todo de los fundamentales; `B` y `mtry`, produzcan un cierto sobreajuste del modelo. Esto provoca una pérdida de la precisión en las predicciones posteriores al entrenamiento, aunque está demostrado mediante la ley de los grandes números, que al aumentar el `ntree` el *out-bag-error* converge. Para más información acerca de la demostración del anterior teorema[7] así como de otras características de los hiperparámetros del algoritmo, se recomienda ver Breiman, Leo (2001).

[7]Destacar que la idea principal, es una aproximación Monte Carlo del error *Bootstrap*.

## 3.4. Boosting

La técnica de modelización *Boosting* es una de las más potentes introducidas en los últimos veinte años. Se diseñó en el año 1989 de la mano del profesor Robert Schapire, aunque un año después se mejora por Jonh E. Freund. Originalmente se creó para abordar problemas de clasificación, pero se llegó a extender también para la regresión. A diferencia del *Bagging*, no se crean versiones del conjunto de entrenamiento, sino que se trabaja secuencialmente siempre con el mismo conjunto de entrada manipulando los pesos de los datos para generar modelos distintos. Es decir, consiste en ir ajustando de manera individual B árboles, los cuáles presentan mucho sesgo pero poca varianza, de modo que cada árbol nuevo no cometa los errores del árbol anterior, y así mejorar iteración tras iteración. De forma resumida, este ajuste responde a clasificaciones individuales (*débiles*) que producen una clasificación general (*fuerte*) sin errores.

Este tipo de algoritmos requiere más hiperparámetros que los vistos anteriormente. Además, es más común que ocurra un sobreajuste si el número de árboles es excesivamente alto. En esta sección se definirá el término de regularización *learning rate* para evitar ese sobreajuste, también se estudiará los algoritmos más conocidos del *Boosting*, en los que se encuentran: el *AdaBoost*, el *Gradient Boosting* y el *XGBoost*. El primero a diferencia de los dos últimos, carece de conjuntos de validación internos (*OOB*) como ocurría con las técnicas vistas en la Subsección 3.3.1.

### 3.4.1. Adaboost

Es el algoritmo *Boosting* más utilizado para modelos de clasificación. A continuación, se describe el procedimiento principal para el caso concreto de una variable respuesta dicotómica.

Considerando la variable respuesta con dos categorías $Y \in \{0, 1\}$, $p$ variables explicativas para cada una de las $N$ observaciones, es decir $(X_i, Y_i)$ para $i = 1, \cdots, N$ con $X_i = (X_{i,1}, \cdots, X_{i,p})$, y una función de clasificación $G(X)$ que predice tomando uno de los dos valores $\{0, 1\}$, entonces la tasa de error en la muestra de entrenamiento[8] se define como:

$$err = \frac{1}{N} \sum_{i=1}^{N} I_{\{Y_i \neq G(X_i)\}}$$

y la tasa media de error en predicciones futuras como $\mathbb{E}_{XY} I_{\{Y \neq G(X)\}}$.

De esta forma se denomina un *clasificador débil* o *weak learner*, aquél cuya tasa de error es sólo ligeramente mejor que la aleatoriedad del modelo. El propósito del *AdaBoost* es aplicar secuencialmente un algoritmo de clasificación débil de forma iterativa a versiones modificadas de los datos, produciendo una secuencia de clasificadores $G_b(X)$ con $b = 1, \cdots, B$.

Cada una de las modificaciones en cada paso del algoritmo se realizan en función de los pesos $w_i$ que se asigna en el inicio a cada una de las observaciones $(X_i, Y_i)$ con $i = 1, \cdots, N$. Las predicciones de todos los $G_b$ se combinan a través de una votación ponderada, donde el peso asignado a cada clasificador depende de su tasa de error, produciendo un efecto de mayor influencia a los clasificadores más precisos.

Normalmente los Árboles de decisión se utilizan como clasificadores débiles construyendo una estructura con pocas ramificaciones, por ello, recuperando la notación utilizada hasta ahora del número de árboles ($B$), el algoritmo *AdaBoost* se estructuraría de la siguiente manera:

1. Inicialmente a todos los datos del conjunto de entrenamiento se les asigna el mismo valor/peso, $w_i = 1/N$, con $i = 1, \cdots, N$ donde N es el tamaño de la muestra.

2. Para $b = 1, \cdots, B$:

   - Se ajusta un *clasificador débil* o *weak learner*, $G_b(X)$ (Árbol de decisión), a los datos de entrenamiento usando los pesos $w_i$ en lugar de las observaciones.

[8] Definida en la Subsección 3.2.2.

- Se calcula el error cometido por el modelo, identificando las muestras mal clasificadas.

$$err_b = \frac{\sum_{i=1}^{N} w_i I_{\{Y_i \neq G_b(X_i)\}}}{\sum_{i=1}^{N} w_i}$$

- Se calcula $\alpha_b = log((1 - err_b)/err_b)$ para medir la influencia en el conjunto de forma proporcional al número de aciertos.
- Se modifican los pesos $w_i \leftarrow w_i \exp[\alpha_b I_{\{Y_i \neq G_b(X_i)\}}]$ para $i = 1, \cdots, N$ y se vuelve a ajustar el modelo.

3. El modelo final de predicción realizará una predicción ponderada sobre cada uno de los pesos de todos los modelos (árboles) que se hayan construido. De esta forma, se dará más fuerza a las predicciones más efectivas que hayan cometido un menor número de errores.

$$\hat{G}(X) = \sum_{b=1}^{B} \alpha_b G_b(X))$$

Destacar que para el árbol $b$, aquellas observaciones que son clasificadas erróneamente por el clasificador $G_{b-1}(X)$ del paso anterior, tienen sus pesos aumentados. De esta forma, mientras que los pesos se reducen para aquellas que se clasificaron correctamente, las observaciones que son difíciles de clasificar reciben una influencia cada vez mayor. La Figura 3.8 es un ejemplo gráfico del funcionamiento del algoritmo *AdaBoost*.

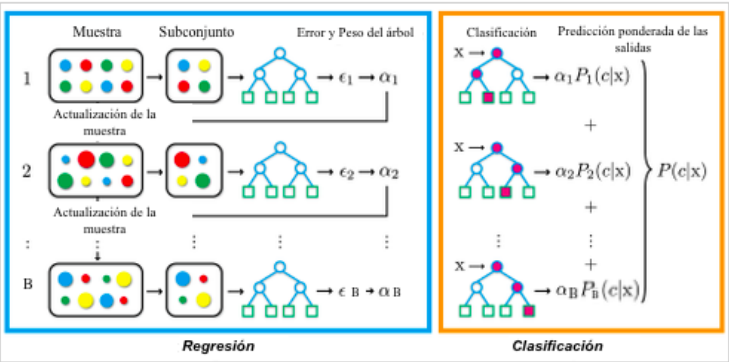



Figura 3.7: Ejemplo gráfico del funcionamiento algoritmo *AdaBoost*.

En el caso de la regresión, donde la variable dependiente es continua, se puede adaptar el clasificador $\hat{G}(X)$ de la siguiente forma:

$$\hat{f}(X) = \sum_{m=1}^{M} \alpha_b f_b(X)$$

que resulta ser un promedio ponderado de los distintos resultados obtenidos por $f_1(X), \cdots, f_B(X)$.

Para más información acerca del método *AdaBoost* se recomienda ver el Capítulo 10 de Trevor Hastie, Robert Tibshirani, Jerome Friedman (2013). Este algoritmo mejora la precisión de los métodos vistos hasta ahora, pero sacrifica rapidez, facilidad para interpretar los resultados y sensibilidad a datos de entrenamiento marcados de forma incorrecta. En la siguiente sección, se trabajará con un método nuevo *Boosting* que modela habitualmente con Árboles de decisión y mitiga los problemas que surgen con el *AdaBoost*.

### 3.4.2. Gradient Boost

Una función de predicción $f(X)$ sobre $Y$ a partir de los valores de entrada $X$, tiene asociada una función de pérdida o coste $L(Y, f(X))$, la cual penaliza los errores de predicción. Existen varias funciones de pérdida, entre ellas la función de mínimos cuadrados $L(Y, f(X)) = (Y - f(X))^2$ que es la más popular y conveniente para el análisis, la función del error absoluto $L(Y, f(X)) = |Y - f(X)|$ o la función exponencial, más robusta que las usadas comúnmente.

El método *Gradient Boosting* es una generalización del algoritmo *AdaBoost* que permite emplear cualquier función de coste siempre que sea diferenciable. Esta flexibilidad ha hecho posible aplicar *Boosting* a multitud de problemas; regresión, clasificación con más de dos clases, etc.

Muchos algoritmos de regresión, incluidos los árboles de decisión, se centran en minimizar alguna función de los residuos, más típicamente la función de pérdida de SSE (suma de los cuadrados del error residual) o de manera equivalente el MSE (error cuadrático medio) o RMSE (raíz del error cuadrático medio). Al añadir nuevos árboles, en vez de reducirse el error por ajustar los errores del árbol anterior como en el *AdaBoost*, se optimiza la función de pérdida, de forma que se detiene cuando alcanza un nivel aceptable o no mejora al aplicarse a un conjunto nuevo de datos de validación. Esta forma específica se corresponde con cómo el *Gradient Boost* minimiza secuencialmente la función de perdida de error cuadrático medio (SSE), en concreto, para la pérdida de SSE el gradiente no es más que el error residual. Sin embargo en otras ocasiones, es necesario centrarse en diferentes funciones de pérdida como el error absoluto medio (MAE) menos sensible a los valores atípicos. El nombre de *"Gradient Boosting Machine"* proviene del hecho de que este procedimiento puede generalizarse a funciones de pérdida que no sean SSE.

El *Gradient Boosting* se considera un algoritmo iterativo de *descenso de gradientes*, un método de optimización muy genérico capaz de encontrar soluciones óptimas para una amplia gama de problemas. Lo que realiza exactamente, es medir el gradiente local de la función de pérdida para un conjunto dado de parámetros ($\theta$) y desciende sobre la función de coste tomando medidas en la dirección del gradiente descendente. Como se muestra en la Figura 3.9, una vez que el gradiente es cero, se ha alcanzado un mínimo.

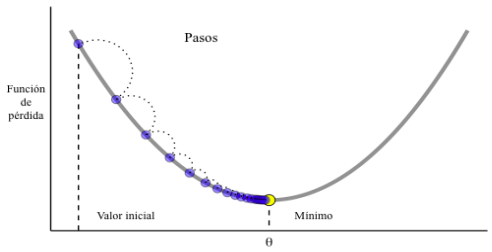



Figura 3.8: Proceso del gradiente descendente del ajuste iterativo de parámetros. (Aurélien Géron (2017)).

En consecuencia, esto permite que los GBM optimicen diferentes funciones de pérdida según lo deseado (ver página 360 de Trevor Hastie, Robert Tibshirani, Jerome Friedman (2013) para diferentes funciones de coste). Un parámetro importante en el descenso de gradiente es el tamaño de los pasos, llamado `learning rate`. Si este parámetro es demasiado pequeño, el algoritmo realizará muchas iteraciones para encontrar el mínimo; por otro lado, si el `learning rate` es demasiado alto, puede saltarse el mínimo y terminar más lejos de dónde empezó. En la Figura 3.10 se representan los dos casos:

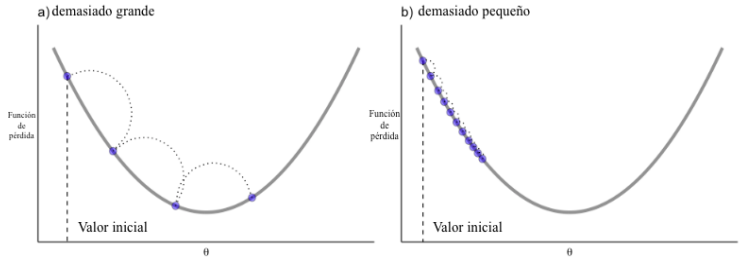



Figura 3.9: Resultado del GBM para un `learning rate` muy bajo o muy alto (Aurélien Géron (2017)).

No todas las funciones de coste son convexas (tienen forma de cuenco), puede haber mínimos locales que dificulten la búsqueda del mínimo global. El descenso de gradiente estocástico ayuda a abordar esta situación utilizando una fracción de la muestra de entrenamiento (sin reemplazamiento, generalmente). Esto provoca que el algoritmo sea más rápido, pero la naturaleza estocástica agrega aleatoriedad al realizar el descenso de gradiente de la función de pérdida que no permite al algoritmo encontrar el mínimo global absoluto, aunque es capaz de atravesar los mínimos locales para acercarse lo suficiente al mínimo global. El *Gradient Boosting* estocástico combina el *Boosting* con el *Bagging*. Este proceso se representa en la Figura 3.11:

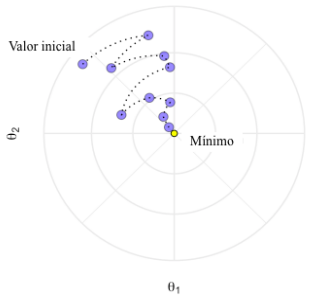



Figura 3.10: Descenso del gradiente estocástico del GBM (Aurélien Géron (2017)).

