Trabajo Fin de Máster

Modelo Predictivo de Siniestralidad Seguro de Coche

Berta Cadaveira Regueiro

Máster en Técnicas Estadísticas

Curso 2023-2024

Propuesta de Trabajo Fin de Máster

Título en galego: Modelo Preditivo de Siniestralidad Seguro de Coche
Título en español: Modelo Predictivo de Siniestralidad Seguro de Coche
English title: Car Insurance Accident Predictive Model
Modalidad: Modalidad B
Autor/a: Berta Cadaveira Regueiro, Universidad de Santiago de Compostela
Director/a: Rubén Fernández Casal, Universidad de A Coruña; Manuel Oviedo de la Fuente, Universidad de A Coruña
Tutor/a: Juan Manuel Mazaira Gómez, ABANCA
Breve resumen del trabajo: El objetivo del trabajo fin de máster, consiste en la construcción de un modelo de predicción de siniestralidad de los clientes que tengan contratado un seguro de coche. Para la realización de este trabajo se utilizará principalmente información financiera (Información diferente a la que tradicionalmente poseen las entidades aseguradoras) que permita explicar y predecir futuras siniestralidades con su vehículo.
Recomendaciones:
Otras observaciones:

Don Rubén Fernández Casal, profesor del departamento de Matemáticas de la Universidad de A Coruña, don Manuel Oviedo de la Fuente, profesor del departamento de Matemáticas de la Universidad de A Coruña, don Juan Manuel Mazaira Gómez, Manager de ABANCA, informan que el Trabajo Fin de Máster titulado

Modelo Predictivo de Siniestralidad Seguro de Coche

fue realizado bajo su dirección por doña Berta Cadaveira Regueiro 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 03 de Junio de 2024.

El/la director/a:
Don Rubén Fernández Casal

El/la director/a:
Don Manuel Oviedo de la Fuente

El/la tutor/a:
Don Juan Manuel Mazaira Gómez

El/la autor/a:
Doña Berta Cadaveira Regueiro

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

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

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

Agradecimientos

Con esto me gustaría agradecer a todas las personas que me han ayudado para poder llevar a cabo todo este proyecto. Por una parte, a mis compañeros de Analítica Predictiva y de Negocio de ABANCA, que me ayudaron en todo lo que precisé durante el desarrollo del trabajo. En especial a mi tutor de ABANCA Juan Manuel Mazaira Gómez y a Rubén Garrudo Díaz, por todo su apoyo, consejos y conocimientos. Por otra parte a mis tutores de la Universidad de A Coruña, Rubén Fernández Casal y Manuel Oviedo de la Fuente por su orientación y ánimos.

Por último, quería dar las gracias a mi familia, amigos y en especial a Rubén, que me han aguantado y apoyado durante todo el desarrollo desde el día uno.

Índice general

Resumen xi

1. Introducción 1
- 1.1. Motivación del problema . . . 1
- 1.2. Estado del arte . . . 2

2. Tratamiento de datos 5
- 2.1. Consultas a una base de datos relacional . . . 5
- 2.2. Construcción de la muestra de datos . . . 6
- 2.3. Análisis descriptivo inicial . . . 9
  - 2.3.1. Análisis de datos faltantes . . . 11
  - 2.3.2. Análisis de correlaciones . . . 13
  - 2.3.3. Representaciones gráficas . . . 16

3. Modelos de clasificación 21
- 3.1. Árboles de decisión . . . 22
  - 3.1.1. Árboles de regresión . . . 23
  - 3.1.2. Árboles de clasificación . . . 25
  - 3.1.3. Parámetros y sobreajuste del modelo . . . 26
- 3.2. Modelos bagging y boosting . . . 26
  - 3.2.1. Bagging . . . 27
  - 3.2.2. Random forest . . . 29
- 3.3. Boosting . . . 31
  - 3.3.1. AdaBoost . . . 31
  - 3.3.2. Gradient Boosting . . . 32
  - 3.3.3. Extreme Gradient Boosting . . . 35
  - 3.3.4. LightGBM . . . 37
- 3.4. Validación y evaluación de los modelos . . . 39

4. Resultados 43
- 4.1. Random Forest . . . 43
- 4.2. GBM . . . 45
- 4.3. XGBoost . . . 48
- 4.4. LightGBM . . . 49
- 4.5. Comparación de modelos . . . 52
- 4.6. Tratamiento del problema de desbalanceo . . . 53
  - 4.6.1. Resultados . . . 54

5. Conclusiones y líneas futuras 57
- 5.1. Conclusiones finales . . . 57
- 5.2. Líneas futuras . . . 57

Resumen

Resumen en español

El presente trabajo será llevado a cabo en la entidad *ABANCA Corporación Bancaria S.A.*, y se centrará en el problema predictivo de siniestralidad en el seguro de automóvil. En la institución no existe un modelo predictivo para la siniestralidad en este seguro, no obstante, se cree que sería de especial utilidad a la hora de asignar las primas a los clientes. Por esta razón se decide construir un modelo predictivo utilizando una herramienta estadística conocida como aprendizaje estadístico.

Primeramente, se extraen los datos que serán usados para construir el modelo, y posteriormente los mismos son tratados. Se elabora un análisis inicial sobre los siniestros y la variable objetivo MSINIESTRO, que identifica los siniestros en los próximos 6 meses. Además se lleva a cabo un análisis de datos faltantes y de correlaciones entre las variables.

Una vez los datos han sido depurados, se comienza la construcción del modelo. Se exploran cuatro modelos predictivos: un *random forest*, un *GBM*, un *XGBoost* y un *LightGBM*. Tras su construcción, evaluación y comparación, se concluye que el modelo que ofrece unos mejores resultados según las métricas utilizadas es el XGBoost.

Finalmente, es importante destacar que los resultados no son demasiado buenos y que a futuro sería interesante valorar otras opciones de variables, metodologías o incluso pensar en otro planteamiento del problema.

English abstract

The present work will be carried out at *ABANCA Corporación Bancaria S.A.*, and will focus on the predictive problem of claims in automobile insurance. In the institution there is no predictive model for claims in this insurance, however, it is believed that it would be particularly useful when allocating premiums to customers. For this reason, it was decided to build a predictive model using a statistical tool known as machine learning.

First, the data that will be used to build the model are extracted and then processed. An initial analysis is performed on the claims and the target variable MSINIESTRO, which identifies the claims in the next 6 months. In addition, an analysis of missing data and correlations between variables is carried out.

Once the data has been cleaned, model building begins. Four predictive models are explored: a *random forest*, a *GBM*, a *XGBoost* and a *LightGBM*. After their construction, evaluation and comparison, it is concluded that the model that offers the best results according to the metrics used is the XGBoost.

Finally, it is important to point out that the results are not very good and that in the future it would be interesting to evaluate other options of variables, methodologies or even to think of another approach to the problem.

Capítulo 1

Introducción

ABANCA Corporación Bancaria S.A. es una entidad financiera con sede en Galicia, la misma cuenta con diferentes áreas de trabajo, entre las que se encuentra *Inteligencia de Clientes*. Dicha unidad es de gran importancia para la institución, pues entre sus funciones destaca el análisis de datos. No obstante, también elabora cierta investigación comercial para mejorar la comercialización de los distintos productos que puede ofrecer ABANCA. El departamento a su vez cuenta con tres subdepartamentos, siendo el más grande *Modelos de Analítica Avanzada*. El mismo está conformado por tres equipos que se dedican a construir diferentes modelos predictivos entre los que se pueden destacar:

- **Modelos familias**: Están dedicados a clientes definidos como personas físicas. Su objetivo es obtener la probabilidad que tienen los clientes que no tienen contratado un producto con ABANCA de adquirir el producto en cuestión. Entre los mismos se encuentran modelos de inversión, de ahorro, de consumo, de planes de pensiones, de seguros...
- **Modelos empresas**: Están enfocados a personas jurídicas y su objetivo se basa en captar posibles clientes que necesiten ser financiados y que en ese momento no lo están siendo por ABANCA. Para ello, debe tenerse en cuenta la vinculación del cliente, la rentabilidad, etc.
- **Aldia empresas**: Mediante técnicas de *web scraping* se recoge información online, y se transforma la misma en eventos comerciales que se traspasan al gestor. Luego éste realiza acciones comerciales sobre clientes con potencial.
- **Redes empresas**: El objetivo es construir una red que representa las relaciones entre empresas, para detectar posibles oportunidades comerciales.
- **Clientes valor**: Se intenta crear una matriz de afinidad de clientes para detectar grupos de clientes. Su objetivo es conocer la penetración de un producto en cada grupo o su afinidad a cada producto.

En este caso, la elaboración de este proyecto se ha llevado a cabo en el equipo de *Modelos de Producto*. El mismo, con la ayuda de técnicas estadísticas, trata de construir modelos predictivos para los productos que ofrece la entidad.

1.1. Motivación del problema

Como se acaba de mencionar, entre los diferentes modelos de productos se encuentran aquellos dedicados a los seguros. Desde hace varios años ABANCA cuenta con varios seguros propios que se ofrecen a través de su filial *ABANCA Seguros Generales*. El objetivo de dichos modelos es estimar la

probabilidad de contratación en los seguros existentes y entre los mismos destaca el seguro de automóvil.

Llegados a este punto, es conocido que las aseguradoras cuentan con mucha información acerca de sus asegurados. Si hacen uso de la misma pueden planificar pólizas con precios personalizados para cada uno de ellos. Les interesa proceder así para intentar aumentar los beneficios y ser justas con los asegurados, es decir, para penalizar un mal comportamiento por parte de los clientes y favorecer el bueno. Dicho concepto se conoce como *pay as you drive*, descrito por Hultkrantz et al. (2012). Además, Guillen et al. (2018) mencionan los beneficios de tener en cuenta información sobre la exposición de los conductores al riesgo o sobre los hábitos de los mismos.

En este contexto, tal y como indican Hanafy and Ming (2021), tener en cuenta la predicción de los siniestros es clave a la hora de estimar las primas de los seguros de automóviles. Además, la misma debe ser bastante precisa, pues de tener una mala predicción, se estaría aumentando el precio a un cliente 'bueno', mientras que se le estaría reduciendo al 'malo'. Esta operación claramente perjudica al cliente, pero también a la propia entidad.

Una vez comprendida la necesidad de tener en cuenta los siniestros, se debe comentar que ABANCA no cuenta con un modelo de predicción de siniestralidad en estos momentos. La entidad para establecer las primas, se basa en un *score financiero*, un sistema de evaluación automático que se calcula a partir de diferentes variables bancarias de los clientes. Esta puntuación cuenta con siete categorías desde el 0 hasta el 6, siendo 0 la mejor puntuación y 6 la peor. De esta forma cada cliente tiene una puntuación y se le ofrece una prima en función de la misma.

Ante dicha situación, la entidad comprende que utilizar únicamente esta puntuación financiera para otorgar las primas no es lo más adecuado. Por tanto, piensan que la construcción de un modelo predictivo que detecte los posibles siniestros del seguro de automóvil puede ser de utilidad en este caso.

Entonces el objetivo del presente proyecto será cubrir la falta de un modelo predictivo de siniestralidad haciendo uso de herramientas estadísticas, entre las que se encuentra el *machine learning*. De esta forma se busca predecir si los clientes tendrán un siniestro en los próximos 6 meses, para que posteriormente esa información pueda incorporarla la aseguradora. Cabe destacar que durante todo el proyecto el término *siniestro* se referirá a cualquier tipo de parte que el cliente pueda informar a la aseguradora. Por ejemplo, partes por vandalismo, por colisión entre vehículos, por fenómenos atmosféricos, entre muchos otros.

1.2. Estado del arte

A continuación se va a realizar una revisión por diferentes estudios previos sobre la predicción de siniestros. En los mismos, los autores han utilizado diferentes variables predictoras, han implantado modelos de *machine learning*, y han discutido sus ventajas y desventajas. Notar que todas las métricas y modelos mencionados en esta sección se han descrito en la Sección 3.4.

Por un lado, Fauzan and Murfi (2018) comparan un modelo XGBoost con otras construcciones como un Random Forest o un modelo basado en el algoritmo Adaboost para predecir la siniestralidad. Al final concluyen que el modelo basado en XGBoost es el que ofrece mejores resultados. Algo similar aseguran Abdelhadi et al. (2020), pues tras construir diferentes modelos como un árbol de decisión y un modelo XGBoost, obtienen resultados parecidos a los descritos por Fauzan and Murfi (2018).

Por otro lado, Pesántez-Narvaez et al. (2019), utilizando una muestra de datos más pequeña, explican que un modelo de regresión logística ha dado lugar a mejores resultados. Además mencionan que

es más interpretable que un modelo XGBoost. Más tarde, Hanafy and Ming (2021) intentan predecir los siniestros pero con un conjunto de datos que ellos consideran masivo. De nuevo explican que modelos basados en árboles proporcionan mejores resultados que el modelo basado en el algoritmo XGBoost.

A la hora de evaluar los modelos construidos, los autores recurren a diversas métricas. Por ejemplo, Abdelhadi et al. (2020) se basa en el Accuracy o en el área bajo la curva ROC (AUC), mientras que Hanafy and Ming (2021), además de utilizar esas dos, también hace uso de la sensibilidad y la especificidad.

Destacar que la gran mayoría de los proyectos cuentan con valores faltantes en su muestra de datos. En muchos de ellos han decidido imputarlos utilizando la media o la moda, proceso que en ocasiones puede llevar a estimaciones no muy adecuadas. En esta ocasión no se procederá de dicha forma. Como se puede ver en la Sección 2.3, en un inicio se decide mantener algunas variables con valores no observados.

En cuanto a las variables que han utilizado, se tienen variables como la edad o sexo del asegurado, la edad del vehículo, porcentaje de kilómetros recorridos por los clientes, entre muchas otras referidas a características del asegurado o del automóvil. Pesántez-Narvaez et al. (2019) comenta que su modelo tiene como variable más importante el porcentaje del total de kilómetros recorridos en zonas urbanas, seguida de la edad del asegurado.

Sin embargo, en ninguno de los estudios se muestran variables bancarias relacionadas con hipotecas, préstamos, saldos, tarjetas, etc, que serán utilizadas en el trascurso del proyecto. Las mismas serán de especial interés a la hora de analizar si son realmente relevantes en los modelos construidos para predecir la siniestralidad.

Conocido el estado del problema predictivo de siniestralidad en la actualidad, se está en condiciones de comenzar el tratamiento de datos que será especialmente importante para la construcción de los modelos. Esto se debe a que si se entrena un modelo predictivo con un conjunto de datos que no ha sido depurado previamente, se tendrán modelos con baja calidad o precisión. Por tanto, éstos no serán realmente útiles para predecir la siniestralidad.

Capítulo 2

Tratamiento de datos

Primeramente se necesita construir el conjunto de datos que se usará para entrenar y evaluar los modelos predictivos. Así, puesto que la entidad bancaria cuenta con su información almacenada en bases de datos, se va a hacer uso de SQL a través de la plataforma de *Teradata* y de la librería `dplyr` de `R`.

2.1. Consultas a una base de datos relacional

Antes de comenzar, se elabora un pequeño resumen de funciones que se usarán posteriormente para elaborar las consultas de SQL necesarias para construir el conjunto de datos.

- **SELECT**: Utilizado para seleccionar registros de una base de datos. Además pueden seleccionarse columnas concretas o todas ellas sin más que añadir un asterisco tras SELECT.
- **SELECT DISTINCT**: Utilizado para seleccionar los registros de forma que los valores de las columnas escogidas sean diferentes.
- **FROM**: Utilizado para indicar la procedencia de las columnas seleccionadas, puede ser una tabla o una subconsulta.
- **WHERE**: Utilizado para filtrar los datos, es decir, se extraen solamente aquellos que cumplen las condiciones declaradas.
- **ORDER BY**: Se usa para ordenar los registros por una o varias columnas. Además pueden ser ordenados de forma ascendente con ASC o descendente con DESC.
- **GROUP BY**: Utilizado para agrupar las diferentes variables según sus niveles.

Además de dichas funciones se tienen otras muchas como pueden ser BETWEEN, AND, OR, IN, LIKE, HAVING, pero las mismas no van a ser especificadas, si se quisiera conocer su utilidad se puede recurrir a Beaulieu (2009). Si es necesario combinar la información de varias tablas que tienen algún campo en común se puede utilizar alguna de las siguientes funciones.

- **INNER JOIN**: Proporciona los registros de todas columnas siempre que haya una coincidencia entre las columnas de las dos tablas.
- **LEFT JOIN**: Da lugar a todos los registros de la tabla de la izquierda y únicamente los registros de la tabla de la derecha que coincidan. Si hay valores en la tabla de la izquierda que no aparecen en la de la derecha, aparecen como valores faltantes.

- **RIGHT JOIN**: Procede de manera análoga al caso anterior, simplemente que ahora se mantienen todas la filas de la tabla de la derecha.
- **OUTER JOIN**: Devuelve todos los registros de las dos tablas, y aparecerán valores perdidos cuando no exista coincidencia entre las tablas.

Una vez entendidas las diferentes funciones hay que destacar que la librería `dplyr` también cuenta con funciones que llevan a cabo lo descrito arriba, pero por evitar la repetición, no se entrará en detalles de la misma. No obstante si se quisiera conocer dicho paquete se puede recurrir a Alonso (2022). Ahora se está en condiciones de explicar en el siguiente apartado la obtención de la muestra de datos que se usará en las posteriores secciones para entrenar y validar los diferentes modelos.

2.2. Construcción de la muestra de datos

Para empezar es importante comentar que debido a los datos proporcionados por la empresa, como intervalo temporal se ha tomado el período de un año, en este caso, desde septiembre de 2022 a agosto de 2023. Sin embargo, para poder construir la variable objetivo es necesaria la información en los 6 meses posteriores, por lo que en un primer momento se contará con la información hasta febrero del 2024. Al final, se tendrá la observación de un cliente a lo sumo 12 veces, una por cada mes del período temporal seleccionado. A continuación se expone la construcción de la muestra de datos que se utilizará en los próximos capítulos.

1. Extracción del CLIENTE_ID (número de identificación de cliente) de los clientes con seguro de automóvil *ABANCA Seguros Generales* cuyo seguro se encontrara vigente antes del 29-02-2024.
2. Extracción de la información referida a los diferentes vehículos, entre la misma se encuentra:
   - **MATRICULA**: Matrícula del vehículo asegurado.
   - **CLIENTE_ID**: Número de identificación del cliente.
   - **MARCA**: Marca del automóvil.
   - **MODELO**: Modelo del automóvil.
   - **MOTOR**: Tipo de motor del automóvil.
   - **POTENCIA**: Potencia en caballos de vapor del vehículo.
   - **PUERTAS**: Número de puertas del vehículo.
   - **PRECIO**: Precio estimado del vehículo en euros.
   - **FECHA_EFECTO_INICIAL**: Fecha de efecto inicial de la póliza.
   - **FECHA_CANCELACION**: Fecha de cancelación de la póliza. Cuando no se ha cancelado aparece un valor faltante.
   - **ANTIGUEDAD_VEHICULO**: Antigüedad del vehículo en días.
3. Extracción de los siniestros registrados entre el 01-09-2022 y 29-02-2024. Cada fila se corresponde a un siniestro donde se recoge, su fecha de ocurrencia traspasada al último día del mes, el identificador del cliente y la matrícula del vehículo.
4. Cada observación se corresponde con una combinación de cliente y vehículo. Entonces se separan las combinaciones en función de si han tenido siniestro o no, para tratar cada conjunto por separado.
5. Creación de una tabla de datos con los 18 meses seleccionados en el intervalo temporal.

6. Extracción de las diferentes variables bancarias acerca de los clientes con las que cuenta la institución. Entre ellas se encuentran variables como las siguientes:

   - **SEXO**: Sexo del cliente.
   - **GRUPO**: Grupo de empleo del cliente. Cuenta con categorías como: pensionistas, ocupaciones elementales, directores y gerentes, en busca del primer empleo, trabajadores de servicios de restauración...
   - **AMBITO_CLIENTE**: Ámbito del cliente en función de su padrón municipal. Tiene las categorías: rural, semiurbano, ciudad, gran ciudad, área metropolitana y desconocido.
   - **NIVEL2**: Indica si el cliente es autónomo o familia. Sus clases son: familia, autónomo o profesión liberal.
   - **SEGMENTO_ID**: Segmento del cliente, presenta las clases: menores, promotores autónomos, autónomos y profesiones liberales, jóvenes, adultos 1, adultos 2, senior 1, senior 2 y autónomos y profesiones liberales (potenciales).
   - **TARJERTAS_DMK_PT**: Saldo medio de las tarjetas del cliente.
   - **MESES_ANTIGUEDAD_CLIENTE**: Antigüedad en meses de los clientes.
   - **SCORE_FINANCIERO**: Puntuación financiera asignada a cada cliente. Toma las clases desde el 0 hasta el 6, siendo 0 la mejor puntuación y 6 la peor.
   - **NPS**: Media de la puntuación de las encuestas realizadas por el cliente.
   - **PASIVO_DMK_CT**: Saldo medio de pasivo en el último mes en los contratos como cualquier titular.
   - **AUTORIZADO_BE**: Suma de los saldos dispuestos y disponibles del cliente en el Banco de España.

   Además de dichas variables se tienen otras muchas referidas a los distintos seguros existentes o a temas bancarios como pueden ser préstamos, hipotecas o saldos.

7. Construcción del conjunto sin siniestros. Se comienza cruzando las observaciones con las fechas de la ventana temporal. Así se obtiene un conjunto donde cada combinación se repite 18 veces, una por cada mes. Mencionar que el hecho de que cada cliente y vehículo se repita 18 veces provocará una cierta dependencia en los datos. Posteriormente se incluye la información de los vehículos extraída en el paso 2, y se eliminan todos los registros donde la fecha de observación se encuentra fuera del período de vigencia del seguro.

   Luego se añade la información de los clientes por fecha de observación, y se crea la variable SINIESTRO. Ésta será utilizada más adelante para crear la variable objetivo, e indica con un 1 si el cliente ha tenido un siniestro en esa fecha y con un 0 en otro caso. En este caso como son clientes y vehículos sin siniestros, únicamente toma ceros.

8. Construcción del conjunto con siniestros. Se comienza con las combinaciones junto con la fecha de ocurrencia del siniestro traspasada al último día del mes. A esa tabla se le eliminan todos los siniestros excepto los primeros de cada cliente. Esto se hace para alinear la muestra de construcción del modelo con la muestra en la que se aplicará el mismo. Es decir, se pretende utilizar dicho modelo para conocer en clientes nuevos que quieran contratar el seguro la probabilidad que tienen de tener un siniestro. Así pues, si un cliente ya ha informado un siniestro, su información a posteriori ya no interesa.

   Ahora como en el caso anterior, se cruza la tabla con el intervalo temporal seleccionado y se crea la variable SINIESTRO mencionada anteriormente. Luego, no interesa la información posterior al primer siniestro de cada cliente, por lo que se eliminan todos los registros siguientes de cada cliente y se añade la información del paso 2. Así de igual manera que con el otro conjunto, se eliminan todos los registros donde la fecha de observación se encuentra fuera del período de vigencia del seguro, y se añade la información bancaria de los clientes.

9. Unión de los dos subconjuntos de datos, obteniendo un único conjunto de datos.

10. Añadido de variables que puedan resultar interesantes como pueden ser variables referidas a los impagos de los clientes. Por ejemplo, MOROSO_ACTUAL que indica con un 1 si el cliente en esa fecha de observación es moroso o con un 0 si no; o IMPAGO_ACTUAL_NUEVO que indica la presencia de un impago en esa fecha de observación.

11. Eliminación de registros de algunos clientes de los que no se recoge información porque no pueden ser usados para modelar por diferentes políticas de la entidad.

12. Creación de la variable objetivo del problema, MSINIESTRO, que toma el valor 1 cuando el cliente en los 6 meses posteriores a la fecha de observación presenta un siniestro, y un 0 en otro caso. Para comprender mejor como se crea, en la Figura 2.1 se ha elaborado un esquema de su construcción. Posteriormente, se eliminan los registros donde el cliente presenta el siniestro pues una vez construida la variable MSINIESTRO estos casos ya no nos interesan.

Figura 2.1: Esquema de la construcción de la variable objetivo MSINIESTRO.

13. Eliminación de los registros con fechas de observación entre 30-09-2023 y el 29-02-2024, para así obtener la foto final con un período anual de las combinaciones de cliente y vehículo. De esta forma, se tiene una muestra de datos con 403952 observaciones y 377 variables referidas a 41440 clientes que presentan un total de 35157 unos en la variable objetivo MSINIESTRO.

Tras la construcción del conjunto de datos y puesto que es un proceso largo y con muchos pasos, se muestra a continuación en la Figura 2.2 un diagrama de flujo de todo el proceso con la intención de mejorar la explicación del mismo.

Para finalizar y antes de comenzar el análisis del conjunto de datos, se eliminan muchas de las variables predictoras ya que no son de utilidad para el problema en cuestión, por ser variables referidas a identificadores o fechas, como pueden ser, identificadores del contrato, fechas de las pólizas, etc. Luego, es importante examinar la clase de cada variable, pues quizás alguna esté definida incorrectamente y sea mejor tenerla en cuenta de otra forma. Por una parte, se tienen variables como CLIENTE_ID, PUERTAS o AMBITO_CLIENTE entre otras, que se encuentran como variables continuas pero que deberían reconocerse como variables de tipo factor, al igual que todas las variables recogidas en `R` como `character`.

Por otra parte, se tiene una variable denominada NPS descrita en la Sección 2.2, la misma se recoge como variable continua, pero se decide que es mejor recodificarla a tipo factor, definiendo 4 niveles: DETRACTOR, NEUTRO, PROMOTOR y NULO. Así, si la variable toma valores de puntuación menores o iguales a 6, será DETRACTOR, si se encuentra entre 6 y 9, NEUTRO, y si es mayor o igual a 9, PROMOTOR. Por último, si presenta valores faltantes, será NULO.

Además, se crean dos variables referidas a la comunidad autónoma a la que pertenece el cliente (AUTONOMIA_ID) y a si el cliente reside en Galicia o no (IN_GALICIA). Esto se ha llevado a cabo porque se elimina la variable PROVINCIA_ID pues es una variable categórica con demasiadas clases.

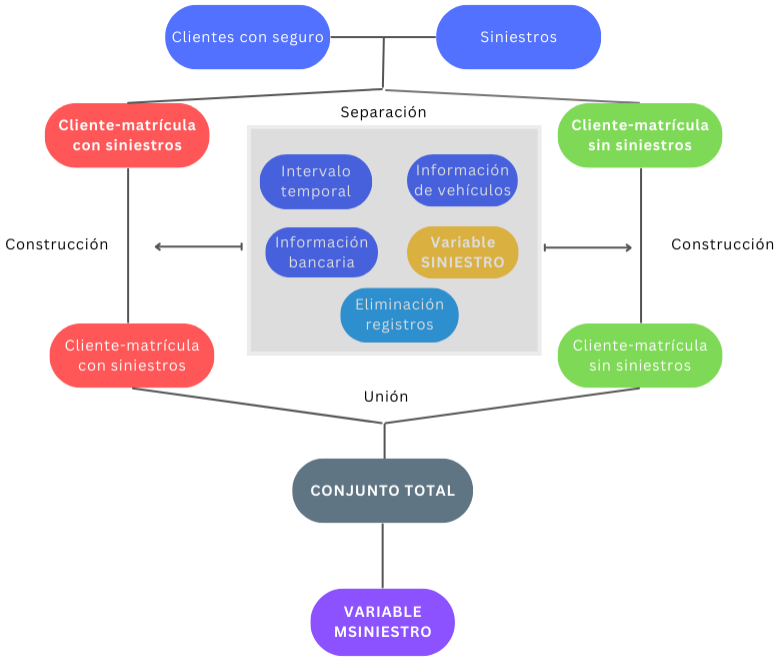

Figura 2.2: Diagrama de flujo del proceso de construcción del conjunto de datos.

2.3. Análisis descriptivo inicial

Una vez obtenido el conjunto de datos que se usará para entrenar los diferentes modelos, interesa elaborar un análisis inicial sobre las variables. La intención de dicho análisis es obtener información previa sobre las propias variables y sobre su comportamiento.

Es interesante observar la que es probablemente la variable más importante de todo el estudio, la variable objetivo MSINIESTRO. Para ello, se analiza el porcentaje de observaciones de cada uno de los valores que toma la misma, y como se puede ver en la Figura 2.3 se tiene que el 91.30 % de las observaciones son ceros, a diferencia del 8.70 % que toma como unos.

A la vista del gráfico de la Figura 2.3 es conveniente decir que el conjunto de datos que se está considerando presenta un diseño muy desbalanceado. Esto quiere decir, que la variable respuesta que tiene dos categorías, presenta muchos más casos de una que de otra. Se tiene un gran número de ceros que identifican el no siniestro en los próximos 6 meses a diferencia de los 1 que identifican siniestro en los posteriores 6 meses. Por este motivo, la precisión y eficacia de los modelos construidos pueden verse perjudicadas, por lo que quizás sea necesario tratar el problema con algún método previo al ajuste de los mismos. El tratamiento de este problema, se ha llevado a cabo en la Sección 4.6.

A continuación es interesante realizar un gráfico sobre el número de siniestros por mes que recoge el conjunto de datos. El objetivo es analizar en que meses se han informado de un mayor y un menor número de siniestros. Así se tiene el gráfico de la Figura 2.4 donde claramente se muestra que los meses con más siniestros son de abril a agosto, y el mes con menos siniestros informados es septiembre.

Figura 2.3: Gráfico de barras del porcentaje de observaciones que toma cada valor de la variable MSINIESTRO.

Figura 2.4: Número de siniestros informados por mes desde septiembre 2022 a agosto 2023.

Figura 2.5: Evolución mensual del porcentaje de siniestros del total de vehículos asegurados por mes desde septiembre 2022 a agosto 2023.

Además también es importante elaborar un gráfico sobre el porcentaje de siniestros del total de vehículos asegurados por mes. Su intención es captar los meses donde se recoge una mayor tasa de siniestralidad y los meses donde la misma es menor. Para ello se procedió de manera análoga al gráfico anterior extrayendo el gráfico de la Figura 2.5. Claramente se muestra que los meses con mayor tasa de siniestralidad son los meses de octubre a diciembre y de abril a junio, y los que tienen un menor número son los meses de febrero y marzo, junto con septiembre.

Es importante mencionar que eliminando el primer mes (septiembre de 2022) la diferencia en las tasas de siniestralidad entre los meses no son muy notables pues el porcentaje no varía más de un 0.5 %. Además, si se tuviese una serie más larga probablemente se mantendrían más o menos constantes en-

torno a dicho porcentaje. Los meses de octubre y diciembre a pesar de tener unas de las tasas más altas, en la Figura 2.4 se mostró que el número de siniestros no era tan elevado. Ésto viene provocado porque el número de seguros en esas fechas era inferior al número de seguros en los meses siguientes.

2.3.1. Análisis de datos faltantes

Finalizado el análisis de las variables referidas a los siniestros, es interesante tratar uno de los problemas más comunes en los conjuntos de datos de grandes dimensiones, el problema de los valores faltantes. Haciendo uso de la librería `dplyr`, se obtiene una lista de las diferentes variables del conjunto de datos junto con el número de valores faltantes que presentan. A continuación se representa en un gráfico de barras en la Figura 2.6 el porcentaje de valores no observados de aquellas variables que los tienen.

Para comenzar, muchas presentan estos datos perdidos debido al cruce de las tablas en la construcción del conjunto de datos. Un claro ejemplo de este tipo de variables son los *leads*. Éstas son variables creadas para referirse a ciertas características que cumplen los clientes y que son especialmente útiles cuando se busca crear un modelo de propensión. Las mismas ayudan a identificar posibles clientes a los que les podría interesar el producto y que en ese instante no poseen en *ABANCA Seguros Generales*.

Un ejemplo sería una variable que identifique con un 1 el pago de peajes y con un 0 si no se ha detectado nada, de esta forma si el valor es un 1 es bastante probable que el cliente tenga un vehículo. Entonces si esa variable tiene un valor perdido es que no se ha detectado dicha característica y por tanto la misma debe tomar el valor 0. Mencionar que la función de dichas variables para el proyecto es distinta a la comentada, ya que en este caso son de utilidad al añadir más información sobre los propios clientes.

Entonces teniendo en cuenta la razón del valor faltante en estas variables, se ha decidido sustituir los valores de todas las que verifican dicho motivo por un 0 ya que realmente su valor sería ese. Algo similar sucede con variables categóricas, por ejemplo con la variable NPS. Ésta fue categorizada al inicio de la Sección 2.3 y si tiene un valor no observado es porque dicho cliente no ha realizado encuestas, por lo que debe tomar la clase definida como NULO. Esto mismo sucede con otras variables, por lo que se procede así.

Finalmente, tras elaborar estos cambios, solamente quedan unas pocas variables con valores perdidos y que son referidas a características del vehículo. En este caso, se tienen 15 variables con muchos datos perdidos, muy próximo al 100 % de las observaciones. Las mismas se podrían eliminar por contar con un número tan elevado de valores faltantes, sin embargo, en este caso se ha decidido mantener dichas variables por el momento, y una vez ajustados los modelos se valorará si eliminarlas o no.

Por otra parte, se tienen dos variables restantes, definidas en la Sección 2.2, que son PRECIO y ANTIGUEDAD_VEHICULO. Su porcentaje de valores perdidos es bastante pequeño, ya que solamente cuentan con 302 y 326 valores no observados, respectivamente. En este contexto, se decide también mantener ambas variables tal y como están, puesto que el número de valores perdidos es demasiado pequeño en comparación con el tamaño de la muestra (403952).

Así tras el trabajo realizado sobre los valores faltantes, se obtiene un conjunto con 327 variables donde solamente algunas de ellas referidas a características propias del vehículo presentan valores faltantes. Llegados a este punto, se está en condiciones de elaborar un análisis de correlaciones de las mismas.

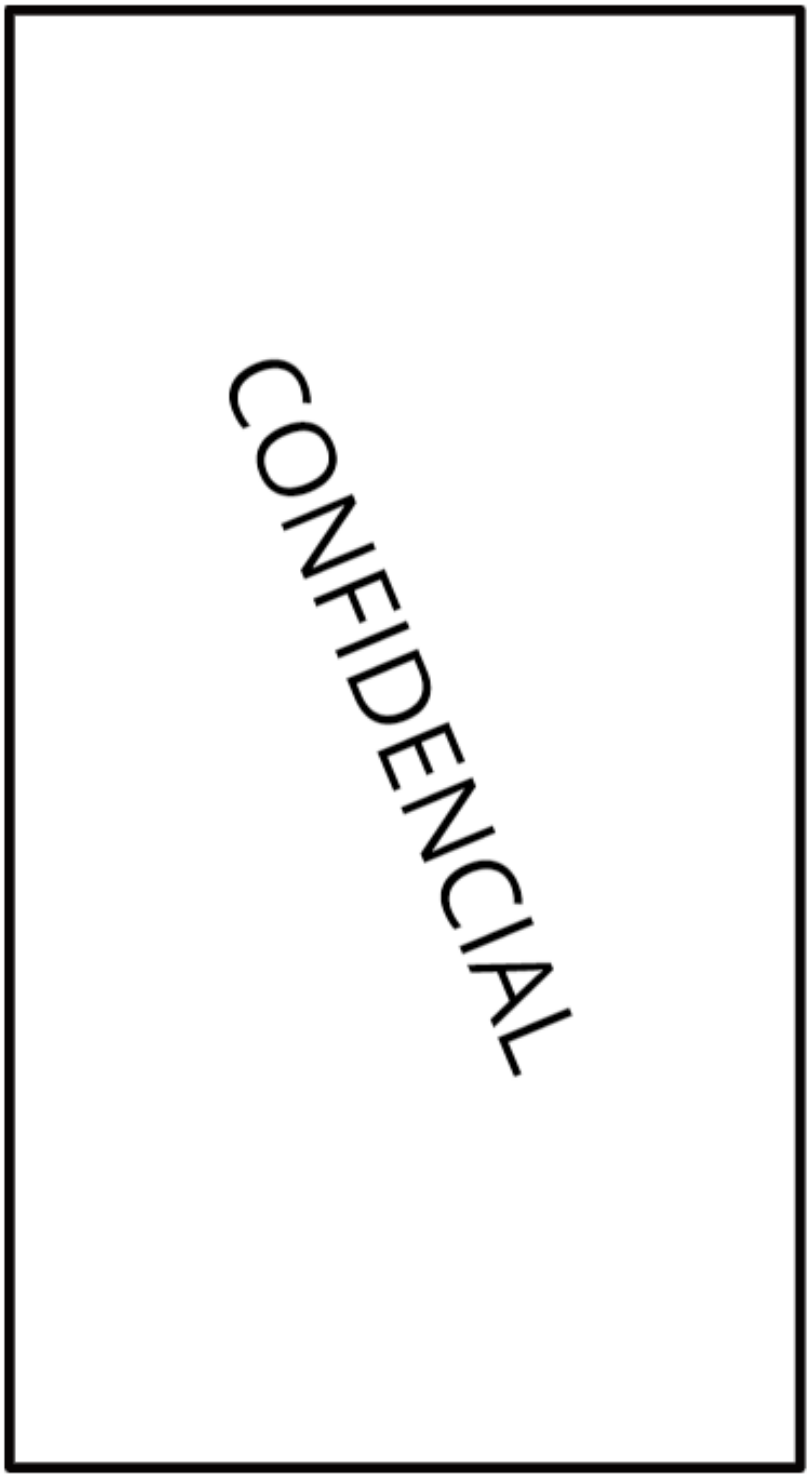

Figura 2.6: Gráfico de barras del porcentaje de valores faltantes por variables.

2.3.2. Análisis de correlaciones

Para el estudio es conveniente elaborar un análisis de correlaciones entre las variables explicativas continuas. Esto se debe a que si se crea un modelo donde varias variables explicativas presentan una correlación elevada podrían aparecer problemas de colinealidad, lo que llevaría a estimaciones incorrectas o muy sesgadas. Primeramente se calculará la correlación lineal de Pearson para cada par de variables y dicho concepto se define a continuación.

El coeficiente de correlación de Pearson se utiliza para medir la correlación lineal entre dos variables continuas y fue introducido en Pearson (1896). Si se consideran X e Y como dos variables continuas, se define el coeficiente como

$$\rho = \frac{\text{cov}(X,Y)}{\sigma_X \sigma_Y}, \tag{2.1}$$

donde $\text{cov}(X,Y)$ es la covarianza entre las variables y σ_X, σ_Y las desviaciones típicas.

En la práctica si se quisiera estimar dicho coeficiente, entre dos variables que toman los siguiente valores $\{(x_1,y_1),\ldots,(x_n,y_n)\}$, éste se puede calcular como

$$r_{xy} = \frac{\sum_{i=1}^{n}(x_i-\bar{x})(y_i-\bar{y})}{\sqrt{\sum_{i=1}^{n}(x_i-\bar{x})^2}\sqrt{\sum_{i=1}^{n}(y_i-\bar{y})^2}}, \tag{2.2}$$

donde $\bar{x},\bar{y}$ representan las medias muestrales de x e y, respectivamente.

Teniendo en cuenta esta definición, el coeficiente toma valores entre el -1 y el 1, de forma que 1 supone una correlación exacta positiva, -1 correlación exacta negativa y 0 no correlación lineal. Notar que dicho valor únicamente aporta información acerca de las relaciones lineales entre las variables.

Volviendo a las variables, en este caso se cuenta con muchas que tienen un carácter similar. Por ejemplo, TARJERTAS_DMK_CT y TARJERTAS_DMK_PT son muy parecidas ya que se refieren al saldo medio de tarjetas en el último mes, en los contratos como cualquier titular y como primer titular, respectivamente. Por esto mismo, cuando se obtienen las correlaciones en `R` se obtiene una lista bastante larga de correlaciones altas. Para ejemplificar lo obtenido, en la Tabla 2.1 se puede apreciar una pequeña parte de los pares de variables que tienen en valor absoluto una correlación mayor que 0.7.

Variable1	**Variable2**	**Correlación**
IN_LEAD_4_HOGAR	IN_LEAD_6_VIDA_RIESGO	1
TARJERTAS_DMK_PT	TARJERTAS_DMK_CT	1
SEGUROS_AHORRO_DMK_PT	SEGUROS_AHORRO_DMK_CT	0.99774
CAPITAL_ASEG_AHORRO	PRIMA_NETA_ANUAL_AHORRO	0.99478
ACTIVO_DMK_CT	DISPUESTO_AB	0.99235
AUTORIZADO_BE	DISPUESTO_BE	0.98970

Tabla 2.1: Subconjunto de las variables con correlación en valor absoluto superior al 0.7.

Además de la tabla, se ha creado un mapa de calor que se puede ver en la Figura 2.7 donde se han coloreado únicamente aquellas que superan el 0.7 en valor absoluto. El rojo representa una correlación

de 1 y el azul de 0.7. Dicho gráfico es útil a la hora de ver las correlaciones altas que existen y como de fuertes son las mismas, además ayuda a comprender mejor la muestra de datos que se va a utilizar.