Existen varias opciones de ajuste de hiperparámetros, pero si se ajustan correctamente, los GBM pueden conducir a algunos de los modelos predictivos más flexibles y precisos que se pueden construir.

Si recordamos el modelo de Árboles de decisión en la Sección 3.2, se divide el espacio de todos los valores de las variables predictoras conjuntas en $R_j$ regiones, con $j = 1, \cdots, J$, que representan los nodos terminales. Si se asigna una constante $c_j$ como valor estimado para cada región, la predicción sería:

$$X \in R_j \rightarrow f(X) = c_j$$

Formalmente un árbol se puede expresar como:

$$T(x;\theta) = \sum_{j=1}^{J} c_j I_{\{x \in R_j\}}$$

con parámetros $\theta = \{R_j, \gamma_j\}_1^J$, y $\gamma_j$ se encuentra minimizando:

$$\hat{\theta} = argmin_\theta \sum_{j=1}^{J} \sum_{X \in R_j} L(Y_i, \gamma_j)$$

Ante el coste computacional para hallar estos parámetros, normalmente se resuelve con soluciones subóptimas aproximadas; se divide el problema en dos partes y la búsqueda de esos parámetros óptimos se realiza por separado (pág 356 del Capítulo 10 de Trevor Hastie, Robert Tibshirani, Jerome Friedman (2013)).

El *Boosting* de árboles se puede expresar como una suma de árboles:

$$f_B(X) = \sum_{b=1}^{B} T(X; \theta_b)$$

luego, en cada iteración el objetivo es resolver es:

$$\hat{\theta}_b = argmin_{\theta_b} \sum_{i=1}^{N} L(Y_i, f_{b-1}(X_i) + T(X_i, \theta_b))$$

para la constante $\theta_b = \{R_{jb}, \gamma_{jb}\}_1^{Jb}$ del siguiente árbol al modelo actual $f_{b-1}$.
Por el otro lado, dadas las regiones $R_{jb}$ encontrar el parámetro óptimo $\gamma_{jb}$ de cada región es sencillo:

$$\hat{\gamma}_{jb} = \arg\min_{\gamma_{jb}} \sum_{X_i \in R_{jB}} L\left(Y_i, f_{b-1}\left(X_i\right) + \gamma_{jb}\right)$$

la dificultad aparece en encontrar esas regiones, y aún más si se tratan de varios árboles.

Para solventar la dificultad de minimizar $L(f) = \sum_{i=1}^{N} L(Y_i, f(X_i))$, el GBM toma el gradiente de $L(f)$ por el método de descenso de gradiente, como sólo se puede calcular sobre los datos de entrenamiento, se construye el algoritmo de forma que los árboles se aproximen al gradiente negativo. De esta forma el algoritmo para problemas de regresión sería:

1. Se inicializa $f_0(X) = argmin_\gamma \sum_{i=1}^{N} L(Y_i, \gamma)$
2. Para $b = 1, \cdots, B$

   *a*) Para $i = 1, \cdots, N$ se calcula
   $$r_{ib} = -[\frac{\partial L(Y_i, f(X_i))}{\partial f(X_i)}]_{f=f_{b-1}}$$

   *b*) Se ajusta un árbol de regresión a la salida de $r_{ib}$ devolviendo regiones terminales $R_{bj}$ con $j = 1, \cdots, J_b$.

   *c*) Para $j = 1, \cdots, J_b$ se calcula
   $$\gamma_{jb} = argmin_\gamma \sum_{X_i \in R_{jb}} L(Y_i, f_{b-1}(X_i) + \gamma)$$

   *d*) Se actualiza $f_b(X) = f_{b-1}(X) + \sum_{j=1}^{J_b} \gamma_{jb} I_{\{X \in R_{jb}\}}$.

3. Se obtiene el valor $\hat{f}(X) = f_B(x)$.

Debido a su fácil interpretación, se ha explicado el algoritmo para regresión, el algoritmo de clasificación es similar; los pasos (2)-(3) se repiten para K-clases en cada iteración, calculando $r_{ikb} = Y_i - p_k(X_i)$ con $p_k(X) = \frac{e^{f_k(X)}}{\sum_{l=1}^{K} e^{f_l}}$ función de clasificación. En el paso 3, se obtiene $f_{kB}(X)$ para $k = 1, \cdots, K$, diferentes expansiones del árbol. Para más información se recomienda ver Malohlava, Michal and Candel, Arno (2017).

### Selección de hiperparámetros

Un modelo GBM simple contiene tres categorías de hiperparámetros en la implementación `h2o`:

- **Hiperparámetros *Boosting*.**
  - **Número de árboles B (ntree).**
    El número de árboles que crecen de forma independiente en el *Bagging* y *Random Forest* hace que sea muy difícil realizar el modelo con un valor alto de `ntree`. Sin embargo, en los GBM esto no sucede; suele depender de los valores de los otros hiperparámetros y normalmente se requieren valores altos aunque pueden sobreajustarse fácilmente.
  - **Learning rate.**
    Determina cuanto influye cada árbol individual en el resultado final y controla la rapidez con la que el algoritmo avanza por el descenso del gradiente.

    Los valores oscilan entre 0 y 1, suelen emplearse valores inferiores a 0.1. Cuanto más pequeño es, más lentamente aprende el modelo y más árboles se necesitan, sin embargo, reduce el riesgo de sobreajuste.
- **Hiperparámetros específicos del árbol**.

  - **Profundidad (max_depth).**
    Controla la profundidad de los árboles individuales. Los valores típicos varían desde una profundidad de 3 a 10 pero no es raro ver una profundidad de árbol de 1. Los árboles de menor profundidad son computacionalmente eficientes (pero requieren más árboles); sin embargo, los árboles de mayor profundidad permiten que el algoritmo capture interacciones únicas pero también aumenta el riesgo de un sobreajuste.
  - **Número mínimo de observaciones en los nodos (min_rows)**.
    Controlan la complejidad de cada árbol. Valores altos ayudan a evitar que un modelo se sobreajuste, pero los valores más pequeños pueden ayudar con clases desequilibradas en problemas de clasificación.

- **Hiperparámetros del comportamiento estocástico (reducen el sobreajuste)**.
  - **sample_rate**.
    Porcentaje de observaciones aleatorias empleadas para el ajuste de cada árbol.
  - **col_sample_rate**.
    Porcentaje de predictores (columnas) seleccionadas de forma aleatoria para el ajuste de cada árbol.

#### Características del GBM

Cómo hemos visto hasta ahora existen diferencias notables entre los métodos *Bagging* y *Boosting*, por ello, se destaca a continuación las principales características del GBM.

- Construye modelos de clasificación o regresión secuencialmente que considera "modelos débiles", optimizando una función de pérdida diferenciable del modelo general. A medida que se agrega un modelo débil en el algoritmo, se ajusta uno nuevo para proporcionar una estimación más precisa.
- Se caracteriza por que los modelos débiles actualizados están correlacionados al máximo con el gradiente negativo de la función de pérdida.
- Debido a que es aplicable a muchas funciones de pérdida, y optimiza la precisión de predicción de esas funciones, posee una ventaja ante los métodos convencionales, siendo más genérico.
- Aborda el problema de la multicolinealidad.
- Computacionalmente es más lento que otros modelos, ya que en cada iteración se analiza un modelo débil, pudiendo acumular errores en el modelo actualizado.

Una de las desventajas del GBM es la tendencia al sobreajuste, para intentar solventar este problema también se realizará un modelo de propensión con la técnica *XGBoost*, explicada en la siguiente sección.

Esta sección se ha realizado con ayuda del Capítulo 10 del libro Trevor Hastie, Robert Tibshirani, Jerome Friedman (2013).

### 3.4.3. Extreme Gradient Boosting

El *XGBoost* o *Extreme Gradient Boosting* es una de las implementaciones más populares y eficientes del algoritmo *Boosting*. Como se ha visto hasta ahora, los algoritmos *Boosting* construyen un modelo predictivo a partir de la combinación de modelos de predicción débiles (Árboles de decisión). Los Árboles de decisión se construyen de manera secuencial suavizando las carencias del árbol anterior, y por tanto, dando lugar a un modelo más eficiente.

La principal diferencia entre los distintos modelos *Boosting*, recae en cómo los algoritmos identifican las deficiencias de los árboles de decisión. En concreto, el método *AdaBoost*, explicado en la Subsección 3.4.1, identifica los defectos mediante clasificaciones de versiones modificadas de los datos, dando más fuerza a los modelos que hayan cometido un menor número de errores. Por el contrario, el método *Gradient Boosting*, explicado en Subsección 3.4.2, proporciona un enfoque concreto para optimizar una función de pérdida diferenciable. De tal forma, que se identifican las deficiencias de los árboles de decisión por gradientes, realizando una selección iterativa de modelos que apuntan en la dirección del gradiente negativo.

El *XGBoost* como el GBM, considera como función objetivo, la función de pérdida, la cual penaliza los errores de predicción. Pero en este algoritmo se agregan divisiones adicionales dentro de cada árbol creado, de manera que la función objetivo se reduce. Además a diferencia del resto de modelos, cada nodo tiene asociado una puntuación. Las puntuaciones de predicción de cada árbol individual se suman para obtener la puntuación final, de manera que los árboles se complementan entre ellos para obtener un modelo con una estructura más eficiente. A continuación, se profundiza en esta idea.

Para entrenar el modelo el *XGBoost*, como el resto de modelos *Boosting*, utiliza una estrategia aditiva fijando aquello que se aprende y añadiendo un nuevo árbol de cada vez. De tal forma, que se puede escribir el valor de predicción en cada paso de la siguiente manera:

$$\begin{aligned}
&\hat{f}_0(X_i) = 0 \\
&\hat{f}_1(X_i) = f_1(X_i) = \hat{f}_0(X_i) + f_1(X_i) \\
&\vdots \\
&\hat{f}_B(X_i) = \sum_{b=1}^{B} f_b(X_i) = \hat{f}_{b-1}(X_i) + f_b(X_i)
\end{aligned}$$

En cada uno de los pasos se irá añadiendo el árbol que mejor optimice la función objetivo. De esta manera, el *XGBoost* encuentra el estimador del árbol óptimo $f_b$, para mejorar la división los árboles existentes $(b-1)$ y corregir el error hasta el momento. Además, añade una función de regularización más compleja que otros modelos para evitar la dependencia estadística entre los árboles que conlleva al sobreajuste. De tal forma, se define:

$$L^{(b)}(f) = \sum_{i=1}^{N} L(Y_i, \hat{f}_b(X_i)) + \sum_{b=1}^{B} \Omega(f_b)$$

siendo $f_{b-1}(X_i)$ la salida de combinaciones de los otros $(b-1)$ árboles en la observación $i$ y $\Omega(f_b)$ la complejidad del árbol.

Suponiendo que la función es convexa y dos veces diferenciable se puede utilizar una aproximación de orden dos para resolver:

$$L^{(b)} \approx \sum_{i=1}^{N} \left[ L\left(Y_i, \hat{f}_{b-1}(X_i)\right) + g_i f_b(X_i) + \frac{1}{2} h_i f_b(X_i)^2 \right] + \sum_{b=1}^{B} \Omega(f_b)$$

donde

$$g_i = \frac{\partial}{\partial z} L(Y_i, z)\Big|_{z=\hat{f}_{b-1}(X_i)} \qquad h_i = \frac{\partial^2}{\partial z^2} L(Y_i, z)\Big|_{z=\hat{f}_{b-1}(X_i)}$$

son gradientes de primer y segundo orden (derivadas parciales) del objetivo local evaluado en $\hat{f}_{b-1}(X_i)$. La aproximación de segundo orden es una optimización más precisa que el gradiente descendente del GBM, en el sentido de que el árbol óptimo que entra en el algoritmo minimiza la aproximación y además también tiende a minimizar la función original $L^{(b)}$.