Figura 2.7: Mapa de calor de las correlaciones entre las variables de la muestra de datos.

Ahora atendiendo a los resultados, se decide eliminar de cada par de variables altamente correlacionadas aquella que tenga una menor correlación con la variable respuesta. Para medir esta correlación con la variable respuesta, se utiliza la correlación de distancia que se define a continuación. Notar que se utiliza dicha medida ya que la variable respuesta es una variable binaria que tiene dos clases $\{0, 1\}$.

La correlación de distancia ($\mathcal{R}$) es una medida útil a la hora de analizar la dependencia entre dos vectores y fue introducida por Székely et al. (2007). Para explicar esta noción es necesario comentar previamente la distancia de covarianza, relacionada con el siguiente test de hipótesis entre dos vectores X e Y

$$\begin{cases} H_0 : \phi_{X,Y} = \phi_X \phi_Y \\ H_1 : \phi_{X,Y} \neq \phi_X \phi_Y, \end{cases} \tag{2.3}$$

donde ϕ_X, ϕ_Y y $\phi_{X,Y}$ son las funciones características de X, Y y el vector aleatorio (X, Y), respectivamente.

La función característica de una variable aleatoria X, Grimmett and Stirzaker (2001), es la función $\phi_X : \mathbb{R} \to \mathbb{C}$ definida por

$$\phi_X(t) = E\left[e^{itX}\right], \tag{2.4}$$

donde $t \in \mathbb{R}$.

Así, en el caso de un vector aleatorio (X, Y), la Ecuación (2.4) sería como sigue

$$\phi_{(X_1,\ldots,X_n)}(t_1, \ldots, t_n) = E\left[e^{i(t_1X_1+\ldots+t_nX_n)}\right]. \tag{2.5}$$

Entonces, si X e Y son dos variables aleatorias con sus primeros momentos finitos, la distancia de covarianza se define como la raíz positiva de

$$\mathcal{V}^2(X, Y) = \|\phi_{X,Y}(t, s) - \phi_X(t)\phi_Y(s)\|^2, \tag{2.6}$$

y la distancia de varianza como la raíz positiva de

$$\mathcal{V}^2(X) = \mathcal{V}^2(X,X) = \|\phi_{X,X}(t,s) - \phi_X(t)\phi_X(s)\|^2. \tag{2.7}$$

Por tanto partiendo de las anteriores ecuaciones, se puede definir la correlación de distancia como la raíz positiva de

$$\mathcal{R}^2(X,Y) = \begin{cases} \frac{\mathcal{V}^2(X,Y)}{\sqrt{\mathcal{V}^2(X)\mathcal{V}^2(Y)}}, & \mathcal{V}^2(X)\mathcal{V}^2(Y) > 0, \\ 0, & \mathcal{V}^2(X)\mathcal{V}^2(Y) = 0. \end{cases} \tag{2.8}$$

Si se quisiera estimar dicho valor para una muestra de la forma $(X,Y) = \{(X_k, Y_k) : k = 1, \ldots, n\}$ procedente de dos vectores aleatorios $X \in \mathbb{R}^p$ e $Y \in \mathbb{R}^q$, es necesario introducir la siguiente notación.

$$a_{kl} = \|X_k - X_l\|_p, \qquad \bar{a}_{k\cdot} = \frac{1}{n}\sum_{l=1}^{n} a_{kl}, \qquad \bar{a}_{\cdot l} = \frac{1}{n}\sum_{k=1}^{n} a_{kl}, \tag{2.9}$$

$$\bar{a}_{\cdot\cdot} = \frac{1}{n^2}\sum_{k,l=1}^{n} a_{kl}, \qquad A_{kl} = a_{kl} - \bar{a}_{k\cdot} - \bar{a}_{\cdot l} + \bar{a}_{\cdot\cdot}, \tag{2.10}$$

donde $k, l = 1, \ldots, n$. Además de forma análoga se define $b_{kl} = |Y_k - Y_l|_q$ y $B_{kl} = b_{kl} - \bar{b}_{k\cdot} - \bar{b}_{\cdot l} + \bar{b}_{\cdot\cdot}$ con $k, l = 1, \ldots, n$.

Ahora se está en condiciones de definir la distancia de covarianza, de varianza y la correlación de distancia para la muestra considerada, como las raíces positivas de las siguientes ecuaciones

$$\hat{\mathcal{V}}_n^2(X,Y) = \frac{1}{n^2}\sum_{k,l=1}^{n} A_{kl}B_{kl}, \tag{2.11}$$

$$\hat{\mathcal{V}}_n^2(X) = \mathcal{V}_n^2(X,X) = \frac{1}{n^2}\sum_{k,l=1}^{n} A_{kl}^2, \tag{2.12}$$

$$\hat{\mathcal{R}}_n^2(X,Y) = \begin{cases} \frac{\hat{\mathcal{V}}^2(X,Y)}{\sqrt{\hat{\mathcal{V}}^2(X)\hat{\mathcal{V}}^2(Y)}}, & \hat{\mathcal{V}}^2(X)\hat{\mathcal{V}}^2(Y) > 0, \\ 0, & \hat{\mathcal{V}}^2(X)\hat{\mathcal{V}}^2(Y) = 0. \end{cases} \tag{2.13}$$

Una vez definida esta nueva noción, se va a utilizar la correlación de distancia para llevar a cabo la selección de variables debido a las altas correlaciones de Pearson que se mostraron anteriormente.

Se calculan las correlaciones de distancia de las diferentes variables con la respuesta, y en la Tabla 2.2 se muestran las variables con mayor correlación de distancia con MSINIESTRO. Posteriormente, de cada par de variables que presentaban una alta correlación de Pearson se selecciona aquella que tiene una correlación de distancia mayor con la variable respuesta, y la otra es eliminada. Así tras este proceso se concluye la elección de variables, eliminando 83 variables altamente correlacionadas con otras y dando lugar a un conjunto de datos con un máximo de 244 variables.

Variables	Correlación de distancia
PRECIO	0.053
ANTIGUEDAD_VEHICULO	0.045
SCORE_FINANCIERO	0.044
TARJERTAS_DMK_PT	0.031
PASIVO_DMK_CT	0.030
AUTORIZADO_BE	0.026

Tabla 2.2: Subconjunto de variables con mayor correlación de distancia con la variable MSINIESTRO.

2.3.3. Representaciones gráficas

Para finalizar el análisis descriptivo, es conveniente mostrar gráficamente como se comporta alguna de las variables predictoras frente a la respuesta. Por esto mismo en las figuras que se pueden ver a continuación se muestran los gráficos donde se han enfrentado la respuesta MSINIESTRO y una variable explicativa. Se decidió recoger solamente las variables con una mayor correlación de distancia con la respuesta, que se habían obtenido y recogido en la Tabla 2.2.

Primeramente en la Figura 2.8 se ha tomado la variable PRECIO, aplicándole una transformación logarítmica. Por una parte en el gráfico de la izquierda se puede ver la densidad de dicha variable en cada una de las categorías de la respuesta. Luego, en el gráfico de la derecha se muestra un diagrama de cajas. En éste último se aprecia que la mediana de los precios de los vehículos que presentarán un siniestro en los próximos 6 meses (azul), es un poco superior (9.49) a la mediana de los que no (9.37) en rojo.

Figura 2.8: Densidad del logaritmo del precio en cada categoría y su boxplot frente a la tenencia de siniestros en los próximos 6 meses.

A continuación en la Figura 2.9 se procede de manera análoga al caso anterior pero con la variable ANTIGUEDAD_VEHICULO, referida a la antigüedad en días del vehículo. En este caso se ha transformado en años, y en el diagrama de cajas se puede ver que la mediana de la antigüedad de los vehículos que no presentan un siniestro en los posteriores 6 meses (15) es superior a la mediana de los que si (12.9). Además, en la densidad de la variable, se pueden ver dos modas bastante claras indicando una posible distribución bimodal. Esto podría indicar la existencia de dos subgrupos en la

propia variable.

Figura 2.9: Densidad de la antigüedad de los vehículos en cada categoría y boxplot frente a la tenencia de siniestros en los próximos 6 meses.

Se continua con la variable SCORE_FINANCIERO que identifica la puntuación financiera del cliente. En este caso se elabora un gráfico de barras, donde se puede ver el porcentaje de unos que toma la variable MSINIESTRO en cada nivel de SCORE_FINANCIERO en función de las observaciones que tiene cada categoría. Sin más que observar el gráfico de barras de la Figura 2.10, se puede ver que la mayor tasa de siniestros en los próximos 6 meses (11.40 %) la tiene la categoría 6, que identifica la peor puntuación financiera que puede tener un cliente. En este caso, parece que a medida que mejora dicha puntuación, se dan menos siniestros en los siguientes 6 meses.

Figura 2.10: Porcentaje de siniestros en los próximos seis meses en función del número de casos de cada categoría de SCORE_FINANCIERO.

En la Figura 2.11 se ha tomado en este caso el logaritmo de la variable TARJERTAS_DMK_PT, que identifica el saldo medio de las tarjetas del cliente como primer titular. Se puede ver que la densidad para cada una de las categorías de la variable objetivo es muy similar. En cuanto al boxplot, se aprecian pequeñas diferencias en el diagrama de cajas, ya que la mediana de los que tendrán un siniestro (5.96) es superior a la de la otra categoría (5.76). Esto quiere decir, que los clientes que tienen un siniestro en los próximos seis meses, recogen saldos de tarjetas globalmente más altos que los otros.

En la Figura 2.12 se ha usado la variable explicativa referida al saldo en pasivo como cualquier titular que tienen los clientes (PASIVO_DMK_CT). En cuanto a las medianas en el boxplot, los clientes que no tienen siniestro, tienen un saldo en pasivo un poco más alto ya que la mediana es de 9.33, a diferencia del 9.06 que toma la otra categoría.

Figura 2.11: Densidad del logaritmo del saldo medio de las tarjetas en cada categoría y su boxplot frente a la tenencia de siniestros en los próximos 6 meses.

Figura 2.12: Densidad del logaritmo del saldo en pasivo en cada categoría y boxplot frente a la tenencia de siniestros en los próximos 6 meses.

Figura 2.13: Densidad del logaritmo de AUTORIZADO_BE y su boxplot frente a la tenencia de siniestros en los próximos 6 meses.

Finalmente, se muestran gráficos como los vistos hasta ahora pero utilizando la transformación logarítmica de la variable AUTORIZADO_BE. Ésta indica la suma de los saldos dispuestos y disponibles del cliente en el Banco de España, y se recogen las representaciones en la Figura 2.13. Se observa una pequeña diferencia en las medianas, siendo la de la categoría de no siniestro en los próximos seis meses (10), un poco inferior a la otra (10.2).

Una vez terminado el análisis de las variables, se está en condiciones de comenzar a construir el modelo en cuestión utilizando distintos tipos de métodos de clasificación. No obstante, antes de comenzar con su construcción es importante exponer la distinta teoría de cada uno de ellos, para así

comprender en que consisten y como funcionan.

Capítulo 3

Modelos de clasificación

En el transcurso de este apartado se definirán y desarrollarán diferentes modelos predictivos que se usarán posteriormente en el Capítulo 4 para predecir la variable respuesta MSINIESTRO definida en la Sección 2.2. Tal y como se ha definido, se tiene que la misma es una variable categórica binaria por lo que se está ante un problema de clasificación. Durante el desarrollo del capítulo se denotará la variable respuesta como Y, y $X = (X_1, \ldots, X_p)$ representarán las variables predictoras.

Para abordar el problema de clasificación se van a utilizar métodos de clasificación supervisada, que se entrenan y aprenden del comportamiento de las observaciones de un conjunto de datos. Luego en base a este aprendizaje predicen el valor de la variable respuesta en una nueva observación.

Para la elaboración de todo el capítulo se utilizaron como referencia distintos libros y artículos que se irán mencionando y citando según sea conveniente, pero antes de exponer los diferentes métodos es necesario comentar la partición del conjunto de datos.

Partición del conjunto de datos

Se comienza separando el conjunto de datos en subconjuntos de menor tamaño con dos propósitos distintos: entrenar el modelo y validar el modelo. Por una parte el conjunto de datos de entrenamiento se va a usar para estimar los parámetros del modelo, mientras que el conjunto de validación será usado para analizar el comportamiento y precisión del modelo en nuevas observaciones.

Es frecuente separar el conjunto de forma que el 70 % conforme el subconjunto de entrenamiento y el 30 % restante el conjunto de validación. No obstante algunos modelos requieren fijar unos hiperparámetros, por lo tanto se procederá dividiendo el conjunto de datos en tres: un conjunto de entrenamiento para entrenar el modelo, un conjunto de validación para estimar los hiperparámetros necesarios y conjunto de test para validar la precisión del modelo en una muestra totalmente distinta.

Cada cliente debe aparecer a lo sumo en uno de los conjuntos para tratar de conseguir cierta independencia entre los conjuntos, por ello dicha partición se realizará a partir de la variable CLIENTE_ID con un muestreo aleatorio. En este caso se ha decidido tomar los subconjuntos de forma que el 80 % de los clientes se encuentren en los datos de entrenamiento, un 10 % en los de test y el otro 10 % restante en el conjunto de validación. Notar que las predicciones se obtendrán a partir de la muestra de test, y así se podrá validar el modelo en términos de predicción.

3.1. Árboles de decisión

Los árboles de decisión son uno de los métodos de clasificación supervisada más sencillos e interpretables que existen a la hora de afrontar un problema de clasificación, o incluso de regresión. Dicha simpleza se debe a que segmentan el conjunto de variables predictoras en múltiples áreas de forma que se pueda entender el proceso mediante un árbol binario. Dichos árboles tienen un poder predictivo bajo, sin embargo, son importantes por ser la base de otros métodos mas complejos y con un mayor potencial como pueden ser los métodos de Bagging y Boosting, que se expondrán más adelante. Mencionar que la idea de los árboles de decisión fue desarrollada inicialmente por Breiman et al. (1984).

Este método es muy utilizado para predecir una variable respuesta, y en función de como se defina la misma existen dos tipos de árboles: los árboles de regresión (variable respuesta continua) y los árboles de clasificación (variable respuesta categórica). En ambos casos lo que se pretende es construir un modelo en forma de árbol y para ello se parte de un nodo inicial que representa el conjunto de entrenamiento. Seguidamente se hace una división del mismo en dos subconjuntos que se representan con nuevos nodos (nodos internos), y dicha segmentación se realiza utilizando una variable explicativa y un punto de corte.

Esto mismo se repite hasta que se cumple una condición de parada, de forma que el árbol para de ramificarse y se obtienen unos nodos finales denominados nodos terminales $(R_1, \ldots, R_j)$. Los mismos son los que se usan posteriormente para elaborar las predicciones para ello, generalmente se utiliza la moda en un problema de clasificación y la media en uno de regresión. Para comprender y visualizar mejor el proceso explicado, en la Figura 3.1 se ha expuesto un esquema extraído de Hastie et al. (2013).

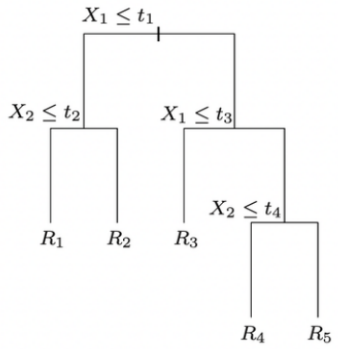

Figura 3.1: Ejemplo de un árbol de decisión binario. Fuente: Hastie et al. (2013).

A continuación se va a explicar la metodología CART (Classification and Regression Trees) descrita por Breiman et al. (1984), que es el método más conocido para los árboles de regresión y de clasificación. La misma propone una segmentación que da lugar a árboles más complejos y profundos, y se basa en un concepto conocido como particionamiento recursivo.

La misión del particionamiento recursivo consiste en obtener unos nodos terminales que sean homogéneos. No obstante, en la práctica es muy complicado obtener una homogeneidad completa. Por tanto, el criterio de parada de CART se basa en maximizar la homogeneidad de las variables resultantes en los nodos terminales.

Para intentar cuantificar la homogeneidad existe el concepto de impureza del nodo, que puede defi-

nirse de distintas formas. Por ejemplo, en problemas de regresión se suele utilizar el criterio de mínimos cuadrados, mientras que en clasificación, existen otras como son la entropía, el índice de Gini... A la hora de seleccionar el criterio de división, se elige de forma que se obtenga la mayor disminución de la impureza posible. No obstante, ha de tenerse en cuenta que debe conseguirse una homogeneidad en ambos nodos hijos. Así a la hora de dividir el nodo inicial, es muy importante tener en cuenta los distintos criterios de parada y de división.

Tras la exposición de dicho concepto se propone un ejemplo de un problema de regresión descrito en Hastie et al. (2013) donde la variable Y es una respuesta continua y $X^T = \{X_1, X_2\}$ que representan las variables explicativas toman valores dentro del intervalo unitario. Teniendo en cuenta esto se decide dividir el espacio en particiones binarias recursivas como se muestra en la Figura 3.2. De esta forma se divide el espacio es dos regiones y se modela Y en cada región eligiendo la variable y el punto de corte que genera un mejor ajuste. Este proceso se repite en cada zona hasta que se verifica un criterio de parada. Por ejemplo, en este caso se selecciona como primer punto de corte $X_1 = t_1$, segmentando el espacio en $X_1 \leq t_1$ y $X_1 > t_1$. Seguidamente la región $X_1 \leq t_1$ se divide de nuevo en $X_2 \leq t_2$ y $X_2 > t_2$, y la región de $X_1 > t_1$ se parte con el punto de corte $X_3 = t_3$. Finalmente el subconjunto de $X_3 > t_3$ se segmenta teniendo en cuenta un último corte $X_4 = t_4$. De esta forma se obtiene una partición del conjunto inicial de datos en 5 regiones R_1, R_2, R_3, R_4, R_5.

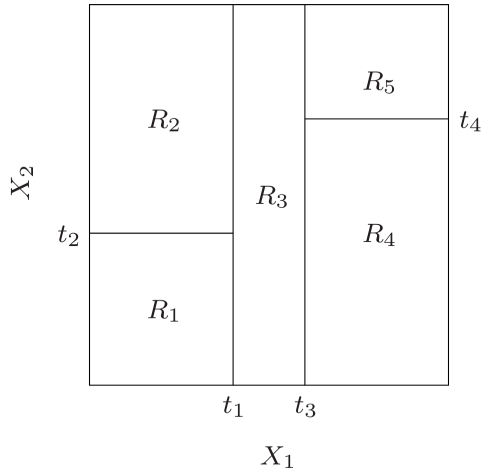

Figura 3.2: Regiones del espacio predictor. Fuente: Hastie et al. (2013).

Considerando dicha segmentación, el modelo de regresión que predice Y es el árbol que se muestra en la Figura 3.1 y las predicciones vienen dadas por la siguiente ecuación

$$\hat{f}(X) = \sum_{j=1}^{5} c_j \mathbb{I}_{\{(X_1, X_2) \in R_j\}}, \tag{3.1}$$

donde c_j es la media de la respuesta de todas las observaciones que pertenecen a R_j.

3.1.1. Árboles de regresión

Se presentan los árboles de regresión con las variables predictoras $X_1, \ldots, X_p$, y la variable respuesta Y continua con un total de N valores, uno por cada observación. Esto quiere decir que la muestra inicial de datos es de la forma $Z = (x_i, Y_i)$ con $i = 1, \ldots, N$ donde $x_i = (x_{i,1}, \ldots, x_{i,p})$. Ahora el objetivo

del problema es predecir Y a partir de los valores de las variables explicativas, o lo que es lo mismo, estimar

$$E\left[Y|X = x_i\right], \tag{3.2}$$

Es conveniente comentar que el algoritmo respecto a las decisiones sobre la profundidad del árbol, las variables y los puntos de corte, las toma de manera automática.

Ante la tenencia de una respuesta continua, se considera que tomar como medida de impureza de los nodos terminales el criterio de mínimos cuadrados es bastante adecuado. Dicho concepto se define como

$$\sum_{i \in R_j} (Y_i - \hat{f}(x_i))^2, \tag{3.3}$$

donde $\hat{f}(x_i)$ representa el promedio de los valores de $Y_i \in R_j$, es decir, $\hat{f}(x_i) = \frac{1}{N_j} \sum_{x_i \in R_j} Y_i$ siendo $N_j = \sum_{i=1}^{N} \mathbb{I}_{\{x_i \in R_j\}}$. Notar el abuso de notación que se comete en $i \in R_j$, pues se refiere a las observaciones $i \in N$ que verifican $x_i \in R_j$.