A continuación, se muestra el resultado de la función objetivo simplificado para el *XGBoost*:

$$\tilde{L}^{(b)} = \sum_{i=1}^{N} \left[ g_i f_b(X_i) + \frac{1}{2} h_i f_b(X_i)^2 \right] + \Omega(f_b)$$

Si se substituye el valor de la complejidad del árbol $\Omega(f_b) = \gamma B + (1/2)\lambda \|w\|^2$ en la ecuación anterior, se obtiene que el valor objetivo para el árbol b-ésimo sería:

$$\tilde{L}^{(b)} = \sum_{b=1}^{B} \left[ \left( \sum_{i \in I_b} g_i \right) w_b + \frac{1}{2} \left( \sum_{i \in I_b} h_i + \lambda \right) w_b^2 \right] + \gamma B$$

La variable $w_b$ representa la puntuación en cada nodo y se elige considerando la simplificación de la función objetivo en términos de los dos gradientes, de tal forma que su valor óptimo es el siguiente:

$$w_b^* = -\frac{\sum_{i \in I_b, g_i} g_i}{\sum_{i \in I_b} h_i + \lambda}$$

En conclusión la estructura de los árboles en el *XGBoost* se determina mediante:

$$Ganancia \rightarrow \frac{1}{2} \left[ \frac{G_L^2}{H_L + \lambda} + \frac{G_R^2}{H_R + \lambda} + \frac{(G_L + G_R)^2}{H_L + H_R + \lambda} \right] - \gamma$$

Con esta fórmula se comprueba si es positiva la división de una nodo en dos nuevos nodos. Esta fórmula tiene cuatro partes: la primera parte de la fórmula es la puntuación en el nuevo nodo de la izquierda (L), la segunda parte es la puntuación en el nuevo nodo de la derecha (R), la tercera parte es la puntuación en el nodo original y $\gamma$ es la regularización en el nodo adicional. Podemos comprobar que si la ganancia es más pequeña que $\gamma$ no se

debería añadir una nueva rama.

La base de este método fue implementada para soportar múltiples estrategias de construcción de árboles. La comparación está en la manera de elegir los nodos para las próximas divisiones. Con otros modelos más comunes, el árbol siempre crecía eligiendo un nodo en el nivel de poca profundidad, dividiendo los nodos de un nivel antes de dividir los nodos del nivel inferior. Esta estrategia recibe el nombre de `depth-wise` y suele ofrecer un árbol equilibrado.

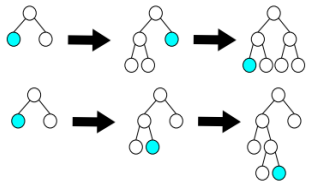


Figura 3.11: Estrategia de profundidad (arriba) y estrategia de pérdida óptima (abajo). Fuente: Cho, Hyunsu. (2016).

El *XGBoost* ofrece una estrategia diferente, se conoce como `lossguide`, pérdida óptima, eligiendo un nodo de cualquier nivel, aumentando la precisión del modelo. La estrategia a menudo produce una mayor reducción de pérdidas, a costa de producir un árbol desequilibrado.

### Selección de hiperparámetros

A diferencia del GBM, el *XGBoost* requiere de más hiperparámetros. A continuación se citan los más importantes:

- **Profundidad máxima (max_depth)**.
  Es el número máximo de nodos de bifurcación de los árboles de decisión usados en el entrenamiento. Los árboles más profundos pueden modelar relaciones más complejas, pero a medida que se profundiza, las divisiones se vuelven menos relevantes y a veces solo se deben al ruido, lo que hace que el modelo se sobreajuste.
  Su valor está entre $[0, \infty]$, y viene por defecto, como 6.
- **Submuestra de las columnas**
  - **colsample_bytree**.
    Se corresponde con la fracción de características (las columnas) a utilizar. De forma predeterminada se establece en 1.
  - **colsample_bylevel**.
    Proporción de submuestra de columnas para cada nivel. El submuestreo se realiza al alcanzar un nuevo nivel de profundidad en un árbol. Por defecto toma el valor 1.
- **Tasa de aprendizaje (eta)**.
  Define la cantidad de "corrección"que se realiza en cada iteración, reduciendo el peso para que el proceso de *Boosting* sea más eficaz, y no se sobreajuste.
  En la práctica su valor se encuentra en el intervalo [0,1]. Un valor alto llega más rápido al mínimo de la función objetivo, es decir, a un modelo mejor, pero puede sobrepasar su valor óptimo. En cambio, un valor pequeño puede que no llegué al óptimo incluso después de muchas iteraciones. Por defecto, se establece como 0.3.
- **Reducción de pérdida mínima (gamma)**.
  Reducción de pérdida mínima requerida para realizar una partición adicional en un nodo del árbol. Su valor oscila en el intervalo $[0, \infty]$, por defecto es 0.
- **Suma mínima del peso (min_child_weight)**.
  Criterio de parada para continuar realizando divisiones adicionales de los nodos, al alcanzar cierto grado de pureza. Oscila en el intervalo $[0, \infty]$ aunque por defecto es 1.
- **Peso delta máximo (max_delta_step)**.
  Por lo general este parámetro no es necesario, pero podría ayudar en la regresión logística cuando la clase está extremadamente desequilibrada. Establecerlo en un valor de 1-10 (ya que oscila entre $[0, \infty]$) podría resultar eficiente.

- **Submuestra (subsample)**.
  Proporción de submuestra de las observaciones de entrenamiento. Con 0.5 el *XGBoost* muestreará al azar la mitad de los datos de entrenamiento evitando el sobreajuste. Se produce una vez en cada iteración del *Boosting*. Puede tener los valores entre $(0, 1]$ aunque viene predeterminado como 1.
- **Términos de regularización**
  - **lambda o reg_lambda**.
    Término de regularización L2 sobre los pesos. Aumentar este valor hará que el modelo sea más conservador. Predeterminado como 1.
  - **alpha o reg_alpha**.
    Término de regularización L1 sobre los pesos de las observaciones. Aumentar este valor hará que el modelo sea más conservador. Predeterminado como 0.
- **Estrategia de crecimiento grow_policy**.
  Por defecto `depthwise` pero puede utilizarse `lossguide`. Parámetro que determina cómo crecerá el árbol; de forma equilibrada eligiendo un nodo en el nivel de poca profundidad, o por lo contrario, de forma desequilibrado escogiendo un nodo de cualquier nivel.

### Características del *XGBoost*

- Regularización a partir de la hessiana de la función de pérdida. Es la ventaja más potente del *XGBoost* que se utiliza para evitar el sobreajuste de modelos.
- Flexibilidad. Además de los modelos de regresión y clasificación, también admite funciones objetivo definidas por el usuario. Se utiliza un función objetivo para medir el rendimiento del modelo dado un determinado conjunto de parámetros.
- A diferencia del GBM donde la poda del árbol se detiene una vez se produce una pérdida negativa, el *XGBoost* ofrece la profundidad máxima elegida y luego poda hacia atrás hasta que la mejora en la función de pérdida está por debajo de un umbral.
- Admite valores faltantes (NA's) en medio del procesamiento del algoritmo.
- Computación paralela ya que tiene un mejor soporte para el procesamiento utilizando todos los núcleos disponibles y reduciendo el tiempo total del proceso.
- Requiere trabajar con datos en forma de matrices numéricas para funcionar correctamente.
- Estructuras de datos mejoradas para una mejor utilización de la memoria caché del procesador.

Debido a su complejidad, para más detalles acerca del algoritmo se recomienda ver Chen, Tianqi and Guestrin, Carlos (2016) y Cho, Hyunsu. (2016).

## 3.5. Validación de los modelos

Una vez construidos los modelos, se debe comprobar la fiabilidad y la precisión de cada uno de ellos. Existen numerosas posibilidades a la hora de evaluar los modelos predictivos de clasificación: eficiencia, coeficiente de Gini normalizado, error medio, error cuadrático medio, etc. En este trabajo, se realizará la comparativa de los modelos de clasificación expuestos en la Parte II, empleando como medida principal el área bajo la curva ROC, también denominada AUC (del inglés, *Area under the curve*).

La curva ROC (del inglés, *Receiver Operating Characteristic*), es una metodología desarrollada para analizar un sistema de decisión. Consiste en una representación gráfica, en la que contrastamos la sensibilidad con la especificidad para un sistema de clasificación, comparando el rendimiento de los distintos modelos.

Debido a ello, es necesario conocer los conceptos de sensibilidad y especificidad para comprender esta métrica de validación. Son términos bastante conocidos en el mundo de la "Inteligencia Artificial" y en el mundo estadístico, por ello se explicarán brevemente. Surgen desde una serie de indicadores que contabilizan si las predicciones de un clasificador se han realizado correctamente (en nuestro caso, clasificador binario, 1's o 0's, según se estime para el cliente incrementos en productos de inversión o no, respectivamente).

La aplicación de un marcador, en situaciones como la descrita, produce cuatro posibles resultados reflejados en la Tabla 3.2, a la que se conoce como *matriz de confunsión*. Esta tabla de contingencia resume los resultados del modelo de predicción que se pueden clasificar como $G = 1$ resultado positivo ya que proporcionaría una inversión para la entidad, $G = 0$ resultado negativo, $Y = 1$ representa un sujeto con éxito y $Y = 0$ un sujeto sin la condición de éxito.

| | Predicción | |
|---|---|---|
| Observación | G=0 | G=1 |
| Y=0 | "True Negative" (TN) | "False Positive" (FP) |
| Y=1 | "False Negative" (FN) | "True Positive "(TP) |

Tabla 3.2: Matriz de confunsión.

Con estas cuatro métricas se calculan los criterios de sensibilidad y especificidad:

- **Sensibilidad (VPR)**.
  Proporción de observaciones con éxito correctamente identificados. Se le suele llamar "razón de éxitos", en otras palabras:
$$VPR = \frac{TP}{TP + FN}$$
- **Especificidad (SPC)**.
  Probabilidad o "razón de verdaderos negativos", por ejemplo, que un cliente que no invierta, tenga un resultado positivo. Se define como:
$$SPC = \frac{TN}{TN + FP}$$

Es importante estudiar la relación entre estos dos conceptos, ya que lo ideal para un modelo sería que no se solapasen, obteniendo verdaderos positivos y verdaderos negativos, pero no es así. Lo normal, es que el aumento de uno de estos valores disminuye el del otro. Por ello, se plantea un gráfico que ayude a identificar cuanto de preciso es el modelo utilizado. Se representa el valor que se obtiene al restarle la especificidad a la unidad, $FPR = (1 - SPC)$ "falsos positivos" (en el eje de abscisas), frente a la sensibilidad, $VPR$ "verdaderos positivos" (en el eje de ordenadas). La situación ideal sería estar cerca del vértice superior izquierdo, ya que en este caso, habría mucha sensibilidad y especificidad. A este gráfico se le conoce como curva ROC, y presenta diversas aplicaciones para diagnosticar un modelo con total exactitud.

En situaciones de clasificación en las que se usan sólo dos clases, cada objetivo, sujeto o caso es etiquetado con uno de los elementos del conjunto {0,1}, asegurándose la clase a la que pertenece dicho caso. Existen otros clasificadores que, de forma natural, producen la probabilidad del sujeto, que es un valor numérico que representa el grado en que ese sujeto es miembro de la clase. Estos valores pueden ser probabilidades estrictas, o pueden ser puntuaciones. Los marcadores que producen puntuaciones pueden ser usados junto a un punto de corte o umbral de decisión (c), obteniendo un clasificador binario. Denotando por c al umbral de decisión de éxito o punto de corte, de manera que si los métodos de clasificación estiman la probabilidad para $f(X) \geq c$, se considera resultado positivo, y negativo en otro caso. Entonces, las proporciones ligadas a ese punto c, $(FPR, VPR)$ serían:

$$FPR(c) = P[f(X) \geq c|Y = 0]$$

$$VPR(c) = P[f(X) \geq c|Y = 1]$$

Si el punto de corte c se incrementa ambas fracciones disminuyen y si c disminuye ambas aumentan.

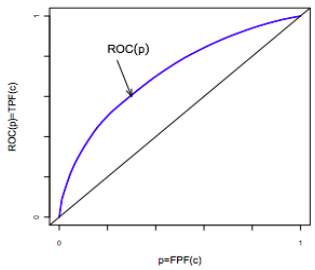



Figura 3.12: Ejemplo de una curva ROC.

El área bajo la curva ROC estima la capacidad de distinguir o de discriminar entre éxitos y no éxitos que tiene un modelo. Se define como:

$$AUC = \int_0^1 ROC(p)dp$$

Si el área bajo la curva valiese 1 (la curva ROC se dirige hacia la esquina superior izquierda) el modelo sería perfecto, ya que clasificaría al 100 % de los éxitos como éxitos y al 100 % de los exentos de éxito como no éxitos. En cambio, si el área bajo la curva valiese 0.5 (50 % área debajo de la línea diagonal) existiría la misma probabilidad de clasificar una observación como éxito que como no éxito. Un área de 0.5 bajo la curva, equivale a un modelo no informativo, de hecho tendría valores de sensibilidad iguales a 1-especificidad en todos los puntos de corte posibles. Esta situación no reduce nada el grado de incertidumbre previo acerca de la eficacia del modelo. Normalmente, los valores de AUC, son mayores a 0.5, considerando un buen modelo a partir de 0.7. Si por lo contrario, se obtiene un AUC menor de 0.5, requiere invertir los criterios de diagnóstico, es decir, considerar los negativos como positivos y viceversa.