No obstante, a la hora de la práctica si se quisiera obtener la mejor partición binaria en base a la suma de cuadrados, esto puede ser una tarea computacionalmente muy costosa. Por esta razón, se presenta a continuación un algoritmo de partición binaria recursiva. Sea k una variable y s un punto de corte, se define el par de regiones de esta forma

$$R_L(k, s) = \{x|X_k \leq s\} \text{ y } R_R(k, s) = \{X|X_k > s\}. \tag{3.4}$$

Ahora, si se quisiera dividir el un conjunto en dos regiones, R_L y R_R, se debe buscar la variable k y el punto de división s tal que se minimice

$$\min_{\hat{f}_L} \sum_{x_i \in R_L(k,s)} (Y_i - \hat{f}_L(x_i))^2 + \min_{\hat{f}_R} \sum_{x_i \in R_R(k,s)} (Y_i - \hat{f}_R(x_i))^2, \tag{3.5}$$

siendo $\hat{f}_L(x_i)$, $\hat{f}_R(x_i)$ los promedios de los valores de Y_i en cada región.

Una vez obtenida la variable de división y el punto de corte, se segmenta el conjunto de datos y se repite este proceso para cada nueva subregión. En cuanto al tamaño del árbol es fácil ver que si éste es demasiado grande se podría producir un problema de sobreajuste. Por el contrario si el mismo es demasiado pequeño es posible que no se estuviese capturando la estructura de los datos. Por este motivo, se considera el tamaño del árbol como un parámetro a ajustar que además puede ser controlado usando un criterio de parada.

Existen varios criterios de parada y como ejemplo, se construye un árbol muy grande y se detiene su crecimiento cuando éste tiene un tamaño mínimo de nodos. Luego se utiliza la poda de complejidad de costes, como se explica a continuación. Sea $A \subset A_0$ un subárbol, y J el número total de nodos terminales de A. Entonces se define como medida de impureza en el nodo, el error promedio cuadrático entre Y y el promedio de la región R_j,

$$Q_j(A) = \frac{1}{N_j} \sum_{x_i \in R_j} (Y_i - \hat{f}(x_i))^2, \tag{3.6}$$

y como criterio de complejidad de costes se tiene

$$C_\alpha(A) = \sum_{j=1}^{J} N_j Q_j(A) + \alpha J. \tag{3.7}$$

De esta forma la idea es encontrar el subárbol de A que minimiza $C_\alpha(A)$, teniendo en cuenta que α es un hiperparámetro que equilibra el tamaño del árbol y la bondad de ajuste de los datos. Esto supone que valores pequeños proponen árboles más grandes, valores grandes dan lugar a árboles pequeños, y cuando $\alpha = 0$ se tiene el árbol original A_0.

3.1.2. Árboles de clasificación

En este apartado se plantea predecir la variable Y que en este caso presenta clases que vienen dadas por $1, 2, \ldots, K$. Para ello se decide utilizar el mismo procedimiento de partición recursiva binaria que se usa con los árboles de regresión, no obstante, es necesario modificar los criterios de división y de parada.

Primeramente no se hablará de la media aritmética $\hat{f}(x_i)$ para representar a los valores pertenecientes a la región R_j. En este caso, se hará uso de la categoría que contenga más observaciones en dicha región. Así la predicción que se asignará a todo nuevo registro que entre en R_j será esa clase. Entonces si se considera un nodo j, representando una región R_j que tiene N_j observaciones, se tiene que la proporción de observaciones de la categoría k en el nodo j viene dada por

$$\hat{p}_{jk} = \frac{1}{N_j} \sum_{x_i \in R_j} \mathbb{I}_{\{Y_i = k\}}. \tag{3.8}$$

De esta forma se clasifican las observaciones x_i en el nodo j como $\hat{H}(x_i) = \arg\max_k \hat{p}_{jk}$, siendo $\hat{H}$ el clasificador que representa la categoría mayoritaria en ese nodo.

A continuación, se habla de distintas medidas de impureza que son utilizadas para los árboles de clasificación:

- Error de clasificación: $1 - \max_k(\hat{p}_{jk})$.
- Índice de Gini: $\sum_{k \neq k'} \hat{p}_{jk}\hat{p}_{jk'} = \sum_{k=1}^{K} \hat{p}_{jk}(1 - \hat{p}_{jk})$.
- Entropía: $-\sum_{k=1}^{K} \hat{p}_{jk} \log(\hat{p}_{jk})$

Por una parte se tiene el *error de clasificación*, éste se refiere al porcentaje de observaciones que no pertenece a la categoría mayoritaria del nodo. Luego el *índice de Gini*, es una métrica que mide con que frecuencia una observación seleccionada al azar se clasifica incorrectamente. Por tanto se busca alcanzar un valor pequeño, pues supondría una mayor pureza en el nodo. Finalmente, la *entropía* contabiliza la dispersión de cada clase. Esto quiere decir, que si en una división solamente se tiene una categoría, la misma será pura y por tanto la entropía será 0. Sin embargo, si la dispersión de las categorías es igual, se tendrá que la entropía toma un 1. Así lo que interesa es obtener valores pequeños, próximos al 0.

Notar que si se está ante un problema de dos clases, es decir, un problema con la variable respuesta Y dicotómica, si se considera p la proporción en la segunda clase $Y = 1$, se tiene que las medidas explicadas arriba se pueden escribir como $1 - \max(p, 1-p)$, $2p(1-p)$ y $-p\log(p) - (1-p)\log(1-p)$, respectivamente. Todas son similares, sin embargo, las dos últimas son diferenciables por lo que son más susceptibles ante la optimización numérica. Además al igual que en los árboles de regresión, es necesario ponderar las medidas de impureza de cada nodo por el número de observaciones presentes en cada uno.

Comentar que Hastie et al. (2013) expone que es mejor utilizar la entropía o el índice de Gini como medida de impureza en los árboles de clasificación. Ellos explican que el error de clasificación, no es suficientemente sensible para crear buenas estructuras en los árboles.

En cuanto al criterio de parada, éste es el mismo que en los árboles de regresión. No obstante, la única diferencia se encuentra en que la medida de impureza que debe usarse es alguna de las descritas antes, es decir, la *entropía*, el *error de clasificación* o el *índice de Gini*. Por tanto se podría obtener un criterio de complejidad sin más que sustituir $Q_j(A)$ en la Ecuación 3.7 por alguna de las medidas de impureza mencionadas.

3.1.3. Parámetros y sobreajuste del modelo

A la hora de construir árboles de decisión uno de los problemas principales es el sobreajuste de los mismos. Esto se produce cuando los parámetros se ajustan muy bien al conjunto de datos de entrenamiento, pero a la hora de predecir usando otro conjunto de datos, la precisión del modelo es limitada, llevando así a malas predicciones. Para evitar dicho problema existen dos métodos:

1. Parada temprana del árbol: Con esta técnica se intenta reducir el sobreajuste limitando el crecimiento del árbol y el número de nodos creados. Entre los hiperparámetros utilizados se encuentran algunos que indican el número mínimo de observaciones que debe haber en un nodo para poder dividirse, la profundidad de los árboles, el número máximo de nodos terminales, la reducción mínima del error para que se pueda dividir de nuevo, entre otros.
2. Podado del árbol: En este caso se crea un árbol sin casi restricciones de parada y seguidamente se selecciona el subárbol que minimiza $C_\alpha(A)$ (Ecuación (3.7)).

Finalmente, destacar que para la elaboración de toda la sección se ha hecho uso del Capítulo 9 de Hastie et al. (2013).

3.2. Modelos bagging y boosting

Uno de los problemas que suelen presentar los árboles de decisión, es el equilibrio entre el sesgo y la varianza. Como es conocido, el sesgo establece la diferencia en promedio entre los valores reales y las predicciones. Por el contrario, la varianza indica la variación de los distintos estimadores en función de la muestra de entrenamiento que se utilice.

Dicho problema en los árboles de decisión se traduce en que al crear árboles complejos y más grandes, el sesgo se reduce, ya que hay un mayor parecido con el conjunto de entrenamiento. Sin embargo, al mismo tiempo se produce un aumento de la varianza, provocando un árbol que no predice correctamente observaciones nuevas (sobreajuste). Entonces, se puede plantear la idea de construir árboles más simples y más pequeños. Si se procede así, aunque los mismos reduzcan esa varianza, no van a captar bien la combinación entre las variables, lo que provocará un aumento del sesgo. Para ejemplificar y visualizar mejor el problema, en la Figura 3.3 se puede ver un gráfico.

Ante este problema surgen los métodos de emsamblado o ensemble, y entre los que se encuentran dos técnicas como son el *bagging* y el *boosting*. La idea principal es utilizar la combinación de métodos de predicción sencillos, es decir, que tengan poca capacidad predictiva, y a partir de su combinación alcanzar un método de predicción potente y robusto. Un ejemplo de método simple son los árboles de decisión ya que son sencillos pero también rápidos a la hora de construirse. De esta forma, generan múltiples árboles y posteriormente éstos se combinan obteniendo así las predicciones.

Al utilizar el método *bagging*, se usarán como modelos simples aquellos que tengan poco sesgo pero mucha varianza, de forma que al combinarlo se consiga reducir la varianza sin aumentar el sesgo. Por el contrario, con el *boosting*, se parte de modelos con poca varianza pero mucho sesgo, por lo que al combinar se busca reducir dicho sesgo sin aumentar la varianza.

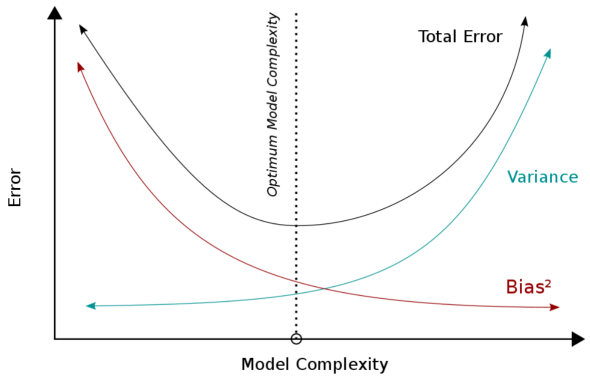

Figura 3.3: Equilibrio entre sesgo y varianza. Fuente: Fernández et al. (2021).

3.2.1. Bagging

Se comienza exponiendo el concepto de *bagging* que fue introducido por Breiman (1996), un método basado en el uso del bootstrap junto con un modelo de regresión o clasificación como podría ser un árbol de decisión.

Primeramente se considera un problema de regresión, y se considera el conjunto de entrenamiento $Z = \{(x_1, Y_1), \ldots, (x_N, Y_N)\}$. Ahora entra en juego el bootstrap y a partir de dicho conjunto se generan B muestras bootstrap definidas como Z^{*b} con $b = 1, \ldots, B$. A cada una de ellas se le ajusta un árbol de regresión y seguidamente se obtiene la función $\hat{f}^{*b}(x)$ que proporciona las predicciones. De esta forma se tiene que la estimación bagging viene dada por el promedio de las predicciones

$$\hat{f}(x) = \frac{1}{B} \sum_{b=1}^{B} \hat{f}^{*b}(x). \tag{3.9}$$

Con esta técnica se consigue reducir la varianza y por tanto aumentar la capacidad predictiva. Para ello, simplemente ha de dividirse el conjunto de entrenamiento en B muestras bootstrap y con cada una construir un árbol y obtener sus predicciones. Luego éstas se usarán para promediar la predicción en todos los árboles.

A continuación se considera un problema de clasificación por lo que se define un clasificador $\hat{H}(x)$ para una respuesta que tiene K clases, y $\hat{p}_k^*$ representa la proporción de árboles que predicen con clase k. Entonces la estimación bagging en este caso será un vector de la forma $[\hat{p}_1^*(x), \ldots, \hat{p}_K^*(x)]$ y el clasificador bagging selecciona como predicción final

$$\hat{H}(x) = \arg\max_k \hat{p}_k^*(x). \tag{3.10}$$

A continuación en la Figura 3.4 se muestra un esquema del procedimiento que sigue esta técnica para así comprender mejor su funcionamiento.

Una de las ventajas de este método es la posibilidad y facilidad de estimar el error de predicción sin recurrir a un conjunto de validación, o a la validación cruzada. Es conocido que cada árbol utiliza aproximadamente de media, dos tercios de las observaciones del conjunto de entrenamiento, el tercio de observaciones restantes se conoce como observaciones *out-of-bag (OOB)*, resultado determinado por

$$\lim_{N \to \infty} \left(1 - \frac{1}{N}\right)^N = e^{-1}. \tag{3.11}$$

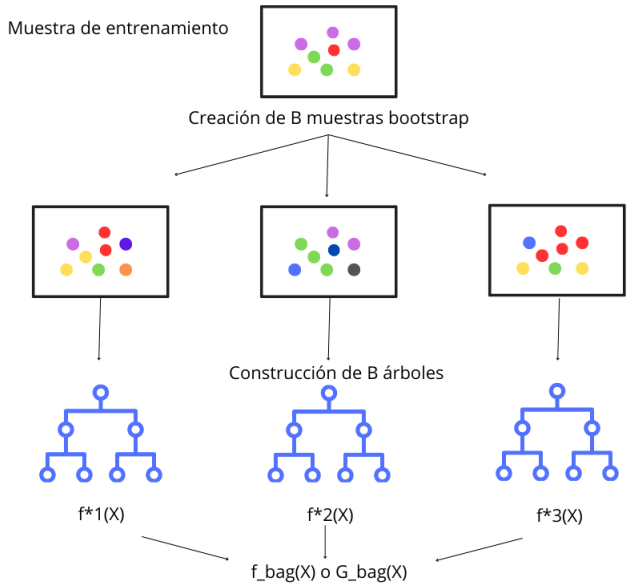

Figura 3.4: Procedimiento del algoritmo bagging. Fuente: Elaboración propia.

Teniendo en cuenta esto, si para cada árbol que se ajusta durante el proceso se guardan las observaciones no empleadas en ellos, se puede predecir la respuesta de la observación i. Para predecir cada observación, se utilizan los árboles que no han usado dicho dato, es decir, usando los árboles en los que la observación fue OOB. Entonces, si se elabora este procedimiento para las N observaciones se puede calcular lo que se conoce como *OOB-mean square error* o *OOB-classification error*, en otras palabras, que se pueden utilizar las observaciones excluidas de un árbol para validar el mismo.

A la hora de utilizar este método es importante el valor B (número de muestras bootstrap) que hay que tomar, ya que al fin y al cabo, es el número de árboles que se construirán. Esto se basa en una aproximación de Monte Carlo, por lo que se suele utilizar la convergencia del error OOB al aumentar el número de árboles. Si se produce la convergencia con unos pocos árboles, por mucho que el número aumente las predicciones no van a mejorar. Además a medida que el número es mayor, el coste computacional aumenta, no obstante, la construcción y evaluación de los modelos se pueden paralelizar. Por el contrario, cuando el número es pequeño es normal obtener pocas predicciones OOB o incluso ninguna para alguno de los casos.

Finalmente es conveniente mencionar que a pesar de mejorar el poder predictivo, con estos métodos se pierde cierta interpretabilidad que existía en los árboles. Esto se debe a que es más complicado obtener la importancia de las variables predictoras, aunque existen métodos para su cálculo. Por ejemplo, si se fija una de ellas y una medida de error, se puede calcular para cada árbol la reducción del error que se obtiene cada vez que hay una división que utilice esa variable. Luego se promedia sobre todos los árboles y se tiene una medida global de importancia, de forma que si se tiene un valor elevado el predictor será importante.

Para la realización de todo el apartado acerca del *bagging* se ha utilizado Hastie et al. (2013), pero también Fernández et al. (2021) y James et al. (2017).

3.2.2. Random forest

Cuando se procede a utilizar el método *bagging*, interesa que los árboles sean distintos. Sin embargo, a veces no son lo suficientemente diferentes ya que tienen estructuras parecidas, sobretodo al inicio de los mismos. Esto se conoce como correlación entre árboles, y principalmente aparece cuando una variable explicativa es muy fuerte en términos de importancia de variables para predecir la respuesta. Esto provoca que en la mayoría de árboles, dicha variable sea la usada para realizar la primera división. También se da cuando el árbol es un modelo bueno a la hora de explicar la relación entre las variables explicativas y la respuesta. De esta forma se vería perjudicada la reducción de la varianza que se buscaba con el *bagging*.

Ante dicho problema, Breiman (2001) propone una variante del *bagging*, conocida como *random forest* (o *bosques aleatorios*) que introduce cierta aleatoriedad en el proceso de construcción para reducir la correlación entre ellos.

Los *bosques aleatorios* proceden de igual manera que el *bagging*, la diferencia se encuentra a la hora de construir los diferentes árboles de decisión. Cada vez que se quiere realizar un corte en un árbol, se escoge una muestra aleatoria de m predictores como candidatos. En este caso, si se tomase $m = p$ se tendría el mismo resultado que utilizando el *bagging*. Generalmente se utiliza $m = \sqrt{p}$ en problemas de clasificación y $m = p/3$ en regresión, siendo p el número de predictores totales. Es importante comentar que lo más adecuado sería tratar este valor como un hiperparámetro, es decir, estudiar el error OOB para diferentes valores de m y seleccionar el mejor.

Además de m también se puede considerar el número de árboles como un hiperparámetro, pero lo más común es tratar dicho número como un problema de convergencia. A continuación, se va a mostrar el algoritmo que siguen los modelos de *random forest* ante problemas de clasificación y regresión, y en la Figura 3.5 se muestra lo descrito.

1. Para cada b con $b = 1, \ldots, B$:

   a) Se construye una muestra bootstrap Z_b^* de tamaño N a partir del conjunto de entrenamiento.

   b) Se construye un *random forest* para la muestra Z^*, repitiendo de forma recursiva los pasos siguientes para cada nodo del árbol hasta alcanzar el criterio de parada.

   - Se seleccionan m predictores aleatoriamente de los p totales.
   - Se elige la mejor variable y el mejor punto de corte entre los m posibles.
   - Se divide el nodo en dos.

2. Se obtiene el conjunto de árboles y se calcula la predicción individual de cada uno, obteniendo los estimadores $\hat{f}_1^*(x), \ldots, \hat{f}_B^*(x)$.

3. Se calcula el estimador de predicción para el *random forest*:

   - Regresión: $\hat{f}(x) = \frac{1}{B} \sum_{b=1}^{B} \hat{f}_b(x)$.
   - Clasificación: Sea $\hat{p}_k^*$ la proporción de árboles que predicen con clase k, el estimador viene dado por $\hat{H}(x) = \arg\max_k \hat{p}_k^*(x)$.

Ahora para visualizar este algoritmo, se recoge en la Figura 3.5 un esquema como se hizo con el método *bagging* para ejemplificar el procedimiento explicado.

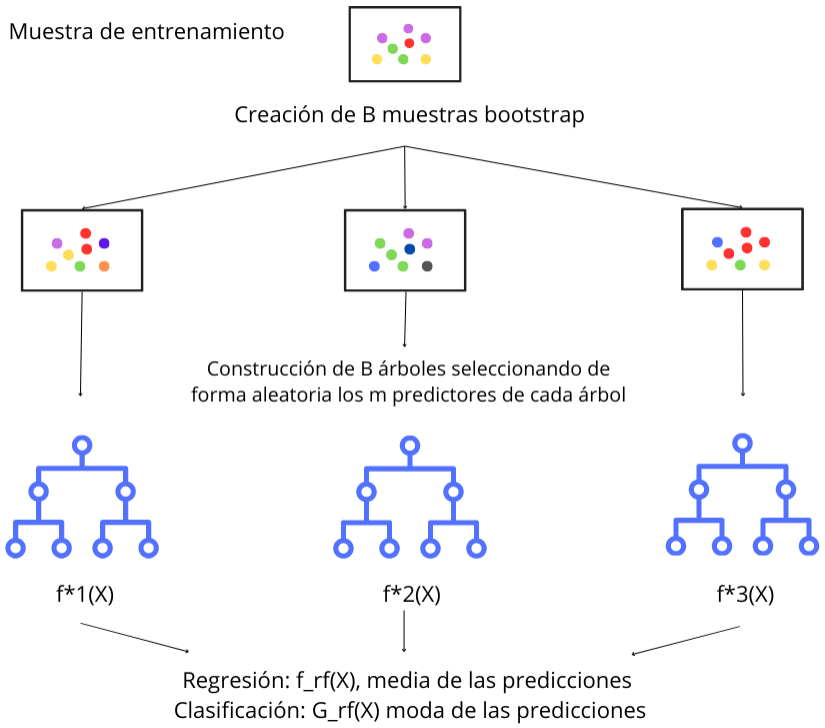

Figura 3.5: Procedimiento del algoritmo de random forest. Fuente: Elaboración propia.