Si se tienen dos modelo, tal que el modelo A es uniformemente mejor que un modelo B, se tiene que

$$ROC_A(p) \geq ROC_B(p) \; \forall p \in (0,1)$$

entonces los AUC estadísticamente mantienen el mismo orden,

$$AUC_A \geq AUC_B$$

aunque lo inverso no es necesariamente cierto, es decir, una curva ROC cualquiera con mayor AUC que otra, puede no ser mejor. Con esto, se concluye que solo es posible afirmar que si una curva ROC es uniformemente mejor que otra, tendrá mayor área bajo ella y el modelo correspondiente será el seleccionado, debido a su superior capacidad de discriminación.

A partir de la información vista en Tabla 3.2, se definen los errores FNR (del inglés, *False Negative Rate*), correspondientes a valores predichos por el modeo como $G = 0$ de forma incorrecta, y FPR (del ingés, *False Positive Rate*), valores predichos como $G = 1$ por el modelo pero que en los datos de entrenamiento corresponden a valores negativos .

Un factor importante a la hora de evaluar un modelo y compararlo con otros, reside en qué tipo de error es más tolerable; esto dependerá en gran medida del contexto para el que se esté generando un modelo predictivo. En este trabajo, la "visión de negocio" es adquirir el conocimiento necesario para adelantarse a las necesidades del cliente. Por tanto, es preferible que el error de FPR sea mayor que el de FNR, ya que esto implica que es mejor equivocarse en priorizar los beneficios de una persona que no contrará productos de inversión de la entidad, que confundirse al perder una ganancia por no haber considerado a un cliente que si lo contraría.

Por otro lado, se pueden definir métricas que resultan muy utiles a partir de los cuatro posibles resultados de la Tabla 3.2. El término *Exactitud* se refiere a cuán cerca del valor observado se encuentra el valor de la predicción. Está relacionada con el sesgo de la estimación, cuanto menor es el sesgo más exacta es la predicción. Se representa por la proporción de predicciones correctas:

$$Exact = \frac{TN + TP}{TN + TP + FN + FP}$$

Sería equivalente a restar la proporción del error de clasificación de la unidad, definiendo al error de clasificación como:

$$err = \frac{FP + FN}{TP + TN + FP + FN}$$

Aunque es más interesante obtener el valor de *Precisión* o valor de predicción positiva, ya que genera la probabilidad de que, dada una predicción positiva, la realidad sea positiva también. Se calcula de la siguiente forma:

$$Precision = \frac{TP}{FP + TP}$$

.

Esta sección se ha construido siguiendo Alonzo T.A., Pepe M.S. (2002) y Fawcett, Tom (2006).

# Parte II

# Resultados

# Capítulo 4

# Modelización del modelo de propensión de ahorro e inversión

A lo largo de este Capítulo, se comentarán los resultados obtenidos al realizar la predicción de la tasa de éxito en la tenencia de productos de ahorro e inversión para el modelo de propensión vigente en la actualidad en ABANCA, y para el modelo de propensión ampliado generado al añadir la variable *GRUPO* creada (ver Capítulo 2). Para ello, se emplearán las variables mensuales obtenidas mediante las variables internas de ABANCA, con las que se ajustarán las distintas metodologías estudiadas en el Capítulo 3 con el objetivo de demostrar, que la información no convencional extraída de la RRSS de Linkedin, es capaz de mejorar la capacidad predictiva del modelo de propensión de ahorro e inversión.

Una vez se ha realizado el análisis de todas las variables que entran en el modelo, visto en la Sección 2.4, disponemos de un conjunto de 18 variables explicativas y de la variable indicador de incrementos en posiciones de productos de inversión, como variable respuesta. Para todas las variables disponemos de un período temporal comprendido entre Mayo de 2018 hasta Abril de 2019. La variable objetivo se genera a partir de los posteriores 6 meses a la fecha de observación. Con el fin de ajustar los modelos y comprobar la calidad de los mismos a la hora de realizar las predicciones se dividirá el conjunto de datos a partir del *Cliente_ID* en muestras independientes, se tomará el 60 % de los datos para el conjunto de entrenamiento, 20 % para el test, y 20 % para la validación del modelo. La elección de los hiperparámetros se realiza considerando la muestra de test y de entrenamiento, y se evaluarán las predicciones en la muestra de validación.

En las siguientes secciones, se mostrarán los diferentes modelos de clasificación ajustados para dicha tarea. En concreto en la Sección 4.1 se ajustará un modelo de regresión logística, en la Sección 4.2 se ajustará un modelo *Bagging*, concretamente un *Random Forest*, en la Sección 4.3 un modelo *Boosting*, en especial un *Gradient Boosting*, y por último, en la Sección 4.4 un *XGBoost*, caso particular de los modelos *Boosting* y el cual proporcionó un mejor resultado. Por último, en la Sección 4.5, se realizará una breve comparación del comportamiento de los distintos modelos.

Destacar que en la práctica, es necesario dedicar la mayoría del tiempo y los esfuerzos en el tratamiento de los datos. Si la calidad de información que se introduce en el modelo no es lo suficientemente correcta, los resultados obtenidos no serán exactos ni tampoco fiables. Las principales tareas realizadas para el análisis de datos fueron las siguientes: el estudio de la función objetivo, la transformación de los datos faltantes (eliminar las observaciones concretas que los tengan o imputarlos, es decir, sustituirlos por estimaciones) o el estudio de las variables explicativas que vamos a introducir (como se comportan frente a la variable objetivo o si se pueden transformar para mitigar su efecto), ver Sección 2.4. Por otro lado, es imprescindible dedicar tiempo a la construcción y elección del mejor modelo, lo cual resulta ser la parte más laboriosa computacionalmente. Para ello, como veremos a lo largo de este Capítulo, se ajustan los modelos en los cuales los hiperparámetros de cada uno van variando de manera aleatoria entre una serie de valores predeterminados que se han impuesto. Para finalizar, se validan los resultados de las predicciones mediante métricas concretas como las vistas en la Sección 3.5, y además, es preciso estudiar al detalle la importancia que se le da a las variables y asegurarse que se mantiene un sentido lógico.

La construcción de estos modelos se realizó en el entorno de programación `R` a través de las librería `h2o`, salvo para el modelo *Extreme Gradient Boosting* que se construyó con la librería `xgboost`. Se debe destacar que la primera librería nombrada permite paralelizar el proceso, disminuyendo con ello el tiempo de computación. Para la mayoría de los métodos, como el *Random Forest*, es fácil paralelizar ya que se basan en la creación de árboles de manera independiente. Debido a la gran cantidad de tiempo que lleva realizar la creación y optimización de

árboles, paralelizar el proceso aprovecha toda la capacidad computacional disponible[1].

## 4.1. GLM

Como se ha comentado, se realizará el modelo con todo el conjunto de datos de entrenamiento sobre el cual se obtendrá la tasa de éxito para los 6 meses posteriores a la observación del período anual de cada cliente. En primer lugar, antes de crear el modelo de regresión logística a partir del conjunto de datos, se comentará el ajuste de hiperparámetros previo.

El ajuste de todos los modelos GLM consiste en identificar el valor de los coeficientes $\beta$ que maximizan la verosimilitud de los datos, como se ha visto en la Subsección 3.1.2. Es decir, los valores de los coeficientes que dan lugar al modelo que con mayor probabilidad puede haber generado los datos observados. Para la familia Binomial, no existe una solución analítica con la que encontrar la máxima verosimilitud, por lo que se requiere un método de optimización iterativo como el método de Newthon-Rapshon. La librería `h2o`, selecciona automáticamente el método apropiado dependiendo de la función link especificada. La librería incorpora dos métodos: *Iteratively Reweighted Least Squares Method (IRLSM)* y *Broyden-Fletcher-Goldfarb-Shanno (BFGS)*, cuya correcta elección, que depende del tipo de datos, puede influir notablemente en el rendimiento computacional.

También incorpora los 3 tipos de regularización (Lasso, Ridge y Elastic Net), con el fin de evitar un sobreajuste, haciendo innecesario emplear métodos de selección de variables, reducir varianza y atenuar el efecto de la correlación entre predictores. Por lo general, aplicando regularización se consigue modelos con mayor poder predictivo.

Encontrar el mejor modelo implica identificar los valores óptimos de los hiperparámetros $\alpha$ y $\lambda$ de la regularización Elastic Net, que consigue el equilibrio óptimo entre el Lasso y el Ridge. Por defecto, se selecciona $\alpha = 0.5$, y con cada uno de ellos, analiza un rango de valores de $\lambda$, esta búsqueda del valor óptimo se habilita indicando `lambda_search=TRUE`. Se pueden emplear `validation_frame` para elegir el método de validación empleado, puede ser mediante un único conjunto de validación o mediante validación cruzada, sin necesidad de crear un conjunto de validación, las particiones (en este caso `nfolds`=5) se generan a partir del conjunto de entrenamiento.

Destacar que los algoritmos `h2o`, están pensados para poder ser aplicados sobre grandes volúmenes de datos, por lo que, entre sus características, incorporan criterios de parada para que no se continúe mejorando un modelo si ya se ha alcanzado una solución aceptable. En el caso de los GLM, el criterio de parada es el número máximo de predictores permitidos, que se puede especificar con `ma_active_predictors`. Esto resulta útil para evitar tiempos de computación excesivos cuando se trabaja con sets de datos que contienen cientos de predictores pero se sabe que solo unos pocos son importantes.

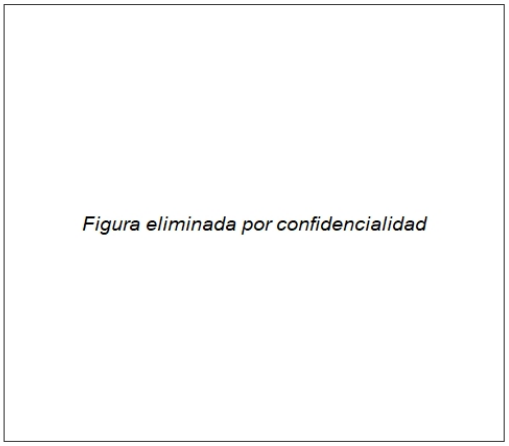
A lo largo del trabajo, se han realizado búsquedas del valor óptimo de $\alpha$, repitiendo diferentes ajustes del modelo. De entre los valores comparados, los que muestran mayor $AUC$ promedio de validación cruzada se obtiene con:

| Modelo de propensión | Modelo de propensión al añadir la variable |
|---|---|
| `alpha=0.5` | `alpha=0.95` |
| `nfolds=5` | `nfolds=5` |

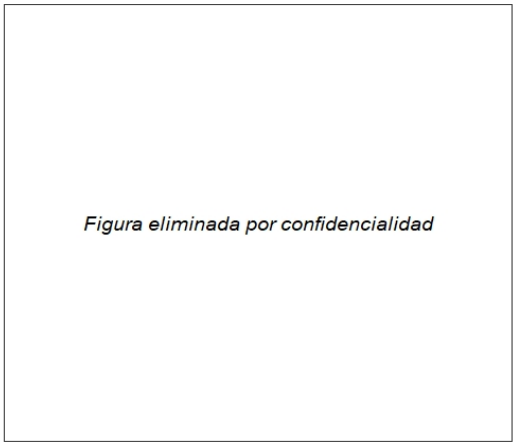
Tabla 4.1: Hiperparámetros seleccionados para el GLM.

Además, se puede especificar en la creación del modelo que se estandaricen las variables explicativas, con `standardize=TRUE`, generando de esta forma dos tipos de coeficientes de regresión. Por un lado, los "coeficientes" con las mismas unidades con las que entran en el modelo, y por otro lado, "coeficientes estandarizados", obtenidos directamente del algoritmo resultando de utilidad para ordenar de mayor a menor influencia en el modelo. En los modelos lineales generalizados, la importancia de las variables puede estudiarse a partir del valor absoluto de los coeficientes de regresión estandarizados. Para obtener más información, sobre la correcta interpretación de los coeficientes de regresión del modelo de regresión logística se recomienda volver a ver la Sección 3.1.2.

[1] De todos modos, también se puede hacer uso de librerías diferentes, como `randomForest` y `gbm`. En este trabajo, también se usaron para la generación de algunos de los gráficos




(a) Importancia relativa de variables.




(b) Importancia relativa de variables, modelo con RRSS.

Figura 4.1

Como se puede ver en la Figura 4.1, la variable con más importancia relativa en el modelo es la *EDAD*. En la Sección 2.4, se explicó el comportamiento lineal que presenta esta variable frente a la variable objetivo. Recordar, que para el perímetro restringido de personas que tienen Linkedin, los incrementos en la contratación de productos de inversión, son mayores cuánto mayor es la edad del cliente. Ya que una trayectoria laboral mas amplia implica una mayor experiencia, y por ende, un mayor salario, lo que desemboca en mayor presupuesto disponible a destinar en bienes de consumo no esenciales como son los productos de inversión.

Además, como se puede observar en la Figura 4.1b, en el modelo en el que se ha añadido la nueva variable, las categorías de empleo con más importancia relativa, que afectan positivamente al indicador de incrementos en productos de inversión son: el grupo 2 (correspondiente a clientes profesionales del sector de la sanidad), el grupo 1-6 (gerentes de establecimiento o comercio) y el grupo 1-4 (directores comerciales y restauración). Estando todas ellas vinculadas a profesiones con una mayor remuneración y por tanto con una mayor renta disponible.

Se pueden obtener toda una serie de métricas calculadas a partir de los datos de validación, pero se van a considerar las explicadas en la Sección 3.5 como medidas principales. Al tratarse de métricas de validación (ni de test, ni de entrenamiento), son válidas y fiables como estimación del error que comete el modelo al predecir nuevos valores. También nos servirán, para realizar más tarde, la comparación entre los diferentes modelos utilizados.

| $GLM_{existente}$ | Predicción | | |
|---|---|---|---|
| Observación | G=0 | G=1 | err_condicional |
| Y=0 | 21667 | 2570 | FPR=0.11 |
| Y=1 | 970 | 528 | FNR=0.63 |
| | $Exact = 0.85$ | $err = 0.14$ | $Precision = 0.17$ |

| $GLM_{ampliado}$ | Predicción | | |
|---|---|---|---|
| Observación | G=0 | G=1 | err_condicional |
| Y=0 | 22013 | 2224 | FPR=0.09 |
| Y=1 | 946 | 552 | FNR=0.62 |
| | $Exact = 0.87$ | $err = 0.12$ | $Precision = 0.20$ |

Tabla 4.2: Matriz de confunsión del modelo GLM existente y del GLM ampliado.

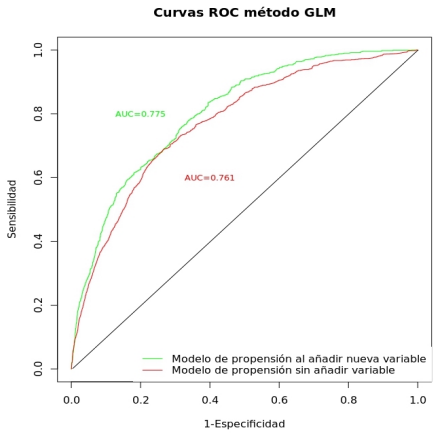



Figura 4.2: Curva ROC de los modelos GLM.

En la Figura 4.2, se puede observar como el área debajo de las dos curvas son mayores a 0.7, lo que nos dice que los dos modelos clasifican correctamente. Siendo el modelo nuevo, el que mayor precisión de clasificación presenta, gráficamente se puede deducir porque el área bajo su curva es mayor. Además, en los resultados de la Tabla 4.2, el modelo ampliado presenta valores más óptimos para la *Exactitud*. El error total es ligeramente mayor

para el modelo vigente en la actualidad, y lo mismo ocurre para el error FPR y FNR. Como se ha comentado, desde un punto de vista empresarial es preferible obtener un valor mayor del error FPR que FNR, ya que esto implica que es mejor equivocarse en priorizar las necesidades de un cliente que no contrará productos de inversión en la entidad, que no percatarse de ganancias por no haber considerado a un cliente que si invertiría .

Por tanto, se puede concluir que para el modelo GLM se cumple el objetivo definido para este trabajo, demostrando que la información extraída influye en la variable objetivo y aparentemente aumenta la probabilidad de predicción correcta de este modelo de propensión actual de ABANCA.

## 4.2. Random Forest

El método de *Random Forest* es una modificación del proceso *Bagging*, que como se ha contemplado en la Subsección 3.3.2, consigue mejores resultados gracias a que decorrelaciona los árboles de decisión generados en el proceso. Es conveniente recordar que el proceso *Bagging* se basa en el hecho de que, promediando un conjunto de modelos, se consigue reducir varianza. Esto es cierto siempre y cuando los modelos no estén correlacionados, ya que si la correlación es alta, la reducción de varianza que se puede lograr es pequeña. La principal diferencia entre estos dos métodos es que, el *Random Forest*, antes de cada división se seleccionan aleatoriamente $m$ predictores. De esta forma se mitigará el efecto producido por los predictores influyentes consiguiendo decorrelacionar los árboles, provocando una reducción de la varianza.

Como se ha comentado, la mejor forma de escoger el valor óptimo de $m$, `mtry`, es evaluar el $out-of-bag$ para diferentes valores. Destacar que el *Random Forest*, no sufre problemas de sobreajuste por aumentar el número de árboles creados en el proceso, ya que alcanzado un determinado número la reducción de *testerror* se estabiliza.

Como se muestra en la Figura 4.3, se identifica 7 como número óptimo de predictores (`mtries`) a evaluar en cada división, aunque para que computacionalmente sea más fácil, como la diferencia entre `mtry=4` y `mtry=7` es muy poco significativa, se tiende a escoger el menor. Además del número de predictores evaluados en cada división, el método de *Random Forest* tiene otros dos hiperparámetros importantes, el número mínimo de observaciones que deben tener los nodos terminales (`nodesize`) y el número de árboles (`ntree`). Por defecto, `h2o` considera `nodesize`=1 y `ntree`=500, pero para ambos hiperparámetros se puede representar su optimización empleando el *out-bag-error* de la misma manera que se hizo con `mtries`, pudiendo observar que aunque se seleccione un número suficientemente grande de forma que converga, llega un punto en que el modelo se estabiliza. Como se puede ver en la Figura 4.4, alcanzados en torno a 200 árboles .

| Modelo de propensión | Modelo de propensión al añadir la variable |
|---|---|
| ntrees=300 | ntrees=200 |
| max_depth=35 | max_depth=5 |
| min_rows=300 | min_rows=201 |
| mtries=4 | mtries=7 |
| sample_rate=0.632 | sample_rate=0.75 |

Tabla 4.3: Hiperparámetros seleccionados.

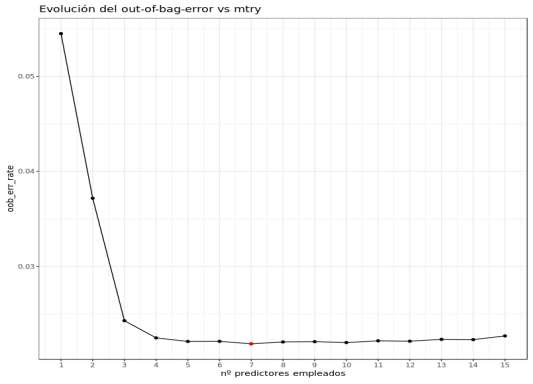



Figura 4.3: Evolución del *out-bag-error* para el modelo actual.

Tras varias simulaciones se seleccionaron los hiperparámetros que se muestran en la Tabla 4.3, que proporcionaban un mejor resultado y menor diferencia entre los resultados de las medidas de validación para la muestra de entrenamiento y para la muestra de test.

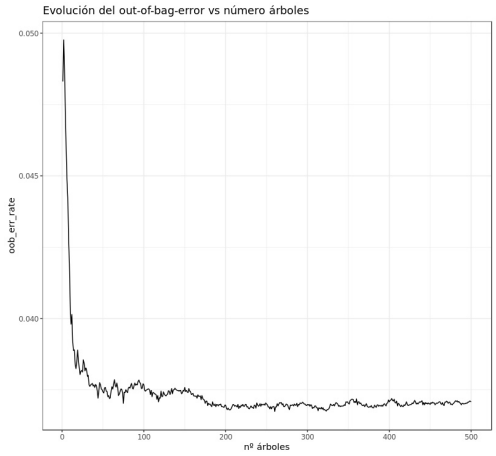



Figura 4.4: Evolución del *out-bag-error* para el modelo actual.

Para identificar los predictores más influyentes, se utilizó la medida de impureza de nodos a partir de la tasa de observaciones mal clasificadas comentada en la Subsección 3.3.1. Como se ha visto, también se podría realizar con otra medida de impureza como el índice de Gini, aunque el resultado apenas varía. A continuación se muestran las variables más influyentes para el modelo de propensión actual en la entidad y el construido al incluir la nueva variable.

*Figura eliminada por confidencialidad*

(a) Importancia relativa de variables.

*Figura eliminada por confidencialidad*

(b) Importancia relativa de variables, modelo con RRSS.

Figura 4.5

Parece que en los dos modelos las variables $CAPACIDAD\_AHORRO$ y $TRIAD\_LIMITE\_PP$ son con diferencia las más importantes en el modelo, seguidas de la $EDAD$ y los $INGRESOS$. Cabe destacar que la variable $EDAD$ reduce su importancia debido a que estos modelos no contemplan solamente la influencia lineal de las variables sobre la variable respuesta.

Además, como se puede examinar en la Figura 4.5b, el incluir la variable $GRUPO$ construida a lo largo del trabajo, también genera una influencia importante, y por tanto se puede deducir, que su valor contribuye a la predicción de la variable objetivo.

Para finalizar, se muestra una medida simple para examinar los resultados de los modelos. Se obtuvieron las métricas de error empleadas en la Sección 4.1.

| $RF_{existente}$ | Predicción | | |
|---|---|---|---|
| Observación | G=0 | G=1 | err_condicional |
| Y=0 | 22241 | 1996 | FPR=0.08 |
| Y=1 | 922 | 576 | FNR=0.61 |
| | $Exact = 0.88$ | $err = 0.12$ | $Precision = 0.22$ |

| $RF_{ampliado}$ | Predicción | | |
|---|---|---|---|
| Observación | G=0 | G=1 | err_condicional |
| Y=0 | 22075 | 2162 | FPR=0.09 |
| Y=1 | 885 | 613 | FNR=0.59 |
| | $Exact = 0.88$ | $err = 0.12$ | $Precision = 0.23$ |

Tabla 4.4: Matriz de confunsión del modelo RF existente y del RF ampliado.

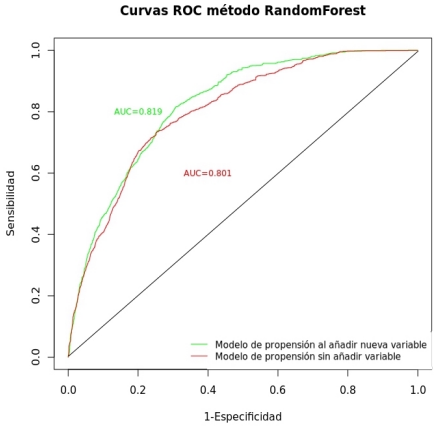



Figura 4.6: Curva ROC de los modelos *Random Forest*.

Como se puede observar en la Figura 4.6, el modelo de propensión con la intervención de la nueva información, mejora a un 0.819 de AUC frente a 0.801, cuantificando una mejora en las predicciones del modelo. Destacar, que cuando se obtienen modelos con un AUC superior a 0.8, perfeccionarlos es una tarea muy complicada. En los resultados de la Tabla 4.4, aunque las métricas de *Precisión* y *Exactitud* son prácticamente iguales, para el modelo existente el error de FNR, es mayor que para el modelo ampliado, por lo que este último modelo sería preferible para la entidad (con un 3 % menos de error condicional cometido).

## 4.3. GBM

Esta sección, se centra en la construcción del modelo *Gradient Boosting Machine*, un caso particular de los modelos *Boosting*. Los cuales agregan secuencialmente clasificadores débiles, basados en árboles de decisión, de forma que disminuyan los errores del anterior. Su funcionamiento en particular, se puede revisar en la Sección 3.4.

Como se ha explicado en la Subsección 3.4.2, se aplica el método de descenso de gradiente para optimizar cualquier función de coste durante el ajuste del modelo. El valor que se predice, es la agregación (la moda en clasificación) de las predicciones de todos los modelos individuales que forman el *ensemble*. Dado que el ajuste de los clasificadores débiles o *weak learners*, se realiza de forma secuencial, las posibilidades de paralelizar son limitadas, pero `h2o`, consigue acelerar el proceso en cierta medida para los árboles individuales.

Una de las características del GBM es que, con un número suficientemente alto de árboles, el modelo final tiende a ajustarse perfectamente a los datos de entrenamiento produciendo sobreajuste. Esto suele ser poco eficiente en términos de tiempo, ya que se están ajustando árboles innecesarios. Por ello, el ajuste correcto de los hiperparámetros (definidos en la Subsección 3.4.2), es fundamental. Por defecto, la métrica del ajuste se calcula con los datos de validación, aquí es cuando se refleja la importancia de separar el conjunto de datos, en 3 grupos, con la muestra de validación independiente.

Dada la gran cantidad de hiperparámetros que tiene el algoritmo GBM y, aunque todos interaccionan entre sí, no es posible explorar todo el espacio de posibles valores a la vez. Una estrategia que suele funcionar bien es establecer primero los hiperparámetros mas influyentes: número de árboles, learning rate, profundidad de árboles y el porcentaje de observaciones empleadas en cada ajuste. Tras haber realizado varias pruebas para cada modelo, los hiperparámetros seleccionados se muestran en la Tabla 4.5.