Hiperparámetros de random forest

Para llevar a cabo el algoritmo explicado anteriormente, es necesario ajustar correctamente los hiperparámetros del modelo. Un mal ajuste de los mismos, puede llevar a predicciones poco precisas. Entre los mismos se encuentran los siguientes, que serán ajustados en el Capítulo 4 al construir un *random forest* con la librería `h2o` de `R`.

- **Número de árboles (ntree)**: Se corresponde con B y su valor es muy importante para la precisión del modelo, por lo que se busca el valor que minimiza el error. Si se utilizara un número muy elevado de árboles, se tendría un costo computacional importante. Por defecto en este caso se suele utilizar $B = ntree = 500$.

- **Número de variables para cada división (mtries)**: Se refiere al parámetro m, número de predictores seleccionados al azar en cada corte. Si el valor es pequeño se reduce la correlación entre los árboles, pues hay menos posibilidades de variables en cada división. Pero también es conveniente tener en cuenta que reducir mucho este valor puede provocar una reducción en la precisión del modelo. Como se mencionó anteriormente, los valores recomendados son $mtries = m = mtry = \sqrt{p}$ en clasificación y $mtries = m = mtry = p/3$ en regresión, pero en la práctica y en el presente proyecto lo más habitual es analizar distintos valores.

- **Profundidad máxima (max_depth)**: Se refiere a la profundidad máxima del árbol.

- **Muestras mínimas (min_rows)**: Se refiere al número mínimo de muestras que debe haber en los nodos terminales. En general se suele utilizar $min\_rows = 1$ en clasificación y $min\_rows = 5$ en regresión.

- **Tamaño de la muestra bootstrap (sample_rate)**: Se refiere a la proporción de muestra utilizada para entrenar cada árbol. Cuando toma un valor pequeño, se tienen árboles menos correlados entre ellos, pero supone una reducción de cada uno de los árboles. Toma valores en el intervalo $(0, 1]$ y por defecto se fija en 0.6320000291.

El apartado se ha elaborado a partir de Hastie et al. (2013), y si se quisiera conocer más acerca de dicho algoritmo y sus características se puede recurrir a su capítulo 15.

3.3. Boosting

El *boosting* es la segunda técnica de ensamblado mencionada en la Sección 3.2, que se parece al *bagging*, pero tiene sus diferencias. Con este método también se combinan múltiples modelos con poca capacidad predictiva (débiles) para luego dar lugar a un modelo mucho más potente. Al igual que sucedía con el *bagging*, los árboles de decisión simples son un buen candidato ya que destacan por ser fácilmente interpretables, por tener poca capacidad predictiva y por construirse de forma rápida.

A pesar de que la idea se introdujo años antes, no fue hasta 1996 cuando se desarrolló gracias a Freund and Schapire (1996) una implementación para los problemas de clasificación y que posteriormente se consiguió extender a problemas de regresión. La diferencia principal con la otra técnica de emsamblado, se encuentra en que en lugar de utilizar B muestras bootstrap se trabaja constantemente con el mismo conjunto de datos, modificando los pesos de las diferentes observaciones para generar modelos distintos. Así se tendrán B árboles que presentarán mucho sesgo y poca varianza. La idea es construirlos iteración tras iteración de forma que al añadir el siguiente se disminuya el error.

A continuación se van exponer cuatro algoritmos basados en el *boosting* que son el *AdaBoost*, el *Gradient Boosting*, el *XGboost* y el *LightGBM*.

3.3.1. AdaBoost

Primeramente se va a considerar un problema de clasificación donde la variable respuesta Y tiene dos categorías, es decir, $Y \in \{0, 1\}$. Además, se cuenta con un total de p variables explicativas y N observaciones, dando lugar a un conjunto de datos de la forma $x_i = (x_{i,1}, \ldots, x_{i,p})$. También se toma un clasificador $\hat{H}(x)$ que predice tomando uno de los valores de la respuesta. Entonces, si se considera la muestra de entrenamiento, se tendrá que la tasa de error viene dada por

$$err = \frac{1}{N} \sum_{i=1}^{N} \mathbb{I}_{\{Y_i \neq \hat{H}(x_i)\}}, \tag{3.12}$$

así la tasa media de error en predicciones nuevas se define como $E_{xY} \mathbb{I}_{\{Y \neq \hat{H}(x)\}}$.

Conocida la tasa de error, se puede definir un clasificador débil como aquel cuya *err* solamente es un poco mejor que la aleatoriedad del modelo. La intención de esta técnica es aplicar repetidamente el algoritmo de clasificación débil a modificaciones de los datos, dando lugar así a una secuencia de clasificadores $\hat{H}_b(x)$ con $b = 1, \ldots, B$. En cada paso del algoritmo se elabora una modificación utilizando los pesos w_i, que fueron asignados en el inicio del mismo a cada observación (x_i, Y_i) con $i = 1, \ldots, N$. Una vez obtenidas las predicciones de todos los $\hat{H}_b$, se combinan de forma ponderada de forma que el peso de cada clasificador depende de su tasa de error. Así se tiene un efecto de mayor influencia en los que son más precisos.

A continuación se presenta el algoritmo *AdaBoost* para un problema de clasificación, considerando como predictores débiles árboles de decisión.

1. Se selecciona el número B de iteraciones.
2. A todas las observaciones de la muestra de entrenamiento se le asigna el mismo peso, $w_i = 1/N$ con $i = 1, \ldots, N$ siendo N el tamaño de la muestra.
3. Para $b = 1, \ldots, B$
   a) Se ajusta un clasificador débil, que en este caso es un árbol de decisión, $\hat{H}_b(x)$, a los datos de entrenamiento utilizando las observaciones ponderadas.
   b) Se calcula el error cometido en la clasificación a partir de
   $$err_b = \frac{\sum_{i=1}^{N} w_i \mathbb{I}_{\{Y_i \neq \hat{H}_b(x_i)\}}}{\sum_{i=1}^{N} w_i}. \tag{3.13}$$
   c) Se calcula $\alpha_b = \log\left((1 - err_b)/err_b\right)$ para medir la influencia de $\hat{H}_b$ de forma proporcional al número de aciertos, es decir, dar mayor influencia a los más precisos.
   d) Se actualizan los pesos w_i recodificando su valor como $w_i \exp(\alpha_b \mathbb{I}_{\{Y_i \neq \hat{H}_b(x_i)\}})$ con $i = 1, \ldots, N$ y se vuelve al paso *a*).
4. Una vez obtenido el modelo final se elabora la predicción ponderada sobre cada uno de los pesos de los árboles. De esta forma, tendrán más peso las predicciones más efectivas, es decir, las que hayan tenido un menor error. Dicha predicción se obtiene a partir de
   $$\hat{H}(x) = \sum_{b=1}^{B} \alpha_b \hat{H}_b(x). \tag{3.14}$$

Tal y como funciona el algoritmo, cuando en una iteración unas observaciones se clasifican erróneamente, en la siguiente iteración las mismas tendrán un peso mayor. Por el contrario, aquellos datos bien clasificados mantendrán sus pesos. En la Figura 3.6 se muestra un esquema del funcionamiento del algoritmo para así comprender mejor su comportamiento.

En cuanto al algoritmo para los problemas de regresión, éste es el mismo que para clasificación, sin más que intercambiar el clasificador $\hat{H}(x)$ por

$$\hat{f}(x) = \sum_{b=1}^{B} \alpha_b \hat{f}_b(x). \tag{3.15}$$

Si se quisiera conocer más información acerca del algoritmo expuesto se puede recurrir al Capítulo 10 de Hastie et al. (2013), usado para elaborar este apartado, junto con Fernández et al. (2021).

3.3.2. Gradient Boosting

Después de varios años del desarrollo del algoritmo Adaboost del apartado anterior, Friedman (2001) desarrolla una nueva técnica conocida como *gradient boosting machine* (GBM). Dicho algoritmo es un método iterativo de descenso de gradiente y lo que busca es encontrar un modelo aditivo que minimice la función de pérdida utilizando predictores débiles como pueden ser los árboles de decisión.

Sea $f(x)$ una función de predicción sobre Y a partir de los valores que toma X, entonces se define una función de pérdida o coste asociada a $f(x)$ como la función que penaliza los errores de predicción,

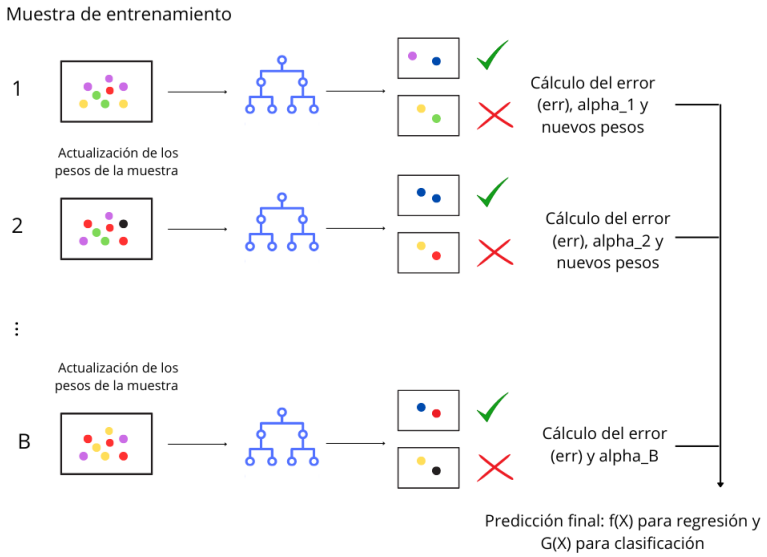

Figura 3.6: Procedimiento del algoritmo Adaboost. Fuente: Elaboración propia.

$L(Y, f(x))$. Se conocen múltiples funciones de pérdida, pero generalmente las más usadas son la función de mínimos cuadrados definida como $(Y - f(x))^2$, la función de error absoluto $|Y - f(x)|$ y la función exponencial.

La ventaja de esta nueva técnica es que permite el uso de cualquier función de coste siempre que la misma sea diferenciable. En este caso, en lugar de ajustar los errores del árbol anterior como procedía el algoritmo Adaboost, éste optimiza secuencialmente la función de pérdida hasta que se alcance el nivel requerido o hasta que no se mejore al utilizarlo en un conjunto de validación. Este comportamiento provoca que el algoritmo minimice la función de pérdida de error cuadrático medio utilizando como gradiente al error residual.

Volviendo a la Sección 3.1 se tiene que los árboles de decisión dividían el espacio de todos los valores de las variables predictoras en J regiones denotadas por R_j con $j = 1, \ldots, J$, y que representaban los nodos terminales. Ahora si a estas regiones se les asigna un valor estimado c_j, entonces la predicción de $x \in R_j$ sería $\hat{f}(x) = c_j$. Teniendo en cuenta esto y que $\theta = \{R_j, c_j\}_{j=1}^{J}$ entonces se podría expresar un árbol como sigue

$$T(x; \theta) = \sum_{j=1}^{J} c_j \mathbb{I}_{\{x \in R_j\}}. \tag{3.16}$$

Por tanto, para encontrar los c_j solamente hay que minimizar

$$\hat{\theta} = arg\,\min_{\theta} \sum_{j=1}^{J} \sum_{x_i \in R_j} L(Y_i, c_j), \tag{3.17}$$

y así simplemente se está ante un problema de optimización combinatoria para el cual suele bastar encontrar una solución subóptima aproximada.

A continuación y haciendo caso a lo descrito, se tiene que el boosting de árboles de decisión se puede expresar como

$$\hat{f}_B(x) \sum_{b=1}^{B} T(x; \theta_b), \tag{3.18}$$

por lo que en cada iteración dado el modelo actual $\hat{f}_{b-1}(x)$ para la constante $\theta_b = \{R_{jb}, c_{jb}\}_1^{J_b}$ del siguiente árbol, se busca calcular

$$\theta_b = arg\, \min_{\theta_b} \sum_{i=1}^{N} L(Y_i, \hat{f}_{b-1}(x_i) + T(x_i; \theta_b)). \tag{3.19}$$

Resolver esta ecuación puede ser un proceso complicado en múltiples ocasiones, pues minimizar $L(\hat{f}) = \sum_{i=1}^{N} L(Y_i, \hat{f}(x_i))$ suele traer dificultad. Por esta razón, el algoritmo de gradient boosting decide tomar el gradiente de la función de pérdida por el método de descenso de gradiente. Notar que como solamente se puede utilizar el conjunto de entrenamiento para calcular esto, se construye el algoritmo para que los árboles se aproximen al gradiente negativo. A continuación se presenta el algoritmo del GBM considerando un problema de regresión:

1. Se inicializa $\hat{f}_0(x) = arg\, \min_c \sum_{i=1}^{N} L(Y_i, c)$.

2. Para cada $b = 1, \ldots, B$:

   a) Para $i = 1, \ldots, N$ se obtienen

   $$r_{ib} = - \left[\frac{\partial L(Y_i, \hat{f}(x_i))}{\partial \hat{f}(x_i)} \right]_{\hat{f}=\hat{f}_{b-1}} \tag{3.20}$$

   b) Se ajusta un árbol de regresión a los resultados de r_{ib} dando lugar a las regiones terminales definidas como R_{jb} con $j = 1, \ldots, J_b$.

   c) Para cada $j = 1, \ldots, J_b$ se calcula

   $$c_{jb} = arg\, \min_c \sum_{x_i \in R_{jb}} L(Y_i, \hat{f}_{b-1}(x_i) + c). \tag{3.21}$$

   d) Se actualiza $\hat{f}_b(x) = \hat{f}_{b-1}(x) + \sum_{j=1}^{J_b} c_{jb} \mathbb{I}_{\{x \in R_{jb}\}}$.

3. Se consigue $\hat{f}(x) = \hat{f}_B(x)$.

Por el contrario si se estuviese ante un problema de clasificación, se procedería de igual manera salvo que los pasos 2 y 3 es necesario repetirlos para el total de clases de la variable respuesta en cada una de las iteraciones. Así en el paso 2 sería necesario calcular

$$r_{ikb} = Y_i - \hat{p}_k(x_i), \tag{3.22}$$

siendo la función de clasificación $\hat{p}_k(x) = \frac{e^{f_k(x)}}{\sum_{l=1}^{K} e^{f_l}}$, y en el paso 3 se obtendría $\hat{f}_{kB}(x)$ para cada $k = 1, \ldots, K$.

Hiperparámetros de GBM

A continuación se van a comentar los diferentes hiperparámetros que se ajustarán posteriormente en el Capítulo 4 cuando se considere construir un modelo GBM utilizando la librería `h2o` de `R`.

- **Número de árboles (ntree)**: En este caso, a diferencia de lo que sucedía en el random forest, interesa que sea un número elevado y además suele depender de otros hiperparámetros. No obstante, según el valor que se tome puede llevar a un sobreajuste del modelo.
- **Tasa de aprendizaje (learn_rate)**: Dicho parámetro establece cuanto ha de influir cada árbol del algoritmo en la predicción final y además determina la velocidad del algoritmo. Su valor se encuentra entre el 0 y el 1, de forma que cuanto más pequeño sea dicho valor más lento avanzará, utilizando más árboles y pudiendo llevar a un sobreajuste.
- **Profundidad máxima (max_depth)**: Establece la profundidad máxima de los árboles. Los árboles que tienen una profundidad pequeña son más eficientes computacionalmente hablando, sin embargo suponen un menor número de árboles. Y por el contrario, aquellos con una profundidad grande capturan ciertas relaciones pero pueden llevar a un sobreajuste.
- **Muestras mínimas (min_rows)**: Establecen el número mínimo de muestras que ha de tener cada nodo terminal. Así cuando el valor es elevado ayudan con el problema del sobreajuste, y valores pequeños cuando se está ante una clase desequilibrada pueden aportar ciertas mejoras.
- **sample_rate**: Determina el porcentaje de observaciones escogidas al azar que se emplean en el ajuste de cada uno de los árboles.
- **col_sample_rate**: Establece el porcentaje de predictores que se seleccionan aleatoriamente en el ajuste de cada árbol.
- **col_sample_per_tree**: Fija el porcentaje de predictores que se seleccionan de forma aleatoria en cada división de nodo de un árbol.

El desarrollo de este apartado acerca del algoritmo GBM se ha llevado a cabo con la ayuda del Capítulo 10 de Hastie et al. (2013), por lo que si se quisiera conocer alguna información a mayores bastaría con recurrir a dicha referencia.

3.3.3. Extreme Gradient Boosting

El algoritmo *extreme gradient boosting*, también conocido como *XGBoost*, es una técnica muy eficiente y cada vez más popular entre los algoritmos basados en el boosting y fue propuesta por Chen and Guestrin (2016). Este método procede según el boosting, pues parte de modelo predictivos débiles, como son los árboles de decisión, para construir un modelo predictivo más potente. Además dichos árboles se construyen por pasos, de forma que el árbol siguiente siempre busca reducir el error cometido en el modelo anterior.

La diferencia de los distintos algoritmos de boosting radica en como identifican los mismos los errores de los árboles. En el caso de este método, se procede de manera similar al GBM explicado en la Sección 3.3.2 la única diferencia se encuentra en que ahora se añaden divisiones a mayores dentro de los árboles, buscando así reducir la función objetivo (función de pérdida). Además en cada nodo se establece una puntuación, que posteriormente se sumarán para obtener la predicción final buscando así una estructura eficiente. Es el momento de explicar el funcionamiento de dicha técnica, por lo que a continuación se exponen sus fundamentos y para ello se ha utilizado Chen and Guestrin (2016).

Al igual que sucedía con otros modelos boosting, el XGBoost se entrena de forma aditiva, es decir, se fija lo aprendido en cada paso y en la siguiente iteración se incluye un nuevo árbol. Teniendo en cuenta que es un proceso aditivo, se pueden expresar las predicciones como sigue

$$\begin{aligned}
\hat{f}_0(x_i) &= 0, \\
\hat{f}_1(x_i) &= \hat{f}_0(x_i) + f_1(x_i) = f_1(x_i), \\
&\vdots \\
\hat{f}_B(x_i) &= \sum_{b=1}^{B} f_b(x_i) = \hat{f}_{B-1}(x_i) + f_B(x_i).
\end{aligned} \tag{3.23}$$

Así en cada paso se añade el árbol que mejor optimiza la función de pérdida, y el algoritmo encuentra el estimador del árbol óptimo f_b que reduce el error cometido en el paso anterior y mejorar la división de los árboles. Mencionar que éste, también incluye una función de regularización más compleja para intentar evitar el sobreajuste del modelo. Por tanto, sea $\hat{f}_B(x_i)^{(t)}$ la predicción de la observación i en la iteración t, y $\omega(f_b)$ la complejidad del árbol, se puede definir

$$L^{(t)} = \sum_{i=1}^{N} L(Y_i, \hat{f}_B(x_i)^{(t-1)} + f_t(x_i)) + \omega(f_t). \tag{3.24}$$

Ahora si se supone que dicha función es convexa y además dos veces diferenciable, se puede aproximar la misma mediante una aproximación de orden dos tal y como se ve a continuación

$$L^{(t)} \approx \sum_{i=1}^{N} \left[L(Y_i, \hat{f}_B(x_i)^{(t-1)} + g_i f_t(x_i) + \frac{1}{2} h_i f_t(x_i)^2 \right] + \omega(f_t), \tag{3.25}$$

donde $g_i = \frac{\partial}{\partial z} L(Y_i, z)\big|_{z=\hat{f}_B(x_i)^{(t-1)}}$ y $h_i = \frac{\partial^2}{\partial^2 z} L(Y_i, z)\Big|_{z=\hat{f}_B(x_i)^{(t-1)}}$ son las derivadas de primer y segundo orden de la función de pérdida.

Puesto que $\omega(f) = \gamma T + \frac{1}{2}\lambda\|\nu\|^2$ e I_j es el conjunto de observaciones en el nodo j con $j = 1, \ldots, T$, a partir de la Ecuacion (3.25) se extrae el valor de la función objetivo para cada iteración t

$$\tilde{L}^{(t)} = \sum_{j=1}^{T} \left[\left(\sum_{i \in I_j} g_i \right) \nu_j + \frac{1}{2} \left(\sum_{i \in I_j} h_i + \lambda \right) \nu_j^2 \right] + \gamma T, \tag{3.26}$$

donde ν_j es la puntuación en cada nodo y se elige para que su valor óptimo venga dado por

$$\nu_j^* = -\frac{\sum_{i \in I_j} g_i}{\sum_{i \in I_j} h_i + \lambda}. \tag{3.27}$$

Por tanto la estructura que seguirán los árboles vendrá determinada por la siguiente ecuación donde se puede deducir si la división de un nodo en dos nuevos es positiva o no

$$\begin{aligned}
L_{split} &= \frac{1}{2} \left[\frac{(\sum_{i \in I_L} g_i)^2}{\sum_{i \in I_L} h_i + \lambda} + \frac{(\sum_{i \in I_R} g_i)^2}{\sum_{i \in I_R} h_i + \lambda} - \frac{(\sum_{i \in I_{L+R}} g_i)^2}{\sum_{i \in I_{L+R}} h_i + \lambda} \right] - \gamma \\
&= \frac{1}{2} \left[\frac{G_L^2}{H_L + \lambda} + \frac{G_R^2}{H_R + \lambda} - \frac{(G_L + G_R)^2}{H_L + H_R + \lambda} \right] - \gamma,
\end{aligned} \tag{3.28}$$

y en ella se presenta la puntuación del nodo de la izquierda (L), del de la derecha (R), del nodo original $(L + R)$ y la regularización del nodo adicional.

Hiperparámetros de XGBoost

A continuación se van a recoger los diferentes hiperparámetros que se ajustarán a la hora de construir el modelo usando esta técnica y la librería `h2o` de `R`. Notar que existen otros hiperparámetros que no se mencionarán, pero si se quisiera conocer algún otro puede recurrirse a la ayuda de la librería `h2o`.

- **Número de árboles (ntree)**: Determina el número de árboles que se construirán en el modelo.
- **Tasa de aprendizaje (learn_rate)**: Establece el nivel de corrección que se lleva a cabo en cada iteración, así se reduce el peso de la estimación resultante de ajustar cada árbol para que sea más eficiente y así evitar el sobreajuste. Dicho valor se encuentra entre el 0 y el 1, así cuanto más alto sea se alcanza de forma más rápida el mínimo de la función objetivo, sin embargo puede llegar pasarse el óptimo. Por el contrario, si se considera un valor pequeño es posible que no se obtenga el óptimo aunque se utilicen muchas iteraciones.
- **Profundidad máxima (max_depth)**: Determina la profundidad máxima de los árboles. Cuando se tienen valores altos el modelo es más complejo con relaciones más específicas, sin emabrgo si se profundiza demasiado es posible que se lleven a cabo ramificaciones que únicamente van a producir un sobreajuste.
- **Muestras mínimas (min_rows)**: Fija el número mínimo de observaciones que debe tener cada nodo terminal.
- **sample_rate**: Especifica el porcentaje de observaciones de seleccionadas al azar para ajustar cada árbol. Si se toma como 0.5 el algoritmo utiliza la mitad de las observaciones de entrenamiento para ajustar cada árbol. Toma valores en el intervalo $(0, 1]$.
- **col_sample_rate**: Fija el porcentaje de variables explicativas (escogidas al azar) que se utilizan en cada división de cada nivel.
- **col_sample_rate_per_tree**: Se asocia al porcentaje de predictores seleccionados aleatoriamente que se han de utilizar.
- **Término de regularización L1 (reg_alpha)**: Se refiere al término de regularización L1 sobre los pesos de las observaciones. Toma valores entre el 0 y el 1, y cuanto mayor sea dicho valor más conservador será el modelo.
- **Término de regularización L2 (reg_lambda)**: Se refiere al término de regularización L2 sobre los peso. Toma valores entre el 0 y el 1, y al igual que el anterior, cuanto mayor sea dicho valor más conservador será el modelo.

3.3.4. LightGBM

Finalmente se va a presentar el último algoritmo, el mismo surge a partir de Ke et al. (2017) junto con Microsoft con la intención de mejorar los métodos boosting ya existentes. Dicha técnica, a diferencia de las vistas hasta ahora, presenta un tiempo de computación muy inferior a otros algoritmos y Ke et al. (2017) mencionan que el mismo puede llegar a ser 20 veces más veloz que los demás métodos de gradient boosting.

Éste se caracteriza principalmente por la forma en que se construyen los árboles de decisión. Esto se debe a que durante el uso de otros algoritmos de gradient boosting, se analizan todos los posibles puntos de corte a la hora de dividir un nodo, lo que conlleva a un tiempo de ejecución muy alto. Por el contrario el LightGBM trata de considerar aquellos que producen una mayor ganancia y así se obtienen árboles más precisos pero que también pueden presentar un mayor sobreajuste y desequilibrio. Entonces esta técnica para llevar a cabo ese procedimiento, propone dos estrategias basadas en la

reducción del número de observaciones y de variables predictoras en cada iteración, y que se conocen como: GOSS (Gradient-based One-Side Sampling) y EFB (Exclusive Feature Bundling).

Por una parte, GOSS se basa en que las observaciones que son más influyentes en el aumento de la ganancia cuando se lleva a cabo una división en un nodo, son aquellas que tienen gradientes más altos de la función de pérdida. Entonces dicho método procede a reducir el número de observaciones sin perder información importante para la ganancia. Para ello utiliza un muestreo escogiendo así las observaciones con gradientes más altos y una muestra aleatoria de los datos con gradientes más pequeños. A continuación se desarrolla la estrategia o método en cuestión.

Sea O el conjunto de entrenamiento en un nodo específico del árbol de decisión, entonces la ganancia que se obtiene al dividir el nodo utilizando la variable predictora k en el punto de corte s se puede definir como sigue

$$V_{k|O}(s) = \frac{1}{N_O}\left(\frac{(\sum_{\{x_i\in O: x_{ik}\leq s\}} q_i)^2}{N^k_{l|O}(s)} + \frac{(\sum_{\{x_i\in O: x_{ik}> s\}} q_i)^2}{N^k_{r|O}(s)}\right), \tag{3.29}$$

donde el número de observaciones del nodo original es $N_O = \sum \mathbb{I}_{\{x_i\in O\}}$, $N^k_{l|O}(s) = \sum \mathbb{I}_{\{x_i\in O: x_{ik}\leq s\}}$ es el número de observaciones del nodo de la izquierda tras la división del nodo, y $N^k_{r|O}(s) = \sum \mathbb{I}_{\{x_i\in O: x_{ik}> s\}}$ el número del nodo de la derecha. También q_i se refiere al opuesto del gradiente de la función de pérdida evaluado en la predicción proporcionada por el modelo para la observación i. Teniendo en cuenta dicha ecuación, para la variable k el algoritmo selecciona el punto de corte $s^*_k = arg\,\text{máx}_s\, V_k(s)$ y calcula la mayor ganancia con $V_k(s^*_k)$.

Ahora para llevar a cabo el algoritmo, la técnica GOSS propone el siguiente procedimiento aproximando la ganancia:

1. Se ordenan de forma descendiente las observaciones del conjunto de entrenamiento según el valor absoluto de sus gradientes asociados.

2. Se toma un subconjunto A de observaciones formado por la proporción a de observaciones totales que tienen los valores más altos en el anterior paso.

3. Se toma un subconjunto B de observaciones formado por una proporción b de las observaciones que forman el conjunto complementario de A.

4. Se dividen las observaciones en función de la ganancia estimada sobre el subconjunto $A \cup B$, es decir,

$$\tilde{V}_k(s) = \frac{1}{N}\left(\frac{(\sum_{x_i\in A_l} q_i + \frac{1-a}{b}\sum_{x_i\in B_l} q_i)^2}{N^k_l(s)} + \frac{(\sum_{x_i\in A_r} q_i + \frac{1-a}{b}\sum_{x_i\in B_r} q_i)^2}{N^k_r(s)}\right), \tag{3.30}$$

   siendo $A_l = \{x_i \in A : x_{ik} \leq s\}$, $A_r = \{x_i \in A : x_{ik} > s\}$, $B_l = \{x_i \in B : x_{ik} \leq s\}$, $B_r = \{x_i \in B : x_{ik} > s\}$ y por último el coeficiente $\frac{1-a}{b}$ que es usado para normalizar la suma de los gradientes sobre B.

Tras la explicación de esta técnica, es importante comentar que la aproximación comete un error que converge a cero cuando n tiende a infinito, por lo que cuantas más observaciones tiene el conjunto de datos, más precisa es la aproximación. Si se quisiera conocer más información acerca del error cometido se puede recurrir a la Sección 3.2 de Ke et al. (2017).

Por otra parte, se encuentra la técnica EFB basada en reducir el número de predictores. Para ello, puesto que existen variables que en pocas ocasiones toman valores distintos de cero simultáneamente,

intenta tomar éstas variables excluyentes y agruparlas obteniendo así un menor número de predictores. De esta forma se está reduciendo el problema a un problema de coloración de grafos donde se toman las variables explicativas como vértices y donde se añaden aristas por cada par de predictores que no son mutuamente excluyentes. Tras esto se diseña un algoritmo para seleccionar dichos predictores y posteriormente la estrategia EFB introduce un método para combinar las variables utilizando histogramas. Igual que con la estrategia GOSS si se quiere obtener más información, puede verse en la Sección 4 de Ke et al. (2017).

Hiperparámetros de LightGBM

A continuación, al igual que se hizo con los anteriores algoritmos, se presentan los hiperparámetros que se ajustarán posteriormente en `R` utilizando la librería `lightgbm` para construir el modelo LightGBM.

- **Número de iteraciones (num_iterations)**: Número de árboles que se construirán en el modelo.
- **Tasa de aprendizaje (learning_rate)**: Establece la magnitud con la que se reduce el peso en la estimación dada por cada árbol para obtener así un modelo más preciso y evitar el sobreajuste. Si tiene un valor alto llegará más rápido al mínimo de la función objetivo, pero puede llevar a un sobreajuste. Por el contrario, si el valor es demasiado pequeño el proceso será mucho más lento pudiendo no llegar al óptimo.
- **Profundidad máxima (max_depth)**: Determina la profundidad máxima de los árboles. Cuando toma valores altos es más probable que el modelo cometa un sobreajuste pues las divisiones de los nodos son menos importantes.
- **Número máximo de hojas por árbol (num_leaves)**: Representa el número máximo de nodos terminales que debe tener cada uno de los árboles.
- **Ganancia mínima para la partición (min_gain_to_split)**: Establece la ganancia mínima que debe existir para poder realizar una partición en cualquiera de los nodos del árbol.
- **Término de regularización tipo L1 (lambda_l1)**: Fija el término de regularización que controla la penalización L1. Si el valor es alto se obtiene un modelo más conservador.
- **Término de regularización tipo L2 (lambda_l2)**: Fija el término de regularización que controla la penalización L2. Si el valor es alto se obtiene un modelo más conservador.
- **Fracción de predictores (feature_fraction)**: Selecciona aleatoriamente el porcentaje de predictores que se usará en cada iteración (árbol), es decir, si se toma como 0.8 toma el 80 % de explicativas para entrenar el árbol.

3.4. Validación y evaluación de los modelos

Una vez explicados y desarrollados los distintos modelos del Capítulo 3 es conveniente hablar de las distintas técnicas de validación y evaluación de los modelos, pues son las que establecerán la precisión y fiabilidad de los mismos. Existen múltiples métricas y por ello a continuación se explicarán algunas de ellas.

Para comparar los diferentes modelos construidos en el Capítulo 4 se usará principalmente el área bajo la curva ROC denotada como AUC. Dicha curva se utiliza para evaluar un modelo predictivo y consiste en una gráfica donde se recoge la sensibilidad frente al complementario de la especificidad de un sistema de clasificación. Puesto que participan dos conceptos importantes como son la sensibilidad

y la especificidad, las mismas se explican a continuación.

Se considera un problema donde la variable respuesta es categórica y únicamente presenta dos clases, entonces una vez obtenidas las predicciones se tienen cuatro situaciones y con éstas se construye una tabla de contingencia conocida como matriz de confusión. Dicha matriz se construye sin más que enfrentar las predicciones dadas por el modelo de las observaciones y las categorías reales de las mismas. A continuación en la Tabla 3.1 se puede ver reflejado este concepto.

	Predicción	
Observación	$H = 0$	$H = 1$
Y=0	Verdadero negativo (TN)	Falso positivo (FP)
Y=1	Falso negativo (FN)	Verdadero positivo (TP)

Tabla 3.1: Matriz de confusión.

Por una parte se tienen los verdaderos negativos (TN) que son el número de observaciones de la clase $Y = 0$ que el modelo clasifica correctamente como $H = 0$. De la misma forma se pueden definir los verdaderos positivos (TP) pues son aquellas observaciones que son éxitos ($Y = 1$) y que el modelo clasifica correctamente como $H = 1$.

Por otra parte se tienen los falsos negativos (FN) observaciones que son éxitos realmente $Y = 1$ pero que el modelo los clasifica erróneamente como $H = 0$, al contrario de los falsos positivos que son observaciones $Y = 0$ y que el modelo está clasificando como éxito $H = 1$.

Así a partir de estas definiciones se puede definir la especificidad y la sensibilidad que se quería explicar.

- **Sensibilidad (TPR)**: Se trata de la proporción de observaciones con éxito, $Y = 1$, que son clasificados correctamente.
$$TPR = \frac{TP}{TP + FN}. \tag{3.31}$$
- **Especificidad (TNR)**: Se trata de la proporción de observaciones que son fracasos $Y = 0$ que fueron clasificados correctamente como fracasos.
$$TNR = \frac{TN}{TN + FP}. \tag{3.32}$$

Llegados a este punto es conveniente comentar que lo ideal sería aumentar tantos los verdaderos positivos como los negativos simultáneamente, sin embargo, llevar a cabo ésto no suele ser sencillo pues normalmente si aumenta uno el otro disminuye. Ante este problema se plantea la idea de construir un gráfico para medir la precisión de los modelos.

Para ello se representa en el eje x, el complementario de la especificidad, es decir, $FPR = 1 - TNR$ que determinan los falsos positivos, y en el eje y la sensibilidad TPR que son los verdaderos positivos. Entonces lo ideal y perfecto sería encontrarse muy próximos a la esquina superior izquierda, pues se tendría una alta especificidad y sensibilidad que es lo que se busca. Este gráfico es el mencionado anteriormente como curva ROC, y en la Figura 4.9 se puede ver un ejemplo de la misma.

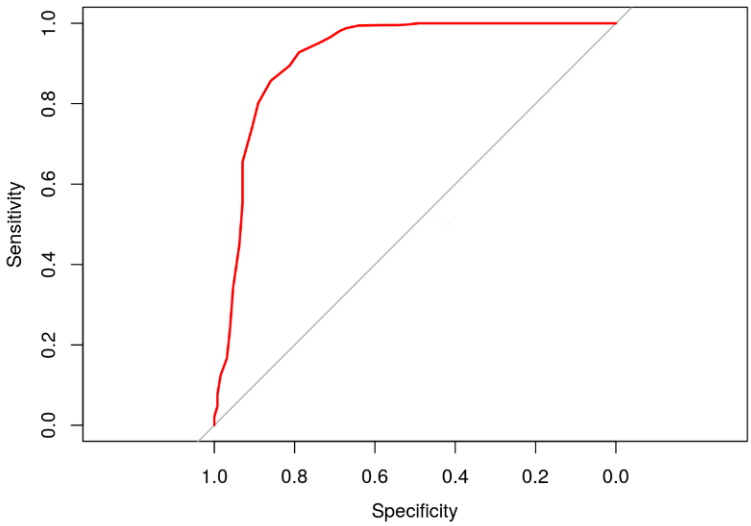

Figura 3.7: Ejemplo de curva ROC.

Ahora que se conoce el concepto de curva ROC, se puede definir el área bajo dicha curva (AUC), que estima la capacidad de diferenciar entre éxitos y fracasos que tiene un modelo, y se puede expresar como

$$AUC = \int_0^1 ROC(p)dp. \tag{3.33}$$

Si se tuviese un valor de 1 para el AUC, la curva pasaría por la esquina superior izquierda y el modelo sería perfecto pues el mismo clasificaría correctamente el 100 % de las observaciones, tanto las consideradas como éxitos $Y = 1$ como los fracasos $Y = 0$. Ahora si dicho valor fuese 0.5 la curva sería una diagonal de la esquina inferior izquierda a la esquina superior derecha, e indicaría que el modelo tiene la misma probabilidad de clasificar las observaciones como éxitos que como fracasos. Luego, si el valor es inferior al 0.5 el modelo realmente no es útil para clasificar las observaciones. Por tanto teniendo en cuenta esto, se considerará un buen modelo aquel que tiene un AUC superior a 0.7.

A la hora de comparar modelos con esta métrica, se dice que un modelo A es uniformemente mejor que un modelo B cuando se verifica lo siguiente

$$ROC_A(p) \geq ROC_B(p), \ \ \forall p \in (0,1), \tag{3.34}$$

por tanto si se cumple esa desigualdad se verifica

$$AUC_A \geq AUC_B. \tag{3.35}$$

Es conveniente mencionar que el inverso no es cierto, pues dada una curva ROC que tiene un AUC superior a otra, ésta no tiene porque ser uniformemente mejor.

Además de las métricas explicadas, a partir de la matriz de confusión también se pueden obtener otras como pueden ser:

- **Tasa de falsos negativos (FNR)**: Proporción de éxitos $Y = 1$ que el modelo clasifica incorrectamente como $H = 0$. Es el complementario de TNR, es decir, $1 - TNR$.
- **Tasa de falsos positivos (FPR)**: Proporción de fracasos $Y = 0$ que el modelo clasifica como $H = 1$ erróneamente. Es el complementario de TPR, es decir, $1 - TPR$.

- **Precisión global o accuracy (ACC)**: Métrica que determina lo próximo que se encuentra la predicción del valor observado original, y viene definido como

$$ACC = \frac{TP + TN}{TP + TN + FP + FN}. \tag{3.36}$$

  Además también se puede obtener el accuracy restando a la unidad, la proporción del error de clasificación que viene dada por

$$err = \frac{FP + FN}{TP + TN + FP + FN}. \tag{3.37}$$

- **Precisión (PPV)**: Se corresponde con la probabilidad de que una predicción positiva, es decir, $H = 1$ en la realidad también sea positiva, es decir, $Y = 1$. Se puede expresar como

$$PPV = \frac{TP}{TP + FP}. \tag{3.38}$$

Este apartado se ha elaborado utilizando Alonzo and Pepe (2002) y Fawcett (2006), pero también Fernández et al. (2021), por lo que para ver más información acerca del AUC y de la curva ROC se puede recurrir a estos artículos.

Capítulo 4

Resultados

Durante el presente capítulo se expondrán los diferentes resultados obtenidos para los modelos predictivos que se han construido para predecir la variable MSINIESTRO, que como ya se comentó, identifica con un 1 la tenencia de un siniestro en los próximos 6 meses y con un 0 su inexistencia.

Tras el tratamiento de los datos, en el Capítulo 2 se comentó que se comenzarían a construir los modelos utilizando 244 variables explicativas. Tras ello lo primero que se debía hacer era dividir el conjunto de datos en tres, y como se explica al inicio del Capítulo 3 se construyeron 3 conjuntos de datos, donde el conjunto de entrenamiento toma el 80 % de los clientes, y el 20 % restante se reparte en dos conjuntos de test y validación. Así se estaba en condiciones de comenzar a construir los modelos, y para entrenarlos se utilizó el conjunto de entrenamiento, para la selección de hiperparámetros el conjunto de validación y para validarlos el conjunto de test.

En las próximas secciones se recogerán los resultados tras ajustar cuatro modelos predictivos, un *Random Forest*, un modelo *GBM*, un modelo *XGBoost* y por último un modelo *LightGBM*. Finalmente para rematar el capítulo se comparará el comportamiento y los resultados de los diferentes modelos para así seleccionar aquel que se considere mejor.

Antes de comenzar, decir que para la elaboración de los distintos modelos, como ya se fue introduciendo, se ha utilizado la librería `h2o` junto con la librería `lightgbm` de `R`. Esto se debe a que `h2o` utiliza una plataforma de código abierto que ayuda a la hora de construir modelos con grandes cantidades de datos, y el uso del `lightgbm` se debe a que la otra librería todavía no cuenta con funciones para la construcción de este tipo de modelos.

4.1. Random Forest

Primeramente para elaborar un modelo *random forest* se ha creado una rejilla con múltiples combinaciones de hiperparámetros entre las que se han seleccionado al azar 200, y para cada una de ellas se ha ajustado un modelo. Tras ello se ha seleccionado aquella combinación de hiperparámetros que proporciona un mejor AUC en el conjunto de validación y se ha creado una segunda rejilla entorno a los valores que toman los hiperparámetros. De esta forma tras ajustar modelos con todas las posibles combinaciones, se volvió a escoger aquel que proporcionaba un mejor resultado en función del AUC del conjunto de validación. Finalmente, tras todas las pruebas realizadas de hiperparámetros se obtuvo la configuración óptima de los mismos, que se recoge en la Tabla 4.1.