Además, hay otros argumentos que controlan la estrategia de parada, `stopping_metric` para cuantificar cuánto mejora el modelo ("MSE", "AUC", "logloss", etc.), `stopping_tolerance` porcentaje mínimo de mejora entre dos mediciones, y `stopping_rounds`, número de mediciones consecutivas en las que no se debe superar el `stopping_tolerance` para que el algoritmo se detenga.

| Modelo de propensión | Modelo de propensión al añadir la variable empleo |
|---|---|
| ntrees=30 | ntrees=100 |
| max_depth=5 | max_depth=10 |
| min_rows=500 | min_rows=1000 |
| learn_rate=0.1 | learn_rate=0.05 |
| sample_rate=0.75 | sample_rate=0.3 |
| col_sample_rate=1 | col_sample_rate=1 |

Tabla 4.5: Hiperparámetros seleccionados.

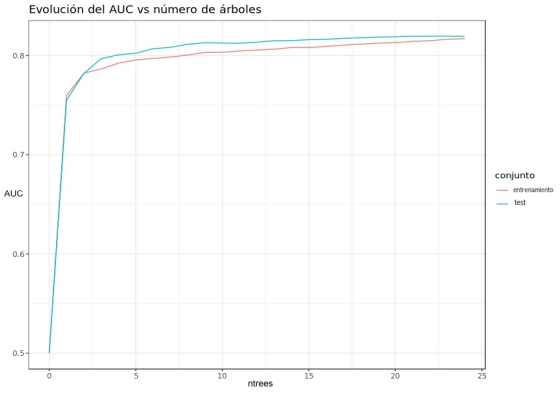



Figura 4.7: Evolución de un métrica para el modelo actual.

Tras ajustar un modelo GBM, es fundamental evaluar como aprende el modelo a medida que se añaden nuevos árboles. Una forma de hacerlo es representar la evolución de una métrica, por ejemplo, el área bajo la curva ROC, de la muestra de entrenamiento y la muestra de test. Puede observarse en la Figura 4.7 que no se han alcanzado los 30 árboles indicados en el modelo, porque alcanzados los 24, el valor AUC se ha estabilizado lo suficiente como para que se ejecute la detención temprana.

Como se ha explicado en la Capítulo 3, en los modelos GBM de `h2o`, se estudia la influencia de las variables explicativas en el modelo cuantificando la reducción de la impureza de los nodos a partir de la tasa de error de clasificación que ha conseguido cada una en el conjunto de todos los árboles que forman el modelo.

*Figura eliminada por confidencialidad*

(a) Importancia relativa de variables.

*Figura eliminada por confidencialidad*

(b) Importancia relativa de variables, modelo con RRSS.

Figura 4.8

El modelo GBM, considera que las variables que más influyen en el modelo son la $CAPACIDAD\_AHORRO$ y el $TRIAD\_LIMITE\_PP$ resultado similar al obtenido con el modelo *Random Forest*. Destacar que en la Figura 4.8b, se observa también la influencia de la variable introducida, $GRUPO$, compuesta por los registros que proporcionó la información de la RRSS (red social). En conclusión, parece que se ha conseguido construir una variable influyente en la estimación de la contratación de productos de inversión.

Para identificar cual de ellos consigue mejores resultados para el problema en cuestión, predecir la contratación de productos de inversión, se valida a través del área debajo de la curva ROC, Figura 4.9. Ajustando los hiperparámetros seleccionados de la Tabla 4.5, se ha conseguido mejorar el AUC del modelo de propensión actual en la entidad 0.794 a un 0.823, tras añadir la nueva información. Los dos modelos ajustados tienen un acierto en la predicción superior al 0.7, aunque el modelo con la nueva variable consigue el AUC más alto. De la misma forma, para los valores de la Tabla 4.6, se muestra un resultado más preciso para el modelo GBM ampliado, concretamente con un 5 % menos de error.

| $GBM_{existente}$ | Predicción | | |
|---|---|---|---|
| Observación | G=0 | G=1 | err_condicional |
| Y=0 | 22655 | 1582 | FPR= 0.06 |
| Y=1 | 946 | 552 | FNR=0.63 |
| | $Exact = 0.9$ | $err = 0.10$ | $Precision = 0.25$ |

| $GBM_{ampliado}$ | Predicción | | |
|---|---|---|---|
| Observación | G=0 | G=1 | err_condicional |
| Y=0 | 22555 | 1682 | FPR= 0.07 |
| Y=1 | 880 | 618 | FNR=0.58 |
| | $Exact = 0.9$ | $err = 0.10$ | $Precision = 0.26$ |

Tabla 4.6: Matriz de confusión del modelo GBM existente y del GBM ampliado.

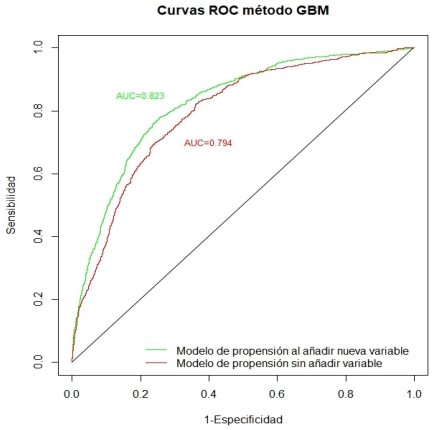



Figura 4.9: Curva ROC de los modelos GBM.

## 4.4. XGBoost

El *Extreme Gradient Boosting* es similar al GBM, pero más eficiente ya que realiza cálculos paralelos, siendo 10 veces más rápido que las implementaciones del *Gradient Boosting*. Como se ha visto en la Subsección 3.4.3, tiene características adicionales para realizar validación cruzada y divisiones adicionales en cada árbol.

El *XGBoost* sólo trabaja con variables numéricas, por tanto, se deben convertir las variables categóricas en numéricas mediante la construcción de variables dummy. Además, también requiere la implementación de datos como matrices, específicamente de tipo DMatrix, así que es necesario convertir los sets de entrenamiento, test y validación a este tipo de estructura[2]. Al usar esta función, se define una matriz numérica como primer argumento y como atributo adicional la variable objetivo. Es muy importante que el conjunto de datos convertidos en matriz numérica no incluya la variable objetivo, ya que de lo contrario, se obtendría una precisión perfecta en las predicciones que será ineficaz a nuevos datos. Una vez realizado este proceso, es necesario ajustar los hiperparámetros. Este algoritmo dispone de una amplia cantidad de parámetros, para recordar la información de cada uno de ellos, se recomienda volver a leer la Subsección 3.4.3.

Tras varias simulaciones, para los dos modelos de propensión, los hiperparámetros que devuelven mejores resultados y producen menos diferencia de métricas entre la muestra de entrenamiento y de test son los que se muestran en la Tabla 4.7.

| Modelo de propensión | Modelo de propensión al añadir la variable |
|---|---|
| nrounds=500 | nrounds=130 |
| eta=0.01 | eta=0.05 |
| gamma=0 | gamma=0 |
| min_child_weigth=1 | min_child_weigth=20 |
| max_delta_step=5 | max_delta_step=10 |
| max_depth=3 | max_depth=3 |
| subsample=0.5 | subsample=0.75 |
| colsample_bytree=1 | colsample_bytree=0.75 |
| colsample_bylevel=0.75 | colsample_bylevel=0.5 |

Tabla 4.7: Hiperparámetros seleccionados.

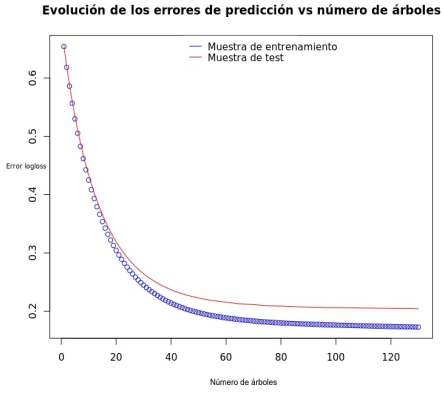


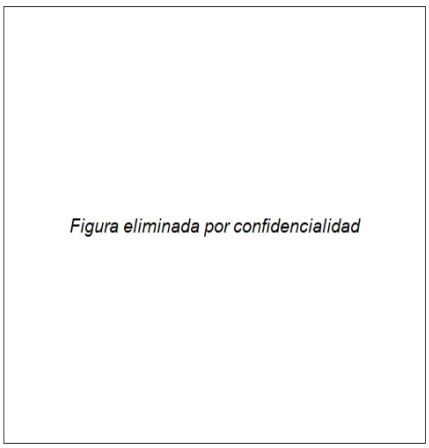

Figura 4.10: Evolución de los errores de predicción del modelo actual.

[2]Para ello, se utilizará la función `xgb.DMatrix()`del paquete `xgboost`.

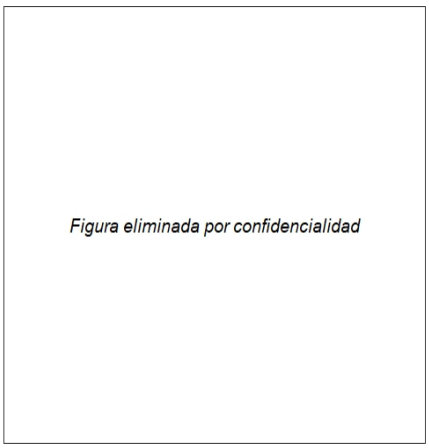
En la Figura 4.10, es fundamental hacer notar como desciende el error de las predicciones que penaliza la función de pérdida, durante la fase de entrenamiento a medida que se desarrollan las 130 iteraciones, así como también sucede lo mismo, en la fase de test. La determinación del número exacto de iteraciones proviene de realizar una validación cruzada, ya que este método permite de forma óptima obtener la coordenada donde la función objetivo obtiene su mínimo, que en este caso, se corresponde con el menor error.

La importancia relativa de las variables explicativas que entran en el modelo, se calcula de una forma diferente a las vistas en la Capítulo 3 para los modelos *Random Forest* o GBM. La idea principal sigue siendo la misma, basándose en la contribución o influencia de una variable sobre los errores de predicción y en la iteración o impacto producido en otras variables. Es decir, las medidas se basan en la cantidad de veces que se selecciona una variable para realizar una ramificación, ponderada por el error al cuadrado del modelo como resultado de cada división y promediada sobre todos los árboles. Pero para el algoritmo *XGBoost*, la influencia de los predictores sobre la variable objetivo, se calcula a través de una matriz de importancia, que contiene como columnas las características empleadas en la construcción de árboles de decisión. Se refiere a características, ya que todas las variables pasan a ser numéricas, por tanto representan rasgos diferentes.

A la inversa del resto de modelos que realizan de forma automática internamente la construcción de variables dummy para cada uno de los niveles de las variables categóricas, en este modelo es necesario realizarlo de antemano pero como ventaja, nos permite conocer la influencia que presentan los diferentes niveles de la variable empleo. Como se puede ver en la Figura 4.11, para el modelo de propensión con la nueva variable, los grupos profesionales que presentan más influencia en el modelo son el 2 (correspondiente a clientes profesionales del sector de la sanidad), el 1-6, 1-4 y 1-3 (gerentes y directores comerciales y restauración), el 4 (científicos intelectuales) y el 5 (profesionales de derecho). Todos los grupos más influyentes se corresponden con puestos altamente remunerados que conllevan a una mayor renta disponible. Aunque no se tiene en cuenta el signo del efecto de cada variable para el modelo, a la vista de los gráficos parece que los clientes pertenecientes a estos grupos tienen mayor predisposición a invertir que clientes pertenecientes a grupos de empleo con remuneraciones menores. Destacar que una vez más, se está demostrando la influencia de esta información para la predicción de la contratación de inversiones.




(a) Importancia relativa de variables.




(b) Importancia relativa de variables, modelo con RRSS.

Figura 4.11

Para evaluar los dos modelos, se utilizaron las medidas obtenidas a partir de la matriz de confusión de la Tabla 4.8. El resultado obtenido por el modelo que dispone actualmente ABANCA presenta un AUC de 0.815, un modelo, que de por sí es muy difícil mejorar, pero añadiendo la nueva variable la medida sube a 0.828, Figura 4.12. Por tanto, se concluye que la precisión en las predicciones y la eficacia del modelo, se puede mejorar al añadir la información de las diferentes categorías de empleos que presentan los clientes en el horizonte temporal seleccionado.

| $XGB_{existente}$ | Predicción | | |
|---|---|---|---|
| Observación | G=0 | G=1 | err_condicional |
| Y=0 | 21221 | 3016 | FPR=0.12 |
| Y=1 | 744 | 754 | FNR=0.50 |
| | $Exact = 0.86$ | $err = 0.14$ | $Precision = 0.20$ |

| $XGB_{ampliado}$ | Predicción | | |
|---|---|---|---|
| Observación | G=0 | G=1 | err_condicional |
| Y=0 | 21521 | 2716 | FPR=0.11 |
| Y=1 | 734 | 764 | FNR=0.48 |
| | $Exact = 0.87$ | $err = 0.13$ | $Precision = 0.22$ |

Tabla 4.8: Matriz de confunsión del modelo *XGBoost* existente y del *XGBoost* ampliado.