Una vez obtenidos los valores, surge la duda acerca de si aquellas variables menos relevantes están

Hiperparámetros	
max_depth	5
min_rows	1000
mtries	26
ntrees	500
sample_rate	0.9

Tabla 4.1: Configuración óptima de los hiperparámetros del modelo random forest.

influyendo bien en el modelo o simplemente están introduciendo ruido provocando un peor ajuste. Ante dicha situación, primero se han extraído las variables más importantes del modelo, y posteriormente se ha probado a eliminar distinto número de variables. Así tras varias pruebas se concluye que con las 39 variables más importantes que se muestran en la Figura 4.1 se mejoraban ligeramente los resultados.

Figura 4.1: Importancia de las variables del random forest.

Una vez entrenado el modelo, es momento de evaluar el mismo en un nuevo conjunto de datos, es decir, en el conjunto de test. Entonces al aplicar el modelo sobre este conjunto se obtiene la siguiente matriz de confusión mostrada en la Tabla 4.2, donde también se muestran algunas métricas de validación explicadas en el anterior capítulo.

	Predicción		
Observación	$H = 0$	$H = 1$	
Y=0	21222	15658	FPR=0.42
Y=1	1603	1986	FNR=0.45
	ACC=0.57	err=0.43	

Tabla 4.2: Matriz de confusión del random forest.

Por una parte se puede ver que se comete un mayor error a la hora de clasificar éxitos (Y=1) pues la tasa de falsos negativos es de 0.45 frente a la tasa de falsos positivos de 0.42. Sin embargo, la diferencia es bastante mínima lo que indica que el modelo no clasifica bien ninguna de las dos categorías. Por otra parte se tiene un valor de accuracy de 0.57 que establece que el modelo no es muy preciso y por lo tanto se encuentra bastante lejos de clasificar correctamente nuevas observaciones.

A continuación, se representa en la Figura 4.2 la curva ROC asociada a este modelo que produce el AUC de test de 0.595. Como se puede ver la curva no se aproxima a la esquina superior izquierda como debería suceder para que fuese un modelo bueno, por lo tanto se puede afirmar que el modelo no es muy adecuado para predecir la variable respuesta. No obstante, se podría pensar que el modelo quizá este sobreajustado y que por lo tanto no se estuviese obteniendo un buen ajuste, pero el AUC del conjunto de entrenamiento, y del conjunto de validación son 0.611 y 0.598, respectivamente. Por lo que el modelo no está cometiendo dicho error, simplemente no es un buen modelo para predecir la variable MSINIESTRO.

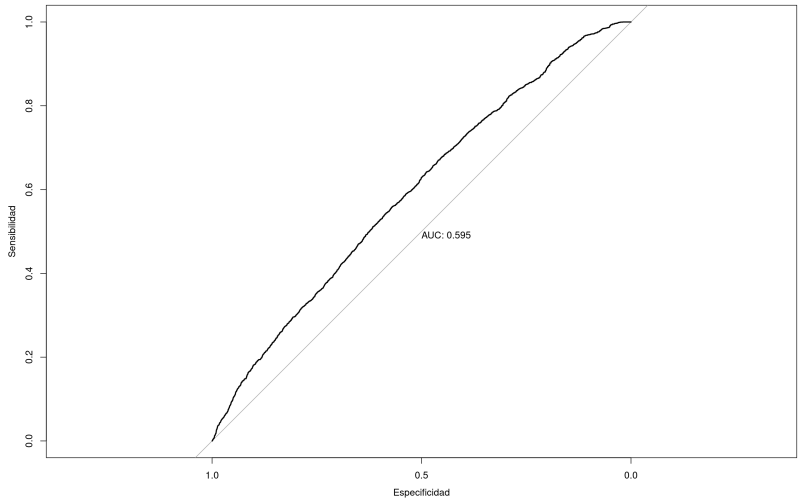

Figura 4.2: Curva ROC del random forest.

4.2. GBM

Para construir un modelo GBM se ha creado una rejilla con múltiples valores para los diferentes hiperparámetros comentados y definidos en el apartado de hiperparámetros de la Sección 3.3.2. En la misma se ha comenzado con valores muy extremos de los mismos, y tras ajustar 200 modelos con 200 combinaciones seleccionadas al azar se extrae el mejor modelo utilizando el AUC del conjunto de validación. Posteriormente, de manera análoga al caso del random forest, se elabora una segunda rejilla con valores de hiperparámetros próximos a los obtenidos antes, y tras elaborar distintas combinaciones se obtiene que la configuración óptima de los mismos es la recogida en la Tabla 4.2.

A continuación se analizan las variables más importantes del modelo, y en la Figura 4.3 se recogen las 40 más importantes, así se aprecia que la más relevante es, de forma análoga a lo sucedido con el modelo random forest, la antigüedad de los vehículos, seguida del precio del vehículo, el grupo de empleo del asegurado y el score financiero del cliente. Como se puede ver las 4 más importantes coinciden con las del modelo anterior lo cual indica que ambos modelos siguen un mismo camino.

Hiperparámetros	
col_sample_rate	0.55
col_sample_rate_per_tree	0.53
learn_rate	0.035
max_depth	3
min_rows	510
ntrees	105
sample_rate	0.6

Tabla 4.3: Configuración óptima de los hiperparámetros del modelo GBM.

Figura 4.3: Importancia de las variables del GBM.

De nuevo se ha probado a eliminar variables del modelo para ver si realmente son importantes y aportan información a la hora de clasificar nuevas observaciones o si únicamente influyen de manera negativa al mismo. En este caso primero se ha decidido eliminar las variables referentes a características del vehículo que presentaban un alto porcentaje de faltantes, y se han obtenido unos mejores resultados pues las mismas debían estar introduciendo ruido al modelo provocando así un peor ajuste. Tras ello, se ha decidido eliminar parte de las variables menos importantes, y así quedándose con la totalidad de 95 variables se obtienen los mejores resultados.

Ahora es el momento de evaluar dicho modelo en una nueva muestra de datos, en este caso se toma el conjunto de test que conformaba el 10 % de los clientes del conjunto original. Tras la aplicación del modelo al mismo se obtiene la matriz de confusión que se muestra en la Tabla 4.4.

En este caso como se puede observar en la matriz de confusión el modelo clasifica bastante mal los éxitos puesto que la tasa de falsos negativos es de 0.54, y dicha categoría es la que realmente interesa predecir. Por esta razón se podría decir que el modelo no es bueno a la hora de clasificar la nuevas observaciones que realmente son éxitos puesto que clasifica más de la mitad de forma incorrecta. No obstante ofrece mejores resultados a la hora de clasificar los fracasos.

A continuación, en la Figura 4.4 se presenta la curva ROC asociada al modelo GBM junto con el

	Predicción		
Observación	$H = 0$	$H = 1$	
Y=0	25254	11626	FPR=0.32
Y=1	1938	1651	FNR=0.54
	ACC=0.56	err=0.44	

Tabla 4.4: Matriz de confusión del GBM.

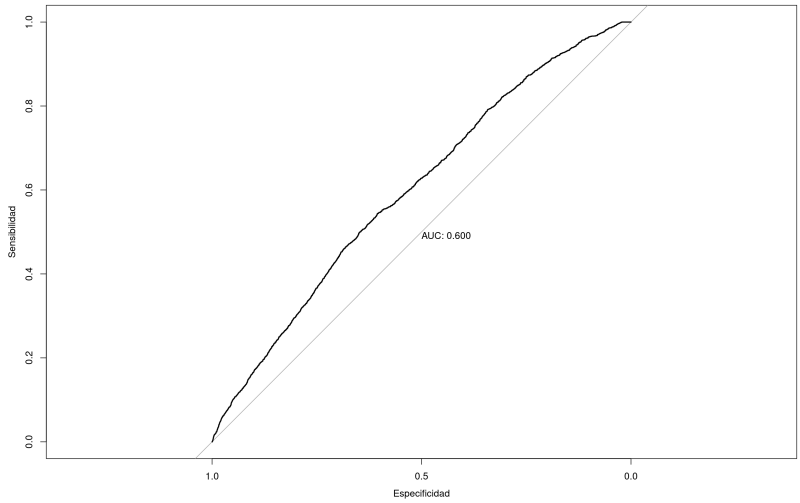

Figura 4.4: Curva ROC del GBM.

valor correspondiente de AUC de 0.600. Ante dicho valor de AUC se puede decir que el modelo no es bueno para clasificar nuevas observaciones, y se podría dudar de un posible sobreajuste a la hora de su construcción. Sin embargo, los valores de AUC que toman los conjuntos de entrenamiento y de validación son 0.617 y 0.600, respectivamente, por lo que dicho problema en este caso no se da.

4.3. XGBoost

En esta ocasión, se va a considerar un modelo XGBoost, y de igual manera que se procedió con los anteriores, se crea una rejilla con distintos valores para los hiperparámetros comentados en la Sección 3.3.3. Se ajustaron un total de 150 modelos con 150 combinaciones de hiperparámetros seleccionados al azar de la rejilla, esto es porque computacionalmente es más costoso que los dos modelos construidos antes. Tras ello se seleccionó aquel que proporcionaba un mejor valor en términos de AUC en el conjunto de validación. Luego, se volvió a ajustar múltiples combinaciones de hiperparámetros, pero tomando valores próximos a los que toman los mismos en el mejor modelo. Así finalmente, se concluye que la configuración óptima de hiperparámetros es la que se muestra en la Tabla 4.3.

A continuación se muestran las variables más importantes del modelo, y se pueden ver en la Figu-

Hiperparámetros	
col_sample_rate	0.7
col_sample_rate_per_tree	0.45
learn_rate	0.03
max_depth	2
min_rows	500
ntrees	500
reg_alpha	0.2
reg_lambda	1
sample_rate	0.45

Tabla 4.5: Configuración óptima de los hiperparámetros del modelo XGBoost.

ra 4.5. Igual que en los anteriores modelos, se muestra que la variable más importante en el modelo es la antigüedad de los vehículos, seguida del precio y del saldo en pasivo como cualquier titular. Además, se probó a eliminar las variables menos importantes por si éstas estuviesen perjudicando la precisión del modelo. No obstante, esto no sucedía por lo que el mejor modelo en este caso es aquel que tiene un total de 232 variables. Notar que en este caso aparecen variables del estilo AMBITO_CLIENTE.RURAL que se trata de una variable dummy, pues este tipo de modelos crea estas variables a partir de las variables categóricas.

Figura 4.5: Importancia de las variables del XGBoost.

En este contexto, es el momento de evaluar el modelo seleccionado, y para ello se utiliza el conjunto de test. Al aplicar el modelo a dicho conjunto de datos, se obtiene la matriz de confusión que se recoge en la Tabla 4.6. Se puede ver que el accuracy en éste caso es de 0.7, que es bastante más alto que en los otros casos. No obstante, la tasa de falsos negativos es bastante alta, clasificándose incorrectamente casi el 60 % de los siniestros. Esto puede ser un problema, puesto que a pesar de que el modelo clasifica bastante bien los ceros (no siniestro), lo que se busca con el modelo es predecir los siniestros y comete

un error bastante elevado.

	Predicción		
Observación	$H = 0$	$H = 1$	
Y=0	27035	9845	FPR=0.27
Y=1	2127	1462	FNR=0.59
	ACC=0.70	err=0.30	

Tabla 4.6: Matriz de confusión del XGBoost.

Finalmente, se representa en la Figura 4.6 la curva ROC del modelo, junto con el valor de AUC que proporciona, 0.603. De nuevo el modelo no es bueno para predecir los siniestros en nuevas observaciones, y dudando de un posible problema de sobreajuste. No obstante, los conjuntos de entrenamiento y validación, tienen unos valores de 0.620 y 0.606, respectivamente. Por esto mismo se puede decir que no existe dicho problema, por lo que el modelo simplemente no es bueno para predecir lo que se busca.

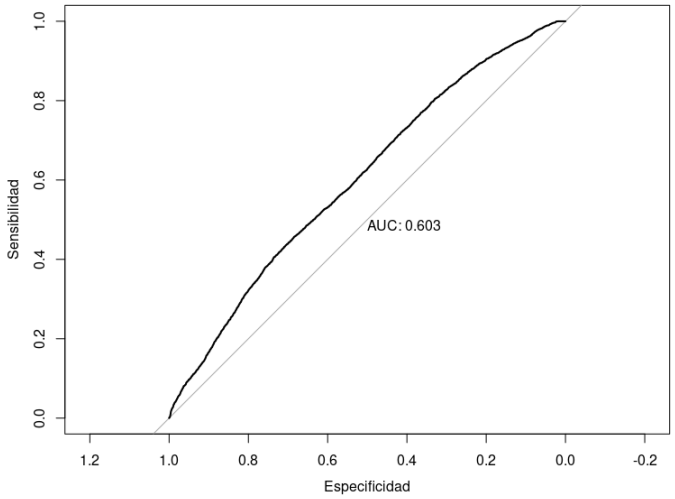

Figura 4.6: Curva ROC del XGBoost.

4.4. LightGBM

Finalmente, se decidió construir un modelo LightGBM, y para ello, a diferencia de los demás modelos, se ha utilizado la librería `lightgbm` de `R`. Para comenzar de forma análoga a los casos anteriores, se creó una rejilla con múltiples valores para los hiperparámetros. Para este algoritmo se han realizado un total de 1500 pruebas iniciales, la diferencia en número de pruebas con los demás se debe a que el mismo es menos costoso computacionalmente.

Tras la prueba se seleccionaron aquellos valores de hiperparámetros que otorgaban un mejor valor de AUC en el conjunto de validación. Posteriormente se elaboraron 81 combinaciones con valores de hiperparámetros muy próximos a los de la mejor combinación, y se obtuvo que la combinación óptima de hiperparámetros en este caso es la recogida en la Tabla 4.7.

Hiperparámetros	
num_iterations	1550
learning_rate	0.0015
min_gain_to_split	10
max_depth	5
num_leaves	700
lambda_l1	0
lambda_l2	0
feature_fraction	0.75

Tabla 4.7: Configuración óptima de los hiperparámetros del modelo LightGBM.

Posteriormente se analizaron las variables más importantes del modelo ajustado, y en la Figura 4.7 se pueden apreciar las 50 más importantes. En este caso la variable más importante es el precio del vehículo, seguida de la antigüedad del mismo y el saldo medio de pasivo en el último mes en los contratos como cualquier titular. De nuevo, se hacen pruebas eliminando variables poco importantes para ver si el modelo en cuestión mejora su precisión. De esta forma se obtienen los mejores resultados, con un total de 73 variables.

Figura 4.7: Importancia de las variables del LightGBM.

Es momento de evaluar el modelo utilizando el conjunto de datos de test. Al evaluar el mismo se obtiene la matriz de confusión que se puede ver en la Tabla 4.8. Fácilmente se puede ver que el valor de Accuracy es de 0.57, luego la tasa de falsos positivos es 0.43 y la de falsos negativos 0.55. En este caso resulta lo mismo que ocurría con el modelo GBM, la tasa de falsos negativos es superior al 0.5. Por esta razón se clasifican más de la mitad de los unos (siniestro en los próximos 6 meses) incorrectamente.

En cuanto a la otra categoría, el modelo tampoco actúa correctamente pues comete un error bastante elevado también.

	Predicción		
Observación	$H = 0$	$H = 1$	
Y=0	21070	15810	FPR=0.43
Y=1	1600	1989	FNR=0.55
	ACC=0.57	err=0.43	

Tabla 4.8: Matriz de confusión del LightGBM.

Finalmente, es hora de representar en la Figura 4.8 la curva ROC que da lugar al valor de AUC resultante de este modelo, que en este caso es 0.598. Se tiene una curva que no se aproxima a la esquina superior izquierda, y el AUC no supera el umbral del 0.7. Por esta razón, se puede afirmar que el modelo no es bueno para predecir la variable MSINIESTRO en nuevas observaciones.

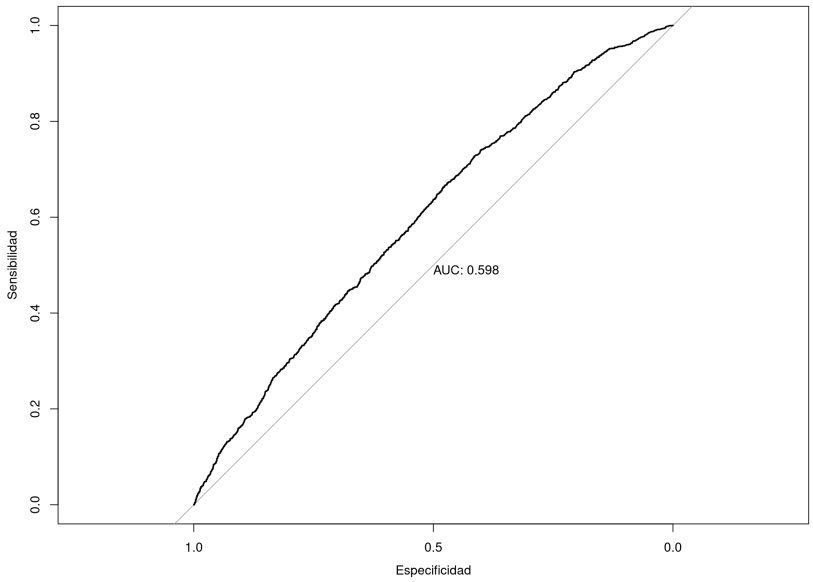

Figura 4.8: Curva ROC del LightGBM.

4.5. Comparación de modelos

Una vez recogidos los distintos resultados obtenidos para los cuatro modelos predictivos se van a comparar los mismos para seleccionar finalmente aquel modelo que ofrezca unos mejores resultados con el objetivo de clasificar nuevas observaciones en las categorías de MSINIESTRO. No obstante antes de comenzar es importante comentar que los resultados de los cuatro modelos en general son

bastante malos, ninguno ofrece unos resultados lo suficientemente buenos como para considerar ese modelo adecuado para predecir la variable respuesta MSINIESTRO en nuevas observaciones.

Cabe mencionar también, que a pesar de esos resultados se han probado múltiples combinaciones de hiperparámetros, a elaborar modelos con diferente número de variables, éstas también se han intentado modificar creando variables que agruparan a otras como puede ser una variable que recoge el importe total de los recibos de los clientes, o el número total de seguros del mismo. A pesar de todos los intentos y pruebas realizadas, no se ha conseguido mejorar ninguno de los modelos por lo que habría que valorar otras opciones como se explica en la Sección 5.2. Así tras este comentario, se comienza la comparación de los cuatro modelos construidos a lo largo del proyecto.

Para seleccionar el mejor modelo, se va a utilizar la métrica del área bajo la curva ROC (AUC). La comparación de las cuatro curvas procedentes de los modelos construidos se puede ver en la Figura 4.9. En este caso, como ya se mencionó los modelos no son muy buenos en términos de AUC, pues ninguno supera un valor de 0.61. Esto quiere decir que se comete mucho error a la hora de predecir nuevas observaciones, no obstante, se decide que el mejor siguiendo dicha métrica modelo es el modelo XGBoost o el GBM con valores AUC de 0.603 y 0.600, respectivamente. Como peor modelo se seleccionaría el Random Forest, aunque solamente difiere en 0.003 en el valor de AUC con el LightGBM.

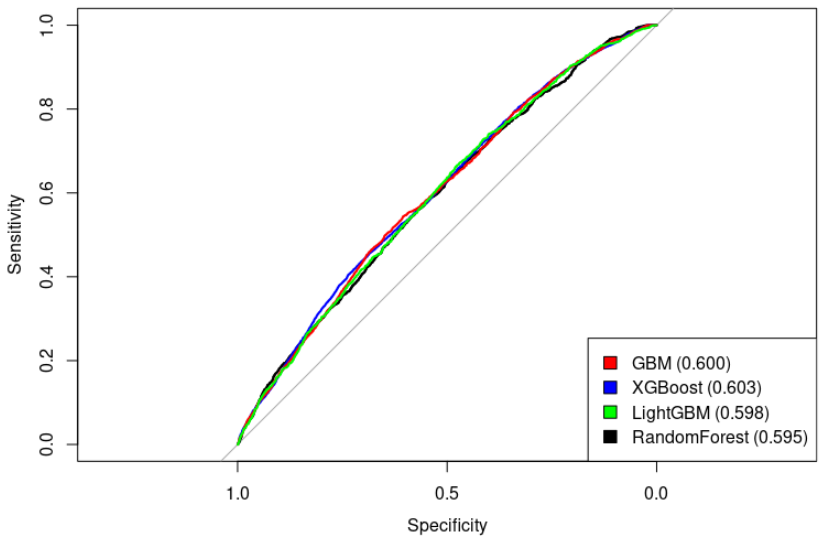

Figura 4.9: Curvas ROC de los diferentes modelos.