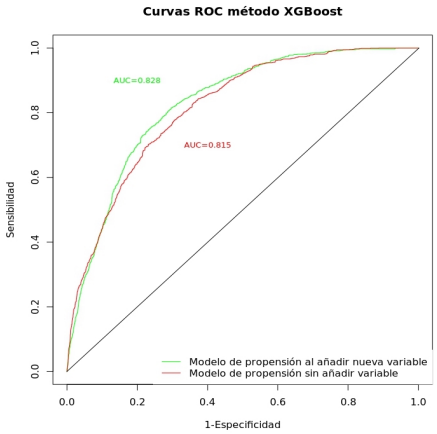



Figura 4.12: Curva ROC de los modelos *XGBoost*.

## 4.5. Comparación de modelos

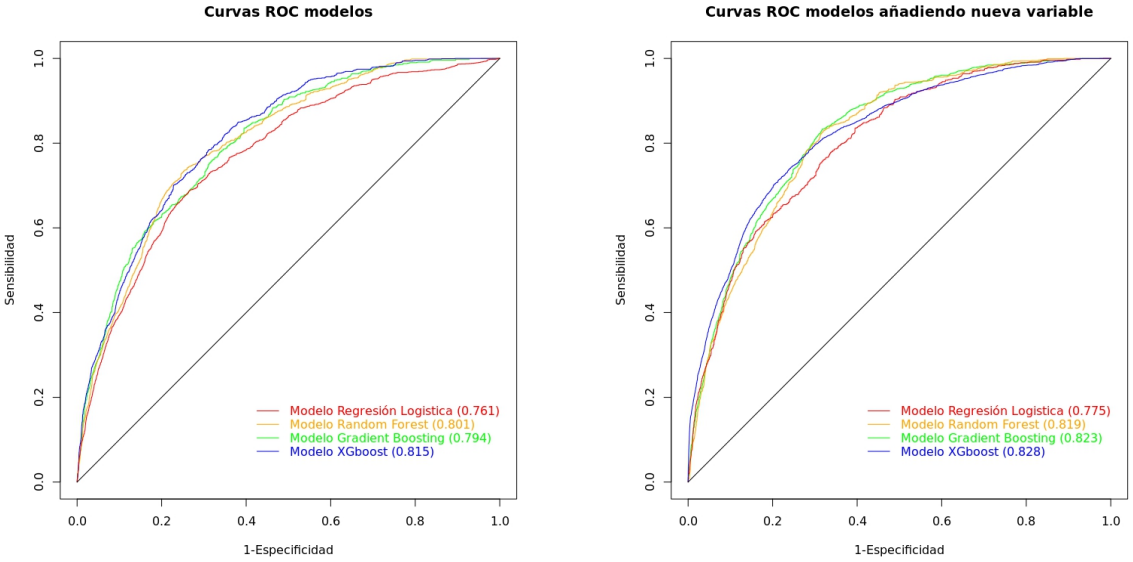
Se ha trabajado con cuatro metodologías muy vinculadas y el objetivo de esta sección es compararlas, así como destacar la capacidad de los modelos de clasificación para adelantarse a la actuación de los clientes en la contratación de productos de ahorro e inversión.

A lo largo del trabajo, se han expuesto características particulares que presenta el procedimiento de cada modelo con el objetivo de entender su funcionamiento y obtener en la práctica los mejores resultados. La clave principal reside en la optimización de los hiperparámetros, que elegidos correctamente, consiguen mejorar problemas internos como el sobreajuste, correlación entre variables o error de predicciones. La regresión logística y el *Random Forest*, tienen menos hiperparámetros, lo que hace más sencillo su correcta implementación. Pero, si existe una proporción de variables irrelevantes, el *Random Forest* puede verse perjudicado, sobre todo a medida que se reduce el número de predictores `mtry` reduciendo la capacidad predictiva. En el caso de los modelos *Boosting*, siempre se evalúan todas las variables predictoras, ignorando las no relevantes. Aunque el *Random Forest*, es el único de los modelos vistos, que no sufre problemas de sobreajuste por muchos arboles que se agreguen.

Existe una clara comparación entre el GLM y los modelos *Bagging* y *Boosting* gracias al *out-of-bag*, estos últimos no necesitan recurrir a validación cruzada para la estimación de los parámetros, mostrando una ventaja muy importante frente al GLM penalizado por métodos de regularización, sobre todo cuando los requerimientos computacionales están limitados.

La puesta a prueba de todos ellos resultó apropiada para el análisis de la contratación de ahorro e inversión utilizando información de un medio no utilizado hasta ahora, Linkedin. Los modelos permitieron el uso eficiente y apropiado de los datos, en la medida en que la estimación sobre la variables objetivo se ajusta a lo real. De entre todos ellos, el modelo que mejor resultados proporciona es el *XGBoost* o el GBM, seguidos del *Random Forest* y la regresión logística.

Cabe señalar que los valores obtenidos para validar la eficacia de la clasificación del modelo, pueden representar diferencias sustanciales con las obtenidas en un modelo diferente. La métrica de evaluación utilizada para la presente investigación fue el área bajo la curva ROC, técnica muy útil y muy utilizada para estimar la confiabilidad de cada modelo con relación al conjunto de datos. La comparación de los resultados obtenidos se encuentran sintetizados en la Figura 4.13a para los modelos actuales en la entidad y, en la Figura 4.13b para los modelos generados a lo largo de este trabajo con una información nueva acerca de los clientes. A la vista de las gráficas, con todos los modelos se consigue un valor superior al 0.76 de AUC, lo que dificulta la cuantificación de mejora al introducir la variable construida. Aun así, se logra demostrar que la vinculación de esta variable con la variable objetivo, es alta, lo que mejora la estimación de las predicciones.




(a) Resultados de validación para los modelos actuales. (b) Resultados de validación para los modelos nuevos.

Figura 4.13

Los resultados obtenidos indican que el modelo de clasificación *XGBoost* proporciona el mejor AUC, además presenta el menor error cometido FNR, es decir, es el modelo que comete menos desaciertos en predecir registros negativos cuando son positivos en el grupo de clientes que realizan inversión, devolviendo los mejores resultados para el modelo de propensión. Esto es producto de sus cualidades al enfrentar el problema de clasificación cuando se da una proporción alta de predictores irrelevantes, así como el complejo funcionamiento que realiza divisiones del árbol hasta encontrar una mejora en la función de pérdida. En cambio, la regresión logística genera los peores resultados en los dos modelos. Como se ha visto en la Sección 2.4, excepto quizás la variable EDAD, la relación entre los predictores y la variable respuesta no aparentan ser lineales, existiendo una relación compleja y altamente no lineal, lo que proporciona mejores resultados para los modelos basados en Árboles de decisión. Por tanto, el modelo que pueda dar mejores resultados depende del problema en cuestión. Para confirmar las suposiones del comportamiento de las variables en el modelo, se realizó la diagnosis y el estudio en detalle de la interpretación de los resultados; algunos de los gráficos más destacados pueden verse en el Apéndice A.

Cabe destacar, que los modelos han sido construidos de forma paralela, es decir, con la misma muestra de clientes observando su comportamiento a lo largo del mismo horizonte temporal, pero la construcción de su estructura se ha realizado de forma independiente. De esta manera para cada metodología, los hiperparámetros óptimos del modelo con las variables actuales en la entidad no tienen porque ser los mismos que para el modelo añadiendo la nueva variable. Es decir, a la hora de seleccionar el conjunto de hiperparámetros que genera un modelo en particular, lo ideal es elegir aquellos con resultados similares entre la muestra de entrenamiento y la muestra de test, utilizando esta última para eliminar modelos sobreajustados. Para ello, se selecciona aquel modelo que cumpla las siguientes condiciones y maximice el AUC del conjunto de test:

$$\frac{err \text{ Test}}{err \text{ Entrenamiento}} > 0.9 \;\; y \;\; \frac{AUC \text{ Test}}{AUC \text{ Entrenamiento}} > 0.9$$

en otras palabras, se elige aquel modelo que obtenga mejores resultados siempre y cuando los valores de las métricas de validación, como el AUC o el MSE, sean similares para las dos muestras de desarrollo.

Si bien en este trabajo se presentan a modo ilustrativo formas alternativas de interpretación de algunos resultados de los modelos y, dada la relevancia del tema de algoritmos Machine Learning, existen numerosas comparaciones entre estas metodologías, ya que aunque presentan una clara similitud por lograr la mejor estimación de la predicción de una variable, su funcionamiento es totalmente diferente.

# Capítulo 5

# Conclusiones y líneas futuras

Para cerrar esta memoria, se concluye con un breve resumen de lo que se ha realizado a lo largo del trabajo, y se describen las líneas de trabajo que han ido surgiendo y que actualmente están en desarrollo o en vías de ser desarrolladas.

## 5.1. Conclusiones finales

El objetivo principal de este trabajo era cuantificar la mejora del modelo de propensión de ahorro e inversión de la entidad, al introducir una variable procedente de una fuente de información desestructurada, en concreto, información procedente de redes sociales (RRSS) que no habían sido explotadas hasta el momento. Para tan fin, se han empleado técnicas *Text Mining* y *Web Scraping*, y se han revisado metodologías diferentes de clasificación supervisada para realizar los modelos de predicción y poder anticipar movimientos y cambios en la dinámica de productos de inversión de ABANCA. Por un lado, se han puesto en práctica modelos lineales generalizados penalizados con la intención de realizar una interpretación sencilla y un ajuste simple. Por otro lado, se han ajustado modelos más laboriosos, como pueden ser *Bagging* y *Boosting*, que permiten encontrar relaciones más complejas, y en ocasiones debido a ello, estimar predicciones más fiables. Con ambas metodologías los resultados han sido bastante precisos y la acogida por parte de los expertos de la entidad, que van a hacer uso de estas proyecciones, ha sido buena.

Además, quiero destacar que la realización de este Trabajo Fin de Máster ha sido un proceso de continuo aprendizaje. Para su desarrollo, se han combinado conocimientos de distintas áreas; desde el ámbito económico se han adquirido conocimientos relacionados con productos de ahorro e inversión y sobre los indicadores utilizados por la entidad a la hora de construir un modelo de propensión del cliente. Desde el punto de vista estadístico, ha sido un reto enfrentarse al estudio de modelos de clasificación, así como al uso de herramientas analíticas empleadas, que han enriquecido el trabajo. Durante su elaboración, un aspecto a destacar ha sido hacer frente a la gran carga computacional requerida. Por último, destacar el descubrimiento de muchas librerías de `R`, que han ayudado a realizar el análisis estadístico.

## 5.2. Líneas futuras de estudio

Este Trabajo de Fin de Máster ha proporcionado buenos resultados desde el punto de vista empresarial, ofreciendo la ventaja de anticiparse a los comportamientos de los clientes en las acciones comerciales del banco. Durante su realización, han surgido ciertas líneas de trabajo que el Departamento de Modelos de Predicción y Prospección de ABANCA quiere poner interés y seguir desarrollando:

- Debido a la mejora del modelo de propensión de ahorro e inversión al introducir una variable de información no convencional totalmente nueva, se han propuesto nuevos objetivos para extraer los datos de la RRSS[1] escogida para el resto de los clientes del banco y construir de nuevo, el modelo de predicción u otros modelos para los que se considere importante la información extraida.
- La introducción de información procedente de RRSS ha sido un valioso descubrimiento, permite enriquecer los modelos y también, adquirir conocimientos de los clientes que no se poseían. Está en desarrollo la implementación de algunos de los procedimientos y estudios que se llevaron a cabo para sustraer la información de Linkedin, en otras redes sociales.
- El conocimiento del algoritmo para clasificar los sectores laborales que dispone el INE, ha dado pie a descubrir métodos de clasificación no supervisada que categoricen de forma similar.

[1] A lo largo del trabajo, RRSS se ha denotado como abreviatura para redes sociales.

- Un modelo lineal es deseable porque es simple de ajustar y existen muchas técnicas disponibles para emplear. Sin embargo, la mayoría de los datos reales no están relacionados de forma lineal, por lo un enfoque diferentes sería utilizar los modelos de regresión no-lineal como los modelos generalizados aditivos. Aunque en la entidad para modelos de propensión actualmente no se utiliza la metodología GAM, se ha propuesto para líneas futuras de trabajo en el área de Modelos Predictivos y de Prospección.
- En general, los modelos estadísticos utilizados se midieron en clientes que son independientes entre sí, pero su observación a lo largo de 12 meses puede generar dependencia. Como la variable respuesta es dicotómica, se asume, de forma muy razonable, que los errores no explicados de los modelos no son normales. Pero puede considerarse la posibilidad de reducir restricciones sobre los errores, contemplando errores no independientes o heterogéneos. Esta generalización, que permite dotar de estructura a la variabilidad de los errores del modelo, da lugar a los modelos mixtos, que se pueden fusionar con los GLM y obtener una nueva modelización con resultados más eficientes. A raíz de esta teoría, se ha propuesto una línea de trabajo.
- Ante el buen funcionamiento de los modelos de clasificación con técnicas de árboles de decisión para un mejor conocimiento de las necesidades de sus clientes, se ha decidido probar este enfoque para el resto de los modelos de propensión de la entidad, añadiendo otras metodologías como *Redes Neuronales* o SVM (del inglés, *Support Vector Machines*).