Por otro lado, si que es cierto que el modelo que comete menos error a la hora de clasificar los éxitos (siniestros en los próximos 6 meses) es el Random Forest, pues su tasa de falsos negativos es 0.45. Esto supone que el modelo clasifica correctamente más de la mitad de los éxitos en observaciones nuevas, a diferencia del resto de modelos que cometían un error por encima del 0.5. Este modelo lo que hace es equilibrar ambos errores, de forma que comete un error muy similar a la hora de predecir ambas categorías. Por esta razón, se podría decir que dependiendo del objetivo a predecir se debería seleccionar un modelo u otro.

Así, si se quieren predecir los éxitos (siniestros en los próximos seis meses) se podría tomar el modelo random Forest, no obstante, si se quisieran predecir los fracasos (no siniestro es lo próximos seis meses) sería mejor utilizar el XGBoost. Sin embargo, como se comentó se selecciona el mejor modelo

en función del valor AUC, por lo que se puede concluir en este caso que el XGBoost es el mejor modelo.

4.6. Tratamiento del problema de desbalanceo

En el desarrollo del trabajo y especialmente en este capítulo, se puede ver que los resultados obtenidos no han sido demasiado buenos. Esto se debe a que los AUC resultantes de los distintos modelos estudiados han sido inferiores al 0.7, que se mencionó que era el umbral para diferenciar un modelo bueno de uno que no lo es suficientemente.

No obstante, como se comentó en la Sección 2.3, y en particular debido a la gráfica recogida en la Figura 2.3 los datos con los que se ha trabajado sufrían un problema bastante común, conocido como problema de desbalanceo. En este caso, la variable MSINIESTRO, que cuenta con dos categorías, recoge un porcentaje muy elevado de ceros (91.30 %) a diferencia del 8.70 % que presenta con unos, es decir, con siniestros en los próximos seis meses.

Dada la situación, la precisión de los modelos tratados y explicados a lo largo de este apartado puede haberse visto perjudicada, dando lugar a modelos poco precisos. Entonces, se decide probar con algún método para solventar este problema basado en técnicas de resmuestreo, y antes de nada se presentan tres técnicas, aunque existen muchas otras.

- **Undersampling**: Este procedimiento elabora un remuestreo aleatorio de los datos de la clase mayoritaria, hasta que se obtiene un número de muestras igual al de la clase minoritaria. Se puede realizar esto utilizando reemplazamiento o no, pero en este caso se decide proceder sin reemplazamiento. El objetivo es evitar que se tome más de una observación igual en la clase mayoritaria y así contar con más información.

- **Oversampling**: En este caso se procede con un remuestreo aleatorio de la categoría minoritaria, hasta que se consiga un conjunto de datos balanceado. La misma lo que hace es duplicar observaciones de la clase minoritaria, y se repetirán un número de veces que dependerá del nivel de balance que se quiera conseguir.

- **Synthetic minority oversampling technique (SMOTE)**: Es un algoritmo desarrollado por Chawla et al. (2002), y procede aumentando el número de datos al crear observaciones sintéticas basadas en las originales. Para ello, introduce nuevos casos en los segmentos que unen cada observación con sus k vecinos más próximos. Para establecer el valor de k es necesario conocer cuantas observaciones se quieren crear. La diferencia de este método con el *oversampling* se encuentra en que en este caso, no hay observaciones repetidas. Si se quisiera conocer más información sobre esta técnica se puede recurrir a Chawla et al. (2002).

En este contexto, se deciden ajustar los modelos seleccionados y comentados en el Capítulo 4 pero habiendo tratado primeramente los datos con las técnicas mencionadas. Posteriormente se compararán los resultados proporcionados por cada una y se concluirá si tener en cuenta algún método de remuestreo es útil en este caso.

4.6.1. Resultados

Se comienza construyendo el modelo *random forest*, y la primera técnica planteada es el **undersampling**. En esta ocasión, como se ha mencionado antes, se elabora un remuestreo aleatorio de la categoría mayoritaria, que en este caso son los fracasos (no siniestro en los 6 meses siguientes) hasta

obtener un número igual de muestras que la clase de los unos. Luego se construye el modelo *random forest*, y se evalúa en el mismo conjunto de datos de test que se usó para evaluar el modelo sin remuestreo. Así se obtiene la siguiente matriz de confusión recogida en la Tabla 4.9.

	Predicción		
Observación	$H = 0$	$H = 1$	
Y=0	23181	13699	FPR=0.37
Y=1	1825	1764	FNR=0.51
	ACC=0.62	err=0.38	

Tabla 4.9: Matriz de confusión del random forest con undersampling.

En este caso el valor de accuracy ha aumentado pasando de un 0.57 a un 0.62, con lo cual la precisión global mejora. No obstante, a pesar de reducir la tasa de falsos positivos, clasificando correctamente más ceros, la tasa de falsos negativos se ha incrementado. Así, se clasifican incorrectamente más de la mitad de las nuevas observaciones que son unos. Por otra parte, el valor del AUC es de 0.583, inferior al 0.595 proporcionado por el modelo sin remuestreo.

A continuación, se procede de forma análoga utilizando la técnica del **oversampling**, y tras probar el modelo en el conjunto de test se obtiene la matriz de la Tabla 4.10.

	Predicción		
Observación	$H = 0$	$H = 1$	
Y=0	23926	12954	FPR=0.35
Y=1	1878	1711	FNR=0.52
	ACC=0.63	err=0.37	

Tabla 4.10: Matriz de confusión del random forest con oversampling.

El valor del acurracy aumenta hasta un 0.63, incrementando la precisión global. En cuanto a la tasa de falsos negativos, la misma aumenta cometiendo un mayor error a la hora de clasificar los éxitos en nuevas observaciones. Esto seguramente se deba, a que el modelo ha conseguido reducir el error cometido en la otra clase, por lo que ha aumentado el de los éxitos. Con respecto al AUC obtenido, éste es de 0.596 aumentando en una milésima el del modelo sin remuestreo. Esta mejora realmente no es relevante, simplemente ha mejorado al clasificar los no siniestros en los próximos 6 meses, en lugar de los siniestros.

Finalmente, se elabora esto mismo realizado hasta ahora pero aplicándole previamente el algoritmo **SMOTE** a los datos. Así la matriz de confusión, tras aplicarle al nuevo modelo el conjunto de test es la recogida en la Tabla 4.11. Con este algoritmo lo que se ha conseguido es reducir drásticamente el error cometido al clasificar los éxitos de nuevas observaciones, puesto que la tasa de falsos negativos ha pasado a ser de 0.27. Simultáneamente, se ha producido un incremento muy pronunciado en la

tasa de falsos positivos, lo que provoca que el accuracy se reduzca gravemente hasta un 0.4. En cuanto al AUC, éste toma el valor de 0.567, que claramente es peor que los obtenidos con los otros métodos.

	Predicción		
Observación	$H = 0$	$H = 1$	
Y=0	13433	23447	FPR=0.64
Y=1	955	2634	FNR=0.27
	ACC=0.40	err=0.60	

Tabla 4.11: Matriz de confusión del random forest con SMOTE.

Llegados a este punto, se ha decidido mostrar en la Tabla 4.12 los resultados de las métricas de los cuatro modelos utilizando las 3 técnicas de remuestreo y sin remuestreo. En cuanto al modelo GBM, se puede ver que utilizando *oversampling* se consigue igualar el AUC, no obstante con los otros dos métodos empeora. En cuanto al accuracy, se mejora con cualquiera de las técnicas pero especialmente con el *oversampling*, y con este mismo método se consigue mejorar la tasa de falsos positivos, a pesar de incrementar el error en la otra clase. En general, parece que las técnicas aplicadas no son efectivas.

A continuación, se elaboró lo mismo para un modelo XGBoost, y en este caso no se obtuvo ningún modelo que se considere mucho mejor que el modelo sin remuestreo. Los valores de AUC son peores, y en cuanto a las tasas de falsos positivos y negativos, se tienen errores más altos o similares. La precisión global (accuracy) no se consiguió mejorar demasiado, por lo que se podría concluir que el modelo sin remuestreo es el mejor de ellos.

Por último se tiene un LightGBM, en términos de AUC y de accuracy ninguna de las técnicas consigue mejorar los valores dados por el modelo sin remuestreo. La tasa de falsos negativos disminuye en todas, de forma que se reduce el error a la hora de clasificar nuevas observaciones positivas, aunque la mejora no es demasiado drástica. Esto hace que la tasa de falsos positivos empeore, cometiéndose un mayor error al clasificar los fracasos.

A la vista de los resultados, se puede confirmar que el modelo que mejor se adapta para predecir los siniestros en los próximos 6 meses es el XGBoost visto en la Sección 4.3. Esto se debe a que, en este estudio, el tratamiento del problema de desbalanceo no ha conseguido mejorar la precisión de los modelos.

Modelo	**Técnica**	**AUC**	**FPR**	**FNR**	**ACC**
Random Forest	Sin Remuestreo	0.595	0.42	0.45	0.57
	UnderSampling	0.583	0.37	0.51	0.62
	OverSampling	0.596	0.35	0.52	0.63
	SMOTE	0.567	0.64	0.27	0.4
GBM	Sin Remuestreo	0.600	0.32	0.54	0.56
	UnderSampling	0.597	0.35	0.51	0.63
	OverSampling	0.600	0.31	0.55	0.67
	SMOTE	0.56	0.40	0.51	0.59
XGBoost	Sin Remuestreo	0.603	0.27	0.59	0.70
	UnderSampling	0.600	0.37	0.50	0.62
	OverSampling	0.602	0.26	0.60	0.71
	SMOTE	0.565	0.56	0.34	0.46
LightGBM	Sin Remuestreo	0.598	0.43	0.55	0.57
	UnderSampling	0.576	0.47	0.43	0.53
	OverSampling	0.596	0.46	0.40	0.55
	SMOTE	0.528	0.51	0.45	0.49

Tabla 4.12: Métricas de los cuatro modelos sin remuestreo, con oversampling, undersampling y SMOTE.

Capítulo 5

Conclusiones y líneas futuras

Para finalizar este proyecto, a continuación se comentarán las conclusiones finales de lo obtenido con el mismo, y se mencionarán posibles líneas futuras de trabajo que podrían ser aplicadas para mejorar lo estudiado hasta el momento.

5.1. Conclusiones finales

Como se comentó al inicio de este trabajo, el objetivo era construir un modelo, utilizando *machine learning*, para predecir en clientes nuevos del seguro de automóvil si tendrían un siniestro en los siguientes seis meses. Esto sería realmente útil a la hora de asignar las primas a los mismos, ya que si el modelo predice que el cliente tendrá un siniestro, se le debería ofrecer un precio más elevado.

Durante el transcurso del proyecto, y en particular, en el Capítulo 4 se probaron múltiples modelos de clasificación, estando basados tres de ellos en el concepto de *boosting*, una técnica de ensamblado. La misma parte de modelos débiles, con poco potencia predictiva, y los combina hasta formar otro con una capacidad predictiva elevada. Luego, tras evaluar los diferentes modelos se concluyó que el mejor modelo en este caso es un modelo XGBoost sin remuestreo. Cabe decir, que se probó a tratar el problema de desbalanceo presente en los datos, pero los resultados obtenidos no mejoraron.

En este contexto, la situación del problema de siniestralidad se comentó en el Capítulo 1, y se destacó que la mayoría de modelos se basaban en el uso de variables referidas a características de los vehículos. Sin embargo, el objetivo era construir, a partir de variables bancarias (variables diferentes de las que tienen las aseguradoras), un modelo para la siniestralidad que luego pudiera aportar información a la aseguradora. Es decir, que una vez creado el modelo, la aseguradora pudiese incorporar la información resultante del mismo, para ayudar a otorgar las primas. Por tanto, se puede decir que el objetivo en este caso, se ha alcanzado.

Finalmente, me gustaría destacar que la elaboración de este Trabajo Fin de Máster me ha ayudado a desarrollar y evolucionar, tanto en el ámbito estadístico como en el financiero. Con el mismo he aumentado mis conocimientos en ambas áreas y he podido entender como la entidad se plantea, desarrolla y construye modelos estadísticos con el fin de mejorar como institución.

5.2. Líneas futuras

Ante la situación del trabajo y del modelo construido, se pueden plantear diversas opciones a futuro, y entre ellas se encuentra una que durante el transcurso del trabajo se intento llevar a cabo, pero

por falta de información no se pudo aplicar por el momento.

De lo que se habla es de la introducción de unas variables explicativas en los modelos y las mismas se obtienen mediante técnicas de *web scraping*. Dicha herramienta es especialmente utilizada a la hora de introducir en modelos estadísticos variables nuevas que se encuentran en la web y que por lo tanto deben ser extraídas.

En este caso lo que se busca es extraer información relativa a multas que se suele publicar todos los meses en el Boletín Oficial del Estado (BOE), y más concretamente en las Notificaciones del Ministerio del Interior en el área de Jefatura Central de Tráfico. Esta información se publica en documentos de tipo PDF, y son relativas a multas de tráfico que no han podido ser notificadas a los conductores por considerarse desconocidos por diversos motivos: no poseer Dirección Electrónica Vial, desconocimiento del domicilio del mismo... Notar que habría que analizar si se puede utilizar dicha información sin incumplir ninguna ley.

En estos documentos se recogen unas tablas donde aparece información como el documento nacional de identidad (DNI) de la persona multada, localidad del mismo, matrícula del vehículo, cuantía de la multa en euros, fecha de la multa, y a mayores el real decreto y el artículo que recogen el motivo de la sanción. Además en alguna de las observaciones se recoge también la pérdida de puntos que supone dicha multa, y también aparece alguna otra columna en las tablas pero las mismas no son de utilidad.

Una vez comprendida la información relevante de dichos documentos, se quiere extraer la misma para que posteriormente pueda ser utilizada en la construcción de modelos. Sin embargo, es importante comentar que los mismos solamente se encuentran accesibles durante 3 meses después de la fecha de su publicación, por lo que no se tiene por el momento toda la información necesaria para que se pueda utilizar ya que se empezó a extraer las tablas con información desde septiembre de 2023. Por este motivo no se ha llegado a utilizar la misma, pero si se ha extraído toda la información posible ya que a futuro sería interesante poder automatizar dicho proceso y utilizar las variables en los modelos. A continuación en la Tabla 5.1 se ha añadido un ejemplo de como sería una tabla de un documento donde se podría ver la información comentada antes. Destacar que REQ, EXPEDIENTE y DENUNCIADO/A son columnas que no van a interesar en este caso, por lo que solamente se usarán las restantes.

EXPEDIENTE	DENUNCIADO/A	IDENTIF	LOCALIDAD	FECHA	MATRÍCULA	CUANTÍA EUROS	PRECEPTO	ARTº	PTOS	REQ
–	–	–	–	–	–	–	–	–	–	–
–	–	–	–	–	–	–	–	–	–	–

Tabla 5.1: Ejemplo de tabla con información de multas.

Además de lo explicado hasta ahora podrían tratarse otras líneas de trabajo como pueden ser las siguientes:

- **Incorporación de la información por parte de la aseguradora**: Como se comentó, el objetivo era construir el modelo para que luego la aseguradora pudiese añadir la información al modelo que utilizan para otorgar las primas. Por tanto, a futuro sería interesante que la aseguradora procediese de dicha manera, analizando la mejora que supondría la incorporación de la misma a su modelo. Destacar que esto no se ha podido llevar a cabo durante el desarrollo del trabajo, puesto que ABANCA y la compañía de seguros funcionan independientemente.

- **Nuevas metodologías**: Puesto que los modelos predictivos que se han utilizado en el desarrollo del proyecto no presentan una precisión elevada, a pesar de las múltiples pruebas y cambios realizados, sería interesante valorar nuevos métodos. Se podrían plantear *Redes Neuronales*, *Support Vector Machines* o incluso modelos lineales generalizados, como posibles modelos para predecir

la variable respuesta. También se podrían estudiar otros métodos para combatir el problema de desbalanceo en los datos.

- **Nuevas variables**: Al igual que la introducción de las variables mencionadas anteriormente sobre sanciones, estaría bien intentar crear otras variables que puedan ser de utilidad para el modelo. Incluso, buscar otro tipo de información utilizando técnicas de *web scraping*.
- **Nuevo planteamiento del problema**: En este proyecto se ha buscado predecir los siniestros a 6 meses sin importar el tipo de siniestro que el asegurado ha informado. Por esta razón podría pensarse en crear un nuevo modelo donde la variable respuesta se cree a 6 meses también, pero que tome diferentes clases, dando lugar así a un problema multiclase. Así cada clase representaría la gravedad del siniestro, por ejemplo, podrían considerarse 4 clases referidas a NO SINIESTRO, LEVE, MEDIO, GRAVE.

Bibliografía

Abdelhadi, S., Abdelsalam, M., and Elbahnasy, K. (2020). A proposed model to predict auto insurance claims using machine learning techniques. *Journal of Theoretical and Applied Information Technology*, 98(22).

Alonso, J. C. (2022). *Empezando a transformar bases de datos con R y dplyr*. Editorial Universidad Icesi.

Alonzo, T. A. and Pepe, M. S. (2002). Distribution-free roc analysis using binary regression techniques. *Biostatistics*, 3(3):421–32.

Beaulieu, A. (2009). *Learning SQL, Second Edition*. O'Reilly Media.

Breiman, L. (1996). Bagging predictors. *Mach Learn*, 24:123–140.

Breiman, L. (2001). Statistical modeling: The two cultures (with comments and a rejoinder by the author). *Statistical Science*, 16(3):199–231.

Breiman, L., Friedman, J., Stone, C. J., and Olshen, R. A. (1984). *Classification and Regression Trees (1st ed.)*. Chapman and Hall/CRC.

Chawla, N. V., Bowyer, K. W., Hall, L. O., and Kegelmeyer, W. P. (2002). Smote: Synthetic minority over-sampling technique. *Journal of Artificial Intelligence Research*, 16:321–357.

Chen, T. and Guestrin, C. (2016). Xgboost: A scalable tree boosting system. *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*, pages 785–794.

Fauzan, M. A. and Murfi, H. (2018). The accuracy of xgboost for insurance claim prediction. *International Journal of Advances in Soft Computing and its Applications*, 10(2):159–171.

Fawcett, T. (2006). An introduction to roc analysis. *Pattern Recognition Letters*, 27(8):861–874.

Fernández, R., Costa, J., and Oviedo, M. (2021). *Aprendizaje estadístico*. https://rubenfcasal.github.io/aprendizaje_estadistico/.

Freund, Y. and Schapire, R. E. (1996). Schapire r: Experiments with a new boosting algorithm. *in: Thirteenth International Conference on ML*.

Friedman, J. H. (2001). Greedy function approximation: a gradient boosting machine. *The Annals of Statistics*, pages 1189–1232.

Grimmett, G. and Stirzaker, D. (2001). *Probability and Random Processes*. Oxford University Press.

Guillen, M., Nielsen, J. P., Ayuso, M., and Pérez-Marín, A. M. (2018). The use of telematics devices to improve automobile insurance rates. *Risk Analysis*, 39(3):662–672.

Hanafy, M. and Ming, R. (2021). Machine learning approaches for auto insurance big data. *Risks*, 9(2):42.

Hastie, T., Tibshirani, R., and Friedman, J. (2013). *The elements of Statistical Learning: Data Mining, Inference, and Prediction (2da ed.)*. Springer.

Hultkrantz, L., Nilsson, J. E., and Arvidsson, S. (2012). Voluntary internalization of speeding externalities with vehicle insurance. *Transportation Research Part A: Policy and Practice*, 46(6):926–937.

James, G., Witten, D., Hastie, T., and Tibshirani, R. (2017). *An Introduction to Statistical Learning with Applications in R*. Springer.

Ke, G., Meng, Q., Finley, T., Wang, T., Chen, W., Ma, W., Ye, Q., and Liu, T. Y. (2017). Lightgbm: A highly efficient gradient boosting decision tree. *Advances in Neural Information Processing Systems*, 30.

Pearson, K. (1896). *Mathematical Contributions to the Theory of Evolution. III. Regression, heredity, and panmixia*. Philosophical Transactions of the Royal Society of London. Series A, Containing Papers of a Mathematical or Physical Character.

Pesántez-Narvaez, J., Guillén, M., and Alcañiz, M. (2019). Predicting motor insurance claims using telematics data—xgboost versus logistic regression. *Risks*, 7(2):70.

Székely, G. J., Rizzo, M. L., and Bakirov, N. K. (2007). Measuring and testing dependence by correlation of distances. *The Annals of Statistics*, 35(6):2769–2794.