# Apéndice A

# Gráficos para la interpretación de los modelos de propensión

Durante el tratamiento de datos realizado en la Sección 2.4, se ha llevado a cabo una exploración descriptiva de cada una de las variables explicativas antes de entrenar los modelos predictivos. Este proceso permite entender qué información contiene cada variable y empezar a dar ideas generales sobre qué variables pueden estar relacionadas con el éxito de la inversión y por tanto, resultarían ser predictores influyentes para el modelo. Pero en muchas ocasiones durante el ajuste del modelo las variables interaccionan entre ellas y su comportamiento se refleja de forma dispar. En este apéndice, se expondrán los gráficos correspondientes a la diagnosis del modelo, así como, la posible interación entre variables explicativas.

En muchas aplicaciones reales, conocer la importancia que tiene cada predictor dentro de un modelo no es suficiente, además, se necesita entender de qué forma influye cada uno en el valor de las predicciones obtenidas por el modelo ajustado.

Como se ha visto en la Sección 4.1, en el modelo GLM penalizado por métodos de regularización, gracias a los coeficientes de regresión, es fácil conocer en qué dirección y magnitud influyen las variaciones de una variable predictora en el valor predicho. Sin embargo, en modelos más complejos como *Random Forest*, GBM o *XGBoost*, no es tan inmediato. Una estrategia para estos últimos son los gráficos *PDP* (del inglés, *Partial Dependence Plots*) y *ICE* (del inglés, *Individual Conditional Expectation*). Ambas técnicas, trazan el cambio en el valor predicho a medida que las variables explicativas varían sobre su distribución marginal.

En concreto, los gráficos *PDP* muestran cómo varían, en promedio, las predicciones a medida que se modifica el valor de un predictor, manteniendo constantes el resto. Este método se presentó por Friedman, Jerome H (2001) en su artículo dedicado a *Gradient Boosting Machine*. Para más detalle acerca de la formulación matemática se recomienda ver Goldstein, Alex and Kapelner, Adam and Bleich, Justin and Pitkin, Emil (2015) y Brandon M. Greenwell (2017a). A la vista de los resultados de los artículos citados, se establecen el resto de variables explicativas como su valor esperado condicionado, refieriéndose al valor del predictor que se está examinado. Estos gráficos se realizan con la librería `h2o`[1], y concretamente, mediante la función `h2o.partialPplot()` que se aplica sobre las observaciones con las que se ha entrenado cada modelo y el nombre de cada uno de los predictores.

Por otro lado, los gráficos *ICE* se pueden considerar como una extensión de los gráficos de dependencia parcial *PDP*. La diferencia principal, es que los *PDP* muestran con una única curva cómo varía en promedio la predicción de la variable respuesta a medida que se modifica uno de los predictores. Los *ICE*, muestran cómo varía la predicción para cada una de las observaciones (una curva distinta para cada observación). Para más detalles sobre este tipo de gráficos se recomienda ver Brandon M. Greenwell (2017a) y Goldstein, Alex and Kapelner, Adam and Bleich, Justin and Pitkin, Emil (2015).

A continuación, se exponen los gráficos correspondientes a las técnicas *Random Forest*, GBM y regresión logística sobre el modelo ampliado con la variable extraída de la red social Linkedin. Para comprender el significado de los distintos gráficos de una forma similar y por tanto más amena, en este Ápendice, sólo se representaran para estos modelos en concreto, ya que su construcción se realizó con la misma librería, `h2o`, permitiendo una comparación más inmediata.

[1] La plataforma *h2o.ai* ofrece una versión interesante para la visualización de los gráficos de modelos *Machine Learning*, ver `https://www.h2o.ai/try-driverless-ai/`.

## A.1. Interpretación exhaustiva de los modelos h2o

Existe una gran variedad de metodologías para interpretar los resultados del aprendizaje automático, como pueden ser: importancia de variables mediante permutación, diagramas de dependencia parcial, contribución de cada una de las variables sobre las predicciones del modelo, etc. Muchos paquetes de `R` se centran exclusivamente en la interpretabilidad de los modelos *Machine Learning*. Uno de esos paquetes es `DALEX`, que se apoya en el paquete `pdp` para realizar gráficos de dependencia parcial, aunque también proporciona una alternativa para las variables predictoras categóricas. Además permite realizar un diagnóstico residual, comparando distribuciones de los residuos de cada modelo.

Como se concluyó en la Parte II, el modelo *XGBoost* muestra el AUC más alto para el modelo ampliado con la variable de información de RRSS, seguido del modelo GBM y luego el *Random Forest*. Sin embargo, la utilización de una métrica de validación puede ser un pobre indicador de rendimiento. Evaluar los residuos de predicción frente a los valores de la observación puede permitir identificar dónde se desvían los modelos en su precisión predictiva.

Observando los cuantiles de la Figura A.1, se puede decir que los residuos medios son más bajos para el *GBM*. Y observando los diagramas de caja, el modelo *Random Forest* tiene el valor residual absoluto medio más bajo. Por lo tanto, aunque el modelo GBM presentó el mejor AUC, en comparación al *Random Forest* y GLM, en realidad tampoco podría descartarse la eficiencia del modelo *Random Forest*. Sin embargo, también puede verse en el gráfico de la izquierda un mayor número de residuos en la cola de la distribución del *Random Forest*, lo que sugiere que puede haber un mayor número de residuos altos en comparación con el modelo GLM.

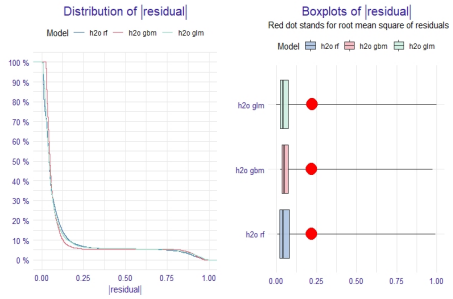



Figura A.1: Distribución de los residuos en valor absoluto.

El estudio al detalle de las variables y su interacción, por motivos de confidencialidad se ha eliminado de esta publicación.

Existe una amplia variedad de gráficos para interpretar los modelos *Machine Learning*, dependiendo de la cantidad de variables, unos funcionan mejor que otros, o por lo menos computacionalmente hablando.También, para algunas metodologías de predicción específicas existen paquetes particulares con los cuales realizar la interpretación de un modelo de forma sencilla. En cualquier caso, este es un campo que interesa estudiar con mayor profundidad.

# Bibliografía


Alan Beaulieu (2005). *Learning SQL*. O'Reilly Media, 1st ed edition.

Alonzo T.A., Pepe M.S. (2002). *Distribution-free ROC analysis using binary regression techniques*.

Andy Liaw and Matthew Wiener (2002). Classification and Regression by randomForest. *R News*, 2(3):18–22.

Aurélien Géron (2017). *Hands-On Machine Learning with Scikit-Learn and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems*. O?Reilly Media, 1 edition.

Brandon Greenwell and Bradley Boehmke and Jay Cunningham and GBM Developers (2019). *gbm: Generalized Boosted Regression Models*. R package version 2.1.5.

Brandon M. Greenwell (2017a). pdp: An R Package for Constructing Partial Dependence Plots. *The R Journal*, 9(1):421–436.

Brandon M. Greenwell (2017b). pdp: An R Package for Constructing Partial Dependence Plots. *The R Journal*, 9(1):421–436.

Breiman, Leo (2001). Random forests. *Machine learning*, 45(1):5–32.

Brian Ripley and Michael Lapsley (2019). *RODBC: ODBC Database Access*. R package version 1.3-16.

Chen, Tianqi and Guestrin, Carlos (2016). XGBoost:A Scalable Tree Boosting System. *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining - KDD 16*.

Cho, Hyunsu. (2016). Speeding up gradient boosting for training and prediction. *University of Washington*.

Christoph Molnar and Bernd Bischl and Giuseppe Casalicchio (2018). iml: An R package for Interpretable Machine Learning. *JOSS*, 3(26):786.

Deborah Nolan, Duncan Temple Lang (2014). *XML and Web Technologies for Data Sciences with R*. Use R! Springer-Verlag New York, 1 edition.

Eric D. Kolaczyk, Gábor Csárdi (2014). *Statistical Analysis of Network Data with R*. Use R! 65. Springer-Verlag New York, 1 edition.

Erin LeDell and Navdeep Gill and Spencer Aiello and Anqi Fu and Arno Candel and Cliff Click and Tom Kraljevic and Tomas Nykodym and Patrick Aboyoun and Michal Kurka and Michal Malohlava (2019). *h2o: R Interface for 'H2O'*. R package version 3.26.0.2.

Fawcett, Tom (2006). An introduction to ROC analysis. *Pattern recognition letters*, 27(8):861–874.

Friedman, Jerome H (2001). 1999 Reitz Lecture. *The Annals of Statistics*, 29(5):1189–1232.

Friedman, Jerome H and Popescu, Bogdan E and others (2008). Predictive learning via rule ensembles. *The Annals of Applied Statistics*, 2(3):916–954.

Gareth James; Trevor Hastie; Robert Tibshirani; Daniela Witten (2013). *An Introduction to Statistical Learning: With Applications in R*. Springer Texts in Statistics. Springer, hardcover edition.

Goldstein, Alex and Kapelner, Adam and Bleich, Justin and Pitkin, Emil (2015). Peeking inside the black box: Visualizing statistical learning with plots of individual conditional expectation. *Journal of Computational and Graphical Statistics*, 24(1):44–65.

Graham Williams (2011). *Data Mining with Rattle and R: The Art of Excavating Data for Knowledge Discovery.* Use R. Springer-Verlag New York, 1 edition.

Grömping, Ulrike (2009). Variable importance assessment in regression: linear regression versus random forest. *The American Statistician*, 63(4):308–319.

Ingo Feinerer and Kurt Hornik (2019). *tm: Text Mining Package.* R package version 0.7-7.

Jeroen Ooms (2014). The jsonlite Package: A Practical and Consistent Mapping Between JSON Data and R Objects. *arXiv:1403.2805 [stat.CO].*

John Harrison (2017). *RSelenium: R Bindings for 'Selenium WebDriver'.* R package version 1.7.1.

Leo Breiman, Jerome Friedman, Richard A. Olshen, Charles J. Stone (1984). *Classification and regression trees.* The Wadsworth statistics / probability series. CRC.

Malohlava, Michal and Candel, Arno (2017). Gradient boosting machine with H2O.

Manning, Christopher D. and Raghavan, Prabhakar and Schütze, Hinrich (2008). Scoring, term weighting and the vector space model. *Introduction to information retrieval*, 100:2–4.

Przemyslaw Biecek (2018). DALEX: Explainers for Complex Predictive Models in R. *Journal of Machine Learning Research*, 19(84):1–5.

Ruppert, David (2004). The Elements of Statistical Learning: Data Mining, Inference, and Prediction. *Journal of the American Statistical Association*, 99(466):1–567.

Simon Munzert, Christian Rubba, Peter Meißner, Dominic Nyhuis (2015). *Automated Data Collection with R: A Practical Guide to Web Scraping and Text Mining.* Wiley, 1 edition.

Sparck, Jones, Karen (1972). A statistical interpretation of term specificity and its application in retrieval. *Journal of documentation*, pages 11–21.

Thomas Lin Pedersen and Michaël Benesty (2019). *lime: Local Interpretable Model-Agnostic Explanations.* R package version 0.5.1.

Tianqi Chen and Tong He and Michael Benesty and Vadim Khotilovich and Yuan Tang and Hyunsu Cho and Kailong Chen and Rory Mitchell and Ignacio Cano and Tianyi Zhou and Mu Li and Junyuan Xie and Min Lin and Yifeng Geng and Yutian Li (2019). *xgboost: Extreme Gradient Boosting.* R package version 0.90.0.2.

Trevor Hastie, Robert Tibshirani, Jerome Friedman (2013). *The Elements of Statistical Learning: Data Mining, Inference, and Prediction.* Springer Series in Statistics. Springer, 2 edition.

Viswanathan, Viswa and Viswanathan, Shanthi and Gohil, Atmajitsinh and Yu-Wei, Chiu David Chiu (2016). *R: Recipes for Analysis, Visualization and Machine Learning.* Packt Publishing.

Wood, Simon N. (2017). *Generalized Additive Models.* Chapman and Hall/CRC.